

DISTRACTOR GENERATION FOR LEXICAL QUESTIONS USING LEARNER CORPUS DATA

NIKITA LOGIN

Independent researcher

LOGIN, Nikita: Distractor Generation for Lexical Questions Using Learner Corpus Data. *Journal of Linguistics*, 2023, Vol. 74, No 1, pp. 345 – 356.

Abstract: Learner corpora with error annotation can serve as a source of data for automated question generation (QG) for language testing. In case of multiple choice gap-fill lexical questions, this process involves two steps. The first step is to extract sentences with lexical corrections from the learner corpus. The second step, which is the focus of this paper, is to generate distractors for the retrieved questions. The presented approach (called DisSelector) is based on supervised learning on specially annotated learner corpus data. For each sentence a list of distractor candidates was retrieved. Then, each candidate was manually labelled as a plausible or implausible distractor. The derived set of examples was additionally filtered by a set of lexical and grammatical rules and then split into training and testing subsets in 4:1 ratio. Several classification models, including classical machine learning algorithms and gradient boosting implementations, were trained on the data. Word and sentence vectors from language models together with corpus word frequencies were used as input features for the classifiers. The highest F1-score (0.72) was attained by a XGBoost model. Various configurations of DisSelector showed improvements over the unsupervised baseline in both automatic and expert evaluation. DisSelector was integrated into an open-source language testing platform LangExBank as a microservice with a REST API.

Keywords: distractor generation, learner corpora, automated question generation

1 INTRODUCTION

According to (Granger 2008), learner corpora can be extremely useful for e-learning. One possible use of learner corpora is to serve as a source of data for exercise generation, reducing the time needed for test item preparation. In the described system LangExBank¹ learner corpus REALEC (Vinogradova – Lyashevskaya 2022) was implemented as a source of text for gap-fill exercises. One of the main challenges in automated test creation is distractor generation (DG below), as the system usually needs to provide multiple independent outputs for one input. This work is aimed at solving this challenge by using a combination of semantic-based and supervised learning-based methods.

¹ <https://github.com/lcl-hse/LangExBank>

2 RELATED WORK

According to the review of Kurdi et al. (2020), general and language gap-fill exercises are among the most popular types of automatically generated questions. However, few works concern DG for this type of questions – while QG can be quite easy (an algorithm needs only to insert a gap at a certain place in the input sentence), DG can be tricky as the generated outputs need to be both not *too good* and not *too bad* in terms of language.

In Kumar et al. (2015) gap-fill questions in the domain of high school-level Biology were generated in a three-stage approach. First, topic-related sentences were selected by a NN-based topic model, then a gap was placed by an SVM trained on crowdsourced data and, finally, distractors were selected using the factors of semantic similarity according to a Word2Vec model (Mikolov et al. 2013), syntactic similarity (Dice Coefficient), and contextual fit (language model output probability). In 48.69% of cases, generated distractors were considered appropriate by student evaluators.

Jiang and Lee (2017) present an unsupervised DG approach for Chinese gap-fill lexical questions. Distractor candidates were proposed on the basis of Part-of-Speech (POS) tag and the same difficulty level, and either spelling similarity, word co-occurrence (as attested by PMI) or semantic similarity (as attested by Word2Vec). Then candidates appearing in author-curated Wikipedia corpus in contexts similar to that of the input sentence were filtered out. The maximum plausibility score (46.6%) in human evaluation was obtained by a combination of POS tag-based and similarity-based distractor generation techniques.

In Sakaguchi et al. paper (2013) learner corpus data was used to predict distractors for gap-fill exercises with English verbs. The system implemented a classification approach: distractor occurrences in corpus mistakes were considered positive examples, word lemmas and syntactic dependency types were used as features, and the verb vocabulary was used as the distractor candidate set. The system produced only one distractor for a given input.

3 SYSTEM OVERVIEW

Our system includes two major components: the LangExBank testing platform and Testmaker – the tool for extracting test questions (Vinogradova 2019).

