

Mobile Technology for Gamification of Natural Language Grammar Acquisition

Marina Purgina

A DISSERTATION

SUBMITTED IN FULFILLMENT OF THE REQUIREMENTS

FOR THE DEGREE OF DOCTOR OF PHILOSOPHY

IN COMPUTER SCIENCE AND ENGINEERING



Graduate Department of Computer and Information Systems
The University of Aizu

2019

Copyright by Marina Purgina

All Rights Reserved

The thesis titled

Mobile Technology for Gamification of Natural Language Grammar Acquisition

by

Marina Purgina

is reviewed and approved by:

Chief referee

Professor

Maxim Mozgovoy



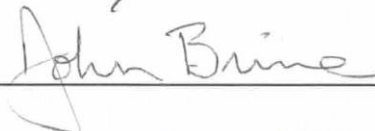
Professor

Vitaly Klyuev



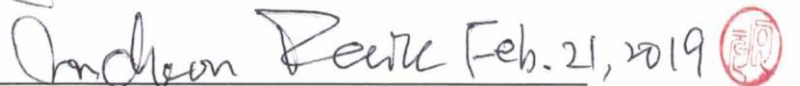
Professor

John Brine



Professor

Incheon Paik



The University of Aizu

2019

Mobile Technology for Gamification of Natural Language Grammar Acquisition

Marina Purgina

Submitted for the Degree of Doctor of Philosophy
2019

Abstract

Recent years are marked with rising interest to technologies of gamification, defined as the use of game design elements in non-gaming contexts. The basic premise of gamification is that the principles making computer games attractive can also increase attractiveness of other activities, such as learning. The interest to gamification technologies is triggered with widespread use of smartphones in general audience, and the growing popularity of casual mobile games, designed for wide range of people. Therefore, application developers can rely on unprecedented reach of their products and expect acceptance of game-like elements by the users. There is also an active discussion on what exactly constitutes “game-like elements”, suitable for the use in educational applications without harm for their primary educational objectives.

In the present work, we discuss a particular case of gamification of language learning via mobile system WordBricks, created at the University of Aizu. Most present systems of language learning are based on traditional learning activities, such as reading, listening, translating, and solving quizzes. WordBricks focuses specifically on the task of natural language grammar acquisition, and implements a concept of more user-centric lab-style experimental activities. The primary purpose of WordBricks is to give the users the capability to construct sentences according to predefined grammatical rules, and thus understand the basics of grammar system of

a particular natural language. The app is based on the concept of visual grammar formalism, aimed to encode the rules of grammar in intuitive and user friendly way.

WordBricks was evaluated in three different use scenarios: 1) as a learning aid at English language classes for computer science students at the University of Aizu, Japan; 2) as a teacher's demonstrational tool for the students of the same background; 3) as a supplementary learning material at Irish language classes for junior students of a public school in Dublin, Ireland. Our experiments demonstrate the feasibility of chosen approach on the basis of user feedback and numerical evidence showing that WordBricks can be as efficient as traditional learning materials, but providing more immersion and user enjoyment.

We also explore the possibility to automate the process of authoring WordBricks exercises with natural language processing modules. A significant part of this work includes manual annotation of grammatical attributes of words and word-word relationships, which can be also done with current language processing algorithms. The resulting markup can be corrected if necessary. In addition, automation of text processing allowed us to implement a procedure of converting arbitrary sentences into structured visualizations. This functionality helps students to understand the structure of sentences, not covered in WordBricks exercises.

As a result of experiments, we outline further directions for subsequent improvement of our technology. It includes introducing additional game-like elements, designing more learning materials, and making the application easily customizable by the educators. We also discuss principal difficulties faced by computer-assisted language learning technology experts due to inherent complexity of natural language, and challenging issues for our system.

Contents

List of Figures	xi
List of Tables.....	xii
1. Introduction	1
1.1 The Rise of Gamification	2
1.2 Technology, Classroom Practice, and Conscious Learners	4
2. Related Works	7
2.1 Duolingo and Anki as Different Cases of Gamification	7
2.2 Virtual Labs and Language Learning	10
2.3 Exploring Grammar with Interactive Exercises	11
2.4 Grammar Visualization Principles	14
3. WordBricks: General Approach	17
3.1 Bricks as Visual Elements	18
3.2 The Structure of Bricks.....	18
3.3 Chapters and Exercises	19
3.4 Sentence Visualization Mode	19
4. WordBricks: User Interface and Capabilities	21
4.1 General Design of the System	21
4.2 User Capabilities.....	22
5. WordBricks: System Architecture.....	25
5.1 System Description.....	25
5.1.1 WordBricks Package Structure	25
5.1.2 Implementation of core functionality	26
5.2 Brick Description Format.....	27
5.3 Brick Onscreen Configuration Format.....	28

5.4	Interactions Between WordBricks Components	29
5.5	Responsive Design of WordBricks GUI	32
5.6	Visualizing sentence structure	33
5.7	Interoperation with Language Processing Modules	34
6.	Classroom Experiments with WordBricks	41
6.1	WordBricks as a Learning Aid	42
6.2	WordBricks as a Demonstration Tool.....	46
6.3	WordBricks at an Irish Language Class.....	48
6.3.1	Evaluation in Irish Language Classroom First Study	50
6.3.2	Evaluation in Irish Language Classroom Second Study.....	52
6.4	User Suggestions	54
6.5	Open Challenges.....	56
7.	Discussion and Conclusion	59
7.1	Discussion	59
7.2	Conclusion	60
	Bibliography	63
	Appendix A. Sample Irish Exercise.....	69
	Appendix B. Sample XML Descriptions for the Current WordBricks Version ...	79

List of Figures

Figure 1. Language class in East Germany. <i>Source: German Federal Archive</i>	3
Figure 2. A fragment of Duolingo user interface	8
Figure 3. Reviewing session with Anki. <i>Source: Wikipedia</i>	9
Figure 4. ChemCollective Virtual Lab. <i>Source: chemcollective.org</i>	11
Figure 5. Combining blocks in Scratch	14
Figure 6. Shaped blocks in natural language learning materials	14
Figure 7. Visualizations obtained with AT&T GraphViz and ZPAR parser	16
Figure 8. Combining blocks with drag-and-drop interface	17
Figure 9. Structure of blocks in Word Bricks	19
Figure 10. Chapters; Exercises; Initial onscreen configuration	20
Figure 11. Combining blocks into sentences in WordBricks	22
Figure 12. Combining blocks with drag-and-drop interface	23
Figure 13. Block shapes reflect the natural flow of words in a sentence	29
Figure 14. Scheme of the structure and interaction of components of WordBricks....	31
Figure 15. Class diagram of bricks visualization subsystem.....	33
Figure 16. Architecture of the system	36
Figure 17. Processing pipeline steps for the sentence <i>The little cat devoured a mouse</i>	39
Figure 18. Processing pipeline steps for the sentence <i>Bhí na hataí agam</i>	40
Figure 19. Progress of individual students in WordBricks and control groups.	45
Figure 20. Model sentence #3 chunked into grammatical units	47
Figure 21. Irish “have” construction in WordBricks.....	49
Figure 22. An example of how WordBricks application works with the Irish language	54

List of Tables

Table 1. Descriptive Statistics for Two Groups in Diagnostic Test	43
Table 2. Results of the Quantitative Experiments.....	44
Table 3. Model Sentences and Target Structures	48
Table 4. Summary of student responses in the first pilot study.....	52
Table 5. Summary of student responses in the second pilot study	53

Chapter 1

1. Introduction

Gamification of language learning is a clear trend of recent years. Widespread use of smartphones and the rise of mobile gaming as a popular leisure activity among general audience contribute to popularity of gamification, as application developers can rely on unprecedented reach of their products and expect acceptance of game-like elements by the users. In practice, most mobile apps implement traditional language learning activities, such as reading, listening, translating, and solving quizzes. This work discusses gamification of learning natural language grammar with a mobile app WordBricks that is based on a concept of more user-centric lab-style experimental activities. WordBricks was evaluated in a number of diverse settings, and shows how the principles of gamification can be applied to this area of second language acquisition. We discuss general features that enable the users to engage in game-playing behavior, and analyze open challenges, relevant for a variety of language learning systems.

We consider development of this technology, and various scenarios for its use. Since the system was designed for students and children in the first place, we paid much attention to the interface and responsive design of the system. The system should be intuitive and easy to use. It should be consistent with pedagogical goals. It should be extensible and adaptable in order to incorporate new assignments and use cases. Due to the growing popularity of mobile platforms and mobile-assisted language learning, the system should be available on a mobile platform. Also the system should be flexible enough to support a large variety of natural languages.

We also use WordBricks application as a visualization module that can display a user-supplied sentence as a parse diagram. This sentence can be a part

of an exercise or freely added by the user. In the latter case, we use natural language processing modules to markup the sentence for further conversion to a set of visual objects. In this regard, specific attention is devoted to developing a format for presenting syntactic elements and parse diagrams.

1.1 The Rise of Gamification

The practice of second language acquisition has been relying on modern concepts and technologies of its day for decades. In particular, language labs equipped with audio recording and playback facilities were in active use in 1970s-1980s (see Fig. 1) [1]. Thus, the general idea of using modern technologies for language learning evokes little debate. The main discussions are related to the development of efficient ways of using the available instruments, and addressing their limitations. Technologically, language labs of the past century often suffered from unreliable tape-based systems and insufficient means of teacher control [1]. Methodologically, the prevalent audio-lingual method of teaching was considered inefficient and fell out of favor in 1970s [2]. However, the technical issues were eventually resolved with the rise of computing machinery, and audio-lingualism was driven out by newer, presumably more efficient methods. In general, we believe that modern practice of computer-assisted language learning (CALL) is evolving within according to the same principles. Language teachers are seeking to use the present technology (with its capabilities and limitations) in the most effective ways, while computer science specialists are trying to advance the technologies, providing more options for their practical use.

One of the most salient trends in modern computer-assisted education research is the rising interest to *gamification* of learning. While the idea to introduce certain game-like elements into learning is definitely not new, the word “gamification” came into wide use only in 2010s [3], together with the surge of related research efforts. It is very likely that this process is connected to the rising popularity of smartphones and mobile games that turned a large number of phone owners into casual gamers. According to AdMob statistics, 59% of smartphone users install games within a week of getting their devices [4], so “*it is difficult to find a person now who hasn’t played at least one video game, making games more of an accepted and integrated part of our society*” [5]. In other

words, certain exposure to computer or mobile game experiences can be expected now from a typical learner, so the developers of educational software can assume that the users perceive game-like elements as something familiar.



Figure 1. Language class in East Germany (1975). Source: *German Federal Archive*

Following the work of Deterding et al. [6], most authors draw a clear distinction between “gamification” (defined as “*the use of game design elements in non-game contexts*”) and related concepts of “(serious) games”, “toys” and “playful design”. While games and toys can be definitely used in educational context, the less restrictive concept of using “game design elements” can be arguably applied to a wider range of scenarios. Pure educational games often hide dull and repetitive tasks behind colorful graphics and animation, thus perceived as “chocolate-covered broccoli” by the users [7]. Recent research efforts identified more subtle “fun factors”, such as *concentration*, *challenge*, or *immersion* that contribute to the enjoyability of the game experience [8]. However, it is difficult to design a game that would combine engaging mechanics with high educational value. One relevant example is DragonBox Algebra game that lets the users practice solving linear equations. Experiments show that its visual formalism is hard to connect with the standard mathematical notation, so the students using DragonBox Algebra do not improve their math tests scores [9, 10].

Still, the motivation to combine game-like experiences with education is strongly supported with a simple observation: “*since video games... can demonstrably motivate users to engage with them with unparalleled intensity*”

and duration, game elements should be able to make other, non-game products and services more enjoyable and engaging as well” [6]. Thus, gamification suggests a somewhat lightweight alternative to engaging in full-fledged educational game projects, evoking another question: what are these “game elements”, able to motivate the users? Some authors express very bitter views on gamification, arguing that in practice it became a collective term for a number of exploitative techniques for increasing user spending, and having no relation to core game process [11, 12].

A work by Morford et al. [3] introduces six basic traits of game-playing behavior:

- 1) direct impact on the game outcome and results;
- 2) clear goals and/or end conditions;
- 3) the presence of rules and barriers;
- 4) probabilistic outcome;
- 5) development of strategies and heuristics;
- 6) non-coerced initiation.

Strikingly, this list does not include elements or activities explicitly labeled as “fun”. One possible interpretation of the results obtained by Morford et al. is to conclude that many processes, possessing the stated traits (1-6) are *perceived as fun* or, at least, *engaging* by the users, and thus can be considered “gamified” even if they lack some other elements, typical for computer games (such as graphics, competitive gameplay, arcade controls or engaging narrative).

1.2 Technology, Classroom Practice, and Conscious Learners

Wide adoption of technologies in education is primarily driven with advancements that make their use attractive for teachers and learners. One example of a technological success story is ubiquitous adoption of flight simulators in pilot training programs [13]. Computer-assisted language learning is also widely used in practice, if we follow its common definition as “the use of computers in language learning activities” [14]. However, a closer look reveals

that computer technologies are most commonly used in traditional learning activities, such as face-to-face chats, watching video clips and listening to audio, participating in group discussions, and reading texts. Consequently, the main emphasis is usually made on general-purpose computer instruments, such as Skype, YouTube, Facebook, and various online forums and resources [15]. On the other hand, dedicated CALL systems and specialized technologies are rarely mentioned.

The situation with specialized CALL instruments is somewhat complicated. To begin with, there is no general agreement about their effectiveness among experts, as shown in Hubbard's survey, conducted in 2002. Hubbard notes: "*...it is interesting that questions of effectiveness still tend to dominate. In fact, the basic questions of "Is CALL effective?" and "Is it more effective than alternatives?" remain popular even among those who have been centrally involved in the field for an extended period of time.*" [16]. We believe that, at least, in part this situation can be explained with relative immaturity of language processing technologies that could potentially be of great benefit for the learners. Among these technologies are machine translation, automated speech recognition, grammar checking, and feedback generation, to name a few. Automated speech analysis is used to certain extent (e.g., in Rosetta Stone software), but its quality is often criticized [17, 18]. In addition, CALL applications are seemingly of limited interest for natural language processing community. As noted in [19], "*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*". In addition, it is probably not easy to find the best use cases even for the existing language technologies in a way that would provide benefits for the learners despite technological limitations. Therefore, the progress in this field is hindered by the necessity of coordinated efforts between the teachers and technology experts, who have different agendas and constraints.

In this situation, gamification is an interesting direction of research, since it often deals with active *conscious learners* rather than participants of predominantly teacher-guided courses. Let us recall that one of the

characteristics of game-playing behavior is *non-coerced initiation*, meaning that “a player plays the game because he wants to, not because he has to” [3]. Therefore, while gamification is possible both inside and outside the classroom, we believe that the best results can be achieved in voluntary user-initiated learning sessions, more closely resembling typical game-playing scenarios. Even if a certain application supports only basic traditional learning activities (such as reading and listening), it can reinforce user motivation and make the process of learning a language less burdensome (or more enjoyable, depending on one’s perspective). Regardless of a student’s attitude towards language learning, we must concede that this process involves numerous repetitive tasks and memory drills. For example, to master Japanese, one has to learn around 2000-3000 Chinese characters [20]. It is difficult to imagine learning strategies that would make this activity inherently fun and enjoyable. Indeed, most learners in practice rely on different variations of drills [21] (mnemonics and other techniques still cannot liberate students from drilling sessions), so any technological tricks that make this undertaking less daunting should be appreciated. At the same time, it is not easy to estimate how many users would be willing to participate in such non-coercive game-like educational activities, and thus benefit most from gamification. However, various studies conclude that at least people engaged in daily learning activities (such as university students) are willing to use their mobile phones for out-of-classroom learning as well [22, 23]. Thus, the ubiquity of mobile devices, wide spread of mobile gaming, and users’ willingness to use mobile devices as learning tools constitutes a perfect combination for the success of gamification techniques at the present time.

Chapter 2

2. Related Works

2.1 Duolingo and Anki as Different Cases of Gamification

To illustrate some of the principles outlined in the previous section, let us quickly consider two successful language-learning applications, Duolingo and Anki, and discuss how these completely different kinds of tools agree with general principles of gamification.

Duolingo is often considered as one of the most successful language learning apps on the market with around 200 mln subscribers worldwide [24]. It received numerous positive responses by the users [25, 26], and its efficiency in keeping user attention and increasing language proficiency is reported in research literature [27, 28]. The developers of Duolingo attribute their success directly to gamification. In particular, they mention four basic pillars of their approach [24]:

- 1) dissection of the large goal (learning a language) into a set of small daily user-chosen goals;
- 2) visual clues to track user progress;
- 3) emails and notifications for motivating inactive users to return to their studies;
- 4) rewards and achievement badges for continuous daily use (known as “the streak”).

Technically, Duolingo implements a number of traditional exercises, such as “translate a sentence”, “match words with their translations”, “type the pronounced phrase” or “pronounce the given phrase” [29]. It is important to note that the developers do not try to cover ordinary tasks with a “chocolate layer” of a game. The exercises are presented in the same way as in conventional textbooks

(see Fig. 2). The application is clearly aimed at conscious learners who fully acknowledge that they are involved in a laborious and not always fun activity of learning a language rather than playing a game. Thus, Duolingo relies on more subtle principles of gamification, aimed to introduce game-playing behavior into learning. Indeed, typical learning sessions in Duolingo possess most of the traits of game-playing behavior listed in [3].

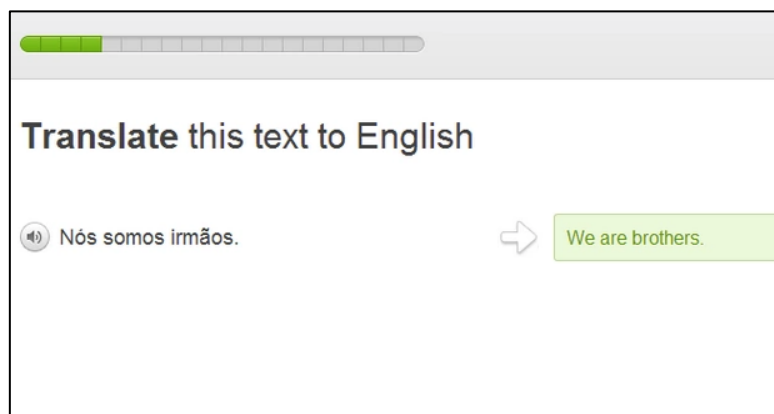


Figure 2. A fragment of Duolingo user interface

Duolingo, however, is a showcase of success story that is hard to reproduce. The app implements vast functionality, so it is difficult to recommend following the same approach in smaller-scale projects. Therefore, it is interesting to consider the case of much smaller (in terms of functionality) project Anki that aims to create an intelligent flashcard organizer for desktop and mobile platforms.

At a glance, Anki is a plain and simple-looking application that implements only the required functions, necessary for its purpose, and does not adhere to any gamification principles. However, Anki is well known among language learners. The Android version of the app is installed on over one million devices, and is rated by over than 32,000 users (as of April, 2018). It is also a subject of several research articles [30, 31], and often praised and recommended by the users [32, 33]. Anki implements a space repetition procedure [34] that constantly rearranges flashcards in such a way that new and poorly memorized cards are shown more frequently. This way, there is no need to review the whole deck of cards during each learning session: the system selects the cards for the next

review automatically. In a sample session shown in Fig. 3, the user has to recall the correct translation of the word “paccka3”, and after revealing the answer (“story, tale”) press the corresponding button. If the card is forgotten, the user should press “Again” and try this card again in a few minutes. Similarly, the button “Good” will schedule this card for review in 3 days.



Figure 3. Reviewing session with Anki. Source: Wikipedia

While gamification was apparently not in the agenda of Anki developers, we should note some casual similarities between certain Anki features and deliberate game-like elements of Duolingo:

- 1) the large goal of memorizing the whole deck of cards is split into daily reviewing sessions;
- 2) users can track their progress by checking statistical data in a special window;
- 3) users are strongly encouraged to keep their “streak” and adhere to daily reviewing sessions to avoid a flood of unreviewed cards.

Since the users can freely create and share cards, there is even a certain social element in this activity. We also believe that reviewing sessions in Anki can be considered game-playing behavior according to [3].

