Generating and authoring high-variability exercises from authentic texts

Tanja Heck
Universität Tübingen / Germany
tanja.heck@
uni-tuebingen.de

Detmar Meurers
Universität Tübingen / Germany
detmar.meurers@
uni-tuebingen.de

Abstract

Integrating adaptivity into Task-Based Language Teaching requires exercises that transmit a specific content but whose complexity is adjusted to the learner's level. Thus, exercises of varying complexity based on the same text are needed. Revising generated exercise variants is time consuming and redundant where the same underlying linguistic annotations can be used for exercise generation. We present a fully implemented approach to generate generalized exercise specifications as an interim step before turning them into concrete exercises, as well as an interface for efficient reviewing of the specifications.

1 Introduction

For Computer-Assisted Language Learning (CALL), Task-Based Language Teaching (TBLT) can serve as a well-motivated, current pedagogical framework (Lai and Li, 2011). Putting a premium on the functional use of language with a focus on meaning, the TBLT perspective can offer a less monotonous learning experience than traditional grammar-focused instruction with decontextualized exercises (Doughty and Long, 2003). However, creating complex learning cycles with functional final tasks preceded by step-wise pre-task activities supporting practice of the task-essential language aspects requires considerable human effort. Form-based exercises, on the other hand, can be generated automatically in rule-based approaches or from authentic texts (Perez-Beltrachini et al., 2012).

Pursuing a kind of hybrid approach, Li et al. (2016) found that Task-Supported Language Teaching (TSLT), where working on a task follows explicit instruction, yielded better learning outcomes for grammar topics targeted in a cycle. Following a Presentation-Practice-Production (PPP) Model as backbone (Ur, 2018), TSLT explicitly teaches new concepts in the Presentation phase, uses traditional form-focused exercises in the Practice phase and more meaning-focused practice in the final task of the Production phase. In order to best support scaffolded learning preparing students for the Production task, the exercises in the Practice phase should preferably cover vocabulary and grammar topics relevant to that task.

The limited time available to teachers is not only an issue for the compilation of teaching materials, but also for taking into account the individual needs for additional support or practice (Aftab, 2015). Intelligent CALL systems can overcome this lack of differentiation through micro- and macro-adaptivity (Rus et al., 2015). Micro-adaptivity supports learners through scaffolded feedback when necessary. Macro-adaptivity adaptively selects and sequences exercises in the student's Zone of Proximal Development. The exercises thus provide practice opportunities for linguistic constructs where a learner struggles but can successfully complete the activity (when scaffolded). In TSLT, approaches to macro-adaptivity are especially valuable in the Practice phase in order to achieve effective and efficient proceduralization of language knowledge.

Macro-adaptivity usually relies on large pools of exercises in order to cover the vast space of possible ability levels a student can have across a range of linguistic constructs (Katinskaia et al., 2018). Since manual compilation of the required number of exercises is not feasible, automatic generation of exercises for the Practice phase become not only possible but necessary. While automatically generating exercises from authentic texts has been explored in various systems, they lack a systematic approach to generating large sets of exercises of varying complexity from source texts. In addition, proceduralization of linguistic knowledge requires exposure in a variety of contexts such as different syntactic structures, questions, or negation. Adaptive sequencing must therefore rely

Tanja Heck and Detmar Meurers. Generating and authoring high-variability exercises from authentic texts.
Proceedings of the 11th Workshop on Natural Language Processing for Computer Assisted Language Learning (NLP4CALL 2022). Linköping Electronic Conference Proceedings 190: 61–71.

on analyzing linguistic structures and differences in complexity of the source texts in order to provide the required variability and serve the needs of all students (Pandarova et al., 2019). This, however, does not allow instructors to also practice specific vocabulary or content at the same time.

Focusing on beginning to intermediate learners of English, the approach suggested by Heck and Meurers (2022a) fills this gap by systematically parameterizing exercises so that a single specification based on one sentence can be used to generate a range of exercises at varying levels of complexity. The approach, however, requires manually written specifications. While being more efficient than creating each exercise individually, the specifications still need to be composed manually, with the additional drawback of lacking intrinsically motivating authenticity (Peacock, 1997). We overcome this limitation by automatically generating the exercise specifications from authentic texts. Since this process might introduce errors, the generated specifications need to be reviewed and possibly revised. When conducting revisions at this stage of the exercise generation process, one only needs to check a single abstract specification instead of dozens of spelled out exercises. However, since each specification contains exercise elements relevant to a range of different exercise types, there is no readily-available authoring interface. We therefore introduce a prototype for a web-based interface serving this purpose.

In this paper, section 2 first reviews existing approaches to exercise generation in terms of their potential support for macro-adaptive systems. Section 3 describes the implementation of our approach with a focus on the user's interaction with the system throughout the exercise generation workflow. Section 4 evaluates the implementation before section 5 summarizes and concludes with an outlook.