3.1 Data Source: REALEC Corpus

REALEC (Russian Error-Annotated Learner English Corpus) is a corpus consisting of essays written by students of HSE University in the format of IELTS writing exam. As of May 2022, it contained 19,079 texts, including 9,494 argumentative essays, 9,584 graph descriptions and 1 short text serving as an example of annotation schema. The size of corpus is 4,925,478 words with 60,001

unique types. The interface and annotation format of corpus is based on the BRAT (Stenetorp et al. 2012) engine. REALEC annotation includes error span selection, classification (based on a broad hierarchical system of tags) and correction.

3.2 Testmaker

Testmaker can be applied both on LangExBank platform and as a separate Python package. It downloads a selected subset of REALEC data and creates a set of gap-fill exercises for the error tags specified by the user. During the test generation, Testmaker creates gaps in places of spans of specified error types, while replacing all other errors with annotator-suggested corrections. Each output sentence contains only one gap, and the corrections are used as the right answers. Testmaker provides different options of exercise generation: a user can add to the test item two sentences from the source text surrounding the sentence with the error span. A user can also decide in which manner exercises should be generated: an exercise from one sentence or an exercise for each mistake in the sentence.

3.3 LangExBank testing platform

LangExBank is a platform for online testing that supports two types of exercises: corpus-based gap-fill questions and IELTS-like Listening, Reading, and Writing exercises. It is freely available from the Github repository and includes a subplatform for reference material.

4 DATASET

The dataset for distractor generation consisted of 6,804 lexical (formed from errors with the *Choice of lexical item* annotation tag) gap-fill exercises retrieved by Testmaker. Contexts of surrounding sentences were included, and multiple gap-fill items were generated from one sentence. After retrieval, each item was equipped with a set of distractor candidates – words from lexical error spans where *Answer* was also used as a correction. This resulted in 66,647 triplets consisting of a gap-fill Sentence (*S*), a right answer (*Ans*) and a candidate distractor (*d*).

The constructed dataset was filtered according to the following rules:

1. *Ans* and *d* should appear in vocabulary of both Word2Vec and Brown Corpus (Francis – Kucera 1979).
2. *Ans* and *d* should have at least one common possible POS tag in Brown Corpus vocabulary.
3. Both *Ans* and *d* should not bear an auxiliary POS tag (CC, CD, DT, EX, IN, LS, MD, PT, POS, PRP, PRP\$, RP, SYM, TO, UH, WDT, WO, WP\$ or WRB) in context of the sentence.
4. The original correction in sentence should not be a *than to that* or vice versa.
5. *d* should not be a one-syllable adjective after a comparative marker.
6. *d* should not be an auxiliary verb.

7. d should not copy the word immediately preceding or succeeding the gap.
8. d should not be a negation of Ans or vice versa.

After filtering, 12,679 triplets remained, from which the first 3,003 were annotated manually – each example was assigned a binary label indicating whether d was a plausible distractor. Additionally, 76 gap-fill items from the source set were saved for the final evaluation.

NLTK (Bird et al. 2009) package was used for contextual POS tagging and for obtaining the data from the Brown Corpus, as it is fast on big amounts of input examples and is therefore suitable for production environments. A Word2Vec model trained on non-normalized 2021 Wikipedia dump from NLPL Repository (Fares et al. 2017) was used for filtering and later was included in the feature extraction process. The Word2Vec implementation from Gensim (Řehůřek – Sojka 2010) package was used. The process of dataset construction is displayed in Fig. 1.