The case of Anki shows that gamification does not necessarily have to be a well-thought strategy. Game-likeness can be an inherent property of a certain

study process, so the software developers just have to recognize game-like elements and support them properly in their product.

2.2 Virtual Labs and Language Learning

Duolingo, Anki and numerous other related apps provide great examples of modern technology-driven way of supporting learning activities and learner motivation. However, in terms of content they represent traditional learning materials and exercises. Their true power comes mostly from mobility, multimedia capabilities, and game-like features, while the use of dedicated language processing technologies is still very limited (which probably can be explained with their relative immaturity, as we discussed above). In practice it means that some important elements of language learning remain outside the scope of most computer-assisted language learning tools. In particular, virtually no instrument can evaluate and correct user-constructed phrases, though such corrective feedback is usually considered an integral part of learning by many researchers [35] (some others, such as Truscott [36], argue against it). In any case, feedback of some kind exist in most learning activities.

The effect of technological maturity can be seen in some educational tools, available for natural sciences, such as physics or chemistry. The foundations of these sciences are more precisely defined in mathematical terms, which opens new possibilities for educational software developers. For instance, Open Source Physics [37] and ChemCollective [38] projects collect a vast amount of interactive simulations in physics and chemistry (see Fig. 4). These instruments can be treated as “virtual labs” that enable the students recreate textbook experiments on their computers and even run their own experimental setups and analyze the outcomes. The equivalent of ChemCollective in language learning would be a virtual character (chatbot), able to discuss a range of predefined topics or engage in a free dialog with the user, and provide different kinds of feedback.

While the currently available technology cannot support such functionality, we argue that certain elements of user feedback can be automated. One example is automated speech analysis and recognition, mentioned previously. Another possible direction is the analysis of the structure of user-supplied text,

implemented, e.g., in a Japanese language tutoring system Robo-Sensei [39]. In the subsequent sections, we will introduce our own system WordBricks that tries to gamify the process of grammar acquisition, using virtual lab approach, found in the systems like Open Source Physics and ChemCollective.

2.3 Exploring Grammar with Interactive Exercises

Natural language grammar is an essential topic of most common language courses. It is often integrated into general textbooks or covered in dedicated literature, such as well-known English grammar reference books by Betty Azar [40] or Raymond Murphy [41]. While some educators, such as Stephen Krashen, argue against explicit teaching of grammar [42], we are not taking sides in this debate and merely state that grammar as a subject is widely taught, and thus grammar acquisition can be considered a legitimate target for a CALL system.

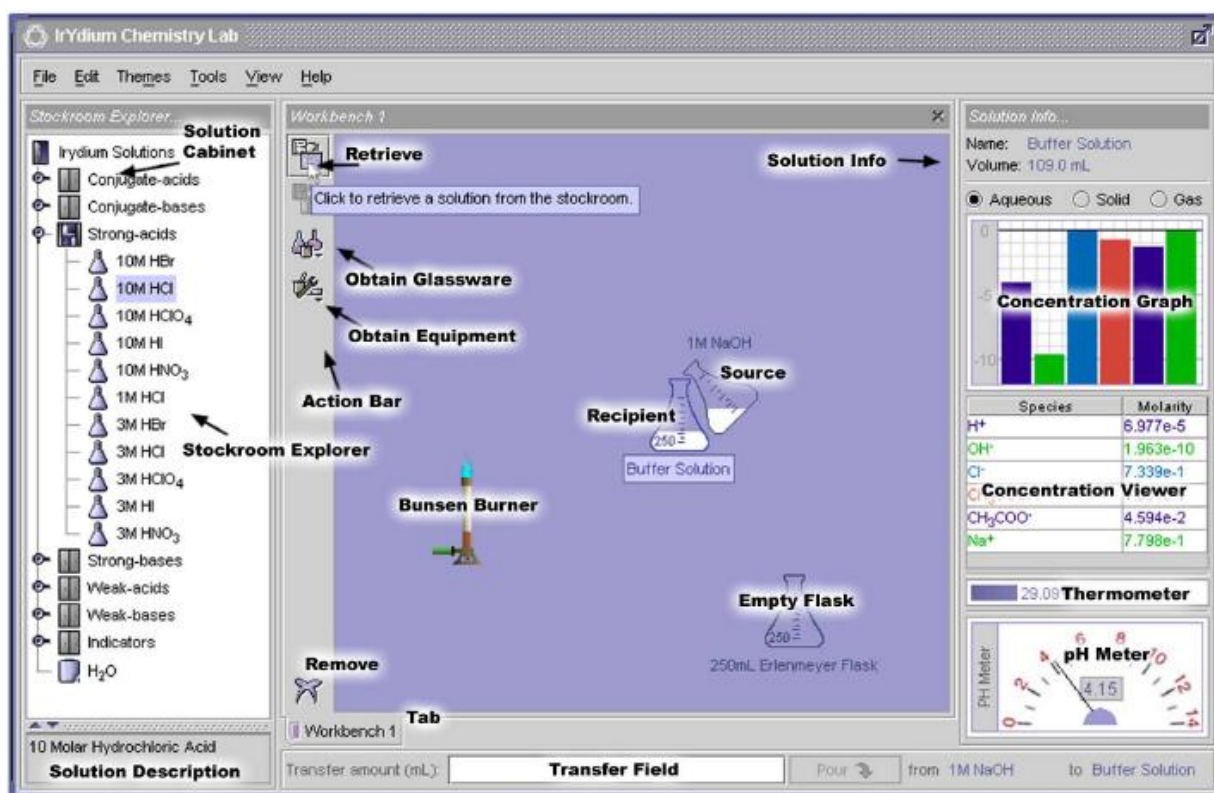


Figure 4. ChemCollective Virtual Lab. Source: chemcollective.org

Today's actual grammar teaching practice is primarily focused on traditional exercises aimed at acquisition of proper grammatical forms and rules. Numerous studies indicate that most research on "innovative" grammar teaching methods

have little impact on textbook content and classroom activities [83]. Jean and Simard [84] note that “grammar instruction is perceived by both students and teachers as necessary and effective”, and thus most educators are reluctant to abandon it, especially in the absence of universal agreement on possible alternatives.

There is, however, an ongoing discussion of particular ways to implement grammar instruction in practice. For example, common advice is to focus on student communication, and to draw attention to grammar forms arising naturally in the process rather than following a predefined list of grammatical structures [85]. Still, this approach can be implemented differently by different teachers, and there are no universally preferred ways to explain grammatical phenomena. For example, Larsen-Freeman [82] suggests to focus on reasons rather than rules (e.g., while considering a sentence “There is a snowstorm coming”, the teacher should explain that there introduces new information, and new information is marked with indefinite determiners such as *a*, rather than quote the corresponding formal grammar rule).

Certain attention is paid to the problem of balance between input processing and production activities [86] and to the creation of “focused tasks” designed to practice specific grammatical structures [87]. In general, most conventional activities are not marked as “inherently (in)efficient” in research literature. Effectiveness depends primarily on their appropriate implementation.

Judging from typical grammar book contents, most common types of exercises require the language learner to form grammatically correct sentences. These exercises come in numerous variations, such as:

- 1) jumbled sentence: put the words in the correct order (possibly, with altering their forms);
- 2) fill the gap: fill the gap in a phrase using the appropriate word from the given list;
- 3) find errors: decide which phrases from the given list are grammatically correct;

- 4) rephrase: rewrite the given phrases using the specified grammatical construction.

The exercises are usually designed to have a single correct answer, provided in the “Answers” section.

We decided to elaborate this scheme by providing the user more interactivity and more visual clues, fostering better understanding of grammatical constructions. We believe that the lack of interactivity is one of the most salient shortcomings of traditional grammar book exercises. A learner can confirm own understanding of how to use certain words in certain combinations using the rules described in the given book section, but has no way to experiment with these words and rules. For instance, the learner might want to try substituting one word with another, using a word in another context, or combining two rules to formulate a more complex sentence. Our roadmap included the following scenarios (partially implemented at the present time):

- 1) The user sees on the screen a number of movable words, related to an individual exercise in a particular grammar book section. The task is to combine the words into a single sentence (so, this is a variation of a “jumbled sentence” exercise type). The user is also able to substitute certain words with their word forms.
- 2) In addition to the first scenario, the user is able to add new words related to the same grammar book section, and freely experiment with them (i.e., change their word forms and connect them into sentences).
- 3) The user can select any words from the available word bank and freely combine them.
- 4) The user can add new words to the word bank and analyze the structure of arbitrary sentences.

The viability of this plan (both in terms of technical feasibility and in terms of pedagogical value) strongly depends on a particular approach to visualization. In our case, graphics reflect a certain “visual grammar language” that directly influences learner perceptions and system capabilities.

2.4 Grammar Visualization Principles

Our approach to designing such a visual language is influenced by Scratch [43], which is a system for learning the basics of programming. Programming languages have a grammar (albeit much simpler than human languages do), so it is essential to understand how individual instructions can be combined into complex structures. Scratch expresses grammar rules by representing instructions as blocks of different shapes, so that only matching blocks can be connected into a single structure (see Fig. 5). Scratch's graphical editor is not just a simpler way to write computer programs, helpful for the beginners. We believe that it can be treated as a construal [44] that forms a model of a programming language in the learner's mind. This way, the learner understands both the rules of grammar and the reasons why they work in a certain way (because one cannot fit a rectangular block into a round hole). A similar idea is used to some extent in natural language learning as well [45] (see Fig. 6).

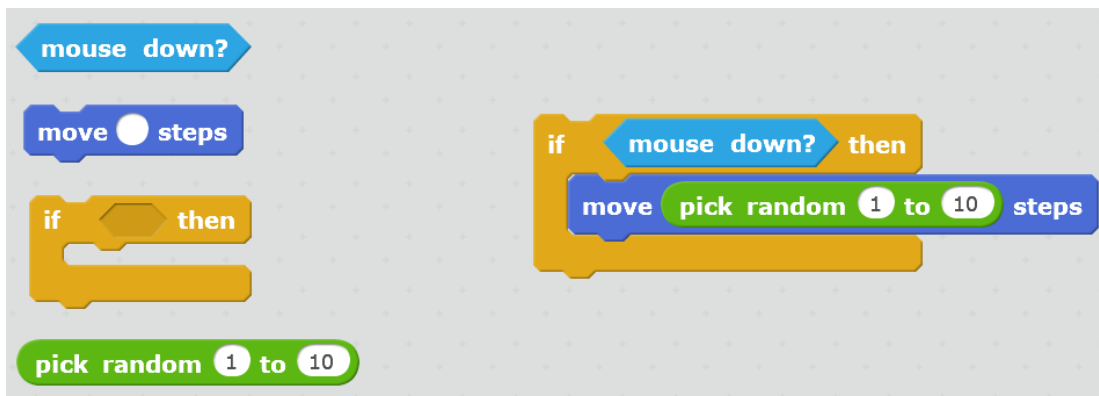


Figure 5. Combining blocks in Scratch

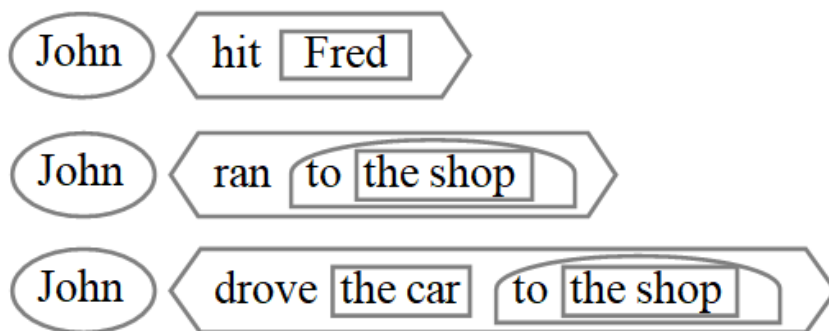


Figure 6. Shaped blocks in natural language learning materials. Source: Ebbels

Obviously, it is much harder to design a consistent set of blocks for a natural language than for a simplified programming environment. Words in natural language have many grammatical attributes (such as part of speech, gender, person and number), and the rules of grammar are often complex and contain numerous exceptions. Therefore, we do not strive for a perfect system (in fact, even Scratch blocks do not always adhere to the principle of shape matching), but aim to illustrate at least the basic phenomena of natural language grammar.

It is probably even more difficult to decide the logic of block arrangement inside an individual sentence. As shown in the Fig. 5, a Scratch program resembles a two-dimensional jigsaw puzzle. Certain blocks in Scratch, such as “if...then” construction, have several “connectors”. Other blocks can be connected to the top or the bottom of an “if” block, placed inside it, or between the words “if” and “then”. Simpler blocks, such as “mouse down?”, can only be placed into the connectors inside other blocks. Therefore, it is necessary to decide what kind of connectors individual blocks should have, and how to arrange them on the screen so that they adequately represent syntactic structure of natural language sentences.

Existing linguistic theories approach the problem of sentence structure from different perspectives. We base our project on dependency grammar theory [46] that suggest connecting individual words in a sentence with direct links, reflecting “head/dependent” relationships. Dependency grammar formalism is widely used in natural language processing, and practical principles of dependency-based sentence markup are well documented [47]. Our main motivation for relying on dependency grammar formalism was its resemblance to the structures of Scratch and to the Shape Coding system introduced by Ebbels [45]. Furthermore, dependency relations require no additional visual blocks (all blocks represent sentence words), which reduces the number of onscreen objects.

As a result of converting a sentence into a set of head/dependent pairs, we obtain a tree-like structure that has to be visualized. Unfortunately, common types of visualizations can be difficult to understand for non-specialists (see

Fig. 7). Therefore, we had to design our own scheme, somewhat similar to the Ebbels's Shaping Coding system.

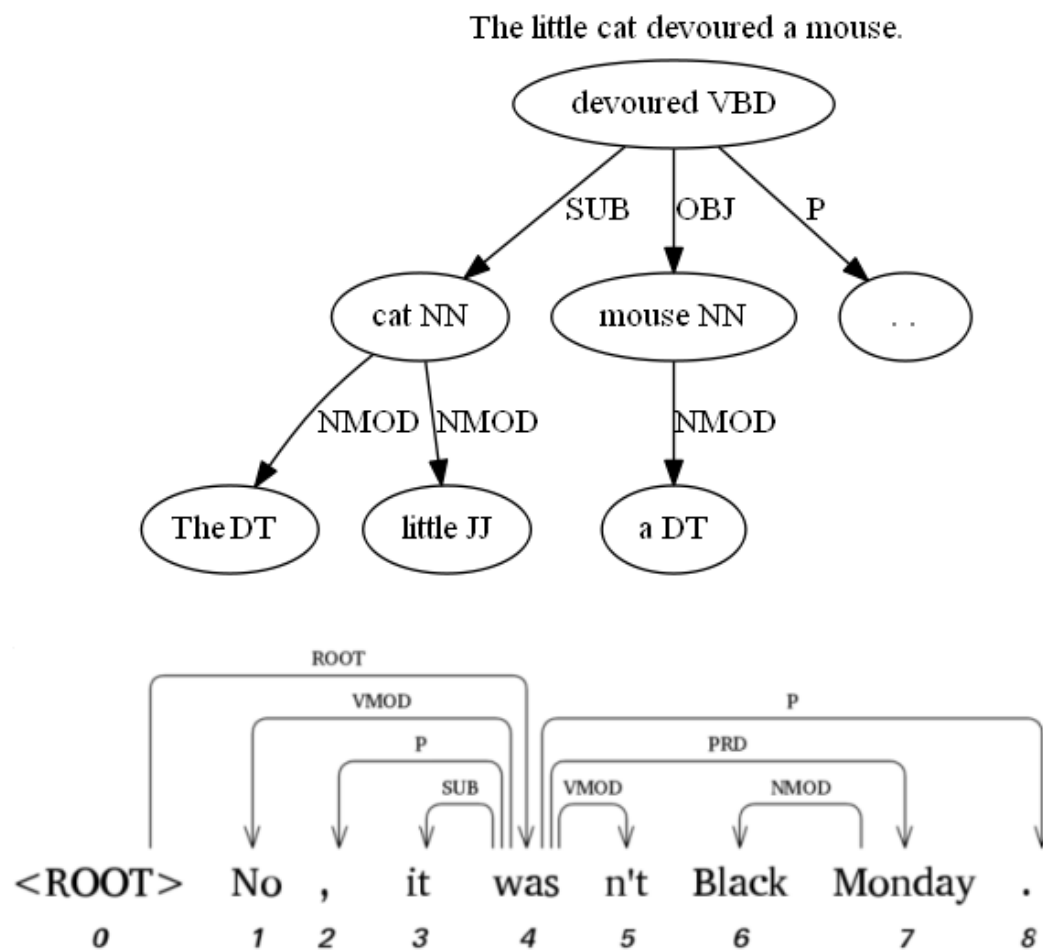


Figure 7. Visualizations obtained with AT&T GraphViz (above) and ZPAR parser (below)

Chapter 3

3. WordBricks: General Approach

Initially, WordBricks was conceived as a desktop application [48]. It followed the traditional methods of visualizing word links and implemented a rich system of graphical elements represented various grammatical phenomena. Recently we redesigned WordBricks as a mobile app based on simpler and cleaner principles of visualization that greatly improved user experience (see Fig. 8).

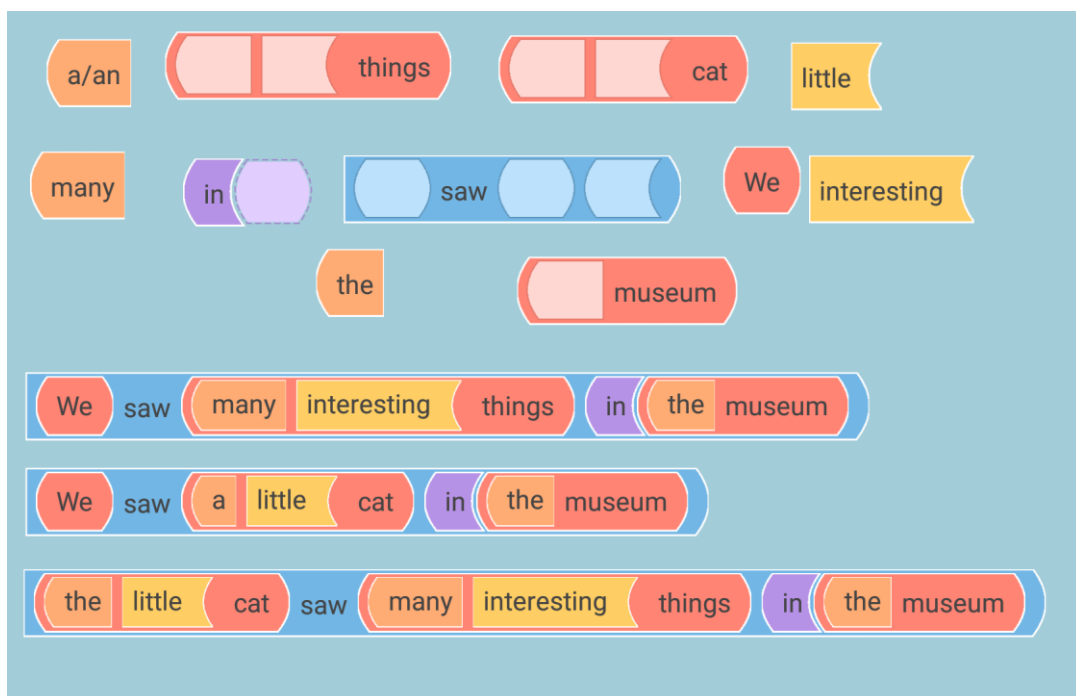


Figure 8. Combining blocks with drag-and-drop interface

In WordBricks, all syntactic elements of a sentence, such as words or the whole phrases are represented with blocks. The shapes and colors of such blocks depend on a set of their language-dependent grammatical attributes, such as part of speech, person, gender, and so on. Some blocks also have one or more same-colored connectors. Each connector is shaped according to the set of grammatical attributes associated with it. Connectors are “placeholders” for dependent

syntactic elements, such as words or phrases. For example, most verbs have a connector for a subject in the left-hand side of the block, and for an object in the right-hand side of the block. If a shape of a connector matches a shape of a certain block, and the set of grammatical attributes of a block forms a subset of grammatical attributes of a connector, the user can insert the block into the connector.