2 Related Work

Addressing the shortcomings of prefabricated language material generally used in text-books, Authentic Intelligent CALL focuses on using authentic texts in language learning (Meurers, 2020). In particular, automatically generating grammar exercises from authentic texts has received considerable attention in the past as a means to meet the demand for practice material in Intelligent Language Tutoring Systems (ILTS) (Malafeev, 2015). Closed activity types such as Multiple Choice (MC) are especially popular due to their ability to automatically score the exercises based on the very restricted space of possible learner answers (Tafazoli et al., 2019), yet supported exercise formats vary from one system to the other. A number of tools integrate a variety of different formats: *MIRTO* automatically generates Fill-in-the-Blanks (FiB) as well as Mark-the-Words (MtW) exercises (Antoniadis et al., 2004); *Arik-Iturri* can generate MC, Error Detection, FiB and Word Formation exercises (Aldabe et al., 2006); an extension of the language aware search Engine *FLAIR*[1] (Heck and Meurers, 2022b) covers a wide range including FiB, MC, MtW, Memory, Jumbled Sentences and Drag and Drop exercises; *Sakumon* (Hoshino and Nakagawa, 2008) and *Cloze-Fox* (Jozef and Sevinc, 2010) support cloze exercises in FiB as well as MC format; *WERTi* (Meurers et al., 2010) and its multilingual extension *View* (Reynolds et al., 2014) in addition feature MtW exercises, the *Language Exercise App* Sentence Shuffling activities (Pérez and Cuadros, 2017), and Ferreira and Pereira Jr. (2018)'s *Verb Tenses System* True/False and Tense transposition exercises. While these systems can generate multiple exercises for a linguistic structure from the same source document, the actual number of exercises is usually quite limited. By varying exercise parameters such as the number of distractors, hints in parentheses, or the span of the target construction, variability can be increased. Notable examples making use of such parameterizations constitute MIRTO which provides parameters for the choice of target constructions, parentheses of FiB exercises and support elements such as reference pages (Antoniadis et al., 2004); the assistant system Sakumon which requires users to manually select target items and distractors from automatically generated suggestions (Hoshino and Nakagawa, 2008); the Language Exercise App where target constructions, distractors and parentheses of FiB exercises are parameterizable (Pérez and Cuadros, 2017); and FLAIR's exercise generation functionality which, in addition to providing parameters for target constructions, distractors and parentheses, allows users to influence the specificity of the exercise instructions (Heck and Meurers, 2022b). However, these systems require users

[1] http://sifnos.sfs.uni-tuebingen.de/FLAIR/

to specify each configuration individually so that generating large numbers of parameterized exercises involves considerable configuration effort as well as manual labour to review the generated exercises for correctness.

Many exercise generation tools provide support to post-edit the generated exercises, either within the tool (e.g., Toole and Heift, 2001; Hoshino and Nakagawa, 2008) or by providing an interface to general-purpose authoring interfaces such as Hot Potatoes[2] or the LMS Moodle[3] (e.g., Bick, 2000; Aldabe et al., 2006; Pérez and Cuadros, 2017). These interfaces are, however, designed to edit a single exercise at a time. Modifications of elements which affect all exercises generated from the same document thus have to be performed on each exercise individually.

There is a clear gap to generate large numbers of exercises from a document with different parameterizations as well as to allow for efficient editing of the generated exercises. We build on Heck and Meurers (2022a)'s approach to high-variability exercise generation by defining abstract exercise specifications as an intermediate step towards exercise generation. Our suggested approach generates specifications for conditionals and relative clauses automatically from authentic texts and provides an authoring interface for the specifications which allows to modify properties of all exercises generated from the same specification in a single step.

3 Implementation

As illustrated by the system architecture design in Figure 1, the implementation consists of three steps in-between which users are presented the interim results and can modify them if they wish to do so. This allows for maximally efficient user interactions as they can be performed on the most condensed representation layer containing the information to edit. The back-end code is implemented in a microservice architecture which supports flexible use of programming languages, thus facilitating the use of best-performing libraries across multiple programming languages.

The front-end implementation is still in its prototype state. It uses HTML, CSS and JavaScript, relying on Ajax for communication with the server.

[2] https://hotpot.uvic.ca
[3] https://moodle.org

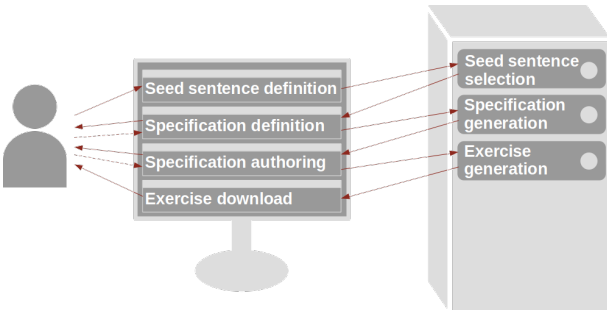

Figure 1: System architecture

The information flow between the user and the front-end, and between the front-end and the back-end is represented by arrows. Dashed arrows indicate optional information flow.

