Manipulating the Difficulty of C-Tests

Ji-Ung Lee and **Erik Schwan** and **Christian M. Meyer**
Ubiquitous Knowledge Processing (UKP) Lab and Research Training Group AIPHES
Computer Science Department, Technische Universität Darmstadt, Germany
https://www.ukp.tu-darmstadt.de

Abstract

We propose two novel manipulation strategies for increasing and decreasing the difficulty of C-tests automatically. This is a crucial step towards generating learner-adaptive exercises for self-directed language learning and preparing language assessment tests. To reach the desired difficulty level, we manipulate the size and the distribution of gaps based on absolute and relative gap difficulty predictions. We evaluate our approach in corpus-based experiments and in a user study with 60 participants. We find that both strategies are able to generate C-tests with the desired difficulty level.

1 Introduction

Learning languages is of utmost importance in an international society and formulated as a major political goal by institutions such as the European Council, who called for action to "teaching at least two foreign languages" (EC, 2002, p. 20). But also beyond Europe, there is a huge demand for language learning worldwide due to increasing globalization, digital communication, and migration.

Among multiple different learning activities required for effective language learning, we study one particular type of exercise in this paper: *C-tests* are a special type of cloze test in which the second half of every second word in a given text is replaced by a gap (Klein-Braley and Raatz, 1982). Figure 1 (a) shows an example. To provide context, the first and last sentences of the text do not contain any gaps. C-tests rely on the reduced redundancy principle (Spolsky, 1969) arguing that a language typically employs more linguistic information than theoretically necessary to communicate unambiguously. Proficient speakers intuitively understand an utterance even if the level of redundancy is reduced (e.g., when replacing a word's suffix with a gap), whereas learners typically rely on the redundant signal to extrapolate the meaning of an utterance.

Besides general vocabulary knowledge, C-tests require orthographic, morphologic, syntactic, and semantic competencies (Chapelle, 1994) to correctly fill in all gaps, which make them a frequently used tool for language assessment (e.g., placement tests). Given that C-tests can be easily generated automatically by introducing gaps into an arbitrary text and that there is usually only a single correct answer per gap given its context, C-tests are also relevant for self-directed language learning and massive open online courses (MOOC), where large-scale personalized exercise generation is necessary.

A crucial question for such tasks is predicting and manipulating the *difficulty* of a C-test. For language assessment, it is important to generate C-tests with a certain target difficulty to allow for comparison across multiple assessments. For self-directed language learning and MOOCs, it is important to adapt the difficulty to the learner's current skill level, as an exercise should be neither too easy nor too hard so as to maximize the learning effect and avoid boredom and frustration (Vygotsky, 1978). Automatic difficulty prediction of C-tests is hard, even for humans, which is why there have been many attempts to theoretically explain C-test difficulty (e.g., Sigott, 1995) and to model features used in machine learning systems for automatic difficulty prediction (e.g., Beinborn et al., 2014).

While state-of-the-art systems produce good prediction results compared to humans (Beinborn, 2016), there is yet no work on *automatically manipulating* the difficulty of C-tests. Instead, C-tests are generated according to a fixed scheme and manually post-edited by teachers, who might use the predictions as guidance. But this procedure is extremely time-consuming for language assessment and no option for large-scale self-directed learning.

In this paper, we propose and evaluate two strategies for automatically changing the gaps of a C-test in order to reach a given target difficulty. Our first

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 360–370
Florence, Italy, July 28 - August 2, 2019.

(a)	(b)	(c)
It i_ being fou___, moreover, i_ fairly cl___ correspondence wi__ the predi_____ of t__ soothsayers o_ the th___ factories. Th__ predicted escal_____, and escal_____ is wh__ we a__ getting. T__ biggest nuc____ device t__ United Sta___ has expl____ measured so__ 15 meg...	It is being fought, more____, in fai___ cl___ corresp_____ with the predi_____ of the sooth_____ of the th___ fact_____. Th__ pred_____ escal_____, and escal_____ is what w_ are get____. The big____ nuc____ dev___ the United States h__ expl____ meas____ some 15 meg...	It i_ being fough_, moreover, i_ fairly clos_ correspondence wit_ the prediction_ of t__ soothsayers o_ the thin_ factories. The_ predicted escalatio_, and escalatio_ is wha_ we ar_ getting. T__ biggest nuclea_ device t__ United State_ has explode_ measured som_ 15 meg...

Figure 1: C-tests with (a) standard gap scheme, (b) manipulated gap position, and (c) manipulated gap size

strategy varies the distribution of the gaps in the underlying text and our second strategy learns to decide to increase or decrease a gap in order to make the test easier or more difficult. Our approach breaks away from the previously fixed C-test creation scheme and explores new ways of motivating learners by using texts they are interested in and generating tests from them at the appropriate level of difficulty. We evaluate our strategies both automatically and in a user study with 60 participants.

2 Related Work

In language learning research, there is vast literature on cloze tests. For example, Taylor (1953) studies the relation of cloze tests and readability. In contrast to C-tests (Klein-Braley and Raatz, 1982), cloze tests remove whole words to produce a gap leading to more ambiguous solutions.

