EdTec-QBuilder: A Semantic Retrieval Tool for Assembling Vocational Training Exams in German Language

Alonso Palomino[1,3], **Andreas Fischer**[2], **Jakub Kuzilek**[1], **Jarek Nitsch**[4], **Niels Pinkwart**[1], **Benjamin Paaßen**[1,3]

[1]Educational Technology Lab, DFKI, `alonso.palomino@dfki.de`
[2]Forschungsinstitut Betriebliche Bildung (f-bb), `andreas.fischer@f-bb.de`
[3]Universität Bielefeld, `benjamin.paasen@uni-bielefeld.de`
[4]bfz gGmbH

Abstract

Selecting and assembling test items from a validated item database into comprehensive exam forms is an under-researched but significant challenge in education. Search and retrieval methods provide a robust framework to assist educators when filtering and assembling relevant test items. In this work, we present EdTec-QBuilder[1], a semantic search tool developed to assist vocational educators in assembling exam forms. To implement EdTec-QBuilder's core search functionality, we evaluated eight retrieval strategies and twenty-five popular pre-trained sentence similarity models. Our evaluation revealed that employing cross-encoders to re-rank an initial list of relevant items is best for assisting vocational trainers in assembling examination forms. Beyond topic-based exam assembly, EdTec-QBuilder aims to provide a crowdsourcing infrastructure enabling manual exam assembly data collection, which is critical for future research and development in assisted and automatic exam assembly models.

1 Introduction

Examination forms consisting of validated, high-quality test items are a crucial tool for estimating the current competence of students in education. While much research has covered the task of *generating* such items, relatively little research has focused on *assembling* new exams from a database of existing, validated test items (Kurdi et al., 2020). Exam assembly is a challenging task on its own as exams need to cover all skills in a given topic comprehensively, at multiple levels of difficulty, and such that the resulting exam is psychometrically valid (Armendariz et al., 2012; Lane et al., 2015; Fischer and Neubert, 2015).

To support educational experts in formative exam assembly, we present EdTec-QBuilder[1], a service and tool to search a database of (validated and high-quality) test items and assemble them for an exam. Beyond its practical utility for exam assembly, our tool is also intended to be a crowdsourcing platform to collect data on exam assembly processes and thus provide data for future research on assisted and automated exam assembly.

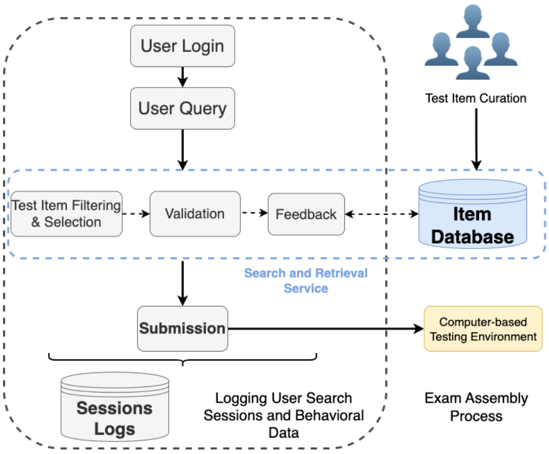

User Login
User Query
Test Item Curation
Test Item Filtering & Selection
Validation
Feedback
Item Database
Search and Retrieval Service
Submission
Computer-based Testing Environment
Sessions Logs
Logging User Search Sessions and Behavioral Data
Exam Assembly Process

Figure 1: The operational flowchart scenario of EdTec-QBuilder.

Figure 1 summarizes the operational flow of our tool. After logging in, users query, filter, and annotate relevant items from a validated item database. While selecting, confirming, and submitting exam forms, users assess the quality of their search experience. Finally, search sessions are stored for analysis and exam assembly model development.

To implement EdTec-QBuilder's semantic search functionality, we evaluated the performance of eight search and retrieval strategies in combination with the top twenty-five trending pre-trained sentence similarity models (representing over 176 different combinations, overall; refer to Section 4). We selected the best-performing strategy and pre-trained model from our evaluation to include it as a core search and retrieval module. Our work makes the following key contributions:

[1] The EdTec-QBuilder system demo version is publicly accessible at: `https://www.dfki.de/kiperweb/about.html`

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations), pages 26–35
June 16-21, 2024 ©2024 Association for Computational Linguistics

- A new perspective to tackle the topic-based exam assembly task, framed as an information retrieval problem.
- A new data resource to study exam assembly[2].
- A system demonstration of the deployed service and tool used by vocational educators[1].
- An extensive evaluation and performance analysis of popular pre-trained sentence similarity models for the exam assembly task.

The evaluation and selection of the best-performing core semantic search functionality can be found in Section 4. The implementation and features of EdTec-QBuilder are described in Section 5. Finally, we discuss future directions and opportunities for improvement in Section 6.

2 Related Work

Research on assisted and automated exam assembly can be traced back to the EVALING system (Fairon, 1999), a platform that employed rule-based and finite state transducers to compile and update language proficiency exams from a question bank. Piton-Gonçalves and Aluísio (2012) presented a multidimensional adaptive test architecture and system that selects test items based on a student's profile and previous performance. Qin et al. (2019) utilized named entity recognition to build a knowledge graph of skills, which a recommendation method harnessed to suggest personalized interview questions. Sangodiah et al. (2016) used text classification methods to estimate exam difficulty from question bank's items via Bloom's taxonomy categories. Han (2018) introduced selection methods for complying with test item criteria for automated item selection, fostering adaptive learning and individualized learning. Ruan et al. (2019) developed a dialog agent to teach factual knowledge of science and safety. When contrasting system usage with a flashcard system, students increased their response accuracy significantly when using the agent. Cole et al. (2020) leveraged classic natural language processing methods with a cluster-based test item generation approach aligned to a dashboard that enables test designers to select generated questions from the available clusters. Datta et al. (2021) tackled the task of automatically generating an interview question plan from the applicant's resume, harnessing knowledge graphs and integer programming methods. Upadhyay et al. (2023) employed large language models to develop a suite for automated item generation and exam assembly. However, the suite delegates the assembly process to human experts to ensure the test items' quality.