3.1 Seed sentence selection

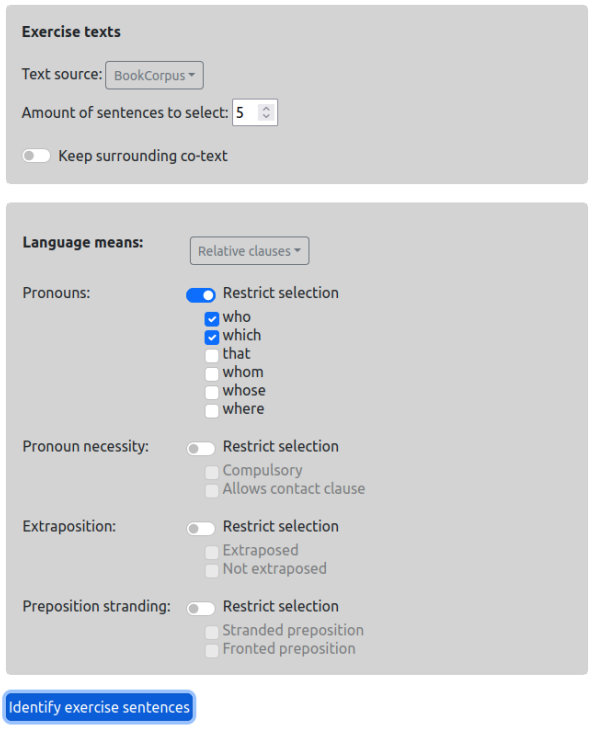

Figure 2: Seed sentence definition UI

Seed sentences, also referred to as *carrier sentences* or *candidate sentences* in the literature, are natural language sentences from which exercises are generated (Pilán et al., 2017). In our implementation, the selection of suitable sentences starts in the web interface shown in Figure 2. It supports three input sources: (1) the web, (2) the BookCorpus[4], and (3) custom texts. If users want

[4] The corpus based on an implementation by Kobayashi (2018) is available at https://the-eye.eu/public/AI/pile_preliminary_components/books1.tar.gz

to search the BookCorpus for candidate sentences, they need to specify the desired number of sentences. Since the space of possible parameter combinations grows exponentially with the number of parameters, the number of seed sentences to select can only be specified globally and not for specific parameter constellations. Crawling the Web in addition allows to search for sentences which appear in a defined semantic context so that users also need to specify a search term. Custom texts must be inserted into the provided input field. They can consist of manually compiled texts or any other texts copied from arbitrary sources.

An additional parameter determines whether some co-text is extracted along with the seed sentences or only the seed sentences themselves. If the co-text option is activated, the text in the same paragraph, delimited by line breaks, will be extracted as well. For contextualized exercises, the number of sentences cannot be specified. Instead, the exercise will contain all occurrences of the targeted linguistic structure in the paragraph as exercise items.

A final set of configuration parameters allows users to restrict the selection of seed sentences which will later be turned into exercise items. Available parameters depend on the targeted linguistic structures. For conditionals, they include the conditional type, the clause order, polarity, aspect, and sentence form. For relative clauses, the parameters consist of the relative pronoun, whether the pronoun is compulsory or can be left out, extraposition, and preposition stranding.

The seed sentence selection algorithm differs from one input source to another. For web texts, a google search is performed for the search term and the content of the search results is processed until the desired number of seed sentences has been extracted. For corpus texts, the documents of the corpus are searched instead, again until the required number of sentences has been identified. Custom texts are processed in their entirety.

For Natural Language Processing (NLP), the Java library Stanford CoreNLP[5], as well as the Python libraries NLTK[6], SpaCy[7] and Stanza[8] were considered. Table 1 summarizes the results of the evaluation of their reliability with respect to the annotations for seed sentence selection of conditionals and relative clauses. SpaCy and Stanza yielded similarly good results, with SpaCy performing considerably faster. Subsequent NLP analyses were therefore implemented based on SpaCy.

	Precision		**Recall**	
	RC	**C**	**RC**	**C**
NLTK	.89	.7	.7417	.9610
Stanza	.98	.81	.8976	.9927
SpaCy	.94	.86	.9039	.9902
Stanford CoreNLP	.96	.76	.7606	.9683
Sample size	100	100	635	410

Table 1: Evaluation of NLP libraries

Precision was computed for a random sample of 100 sentences from the BookCorpus. Recall values were determined for a collection of manually compiled example sentences. All metrics were determined for relative clauses (RC) and conditionals (Cond).

The algorithm processes the texts of all input sources in the same manner: A naive construction identification rule based on dependency parses determines whether a sentence could be a potential candidate. For conditionals, it searches for adverb clauses with some additional conditions such as the existence of a token with value *if* and the absence of a verb token contained in a manually compiled list of reported speech markers[9]. For relative clauses, the algorithm searches for relative clauses with a Wh-pronoun.

However, this rough filtering results in a considerable amount of noise in the sentence candidates. Pilán et al. (2017) identify a number of criteria for good seed sentences, including well-formedness, context independence, linguistic complexity and additional structural and lexical criteria. While we address most of the structural criteria, such as negated or interrogative contexts, with the parameters exposed to users, we deliberately do not restrict seed sentence selection based on lexical criteria, which are often user-dependent and better targeted by a macro-adaptive algorithm in the target ILTS (Gooding and Tragut, 2022). Compliance with context independence will be more likely when the co-text option is activated and can be addressed manually in the subsequent workflow step. In order to account for well-formedness and

[5] https://stanfordnlp.github.io/CoreNLP
[6] https://www.nltk.org
[7] https://spacy.io
[8] https://stanfordnlp.github.io/stanza

[9] Available lists (e.g. Tham and Nhi, 2021; Yilmaz and Özdem Erturk, 2017) contain predominantly affirmative markers. Since only question markers are relevant to confusions with conditional clauses, we compiled a list based on sampled evidence from the BookCorpus.

linguistic complexity, we apply further processing after the naive sentence selection: The algorithm extracts all the information relevant to exercise generation. This includes the exercise targets and their properties as well as properties of the sentences relevant to the configured parameters. The algorithm rejects the sentence as soon as one piece of information cannot be extracted or if it does not comply with the configured parameters. This not only ensures the highest possible success rate for exercise generation in the succeeding step, but also filters out most sentences which passed the naive filter but do not actually contain the targeted linguistic structure. In addition, we hypothesize that the NLP tools' inability to correctly process a sentence would reflect a beginning student's inability to do so, thus also eliminating sentences too complex for our target group.