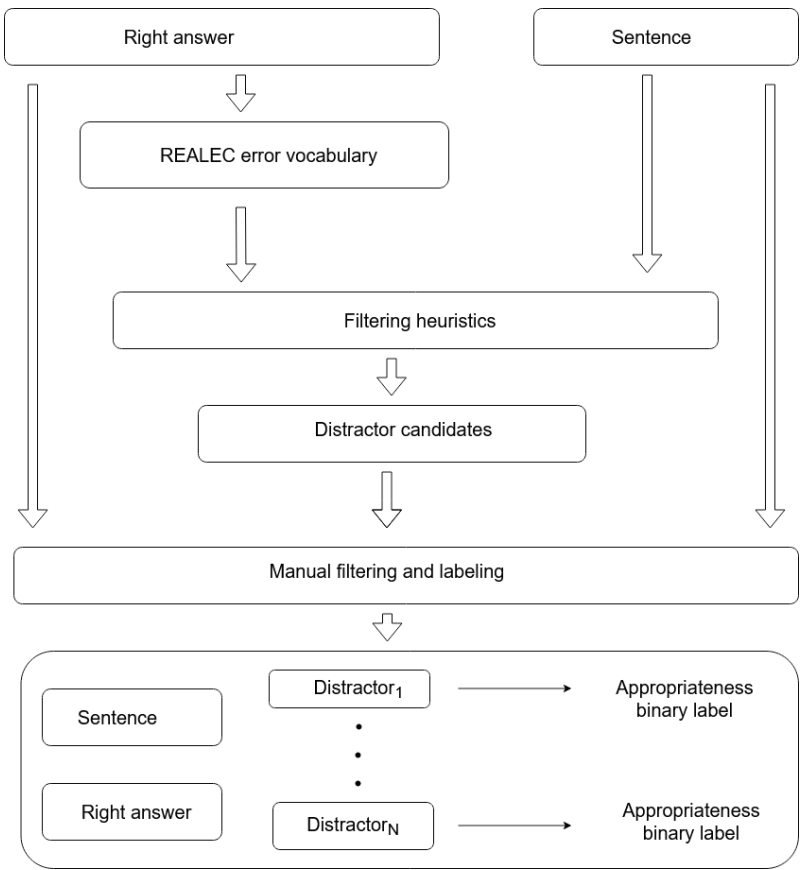


Fig. 1. Dataset construction pipeline

5 MANUAL LABELLING

3,003 distractors from 852 sentences were manually labelled as plausible or implausible. During labelling 69 sentences with 165 distractors were filtered out as implausible due to one of the following factors:

- The original correction was of a properly used word.
- Both *Ans* and *d* were plausible in context of *S* (1).
- *Ans* was an auxiliary word not filtered out automatically.
- *S* had an unclear meaning (2).
- The error consisted in the use of a full form instead of an abbreviation or vice versa.

Examples (1) – (2) illustrate some of the reasons for exclusion (error and correction are **in bold font**, error is ~~crossed out~~).

(1) *Sport centers and sporting goods ~~shops~~ **stores** can reduce prices for sport equipment or for club cards to do different sports.*

(2) *On top of that, parents should tell or show to what can ~~prevent~~ **lead** such activities.*

Example (3) illustrates how the decision whether a distractor was plausible was taken during annotation. The correction (right answer) is **bold**, “too bad” (marginal) distractors are ~~crossed out~~, and “too good” (correct) distractors are underlined:

(3) *The bar chart illustrates the average amount of minutes which were spent by English people doing sports activities in 2012. At first glance, we can _____ that the most active part of the English population in that period was men, while the least active was women. The men aged from 16 to 24 years showed enormous activity, which reached a peak at the point of 282.1 minutes in 2012.*

- a. *see*
- b. admit
- c. say
- d. watch
- e. conduct
- f. find
- g. ~~face~~
- h. observe

After manual filtering, the labelled dataset consisted of 2,837 examples² (1,339 appropriate and 1,498 inappropriate) that corresponded to 781 sentences (1.71 appropriate and 1.92 inappropriate distractors per sentence on average).

² One sentence (corresponding to one example) was filtered out during dataset processing by feature extraction algorithm, as the correction word did not appear in corpus in any other contexts.

6 METHOD

As the dataset was relatively small, DG was viewed as a classification problem, and gradient boosting and traditional machine learning approaches were used. The DG pipeline was implemented in three stages: first, our algorithm proposes distractor options for the given sentences, then input examples (consisting of S , d and Ans) are transformed into feature vectors, and, finally, plausible variants are filtered by a supervised binary classifier. During training, distractor candidates from the labelled dataset were used, and distractor suggestion in inference mode is described in Section 6.3.

6.1 Feature Extraction

For the feature extraction, we used a concatenation of Word2Vec and BERT (Devlin et al. 2019) embeddings and corpus frequency counts (Fig. 2). BERT-based model (Devlin et al. 2019) available from Transformers package (Wolf et al. 2020) was used for extracting BERT embeddings. The full set of features included:

- BERT vector of [MASK] token in place of the error span;
- Averaged BERT vector of S with gap replaced by Ans ;
- Word2Vec vector of Ans ;
- Word2Vec vector of d ;
- Frequency of Ans in corpus corrections;
- Frequency of d misuses instead of Ans in corpus;
- Frequency of Ans in the whole corpus;
- Frequency of d in the whole corpus.

