Evaluation and Fine-Tuning of Encoder-Decoder Models for Question and Answer Generation from the NCERT Dataset

Course: GENERATIVE ARTIFICIAL INTELLIGENCE (CS787)

Instructor: Dr. Arnab Bhattacharya & Dr. Subhajit Roy IIT Kanpur

Team Members:

Ankit Kumar Keshri (251040608)

Lokesh Mahawar (220590)

Nikhil Niranjan (251040624)

Purushottam Kumar (251110607)

Thanush A A (251110615)

Submission Date: November 15, 2025

Department of Computer Science Engineering
Indian Institute of Technology, Kanpur

Contents

Abstract

Manually creating questions for educational assessments is a significant bottleneck for educators. This project aims to solve this problem by fine-tuning encoder-decoder models to generate context-aware questions and answers from educational text, providing practice tools for students. We utilise a dataset of 30,706 samples derived from the NCERT dataset available on HuggingFace [3, 7], focusing on Physics, Chemistry, Biology, and Science for grades 10-12. We conduct a comparative evaluation of encoder-decoder models (T5-Small, T5-Base, BART-Base) against a BERT-SQG (Sequential Question Generation) baseline. We also investigated the impact of prompt engineering by comparing a detailed prompt (P1) with a simplified, more natural prompt (P2). Our results show that T5-Base using the simplified P2 prompt yields state-of-the-art performance, achieving a BLEU-4 score of 28.06 for QG [9]. T5-Base performed best for Answer Generation as well, achieving a BLEU-4 score of 15.88. The high BERT score for AG (0.88) suggest strong semantic correctness despite the lower lexical overlap [12].

1 Source Code Link

GitHub Repository: CS787-Project-Question-Generator

2 Introduction

In the Educational technology (EdTech) domain, the ability to automatically generate assessment materials is a crucial task. Educators often spend significant time manually crafting questions and answers to evaluate student understanding. This project addresses this problem by developing an automated pipeline capable of generating context-aware questions and corresponding answers based on educational text, while also providing students with a tool that can generate questions and answers for them to practice.

Using a dataset derived from NCERT textbooks, specifically for science and math-based subjects from grades 10 to 12, we aim to train generative models that take a specific context (explanation), grade level, difficulty, and subject as input to generate relevant questions. Furthermore, we implement a separate system to generate accurate answers for those questions.

We trained several models on the dataset including **BERT-SQG**, **T5Small-QG-P1/P2**, **T5Base-QG-P1/P2**, and **BARTBase-QG-P1/P2**. For answer generation, we trained prompt-variant models and refer to them as **T5Small-AG-P1/P2**, **T5Base-AG-P1/P2**, and **BART Base-AG-P1/P2**. Our results show that **T5Base-QG-P2** outperforms the BERT-SQG baseline in terms of BLEU score and BERTScore F1 [10, 5, 1].

The remainder of this report is organised as follows: Section 3 provides a review of relevant literature. The link to the GitHub repository of the code is provided in Section 1. Section 4 outlines the details of the dataset. Section 5 provides details regarding the Exploratory Data Analysis carried out in the dataset. Section 8 presents the experimental results and analysis, and Section 9 concludes the report with directions for future work.

3 Literature Review

Automated Question Generation (QG) has progressed over the years from early rule-based approaches to contemporary neural architectures. Early QG systems adopted rule-based techniques, using hand-crafted linguistic rules, syntactic transformations, and templates (e.g., Heilman & Smith, 2010 [2]). Although these systems were interpretable and offered designers control over output, they did not generalize well, depended on high-quality parsing, and were not capable of handling embedded or complex sentence structures.

Advances in neural machine translation enabled neural sequence-to-sequence (Seq2Seq) QG methods that learn implicit linguistic rules directly from data. Early neural QG research framed question generation as a two-step process that used attention-based Seq2Seq models to infer questions either from text or from knowledge-base triples. However, these models typically generate questions without explicitly modeling the answer span that the question targets. This diverges from human question generation, where people aim to discover specific information and form questions centered on that target.

To address this gap, subsequent studies have focused on answer-aware QG, where models either receive or identify ground-truth answer spans before generating questions. Despite improvements, challenges remain: accurately selecting answer spans, handling rare words within spans, and generating contextually relevant questions aligned with the intended answer.