The successfully parsed sentences are stored in a result list. If so specified by the user, the co-text of the paragraph is also stored in that list as individual elements. For seed sentences targeting conditionals, additional filtering is applied when the user has restricted the selection of the conditional type and selected both types. Since such a configuration is usually used for exercises targeting the distinction between conditional types, the seed sentence selection ensures that both conditional types occur in roughly equal numbers in the result list. If the result list already contains enough seed sentences for one conditional type, any subsequently found occurrences of that type will therefore be treated like sentences with no conditional construction. Similarly, a subtopic for relative clauses targets contact clauses for which students need to learn when the pronoun can be left out. It is therefore important to have seed sentences both with optional and with compulsory relative pronoun. If a user activates the selection restriction for pronoun necessity and selects both values, the algorithm therefore makes sure that sentences with compulsory and optional pronoun occur with similar frequency in the results.

Each element in the result list is tagged with its type of either co-text or exercise item. The list is used on the client to populate the user interface designed to configure exercise specification parameters.

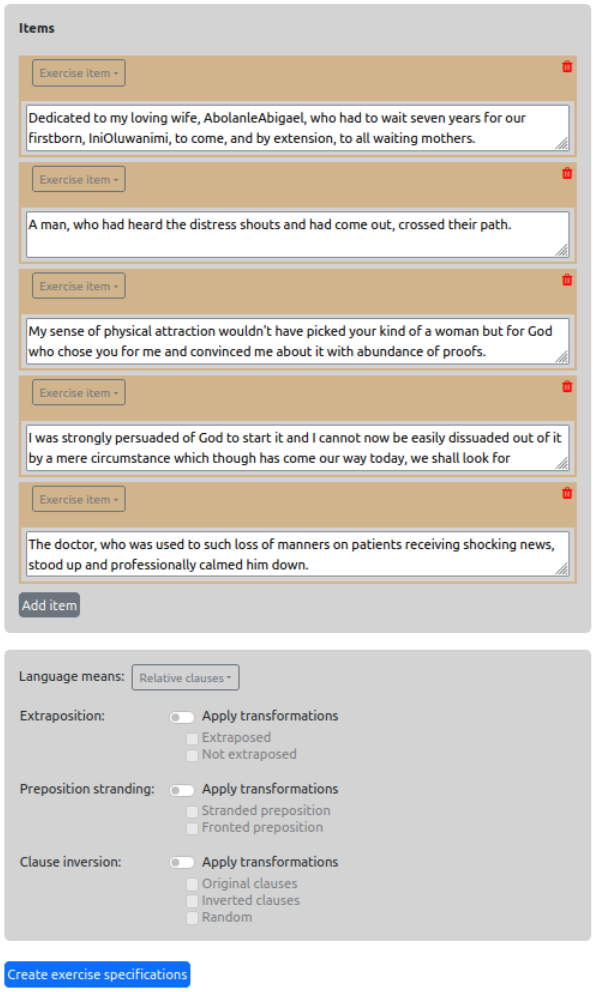

Figure 3: Specification definition UI

3.2 Exercise specification generation

The user interface to specify parameters of exercise specifications, shown in Figure 3, initially contains the exercise and co-text items extracted by the seed sentence selector. They can be edited, deleted, or their type changed from co-text to exercise item or vice versa. Additional items can be added manually. The order of all items can be changed through drag and drop mechanisms.

If no co-text items are specified, users can set additional parameters which will lead to the creation of linguistic transformations of the seed sentences. Transformations include for conditional sentences the aspect, conditional type, polarity, sentence form, and clause order. For relative clauses, preposition stranding, extraposition, and clause inversion are supported. The latter parameter transforms the original relative clause into a main clause and the original main clause into a relative clause, if possible. Whether a transforma-

tion results in a separate exercise specification or merely in an alternative sentence of the same specification depends on whether the target tokens, i.e., the pronoun of a relative clause or the verbs of conditional sentences, are affected by the transformation. For example, negating the main clause of the conditional sentence given in (1a) changes the verb from (*will go*) to *will not go* in (1b), thus requiring a new specification. Reversing the clause order in (1a) to that in (1c) does not affect the verb forms, therefore resulting in alternative sentences of the same specification. All transformations which result in a separate specification also offer the option to apply either of two realizations. In this case, the algorithm randomly applies one of the realizations of the transformation to each item while at the same time making sure that each realization is applied approximately the same number of times. This allows to generate exercises which practice a variety of linguistic phenomena.

(1) a. If he **gets** better, he **will go** to school.

b. If he **gets** better, he **will not go** to school.

c. He **will go** to school if he **gets** better.