6.2 Distractor Classification

The following models were trained on extracted features: K-Nearest Neighbors, SVM, Decision Tree, Logistic and Ridge Regression, Random Forest, Naive Bayesian Classifier, Gradient Boosting, AdaBoost, CatBoost (Dorogush et al. 2018), LightGBM (Ke et al. 2017) and XGBoost (Chen – Guestrin 2016). For all models we used their implementations from Scikit-Learn (Pedregosa et al. 2011) package, except for XGBoost CatBoost and LightGBM which were taken from the eponymous Python packages.

The labelled set was split into training and testing parts in 4:1 ratio using source sentence indices (and not example indices), so no example with S from training set would be encountered in testing.

6.3 Distractor Suggestion on Inference

For the inference mode, additional distractor candidates were extracted from Word2Vec. First, cosine similarities between the given word from vocabulary and all the other words that could have the same POS tag were calculated. Then, this set of words was filtered according to rules described in Section 4.

Calculating cosines between these many vectors is computationally expensive, which is not suitable for a production setting. To optimize inference speed, we saved extracted options vocabulary to a file. The option sets were sorted in the following way: first came the words from corpus (sorted by conditional frequency of appearing in corpus error spans, $F(Ans|d)$), and then came words from the Word2Vec model (sorted by cosine similarity). At inference our algorithm extracts top K (parameter provided by user) distractor candidates from the described set. The full inference pipeline is shown in Fig. 3.

