

Learning to Rank Question Difficulty from Text-Question Alignment using Deep Representations

Vicente Cordova
Facultad de Ingeniería
Universidad Andres Bello
Chile
v.crdovacastillo@uandresbello.edu

Matias Valdenegro-Toro
Department of Artificial Intelligence
University of Groningen
The Netherlands
m.a.valdenegro.toro@rug.nl

Orietta Nicolis
Facultad de Ingeniería
Universidad Andres Bello, Chile
University of Messina, Italy
orietta.nicolis@unab.cl

Billy Peralta
Facultad de Ingeniería
Universidad Andres Bello
Chile
billy.peralta@unab.cl

Abstract—In a context where artificial intelligence and large language models (LLMs) are radically transforming education, there is a pressing need to automatically evaluate and organize the content generated by these technologies—particularly assessment questions. This work addresses the technical challenge of classifying LLM-generated questions based on their difficulty, a problem often overlooked by traditional approaches that do not explicitly model the semantic relationship between the question and its source text. We propose a system that generates questions from input passages using LLMs such as Gemini, and classifies them via deep learning models trained with embeddings and regularization techniques, implemented in TensorFlow and PyTorch. Our methodology includes the creation of a custom dataset derived from SQuAD passages, the vectorization of texts and questions using various embedding strategies, and a comparative evaluation of multiple classification architectures. Experimental results show that the model based on paraphrase-MiniLM-L6-v2 achieves the best performance in the primary three-class setting, reaching 72% validation accuracy. As a secondary analysis, when the easy and medium classes are merged and compared against the hard class, the model achieves 90% precision and 75% recall for hard-question detection. This supports the hypothesis that more difficult questions, due to their less ambiguous patterns, are classified with higher precision.

Keywords—Question classification, LLM, MLP

I. INTRODUCTION

The rise of large language models (LLMs) and their increasing integration into educational tools and digital environments have profoundly transformed the way knowledge is generated, distributed, and assessed. This phenomenon unfolds in a global context where artificial intelligence (AI) plays a central role across various sectors, with education being no exception. LLMs enable the automatic generation of educational content, including explanations, summaries, and assessment questions, thus opening new opportunities for personalized learning, reducing feedback time, and optimizing

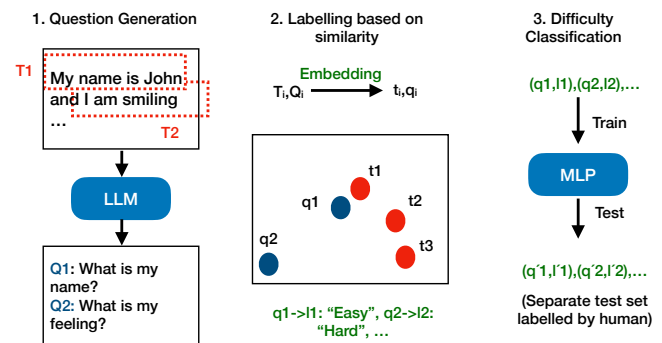


Fig. 1. General flowchart of proposal.

instructional resources. However, these advances introduce significant challenges in automatically classifying, filtering, and validating the generated content, particularly with respect to assessment questions, which may lack coherence, relevance, or an appropriate level of difficulty due to the absence of direct human review. From a technical standpoint, this work addresses the challenge by transforming both the questions and their source texts into vector representations (embeddings), and subsequently analyzing their semantic relationship using deep learning-based classification models.

The core problem tackled in this study is the lack of automated mechanisms for classifying LLM-generated questions according to their difficulty level, an essential requirement for applications such as personalized education, adaptive learning platforms, and automated assessment systems [1]. Despite advancements in natural language generation [2], many existing models overlook the explicit relationship between a generated question and its original context, often assuming the validity or

complexity of the output by default. This oversight limits the practical utility of such models in real-world settings, where it is crucial to distinguish between literal, inferential, or abstract questions [3]. Furthermore, most publicly available datasets lack difficulty annotations, preventing the development of supervised models for this task. Consequently, a comprehensive solution is needed—one that not only generates questions but also estimates their difficulty in relation to the source text.