Chapelle and Abraham (1990) contrast four types of cloze tests, including fixed-ratio cloze tests replacing every i^{th} word with a gap, rational cloze tests that allow selecting the words to replace according to the language trait that should be assessed, multiple-choice tests, and C-tests. Similar to our work, they conduct a user study and measure the difficulty posed by the four test types. They find that cloze tests replacing entire words with a gap are more difficult than C-tests or multiple-choice tests. In our work, we go beyond this by not only varying between gaps spanning the entire word (cloze test) or half of the word (C-test), but also changing the size of the C-test gaps. Laufer and Nation (1999) propose using C-tests to assess vocabulary knowledge. To this end, they manually construct C-tests with only a single gap, but use larger gaps than half of the word's letters. Our work is different to these previous works, since we test varying positions and sizes for C-test gaps and, more importantly, we aim at manipulating the difficulty of a C-test automatically by learning to predict the difficulty of the gaps and how their manipulation affects the difficulty.

Previous work on automatically controlling and manipulating test difficulty has largely focused on multiple-choice tests by generating appropriate distractors (i.e., incorrect solutions). Wojatzki et al. (2016) avoid ambiguity of their generated distractors, Hill and Simha (2016) fit them to the context, and Perez and Cuadros (2017) consider multiple languages. Further work by Zesch and Melamud (2014), Beinborn (2016), and Lee and Luo (2016) employ word difficulty, lexical substitution, and the learner's answer history to control distractor difficulty.

For C-tests, Kamimoto (1993) and Sigott (2006) study features of hand-crafted tests that influence the difficulty, and Beinborn et al. (2014) and Beinborn (2016) propose an automatic approach to estimate C-test difficulty, which we use as a starting point for our work.

Another related field of research in computer-assisted language learning is readability assessment and, subsequently, text simplification. There exists ample research on predicting the reading difficulty for various learner groups (Hancke et al., 2012; Collins-Thompson, 2014; Pilán et al., 2014). A specific line of research focuses on reducing the reading difficulty by text simplification (Chandrasekar et al., 1996). By reducing complex texts or sentences to simpler ones, more texts are made accessible for less proficient learners. This is done either on a word level by substituting difficult words with easier ones (e.g., Kilgarriff et al., 2014) or on a sentence level (Vajjala and Meurers, 2014). More recent work also explores sequence-to-sequence neural network architectures for this task (Nisioi et al., 2017). Although the reading difficulty of a text partly contributes to the overall exercise difficulty of C-tests, there are many other factors with a substantial influence (Sigott, 1995). In particular, we can generate many different C-tests from the same text and thus reading difficulty and text simplification alone are not sufficient to determine and manipulate the difficulty of C-tests.

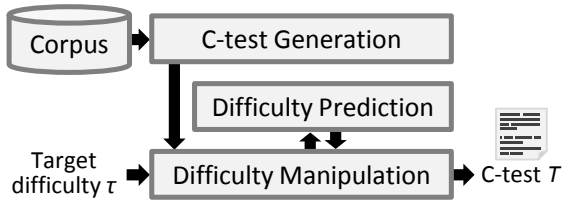


Figure 2: Proposed system architecture

3 Task Overview

We define a C-test $T = (u, w_1, \ldots, w_{2n}, v, G)$ as a tuple of left and right context u and v (typically one sentence) enframing $2n$ words w_i where $n = |G|$ is the number of gaps in the gap set G. In each gap $g = (i, \ell) \in G$, the last ℓ characters of word w_i are replaced by a blank for the learners to fill in. Klein-Braley and Raatz (1982) propose the default gap generation scheme DEF with $G = \{(2j, \lceil \frac{|w_{2j}|}{2} \rceil) \mid 1 \leq j \leq n\}$ in order to trim the (larger) second half of every second word. Single-letter words, numerals, and punctuation are not counted as words w_i and thus never contain gaps. Figure 1 (a) shows an example C-test generated with the DEF scheme.

A major limitation of DEF is that the difficulty of a C-test is solely determined by the input text. Most texts, however, yield a medium difficulty (cf. section 6) and thus do not allow any adaptation to beginners or advanced learners unless they are manually postprocessed. In this paper, we therefore propose two strategies to manipulate the gap set G in order to achieve a given *target difficulty* $\tau \in [0, 1]$ ranging from small values for beginners to high values for advanced learners. To estimate the difficulty $d(T) = \frac{1}{|G|} \sum_{g \in G} d(g)$ of a C-test T, we aggregate the predicted difficulty scores $d(g)$ of each gap. In section 4, we reproduce the system by Beinborn (2016) modeling $d(g) \approx e(g)$ as the estimated mean error rates $e(g)$ per gap across multiple learners, and we conduct additional validation experiments on a newly acquired dataset.

The core of our work is the manipulation of the gap set G in order to minimize the difference $|d(T) - \tau|$ between the predicted test difficulty $d(T)$ and the requested target difficulty τ. To this end, we employ our difficulty prediction system for validation and propose a new regression setup that predicts the relative change of $d(g)$ when manipulating the size ℓ of a gap.

Figure 2 shows our system architecture: Based on a text corpus, we generate C-tests for arbitrary texts (e.g., according to the learner's interests). Then, we manipulate the difficulty of the generated text by employing the difficulty prediction system in order to reach the given target difficulty τ for a learner (i.e., the estimated learner proficiency) to provide neither too easy nor too hard tests.

4 C-Test Difficulty Prediction

Beinborn et al. (2014) and Beinborn (2016) report state-of-the-art results for the C-test difficulty prediction task. However, there is yet no open-source implementation of their code and there is little knowledge about the performance of newer approaches. Therefore, we (1) conduct a reproduction study of Beinborn's (2016) system, (2) evaluate newer neural network architectures, and (3) validate the results on a newly acquired dataset.