Most of the prior research focuses exclusively on test item generation, utilizing either classical NLP methods such as text classification and topic modeling (Brown et al., 2005; Mitkov et al., 2006; Rus et al., 2011; Heilman and Smith, 2010; Chali and Hasan, 2015), or deep neural networks and transformer-based architectures (Du et al., 2017; Chan and Fan, 2019; Tuan et al., 2020; Qu et al., 2021; Yoshimi et al., 2023). However, in real scenarios, where contextual knowledge, quality control, and topic alignment with curricular standards are necessary for effective student skill development, it is not sufficient to generate items; we also need to assemble them to exams that are psychometrically suited to test the competencies they ought to assess. Therefore, in contrast to prior work, our contribution focuses on assisted exam assembly rather than item generation. Further, we conceptualize exam assembly as an information retrieval problem.

3 Methods

EdTec-Qbuilder aims to assist in tackling the topic-based exam assembly task. We formalize the task as an information retrieval problem. Given a query $q \in Q$ – e.g. a text describing the topic of a exam – and a database of possible items $B = \{x_1, \ldots, x_N\}$, we wish to compute similarity scores $s_1 = S(q, x_1), \ldots, s_N = S(q, x_N)$ for some similarity function $S : Q \times B \to \mathbb{R}$. Based on these scores, we rank all possible items from most similar to least similar and let an educational expert assemble an exam from the ranked list. In other words, we support educational experts in exam assembly by assisting them in searching a large item database according to their query (refer to Figure 1). Therefore, the similarity function S needs to correspond to educational experts' notion of relevance. In other words, S must assign higher similarity values to the items experts want to include in their exam. This is precisely the criterion we evaluate in Section 4.

3.1 Item Database

The item database B for our investigation has been provided by the bfz group, one of the largest voca-

[2] The public data fold available under CC BY license at: https://www.dfki.de/kiperweb/about.html#dataset

tional training providers in Germany. The database consists of 2,812 test items across 34 in-demand vocational training topics. For the purpose of a freely available version of our tool (the original 2,812 items are proprietary) and to increase the amount of training data, we augmented the item database with another 2,812 items generated via ChatGPT3.5[23]. All generated items were manually inspected, and duplicates/low-quality items were removed. The remaining items form the basis for our tool's openly available demo version.

Table 1a displays the most prominent topics in the item database, with their three most used terms. Specifically, the topics "Professional Counseling/Learning Skills/Self-assessment", "Business Knowledge", "Communication/Negotiation and Etiquette in the Workplace", "Information Technology Competence" and "Presentation and Visualization Techniques" cover 2,329 items, meaning 41.41% of all items. The average length of the questions is 17.76 words. Overall, the most used terms throughout the collection correspond to "company", "product" and "important". To estimate the difficulty of 2,812 test items of the bfz data datafold, we fitted a 1parameter item response theory model to the answers of prior respondents. Table 1b shows the distribution of the resulting difficulty values, grouped into three levels: easy, medium, and hard. We observed that 78.91% of test items have a medium difficulty, 20.98% were easy to answer, and only 0.10% were hard. Table 1c outlines the question type distribution in the item database. We observed that 64.83% are multiple and single choice test item types, whereas other formats are less frequent.

4 Experiments and Analysis

Our experiments aim to evaluate the capability of various (semantic) search strategies to retrieve matching test items for typical queries in our item database[4]. Due to the lack of historical search data and insufficient human annotation capacity that would allow us to obtain ground-truth data, we opted to use an automated relevance labeling approach. Using the following approach, we built a new synthetic TREC-style (Voorhees et al., 2005; Teufel, 2007; Voorhees et al., 2022) data set of queries and corresponding relevancy items.

Testbed We randomly generated synthetic queries by selecting 15 syntactically different synonyms from the 34 available topic dimensions of the item database (refer to Table 1a). By aggregating the top 100 agreeing search results of 25 trending multilingual and German sentence similarity models (refer to Appendix A.1), with the trec-tools library (Palotti et al., 2019), we constructed a synthetic test dataset for evaluating our experiments. To attach relevance judgments to the elements of the test dataset, we calculated the average similarity score with the available pre-trained models for each query and item pair. Pairs with an average similarity score between 0.60 and 0.75 were labeled as moderately relevant; pairs above 0.75 were labeled as highly relevant. We acknowledge that this labeling strategy could introduce a bias as the assumed ground truth is influenced by the same language models later used as part of the search strategies. However, the labeling was obtained by averaging the similarity scores of all language models, thus reducing the bias toward any single model and enhancing the robustness of the obtained labeling.

Evaluation We employed the ranx library (Bassani, 2022) to calculate standard information retrieval metrics as an evaluation method to identify and select the best-performing strategy to include as a search module in our tool. In addition to Precision, Recall, F1, and mean average precision (MAP), we used normalized discounted cumulative gain (nDCG) and mean reciprocal rank (MRR). The former expresses the relevance of documents ranked at the top, whereas the latter expresses relevant documents' (reciprocal) rank. Both metrics are better if higher.