3.1. Bricks as Visual Elements

Basic onscreen elements in WordBricks are quadrilateral-shaped elements called *word bricks*. Each brick is characterized with its shape, color, word(s), and a set of quadrilateral-shaped connectors that can be located on either side of the word. Bricks also have a number of associated non-graphical attributes, invisible to the user. The most important of them is a textual *lemma*, used to combine bricks into groups. Any bricks sharing a common lemma belong to the same group. Typically, a lemma is simply a base form of the brick's word. Let us use the term *brick signature* to refer to the set of user-perceived graphical features of a brick (shape, color, words and connectors).

3.2 The Structure of Bricks

Each brick is associated with a set of textual *attributes* and characterized with a (background) *color* and a *shape*. The attributes unambiguously define the shape and the color of a brick. Attributes can have names, but this is optional. Each brick contains a linear list of fixed (pre-determined) *words* and *connectors*. Each word is a static pre-defined textual element, drawn on a brick background. Like bricks, each connector is associated with the same set of textual attributes that define its shape. All connectors are drawn using the same background color. Each connector also has an optional textual *label* (see Fig. 9). A connector can be addressed either by its 1-based cardinal number in the list of connectors or by its label (which is not guaranteed to be unique, though). A brick can be addressed by its unique nonempty textual *ID*. Furthermore, a brick is associated with a textual *lemma* that serves as an ID of the group the brick belongs to.

A brick can be placed inside another brick's connector if that connector's attributes form a subset of the set of brick attributes. For example, one may place the brick with the attributes [noun, common case, plural, third person] inside a connector with the attributes [noun, common case].

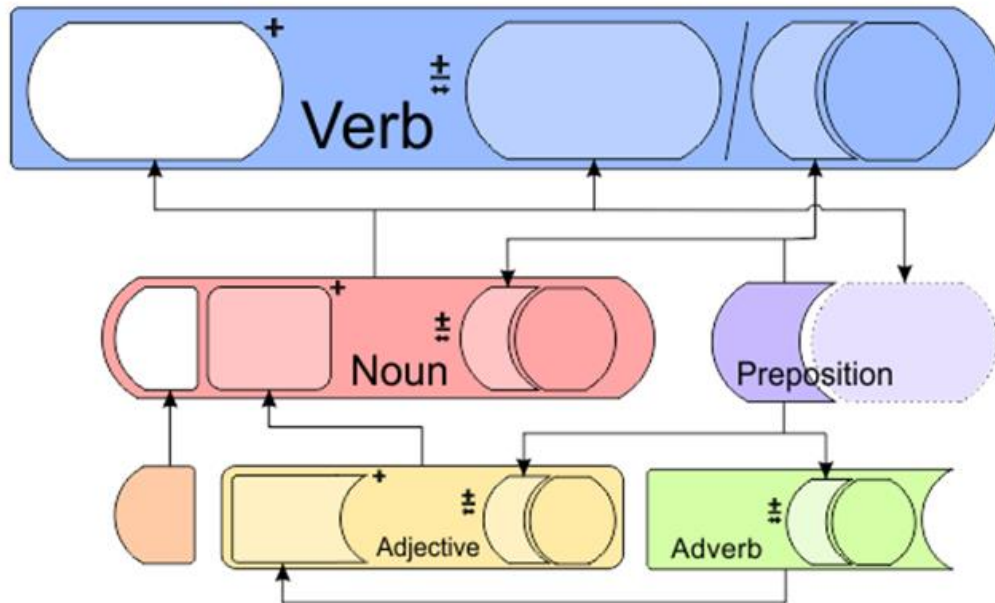


Figure 9. Structure of blocks in Word Bricks

3.3 Chapters and Exercises

Initially, the user sees a title screen with a list of elements, called *chapters*. By tapping on a chapter, the user opens a list of new elements, called *exercises*. In the current design, the first exercise in the list is preloaded, but the user can choose any other exercises from the side menu to load (see Fig. 10). Each exercise, in its turn, is a *brick configuration* that lists all the bricks available for the user within the given exercise, and the initial onscreen content. The user can switch back and forth between individual exercises without losing onscreen content of the previous exercise.

3.4 Sentence Visualization Mode

In the sentence visualization mode, WordBricks lets the user to experiment with any words and word combinations to check which constructions are admissible according to natural language grammar. On startup, the application

displays an input box prompting the user to provide any arbitrary sentence. After the user taps the OK button, the application sends a request to the server side. The server returns an XML document describing the desired brick configuration to be displayed on the screen. This way, WordBricks is used as a visualization module, and its primary purpose is to display a static structure of already linked bricks, corresponding to a user-supplied sentence, while all linguistic markup and word-word linkage is performed on the server side.



Figure 10. (left) Chapters; (center) Exercises; (right) Initial onscreen configuration

Chapter 4

4. WordBricks: User Interface and Capabilities

4.1 General Design of the System

In the current version of the system, the user has to select a particular exercise in the main menu, and the corresponding predefined blocks will appear on the screen. In addition, some optional blocks will be made available via the “word bank” menu of the application.

This way, the user sees on the screen the blocks of different colors and shapes, representing words and phrases, and can connect them to get a completed sentence. In many cases, the user only needs to make sure that the shapes of the block and the connector match, to join them together. If the shapes do match, but the attributes do not, the system will display a hint, explaining which mismatching attributes prevent the elements to be connected. In most our experiments, the shape of a block is defined by its part of speech, but this configuration is flexible. Unfortunately, in practice it is difficult to show all grammatical attributes visually on the block, so we have to rely on the system of hints to provide additional error feedback to the user.

This method of displaying word links can be seen as a way to visualize dependency relationships, similar to the ones shown in Fig. 7. Our approach enforces a certain word order in accordance to the order of connectors and let us display the resulting sentence in a natural linear way (see Fig. 11). However, it cannot handle *non-projective dependencies* that rarely appear in English, but may constitute up to 25-27% of constructions in some languages such as Czech and German [49].

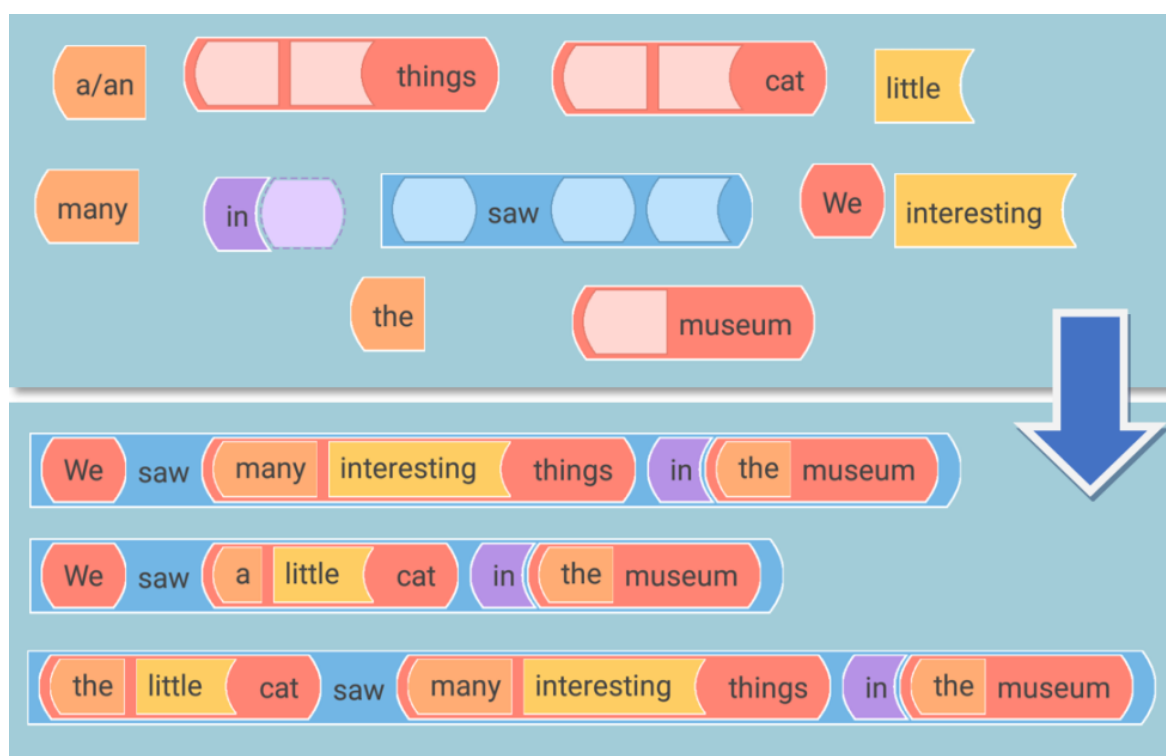


Figure 11. Combining blocks into sentences in WordBricks

Since WordBricks is a mobile application, it follows conventional touchscreen interface conventions. The user can move blocks and fragments of sentences in any direction on the screen using drag-and-drop (see Fig. 12). The whole screen area except the menu bar at the top and the status bar at the bottom is used for brick arrangement. Double tap on a block opens its settings. Currently, the main functionality of the settings dialog is the selection of the desired word form. For example, if an exercise contains the word “cat”, the only way to obtain “cats” on the screen is via this dialog.

4.2 User Capabilities

The user can transform or remove any onscreen brick using a popup menu, activated by double-tapping the brick. In the current version this menu can be only activated for a free brick (that is not linked to another brick and does not link any other bricks). This menu should have two elements: a *delete* button that removes the brick, and a list of bricks that can be chosen to substitute the given brick.

The list can be obtained as follows. First, we select all the bricks with the same lemma as the current brick. Next, we filter the list to keep there the bricks with different signatures only. It makes sense to show actual graphical bricks in this menu instead of textual labels.

“Add brick” window is activated with a *plus* button located on the screen. In the simplest form, it should contain a list of all bricks available in the given exercise. In the same way as in the “Substitute brick” menu, it should contain a filtered list of words, displayed as graphical bricks. The list should be grouped according to lemmas, and alphabetically-sorted (first sort the groups, then sort the bricks inside each group).

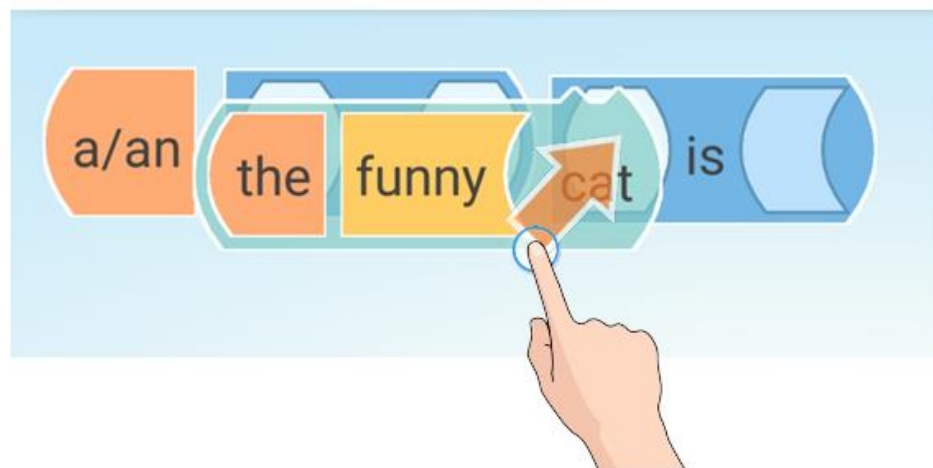


Figure 12. Combining blocks with drag-and-drop interface

Chapter 5

5. WordBricks: System Architecture

The current version of WordBricks is available for Android platform. It relies on standard functionality of the Android framework, and uses the capabilities of Android SDK classes to implement application logic [50]. Previously, each block was represented with a widget based on a customized `View` class of the standard Android library [51]. However, nested blocks caused unacceptable performance drops, so we had to rewrite the entire block rendering functionality ourselves. The application draws all the blocks on the main view. Shapes, colors and the content of the blocks are rendered according to their XML definitions. Let us consider the system in more detail.

5.1 System Description

We designed the architecture of the mobile application by following the process suggested in [75]. Android was the operating system of choice for WordBricks due to its wide availability and openness [76].

5.1.1 WordBricks Package Structure

WordBricks is developed using the Java programming language. It relies on the standard functionality of the Android framework. Thereby, the application consists of the following components:

Java classes that are subclasses of the main Android SDK classes (`View`, `Activity`) and Java classes that have no Android SDK ancestors, i.e. helper classes for implementation of the application logic;

- the Android Manifest file;

- application resources and XML definitions of application GUI layouts;
- exercise descriptions (XML files).

In terms of handling relationships between GUI and a logic supporting GUI, the application architecture follows Model-View-ViewModel architectural pattern [64].

5.1.2 Implementation of core functionality

In addition to “virtual lab” experience, WorkBricks is intended to contribute to the overall gamification of the study process. Therefore, we tried to visualize the grammar through plain and simple forms as much as possible. From a technical point of view the above problem can be logically divided into two sub-tasks: visualization of syntactic forms and semantic description of grammar exercises.

1) *Visualizing syntactic forms.* Android application GUI is a tree of instances of View subclasses, i.e. GUI widgets [65]. The View class is from the Android framework and it is used for all Android GUI widgets. In the system, every element of a sentence is represented with one brick of a certain shape and color having a set of connectors for other sentence elements. These elements may vary from one exercise to another, and may consist of individual words or punctuation symbols, or arbitrary phrase fragments. According to the Android framework, Brick objects displayed on WordBricks screen are defined as subclasses of the View class. The GUI tree is normally defined with XML layout files, and at the runtime expanded automatically into the tree of corresponding objects. However, in our case our custom View of ViewBrick is created and added to the existing GUI layer at runtime. This allows the user to create and delete them at any time.

Brick width is calculated in accordance with a set of parameters. These parameters are word length, empty connectors and non-empty connectors if they are presented. Therefore, brick width is constantly changing in the process of sentences construction, as illustrated in Fig. 8.

2) *Describing grammar exercises semantics.* As mentioned above, the content of each individual brick varies from exercise to exercise. Therefore, each exercise needs its own XML-defined set of word bricks. In other words, each exercise is specified by an XML file with an independent 'vocabulary' of bricks. In fact, this specification of the bricks is a description of the semantics of words and semantic description of its use. The polysemy nature of words can be defined with multiple XML sections.

5.2 Brick Description Format

The brick description document defines the content (words, attributes, connectors) of all the bricks available in the given exercise. The format is designed with simplification in mind. The basic assumption is that any word form and any alternative set of attributes of the given word corresponds to a separate brick. E.g., *table* and *tables* are separate bricks; *am* as in “I am a student” and *am* as in “I am funny” are separate bricks, too (see Listing 1).

The following design decisions should be noted. Typically, a lemma is used as is, but in some cases the author can choose an internal lemma to be different from a “screen name” shown to the user. Therefore, we need a separate optional *screenlemma* attribute. Some connectors can be made optional by adding the *optional* attribute set to *true*. Currently we assume that each connector can be present only once in a brick (so it is not possible to have two subjects or two objects of the same verb). The lists of several objects or when/where/etc. should be collapsed into one compound clause with special combinational bricks, but we may revise this logic in the future.

Listing 1. XML description of the word *devoured*

```
<brick id="devoured_1" lemma="devoured" type="Verb phrase">
  <item type="brickConnector" connector="1" value="Noun phrase">
    <attrs
      case="common" person = "third" number="singular"/>
    <attrs
      case="nominative" person = "third" number="singular"/>
  </item>
  <item type="word">devoured</item>
  <item type="brickConnector" connector="2" value="Noun phrase">
    <attrs
      case="common" person = "third" number="singular"/>
    <attrs
```

```

        case="oblique" person = "third" number="singular"/>
    <attrs
        case="common" person = "second" number="singular"/>
    <attrs
        case="oblique" person = "second" number="singular"/>
    <attrs
        case="common" person = "third" number="plural"/>
    <attrs
        case="oblique" person = "third" number="plural"/>
    <attrs
        case="common" person = "second" number="plural"/>
    <attrs
        case="oblique" person = "second" number="plural"/>
</item>
</brick>

```

5.3 Brick Onscreen Configuration Format

Some bricks present in the given exercise should be preloaded and shown on the screen before the user starts interaction, so the author should have a method to specify a certain initial onscreen brick structure. In the future, the same format can be used to save user progress.

Bricks on the screen form a forest, so in order to save bricks onscreen locations, we will need to save each root brick's (x, y) coordinates as fractions of the overall screen size (to accommodate the screens of any sizes and aspect ratios). Coordinates should not be specified for non-root bricks since their positions are defined by the positions of their parents. Probably, it makes sense to allow omitting coordinates for root bricks, too, which will force the system to use some automated placement algorithm.

The xml file simply lists all onscreen bricks. For each brick we specify its parent (no parent in case of a root brick) and the connector it is attached to. Connectors are addressed with their cardinal numbers or textual labels (see Listing 2).

Listing 2. XML description of a sentence structure

```

<boc>
    <brick id="devoured_1" coords="0.37,0.1"/>
    <brick id="cat_1" parent="devoured_1" connector="1"/>
    <brick id="mouse_1" parent="devoured_1" connector="2"/>
    <brick id="the_1" parent="cat_1" connector="1"/>
    <brick id="little_1" parent="cat_1" connector="2"/>

```



```
<brick id="a_1" parent="mouse_1" connector="1"/>
</boc>
```

Such description defines the content (words, grammatical attributes, and connectors) for all the blocks available in the given exercise. The format is designed to be simple and easy to use. The basic assumption is that any word form and any alternative set of attributes of the given word is described as a separate block. The final section of the XML file describes the exercises, and their expected solutions. Thus, the teacher needs to create an XML document with the description of words, syntactic forms and attributes with connectors to create a new exercise or a subset of language grammar for student experiments. The shapes of the blocks should be developed considering the most frequent combinations of syntactic phrases to further emphasize the correct order of the words with smooth transitions (see Fig. 13). However, the teacher can change the shapes to better adapt WordBricks to another language or lesson. We are also planning to create a graphical tool to design XML rules without actually having to write XML.



Figure 13. Block shapes reflect the natural flow of words in a sentence

When the user attempts to insert a brick into a connector, the system checks whether the connector's part of speech and the brick's part of speech do match. If parts of speech are the same, attributes are being checked next. If the attribute list of the connector includes the attribute list of the brick, then the brick is inserted into the connector successfully. Moreover, it does not matter if the brick has a dependent brick or not. The order of brick connection actions is also not important.

5.4 Interactions Between WordBricks Components

The scheme of the general structure and interaction of components of the application in a simplified form is shown in Fig. 14. The main steps of interaction are as follows.

The user can select words from the vocabulary exercises and add them to the screen (item (1) on the Fig. 14). Descriptions of the corresponding bricks are retrieved from the XML exercise file (item (2) on the Fig. 14). The descriptions are then parsed and stored in the class Brick instance (item (3) on the Fig. 14). After that, a new element of `BrickView` class is added and displayed on the screen (item (4) on the Fig. 14).