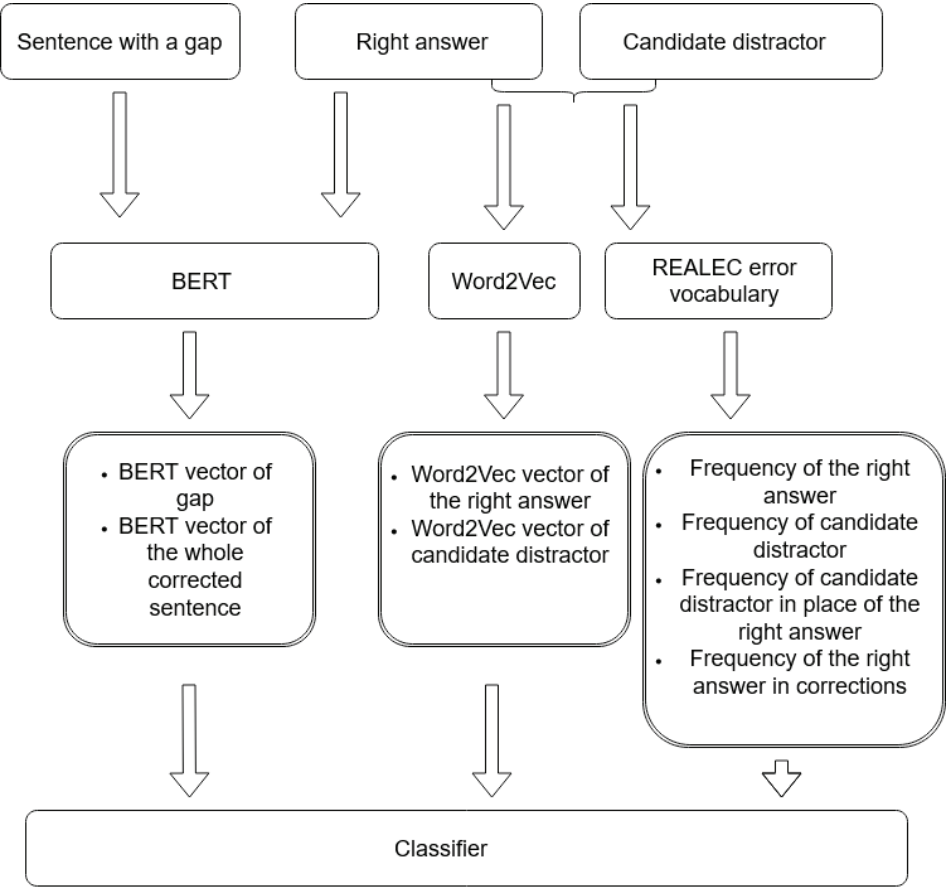


Fig. 2. Feature extraction pipeline

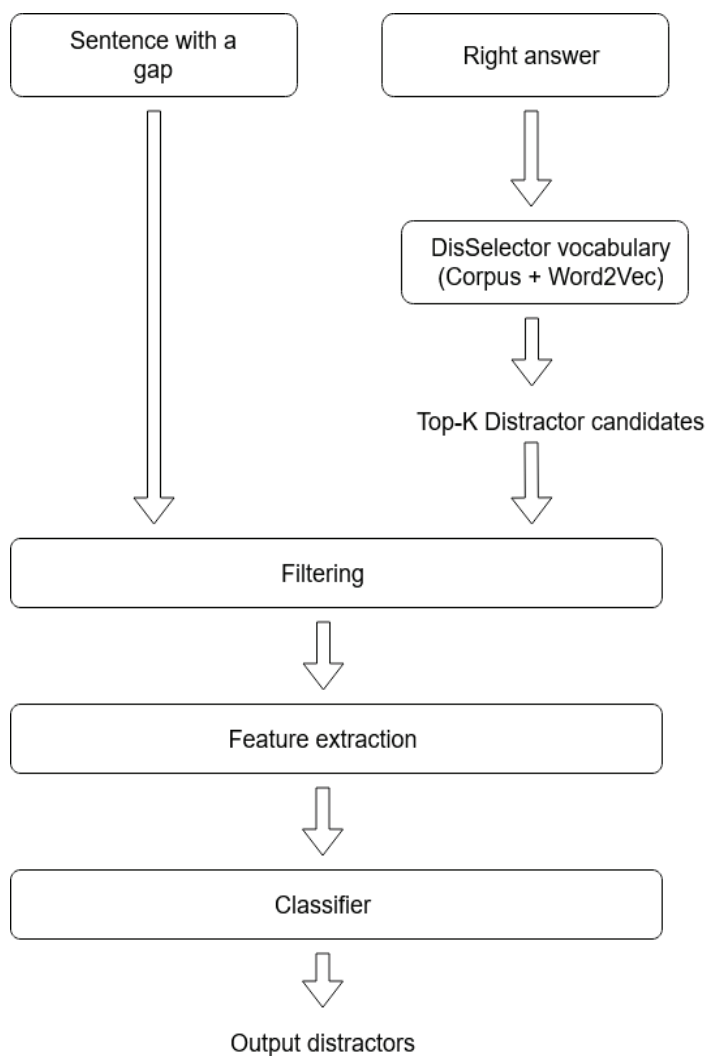


Fig. 3. Inference pipeline

7 EVALUATION

7.1 Automatic evaluation

Distractor classification component was evaluated automatically. As a baseline, we used models trained on the same data, but without manual labelling – only the

original correction was labelled *Appropriate*, as in (Sakaguchi et al. 2013). Class imbalance was encountered in the baseline setup (81.07% examples of *Inappropriate* class), while this was not so for the manually labelled set (52.78% *Inappropriate* examples). The author has also tried training the classifiers only on embeddings (*VecsOnly*), only on frequency features (*FreqsOnly*), and by individually excluding each of the features (*FeatDrop*). Models trained on the whole set of features received *AllFeats* postfix.

7.2 Human evaluation

The inference quality of the whole pipeline was evaluated manually. As a baseline, the author used distractor suggestion component without a classifier on top. The annotator (an EFL expert) was asked to label each of the suggested distractors as “Too good”, “Too bad” or “Appropriate”. The author used different values of *K* (from 3 to 20) for top classification models (from each of the *AllFeats*, *VecsOnly*, *FreqsOnly* and *FeatDrop* settings) in terms of F1-Score.