Search and retrieval strategies We tested eight different search and retrieval strategies:

1. BM25: As a baseline method, we fitted a standard BM25 model (Řehůřek and Sojka, 2010). BM25 is a probabilistic model that, given a query, calculates term and inverse document frequencies to retrieve relevant items.
2. LM+ANN: We employed 25 trending multilingual and German sentence similarity pre-trained language models (LM) (see Appendix A.1) to approximate the top nearest neighbors (ANN) with the NMSLIB library (Boytsov and Naidan, 2013).
3. TF-IDF Weighted Average: To improve the se-

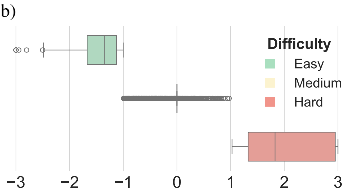
[3] https://chat.openai.com/

[4] The experiments were conducted on a macOS with an ARM64 processor, 32 GB of memory, and 12 physical cores.

(a)

	Top 5 Topics	Avg. Terms per Item	Unique Terms	Top 3 Terms	Dist.
	Professional Counseling, Learning Skills Self-assessment	11.96	2495	learn (140) work (93) professional (91)	671
	Business Knowledge	18.25	2495	company (328) product (210) business plan (199)	586
	Workplace Communication, Negotiation & Etiquette	18.75	2490	customer (184) product (182) communication (177)	514
	Information Technology Competence	17.01	2064	internet (58) digital (47) program (43)	322
	Presentation & Visualization Techniques	17.45	1014	presentation (241) audience (184) represent (86)	236
Total	34	17.76	15023	company (1017) product (639) important (492)	5624

(b)



(c)

	Attribute	Dist.
	Multiple Choice	1902
	Single Choice	1745
	Scale	732
	Single Choice Matrix	753
	Page type	146
	Multiple Choice Matrix	144
	Word Problem	58
	Fill in The Blank	46
	Other	125
Source	bfz	2812
	ChatGPT3.5	2812

Table 1: Summary of (a) topical, (b) test item dificulty, and (c) structural contents of the bfz-EdTec dataset, where topic categories and top terms are translated in English language

mantic representation of the existing pre-trained language models, we averaged the embedding vectors by weighting the term frequency and the inverse document frequency scores (TF-IDF) of the represented terms.

4. Query-term expansion: We expanded search queries and documents with synonym terms as listed in OdeNet (Siegel and Bond, 2021), the German language version of Wordnet.
5. Vertical search: To reduce the search space to only relevant topics, we truncated the search to the top ten most semantically similar topics in terms of cosine similarity, where topics were defined by the item data base (refer to Table 1, first column).
6. MILP: After computing the top 100 most similar items to the query according to cosine similarity of sentence embeddings, we applied the method of (Mitchell et al., 2011) to re-rank the items to optimize the nDCG score via mixed integer linear programming (MILP) with a CBC Solver.
7. LambdaMART: We employed a LambdaMART (Burges, 2010; Chen and Guestrin, 2016) regressor for re-ranking the top 100 most similar items to the query. We evaluated our model with a ten-cross-validation schema. The average nDCG score of the model in cross-validation was 0.35.
8. Cross-encoder: We re-ranked the top 100 most similar items to the query with a multilingual MS Marco cross-encoder (Reimers and Gurevych, 2019)[5]. To re-rank, cross-encoders concatenate query and items, passing them to a transformer model. With a self-attention mechanism, the transformer learns the importance of weights across the concatenated inputs, scoring its relevance. Finally, the model re-orders an initial list of candidates by its relevance scores.

Note that each strategy may have several representatives. Overall, we evaluated 176 different combinations of search strategies and pre-trained sentence similarity models.

4.1 Results and Analysis

Table 2 reports each strategy's top three best-performing representatives (regarding nDCG) for the 100 highest-ranked items. We observed that the most effective strategy for our task is re-ranking an initial candidate list of relevant items with a cross-encoder (0.516 nDCG and 0.713 MRR), followed by a vertical search approach (0.468 nDCG and 0.245 MRR). The least effective strategies were BM25 and query expansion. Overall, we observed that using a pre-trained sentence similarity German language model[6] in conjunction with a multilingual cross-encoder[5] outperformed all the proposed strategies with their corresponding pre-trained sentence similarity model.

We conduct a correlation analysis via Kendall's τ between the best-performing search runs (refer to A.2) to analyze the differences across the best-performing strategies' search results. The LambdaMART re-ranker and the TF-IDF weighted average model had the most similar search results ($\tau = 0.59$). However, considering the nDCG