Natural language generation and question generation technology have been rapidly evolving and are underpinned by large-scale pre-trained transformer-based models. The model used in this study, T5 (Text-to-Text Transfer Transformer), builds on a unified text-to-text framework that treats all natural language processing tasks as sequence-to-sequence generation tasks. Such a framework allows for a single architecture to be potentially used across a variety of tasks that

include but do not limit to text translation, text summarization, and question generation. [10] develop T5 model using an encoder-decoder transformer trained on a large, cleaned web corpus (C4) from a denoising objective. The authors suggest that when used in the context of scaling the model size and a consistent text-to-text formulation, they will find state-of-the-art results on the majority of benchmarks, including question answering and summarization tasks. T5's encoder–decoder design is especially suitable for Question Generation (QG) because it allows the model to encode contextual passages and then auto regressively generate well-formed questions. This matches the architecture used in our implementation, where T5 For Conditional Generation is fine-tuned to generate questions from input text using a supervised text-to-text setup.

Comparative analyses in the literature illustrate how the different variants of Transformers have unique strengths. BART [5] is an encoder-decoder model but is trained as a denoising autoencoder that is robust to any noise transformation. BART combines a bidirectional encoder (like BERT) and a left-to-right decoder (like GPT) and exhibits superior performance on abstraction-based generation tasks such as summarization and QG. The pre-training methodology involving masking text to fill it in and active permutation of sentences, allows BART greater reconstruction capacity for complex input sequences; BART consistently demonstrates a performance advantage in QG preparation tasks from previous work.

In contrast, BERT [1] is trained with a masked-language-model objective and next-sentence prediction on text bidirectionally using an encoder. While it achieves strong performance in classification and token-level prediction tasks, the encoder structure of BERT connotes a lack of a decoder and therefore is unable to generate text. For question generation tasks, BERT is typically embedded into classification-style or extractive QG frameworks, as opposed to generative frameworks. The use of BERT in this case is analogous to your notebook. It is solely used as the baseline comparison, rather than the text generative model.

Overall, existing literature has shown that encoder-decoder Transformers (T5, BART) outperform encoder-only models (BERT) on the task of text generation, particularly in the case of QG tasks. T5's unified framework to convert any text-to-text process, scalable architecture, and robust pre-training are advantageous features that make T5 the appropriate model to implement for the question generation pipeline used in this project.

4 Dataset description

The raw CSV file has the following columns:

```
Topic, Explanation, Question, Answer, Difficulty, StudentLevel,
QuestionType, QuestionComplexity, Prerequisites, EstimatedTime, subject, grade
```

Total rows loaded: **30,706**. The dataset was accessed from HuggingFace and preprocessed using standard tools [7, 11].

5 Exploratory Data Analysis (EDA)

5.1 Missing values

All columns in the dataset have complete entries (no nulls in the original CSV reading step):

```
Topic              0
Explanation        0
Question           0
Answer             0
Difficulty         0
```

```
StudentLevel          0
QuestionType          0
QuestionComplexity    0
Prerequisites         0
EstimatedTime         0
subject               0
grade                 0
```

5.2 Text length statistics (words)

Word-length statistics for the main text fields (computed as word counts):

	count	mean	std	min	50%	max
len_Explanation	30706	77.93	23.01	1	74	308
len_Question	30706	14.08	5.21	1	14	81
len_Answer	30706	32.18	19.56	1	32	177

5.3 Duplicates

Initial duplicate counts discovered during EDA:

- Duplicate **questions**: 2,429
- Duplicate **answers**: 1,677
- Duplicate **question+answer pairs**: 1,008

5.4 Categorical distributions

Subject counts (raw):

- Physics: 10,505
- Chemistry: 9,911
- Biology: 6,894
- Science: 3,396

Grade distribution (raw):

- Grade 11: 14,533
- Grade 12: 12,777
- Grade 10: 3,396

Difficulty / StudentLevel :

- Difficulty: Medium (10,267), Hard (10,242), Easy (10,193), plus a few malformed rows.
- StudentLevel: Intermediate (10,267), Advanced (10,242), Beginner (10,193), plus a few malformed rows.