8 RESULTS

Results of the automatic evaluation for the best models from each of the settings (*AllFeats*, *VecsOnly*, *FreqsOnly*, *FeatDrop* and baseline) are presented in Tab. 1. The best F1-score (0.72) was achieved by a XGBoost model trained on all features. The model trained exclusively on frequency features shows performance comparable to models trained on word and sentence vectors. Baseline models show far lower performance, which illustrates the benefit of manual annotation of distractor plausibleness.

Model	Setting	Dropped feature	F1-Score	Accu- racy	Precision	Recall
XGBoost	AllFeats	—	0.72	0.72	0.69	0.75
Random Forest	FreqsOnly	word and sentence vectors	0.69	0.71	0.70	0.67
CatBoost	VecsOnly	Word frequencies	0.72	0.72	0.68	0.76
CatBoost	FeatDrop	Word2Vec vector of <i>Ans</i>	0.73	0.73	0.70	0.75
Gradient Boosting	AllFeats (unlabelled baseline)	—	0.43	0.83	0.53	0.36

Tab. 1. Top results of automatic evaluation of distractor classifiers (with baseline)

The top results of the manual evaluation are presented in Tab. 2.³ The highest percentage of *Appropriate* distractors is demonstrated by a XGBoost model trained on all features, with K=3. There is a visible trend that the quality of distractors is decreasing with the increase of K.

Model	Setting	Dropped feature	K	% Appropriate	% Too bad	% Too good	N Appropriate	N distractors
XGBoost	AllFeats	—	3	59.87	27.63	12.5	1.4605	2.5132
CatBoost	VecsOnly	—	3	57.46	27.19	12.72	1.3158	2.2763
XGBoost	AllFeats	—	4	56.14	32.68	11.18	1.7368	3.2368
CatBoost	FeatDrop	Word2Vec vector of <i>Ans</i>	3	55.04	30.48	14.47	1.4605	2.6711
CatBoost	VecsOnly	—	4	53.62	31.47	12.28	1.5789	2.9079
XGBoost	All Feats	—	5	53.55	35.77	10.68	1.9342	3.8947
Baseline		—	3	53.07	33.33	13.6	1.5921	3

Tab. 2. Top results of manual evaluation of the whole pipeline

XGBoost model trained on all features with K=4 was selected as the default for DisSelector (as it had the best quality among models producing no less than 3 distractors on average). Its output is presented in Example (4) (with the same highlighting as in Example 3):

- (4) *Manufactured goods take one quarter of the pie chart. The second chart _____ that food products and manufactured goods are the most popular goods transported by road. However, food products have the largest proportion (30%).*
- a. ***shows***
- b. *illustrates*
- c. *~~gives~~*
- d. *pictures*
- e. *presents*

9 CONCLUSION

This paper describes an implemented solution for generating distractors for gap-fill lexical questions sourced from annotated errors in the learner corpus. After manual evaluation, XGBoost classifier with 4 input distractor candidates was chosen as the default distractor generation method. Manual annotation of distractor dataset has resulted in a considerable performance improvement over the baseline with no additional annotation. This improvement is expected to increase with the enlargement

³ The percentages do not always exactly add up to 100 as the number of generated distractors may vary among sentences. The final values were calculated as averages of percentages in each sentence.

and quality enhancement (by using labels from multiple independent annotators) of the labelled distractor dataset.

ACKNOWLEDGEMENTS

The research has been supported by the “Development of second language acquisition models according to existing theories using experiment automation methods on REALEC and another learner corpora data” project (TA-102).

The author would like to thank Olga Vinogradova and Anna Viklova.

Notes:

The demo of DisSelector is accessible at http://langexbank-fikl.ru/distractor_api/docs.

The source code for DisSelector is accessible at <https://github.com/nicklogin/DistractorSelector>.

The full (containing all results) versions of Tab. 1 and 2 are accessible at https://github.com/nicklogin/DistractorSelector/tree/main/full_tables.

References

Bird, S., Klein, E., and Loper, E. (2009). *Natural language processing with Python: analyzing text with the natural language toolkit*. Sebastopol, CA: O'Reilly Media, Inc, 509 p.

Chen, T., and Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 785–794, San Francisco, CA. Accessible at: <http://doi.acm.org/10.1145/2939672.2939785>.