[5] amberoad/bert-multilingual-passage-reranking-msmarco

[6] deutsche-telekom/gbert-large-paraphrase-cosine

Strategy	Pre-trained Language Model	nDCG@100	MRR@100	Prec@100	Recall@100	F1@100	MAP@100
Baseline	bm25	0.07	0.12	0.05	0.09	0.06	0.01
LM+ANN	deutsche-telekom_gbert-large-paraphrase-cosine	0.43	0.20	0.31	0.53	0.37	0.25
	deutsche-telekom_gbert-large-paraphrase-euclidean	0.43	0.20	0.31	0.53	0.37	0.24
	sentence-transformers_paraphrase-multilingual-mpnet-base-v2	0.42	0.24	0.30	0.51	0.36	0.24
TF-IDF Weighted Average	deutsche-telekom_gbert-large-paraphrase-cosine	0.32	0.14	0.24	0.40	0.29	0.15
	PM-AI_sts_paraphrase_xlm-roberta-base_de-en	0.31	0.18	0.23	0.39	0.28	0.15
	nblokker_debatenet-2-cat	0.30	0.14	0.22	0.36	0.26	0.13
Query-Term Expansion	deutsche-telekom_gbert-large-paraphrase-euclidean	0.26	0.08	0.21	0.33	0.24	0.13
	deutsche-telekom_gbert-large-paraphrase-cosine	0.25	0.10	0.20	0.32	0.23	0.13
	PM-AI_sts_paraphrase_xlm-roberta-base_de-en	0.24	0.10	0.18	0.30	0.22	0.13
Vertical Search	deutsche-telekom_gbert-large-paraphrase-euclidean	0.46	0.24	0.34	0.58	0.41	0.26
	deutsche-telekom_gbert-large-paraphrase-cosine	0.45	0.23	0.33	0.57	0.40	0.25
	sentence-transformers_paraphrase-multilingual-mpnet-base-v2	0.44	0.24	0.32	0.55	0.39	0.25
MILP Re-ranker	deutsche-telekom_gbert-large-paraphrase-euclidean	0.39	0.26	0.31	0.53	0.37	0.15
	deutsche-telekom_gbert-large-paraphrase-cosine	0.38	0.18	0.31	0.53	0.37	0.14
	sentence-transformers_paraphrase-multilingual-mpnet-base-v2	0.38	0.18	0.30	0.51	0.36	0.14
LambdaMART Re-ranker	deutsche-telekom_gbert-large-paraphrase-euclidean	0.42	0.32	0.31	0.53	0.37	0.17
	deutsche-telekom_gbert-large-paraphrase-cosine	0.41	0.27	0.31	0.53	0.37	0.16
	sentence-transformers_paraphrase-multilingual-mpnet-base-v2	0.40	0.28	0.30	0.51	0.36	0.16
Cross-encoder Re-ranker	deutsche-telekom_gbert-large-paraphrase-cosine	0.51	0.70	0.31	0.53	0.37	0.28
	deutsche-telekom_gbert-large-paraphrase-euclidean	0.51	0.71	0.31	0.53	0.37	0.28
	0_Transformer	0.49	0.70	0.27	0.49	0.34	0.27

Table 2: Performance metrics for the eight search strategies (rows), combined with the respective best-performing language models, evaluated at a cut-off of 100.

scores, the ranking quality produced by the LambdaMART re-ranker model was 10% better compared to the TF-IDF weighted average. With a correlation of -0.89, the models that showed the slightest similarity in terms of the produced rankings were the MILP re-ranker and the BM25 model.

5 Tool Overview

This section presents EdTec-QBuilder, a semantic search service and web tool to support exam assembly. Figure 2 provides an overview of the tool. While the frontend is a website implemented in HTML and JavaScript, the backend is implemented in Python and Flask. Below, we summarize each of the main components.

User workflow Figure 2a illustrates EdTec-QBuilder's user workflow. First, after notifying users that we will only collect click data from the user interface and annotations for future research and development, users land on the search result page (SERP) interface after successfully authenticating themselves. Second, once in the SERP view, the web application provides a search bar where users can browse and search across the test items in the item database. Third, after triggering a search, the tool returns the top 100 items relevant to their query, grouped on pages with ten results each. We employed a cross-encoder as a re-ranking strategy because it showed the best performance (see Section 4). Fourth, as illustrated in Figure 2b, to assemble exams, users click on the test items they wish to include in their exam. When users click on an item in the ranked list, the item changes to green, indicating its inclusion in the exam. Any selected item can be de-selected by clicking again. Users can flag items if they consider them outdated or inconsistent by pressing a red flag button. Finally, after annotating, users submit the selected test items to another endpoint, e.g., to publish the exam or deliver it to a computer-based testing environment (CBTE). Note that our interface also crowdsources ground-truth annotation data by internally logging and sending the selected items to the tool's backend (refer to Appendix 4 for more details about the user interface).

API endpoints We implemented 13 interoperable functionalities coupled with the search and data collection processes. Figure 2c summarizes the endpoint's inter-module coordination. We provide a JSON Web Token-based service[7] to authenticate and serve client requests from the SERP. After user authentication, the endpoint validates the users' credentials in the user's database. When a user submits a query, the query is embedded as a vector via the deutsche-telekom_gbert-large-paraphrase-cosine model and the 100 closest items according to cosine similarity are retrieved via NMSLIB. To optimize the output of the initial search, the tool