Reproduction study. We obtain the original software and data from Beinborn (2016). This system predicts the difficulty $d(g)$ for each gap within a C-test using a support vector machine (SVM; Vapnik, 1998) with 59 hand-crafted features. The proposed features are motivated by four factors which are deemed important for assessing the gap difficulty: *item dependency*, *candidate ambiguity*, *word difficulty*, and *text difficulty*. We use the same data (819 filled C-tests), metrics, and setup as Beinborn (2016). That is, we perform leave-one-out cross validation (LOOCV) and measure the Pearson correlation ρ, the rooted mean squared error RMSE, and the quadratic weighted kappa $\mathrm{QW}\kappa$ as reported in the original work.

The left hand side of table 1 shows the results of our reproduced SVM compared to the original SVM results reported by Beinborn (2016). Even though we reuse the same code as in their original work, we observe small differences between our reproduction and the previously reported scores.

We were able to trace these differences back to libraries and resources which have been updated and thus changed over time. One example is Ubuntu's system dictionary, the *American English dictionary words* (wamerican), on which the original system relies. We experiment with different versions of the dictionary between Ubuntu 14.04 (wamerican v.7.1.1) and 18.04 (wamerican v.2018.04.16-1) and observe differences of one or two percentage points. As a best practice, we suggest to fix the versions of all resources and avoid any system dependencies.

Neural architectures. We compare the system with deep learning methods based on multi-layer

Model	Original data			New data		
	ρ	RMSE	qwκ	ρ	RMSE	qwκ
SVM (original)	.50	.23	.44	–	–	–
SVM (reproduced)	.49	.24	.47	.50	.21	.39
MLP	.42	.25	.31	.41	.22	.25
BiLSTM	.49	.24	.35	.39	.24	.27

Table 1: Results of the difficulty prediction approaches. SVM (original) has been taken from Beinborn (2016)

perceptrons (MLP) and bi-directional long short-term memory (BiLSTM) architectures, which are able to capture non-linear feature dependencies.[1] To cope for the non-deterministic behavior of the neural networks, we repeat all experiments ten times with different random weight initializations and report the averaged results (Reimers and Gurevych, 2017). While the MLP is trained similar as our reproduced SVM, the BiLSTM receives all gaps of a C-test as sequential input. We hypothesize that this sequence regression setup is better suited to capture gaps interdependencies. As can be seen from the table, the results of the neural architectures are, however, consistently worse than the SVM results. We analyze the RMSE on the train and development sets and observe a low bias, but a high variance. Thus, we conclude that although neural architectures are able to perform well for this task, they lack a sufficient amount of data to generalize.

Experiments on new data. To validate the results and assess the robustness of the difficulty prediction system, we have acquired a new C-test dataset from our university's language center. 803 participants of placement tests for English courses solved five C-tests (from a pool of 53 different C-tests) with 20 gaps each. Similar to the data used by Beinborn (2016), we use the error rates $e(g)$ for each gap as the $d(g)$ the methods should predict.

The right-hand side of table 1 shows the performance of our SVM and the two neural methods. The results indicate that the SVM setup is well-suited for the difficulty prediction task and that it successfully generalizes to new data.

Final model. We train our final SVM model on all available data (i.e., the original and the new data) and publish our source code and the trained model on GitHub.[2] Similar to Beinborn (2016), we cannot openly publish our dataset due to copyright.

Algorithm 1 Gap selection strategy (SEL)

procedure GAPSELECTION(T, τ)
  $G_{\text{FULL}} \leftarrow \{(i, \lceil \frac{|w_i|}{2} \rceil) \mid 1 \leq i \leq 2n\}$
  $G_{\text{SEL}} \leftarrow \emptyset$
  while $|G_{\text{SEL}}| < n$ **do**
    $G_{\leq\tau} \leftarrow \{g \in G_{\text{FULL}} \mid d(g) \leq \tau\}$
    if $|G_{\leq\tau}| > 0$ **then**
      $g^* \leftarrow \arg\min_{g \in G_{\leq\tau}} |d(g) - \tau|$
      $G_{\text{SEL}} \leftarrow G_{\text{SEL}} \cup \{g^*\}$
      $G_{\text{FULL}} \leftarrow G_{\text{FULL}} \setminus \{g^*\}$
    $G_{>\tau} \leftarrow \{g \in G_{\text{FULL}} \mid d(g) > \tau\}$
    if $|G_{>\tau}| > 0$ **then**
      $g^* \leftarrow \arg\min_{g \in G_{>\tau}} |d(g) - \tau|$
      $G_{\text{SEL}} \leftarrow G_{\text{SEL}} \cup \{g^*\}$
      $G_{\text{FULL}} \leftarrow G_{\text{FULL}} \setminus \{g^*\}$
  return G_{SEL}

5 C-Test Difficulty Manipulation

Given a C-test $T = (u, w_1, \ldots, w_{2n}, v, G)$ and a target difficulty τ, the goal of our manipulation strategies is to find a gap set G such that $d(T)$ approximates τ. A naïve way to achieve this goal would be to generate C-tests for all texts in a large corpus with the DEF scheme and use the one with minimal $|d(T) - \tau|$. However, most corpora tend to yield texts of a limited difficulty range that only suit a specific learner profile (cf. section 6). Another drawback of the naïve strategy is that it is difficult to control for the topic of the underlying text and in the worst case, the necessity to search through a whole corpus for selecting a fitting C-test.

In contrast to the naïve strategy, our proposed manipulation strategies are designed to be used in real time and manipulate any given C-test within 15 seconds at an acceptable quality.[3] Both strategies operate on a given text (e.g., on a topic a learner is interested in) and manipulate its gap set G in order to come close to the learner's current language skill. The first strategy varies the position of the gaps and the second strategy learns to increase or decrease the size of the gaps.