To address this gap, we propose a methodological framework that combines question generation via LLMs, semantic similarity computation between questions and source text using embeddings, and supervised classification of question difficulty. The proposed system begins by selecting the most suitable generative model—Gemini—due to its balance between diversity and quality, and constructing a custom dataset based on the SQuAD corpus. From this dataset, we derive vector representations using models such as paraphrase-MiniLM-L6-v2, which are capable of capturing deep semantic relationships between questions and texts. These representations are then fed into neural classifiers implemented in PyTorch and TensorFlow. We evaluate several architectural variants and regularization strategies. Figure 1 presents an overview of the proposed pipeline. The specific components are detailed in the following sections.

In the following sections, we describe the employed methodology in detail, including dataset construction, data preprocessing, and classifier design. We then provide a comparative analysis of different embedding strategies and training procedures, along with the evaluation metrics used. Finally, we discuss the limitations of the approach, its implications for educational applications, and future research directions—particularly improvements in dataset curation and the integration of multilingual models.

II. RELATED WORK

Recent research has addressed the task of question generation and classification from various perspectives. Broadly, existing approaches fall into two main categories: (a) the use of Transformer-based architectures trained for controllable text generation, and (b) the design of alignment mechanisms between structured data and linguistic outputs. While these strategies have shown promising results, several challenges remain unresolved, particularly regarding semantic controllability, automated evaluation, and difficulty estimation of generated questions. Below, we review representative works illustrating the current state of the art.

Gong et al. [4] propose an enhanced Transformer model for generating text from structured data, incorporating latent representations and an auxiliary training objective to explicitly model the relevance of each record. Through data augmentation techniques, their system achieves notable improvements in BLEU scores and contextual precision, particularly in domains such as sports summarization.

Zhang et al. [5] present a comprehensive survey on controllable text generation (CTG) using pretrained language

models. Their taxonomy covers three primary strategies: fine-tuning, retraining with auxiliary modules, and semantic post-processing. In addition to discussing applications, the authors highlight critical limitations such as the absence of standardized evaluation metrics and the high computational cost of training, and propose future directions for ethically and efficiently integrating CTG.

Wang et al. [6] introduce a question generation model based on answer pivots, combining answer-aware and answer-agnostic strategies. Their architecture includes a Transformer encoder, a hidden answer predictor, and an autoregressive decoder, leading to more diverse and relevant question outputs. When evaluated on SQuAD, their model significantly outperforms previous baselines in both automated and qualitative assessments, particularly in reading comprehension tasks.

Patwardhan et al. [7] provide a broad review of the practical impact of NLP in everyday applications, highlighting how Transformer models have redefined tasks such as machine translation, sentiment analysis, and question answering. The paper emphasizes the ability of embeddings to capture deep linguistic structures, which has fueled the widespread adoption of models like BERT and GPT across sectors including healthcare, education, and customer service.

Celikyilmaz et al. [8] analyze evaluation strategies for natural language generation systems. They categorize existing metrics into human evaluations, traditional automatic metrics (e.g., BLEU, ROUGE), and learning-based approaches. The authors argue that traditional metrics fail to capture semantic richness, and advocate for machine learning-based models that emulate human judgment in contextual and structural assessments.

Kowsari et al. [9] review text classification algorithms in the face of exponentially growing unstructured data. Their analysis spans classical approaches such as Naive Bayes and SVM, as well as modern architectures like CNNs and Transformers. The review emphasizes the importance of multi-label classification and deep learning for structuring large-scale text corpora in real-world applications.

Unlike traditional methods, our system introduces an explicit difficulty classifier trained on question-context embedding pairs, allowing us to automatically assess both literalness and semantic depth. This design enables filtering of redundant content and facilitates the automated construction of high-quality, difficulty-aware question banks.