[7] https://jwt.io/

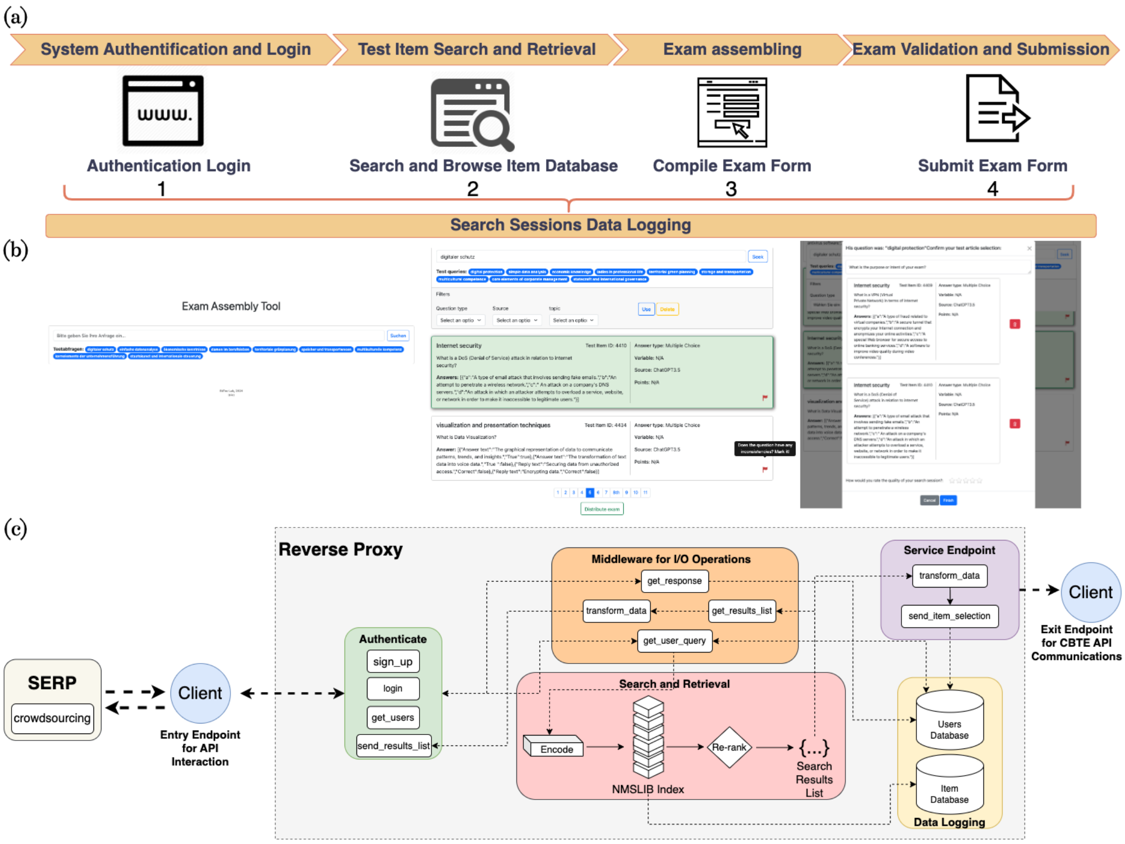


Figure 2: Summary of user workflow (a), API/endpoints architecture (b), SERP user interface (c).

re-ranks the top 100 results with a cross-encoder. Then, it returns a response in JSON format with the resulting test items and their corresponding attributes. The client receives the system's output, and the SERP maps the JSON into a suitable format for the visualization of results. The client then allows users to select relevant items; in the background, the SERP logs the selected items and sends this data to the web tool in a JSON format.

6 Conclusions

More than item generation is required to build high-quality exams in education; items must also be assembled into comprehensive examination forms (Lane et al., 2015; Kurdi et al., 2020). We framed assisted topic-based exam assembly as an information retrieval problem, evaluating 176 search and retrieval strategies and language model combinations on a 5,624-item database from the vocational education domain. The most promising strategy is to embed queries via language models, identify the closest 100 items regarding cosine similarity, and re-rank these 100 results via a cross-encoder, yielding 0.516 nDCG and 0.713 MRR. In other words, about half of the top-ranked results were relevant, and the highest-ranked relevant result was, on average, on rank 1 or 2. Using the cross-encoder re-ranking strategy, we developed EdTec-QBuilder[1], a semantic search service and tool to support exam assembly, which is currently in use at bfz, one of the leading vocational training providers in Bavaria. As competencies are known to be heterogenous constructs, test developers are well advised to combine items on multiple topics and items of varying difficulty (Fischer and Neubert, 2015). Future work should investigate leveraging crowdsourced search sessions and developing models for estimating test item difficulty without entirely depending on respondent data beforehand. Lastly, exploring and incorporating large language models could significantly aid the assembly process.

Limitations and Ethics Statement

While EdTec-QBuilder already supports assisted topic-based exam assembly significantly, some limitations remain. To overcome the cold start problem and rapidly transfer a tool to our industry partner to assist them in exam assembly, we based our evaluation on a synthetic and automated labeling scheme instead of using human judgments as ground truth. However, our tool establishes a research infrastructure for crowdsourcing ground truth expert data that can be used to improve or train future assisted or automatic test assembly models.

Exam assembly is an ethically charged subject matter because exam results can severely impact students' future learning and prospects. Therefore, our system does *not* automate exam assembly but, instead, serves as a tool for vocational educators who still have to make the final test item selection and take responsibility for the assembled examination form. Our system does not reason about the item quality but focuses on scoring the similarity between queries and test items. Finally, we may encounter a popularity bias for items that educators prefer in the early use of the system, and these preferences are "locked in" by using teacher preferences as training data. Our current training data is synthetically generated and thereby circumvents this particular bias, but future work should investigate this issue more closely.

References

Anaje Armendariz, Tomás A Pérez, and Javier López-Cuadrado. 2012. Compiling a test: how to solve calibration issues. *Procedia-Social and Behavioral Sciences*, 46:2253–2257.

Elias Bassani. 2022. ranx: A blazing-fast python library for ranking evaluation and comparison. In *ECIR (2)*, volume 13186 of *Lecture Notes in Computer Science*, pages 259–264. Springer.