5.1 Gap Selection Strategy

The default C-test generation scheme DEF creates a gap in every second word w_{2j}, $1 \leq j \leq n$. The core idea of our first manipulation strategy SEL is to distribute the n gaps differently among the all $2n$ words in order to create gaps for easier or harder words than in the default generation scheme. Therefore, we use the difficulty predic-

[1] Network parameters and a description of the tuning process are provided in this paper's appendix.

[2] https://github.com/UKPLab/acl2019-ctest-difficulty-manipulation (licensed under the Apache License 2.0).

[3] On an *Intel-i5* with 4 CPUs and 16 GB RAM.

tion system to predict $d(g)$ for any possible gap $g \in G_{\texttt{FULL}} = \{(i, \lceil \frac{|w_i|}{2} \rceil) \mid 1 \leq i \leq 2n\}$ (i.e., assuming a gap in all words rather than in every second word). Then, we alternate between adding gaps to the resulting $G_{\texttt{SEL}}$ that are easier and harder than the preferred target difficulty τ, starting with those having a minimal difference $|d(g) - \tau|$.

Algorithm 1 shows this procedure in pseudocode and figure 1 shows a C-test whose difficulty has been increased with this strategy. Note that it has selected gaps at *corresponding* rather than *with*, and *soothsayers* rather than *the*. Our proposed algorithm is optimized for runtime. An exhaustive search would require testing $\binom{2n}{n}$ combinations if the number of gaps is constant. For $n = 20$, this yields 137 billion combinations. While more advanced optimization methods might find better gap selections, we show in section 6 that our strategy achieves good results.

5.2 Gap Size Strategy

Our second manipulation strategy `SIZE` changes the size of the gaps based on a pre-defined gap set. Increasing a gap $g = (i, \ell)$ by one or more characters, yielding $g' = (i, \ell + k)$ increases its difficulty (i.e., $d(g') \geq d(g)$), while smaller gaps make the gap easier. We identify a major challenge in estimating the effect of increasing or decreasing the gap size on the gap difficulty. Although $d(g')$ could be estimated using the full difficulty prediction system, the search space is even larger than for the gap selection strategy, since each of the n gaps has $|w_i| - 2$ possible gap sizes to test. For $n = 20$ and an average word length of six, this amounts to one trillion possible combinations.

We therefore propose a new approach to predict the *relative difficulty change* of a gap $g = (i, \ell)$ when increasing the gap size by one letter $\Delta_{\text{inc}}(g) \approx d(g') - d(g)$, $g' = (i, \ell + 1)$ and correspondingly when decreasing the gap size by one letter $\Delta_{\text{dec}}(g) \approx d(g) - d(g')$, $g' = (i, \ell - 1)$. The notion of relative difficulty change enables gap size manipulation in real time, since we do not have to invoke the full difficulty prediction system for all combinations. Instead, we can incrementally predict the effect of changing a single gap.

To predict Δ_{inc} and Δ_{dec}, we train two SVMs on all gap size combinations of 120 random texts from the Brown corpus (Francis, 1965) using the following features: predicted absolute gap difficulty, word length, new gap size, modified character, a binary indicator if the gap is at a *th* sound, and logarithmic difference of alternative solutions capturing the degree of ambiguity with varying gap size.

Algorithm 2 Gap size strategy (`SIZE`)

```
procedure IncreaseDifficulty(T, τ)
    G_SIZE ← G_DEF
    D ← d(T)
    while D < τ do
        g* = (i, ℓ) ← arg max_{g∈G_SIZE} Δ_inc(g)
        ℓ ← ℓ + 1
        D ← D + Δ_inc(g)
    return G_SIZE
```

With a final set of only six features, our new models are able to approximate the relative difficulty change very well deviating from the original system's prediction only by 0.06 RMSE for Δ_{inc} and 0.13 RMSE for Δ_{dec}. The predictions of both models highly correlate with the predictions achieving a Pearson's ρ of over 0.8. Besides achieving a much faster average runtime of 0.056 seconds for the relative model vs. 11 seconds for the full prediction of a single change, we can invoke the relative model iteratively to estimate $d(T)$ for multiple changes of the gap size more efficiently.

The final manipulation strategy then requires just a single call of the full prediction system. If $d(T) < \tau$, we incrementally increase the gap sizes to make T more difficult and, vice-versa, decrease the gap sizes if $d(T) > \tau$. In each iteration, we modify the gap with the highest relative difficulty change in order to approach the given target difficulty τ as quickly as possible. Algorithm 2 shows pseudocode for creating G_{size} with increased difficulty (i.e., $d(T) < \tau$) based on the default gap scheme `DEF`. The procedure for $d(T) > \tau$ works analogously, but using Δ_{dec} and decreasing the gap size. Figure 1 (c) shows a much easier version of the example C-test, in which a learner often only has to complete the last one or two letters.

6 Evaluation of the Manipulation System

To evaluate our C-test manipulation strategies, we first test their ability to cover a higher range of target difficulties than the default generation scheme and then measure how well they meet the desired target difficulty for texts from different domains. We conduct our experiments on 1,000 randomly chosen paragraphs for each of the Gutenberg (Lahiri, 2014), Reuters (Lewis et al., 2004), and Brown (Francis, 1965) corpora. We conduct our experiments on English, but our strategies can be adapted to many related languages.