III. PROPOSED METHOD

The core intuition behind the proposed method is that the semantic proximity between a question and its source text serves as a meaningful proxy for estimating question difficulty. Questions that are closely aligned with the textual content—often reflecting literal recall—tend to be easier, while those that require inference or abstract reasoning show lower semantic similarity. By leveraging pretrained embeddings to quantify this alignment, we enable the estimation of question difficulty *without* relying on human annotation, provided the alignment-based assumption holds.

The proposed method consists of a hybrid pipeline that combines automatic question generation using large language models (LLMs), semantic processing via embeddings, and supervised difficulty classification using deep neural networks. The overall architecture comprises three main stages: data generation, semantic similarity computation, and classifier training.

A. Question Generation and Dataset Construction

The initial stage of our pipeline focuses on generating candidate questions from input passages and constructing a labeled dataset for difficulty classification. Pretrained large language models (LLMs) were employed to generate questions from text passages extracted from the SQuAD corpus. Five generative models—Gemini, Llama2, Llama3, Hermes, and TinyLlama—were evaluated.

Each input passage produced multiple candidate questions, filtered through BLEU similarity metrics to eliminate near-duplicates. Following this, difficulty labels (easy, medium, hard) were automatically assigned based on the semantic similarity between each question and its corresponding passage.

Three embedding-based similarity strategies were tested for this purpose: (1) TF-IDF with cosine similarity, (2) paraphrase-MiniLM-L6-v2 embeddings with PyTorch, and (3) all-MiniLM-L6-v2 embeddings. The third method demonstrated the highest reliability and was chosen as the primary annotation technique.

Difficulty thresholds were defined as follows: a similarity score above 0.80 denoted an *easy* question, scores between 0.725 and 0.80 corresponded to *medium* difficulty, and scores equal to or below 0.725 were labeled as *hard*. These thresholds were selected heuristically after inspecting the empirical distribution of similarity scores and representative question-text pairs. The goal was to obtain three interpretable difficulty regions while preserving a reasonable number of examples per class. Therefore, the thresholds should not be interpreted as universal cut-off values, but as dataset-specific operational criteria.

This automatic labeling process was cross-validated against the LLM’s own difficulty assessments, revealing strong alignment. The resulting dataset consists of 1,351 annotated question-text pairs. Table I presents representative examples of generated questions categorized by difficulty level.

TABLE I
EXAMPLES OF QUESTIONS BY DIFFICULTY LEVEL

Question	Difficulty
How was last week?	Easy
Where did the lightning strike?	Medium
What feeling did the narrator have?	Hard

B. Vectorization and Embeddings

Both questions and texts were transformed into vector representations using different embedding techniques: Bert-

Algorithm 1: Question Difficulty Classification Pipeline

Input: Text passages $T = \{t_1, t_2, \dots, t_n\}$

Output: Predicted difficulty levels

$D = \{d_1, d_2, \dots, d_n\}$

foreach $t_i \in T$ **do**

$Q_i \leftarrow \text{GenerateQuestions}(t_i)$ // Use LLM to generate questions $Q_i \leftarrow$

$\text{FilterRedundant}(Q_i)$ // Apply BLEU-based filtering

foreach $q_{ij} \in Q_i$ **do**

$s_{ij} \leftarrow \text{ComputeSimilarity}(q_{ij}, t_i)$ // Using embedding-based similarity $d_{ij} \leftarrow$

$\text{AssignDifficulty}(s_{ij})$ // Label as Easy, Medium, or Hard $v_{ij} \leftarrow$

$\text{EmbedPair}(q_{ij}, t_i)$ // Concatenate embeddings

$M \leftarrow \text{TrainClassifier}(\{v_{ij}, d_{ij}\})$ // Train neural network classifier

foreach *New question-text pair* (q, t) **do**

$v \leftarrow \text{EmbedPair}(q, t)$ // Embed pairs $d \leftarrow$

$\text{PredictDifficulty}(M, v)$ // Predict difficulty using a MLP network.