5.5 Token-length analysis

We tokenized **Explanation** texts with `t5-base` tokenizer (fast tokenizer) to get realistic token lengths for transformer inputs [10, 4]. Summary of `expl_tok_len` (token counts for Explanation):

```
count    30706.000000
mean       117.108904
std         40.394809
min          4.000000
25%         90.000000
50%        109.000000
75%        134.000000
90%        167.000000
95%        194.000000
98%        232.000000
99%        258.950000
max        502.000000
```

Coverage at candidate truncation cutoffs:

- 256 tokens covers **98.95%** of Explanations
- 320 tokens covers **99.79%**
- 384 tokens covers **99.97%**
- 512 tokens covers **100%**

6 Methodology

6.1 Overall Pipeline Diagram

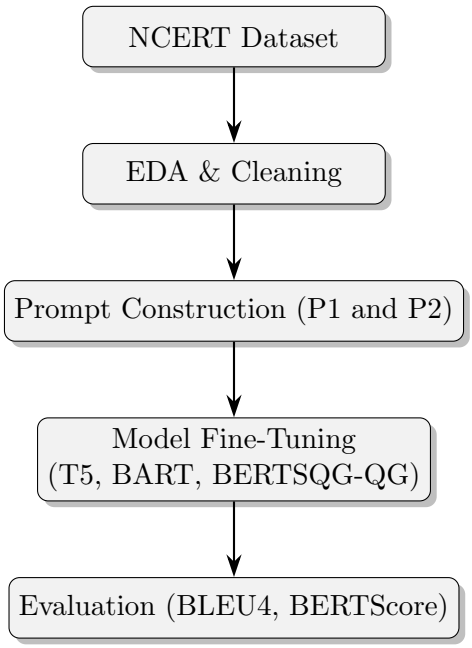

Figure 1: End-to-End System Pipeline

7 Model Architecture

This work evaluates several encoder–decoder models for the tasks of Question Generation (QG) and Answer Generation (AG). All models follow a sequence-to-sequence (Seq2Seq) formulation, where the input prompt—containing the educational context, grade, difficulty, subject, and optional complexity score—is encoded into a latent representation by the encoder. The decoder then autoregressively generates the target question or answer token-by-token.

A summary of all architectural configurations used in this project is presented in Table 2. The table includes encoder and decoder depth, hidden dimensions, feed-forward size, number of attention heads, and approximate parameter counts.

7.1 T5 Architecture

The T5 (Text-to-Text Transfer Transformer) architecture is naturally suitable for QG and AG because it reformulates every language task in a text-to-text format [10]. T5 consists of a Transformer encoder and decoder, both built from stacked self-attention and feed-forward layers, with shared token embeddings across encoder and decoder. The model uses *relative positional biases* rather than absolute position embeddings, enabling improved generalization to variable-length inputs, which is particularly relevant for NCERT explanations of varying length.

Two size variants of T5 are fine-tuned in this work (see Table 2):

- **T5Small** (used in: T5Small-QG-P1/P2 and T5Small-AG-P1/P2), with a 6-layer encoder and 6-layer decoder, hidden size of 512, feed-forward dimensionality of 2048, and approximately 60M parameters.
- **T5Base** (used in: T5Base-QG-P1/P2 and T5Base-AG-P1/P2), with a 12-layer encoder and 12-layer decoder, hidden size of 768, feed-forward dimensionality of 3072, and roughly 220M parameters.

The T5 decoder contains both a masked self-attention block and an encoder–decoder cross-attention block, enabling it to condition question and answer generation directly on the information present in the Explanation text [10].

7.2 BART Architecture

BART is a denoising autoencoder built using a bidirectional Transformer encoder (similar to BERT) and an autoregressive Transformer decoder (similar to GPT) [5]. The encoder processes the context bidirectionally, making it effective for explanations/summarisation-based tasks. The decoder then generates sequences autoregressively based on the encoder outputs.

The models fine-tuned for this work include:

- **BARTBase-QG-P1/P2** for question generation.
- **BARTBase-AG-P1/P2** for answer generation.

As shown in Table 2, BARTBase consists of a 6-layer encoder, 6-layer decoder, hidden size of 768, feed-forward dimensionality of 3072, and approximately 139M parameters [5].

7.3 BERTSQG-QG Baseline

To provide a classical baseline, we include the **BERTSQG-QG** model. This architecture uses a BERT-base encoder (12 layers, hidden size 768) [1], combined with a lightweight decoder head. Because it does not include a full Transformer decoder stack, it is less expressive for generative tasks compared to T5 and BART. Additionally, due to preprocessing constraints, the baseline is trained only on 5,000 samples, which significantly limits its performance compared to the fully fine-tuned encoder–decoder models.

7.4 Model Comparison Table

Table 2 provides a unified comparison of all model variants used in the project, including model size, encoder and decoder depth, dimensionality, and architectural notes.