Based on these configurations, the algorithm processes the texts declared as exercise items while keeping the co-text elements unchanged. Since it has been established in the previous step that the processed sentences must contain an occurrence of the targeted language means, the algorithm this time does not reject sentences which cannot be fully processed. Instead, it uses default values whenever a feature cannot be extracted. By shifting the focus from precision for seed sentence selection to recall for exercise specification generation, the same code can be used for both steps.

The extracted features are used to generate abstract exercise specifications which support a range of exercise types: Fill-in-the-Blanks, Single Choice, Memory, Jumbled Sentences, Short Answers, Mark-the-Words, and Categorization. These specifications are in addition enriched with exercise elements such as distractors for Single Choice exercises or parentheses for Fill-in-the-Blanks exercises. The distractor generation relies on Natural Language Generation (NLG). Since openly available Python libraries did not yield the desired output, the Java-based SimpleNlg[10] library is used to this purpose. The integration of this code is facilitated by the microservice architecture.

The generated exercise specifications are sent to the client where they are used to populate the exercise specification authoring interface.

3.3 Exercise generation

In order to finalize the specifications used for exercise generation, users can review them in the web interface shown in Figure 4. The grouping of multiple transformations into a single specification allows to reduce revision effort to a minimum. The transformations can be edited individually, deleted or added to. Each transformation can be marked as exercise seed from which to actually generate an exercise. If this option is not activated, the transformation merely serves as accepted correct answer alternative (provided the exercise context such as given prompts licenses the sentence). In order to make sure that all resulting exercises have an associated transformation for all items, the sentences are linked per parameter constellation across items. Deletion of one transformation therefore also deletes the corresponding sentence of all other items of the specification. Although some transformations of the same seed sentence require individual specifications, all specifications associated with the same seed sentences are linked by a common identifier. This enables adaptive systems using the generated exercises to avoid selecting similar activities in succession for the same learner. In addition to reviewing the generated exercise parameters such as target constructions, chunking, distractors, and hints in parentheses, the interface allows users to specify what exercises should be generated. As can be seen in Figure 5, this entails not only the exercise type, but also more specific parameters such as the number of distractors, whether to keep relative pronouns as individual chunks or combine them with adjoining ones, whether to insert exercise targets in both clauses or only one, or in which order to display the clauses from which to form relative sentences in the prompt. In addition, exercises can be generated for all linked items of a specification which are associated with the same transformation, as well as for a random choice of transformation of each item.

Based on these specifications, subsequent exercise generation is straightforward. All necessary information is already contained in the specifica-