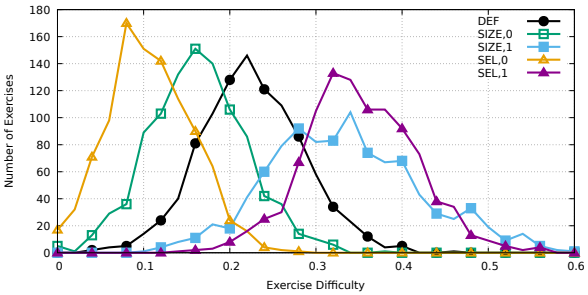


Figure 3: Difficulty distribution of exercises generated with DEF, SEL, and SIZE for extreme τ values

Difficulty range. The black ●-marked line of figure 3 shows the distribution of $d(T)$ based on our difficulty prediction system when creating a C-test with the default generation scheme DEF for all our samples of the Brown corpus. The vast majority of C-tests range between 0.15 and 0.30 with a predominant peak at 0.22.

To assess the maximal difficulty range our strategies can achieve, we generate C-tests with maximal ($\tau = 1$) and minimal target difficulty ($\tau = 0$) for both strategies $S \in \{\texttt{SEL}, \texttt{SIZE}\}$, which are also shown in figure 3 as (S, τ). Both strategies are able to clearly increase and decrease the test difficulty in the correct direction and they succeed in substantially increasing the total difficulty range beyond DEF. While SEL is able to reach lower difficulty ranges, it has bigger issues with generating very difficult tests. This is due to its limitation to the fixed gap sizes, whereas SIZE can in some cases create large gaps that are ambiguous or even unsolvable. Since SIZE is, however, limited to the 20 predefined gaps, it shows a higher variance. Especially short gaps such as *is* and *it* cannot be made more difficult. Combining the two strategies is thus a logical next step for future work, building upon our findings for both strategies. We make similar observations on the Reuters and Gutenberg corpora and provide the respective figures in the appendix.

Manipulation quality. We finally evaluate how well each strategy S reaches a given target difficulty. That is, we sample a random corpus text and τ, create the C-test using strategy S, predict the test difficulty $d(T)$ and measure its difference to τ using RMSE. Table 2 shows the results for our three corpora. Throughout all three corpora, both manipulation strategies perform well. SEL consistently outperforms SIZE, which matches our observations from the previous experiment. Mind that these results depend on the quality of the automatic difficulty predictions, which is why we conduct a user-based evaluation in the next section.

Strategy	Brown	Reuters	Gutenberg
SEL	.11	.12	.10
SIZE	.13	.15	.12

Table 2: RMSE for both strategies on each corpora with randomly sampled target difficulties τ

7 User-based Evaluation

Hypothesis. To evaluate the effectiveness of our manipulation strategies in a real setting, we conduct a user study and analyze the difficulty of the manipulated and unmanipulated C-tests. We investigate the following hypothesis: When increasing a test's difficulty using strategy S, the participants will make more errors and judge the test harder than a default C-test and, vice versa, when decreasing a test's difficulty using S, the participants will make less errors and judge the test easier.

Experimental design. We select four different English texts from the Brown corpus and shorten them to about 100 words with keeping their paragraph structure intact. None of the four texts is particularly easy to read with an average grade level above 12 and a Flesh reading ease score ranging between 25 (very difficult) to 56 (fairly difficult). In the supplementary material, we provide results of an automated readability analysis using standard metrics. From the four texts, we then generate the C-tests T_i, $1 \leq i \leq 4$ using the default generation scheme DEF. All tests contain exactly $n = 20$ gaps and their predicted difficulties $d(T_i)$ are in a mid range between 0.24 and 0.28. T_1 remains unchanged in all test conditions and is used to allow the participants to familiarize with the task. For the remaining three texts, we generate an easier variant $T_i^{S,\text{dec}}$ with target difficulty $\tau = 0.1$ and a harder variant $T_i^{S,\text{inc}}$ with $\tau = 0.5$ for both strategies $S \in \{\texttt{SEL}, \texttt{SIZE}\}$.

From these tests, we create 12 sequences of four C-tests that we give to the participants. Each participant receives T_1 first to familiarize with the task. Then, they receive one easy $T_i^{S,\text{dec}}$, one default T_i, and one hard $T_i^{S,\text{inc}}$ C-test for the same strategy S based on the texts $i \in \{2, 3, 4\}$ in random order without duplicates (e.g., the sequence $T_1\, T_2^{\texttt{SEL,dec}}\, T_3\, T_4^{\texttt{SEL,inc}}$). Having finished a C-test, we ask them to judge the difficulty of this test on a

five-point Likert scale ranging from *too easy* to *too hard*. After solving the last test, we additionally collect a ranking of all four tests by their difficulty.

Data collection. We collect the data from our participants with a self-implemented web interface for solving C-tests. We create randomized credentials linked to a unique ID for each participant and obfuscate their order, such that we can distinguish them but cannot trace back their identity and thus avoid collecting any personal information. Additionally, we ask each participant for their consent on publishing the collected data. For experiments with a similar setup and task, we obtained the approval of the university's ethics commission. After login, the participants receive instructions and provide a self-assessment of their English proficiency and their time spent on language learning. The participants then solve the four successive C-tests without knowing the test difficulty or the manipulation strategy applied. They are instructed to spend a maximum of five minutes per C-test to avoid time-based effects and to prevent them from consulting external resources, which would bias the results.

Participants. A total of 60 participants completed the study. We uniformly distributed the 12 test sequences (six per strategy), such that we have 30 easy, 30 default, and 30 hard C-test results for each manipulation strategy. No participant is native in English, 17 are taking language courses, and 57 have higher education or are currently university students. The frequency of their use of English varies, as we found a similar number of participants using English daily, weekly, monthly, and (almost) never in practice. An analysis of the questionnaire is provided in the paper's appendix.