Table 2: Comparison of model architectures used for QG and AG experiments

Model Name	Base Arch.	Variant	Enc. Layers	Dec. Layers	d_{model}	d_{ff}	Notes
T5Small-QG-P1/P2	T5-small	QG	6	6	512	2048	$\sim$ 60M params, 6 heads
T5Base-QG-P1/P2	T5-base	QG	12	12	768	3072	$\sim$ 220M params, 12 heads
BARTBase-QG-P1/P2	BART-base	QG	6	6	768	3072	$\sim$ 139M params, 12 heads
T5Small-AG-P1/P2	T5-small	AG	6	6	512	2048	Same architecture as QG
T5Base-AG-P1/P2	T5-base	AG	12	12	768	3072	Same architecture as QG
BARTBase-AG-P1/P2	BART-base	AG	6	6	768	3072	Same architecture as QG
BERTSQG-QG (baseline)	BERT-base	QG baseline	12	—	768	3072	Encoder-only; trained on 5k samples

Table 3: Training / decoding parameters extracted from `cs787-question-generator(2).ipynb`

Model (experiment)	Base model id	Batch	Epochs	Learning rate	Weight decay	Warmup	Max input	Max target	Decoding / notes
T5Small-QG-P1 / P2	(t5-small)	4	3	5e-5	0.01	500	512	128 / 512	beam search: num_beams=5; max_length 128/256
T5Base-QG-P1 / P2	(t5-base)	4	3	5e-5	0.01	500	512	128 / 512	beam search: num_beams=5; max_length 128/256
BARTBase-QG-P1 / P2	(facebook/bart-base)	8	3	5e-5	0.01	500	512	128 / 512	beam search: num_beams=5; max_length 128/256
T5Small-AG-P1 / P2	(t5-small)	4	3	5e-5	0.01	500	512	128 / 512	AG prompt P1/P2; same decoding
T5Base-AG-P1 / P2	(t5-base)	4	3	5e-5	0.01	500	512	128 / 512	AG P1/P2; P2 best scores
BARTBase-AG-P1 / P2	(facebook/bart-base)	8	3	5e-5	0.01	500	512	128 / 512	same decoding as QG/BART
Baseline: BERTSQG-QG	(bert-base-uncased)	4	3	5e-5	0.01	500	512	128	**Trained on reduced dataset (5,000 samples)**; encoder-centric

8 Analysis and Results

This section details the quantitative results of our experiments. We evaluated three model architectures (T5, BART, and BERT-SQG) on our two primary tasks: Question Generation (QG) and Answer Generation (AG). Furthermore, we analyze the impact of iterative prompt engineering on model performance.

8.1 Prompt Engineering

The design of the input prompt is critical for guiding a model's generative process. We experimented with two distinct prompt structures.

8.1.1 Prompt 1 (P1)

Our initial prompt was highly detailed, explicitly including all features from the dataset, such as `QuestionComplexity`.

- **QG Prompt 1:** `Generate a {Difficulty} question with complexity {Score} for a grade {Grade} {Subject} student. Explanation: {Context}`
- **AG Prompt 1:** `Answer the following {Difficulty} grade {Grade} {Subject} question with complexity {Score}. Explanation: {Context} Question: {Question}`

8.1.2 Prompt 2 (P2)

Based on initial observations, we hypothesized that the `QuestionComplexity` score was redundant and potentially confusing to the model. We revised the prompt to be more direct and natural.

- **QG Prompt 2:** `Generate a {Difficulty} question for a grade {Grade} {Subject} student using this context: {Context}`
- **AG Prompt 2:** `Answer the following {Difficulty} grade {Grade} {Subject} question. Explanation: {Context} Question: {Question}`

8.2 Experimental Results

We evaluated all generated outputs against their reference counterparts using two standard metrics: BLEU-4 and BERTScore F1 [9, 12].

NOTE: BERT-SQG baseline model was trained on a significantly reduced dataset of **5,000 samples**. This was due to the model's architectural inefficiency, which resulted in a ∼15x data explosion during preprocessing, making training on the full 30,000-sample dataset computationally infeasible within our time constraints. All T5 and BART models were trained on the full 30,000-sample dataset.

8.2.1 Performance with Prompt 1

The initial results from our first prompt set the baseline for our models are given in Tables 4 and 5 for question generation and answer generation respectively.

Table 4: Question Generation (QG) Performance using Prompt 1

Model	BLEU-4 Score	BERTScore F1
T5Small-QG-P1	19.01	0.9082
T5Base-QG-P1	25.75	0.9201
BARTBase-QG-P1	23.26	0.9126
BERT-SQG	16.92	0.8750

Table 5: Answer Generation (AG) Performance using Prompt 1

Model	BLEU-4 Score	BERTScore F1
T5Small-AG-P1	9.88	0.8737
T5Base-AG-P1	13.35	0.8776
BARTBase-AG-P1	11.79	0.8743

8.2.2 Performance with Prompt 2 (Improved)

After modifying the QG prompt (and AG prompt for P2), we re-trained and evaluated the T5-Base and BART-Base models (and AG variants). The results are given in Tables 6 and 7 for Question Generation and Answer Generation, respectively.