[10] http://github.com/simplenlg/simplenlg

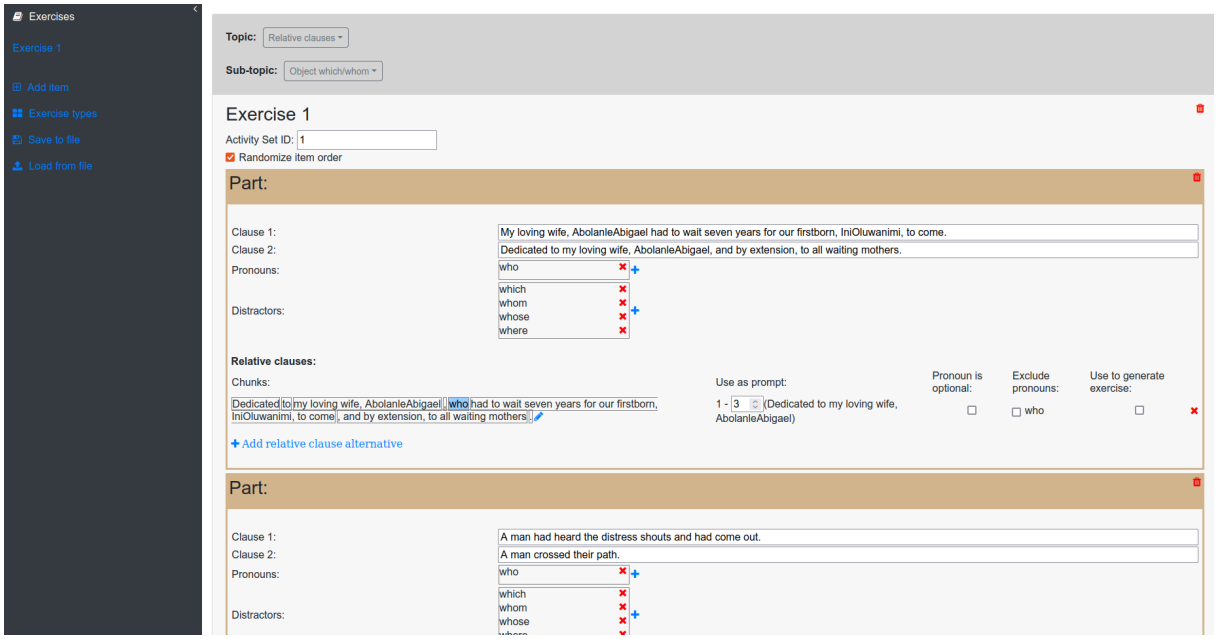

Figure 4: Specification authoring UI

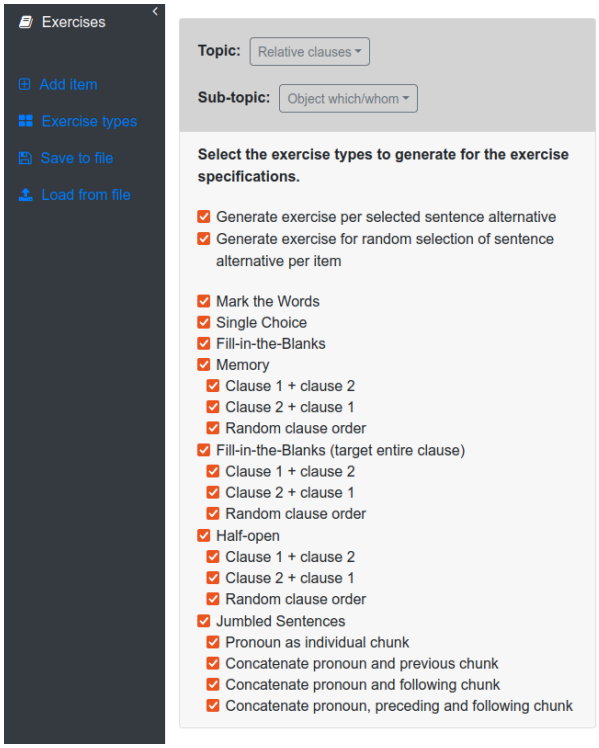

Figure 5: Exercise type definition UI

tions apart from instructions. These are stored in the code for each exercise type and linguistic structure. Apart from this, exercise generation consists in converting the specifications into the desired output format. Supported formats include the standardized H5P file format and a proprietary xml format for the in-house developed ILTS. The generated files are returned to the client where they can be downloaded by the user.

4 Evaluation

We evaluated precision and recall on candidate sentence selection for corpus texts and for manually compiled texts as well as the usability of the generated exercise specifications.

4.1 Methodology

We searched the BookCorpus for 100 occurrences of conditional sentences and relative sentences each with the naive sentence selection algorithm. The selection was not further restricted. We annotated them as *true positives* or *false positives* and computed precision values. We then determined which of these sentences were rejected by the sophisticated sentence selection algorithm and computed recall and precision values for this algorithm based on the data set obtained from the naive sentence selection. For a collection of 100 manually composed sentences for each of the two linguistic structures, we only applied the naive selection since for this input type, the sophisticated algorithm is bypassed. We computed recall values for the algorithm's acceptance of the input as seed sentences.

	RC	C	Corpus
Recall (N)	.91	1.0	M
Precision (N)	.93	.89	BC
Recall (S)	.3656	.7528	BC
Precision (S)	.8947	.9306	BC

Table 2: Evaluation of the seed sentence selection

Recall and precision were calculated for the naive (N) and for the sophisticated (S) algorithm. *Precision (S)* corresponds to overall precision. Recall of the naive algorithm was calculated on manually compiled texts (M), the remaining metrics on the BookCorpus (BC). For each metric, samples of 100 sentences were considered.

4.2 Results and Discussion

The results are summarized in Table 2: For seed sentence selection from the corpus, relative clauses obtain a precision of .93 on the naive selection. The precision of the sophisticated selection, which is also the precision of the overall seed sentence selection, is slightly lower at .8947. The decrease in precision is due to the high rejection rate, also resulting in a low recall of .3656, so that the percentage of accepted incorrect findings increases relative to the overall number of accepted sentences. While this might suggest that the additional filtering should be removed, the filtering also serves as a pre-selection with regard to the ensuing exercise generation from the specifications, thus rejecting sentences early on which cannot be processed successfully.

Results for conditionals are more in line with the expected behaviour. Precision on the corpus is already high (.89) for the naive sentence selection and increases further to .9306 with the sophisticated sentence selection. Recall of the sophisticated selection is also considerably higher than for relative clauses (.7528). Of the 89 sentences accepted as conditional sentences, 44 are actually not stereotypical conditional sentences taught in introductory language classes. They deviate in tense (e.g., Example 2a) or sentence structures such as using elliptical if-clauses (e.g., Example 2b). This highlights the relevance of parameters to restrict the selection of seed sentences which allows users to only select sentences with textbook properties.

(2) a. If I can't spoil my only daughter on her birthday, I'm not much of a father, now am I?

b. What if someone sees us?

Although the poor recall values indicate that a considerable amount of potential exercise sentences is lost in the process, this constitutes an accepted shortcoming when parsing large corpora. Considering the trade-off between fast performance and finding sentences lending themselves well to exercise generation, we put a focus on the latter criterion.

On the manually compiled sentences, the naive algorithm achieves recall values of 1.0 and .92 for conditionals and relative clauses respectively. Since each sentence of the data set contains a relevant construction, all conditional sentences are recognized by the algorithm while some relative clauses are rejected. These constitute either extraposed relative clauses such as example (3a) or sentences with the pronoun *whom* as in (3b). The issues can be traced back to incorrect parsing outputs obtained from the employed NLP tools.

(3) a. The kids screamed who are not from our school.

b. My parents called my teacher whom I saw today.