Leonid Boytsov and Bilegsaikhan Naidan. 2013. Engineering efficient and effective non-metric space library. In *Similarity Search and Applications - 6th International Conference, SISAP 2013, A Coruña, Spain, October 2-4, 2013, Proceedings*, volume 8199 of *Lecture Notes in Computer Science*, pages 280–293. Springer.

Jonathan Brown, Gwen Frishkoff, and Maxine Eskenazi. 2005. Automatic question generation for vocabulary assessment. In *Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing*, pages 819–826, Vancouver, British Columbia, Canada. Association for Computational Linguistics.

Christopher JC Burges. 2010. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81.

Yllias Chali and Sadid A. Hasan. 2015. Towards topic-to-question generation. *Computational Linguistics*, 41(1):1–20.

Ying-Hong Chan and Yao-Chung Fan. 2019. A recurrent BERT-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 154–162, Hong Kong, China. Association for Computational Linguistics.

Tianqi Chen and Carlos Guestrin. 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.

Brian S Cole, Elia Lima-Walton, Kim Brunnert, Winona Burt Vesey, and Kaushik Raha. 2020. Taming the firehose: Unsupervised machine learning for syntactic partitioning of large volumes of automatically generated items to assist automated test assembly. *Journal of Applied Testing Technology*, pages 1–11.

Soham Datta, Prabir Mallick, Sangameshwar Patil, Indrajit Bhattacharya, and Girish Palshikar. 2021. Generating an optimal interview question plan using a knowledge graph and integer linear programming. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1996–2005, Online. Association for Computational Linguistics.

Xinya Du, Junru Shao, and Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1342–1352, Vancouver, Canada. Association for Computational Linguistics.

Cédrick Fairon. 1999. A web-based system for automatic language skill assessment: Evaling. In *Computer Mediated Language Assessment and Evaluation in Natural Language Processing*.

Andreas Fischer and Jonas Neubert. 2015. The multiple faces of complex problems: A model of problem solving competency and its implications for training and assessment. *Journal of Dynamic Decision Making*, 1(6):1–14.

Kyung Chris Tyek Han. 2018. Components of the item selection algorithm in computerized adaptive testing. *Journal of Educational Evaluation for Health Professions*, 15.

Michael Heilman and Noah A. Smith. 2010. Good question! statistical ranking for question generation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 609–617, Los Angeles, California. Association for Computational Linguistics.

Ghader Kurdi, Jared Leo, Bijan Parsia, Uli Sattler, and Salam Al-Emari. 2020. A systematic review of automatic question generation for educational purposes. *International Journal of Artificial Intelligence in Education*, 30:121–204.

Suzanne Lane, Mark R Raymond, and Thomas M Haladyna. 2015. *Handbook of test development*. Routledge.

Stuart Mitchell, Michael OSullivan, and Iain Dunning. 2011. Pulp: a linear programming toolkit for python.

The University of Auckland, Auckland, New Zealand, 65.

Ruslan Mitkov, Ha Le An, and Nikiforos Karamanis. 2006. A computer-aided environment for generating multiple-choice test items. *Natural language engineering*, 12(2):177–194.

Joao Palotti, Harrisen Scells, and Guido Zuccon. 2019. Trectools: an open-source python library for information retrieval practitioners involved in trec-like campaigns. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1325–1328.

Jean Piton-Gonçalves and Sandra Maria Aluísio. 2012. An architecture for multidimensional computer adaptive test with educational purposes. In *Proceedings of the 18th Brazilian Symposium on Multimedia and the Web*, WebMedia '12, page 17–24, New York, NY, USA. Association for Computing Machinery.

Chuan Qin, Hengshu Zhu, Chen Zhu, Tong Xu, Fuzhen Zhuang, Chao Ma, Jingshuai Zhang, and Hui Xiong. 2019. Duerquiz: A personalized question recommender system for intelligent job interview. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, KDD '19, page 2165–2173, New York, NY, USA. Association for Computing Machinery.

Fanyi Qu, Xin Jia, and Yunfang Wu. 2021. Asking questions like educational experts: Automatically generating question-answer pairs on real-world examination data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2583–2593, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. `http://is.muni.cz/publication/884893/en`.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Sherry Ruan, Liwei Jiang, Justin Xu, Bryce Joe-Kun Tham, Zhengneng Qiu, Yeshuang Zhu, Elizabeth L Murnane, Emma Brunskill, and James A Landay. 2019. Quizbot: A dialogue-based adaptive learning system for factual knowledge. In *Proceedings of the 2019 CHI Conference on Human Factors in Computing Systems*, pages 1–13.

Vasile Rus, Paul Piwek, Svetlana Stoyanchev, Brendan Wyse, Mihai Lintean, and Cristian Moldovan. 2011. Question generation shared task and evaluation challenge: Status report. In *Proceedings of the 13th European Workshop on Natural Language Generation*, ENLG '11, page 318–320, USA. Association for Computational Linguistics.

Anbuselvan Sangodiah, Rohiza Ahmad, and Wan Fatimah Wan Ahmad. 2016. Integration of machine learning approach in item bank test system. In *2016 3rd International Conference on Computer and Information Sciences (ICCOINS)*, pages 164–168.

Melanie Siegel and Francis Bond. 2021. OdeNet: Compiling a German wordnet from other resources. In *Proceedings of the 11th Global Wordnet Conference (GWC 2021)*, pages 192–198.