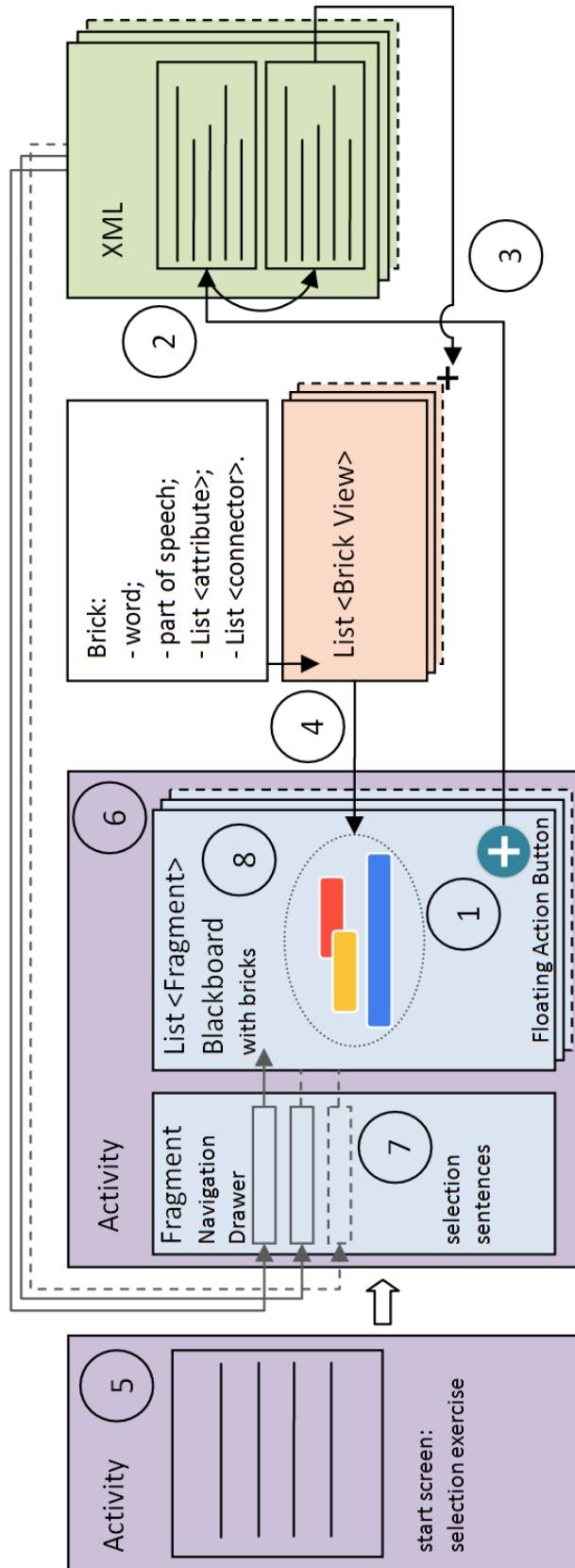


Figure 14. Scheme of the structure and interaction of components of WordBricks

5.5 Responsive Design of WordBricks GUI

The Start Activity (item (5) on the Fig. 14) is a list of chapters of the textbook. After selecting the exercise, the user goes to the main screen of the application. The user interface of the main Activity (item (6) on the Fig. 14) is a standard Navigation Drawer (item (7) on the Fig. 14) recommended from the official Google documentation [66]. It is used for switching fragments of the working area (item (8) on the Fig. 14).

In WordBricks, the Drawer on the left part of the screen allows the user to switch between the exercises. Working area of the Blackboard is implemented through a `Fragment` class from the Android framework.

New bricks can be added using the floating action button, located in the bottom right corner of the application screen. Editing or deleting a block is performed by double-clicking on the block. In addition, this can be done via the Action Bar.

Blackboard: The user of the WordBricks can move bricks and create sentences by employing a drag-and-drop interface. Bricks can be connected in any order. GUI contains color hints and pop-up information about words at the bottom of the screen for convenience of the user.

Responsive design: Sizes of bricks are calculated dynamically according to the screen resolution. Thus, the bricks look normally on the screens of any screen size and density. Also, the application has an ability to zoom the screen of the Blackboard for individual user settings. However, a horizontal orientation is preferred since a relatively long sentences hardly fit the screen.

A number of technical and pedagogical challenges is caused by the distinctive features of mobile platforms. The small mobile screen cannot display the complete set of brick attributes, so we had to intentionally hide some of them, and make the remaining attributes easy to see and understand. We also had to support numerous possible user actions via limited tap interface, and ensure proper auto-positioning and sizing of bricks.

5.6 Visualizing sentence structure

WordBricks visualizes sentence structure with a combination of nested colored bricks of different shapes. Each brick is associated with a set of textual attributes that unambiguously define the shape and the color of a brick. A brick also contains a linear list of child items, consisting of words and connectors. A word is a static predefined textual element, drawn on a brick background. Each connector is a placeholder for a brick that forms a dependency relation with the connector's parent brick. Like bricks, connectors are associated with textual attributes that define their shapes. From the technical point of view, the classes for bricks and connectors are based on the standard Android framework (see Fig. 15). When a brick is inserted into a connector, the respective `BrickView` object is placed on the corresponding `ConnectorView` object. This operation is possible since both classes are inherited from the standard class `RelativeLayout` that can work both as a graphical element, and as a drawing canvas holding other `View` objects.

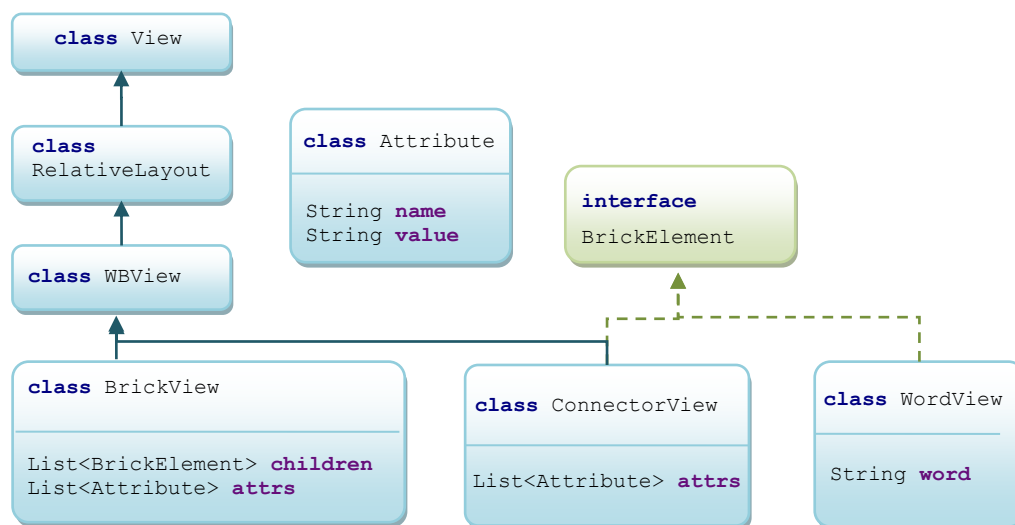


Figure 15. Class diagram of bricks visualization subsystem

A brick can be placed inside another brick's connector if that connector's attributes form a subset of the set of brick attributes. Let us state once again that child bricks are displayed inside the connectors of their parents, thus the whole structure preserves the original linear form of a sentence. When WordBricks is

used as a visualization module, it displays a static structure of already linked bricks, corresponding to a user-supplied sentence.

Since WordBricks performs no linguistic processing of the input data, it can be adapted to a variety of natural languages and grammar formalisms. This design decision also makes it easy to use WordBricks as a visualization module for any given sentence structure, properly encoded in compatible XML documents. Each XML file corresponds to a single sentence and contains three sections. Brick descriptions section defines the attributes, words and connectors of each brick present in the given sentence.

5.7 Interoperation with Language Processing Modules

One of the primary design decisions in WordBricks was to enable full control over pedagogical aims of individual exercises, which in practice means teacher-specified rules of grammar and list of words, available in each exercise. However, this approach requires a large amount of manual work for composing XML rules and limits the students with a set of predefined words and rules, thus reducing their options for open experiments with language grammar. To address this problem, we are experimenting with elements of natural language processing that automate the process of words-to-bricks conversion. Currently, there are two scenarios for using these capabilities: 1) convert a whole user-supplied sentence into a brick representation to make its structure clear; 2) convert individual user-supplied words into bricks and add them into the current exercise. Unfortunately, these scenarios rely on language-specific algorithms, so only English and Irish are supported at the moment.

The basic idea of sentence visualization mode is to substitute handcrafted bricks with bricks descriptions, generated with natural language processing algorithms. In this mode, the system operates according to the following scenario:

1. The user inputs an arbitrary phrase or a sentence into the system.
2. The system uses natural language processing modules to tokenize the supplied text into individual words and perform necessary linguistic analysis of their features.
3. The system uses this information to build a brick structure.

4. The resulting structure is shown on the screen.
5. We should note three obvious drawbacks of this solution comparing to handcrafting word bricks:
6. Natural language processing modules are language-specific, so we have to provide a separate module for each supported language.
7. Language processing algorithms are imperfect, so the resulting structure might actually be incorrect.
8. Automated algorithms produce a separate brick for each word, which might not be the best choice for a particular situation from a pedagogical point of view.

However, our experiments show that capability to visualize any arbitrary phrases outweigh these shortcomings, and it should be investigated further in practice.

Since natural language processing modules require to use specialized libraries and large binary data files, it was decided to perform language processing operations on a remote server, keeping the client side of WordBricks lightweight. However, we might revise this architecture in the future. In the present system, mobile WordBricks application processes specific XML description of individual bricks. This description, in turn, can be supplied either by the user (in form of predefined exercises) or by the server as a result of user query.

Let us consider the case of visualizing arbitrary user-supplied sentences [51]. When the user selects the option “Visualize sentence” in the main menu, the application displays an input box prompting to provide any arbitrary sentence. After the user taps the OK button, the application sends an HTTP GET request to the server side. The server returns an XML document describing the desired brick configuration to be displayed on the screen.

The server side (backend) is a Python CGI script, accessible via HTTP interface (see Fig. 16). Most of linguistic processing is performed in external executable modules, invoked by this script.

We consider the input to be individual sentences, thus there is no dedicated sentence splitting stage. For Irish language our natural language processing pipeline is based on open source Python-based NLTK platform [77] and owes much insight to the substantial work of T. Lynn [78]. We also rely on the Irish UD Treebank (IDT-UD), obtained by converting the original Lynn’s Irish Dependency Treebank to the universal dependencies annotation scheme [79].

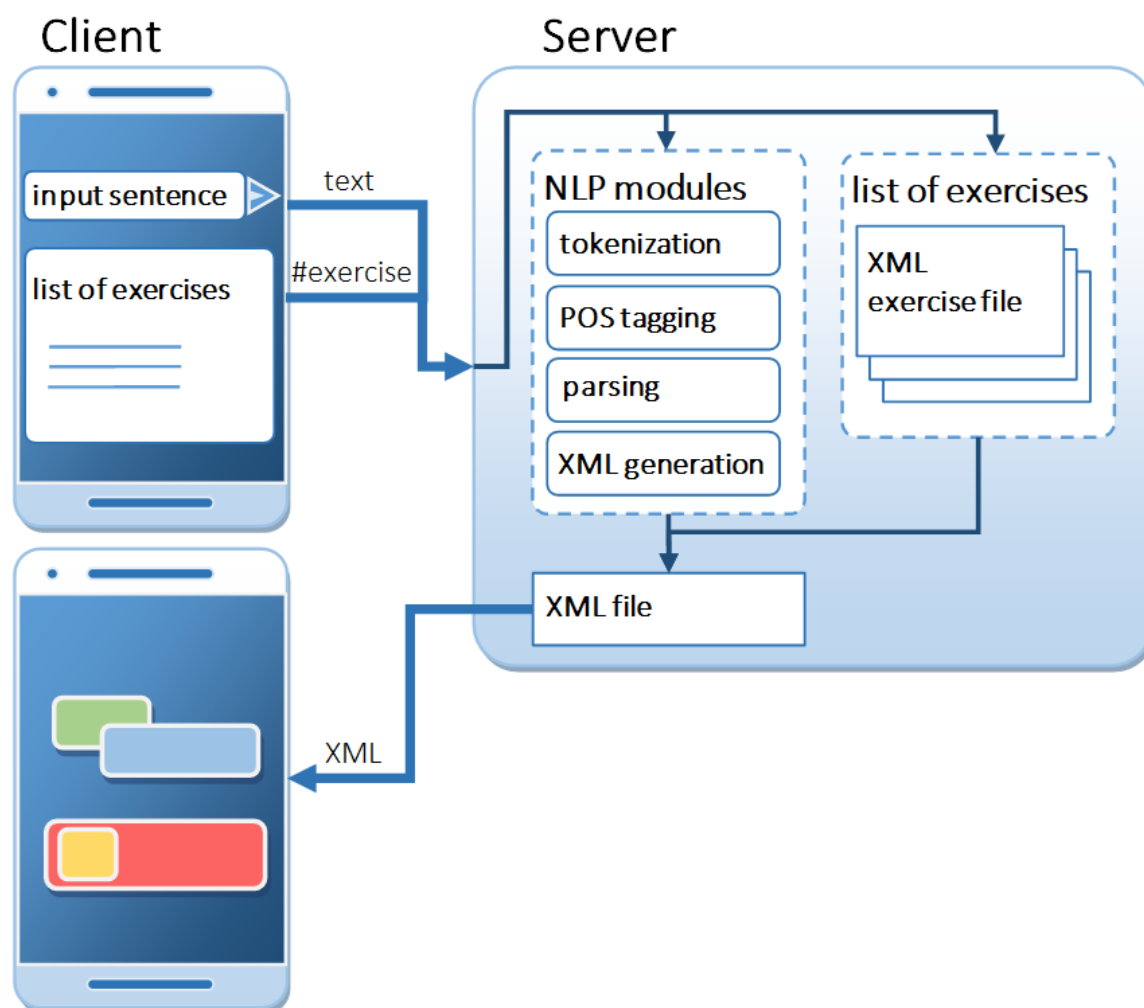


Figure 16. Architecture of the system

The processing pipeline includes the following steps (see Fig. 17 for English language and Fig. 18 for Irish language).

Tokenization. The input sentence is divided into a number of tokens, corresponding to individual words and punctuation marks of the sentence. Tokenization is not a trivial process, since there are no simple and clear rules to identify token boundaries. For example, most existing NLP modules expect

contracted constructions such as *you'll* and *don't* to be represented as separate tokens (*you'll*, *do/n't*). There are different approaches to independent semantic units with spaces or dots inside, such as *10 000* or *e.g.*. The current tokenizer uses a rule-based algorithm that respects most requirements of the subsequent modules, and works correctly in most cases we deal with, though additional evaluation would be useful.

In case of Irish language we follow the approach suggested by Lynn [78]: first, the text is processed with a conventional rule-based English tokenizer, then a set of Irish-specific rules is applied to treat contractions b', d', and m' as separate tokens. In the original Lynn's work, compound prepositions such as *go dtí* are concatenated and treated as single tokens, but we do not do it to be consistent with IDT-UD annotation.

Part of speech (POS) tagging. The purpose of this step is to mark the tokenized sentence with part-of-speech tags. We use Penn Treebank tagset as a de-facto standard for natural language processing [66]. Our tagger is based on the *maxent* toolkit [67] that implements maximum entropy modeling technique [68]. The tagger first had to be trained on a sample tag-annotated corpus. We relied on the manually annotated part of Open American National Corpus (it includes around 500K words chosen from a large variety of sources) [69]. In our experiments, the tagger exhibited 96.40% accuracy, comparable to the state of the art.

To mark the tokenized sentence with part-of-speech tags for Irish language, we trained NLTK's built-in Perceptron Tagger [80] on the IDT-UD treebank without any adjustments of default features. The resulting accuracy (verified by splitting IDT-UD into 920-sentence training set and 100-sentence test set) reached a value of 90.42%, which is close to the best attempt of Lynn (93.02%), achieved for a twitter messages tagging task with more advanced tools.

Syntactic parsing. The sequence of tagged tokens is passed to the dependency parser that converts the input into a dependency tree, providing the output as a CoNLL-U-formatted text document [70]. Our parser is based on the source code of Layer-Based Dependency Parser LDParse [71]. Similarly to our part-

of-speech tagger, this parser had to be trained on an dependency-annotated corpus (a *treebank*). For this purpose, we used the WSJ section of Penn Treebank, containing over 2 million words parsed for predicate-argument structure [72]. The treebank had to be converted first into dependency structures with a *pennconverter* tool [73]. The resulting accuracy can be considered acceptable: 84.54% for unlabeled, and 83.28% for labeled parsing.

In case of Irish language following Lynn’s approach, we used MaltParser [81], trained on IDT-UD, and tested the accuracy by splitting the treebank into 920/100-sentence training and test sets. Default settings yield lower performance (LAS = 0.70) comparing to the values achieved by Lynn (0.74...0.79 in different experiments). However, these values should be considered relatively rough estimations due to small size of the test set.

XML generation. The final processing stage involves conversion of the CoNLL-U data into the XML document supported by WordBricks. Since CoNLL-U format contains all required information (word boundaries, part-of-speech tags, and labeled syntactic dependencies), this operation is relatively straightforward for both languages, and is performed by the main Python server module.

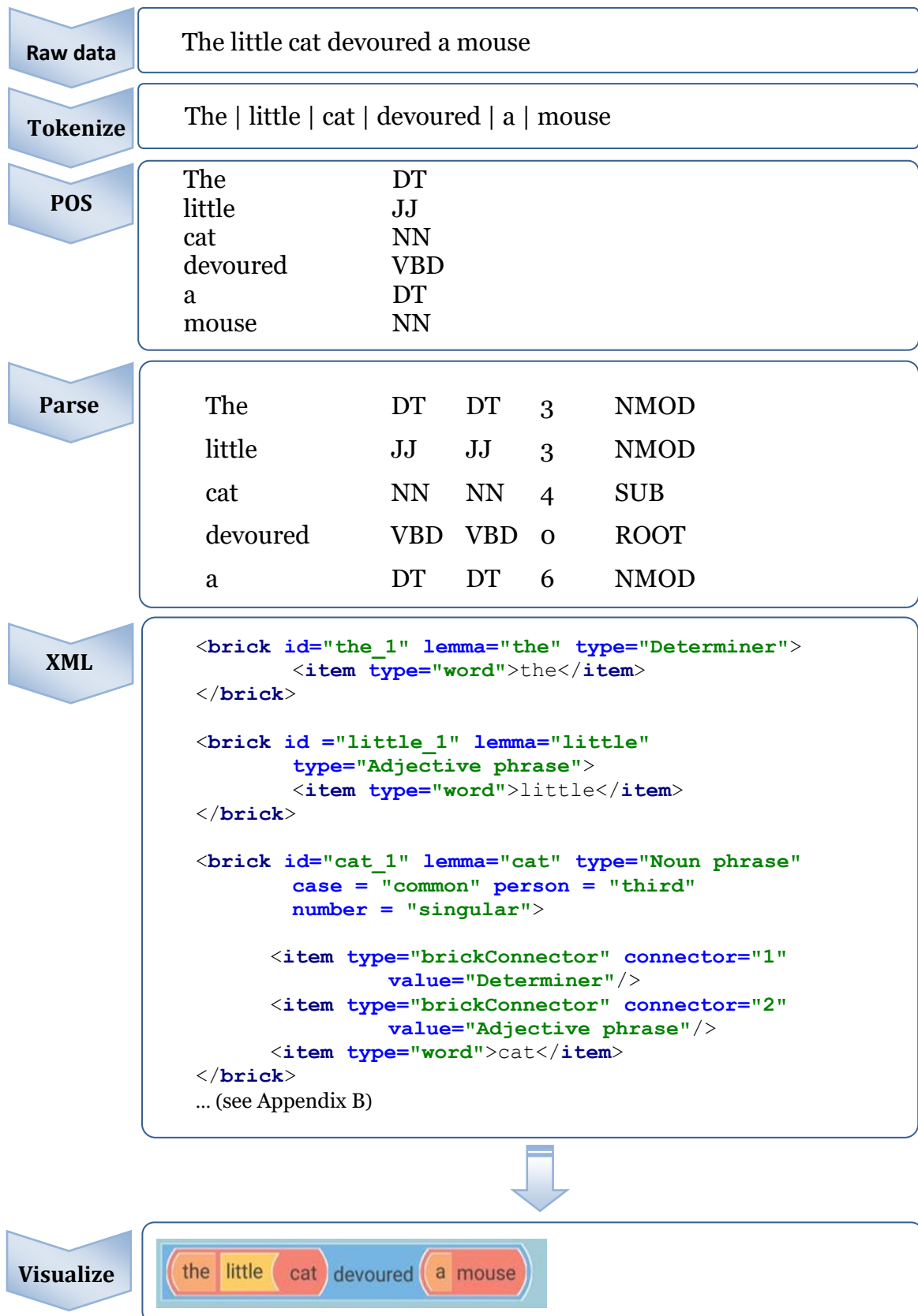


Figure 17. Processing pipeline steps for the sentence *The little cat devoured a mouse*