The number of exercises that can be generated from each seed sentence depends on three factors: (1) the user selections for sentence transformations in the specification definition UI and for exercise types in the specification authoring UI, (2) the algorithm's success in generating sentence transformations, and (3) the grammar subtopic.

$$\mathrm{e} = \mathrm{types} * \prod_{i=1}^{item-params} \mathrm{options}_i * \prod_{i=1}^{alternatives-params} \mathrm{options}_i \quad (4)$$

The maximum number of exercises breaks down according to the formula given in Equation 4: The number of generated exercise specification items constitutes the product of the options per activated transformation parameter of those parameters resulting in separate items. If all sentence alternatives are turned into exercises, the number of alternatives per exercise specification item is also considered. It constitutes the product of the options per activated transformation parameter of those parameters resulting in sentence alternatives. If instead only one randomly selected alternative is used per specification item, this number does not figure in the equation. The overall number of exercises constitutes the product of the

number of exercise types with the number of exercise specification items and, if applicable, the number of sentence alternatives per item.

	C_{sent}	C_{diff}	RC_{pron}	RC_{cont}
types	34	21	16	3
items	81	81	3	3
n_{rand}	**2754**	**1701**	**48**	**9**
alternatives	2	2	4	4
n_{all}	**5508**	**3402**	**192**	**36**

Table 3: Maximum exercise counts

For random alternative selection (n_{rand}), the maximum number of generated exercises depends on the available exercise types and the number of specification items. If each alternative is turned into an exercise (n_{all}), the number of alternatives per exercise specification item is considered in addition. Available exercise types differ between the subtopic differentiating conditional types (C_{diff}), the remaining subtopics on conditionals (C_{sent}), contact clauses (RC_{cont}), and the remaining subtopics (RC_{pron}) on relative clauses.

Table 3 illustrates that applying this formula to the subtopics conditional sentences, differentiation of conditional types, relative clauses with relative pronouns, and contact clauses results in up to more than 5500 exercises for a single seed sentence.

5 Conclusion

We presented a fully implemented approach to step-by-step generation of form-based grammar exercises from authentic texts. We showed that our approach applying the annotation algorithm in the seed sentence selection step successfully eliminates false positives of more complex linguistic constructions such as conditionals, and it reduces issues for all language means in subsequent processing steps. We also found evidence in our evaluation that allowing users to specify selection restrictions can be crucial for the usability of the tool in classroom instruction to support the identification of pedagogically suitable sentences.

Future work will improve the user interface both in design and maintainability. The envisioned React[11] implementation will make use of state-of-the-art web technologies. We also plan to extend the implementation to additional language means. The generated exercises will be tested in the AI2Teach[12] project extending the FeedBook ILTS (Rudzewitz et al., 2017) successfully used in field studies in regular high schools in Germany (Meurers et al., 2019). This will yield further insights as to whether the authentic texts are suitably complex and of appropriate content for the target group.

[11] https://reactjs.org

[12] https://fit.uni-tuebingen.de/Project/Details?id=7942

References

Jaweria Aftab. 2015. Teachers' Beliefs about Differentiated Instructions in Mixed Ability Classrooms: A Case of Time Limitation. *Journal of Education and Educational Development*, 2(2):94–114.

Itziar Aldabe, Maddalen Lopez de Lacalle, Montse Maritxalar, Edurne Martínez, and Larraitz Uria. 2006. ArikIturri: An Automatic Question Generator Based on Corpora and NLP Techniques. In *Proceedings of the 8th International Conference on Intelligent Tutoring Systems (ITS'06), Jhongli (Taiwan)*, pages 584–594. Springer-Verlag.

Georges Antoniadis, Sandra Echinard, Olivier Kraif, Thomas Lebarbé, Mathieu Loiseau, and Claude Ponton. 2004. NLP-based scripting for CALL activities. In *Proceedings of the Workshop on eLearning for Computational Linguistics and Computational Linguistics for eLearning*, pages 18–25.

Eckhard Bick. 2000. Live use of Corpus data and Corpus annotation tools in CALL: Some new developments in VISL. *Nordic Language Technology, Årbog for Nordisk Sprogteknologisk Forskningsprogram*, 2004:171–185.

Catherine Doughty and Michael Long. 2003. Optimal psycholinguistic environments for distance foreign language learning. *Language Learning & Technology*, 7(3):50–80.

Kledilson Ferreira and Álvaro R. Pereira Jr. 2018. Verb tense classification and automatic exercise generation. In *WebMedia '18: Proceedings of the 24th Brazilian Symposium on Multimedia and the Web*, pages 105–108.

Sian Gooding and Manuel Tragut. 2022. One Size Does Not Fit All: The Case for Personalised Word Complexity Models. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 353–365, Seattle, United States. Association for Computational Linguistics.

Tanja Heck and Detmar Meurers. 2022a. Automatic exercise generation to support macro-adaptivity in intelligent language tutoring systems. In *Intelligent CALL, granular systems and learner data: short papers from EUROCALL 2022*. Virtual.

Tanja Heck and Detmar Meurers. 2022b. Parametrizable exercise generation from authentic texts: Effectively targeting the language means on the curriculum. In *Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Appli-*

cations (BEA 2022), pages 154–166, Seattle, Washington. Association for Computational Linguistics.