Simone Teufel. 2007. An overview of evaluation methods in trec ad hoc information retrieval and trec question answering. *Evaluation of text and speech systems*, pages 163–186.

Luu Anh Tuan, Darsh Shah, and Regina Barzilay. 2020. Capturing greater context for question generation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9065–9072.

Shriyash Upadhyay, Chris Callison-burch, and Etan Ginsberg. 2023. Learn with martian: A tool for creating assignments that can write and re-write themselves. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 267–276, Dubrovnik, Croatia. Association for Computational Linguistics.

Ellen M Voorhees, Donna K Harman, et al. 2005. *TREC: Experiment and evaluation in information retrieval*, volume 63. MIT press Cambridge.

Ellen M Voorhees, Ian Soboroff, and Jimmy Lin. 2022. Can old trec collections reliably evaluate modern neural retrieval models?

Nana Yoshimi, Tomoyuki Kajiwara, Satoru Uchida, Yuki Arase, and Takashi Ninomiya. 2023. Distractor generation for fill-in-the-blank exercises by question type. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pages 276–281, Toronto, Canada. Association for Computational Linguistics.

A Appendix

A.1 Pre-trained sentence similarity models Details

Table 3 displays the complete list of pre-trained semantic sentence similarity models used in the experiments.

The language models in this list were the top-trending pre-trained sentence similarity language models as of December 1, 2023, according to huggingface[8]. These models were evaluated in combination with the eight search strategies in Section 4.

[8] `https://huggingface.co/models?pipeline_tag=sentence-similarity&language=de&sort=trending`

#	Pre-trained Language Model
1	sentence-transformers/paraphrase-multilingual-mpnet-base-v2
2	aari1995/German_Semantic_STS_V2
3	sentence-transformers/LaBSE
4	PM-AI/bi-encoder_msmarco_bert-base_german
5	efederici/e5-base-multilingual-4096
6	intfloat/multilingual-e5-base
7	clips/mfaq
8	PM-AI/sts_paraphrase_xlm-roberta-base_de-en
9	deutsche-telekom/gbert-large-paraphrase-euclidean
10	LLukas22/all-MiniLM-L12-v2-embedding-all
11	LLukas22/paraphrase-multilingual-mpnet-base-v2-embedding-all
12	sentence-transformers/distiluse-base-multilingual-cased-v1
13	sentence-transformers/distiluse-base-multilingual-cased-v2
14	deutsche-telekom/gbert-large-paraphrase-cosine
15	shibing624/text2vec-base-multilingual
16	Sahajtomar/German-semantic
17	setu4993/LaBSE
18	symanto/sn-xlm-roberta-base-snli-mnli-anli-xnli
19	and-effect/musterdatenkatalog_clf
20	nblokker/debatenet-2-cat
21	setu4993/LEALLA-large
22	dell-research-harvard/lt-wikidata-comp-de
23	ef-zulla/e5-multi-sml-torch
24	barisaydin/text2vec-base-multilingual
25	meta-llama/Llama-2-7b-chat-hf

Table 3: The full list of tested pre-trained language models.

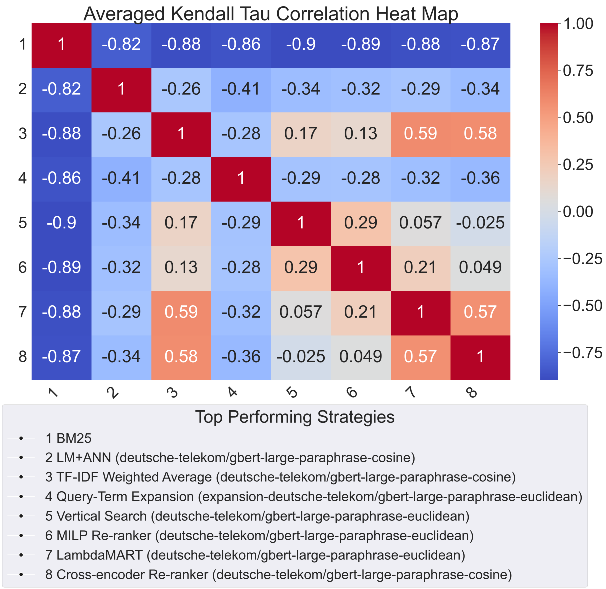


Figure 3: The averaged Kendall τ rank correlation coefficient across all the best-performing search runs for each search strategy.

A.2 Kendall Tau Correlation Analysis

To understand how much the different search strategies (in combination with the respective best-performing language odel) agree in their search results, we calculated Kendall's τ coefficients (refer to Section 4.1). Figure 3 summarizes the correlation coefficients when comparing the best search run outputs.