Hypothesis testing. We evaluate our hypothesis along three dimensions: (1) the actual error rate of the participants, (2) the perceived difficulty after each individual C-test (Likert feedback), and (3) the participants' final difficulty ranking. While the latter forces the participants to provide an explicit ranking, the former allows them to rate C-tests equally difficult. We conduct significance testing at the Bonferroni-corrected $\alpha = \frac{0.05}{2} = 0.025$ for each dimension using one-tailed t-tests for the continuous error rates and one-tailed Mann–Whitney U tests for the ordinal-scaled perceived difficulties and rankings. Figure 4 shows notched boxplots of our results.

To test our hypothesis, we first formulate a null hypothesis that (a) the mean error rate, (b) the median perceived difficulty (Likert feedback), and (c) the median rank of the manipulated tests equal the default tests. While the participants have an average error rate of 0.3 on default C-tests, the $T_i^{S,\mathrm{dec}}$ tests are significantly easier with an average error rate of 0.15 ($t = 7.49$, $p < 10^{-5}$) and the $T_i^{S,\mathrm{inc}}$ tests are significantly harder with an average error rate of 0.49 ($t = -7.83$, $p < 10^{-5}$), so we can safely reject the null hypothesis for error rates.

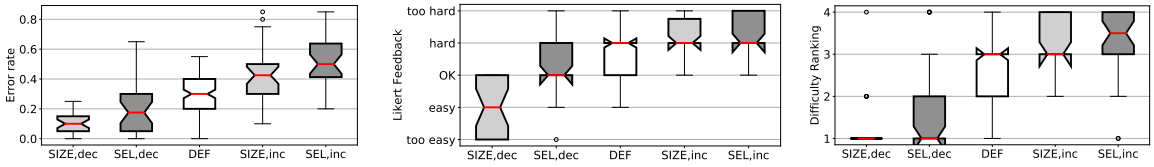
	easy (dec)		default	hard (inc)	
	SEL	SIZE	DEF	SEL	SIZE
T_1	–	–	.30	–	–
T_2	.17*	.11*	.34	.66*	.44*
T_3	.16*	.10*	.27	.52*	.43*
T_4	.28	.09*	.30	.43*	.45*
Average	.20*	.10*	.30	.53*	.44*

Table 3: Mean error rates $e(T)$ per text and strategy. Results marked with * deviate significantly from DEF

Table 3 shows the error rates per C-test and strategy. Both SEL and SIZE are overall able to significantly ($p < 0.025$) increase and decrease the test's difficulty over DEF, and with the exception of $T_4^{\mathtt{SEL},\mathrm{dec}}$, the effect is also statistically significant for all individual text and strategy pairs. Figure 5 shows the 30 participants per strategy on the x-axis and their error rates in their second to fourth C-test on the y-axis. C-tests, for which we increased the difficulty (S, inc), yield more errors than C-tests with decreased difficulty (S, dec) in all cases. The easier tests also yield less errors than the test with the default scheme DEF in most cases. While hard tests often have a much higher error rate than DEF, we find some exceptions, in which the participant's error rate is close or even below the DEF error rate.

Regarding the perceived difficulty, we find that the participants judge the manipulated C-tests with lower $d(T)$ as easier on both the Likert scale ($z = 6.16$, $p < 10^{-5}$) and in the rankings ($z = 6.59$, $p < 10^{-5}$) based on the Mann-Whitney-U test. The same is true for C-tests that have been manipulated to a higher difficulty level, which the participant judge harder ($z = -4.57$, $p < 10^{-5}$) and rank higher ($z = -3.86$, $p < 6 \cdot 10^{-5}$). We therefore reject the null hypotheses for the Likert feedback and the rankings and conclude that both strategies can effectively manipulate a C-test's difficulty.

Manipulation quality. We further investigate if the strategies yield different difficulty levels. There-



(a) (b) (c)

Figure 4: Notched boxplots for the (a) observed error rates, (b) Likert feedback, and (c) the participants' rankings

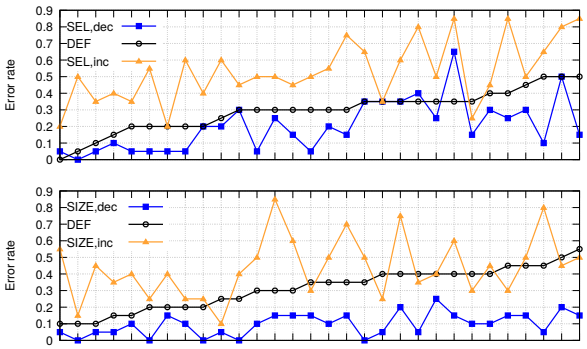


Figure 5: Error rates per participant and strategy

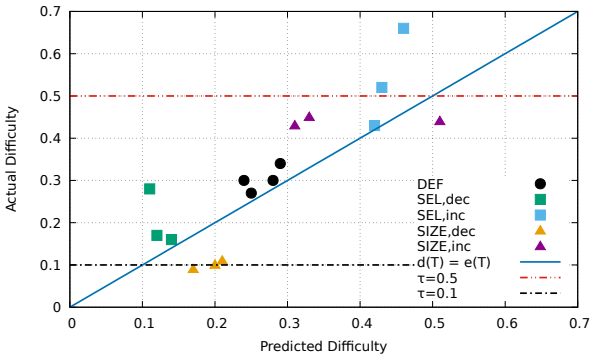


Figure 6: Predicted difficulties $d(T)$ vs the actual error rates $e(T)$.