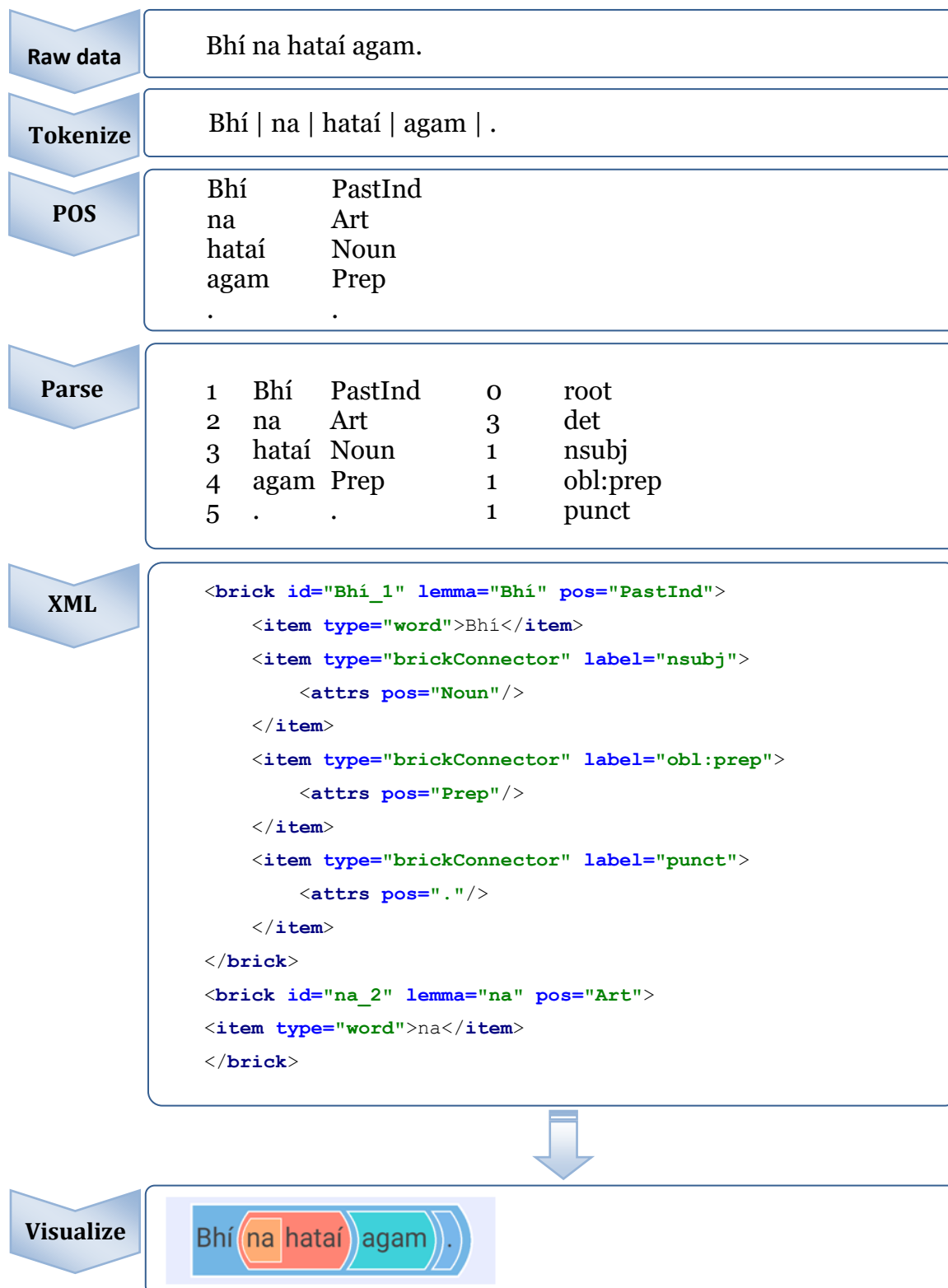


Figure 18. Processing pipeline steps for the sentence *Bhí na hataí agam*

Chapter 6

6. Classroom Experiments with WordBricks

In this section, we will discuss several experiments conducted to prove feasibility of WordBricks. We wanted to evaluate its pedagogical merits in diverse environments and scenarios. To the present day, three independent studies were completed [89]. The first study was aimed to prove that WordBricks can be helpful as a learning aid in a conventional English as a second language classroom, taught for the students of computer science at a Japanese university. The second study explored the capability of WordBricks to serve as a demonstration tool used by the teacher to illustrate certain grammatical phenomena. It was conducted with a different group of students at the same university. The goal of the third study was to test WordBricks in a course of some other language rather than English. WordBricks is, in principle, language-independent, but the difficulty of expressing grammatical constructions with specific types of visual blocks may vary. This study was performed at an Irish language class of a junior school in Ireland.

As a part of each experiment, we also asked the users to provide their feedback on the interface of WordBricks and their suggestions for its subsequent development. Some of these suggestions were implemented in later versions of the software.

The diversity of experimental setups and different approaches to evaluation of the system is driven by the needs of teachers and students participating in our studies. As mentioned above, WordBricks was initially designed as a tool for “conscious learners” who would download the app and use it for their own language learning needs (like Duolingo). However, teacher interest to the system motivated us to do a series of pilot studies in classrooms, which would give us

some perspective on the possibility to use WordBricks in schools. Experimental settings reflect the difference in educational goals. The teacher in the first study was motivated to improve his students' test scores. Many of these students were intended to re-take a TOEIC exam after the course and wanted to see how their results improve after the course. The teacher in the second study conducts a dedicated English grammar course, based on traditional rules-and-exercises textbook. He was looking for a way to provide better visualizations of grammatical phenomena he had to explain (mostly in a non-interactive style). The teacher in the third study deals with young learners of primary and junior high school, having low motivation to study Irish language, which is widely regarded as a compulsory subject with little practical utility. Thus, her primary interest was to introduce interactive, game-like experiences that would increase learners' motivation to engage in educational activities. Therefore, our evaluation concerns three loosely related categories of merits of the app: a) capability to serve as a learning aid that facilitates better understanding of language grammar (that results in higher test scores); b) capability to serve as a visualization tool for illustrating particular grammatical points; c) capability to introduce game-like elements that make educational process more enjoyable even if it does not immediately translate to language proficiency.

6.1 WordBricks as a Learning Aid

Let us begin with a brief discussion of our first attempt to use a relatively early version of WordBricks in a real classroom environment. Our goal was to gather initial responses from the teacher and the students, and assess the feasibility of the chosen approach (the study is described in more detail in [52]). The participants of the experiment were two randomly chosen groups of sophomore computer science students at a public Japanese university. The students were enrolled in an elective advanced English grammar course and ranged in age from 19 to 21 years. First, we conducted a diagnostic English grammar test to ensure that the pre-course English level of both groups is very similar (see Table 1). Approximately one half of these students have passed a TOEIC test recently, their average score is 350 points.

To investigate whether WordBricks had any observable effect on students' grammar learning, we adopted a pre-test/post-test design with a control group and experimental group. In this setup, all 21 participants studied units 69 and 70 from the same *English Grammar in Use* textbook [41] with the same teacher. Both units are about countable and uncountable nouns. However, Unit 70 seems to be more demanding than introductory Unit 69, since it is dedicated to more advanced grammatical points.

Table 1. Descriptive Statistics for Two Groups in Diagnostic Test

Group	Number of participants (N)	Mean Test Score (M)	Standard deviation (SD)
Control Group (G1)	11	65.25	7.50
Experimental Group (G2)	10	65.90	7.99

Though each group covered the same content and underwent the same English grammar assessment procedures, the control group was taught with an English grammar textbook in a traditional way (teacher-centered, grammar focused), but the experimental group autonomously interacted with WordBricks using Android-based tablets, given to each participant. Before the experiment, control group members could familiarize themselves with WordBricks by solving pre-designed exercises, based on the first paragraphs of the Azar and Hagen's grammar book [40].

Based on the course textbook, we developed two sets of paper-based English grammar tests to measure participants' English grammar performance over two course units. For Unit 69 pre-/post-test, participants were asked to correct given sentences focusing on the nouns of the sentences. For Unit 70 pre-/post-test, they were asked to complete sentences using correct noun form. According to the test results, the experimental group showed greater improvement for both topics. The average score of the experimental group (G2) increased from 15.90 to 24.20 points (out of 30 possible) for the first grammar topic, while the average score of

the control group (G1) increased from 15.18 to 21.00. Subsequently we conducted a similar experiment with a group of 16 students who studied material of both units 69-70 as a single block, where the average score improved from 17.13 to 20.69 for the control group, and from 17.94 to 20.31 for the WordBricks group (see Table 2). These results show that our application can be as efficient as a textbook, at least, in certain contexts. To compare groups' achievements, we performed a paired samples t-test using aggregate pre-test/post-test data of G1 and G2. The resulting values (0.00005 for G1, 0.00026 for G2) indicate that both groups achieved statistically significant progress, and WordBricks (G1) group performed better. These results show that our application can be as efficient as a textbook, at least, in certain contexts.

Table 2. Results of the Quantitative Experiments

Exp.	Test type	Group	Group size	Mean score	Std. dev.
#1 Unit 69	pre-test	G1	11	15.18	5.04
		G2	10	15.90	4.43
	post-test	G1	11	21.00	5.80
		G2	10	24.20	4.02
#2 Unit 70	pre-test	G1	11	6.00	2.72
		G2	10	4.20	2.57
	post-test	G1	11	9.18	4.17
		G2	10	11.60	2.84
#3 Units 69-70	pre-test	G1	8	17.13	3.80
		G2	8	17.94	4.64
	post-test	G1	8	20.69	2.91
		G2	8	20.31	2.83

Figure 19 illustrates progress achieved by individual students of both groups. Test scores indicate the overall percentage of correctly accomplished assignments. The diagrams show that in both groups teaching was effective, and students in general were able to improve their test scores. Most progress was made by the participants having lower initial scores, which is not surprising. It is

also noticeable that the difference in pre-test and post-test scores is larger in the WordBricks group.

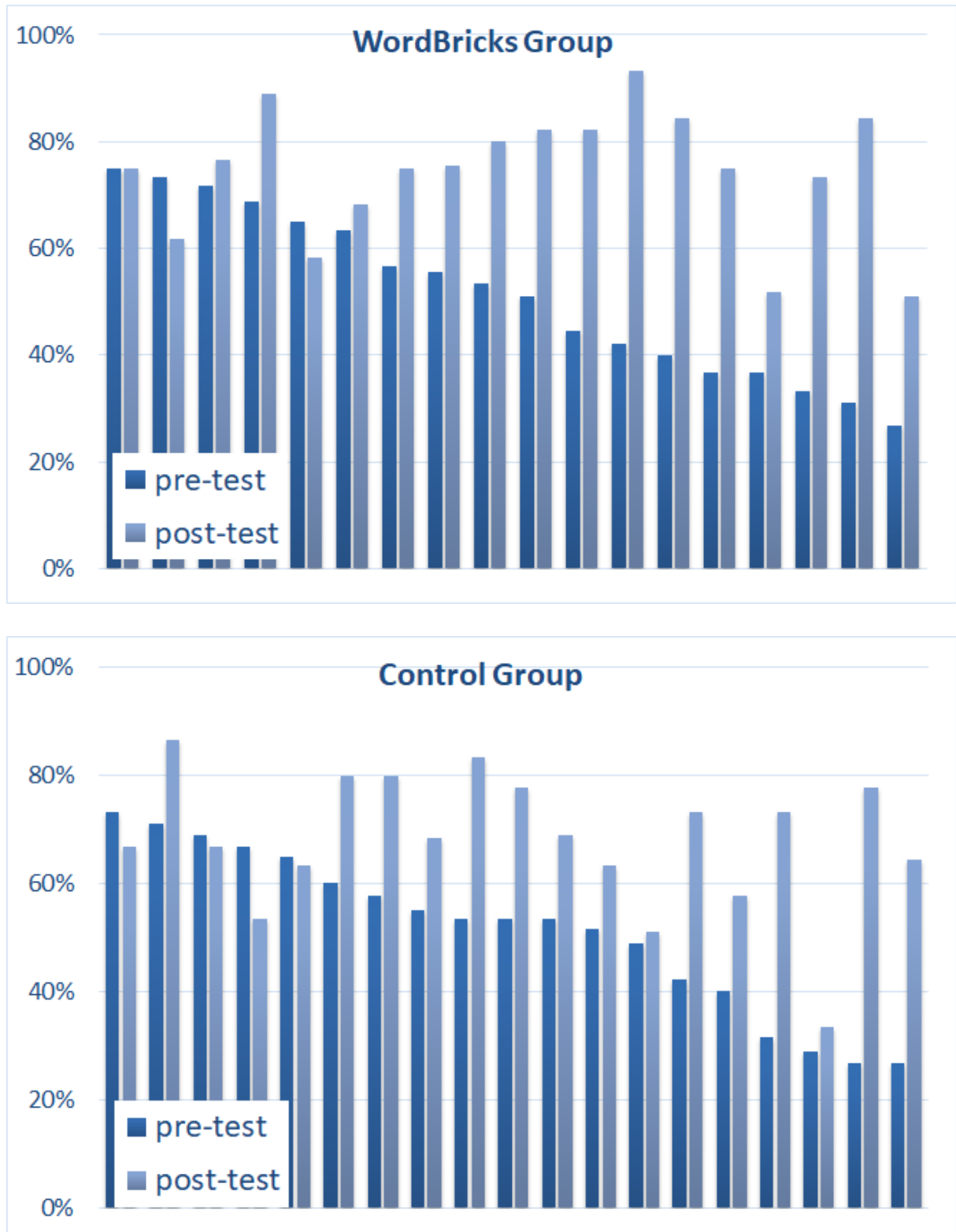


Figure 19. Progress of individual students in WordBricks and control groups.

6.2 WordBricks as a Demonstration Tool

Early classroom experiments with WordBricks made us believe that the system can also be used by the teachers as a demonstration tool within the framework of traditional grammar instruction. Currently, teachers often rely on PowerPoint or similar presentation software to present grammatical concepts or analyze the grammar of sentences [53]. However, PowerPoint presentations are comprised of linearly organized display units (slides) [54], which might not be the best way to present non-linear or hierarchical concepts to the audience. When using presentation software, language teachers tend to use different fonts, colors and shapes to visualize the grammar of a sentence. WordBricks alleviates this need by providing ready-made building blocks for typical grammatical structures. More importantly, as discussed above, their selection was based on established theories of grammar [46] and pedagogically sound approaches [45]. Therefore, there is no need for language teachers to create customized elements, saving their time and providing advantage over more generic demonstration tools.

The possibility of using WordBricks as teacher's aid was evaluated in a small group of seven 4-year computer science undergraduate students (22-25 years of age), enrolled in an Advanced English course at a Japanese public university. This course is primarily focused on helping the students write a graduation thesis, which is structurally similar to short research articles in computer science. As part of this thesis writing course, useful sentence structures are discussed using sentences extracted from an example research article [55]. We selected ten sentences for division into WordBricks blocks to explain particular target structures (see Table 3). Since the course was focused on larger structures than individual words, we had to design custom WordBrick blocks that represent these structures (see Fig. 20).

Since WordBricks is mobile software, we had to setup a virtual Android machine using Oracle VirtualBox on a teacher's Windows 10 laptop to display presentations. In total, five presentations of 15-20 minutes were delivered. Each presentation analyzed two new model sentences, and reviewed previous sentences. Presentations consisted of the teacher constructing sentences using the blocks while eliciting and explaining the reasons from the placement of each

block. Both suitable and unsuitable selections of blocks were made to provide students with opportunity to contribute ideas. After each presentation, students worked in pairs or threes to discuss the two new target structures. This was followed up with a writing task in which students created a sentence related to their research using the target structures.

As a result of this study, the teacher identified several strong features of WordBricks, helpful in the context of demonstrating grammatical constructions. In particular, he noted that automatic handling of colors, shapes, and grammatical correctness of the resulting phrase helps to avoid errors during presentation, when attention is focused on the audience rather than manipulating elements using a mouse. The interactive nature of WordBricks that allows to build sentences gradually, block by block, was also mentioned. Finally, it was suggested that preparing demonstrations in WordBricks can be faster than using other demonstrational software, though at the present moment it requires manual XML file editing.

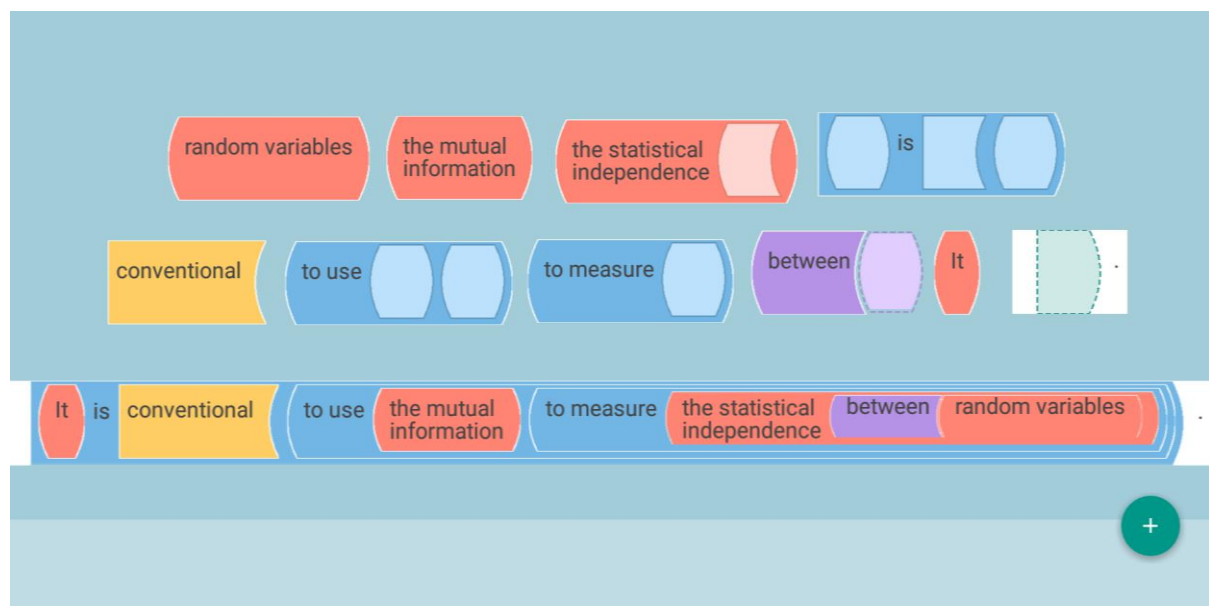


Figure 20. Model sentence #3 chunked into grammatical units

Table 3. Model Sentences and Target Structures

#	Sentence	Target structure
1	Secret sharing is a method of encrypting a secret into multiple pieces called shares so that only qualified sets of shares can be employed to reconstruct the secret.	A is a method of B so that C can be D.
2	Audio secret sharing (ASS) is an example of secret sharing whose decryption can be performed by human ears.	A is an example of B whose C can be D.
3	This paper examines the security of an audio secret sharing scheme encrypting audio secrets with bounded shares, and optimizes the security with respect to the probability distribution used in its encryption.	This A examines B and optimizes C.
4	Figure 1 illustrates an example of two shares and their superposition of a (2; 2)-threshold VSS scheme.	Figure # illustrates X.
5	Desmedt <i>et al.</i> [4] proposed information-theoretically secure schemes that encrypt a binary string secret.	X [#] proposed A that B.
6	It is conventional to use the mutual information to measure the statistical independence between random variables.	It is A to B.
7	Let P be a finite set, and let A_Q and A_F be subsets of 2^P .	Let A be B.
8	Table 1 summarizes the existing works on ASS and VSS schemes as well as this work.	Table # summarizes A.
9	First, we provide a formal definition of ASS schemes and a construction of the simplest ASS scheme, namely a (2; 2)-threshold ASS scheme.	We provide A and B, namely C.
10	The result indicates that the security is optimized when the variance of the normal distribution approaches infinity.	A indicates that B when C.

6.3 WordBricks at an Irish Language Class

WordBricks was initially designed to be language-independent in a sense that it is not based on presumptions, specific for certain particular language. However, we wanted to have practical evidence that WordBricks is flexible enough to handle non-English grammatical structures. Our current experiments

are focused on integrating WordBricks into Irish language classes at a junior school in Ireland [56]. We also hope that gamification of Irish classes through WordBricks would help to support learners' interest in the subject. While the Irish language is a compulsory subject in Irish schools, only a very small minority (3%) of country's population use Irish as a community and household language [57]. These Irish speakers are bilingual (Irish/English) and there is no communicative need to learn Irish [58]. The overwhelming majority of people in Ireland (82%) believe that Irish should be taught in schools [59], but in practice many learners tend to struggle both with the language and with lack of motivation.