A.3 Search Results Page

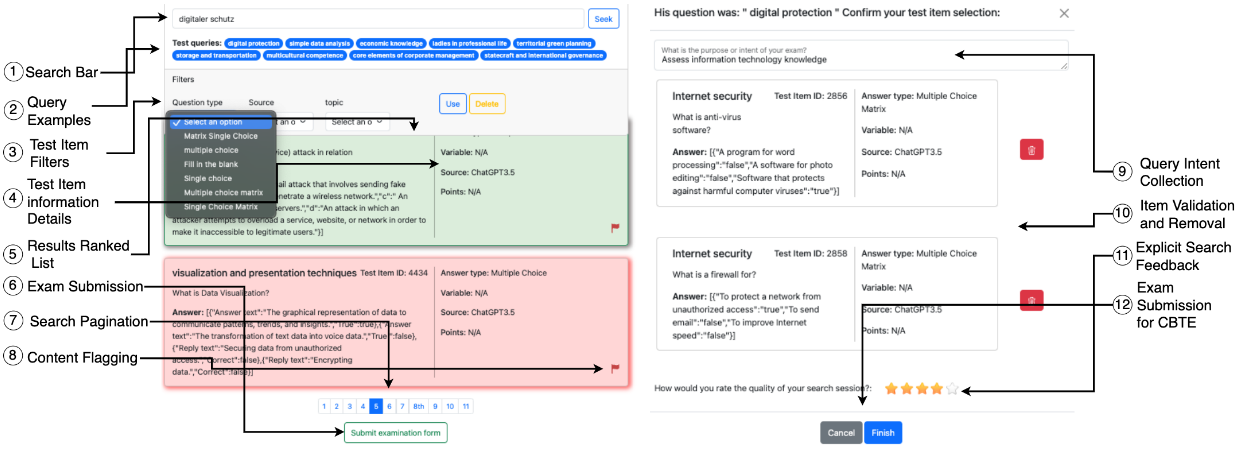
An essential component of our search service and tool is the search results page (SERP), which allows and simplifies browsing and assembling test items from the item database. The SERP will also enable us to gather expert crowdsourced data from manual exam assembly processes, which was nonexistent at the time of development (refer to Section 5).

User Interface Client: The user interface client of the SERP primarily consists of 12 major elements. Figure 4 summarizes these major components. After the search module retrieves relevant test items from the database, the response is sent to the user interface client in JSON format. Then, users can select and determine for each query what items are relevant (Figure 4a, the item filtering mode). Once relevant items are identified, the users validate their selections by leaving or removing them (Figure 4b, the validation mode). The user interface requests the intent of the exam and explicit feedback about the quality of the search. The feedback is collected by grading the search experience from one to five stars. Finally, when users finish validating their selection, a submit button sends the selected items to another endpoint, e.g. a computer-based testing environment. All the interactions and annotations are securely stored in the database. When users accept the general data protection regulation policy, we inform them that we only collect relevant judgments and annotations that we will use later for future improvements and development of exam assembly models.

Crowdsourcing Functionality: In addition to serving as a user interface, the SERP interface stores selected and not-selected items for each query and sends them back to an internal API endpoint. We will later use the stored interactions to develop assisted and automatic exam assembly models further. Figure 5 summarizes an example of a compressed version of an interaction captured and stored in the search service database. The system stores the search queries, the IDs of the elements marked as relevant, all elements not selected but displayed in the user interface, the selected filters, the value of the feedback, and the intention of the query (for more details, refer to 5).

A.4 Backend

As mentioned in section 5, the backend of the search service is fully implemented as a Flask web



digitaler schutz

Test queries: digital protection simple data analysis economic knowledge ladies in professional life territorial green planning storage and transportation multicultural competence core elements of corporate management statecraft and international governance

Filters

Question type Source topic

ice) attack in relation

ail attack that involves sending fake
netrate a wireless network.","c":" An
ervers.","d":"An attack in which an
attacker attempts to overload a service, website, or network in order to make it inaccessible to legitimate users."}]

Variable: N/A
Source: ChatGPT3.5
Points: N/A

visualization and presentation techniques Test Item ID: 4434

What is Data Visualization?

Answer: [{"Answer text":"The graphical representation of data to communicate patterns, trends, and insights.","True":true},{"Answer text":"The transformation of text data into voice data.","True":false},{"Reply text":"Securing data from unauthorized access.","Correct":false},{"Reply text":"Encrypting data.","Correct":false}]

Answer type: Multiple Choice
Variable: N/A
Source: ChatGPT3.5
Points: N/A

(a) Item Filtering and Selection Mode

His question was: " digital protection " Confirm your test item selection:

What is the purpose or intent of your exam?
Assess information technology knowledge

Internet security Test Item ID: 2856

What is anti-virus software?

Answer: [{"A program for word processing":"false","A software for photo editing":"false","Software that protects against harmful computer viruses":"true"}]

Answer type: Multiple Choice Matrix
Variable: N/A
Source: ChatGPT3.5
Points: N/A

Internet security Test Item ID: 2858

What is a firewall for?

Answer: [{"To protect a network from unauthorized access":"true","To send email":"false","To improve Internet speed":"false"}]

Answer type: Multiple Choice Matrix
Variable: N/A
Source: ChatGPT3.5
Points: N/A

How would you rate the quality of your search session?: ★★★★☆

(b) Item Validation Mode

Figure 4: EdTec-QBuilder's search results page fuctionality details (contents automatically translated to English language).

```
{
    "search_log": {
        "query": "digitaler schutz",
        "relevant_ids": ["2856", "2858"],
        "irrelevant_ids": ["4753", "5035",
                                    ...,"4086"],
        "filters": ["filters"],
        "seconds": "88",
        "timestamp": "1/14/2024 5:22:28 AM",
        "intent": "the query intent",
        "feedback": {"value": "4"},
        "user_id": "83ee3871840871c2a3a7"
    },
    "browsed_ids": ["2815", "2832",
                                   ..., "2879"]
}
```

Figure 5: A compressed representation of the captured interactions, as stored in the backend database of the search service.

application. All the API endpoints are implemented in the Flask framework. All the search sessions are securely stored in an SQLite[9] database. The backend is mounted on an Amazon EC2 instance with four processor cores, 15GB of total memory, and a storage drive of 16 GB.

[9] https://www.sqlite.org/