 Append d to D

return D

base-uncased (BBC) [10], CountVectorizer (CV) [11], TfidfVectorizer (TV) [12], and Paraphrase-MiniLM-L6-v2 (PML) [13]. Each embedding strategy was evaluated independently to analyze its impact on classifier performance. For training, the text and question embeddings were concatenated to form combined vectors capturing the semantic alignment between both inputs.

C. Difficulty Classification

We formulate difficulty estimation as a multiclass classification problem with three categories. Multiple neural network architectures were trained using PyTorch and TensorFlow. The networks consist of four to six dense layers with ReLU or LeakyReLU activations, Dropout for regularization, and batch normalization. Training was performed using the class-weighted CrossEntropyLoss function to address dataset imbalance, and the Adam optimizer.

Each input to the neural network consists of the concatenation of a question embedding and its corresponding text embedding, which duplicates the size of embedding output. Depending on the embedding method used, the resulting input vector size ranges from 768 (for paraphrase-MiniLM-L6-v2) up to 2972 (for TF-IDF-based encodings). The output variable is the estimated difficulty level of the question, easy, medium, or hard, based on the semantic alignment score computed in Section 3.1.

Algorithm Overview: The proposed pipeline initiates by employing a Large Language Model (LLM), such as Gemini,

to generate a set of candidate questions from each input text passage. To ensure diversity and eliminate redundancy, these questions undergo filtering based on BLEU scores. Subsequently, each filtered question is compared to its source text using embedding-based semantic similarity measures (e.g., paraphrase-MiniLM-L6-v2). The resulting similarity scores serve as proxies for question difficulty, allowing for the assignment of difficulty labels (Easy, Medium, Hard) based on predefined thresholds. Both the questions and their corresponding texts are then transformed into vector representations through embedding techniques and concatenated to form combined feature vectors. These vectors, along with their assigned difficulty labels, constitute the dataset used to train a supervised neural network classifier. Once trained, this classifier can predict the difficulty level of new question-text pairs by processing their concatenated embeddings, thereby facilitating automated assessment of question complexity.

IV. EXPERIMENTAL DESIGN

The experimental design focused on evaluating the performance of different combinations of vector representations and classification architectures for predicting the difficulty level of questions generated by LLMs. A modular pipeline was developed to systematically analyze how each technical decision influences the final model’s accuracy.

A. Model Architectures and Training Configurations

Each embedding technique was evaluated with at least two Multilayer Perceptron (MLP) neural network architectures, varying depth, activation functions, Dropout usage, batch normalization, and loss functions. The full experimental suite included 10 different models, implemented in both PyTorch and TensorFlow. Across all experiments, optimization hyperparameters were kept consistent: Adam optimizer, learning rates of 0.001 or 0.0001, fixed epoch counts, and 20% of the dataset reserved for validation. The final models use a 20% dropout rate, except for BBC where no dropout is applied. Additionally, batch normalization is used in the CV model. Table II shows the best configuration for each embedding model considering validation set. The column structure shows the number of neurons by hidden layer, while the output layer always has 3 neurons given the difficulty classes.

TABLE II
NEURAL NETWORK CONFIGURATIONS YIELDING BEST VALIDATION
PERFORMANCE PER EMBEDDING METHOD

Method	Layers	Act.	Structure
BBC	5 dense	ReLU	1024-512-256-128-32
CV	1 dense	LeakyReLU	1024-512-256-128-32
TV	5 dense	ReLU	1024-512-256-128-32
PML	3 dense	ReLU	64-32-16

B. Experimental Setup

The dataset consists of 1,351 question-text pairs automatically annotated with difficulty levels based on semantic similarity. For supervised training and evaluation, we partitioned

the dataset by allocating 80% of the samples for training and 20% for validation. To further assess model generalization, we constructed an independent test set composed of 100 manually labeled question-text pairs. During training, stratified sampling was applied to preserve the original class distribution across difficulty levels (easy, medium, hard). All inputs were preprocessed into concatenated text-question embeddings using the methods described in Section 3.2.