	SEL		DEF	SIZE	
τ	.10	.50	–	.10	.50
RMSE(e, d)	.10	.13	.04	.09	.11
RMSE(e, τ)	.12	.10	–	.01	.06

Table 4: RMSE between the actual difficulty $e(T)$ and predicted difficulty $d(T)$ as well as target difficulty τ.

fore, we use two-tailed significance testing between SEL and SIZE for all three dimensions. We find that SIZE yields significantly easier C-tests than SEL in terms of error rates ($p = 0.0014$) and Likert feedback ($p = 6 \cdot 10^{-5}$), and observe $p = 0.0394$ for the rankings. For increasing the difficulty, we, however, do not find significant differences between the two strategies. Since both strategies successfully modify the difficulty individually, this motivates research on combined strategies in the future.

We furthermore investigate how well our strategies perform in creating C-tests with the given target difficulty τ. Table 4 shows the RMSE for $e(T)$ and $d(T)$ as well as for $e(T)$ and τ for both strategies. As expected, our difficulty prediction system works best for C-tests generated with DEF as they use the same scheme as C-tests in the training data. Though slightly worse than for DEF, we still find very low RMSE scores for manipulated C-tests. This is especially good when considering that the system's performance on our newly acquired dataset yields and RMSE of 0.21 (cf. section 6). Computing the RMSE with respect to our chosen target difficulties τ yields equally good results for SEL and exceptionally good results for SIZE. Figure 6 displays $d(T)$ in comparison to $e(T)$ for each individual text and strategy. With the exception of $T_2^{\texttt{SEL,inc}}$ and $T_4^{\texttt{SEL,dec}}$, all predictions are close to the optimum (i.e., the diagonal) and also close to the desired target difficulty τ.

In a more detailed analysis, we find two main sources of problems demanding further investigation: First, the difficulty prediction quality when deviating from DEF and second, the increasing ambiguity in harder C-tests. However, it underestimates the $d(T) = 0.11$ for $T_4^{\texttt{SEL,dec}}$ (the same text used in figure 1), for which we found an actual error rate of 0.28. This is due to chains of four successive gaps, such as:

gap g	i_	wh__	w_	a__
solution	is	what	we	are
$d(g)$	0.17	0.22	0.23	0.19
$e(g)$	0.70	0.40	0.10	0.20

As the prediction system has been trained only on DEF-generated C-tests, it underestimates $d(g)$ for cases with limited context. It will be interesting for future work to focus on modeling gap interdependencies in C-tests deviating from DEF.

Another issue we observe is that the gap size strategy might increase the ambiguity of the C-test. In the standard scheme, there is in most cases only a single correct answer per gap. In $T_2^{\texttt{SIZE,inc}}$, how-

ever, the SIZE strategy increased the gap of the word *professional* to its maximal length yielding p__________. One participant answered *popularising* for this gap, which also fits the given context. We carefully checked our datasetfor other ambiguity, but only found one additional case: In T_4, instead of the word *close*, 13 participants out of 30 used *clear* as a modifier of *correspondence*, which both produce meaningful contexts. Given that this case is already ambiguous in the DEF scheme yielding the gap cl___, we conclude that the issue is not severe, but that the difficulty prediction system should be improved to better capture ambiguous cases; for example, by introducing collocational features weighted by their distribution within a corpus into Δ_{inc} and Δ_{dec}.

8 Conclusion

In this work, we proposed two novel strategies for automatically manipulating the difficulty of C-test exercises. Our first strategy selects which words should be turned into a gap, and the second strategy learns to increase or decrease the size of the gaps. Both strategies automatically predict the difficulty of a test to make informed decisions. To this end, we reproduced previous results, compared them to neural architectures, and tested them on a newly acquired dataset. We evaluate our difficulty manipulation pipeline in a corpus-based study and with real users. We show that both strategies can effectively manipulate the C-test difficulty, as both the participants' error rates and their perceived difficulty yield statistically significant effects. Both strategies reach close to the desired difficulty level.

Our error analysis points out important directions for future work on detecting ambiguous gaps and modeling gap interdependencies for C-tests deviating from the default generation scheme. An important observation is that manipulating the gaps' size and position does not only influence the C-test difficulty, but also addresses different competencies (e.g., requires more vocabulary knowledge or more grammatical knowledge). Future manipulation strategies that take the competencies into account have the potential to train particular skills and to better control the competencies required for a placement test. Another strand of research will be combining both strategies and deploying the manipulation strategies in a large scale testing platform that allows the system to adapt to an individual learner over time. A core advantage of our manipulation strategies is that we can work with any given text and thus provide C-tests that do not only have the desired difficulty, but also integrate the learner's interest or the current topic of a language course.

Acknowledgments

This work has been supported by the Hessian research excellence program "Landes-Offensive zur Entwicklung Wissenschaftlich-ökonomischer Exzellenz" (LOEWE) as part of the *a! – automated language instruction* project under grant No. 521/17-03 and by the German Research Foundation as part of the Research Training Group "Adaptive Preparation of Information from Heterogeneous Sources" (AIPHES) under grant No. GRK 1994/1. We thank the anonymous reviewers for their detailed and helpful comments. We furthermore thank the language center of the Technische Universität Darmstadt for their cooperation and Dr. Lisa Beinborn for providing us with the code for our reproduction study.

References

Lisa Beinborn, Torsten Zesch, and Iryna Gurevych. 2014. Predicting the Difficulty of Language Proficiency Tests. *Transactions of the Association for Computational Linguistics*, 2:517–529.