Ayako Hoshino and Hiroshi Nakagawa. 2008. A Cloze Test Authoring System and Its Automation. In *Advances in Web Based Learning – ICWL 2007*, pages 252–263, Berlin, Heidelberg. Springer Berlin Heidelberg.

Colpaert Jozef and Emre Sevinc. 2010. ClozeFox: Gap Exercise Generator with Scalable Intelligence for Mozilla Firefox. https://github.com/emres/clozefox. [Online; accessed 08-November-2022].

Anisia Katinskaia, Javad Nouri, and Roman Yangarber. 2018. Revita: a language-learning platform at the intersection of ITS and CALL. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*. European Language Resources Association (ELRA).

Sosuke Kobayashi. 2018. Homemade BookCorpus. https://github.com/BIGBALLON/cifar-10-cnn. [Online; accessed 08-November-2022].

Chun Lai and Guofang Li. 2011. Technology and Task-Based Language Teaching: A Critical Review. *CALICO Journal*, 28.

Shaofeng Li, Rod Ellis, and Yan Zhu. 2016. Task-Based Versus Task-Supported Language Instruction: An Experimental Study. *Annual Review of Applied Linguistics*, 36:205–229.

Alexey Malafeev. 2015. Exercise Maker: Automatic Language Exercise Generation. In *Computational Linguistics and Intellectual Technologies*, pages 441–452.

Detmar Meurers. 2020. Natural language processing and language learning. In Carol A. Chapelle, editor, *The Concise Encyclopedia of Applied Linguistics*, pages 817–831. Wiley, Oxford.

Detmar Meurers, Kordula De Kuthy, Florian Nuxoll, Björn Rudzewitz, and Ramon Ziai. 2019. Scaling up intervention studies to investigate real-life foreign language learning in school. *Annual Review of Applied Linguistics*, 39:161–188.

Detmar Meurers, Ramon Ziai, Luiz Amaral, Adriane Boyd, Aleksandar Dimitrov, Vanessa Metcalf, and Niels Ott. 2010. Enhancing authentic web pages for language learners. In *Proceedings of the 5th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 10–18, Los Angeles. ACL.

Irina Pandarova, Torben Schmidt, Johannes Hartig, Ahcene Boubekki, Roger Jones, and Ulf Brefeld. 2019. Predicting the Difficulty of Exercise Items for Dynamic Difficulty Adaptation in Adaptive Language Tutoring. *International Journal of Artificial Intelligence in Education*.

Matthew Peacock. 1997. The Effect of Authentic Materials on the Motivation of EFL Learners. *ELT Journal*, 51(2):144–156.

Naiara Pérez and Montse Cuadros. 2017. Multilingual CALL Framework for Automatic Language Exercise Generation from Free Text. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 49–52, Valencia, Spain. Association for Computational Linguistics.

Laura Perez-Beltrachini, Claire Gardent, and German Kruszewski. 2012. Generating Grammar Exercises. In *Proceedings of the Seventh Workshop on Building Educational Applications Using NLP*, pages 147–156. Association for Computational Linguistics.

Ildikó Pilán, Elena Volodina, and Lars Borin. 2017. Candidate sentence selection for language learning exercises: From a comprehensive framework to an empirical evaluation. *Traitement Automatique des Langues (TAL), special issue on NLP for learning and teaching*, 57.

Robert Reynolds, Eduard Schaf, and Detmar Meurers. 2014. A view of Russian: Visual input enhancement and adaptive feedback. In *Proceedings of the third workshop on NLP for computer-assisted language learning*, pages 98–112, Uppsala. ACL.

Björn Rudzewitz, Ramon Ziai, Kordula De Kuthy, and Detmar Meurers. 2017. Developing a web-based workbook for English supporting the interaction of students and teachers. In *Proceedings of the Joint 6th Workshop on NLP for Computer Assisted Language Learning and 2nd Workshop on NLP for Research on Language Acquisition*, pages 36–46.

Vasile Rus, Nobal B. Niraula, and Rajendra Banjade. 2015. DeepTutor: An Effective, Online Intelligent Tutoring System That Promotes Deep Learning. In *Association for the Advancement of Artificial Intelligence*.

Dara Tafazoli, Mª Elena Gómez Parra, and Cristina Huertas Abril. 2019. Intelligent language tutoring system: Integrating intelligent computer-assisted language learning into language education. *International Journal of Information and Communication Technology Education*, 15:60–74.

Duong My Tham and Tran Phuong Nhi. 2021. A Corpus-Based Study on Reporting Verbs Used in Tesol Research Articles by Native and Non-Native Writers. *VNU Journal of Foreign Studies*, 37(3).

Janine Toole and Trude Heift. 2001. Generating Learning Content for an Intelligent Language Tutoring System. In *Proceedings of NLP-CALL Workshop at the 10th Int. Conf. on Artificial Intelligence in Education (AI-ED). San Antonio, Texas*, pages 1–8.

Penny Ur. 2018. PPP: Presentation–Practice–Production. *The TESOL Encyclopedia of English Language Teaching*, pages 1–6.

Maide Yilmaz and Zeynep Özdem Erturk. 2017. A Contrastive Corpus-Based Analysis of the Use of Reporting Verbs by Native and Non-Native ELT Researchers. *Novitas-ROYAL (Research on Youth and Language)*, 11(2):112–127.