Devlin, J., Chang, M. W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Accessible at: <http://dx.doi.org/10.18653/v1/N19-1423>.

Dorogush, A., Ershov, V., and Gulin, A. (2018). CatBoost: gradient boosting with categorical features support. *ArXiv preprint*, 7 p. Accessible at: <https://arxiv.org/abs/1810.11363>.

Fares, M., Kutuzov, A., Oepen, S., and Velldal, E. (2017). Word vectors, reuse, and replicability: Towards a community repository of large-text resources. In *Proceedings of the 21st Nordic Conference on Computational Linguistics*, pages 271–276, Gothenburg, Sweden. Accessible at: <https://aclanthology.org/W17-0237/>.

Francis, W., and Kucera, H. (1979). *Brown Corpus*. Providence, Rhode Island: Department of Linguistics, Brown University. Accessible at: <http://korpus.uib.no/icame/manuals/BROWN/INDEX.HTM>.

Granger, S. (2008). Learner Corpora. In A. Lüdeling – M. Kyto (eds.): *Corpus Linguistics. An International Handbook*. Volume 1. Berlin: Walter de Gruyter, pages 259–275.

Jiang, S., and Lee, J. (2017). Distractor Generation for Chinese Fill-in-the-blank Items. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 143–148, Copenhagen, Denmark. Accessible at: <http://dx.doi.org/10.18653/v1/W17-5015>.

Ke, G., Meng, Q., Finley, T., Wang, T., Chen, W., Ma, W., Ye, Q., and Liu, T. Y. (2017). LightGBM: A Highly Efficient Gradient Boosting Decision Tree. In *Proceedings of the 31st Internal Conference on Neural Information Processing Systems (NIPS 2017)*, pages 3149–3157, Long Beach, CA. Accessible at: <https://dl.acm.org/doi/10.5555/3294996.3295074>.

Kumar, G., Banchs, R., and D’Haro, L. (2015). Automatic fill-the-blank question generator for student self-assessment. In *Proceedings of 2015 IEEE Frontiers in Education Conference (FIE)*, pages 1–3, El Paso, TX. Accessible at: <https://doi.org/10.1109/FIE.2015.7344291>.

Kurdi, S. (2020). A Systematic Review of Automatic Question Generation for Educational Purposes. *International Journal of Artificial Intelligence in Education*, 30(1), pages 121–204.

Liu, M., Rus, V., and Liu, L. (2018). Automatic Chinese Multiple Choice Question Generation Using Mixed Similarity Strategy. *IEEE Transactions on Learning Technologies*, 11(2), pages 193–202.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013). Efficient Estimation of Word Representations in Vector Space. *ArXiv preprint*, 12 p. Accessible at: <https://arxiv.org/abs/1301.3781>.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12, pages 2824–2830.

Řehůřek, R., and Sojka, P. (2010). Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 46–50, Malta. Accessible at: <http://dx.doi.org/10.13140/2.1.2393.1847>.

Sakaguchi, K., Arase, Y., and Komachi, M. (2013). Discriminative Approach to Fill-in-the-Blank Quiz Generation for Language Learners. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 238–242, Sofia, Bulgaria. Accessible at: <https://aclanthology.org/P13-2043/>.

Stenetorp, P., Pyysalo, S., Topic, G., Ohta, T., Ananiadou, S., and Tsujii, J. (2012). brat: a Web-based Tool for NLP-Assisted Text Annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France. Accessible at: <https://aclanthology.org/E12-2021/>.

Vinogradova, O. (2019). To automated generation of test questions on the basis of error annotations in EFL essays: A time-saving tool? In S. Götz – J. Mukherjee (eds.): *Learner Corpora and Language Teaching*. Volume 29. Amsterdam, Netherlands: John Benjamins, pages 29–48.

Vinogradova, O., and Lyashevskaya, O. (2022). Review of Practices of Collecting and Annotating Texts in the Learner Corpus REALEC. In P. Sojka – A. Horák – I. Kopeček – K. Pala (eds.): *Text, Speech and Dialogue*. 25th International Conference, TSD 2022, Brno, Czech Republic, September 6–9, 2022, *Proceedings*. Cham, Switzerland: Springer Nature Switzerland AG, pages 77–88.