Lisa Marina Beinborn. 2016. *Predicting and manipulating the difficulty of text-completion exercises for language learning*. Ph.D. thesis, Technische Universität Darmstadt.

Raman Chandrasekar, Christine Doran, and Bangalore Srinivas. 1996. Motivations and methods for text simplification. In *Proceedings of the 16th International Conference on Computational Linguistics (COLING): Volume 2*, pages 1041–1044, Copenhagen, Denmark.

C. A. Chapelle. 1994. Are C-tests valid measures for L2 vocabulary research? *Second Language Research*, 10(2):157–187.

Carol A. Chapelle and Roberta G. Abraham. 1990. Cloze method: what difference does it make? *Language Testing*, 7(2):121–146.

Kevyn Collins-Thompson. 2014. Computational assessment of text readability: A survey of current and future research. *International Journal of Applied Linguistics – Special Issue on Recent Advances in Automatic Readability Assessment and Text Simplification*, 165(2):97–135.

EC. 2002. Presidency Conclusions. Barcelona European Council 15 and 16 March 2002. Report SN 100/1/02 REV 1, Council of the European Union.

W. Nelson Francis. 1965. A standard corpus of edited present-day american english. *College English*, 26(4):267–273.

Julia Hancke, Sowmya Vajjala, and Detmar Meurers. 2012. Readability classification for german using lexical, syntactic, and morphological features. In *Proceedings of the 24th International Conference on Computational Linguistics (COLING)*, pages 1063–1080, Mumbai, India.

Jennifer Hill and Rahul Simha. 2016. Automatic generation of context-based fill-in-the-blank exercises using co-occurrence likelihoods and google n-grams. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 23–30, San Diego, CA, USA.

Tadamitsu Kamimoto. 1993. Tailoring the Test to Fit the Students: Improvement of the C-Test through Classical Item Analysis. *Language Laboratory*, 30:47–61.

Adam Kilgarriff, Frieda Charalabopoulou, Maria Gavrilidou, Janne Bondi Johannessen, Saussan Khalil, Sofie Johansson Kokkinakis, Robert Lew, Serge Sharoff, Ravikiran Vadlapudi, and Elena Volodina. 2014. Corpus-based vocabulary lists for language learners for nine languages. *Language Resources and Evaluation*, 48(1):121–163.

Christine Klein-Braley and Ulrich Raatz. 1982. Der C-Test: ein neuer Ansatz zur Messung allgemeiner Sprachbeherrschung. *AKS-Rundbrief*, 4:23–37.

Shibamouli Lahiri. 2014. Complexity of Word Collocation Networks: A Preliminary Structural Analysis. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*, pages 96–105, Gothenburg, Sweden.

Batia Laufer and Paul Nation. 1999. A vocabulary-size test of controlled productive ability. *Language Testing*, 16(1):33–51.

John Lee and Mengqi Luo. 2016. Personalized exercises for preposition learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*, pages 115–120, Berlin, Germany.

David D. Lewis, Yiming Yang, Tony G. Rose, and Fan Li. 2004. RCV1: A New Benchmark Collection for Text Categorization Research. *Journal of Machine Learning Research*, 5(Apr):361–397.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL): Short Papers*, volume 2, pages 85–91, Vancouver, Canada.

Naiara Perez and Montse Cuadros. 2017. Multilingual call framework for automatic language exercise generation from free text. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL): Software Demonstrations*, pages 49–52, Valencia, Spain.

Ildikó Pilán, Elena Volodina, and Richard Johansson. 2014. Rule-based and machine learning approaches for second language sentence-level readability. In *Proceedings of the Ninth Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 174–184, Baltimore, MD, USA.

Nils Reimers and Iryna Gurevych. 2017. Reporting Score Distributions Makes a Difference: Performance Study of LSTM-networks for Sequence Tagging. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 338–348, Copenhagen, Denmark.

Günther Sigott. 1995. The C-Test: Some Factors of Difficulty. *AAA: Arbeiten aus Anglistik und Amerikanistik*, 20(1):43–53.

Günther Sigott. 2006. How fluid is the c-test construct? In *Der C-Test: Theorie, Empirie, Anwendungen – The C-Test: Theory, Empirical Research, Applications*, Language Testing and Evaluation, pages 139–146. Frankfurt am Main: Peter Lang.

Bernard Spolsky. 1969. Reduced Redundancy as a Language Testing Tool. In G.E. Perren and J.L.M. Trim, editors, *Applications of linguistics*, pages 383–390. Cambridge: Cambridge University Press.

Wilson L. Taylor. 1953. "Cloze Procedure": A New Tool for Measuring Readability. *Journalism Bulletin*, 30(4):415–433.

Sowmya Vajjala and Detmar Meurers. 2014. Assessing the relative reading level of sentence pairs for text simplification. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*, pages 288–297, Gothenburg, Sweden.

Vladimir N. Vapnik. 1998. *Statistical Learning Theory*. New York: Wiley.

Lev Vygotsky. 1978. *Mind in society: The development of higher psychological processes*. Cambridge: Harvard University Press.

Michael Wojatzki, Oren Melamud, and Torsten Zesch. 2016. Bundled gap filling: A new paradigm for unambiguous cloze exercises. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications (BEA)*, pages 172–181, San Diego, CA, USA.

Torsten Zesch and Oren Melamud. 2014. Automatic generation of challenging distractors using context-sensitive inference rules. In *Proceedings of the*

Ninth Workshop on Innovative Use of NLP for Building Educational Applications (BEA), pages 143–148, Baltimore, MD, USA.