To validate the proposed method and systematically assess each component of the pipeline, we designed a series of controlled experiments. These experiments aim to measure the influence of question generation quality, embedding strategy, and classifier behavior on the final prediction performance. Specifically, we propose the following experimental evaluations:

- **Question Generation Evaluation:** Measure the syntactic and semantic quality of questions generated by different LLMs, using BLEU scores computed against reference questions.
- **Embedding Impact Analysis:** Compare classification performance across various embedding strategies, including bert-base-uncased, CountVectorizer, TfidfVectorizer, and paraphrase-MiniLM-L6-v2.
- **Difficulty Classification Analysis:** Analyze model behavior at the class level through confusion matrices, focusing on systematic errors across difficulty levels (easy, medium, hard).

In all experiments, training and validation sets were used to measure performance under controlled conditions. Specifically, BLEU-based LLM evaluation and embedding comparison analyses were conducted using the validation set only. The independent manually labeled test set was exclusively reserved for final classification behavior analysis, providing an unbiased assessment of model generalization to unseen examples. Classification performance will be evaluated using accuracy and confusion matrix to investigate detailed class-wise prediction patterns.

V. RESULTS

In this section, we first conducted a series of experiments analyzing key components of the pipeline. First, we evaluated the fluency and semantic quality of the questions generated by different LLMs using BLEU scores. We then compared the impact of multiple embedding strategies on classification accuracy. Finally, we examined per-class behavior through confusion matrix analysis, highlighting systematic patterns in difficulty prediction in a separate test set. A demo code is available in <https://zenodo.org/records/20060236>.

A. LLM Evaluation via BLEU

Table III presents the average BLEU scores for five generative language models evaluated across three different texts. The models are sorted by their BLEU-1 scores, which reflect unigram lexical overlap with reference questions. *Llama2* and *Gemini* clearly outperformed the others, both reaching the highest BLEU-1 score ranges (0.537 and 0.543, respectively),

and matching BLEU-2 and BLEU-3 values. Notably, *Gemini* achieved the best BLEU-4 score (0.217), which captures n-gram precision with longer fragments, suggesting superior syntactic structure and semantic coherence. These results validate the choice of Gemini as the preferred generator in the pipeline, due to its consistent balance of diversity, syntactic fluency, and semantic relevance across varied input contexts.

TABLE III
BLEU AVERAGE SCORES ACROSS THREE TEXTS (SORTED BY BLEU-1)

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4
TinyLlama	0.353	0.233	0.160	0.110
Hermes	0.420	0.287	0.200	0.145
Llama3	0.490	0.360	0.273	0.187
Llama2	0.537	0.413	0.313	0.213
Gemini	0.543	0.413	0.313	0.217

B. Embedding Comparison

Table IV summarizes the training and validation accuracy achieved by the best-performing model for each embedding strategy. The results reveal clear differences in generalization capacity. Classical approaches like *CountVectorizer* and *TfidfVectorizer* performed better than *bert-base-uncased*, achieving 70% and 68% validation accuracy respectively, albeit with signs of overfitting (18–22 point gaps between training and validation). In contrast, *paraphrase-MiniLM-L6-v2* achieved the highest accuracy overall (72% validation, 93% training), confirming its suitability for modeling semantic alignment between text and question pairs. As this embedding is pretrained specifically for paraphrase and similarity tasks, it provides stronger inductive bias for inferring question difficulty. These results justify the selection of MiniLM as the embedding of choice in our final architecture.

It is important to note that this 72% validation accuracy corresponds to the primary three-class formulation of the problem, involving easy, medium, and hard questions. Therefore, this result is considered the main classification performance of the proposed model.