The current version of the Irish WordBricks application deals with some of the basic constructs of Irish that learners must master yet find difficult due to their structural difference from English. Most school teaching of Irish follows the traditional model of books, workbooks and teacher-led activities. Irish WordBricks introduces some game-like aspects by enabling learners to construct their own grammatically correct sentences in Irish and reinforce Irish word order. For example, the phrase *"I have a hat"* is *"Tá hata agam"* in Irish (literally, "Is a hat with me"). Learners can find this structure difficult, as they may try to map the Irish words onto the English word order, and WordBricks can help them to see the real word connections in this phrase (see Fig. 21).

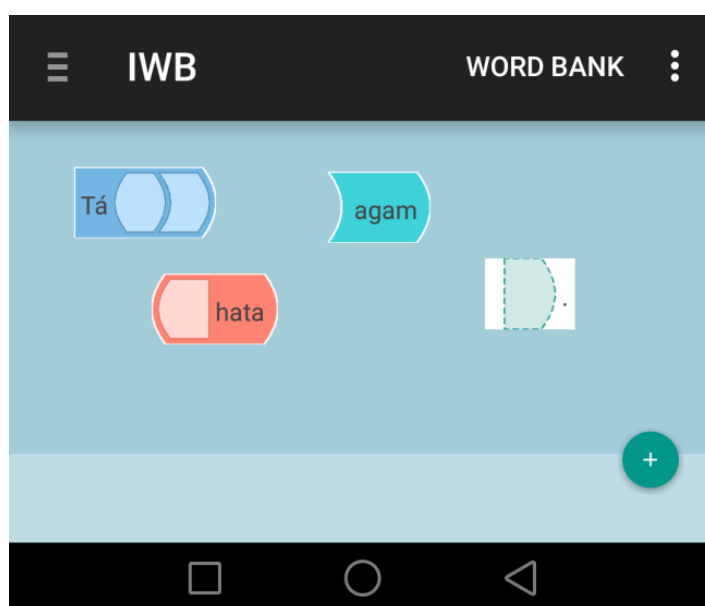


Figure 21. Irish "have" construction in WordBricks

We had no chance to perform an extensive evaluation of the system, but the initial responses of the teachers, parents, and school pupils were positive. The learners in general reported that they enjoyed working with the application, found it easy to use, and would like to use it as a part of their homework. The parents also enjoyed using the application and thought it was a very good idea to have such an application for Irish. Several primary school teachers also reviewed Irish WordBricks. They were positive about the application, and found its interactive elements appealing for their students. Even though Irish WordBricks was initially designed for a single user in an independent learning situation, the teachers plan to use the application in their classrooms. The idea is to ask the students to form sentences using the classroom computer so that all students can see and become familiar with the grammatical structure being studied.

We conducted several pilot studies to find out how enjoyable is the system for users, and the initial responses of the teachers, parents, and school pupils were positive [88]. Let us briefly report the results of the studies of using WordBricks in visualization mode, implemented in several different primary school classes.

6.3.1 Evaluation in Irish Language Classroom First Study

The first study involved a mixed group of 46 school students, 8-12 years of age. Our goal at this stage was to gather their initial impressions about the app and understand possible directions of subsequent development. We asked the students to play freely with the app, perform basic assignments, and analyze the structure of several suggested sentences. Quantitative evaluation (See Table 4) was carried out via anonymous student surveys. Naturally, we are aware that test participants (especially young children) might be tempted to give “the right” positive feedback, so the results should be treated with caution. However, the teachers tried to explain them that both positive and negative comments are highly welcome, as they help to improve the app.

The initial version of Irish WordBricks has been trialled by both parents and young learners. The young learners reported that they enjoyed using the application and they found it easy to use. They thought it would be useful for learning Irish. Without prompting, one young learner suggested that there could

be more vocabulary words so students could make more interesting and longer sentences (the initial version included a limited vocabulary so that learners could focus on structure). The young learners suggested that the application could be used for learning various types of sentences. When asked if they would use the application out of school, they said they might and that they would like a new topic each week. Initial feedback from parents has also been positive. Parents whose children attend an Englishmedium participated in an Irish course for parents. Some parents had spent 13 years learning Irish (in both primary and secondary school), but had very limited mastery or recall of the language. There were also several immigrant parents who had never studied Irish before, but they were usually multilingual and were comfortable with other languages. The parents enjoyed using the application and thought it was a very good idea to have such an application for Irish. Several parents reported that they struggle to help their children with their Irish homework and have tried in vain to find something useful for them as parents to either revise their knowledge of Irish or learn it from scratch in the case of immigrant parents. They thought the application would be very useful for their children and would like their children to use it at home. Several primary school teachers also reviewed Irish WordBricks. They have extensive experience of teaching Irish and are very aware of the need to use modern tools and techniques in (and outside) the classroom. They were positive about the application and thought that it would be a useful tool in their classroom. They liked the interactive element of the application and thought it would appeal to their students. Even though Irish WordBricks was initially designed for a single user in an independent learning situation, the teachers plan to use the application in the classroom with their students. They will ask students to form sentences using the classroom computer so that all students can see and become familiar with the grammatical structure being studied.

Table 4. Summary of student responses in the first pilot study

Question	Yes (%)	No (%)	A bit (%)
Did you enjoy the Irish WordBricks app?	95	0	5
Did you find the Irish WordBricks app easy to use?	78	0	78
Do you think IWB helped you to learn Irish?	78	2	20
Would you like to use the Irish WordBricks app at home?	56	2	42

6.3.2 Evaluation in Irish Language Classroom Second Study

The purpose of the second study was to test the applicability of WordBricks in real classroom setting, i.e., when it is used to illustrate a particular language phenomenon according to the plan of a lesson, and the app is primarily used by the teacher rather than students.

The participants in this case represented two cohorts:

- A. Two groups of 5th year school students (10-12 years of age), 44 participants in total. Had 7 years of Irish language education, including 5 years of written Irish. Worked with five different grammatical constructs over a five-week period.
- B. Three groups of 3rd year school students (8-9 years of age), 75 participants in total. Had 5 years of Irish language education, including 3 years of written Irish. Worked with three grammatical constructs over a four-week period.

WordBricks was used by the teacher, who ran it from a desktop machine with Android emulator installed. Each grammar topic was illustrated with two or more example sentences, processed by the app. In general, user/app interaction was kept at relatively low level, since primary school timetable and focus on particular topics have to be tightly managed by the teachers, and each lesson lasts for only 30 minutes.

Qualitative evaluation (see Table 5) was performed via anonymous student surveys, teacher surveys and observation. Students filled out a questionnaire after each session with the app. Since we found little differences between the cohorts, the results we provide represent combined data.

Table 5. Summary of student responses in the second pilot study

Question	Yes (%)	No (%)	A bit (%)
Did you find WB easy to use?	82	2	16
Did you think it helped you to learn Irish?	73	7	20
Would you like your teacher to use WB in class?	81	5	14
Did you enjoy WB?	84	1	15

We recognize that the actual impact of WordBricks on language education has to be reconfirmed with both quantitative and qualitative evaluations involving larger groups. We must note, however, that such experiments are hard to perform in the context of minority-language primary school classes, where there are many variables outside our control (including differences in teaching approaches, students abilities, textbooks used, and class size), and the total number of learners is limited. Still, we consider the obtained results encouraging, as they demonstrate the feasibility of our approach, serving as primary motivation for subsequent work.

In this way we use WordBricks application as a visualization module that can display a user-supplied sentence as a parse diagram. (see Fig. 22). This sentence can be a part of an exercise or freely added by the user.

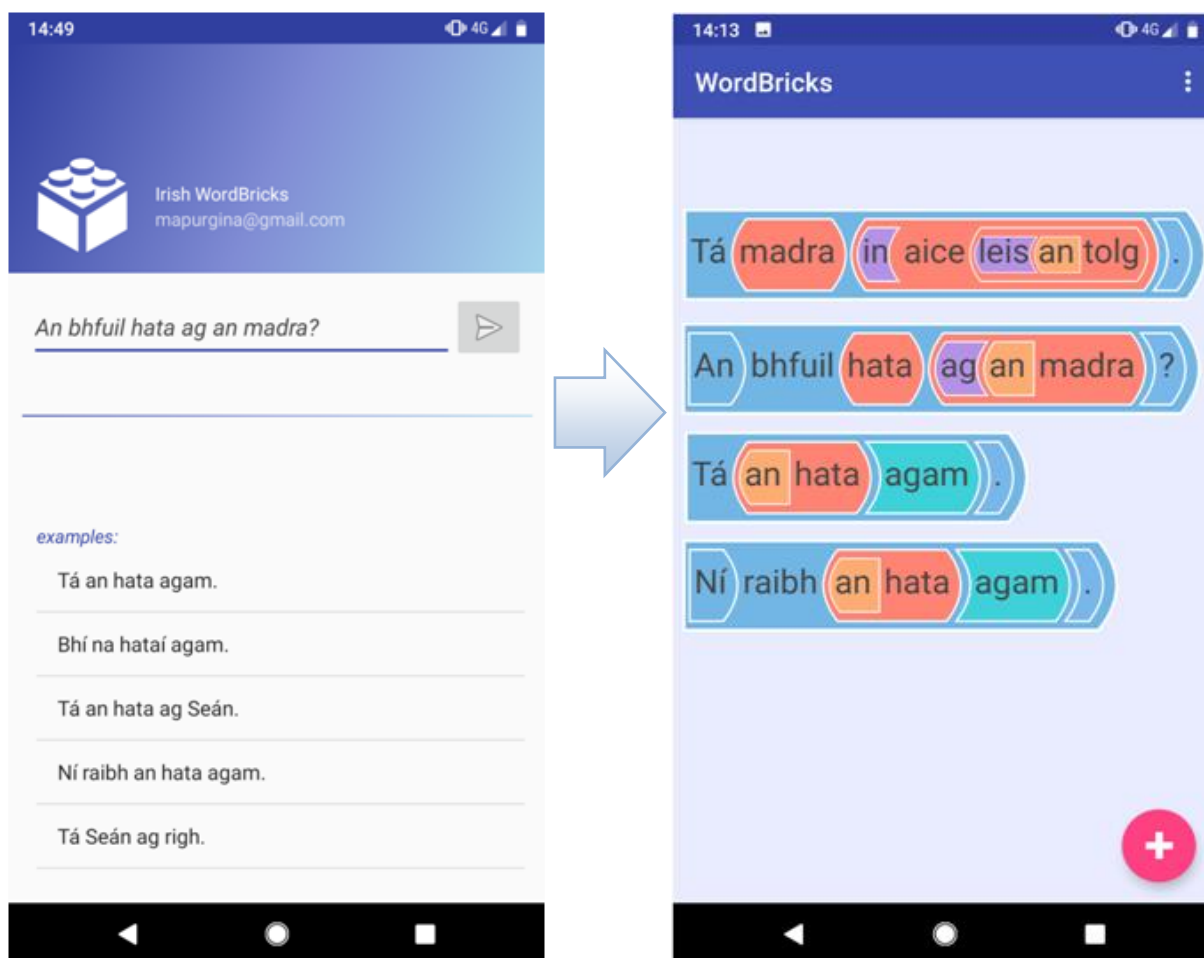


Figure 22. An example of how WordBricks application works with the Irish language

6.4 User Suggestions

As one might expect, we also received numerous report about shortcomings of the present version of our software. Some of them were related to particular bugs or user interface inconsistencies that were later improved. Some users suggested to turn WordBricks into a universal application, able to work on both mobile and desktop operating systems, which is especially important for its use as a demonstration tool. There were some specific requests such as “let the user save the current block arrangement to return to the sentence later”. Other comments reveal common needs of language learning software users in general, so we believe that they deserve additional discussion.

It seems that certain fraction of students in any given group have natural proclivity for teacher- and book-centered learning. They perceive mobile apps as

“not serious” types of learning aids, and often ask to make WordBricks more “book-like”, for example, by following a traditional structure of sections, containing explanations, examples, and exercises. Some of these students are also sufficiently proficient, so they found WordBricks content irrelevant for their knowledge. They also feel more comfortable when the application is designed as a “textbook companion”, containing exercises strongly following the textbook structure and vocabulary.

Likewise, certain users like WordBricks just for the sake of being a mobile app, as they find appealing more “technological” way of learning a language. Such people explained their positive attitude with responses such as *“I like fiddling with a tablet”*, *“WordBricks is like puzzle games, and I enjoy to study and play games”*.

Probably, the largest number of suggestions were related to the current limited set of supported words and constructions. Users found the system too restrictive, as it only implements predefined constructions taken from the textbook units used in our experiments. Some users explicitly requested the capability to add own words and rules for independent studies.

Another large portion of suggestions was directly related to gamification. During the above described experiments, users had to deal with a “plain” version of WordBricks implementing only the basic functionality of building up phrases from blocks, and containing no explicit game-like features. However, they immediately recognized potential for further gamification, and requested to implement simple additions, such as victory fanfare sounds, scoring system, and explicit user progression through goals and subgoals. This observation supports our earlier note that mobile gaming is such a common leisure activity nowadays so that the users often suggest moving the project further into this direction themselves (we must note though that most of our testers are young people, more likely to be engaged in gaming). Here we should also mention common requests to implement a system of hints and other feedback mechanisms.

6.5 Open Challenges

It would be unbalanced to focus on WordBricks advantages without discussing principal shortcomings of the present system. User interface limitations, limited vocabulary or inadequate user feedback can be addressed, but there are also harder challenges that should be discussed, as they highlight inherent difficulty of natural language, and problems relevant to a wider range of CALL systems.

Scalability issues. The present version of WordBricks assumes that the user picks up blocks from a “tray” and moves them to the main application window. Alternatively, a predefined set of blocks is assigned to a specific exercise, so when the user opens the exercise, the corresponding blocks appear in the main window. This approach is hardly applicable for large word lists, and organizing words into classes according to their part of speech is not sufficient either. We are working on a combination of word tray and text input interface to facilitate easier search of words.

Unintuitive structures. Dependency grammars provide intuitive word-linking rules for simple types of dependency, such as “subject-verb” or “noun-property”. However, for certain structures these rules are often based on conventions rather than on rigorous linguistic theory. They include relations between words making up proper names (such as “Joe” and “Doe” in “John Doe”); relations between the main and subordinate clauses; relations between the words in phrasal verbs (such as “look up”); relations involving auxiliary words, such as “have been doing”, and so on. Arguably, understanding sentence structure is beneficial to the learner (explicit structural diagrams are used, e.g., in Richard Webb’s *80/20 Japanese* textbook [60]), but some of the present constructions can be more confusing than helpful.

This situation can be improved to some extent by designing blocks corresponding to separate *logical entities* rather than words. For example, we can consider the construction “will have been” as an atomic block, thus removing the need to examine the relations between words inside this entity. In fact, this approach is in line with the original concept of dependency grammars described

by Lucien Tesnière, who distinguished *words* as syntactic elements from *nuclei* as compound elements carrying the same role as words [61].

Interface/visualization constraints. Many blocks should have optional, variable or dynamic list-like connectors, while the current system assumes that blocks have predefined connectors, specified in the configuration. For example, nouns can have optional associated properties (“[large] book”), the verb *to be* can be used with a noun or an adjective as an object (“I am a student / I am funny”), and many verbs can be linked with a number of indirect objects (“I bought a book [where / when / why]”).

One way to handle such flexibility is to let the users to add, remove or change block connectors while arranging sentences, if these changes do not violate grammatical rules. In other words, the system will provide certain “basic” blocks, and it will be a user’s responsibility to configure them properly. Such method is adopted in Scratch. It provides a range of mathematical functions, such as $\sin(x)$ or $\log(x)$, but the user sees only the \sqrt{x} block in the tray. Other functions are accessible via a drop-down list of the \sqrt{x} block. However, we must acknowledge that this approach will make user interface more complicated and will introduce new required actions into sentence building.

In addition, as noted above, our current visualization subsystem supports projective dependences only. However, so far we had no requirements to deal with non-projective dependencies in practice.

Pedagogical considerations. One may feel compelled to use WordBricks to encode a large number of specified rules of grammar. However, the flexibility of natural language grammar lets the system to interpret certain constructions as correct, while in practice they are most likely to be erroneous. Many “grammatical rules” described in textbooks are actually dictated by semantics rather than syntax. For example, *English Grammar in Use* [41] clearly states: “*we do not use **the** with names of people (‘Helen’, ‘Helen Taylor’, etc.)*”. However, a book on advanced grammar actually provides a case where *the* is used to disambiguate the subject of speech: “*that’s not **the** Stephen Fraser I went to school with*” [62]. Similarly, rules related to the choice of past vs. present perfect tenses in English

often mention that the words *already*, *yet*, and *just* are used with present perfect tense [41]. However, one may argue that they deal with semantics rather than syntax. Syntactically, adverbs (such as *already*) can be used with any verb forms. Finally, the student-produced sentence can be considered grammatical only with the help of counter-intuitive interpretations, such as in the classic garden-path sentence “*The old man the boat*” that relies on the meaning of “man” as “operate” [63].

The teachers designing the blocks have to decide which constructions are include and which are exclude, given the target level of learners. It is far more likely that the beginners will erroneously use “the” with a person’s name rather than do it correctly in few situations where it is acceptable. However, many cases require deeper involvement of semantics, and thus are beyond the scope of WordBricks.

Chapter 7

7. Discussion and Conclusion

7.1 Discussion

State-of-the art technologies have been used in language education for a long time. One of the recent trends is the rise of gamified mobile apps for language learning, supported by widespread reach of smartphones, and by the rise of mobile gaming as a popular leisure activity. This allows application developers to presume that many of their potential users are ready for game-like activities, and even expect to experience them in non-game apps. Language learning requires long-time commitment, and often involves going through routine tasks that hardly can be considered entertaining, so reasonable attempts to exploit human propensity for games should be supported. However, it might be tempting to interpret this suggestion too literally and endeavor to develop a real “educational game”, which in practice often turns out to be a substandard educational tool, and a substandard game. Successful projects are typically targeted at conscious learners and do not try to disguise themselves as “games”. Instead, they implement certain game-inspired tricks that help the users to stay on track.

In terms of content, most projects are based on traditional learning materials, (such as texts for reading, audio- and videoclips, and textbook-style explanations), and traditional exercise activities (quizzes, jumbled sentences / fill the gap / translate phrases grammatical exercises). We believe that natural language processing technologies are potentially able to support a variety of innovative educational scenarios, not available with traditional learning materials, but in practice few technologies are mature enough to reliably address

learners needs. For example, automated speech analysis is often criticized for providing misleading feedback.

Our primary motivation for creating WordBricks was to explore certain “technology-driven” educational scenarios that would make use of dedicated technologies, specifically designed for a purpose of language learning. At the same time, we tried to address the problem of technological limitations by restricting the users with activities that can be reliably supported. For example, it is nearly impossible to design a reliable grammar checker that would evaluate any given sentence and find errors. However, it is possible to restrict the users with the set of grammar rules and let them compose sentences that are considered correct according to the rules. Our current experiment show potential of this approach, and WordBricks is regarded highly both by teachers and learners. However, the flexibility of human language and the lack of formalized grammar rules presented in a textbook order makes the design of WordBricks exercises a very nontrivial and challenging task. Fortunately, in many scenarios it is sufficient to implement the structures that makes sense from pedagogical point of view, which is only a subset of all grammatically correct constructions. These considerations give us motivation to continue experiments.

7.2 Conclusion

In this work, we have briefly discussed the rising gamification of language learning via mobile apps and introduced our work-in-progress system WordBricks, targeted for natural language grammar acquisition. WordBricks allows the users to combine words into sentences using Scratch-inspired “blocks and connectors” approach that prevents them to form ungrammatical constructions. Currently, the system supports three primary use cases:

- 1) as an “open lab” for free experiments with language grammar structures;
- 2) as an exercises platform to be used in combination with a grammar textbook;
- 3) as a demonstration tool for a teacher.

We are evaluating WordBricks in diverse settings, involving different educational goals, student profiles, and different target languages. Our first

experimental setup confirmed that the system was able to help students to improve their English grammar test scores within the context of a dedicated grammar course. The second study demonstrated the capability of WordBricks to serve as a handy visualization mechanism of particular grammatical constructions in a primarily non-interactive lecture-based course. The third study emphasized user enjoyment and game-like elements of the app, appealing to young learners with low motivation to learn a language, taught as a compulsory school subject.

We are evaluating WordBricks in diverse settings, involving different student profiles and different target languages. Our evaluation shows that the chosen approach is regarded positively by all involved parties. Students also feel game potential in the app, and request for more game-like features, such as the ones found in Duolingo. Implementing them is our primary goal. At the same time, we have to admit that even formal adherence to textbook grammar cannot hide the whole degree of complexity of natural language. Grammar rules often rely on vaguely defined categories, semantics, and general knowledge, and thus can be hard to implement in WordBricks. Furthermore, the system of blocks provides an impression that all constructions are “equal” in a sense that they are equally correct according to the rules of grammar. However, in practice from a didactical point of view it might be preferable to stick to fewer rules, and to introduce less commonly used constructions at later stages.

To extend current experiments, we are also working on an improved and simplified version of XML format, describing blocks and block linkage rules. Ultimately, we are planning to make this process accessible to a wider audience of educators and language learners. In general, we hope to see more works in technology-driven language education, and more apps implementing innovative approaches to facilitate second language acquisition.

We also want to emphasize that despite obvious shortcomings, the use of automated natural language processing is perhaps among few realistic ways to create language learning apps for less commonly taught languages. Creating high-quality educational materials takes substantial efforts, so there is no surprise that

they are primarily focused on languages with high demand and wide potential user base.

Current experimental versions of Irish WordBricks with NLP server and with preconfigured exercises are available on Google Play:

<https://play.google.com/store/apps/details?id=mprain.wordbricks>

<https://play.google.com/store/apps/details?id=mprain.basicenglishgrammar>

The source code repository is located at

<https://repo.rt247a.ddns.me/MobileWB>

Bibliography

- [1] D. Khampusaen, “Past, Present and Future: From Traditional Language Laboratories to Digital Language Laboratories and Multimedia ICT Suites,” in *Tenth International Conference on eLearning for Knowledge-Based Society*, 2013, pp. 12–13.
- [2] W. Decoo, “On the mortality of language learning methods,” *L. Barker Lecture*, 2001.
- [3] Z. H. Morford, B. N. Witts, K. J. Killingsworth, and M. P. Alavosius, “Gamification: the intersection between behavior analysis and game design technologies,” *The Behavior Analyst*, vol. 37, no. 1, pp. 25–40, 2014.
- [4] AdMob, *Six Essential Tips for App Developers*. Available: <https://www.thinkwithgoogle.com/advertising-channels/apps/six-essential-tips-for-app-developers>.
- [5] A.-M. Lavandier, *Mobile Gaming is Huge...and it's Staying: Rise of the Casual Gamer*. Available: <https://medium.com/the-nerd-castle/mobile-gaming-is-huge-and-its-staying-rise-of-the-casual-gamer-12a07333df66>.
- [6] S. Deterding, D. Dixon, R. Khaled, and L. Nacke, “From game design elements to gamefulness: defining gamification,” in *Proceedings of the 15th international academic MindTrek conference: Envisioning future media environments*, 2011, pp. 9–15.
- [7] S. Chen, *Facing Edutainment's Dark Legacy*. Available: <http://www.gamesandlearning.org/2016/01/25/facing-edutainments-dark-legacy/>.
- [8] P. Sweetser and P. Wyeth, “GameFlow: a model for evaluating player enjoyment in games,” *Computers in Entertainment (CIE)*, vol. 3, no. 3, p. 3, 2005.
- [9] Y. Long and V. Aleven, “Gamification of joint student/system control over problem selection in a linear equation tutor,” in *International Conference on Intelligent Tutoring Systems*, 2014, pp. 378–387.
- [10] J. Dolonen and A. Kluge, “Algebra Learning through Digital Gaming in School,” in *11th International Conference on Computer Supported Collaborative Learning*, 2015, pp. 252–259.
- [11] I. Bogost, *Persuasive Games: Exploitationware*. Available: https://www.gamasutra.com/view/feature/134735/persuasive_games_exploitationware.php.
- [12] M. Robertson, *Can't Play, Won't Play*. Available: <http://www.hideandseek.net/2010/10/06/cant-play-wont-play>.
- [13] E. Farmer, J. van Rooij, J. Riemersma, and P. Jorna, *Handbook of simulator-based training*: Routledge, 2017.

- [14] M. Levy, *Computer-assisted language learning: Context and conceptualization*. Oxford [u.a.]: Clarendon Press, 1997.
- [15] D. Chun, R. Kern, and B. Smith, “Technology in language use, language teaching, and language learning,” *The Modern Language Journal*, vol. 100, no. S1, pp. 64–80, 2016.
- [16] P. Hubbard, *Survey of unanswered questions in Computer Assisted Language Learning*. Stanford University. Available: <http://www.stanford.edu/~efs/callsurvey/index.html>.
- [17] V. Santos, “Rosetta Stone Portuguese (Brazil) levels 1, 2, & 3 Personal Edition Version 4 (TOTALe),” *Calico Journal*, vol. 29, no. 1, pp. 177–194, 2011.
- [18] B. Lewis, *Review of Rosetta Stone: Detailed and honest look at latest version (TOTALe)*. Available: <https://www.fluentin3months.com/rosetta-stone-review/>.
- [19] L. Amaral, D. Meurers, and R. Ziai, “Analyzing learner language: towards a flexible natural language processing architecture for intelligent language tutors,” *Computer Assisted Language Learning*, vol. 24, no. 1, pp. 1–16, 2011.
- [20] J. W. Heisig, *Remembering the kanji*. Honolulu: University of Hawai'i Press, 2011.
- [21] G. H. Gamage, “Perceptions of kanji learning strategies,” *Australian Review of Applied Linguistics*, vol. 26, no. 2, pp. 17–30, 2003.
- [22] M. K. Foti and J. Mendez, “Mobile learning: how students use mobile devices to support learning,” *Journal of Literacy and Technology*, vol. 15, no. 3, pp. 58–78, 2014.
- [23] J. Gikas and M. M. Grant, “Mobile computing devices in higher education: Student perspectives on learning with cellphones, smartphones & social media,” *The Internet and Higher Education*, vol. 19, pp. 18–26, 2013.
- [24] R. Draycott, *Gamification is the key to Duolingo success says product manager Gilani at Canvas conference*. Available: <http://www.thedrum.com/news/2017/10/26/gamification-the-key-duolingo-success-says-product-manager-gilani-canvas-conference>.
- [25] D. R. Bogdan, “Duolingo as an “Aid” to Second-language Learning. An Individual Case Study,” *愛媛大学教育学部紀要*, vol. 63, pp. 199–212, 2016.
- [26] S. Kumar, *My Gamified Language Learning Experience With Duolingo*. Available: <https://elearningindustry.com/duolingo-gamified-language-learning>.
- [27] de Castro, Ana Paula, da Hora Macedo, Suzana, and Bastos, Hélvia Pereira Pinto, “Duolingo: an Experience in English Teaching,” *Journal of Educational & Instructional Studies in the World*, vol. 6, no. 4, 2016.

- [28] D. Huynh and H. Iida, "An Analysis of Winning Streak's Effects in Language Course of "Duolingo"," *Asia-Pacific Journal of Information Technology and Multimedia*, vol. 6, no. 2, 2017.
- [29] A. Murdoch, *Duolingo Review: The Quick, Easy and Free Way to Learn A Language*. Available: <https://www.fluentin3months.com/duolingo/>.
- [30] S. Librenjak, K. Vučković, and Z. Dovedan, "Multimedia assisted learning of Japanese kanji characters," in *MIPRO, 2012 Proceedings of the 35th International Convention*, 2012, pp. 1284–1289.
- [31] R. C. Bailey and J. Davey, "Internet-based spaced repetition learning in and out of the classroom: Implementation and student perception," *CELE Journal*, vol. 20, pp. 39–50, 2011.
- [32] N. Walker, *Hacking the Kanji: 2,200 Kanji in 97 Days*. Available: <https://nihongoshark.com/learn-kanji/>.
- [33] E. Kidd, *Learning Ancient Egyptian in an Hour Per Week with Beeminder*. Available: <https://blog.beeminder.com/hieroglyphs/>.
- [34] G. H. Teninbaum, "Spaced Repetition: A Method for Learning More Law in less Time," *J. High Tech. L.*, vol. 17, p. 273, 2016.
- [35] J. Bitchener, "Evidence in support of written corrective feedback," *Journal of second language writing*, vol. 17, no. 2, pp. 102–118, 2008.
- [36] J. Truscott, "The case against grammar correction in L2 writing classes," *Language learning*, vol. 46, no. 2, pp. 327–369, 1996.
- [37] W. Christian *et al*, *Open Source Physics*. <http://www.opensourcephysics.org>, 2013.
- [38] D. Yaron, C. Ashe, M. Karabinos, K. Williams, and L. Ju, *ChemCollective*. <http://www.chemcollective.org>, 2013.
- [39] N. Nagata, "Robo-Sensei's NLP-based error detection and feedback generation," *Calico Journal*, vol. 26, no. 3, pp. 562–579, 2009.
- [40] B. Azar and S. Hagen, *Basic English Grammar, 3rd Ed*: Pearson Longman, 2005.
- [41] R. Murphy, *English Grammar in Use, 4th Ed*: Cambridge University Press, 2012.
- [42] S. D. Krashen, *Explorations in language acquisition and use*: Heinemann Portsmouth, NH.
- [43] M. Resnick *et al*, "Scratch: Programming for All," *Commun. ACM*, vol. 52, no. 11, pp. 60–67, 2009.
- [44] D. Gooding, *Experiment and the making of meaning: Human agency in scientific observation and experiment*: Kluwer Academic Publishers Dordrecht, 1990.

- [45] S. Ebbels, “Teaching grammar to school-aged children with specific language impairment using shape coding,” *Child Language Teaching and Therapy*, vol. 23, no. 1, pp. 67–93, 2007.
- [46] R. Debusmann, “An introduction to dependency grammar,” *Hausarbeit für das Hauptseminar Dependenzgrammatik SoSe*, vol. 99, pp. 1–16, 2000.
- [47] M.-C. de Marneffe and C. D. Manning, “Stanford typed dependencies manual,” *Stanford University*, 2008.
- [48] M. Mozgovoy and R. Efimov, “WordBricks: a virtual language lab inspired by Scratch environment and dependency grammars,” *Human-centric Computing and Information Sciences*, vol. 3, no. 1, pp. 1–9, 2013.
- [49] J. Havelka, “Beyond projectivity: Multilingual evaluation of constraints and measures on non-projective structures,” in *45th Annual Meeting of the Association of Computational Linguistics*, 2007, pp. 608–615.
- [50] M. Purgina, M. Mozgovoy, and V. Klyuev, “Developing a Mobile System for Natural Language Grammar Acquisition,” in *Dependable, Autonomic and Secure Computing, 14th Intl Conf on Pervasive Intelligence and Computing, 2nd Intl Conf on Big Data Intelligence and Computing and Cyber Science and Technology Congress (DASC/PiCom/DataCom/CyberSciTech), 2016 IEEE 14th Intl C*, 2016, pp. 322–325.
- [51] M. Purgina and M. Mozgovoy, “Visualizing Sentence Parse Trees with WordBricks,” in *Cybernetics (CYBCONF), 2017 3rd IEEE International Conference on*, 2017, pp. 1–4.
- [52] M. Park, M. Purgina, and M. Mozgovoy, “Learning English Grammar with WordBricks: Classroom Experience,” in *Proceedings of the 2016 IEEE International Conference on Teaching and Learning in Education*, 2016.
- [53] P. J.-H. Hu, T. H. K. Clark, and W. W. Ma, “Examining technology acceptance by school teachers: a longitudinal study,” *Information & management*, vol. 41, no. 2, pp. 227–241, 2003.
- [54] D. K. Farkas, “Explicit structure in print and on-screen documents,” *Technical communication quarterly*, vol. 14, no. 1, pp. 9–30, 2005.
- [55] S. Washio and Y. Watanabe, “Security of audio secret sharing scheme encrypting audio secrets with bounded shares,” in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*, 2014, pp. 7396–7400.
- [56] M. Purgina, M. Mozgovoy, and M. Ward, “Learning Language Grammar with Interactive Exercises in the Classroom and Beyond,” *Proceedings of the 9th International Conference on Computer Supported Education*, 2017.
- [57] Government of Ireland, *Statement on the Irish language*, 2006.
- [58] I. Watson, “Irish language and identity,” in *A New View of the Irish Language*, C. Nic Pháidín and S. Ó Cearnaigh, Eds.: Cois Life, 2008, pp. 66–75.

- [59] M. Darmody and T. Daly, “Attitudes towards the Irish Language on the Island of Ireland,” *The Economic and Social Research Institute*, 2015.
- [60] R. Webb, *80/20 Japanese*, 2016.
- [61] S. Kahane, “If HPSG were a dependency grammar...,” *Actes de TALN*, pp. 22–24, 1996.
- [62] M. Hewings, *Advanced grammar in use: A self-study reference and practice book for advanced learners of English : with answers*, 3rd ed. Cambridge, New York: Cambridge University Press, 2013.
- [63] J. Guo, *Google’s new artificial intelligence can’t understand these sentences. Can you?* Available: <https://www.washingtonpost.com/news/wonk/wp/2016/05/18/googles-new-artificial-intelligence-cant-understand-these-sentences-can-you>.
- [64] J. Smith, “WPF Apps With The Model-View-ViewModel Design Pattern,” *MSDN Magazine*, February 2009.
- [65] Google Inc. and the Open Handset Alliance, *API Guides: UI Overview*. Available: <http://developer.android.com/guide/topics/ui/overview.html>, 2016.
- [66] Google Inc, Material Design Specification. Patterns – Navigation. Available: <https://www.google.com/design/spec/patterns/navigation.html>, 2016.
- [67] B. Santorini, “Part-of-speech tagging guidelines for the Penn Treebank Project (3rd revision),” *Technical Reports (CIS), Paper 570, University of Pennsylvania, Department of Computer and Information Science*, 1990.
- [68] Z. Le, “Maximum entropy modeling toolkit for Python and C++,” *Natural Language Processing Lab, Northeastern University, China*, 2004.
- [69] A. L. Berger, V. J. Della Pietra, and S. A. Della Pietra, “A maximum entropy approach to natural language processing,” *Computational linguistics*, vol. 22, no. 1, pp. 39–71, 1996.
- [70] N. Ide, C. Baker, C. Fellbaum, and C. Fillmore, “MASC: The manually annotated sub-corpus of American English,” in *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC)*, 2008.
- [71] J. Nivre, CoNLL-U Format. Available: <http://universaldependencies.org/format.html>.
- [72] P. Jian and C. Zong, “Layer-Based Dependency Parsing,” in *PACLIC*, pp. 230–239, 2009.
- [73] M. P. Marcus, M. A. Marcinkiewicz, and B. Santorini, “Building a large annotated corpus of English: The Penn Treebank,” *Computational linguistics*, vol. 19, no. 2, pp. 313–330, 1993.
- [74] R. Johansson and P. Nugues, “Extended Constituent-to-dependency Conversion for English”, in *Proceedings of NODALIDA 2007*. Tartu, Estonia, pp. 105-112, 2007.

- [75] F. Salazar and M. Brambilla, "Tailoring Software Architecture Concepts and Process for Mobile Application Development," in *Proceedings of the 3rd International Workshop on Software Development Lifecycle for Mobile*, New York, NY, USA: ACM, 2015, pp. 21–24.
- [76] B. A. Lucini, T. Hatt, C. Gardner, and B. Pon, "Mobile platform wars", *GSMA Intelligence*, 2014.
- [77] S. Bird & E. Loper, "NLTK: the natural language toolkit", in *Proceedings of the ACL 2004 on Interactive poster and demonstration sessions*, p. 31., 2004.
- [78] T. Lynn, "Irish dependency treebanking and parsing", *Sydney, Australia, Macquarie University*, 2016.
- [79] T. Lynn & J. Foster, "Universal dependencies for Irish", in *Celtic Language Technology Workshop*, pp. 79–92, 2016.
- [80] M. Collins, "Discriminative training methods for hidden markov models: Theory and experiments with perceptron algorithms", in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pp. 1–8, 2002.
- [81] J. Nivre, J. Hall & J. Nilsson, "Maltparser: A data-driven parser-generator for dependency parsing", in *Proceedings of LREC*, vol 6, pp. 2216–2219, 2006.
- [82] D. Larsen-Freeman, "Grammar: Rules and Reasons Working Together", *ESL Magazine*, 3(1), 10–12, 2000.
- [83] D. Larsen-Freeman, "Research into practice: Grammar learning and teaching", *Language Teaching*, 48(2), 263–280, 2015.
- [84] G. Jean & D. Simard, "Grammar learning in English and French L2: Students' and teachers' beliefs and perceptions", *Foreign Language Annals*, 44(4), 465–492, 2011.
- [85] M. H. Long, "Focus on form: A design feature in language teaching methodology", *Foreign language research in cross-cultural perspective*, 2(1), 39–52, 1991.
- [86] N. Shintani, S. Li, & R. Ellis, "Comprehension-based versus production-based grammar instruction: A meta-analysis of comparative studies", *Language learning*, 63(2), 296–329, 2013.
- [87] T. Pica, "Classroom learning, teaching, and research: A task-based perspective", *The Modern Language Journal*, 89(3), 339–352, 2005.
- [88] M. Ward, M. Mozgovoy & M. Purgina, "Can Word Bricks Make Learning Irish More Engaging For Students?", *International Journal of Game-Based Learning*, in press, 2018.
- [89] M. Purgina, M. Mozgovoy, J. Blake, "WordBricks: Mobile Technology and Visual Grammar Formalism for Gamification of Natural Language Grammar Acquisition", *Journal of Educational Computing Research*, in press, 2019.

Appendix A. Sample Irish Exercise

Description: Irish uses *bí* + noun + prep + (optional article) + noun to state where something is located. There are several different words used include ‘ar’ (on), ‘sa’ (inside), ‘in aice leis an’ (beside). The task is to drill such constructions in interrogative sentences.

Pattern	Example
to-be subject prep-phrase.	Tá hata ar an mbord. Is hat on the table. Tá hata sa bhosca. Is hat in the box. Tá hata in aice leis an gcathaoir. Is hat beside the door.

Bricks Required:

Group: <i>bí</i>, POS: verb
<i>Word:</i> Tá, Bhí, Níl, Ní raibh
<i>Post-conn 1:</i> noun or pronoun with any attributes
<i>Post-conn 2:</i> prep-phrase with any attributes

POS: noun
<i>Word:</i> caipín, laithróid, leabhar, rothar, madra

Group: prep, POS: prep-phrase
<i>Word:</i> ar an
<i>Post-conn:</i> noun with eclipsis
<i>Word:</i> sa
<i>Post-conn:</i> noun with lenition
<i>Word:</i> in aice leis an
<i>Post-conn:</i> noun with eclipsis

Group: noun with eclipsis

Word: urlár, mbord, teilifís, mballa, tolg, mbosca, gcathaoir, gclog, mála, bhfuinneog, gclár dubh

Group: noun with lenition

Word: seomra, chúinne, phictiúr, bhosca, chupán, mhála, seomra suí

Exercises: write sentences like “*Tá, leabhar, ar an, mbord*”.