TABLE IV
TRAINING AND VALIDATION ACCURACY BY EMBEDDING TYPE
(BEST-PERFORMING MODEL)

Embedding	Accuracy	
	Training	Validation
BERT (bert-base-uncased)	0.59	0.58
CountVectorizer	0.88	0.70
TfidfVectorizer	0.90	0.68
Paraphrase-MiniLM-L6-v2	0.93	0.72

C. Difficulty Classification Analysis

Fig. 2 shows the confusion matrix obtained on the independent test set. The model exhibits a tendency to assign questions to higher difficulty categories: hard questions are identified more consistently, whereas easy questions are frequently confused with the medium class. This behavior suggests that low semantic alignment with the source passage is easier

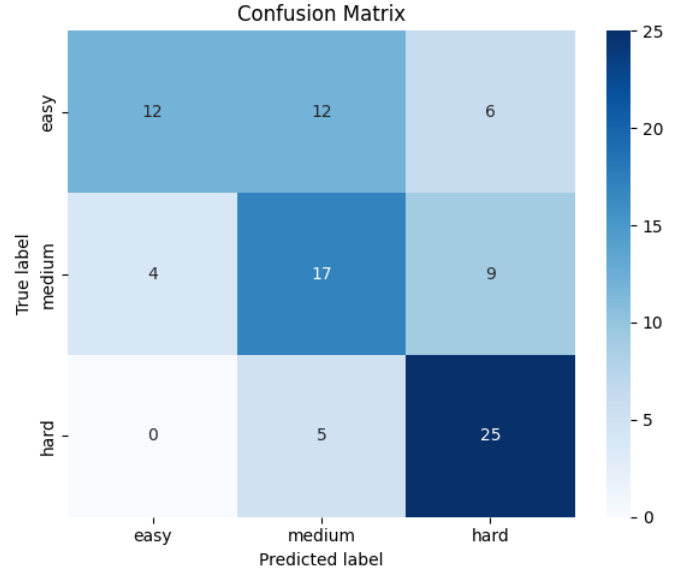


Fig. 2. Matrix confusion considering three classes

to detect than the boundary between literal and moderately inferential questions. Such ambiguity may be partly induced by the threshold-based labeling strategy, which provides a reproducible annotation rule but may introduce noise near class boundaries.

To further analyze this behavior, we also conducted a secondary binary evaluation by merging the easy and medium classes and comparing them against the hard class. This analysis does not replace the original three-class formulation, but provides an additional view of the model’s ability to separate hard from non-hard questions. Under this secondary binary setting, the model reaches 90% precision and 75% recall for the detection of hard questions.

VI. DISCUSSION

The experimental outcomes demonstrate that combining dense semantic embeddings with supervised classification architectures enables reasonably accurate identification of the difficulty level of questions generated from text. Among all embedding strategies, *paraphrase-MiniLM-L6-v2* emerged as the most effective, consistently outperforming classical representations such as *CountVectorizer* and *TfidfVectorizer*.

A key finding is the class-wise performance asymmetry: “hard” questions were classified more accurately than “easy” or “medium” ones. This suggests that lower semantic similarity with the source text is more easily detected by the model, while literal or near-verbatim questions pose ambiguity near class boundaries. This effect may be amplified by the threshold-based labeling scheme, which, though reproducible, introduces semantic noise in borderline cases.

A further limitation is the potential circularity introduced by the automatic labeling strategy. Since the difficulty labels are derived from embedding-based similarity thresholds, and the

classifier is trained on text-question representations obtained from related embedding spaces, the model may partially learn to approximate the similarity function used during annotation rather than fully capture pedagogical difficulty. This issue is particularly relevant for the easy and medium classes, where the confusion matrix shows substantial overlap. Therefore, the reported results should be interpreted as evidence of alignment-based difficulty estimation, rather than as a definitive measure of human-perceived question difficulty.

Additionally, analysis of the generative models highlighted the influence of content quality on classification accuracy. Gemini produced the most semantically diverse and relevant questions, achieving the highest BLEU scores and avoiding duplication. These observations imply that robust classification systems should be tightly coupled with high-quality, controlled generation stages.