XML file:

```
<?xml version="1.0" encoding="utf-8" ?>
<task rule = "Example 1">
  <exercise>
    <!-- An bhfuil hata ag an madra? -->
    <unit brick = "An bhfuil with pron-phrase"/>
    <unit brick = "hata"/>
    <unit brick = "ag with noun"/>
    <unit brick = "an"/>
    <unit brick = "madra"/>
    <unit brick = "?"/>
  </exercise>
  <type neme = "Determiner">
    <brick name="an" type="Determiner" used_with = "singular">
      <item type="text" value="an"/>
    </brick>
    <brick name="na" type="Determiner" used_with = "plural">
      <item type="text" value="na"/>
    </brick>
    <brick name="∅" type="Determiner" used_with_1 = "singular"
      used_with_2 = "plural" used_with_3 = "uncountable">
      <item type="text" value=""/>
    </brick>
  </type>
  <type neme = "Sentence">
    <brick name="?" type="Sentence">
      <item type="brickConnector" value="Verb phrase"/>
      <item type="text" value=" ?"/>
    </brick>
  </type>
  <type neme = "Verb">
    <brick name="An bhfuil with pron-phrase" type="Verb phrase"
      form = "An bhfuil">
      <item type="text" value="An bhfuil"/>
      <item type="brickConnector" value="Noun phrase"
        case_1="nominative" case_2="common">
```

```

        person = "third" number_1="singular" number_2="plural"
mutation = "unchanged" />
    <item type="brickConnector" value = "Pron phrase"/>
</brick>
<brick name="An bhfuil with prep-phrase" type="Verb phrase"
    form = "An bhfuil">
    <item type="text" value="An bhfuil"/>
    <item type="brickConnector" value="Noun phrase"
        case_1="nominative" case_2="common"
        person = "third" number_1="singular" number_2="plural"
        mutation = "unchanged" />
    <item type="brickConnector" value = "Prepositional phrase"/>
</brick>

<brick name="An raibh with pron-phrase" type="Verb phrase"
    form = "An raibh">
    <item type="text" value="An raibh"/>
    <item type="brickConnector" value="Noun phrase"
        case_1="nominative" case_2="common"
        person = "third" number_1="singular" number_2="plural"
        mutation = "unchanged" />
    <item type="brickConnector" value = "Pron phrase"/>
</brick>
<brick name="An raibh with prep-phrase" type="Verb phrase"
    form = "An raibh">
    <item type="text" value="An raibh"/>
    <item type="brickConnector" value="Noun phrase"
        case_1="nominative" case_2="common"
        person = "third" number_1="singular" number_2="plural"
        mutation = "unchanged" />
    <item type="brickConnector" value = "Prepositional phrase"/>
</brick>
</type>
<type neme="Names">
    <brick name="Seán" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "unchanged" >
        <item type="text" value="Seán"/>
    </brick>
    <brick name="Áine" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "unchanged" >
        <item type="text" value="Áine"/>
    </brick>
    <brick name="Liam" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "unchanged" >
        <item type="text" value="Liam"/>
    </brick>
    <brick name="Máire" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "unchanged" >
        <item type="text" value="Máire"/>
    </brick>
    <brick name="Róisín" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "unchanged" >
        <item type="text" value="Róisín"/>
    </brick>
    <brick name="Mamáí" type="Noun phrase"
        case = "common" person = "third" number = "singular"

```

```

        mutation = "unchanged" >
        <item type="text" value="Mamaí"/>
</brick>
<brick name="Daidí" type="Noun phrase"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged" >
    <item type="text" value="Daidí"/>
</brick>

</type>
<type neme = "Verbal Noun">
    <brick name="rith" type="Verbal Noun">
        <item type="text" value="rith"/>
    </brick>
    <brick name="ithe" type="Verbal Noun">
        <item type="text" value="ithe"/>
    </brick>
    <brick name="ól" type="Verbal Noun">
        <item type="text" value="ól"/>
    </brick>
    <brick name="siúl" type="Verbal Noun">
        <item type="text" value="siúl"/>
    </brick>
    <brick name="gáire" type="Verbal Noun">
        <item type="text" value="gáire"/>
    </brick>
    <brick name="léamh" type="Verbal Noun">
        <item type="text" value="léamh"/>
    </brick>
    <brick name="súgradh" type="Verbal Noun">
        <item type="text" value="súgradh"/>
    </brick>
    <brick name="téascáil" type="Verbal Noun">
        <item type="text" value="téascáil"/>
    </brick>
    <brick name="rothaíocht" type="Verbal Noun">
        <item type="text" value="rothaíocht"/>
    </brick>
    <brick name="scátáil" type="Verbal Noun">
        <item type="text" value="scátáil"/>
    </brick>
</type>
<type neme = "Pron-phrase">
    <brick name="ag with noun" type="Pron phrase" form = "ag">
        <item type="text" value="ag"/>
        <item type="brickConnector" value="Noun phrase"
            case_1="nominative" case_2="common"
            person = "third" number_1="singular" number_2="plural"
            mutation = "unchanged" />
    </brick>
    <brick name="ag with verbal noun" type="Pron phrase" form =
"ag">
        <item type="text" value="ag"/>
        <item type="brickConnector" value="Verbal Noun"/>
    </brick>
    <brick name="ar - pron" type="Pron phrase" form = "ar">
        <item type="text" value="ar"/>
        <item type="brickConnector" value="Noun phrase"
            case_1="nominative" case_2="common"
            person = "third" number_1="singular" number_2="plural"
            mutation = "unchanged" />

```

```

</brick>
<brick name="orm" type="Pron phrase" form = "ar">
  <item type="text" value="orm"/>
</brick>
<brick name="ort" type="Pron phrase" form = "ar">
  <item type="text" value="ort"/>
</brick>
<brick name="air" type="Pron phrase" form = "ar">
  <item type="text" value="air"/>
</brick>
<brick name="urithi" type="Pron phrase" form = "ar">
  <item type="text" value="urithi"/>
</brick>
</type>
<type neme = "Preposition">
  <brick name="ar - preposition" type="Prepositional phrase">
    <item type="text" value="ar"/>
    <item type="brickConnector" value="Noun phrase"
      case_1="oblique" case_2="common"
      person_1 = "first" person_2 = "second" person_3 =
"third"
      number_1="plural" number_2="singular"
      mutation = "eclipsis"/>
  </brick>
  <brick name="sa" type="Prepositional phrase">
    <item type="text" value="sa"/>
    <item type="brickConnector" value="Noun phrase"
      case_1="oblique" case_2="common"
      person_1 = "first" person_2 = "second" person_3 =
"third"
      number_1="plural" number_2="singular"
      mutation = "lenition"/>
  </brick>
  <brick name="in aice leis" type="Prepositional phrase">
    <item type="text" value="in aice leis"/>
    <item type="brickConnector" value="Noun phrase"
      case_1="oblique" case_2="common"
      person_1 = "first" person_2 = "second" person_3 =
"third"
      number_1="plural" number_2="singular"
      mutation = "eclipsis"/>
  </brick>
</type>
<type neme = "Noun">
  <!-- Mutation: unchanged -->
  <brick name="ocras" type="Noun phrase" form = "ocras"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged">
    <item type="text" value="ocras"/>
  </brick>
  <brick name="eagla" type="Noun phrase" form = "eagla"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged">
    <item type="text" value="eagla"/>
  </brick>
  <brick name="fearg" type="Noun phrase" form = "fearg"
    case = "common" person = "third" number = "singular"

```

```

    mutation = "unchanged">
    <item type="text" value="fearg"/>
  </brick>
  <brick name="áthas" type="Noun phrase" form = "áthas"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged">
    <item type="text" value="áthas"/>
  </brick>
  <brick name="brón" type="Noun phrase" form = "brón"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged">
    <item type="text" value="brón"/>
  </brick>
  <brick name="hata" type="Noun phrase" form = "hata"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged">
    <item type="text" value="hata"/>
  </brick>
  <brick name="hataí" type="Noun phrase" form = "hata"
    case = "common" person = "third" number = "plural"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "plural"/>
    <item type="text" value="hataí"/>
  </brick>
  <brick name="cóta" type="Noun phrase" form = "cóta"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "singular"/>
    <item type="text" value="cóta"/>
  </brick>
  <brick name="mála" type="Noun phrase" form = "mála"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "singular"/>
    <item type="text" value="mála"/>
  </brick>
  <brick name="ubh" type="Noun phrase" form = "ubh"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "singular"/>
    <item type="text" value="ubh"/>
  </brick>
  <brick name="bláth" type="Noun phrase" form = "bláth"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "singular"/>
    <item type="text" value="bláth"/>
  </brick>
  <brick name="bláthanna" type="Noun phrase" form = "bláth"
    case = "common" person = "third" number = "plural"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "plural"/>
    <item type="text" value="bláthanna"/>
  </brick>
  <brick name="scannáin" type="Noun phrase" form = "scannáin"

```

```

    case = "common" person = "third" number = "plural"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "plural"/>
    <item type="text" value="scannáin"/>
  </brick>
  <brick name="cartúin" type="Noun phrase" form = "cartúin"
    case = "common" person = "third" number = "plural"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "plural"/>
    <item type="text" value="cartúin"/>
  </brick>
  <brick name="bréagáin" type="Noun phrase" form = "bréagáin"
    case = "common" person = "third" number = "plural"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "plural"/>
    <item type="text" value="bréagáin"/>
  </brick>
  <brick name="banana" type="Noun phrase" form = "banana"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "singular"/>
    <item type="text" value="banana"/>
  </brick>
  <brick name="bananaí" type="Noun phrase" form = "banana"
    case = "common" person = "third" number = "plural"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "plural"/>
    <item type="text" value="bananaí"/>
  </brick>
  <brick name="ceapairí" type="Noun phrase" form = "ceapairí"
    case = "common" person = "third" number = "plural"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "plural"/>
    <item type="text" value="ceapairí"/>
  </brick>
  <brick name="milseáin" type="Noun phrase" form = "milseáin"
    case = "common" person = "third" number = "plural"
    mutation = "unchanged">
    <item type="brickConnector" value="Determiner"
      used_with = "plural"/>
    <item type="text" value="milseáin"/>
  </brick>

  <brick name="caipín" type="Noun phrase"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged" >
    <item type="text" value="caipín"/>
  </brick>
  <brick name="laithróid" type="Noun phrase"
    case = "common" person = "third" number = "singular"
    mutation = "unchanged" >
    <item type="text" value="laithróid"/>
  </brick>
  <brick name="leabhar" type="Noun phrase"
    case = "common" person = "third" number = "singular"

```

```

        mutation = "unchanged" >
        <item type="text" value="leabhar"/>
</brick>
<brick name="rothar" type="Noun phrase"
  case = "common" person = "third" number = "singular"
  mutation = "unchanged" >
  <item type="text" value="rothar"/>
</brick>
<brick name="madra" type="Noun phrase"
  case = "common" person = "third" number = "singular"
  mutation = "unchanged" >
  <item type="brickConnector" value="Determiner"
    used_with = "singular"/>
  <item type="text" value="madra"/>
</brick>

<!-- Mutation: eclipsis -->
<brick name="urlár" type="Noun phrase"
  case = "common" person = "third" number = "singular"
  mutation = "eclipsis">
  <item type="brickConnector" value="Determiner"
    used_with = "singular"/>
  <item type="text" value="urlár"/>
</brick>
<brick name="mbord" type="Noun phrase"
  case = "common" person = "third" number = "singular"
  mutation = "eclipsis">
  <item type="brickConnector" value="Determiner"
    used_with = "singular"/>
  <item type="text" value="mbord"/>
</brick>
<brick name="teilifís" type="Noun phrase"
  case = "common" person = "third" number = "singular"
  mutation = "eclipsis">
  <item type="brickConnector" value="Determiner"
    used_with = "singular"/>
  <item type="text" value="teilifís"/>
</brick>
<brick name="mballa" type="Noun phrase"
  case = "common" person = "third" number = "singular"
  mutation = "eclipsis">
  <item type="brickConnector" value="Determiner"
    used_with = "singular"/>
  <item type="text" value="mballa"/>
</brick>
<brick name="tolg" type="Noun phrase"
  case = "common" person = "third" number = "singular"
  mutation = "eclipsis">
  <item type="brickConnector" value="Determiner"
    used_with = "singular"/>
  <item type="text" value="tolg"/>
</brick>
<brick name="mbosca" type="Noun phrase"
  case = "common" person = "third" number = "singular"
  mutation = "eclipsis">
  <item type="brickConnector" value="Determiner"
    used_with = "singular"/>
  <item type="text" value="mbosca"/>
</brick>
<brick name="gcathaoir" type="Noun phrase"
  case = "common" person = "third" number = "singular"

```



```

        mutation = "eclipsis">
        <item type="brickConnector" value="Determiner"
            used_with = "singular"/>
        <item type="text" value="gcathaoir"/>
    </brick>
    <brick name="gclog" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "eclipsis">
        <item type="brickConnector" value="Determiner"
            used_with = "singular"/>
        <item type="text" value="gclog"/>
    </brick>
    <brick name="mála" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "eclipsis">
        <item type="brickConnector" value="Determiner"
            used_with = "singular"/>
        <item type="text" value="mála"/>
    </brick>
    <brick name="bhfuinneog" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "eclipsis">
        <item type="brickConnector" value="Determiner"
            used_with = "singular"/>
        <item type="text" value="bhfuinneog"/>
    </brick>
    <brick name="gclárdubh" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "eclipsis">
        <item type="brickConnector" value="Determiner"
            used_with = "singular"/>
        <item type="text" value="gclárdubh"/>
    </brick>

    <!-- Mutation: lenition -->
    <brick name="seomra" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "lenition">
        <item type="text" value="seomra"/>
    </brick>
    <brick name="chúinne" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "lenition">
        <item type="text" value="chúinne"/>
    </brick>
    <brick name="phictiúr" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "lenition">
        <item type="text" value="phictiúr"/>
    </brick>
    <brick name="bhosca" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "lenition">
        <item type="text" value="bhosca"/>
    </brick>
    <brick name="chupán" type="Noun phrase"
        case = "common" person = "third" number = "singular"
        mutation = "lenition">
        <item type="text" value="chupán"/>
    </brick>
    <brick name="mhála" type="Noun phrase"

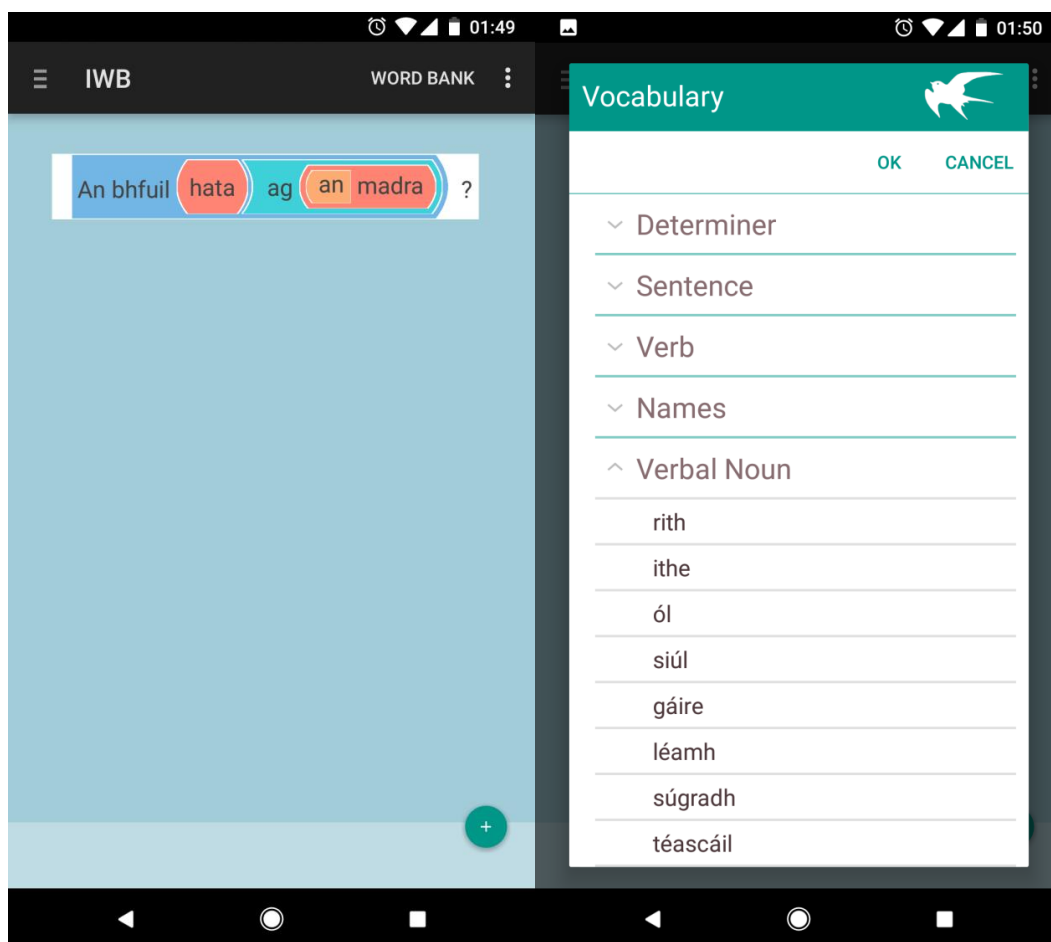
```

```

    case = "common" person = "third" number = "singular"
    mutation = "lenition">
    <item type="text" value="mhála"/>
  </brick>
  <brick name="seomra suí" type="Noun phrase"
    case = "common" person = "third" number = "singular"
    mutation = "lenition">
    <item type="text" value="seomra suí"/>
  </brick>
</type>
</task>

```

Exercises within WordBricks GUI:



Appendix B. Sample XML Descriptions for the Current WordBricks Version

HTTP Request: *“the little cat devoured a mouse”*

XML Result:

```
<?xml version="1.0" encoding="utf-8"?>
<wordbricks>
  <boc>
    <brick id="devoured_1" coords="0.37,0.1"/>
    <brick id="cat_1" parent="devoured_1" connector="1"/>
    <brick id="mouse_1" parent="devoured_1" connector="2"/>
    <brick id="the_1" parent="cat_1" connector="1"/>
    <brick id="little_1" parent="cat_1" connector="2"/>
    <brick id="a_1" parent="mouse_1" connector="1"/>
  </boc>
  <bdf>
    <brick id="cat_1" lemma="cat" type="Noun phrase"
      case = "common" person = "third" number = "singular">
      <item type="brickConnector" connector="1"
        value="Determiner"/>
      <item type="brickConnector" connector="2"
        value="Adjective phrase"/>
      <item type="word">cat</item>
    </brick>
    <type id="little_1" lemma="little" type = "Adjective">
      <brick name="little" type="Adjective phrase">
        <item type="word">little</item>
      </brick>
    </type>
    <brick id="mouse_1" lemma="mouse" type="Noun phrase"
      case = "common" person = "third" number = "singular">
      <item type="brickConnector" connector="1"
        value="Determiner"/>
      <item type="word">mouse</item>
    </brick>
    <brick id="devoured_1" lemma="devoured" type="Verb phrase">
      <item type="brickConnector" connector="1"
        value="Noun phrase">
        <attrs case="common" person = "third"
          number="singular"/>
        <attrs case="nominative" person = "third"
          number="singular"/>
      </item>
    </brick>
  </bdf>
</wordbricks>
```

```

    </item>
    <item type="word">devoured</item>
    <item type="brickConnector" connector="2"
      value="Noun phrase">
      <attrs
        case="common" person = "third" number="singular"/>
      <attrs
        case="oblique" person = "third" number="singular"/>
      <attrs
        case="common" person = "second" number="singular"/>
      <attrs
        case="oblique" person = "second" number="singular"/>
      <attrs
        case="common" person = "third" number="plural"/>
      <attrs
        case="oblique" person = "third" number="plural"/>
      <attrs
        case="common" person = "second" number="plural"/>
      <attrs
        case="oblique" person = "second" number="plural"/>
    </item>
  </brick>

  <brick id="a_1" lemma="a" type="Determiner">
    <item type="word">a</item>
  </brick>

  <brick id="the_1" lemma="the" type="Determiner">
    <item type="word">the</item>
  </brick>

</bdf>

<bcs>
  <brickshape shape="shape 1" color="red">
    <attrs>
      <attr>Noun phrase</attr>
    </attrs>
  </brickshape>

  <brickshape shape="shape 2" color="yellow">
    <attrs>
      <attr>Adjective</attr>
    </attrs>
  </brickshape>

  <brickshape shape="shape 3" color="blue">
    <attrs>
      <attr>Verb phrase</attr>
    </attrs>
  </brickshape>

  <brickshape shape="shape 4" color="orange">
    <attrs>
      <attr>Determiner</attr>
    </attrs>
  </brickshape>
</bcs>
</wordbricks>

```