Despite promising results, certain limitations remain. First, the absence of human-annotated labels may introduce training bias. Second, generalization was tested primarily in English; preliminary experiments in Spanish yielded near-random results, indicating limited multilingual robustness. Lastly, while the model captures semantic alignment, it does not incorporate explicit reasoning or pragmatic structures—critical for deeper difficulty assessment.

Finally, in relation to the alternative to prompting a LLM, like Gemini, to generate easy or hard questions, our method introduces an explicit, reproducible mechanism for difficulty estimation based on semantic alignment between the question and its source text. This approach mitigates the inherent subjectivity of LLM outputs by using pretrained embeddings and a supervised classifier to quantify difficulty objectively.

VII. CONCLUSIONS

This work presented an integrated system capable of automatically classifying the difficulty level of questions generated by LLMs, based on their semantic alignment with source texts. We demonstrated that, using contextual embeddings and supervised classifiers, this task can be modeled with up to 90% test bi-class accuracy with paraphrase-Mini-LM-L6-v2 embeddings.

Through a rigorous experimental design, we observed that the system performs especially well in detecting difficult questions, while further work is needed to better discriminate between literal and moderately complex items. Furthermore, we confirmed that the quality of the generative model directly impacts overall pipeline performance.

For future work, the main priority is the creation of a human-annotated benchmark dataset, where expert or teacher judgments are used as ground truth. Such a benchmark would allow us to quantify the gap between alignment-based automatic labels and human-perceived difficulty, and to validate whether the proposed classifier generalizes beyond the similarity function used during annotation. Additional directions include semi-supervised learning techniques and the evaluation of multilingual embeddings such as LaBSE or MPNet.

REFERENCES

- [1] Debanjan Mandal, Debasis Koley, and Arnab Bhattacharyya, “A survey on automatic question generation using deep learning,” *ACM Computing Surveys (CSUR)*, vol. 55, no. 3, pp. 1–38, 2022.
- [2] Sonam Soni, Praveen Kumar, and Amal Saha, *Automatic Question and Answer Generation from Text Using Neural Networks*, pp. 497–504, 05 2021.
- [3] Michael Heilman and Noah A Smith, “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*, 2010, pp. 609–617.
- [4] Li Gong, Josep Crego, and Jean Senellart, “Enhanced transformer model for data-to-text generation,” in *Proceedings of the 3rd Workshop on Neural Generation and Translation*, ACL, 2019, pp. 148–156.
- [5] Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song, “A survey of controllable text generation using transformer-based pre-trained language models,” *arXiv preprint arXiv:2302.03027*, 2023.
- [6] B. Wang, X. Wang, T. Tao, Q. Zhang, and J. Xu, “Neural question generation with answer pivot,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 9138–9145.
- [7] Narendra Patwardhan, Stefano Marrone, and Carlo Sansone, “Transformers in the real world: A survey on nlp applications,” *Information*, vol. 14, no. 4, 2023.
- [8] Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao, “Evaluation of text generation: A survey,” *arXiv preprint arXiv:2006.14799*, 2021.
- [9] Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura Barnes, and Donald Brown, “Text classification algorithms: A survey,” *Information*, vol. 10, no. 4, 2019.
- [10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, “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)*, Jill Burstein, Beate Dorow, Mari Ostendorf, and Nakamoto Tetsuro, Eds., Minneapolis, Minnesota, June 2019, pp. 4171–4186, Association for Computational Linguistics.
- [11] Gerard Salton, Anita Wong, and C S Yang, “A vector space model for automatic indexing,” *Communications of the ACM*, vol. 18, no. 11, pp. 613–620, 1975.
- [12] Karen Sparck Jones, “A statistical interpretation of term specificity and its application in retrieval,” *Journal of Documentation*, vol. 28, no. 1, pp. 11–21, 1972.
- [13] Nils Reimers and Iryna Gurevych, “Sentence-bert: Sentence embeddings using siamese bert-networks,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, 11 2019, Association for Computational Linguistics.