Table 6: Question Generation (QG) Performance using Prompt 2

Model	BLEU-4 Score	BERTScore F1
T5Small-QG-P2	22.68	0.9135
T5Base-QG-P2	**28.06**	0.9202
BARTBase-QG-P2	25.79	**0.9193**
BERT-SQG	16.92	0.8750

Table 7: Answer Generation (AG) Performance using Prompt 2

Model	BLEU-4 Score	BERTScore F1
T5Small-AG-P2	13.35	0.8776
T5Base-AG-P2	**15.88**	**0.8800**
BARTBase-AG-P2	12.71	0.8796

8.3 Interpretation of Results

Our results lead to the following conclusions:

1. **Effectiveness of Prompt Engineering:** Comparing Table 4 and Table 6, the impact of our prompt refinement is clear. The T5Base-QG model's BLEU-4 score for QG improved when using Prompt 2 (T5Base-QG-P2) from 25.75 to 28.06, and BARTBase increased from 23.26 to 25.79. We could infer from these results that the simpler, more natural language prompt (P2) allowed the models to generate more accurate questions.

2. **T5Base-QG provided optimal performance:** Across all experiments, the **T5Base-QG-P2** variant provided better performance for question generation. For Answer Generation, the **T5Base-AG-P2** variant achieved the highest BLEU (15.88) and BERTScore (0.8800). This suggests the T5 pre-training objective is well-suited for text-to-text generation tasks when fine-tuned appropriately and when paired with improved prompt phrasing [10].

3. **Semantic vs. Lexical Scores:** For QG (Table 6), the BERTScores for T5Base-QG (0.9201, P1) and BARTBase-QG (0.9193, P2) are extremely high and very close. This indicates that both models are generating questions that are semantically **identical** to the ground truth, even if they use different wording. The T5Base-QG's higher BLEU score suggests it is slightly better at matching the exact phrasing of the reference questions [9, 12].

4. **Lower Scores for Answer Generation:** The BLEU scores for Answer Generation (Tables 5 and 7) are consistently lower than their QG counterparts. This is expected, as an answer can be phrased correctly in many ways, making an exact lexical match (which BLEU measures) difficult. The high BERTScores (all ∼0.87-0.88) are more informative here, through which we can infer that the models are generating semantically correct answers [12].

5. **BERT-SQG Baseline:** The BERT-SQG model was used as a baseline, showing the lowest performance (BLEU 16.92 on P1). This is attributable to both its inefficient architecture for generation and its limited training dataset (5,000 samples), which validates our decision to focus on T5 and BART as the primary models for this project [1].

9 Conclusion and Future Work

9.1 Conclusion

In this project, we have successfully fine-tuned and evaluated the performance of T5Small-QG-P1, T5Base-QG-P1, BARTBase-QG-P1, T5Small-AG-P1, T5Base-AG-P1, BARTBase-AG-P1, T5Small-QG-P2, T5Base-QG-P2, BARTBase-QG-P2, T5Small-AG-P2, T5Base-AG-P2, and BARTBase-AG-P2 for the automatic generation of questions and answers from the NCERT textbook dataset using BLEU score and BERT score F1 as our evaluation metrics [9, 12]. Our findings demonstrate that the **T5Base** model is the most effective and robust architecture for this task, achieving a top BLEU score of **28.06** for Question Generation.

Our simplified natural language prompt (P2) significantly improved performance in all models, boosting the BLEU scores of all fine-tuned models. This confirms that removing redundant features like `QuestionComplexity` and using a more direct instruction enhances the model focus. The encoder-decoder models (T5 and BART) outperformed the encoder-only BERT-SQG baseline in terms of both accuracy and efficiency, validating our choice of architecture [10, 5, 1].

Finally, our analysis showed that Answer Generation (AG) is a more lexically diverse task than Question Generation (QG), resulting in lower BLEU scores. However, the high BERTScore F1 results (all $\sim$0.88) confirm that our models are generating semantically correct answers, even if they are not exact word-for-word matches. This work validates that a fine-tuned T5-Base model is a viable and effective solution to creating automated educational assessment tools.

9.2 Future Work

We propose that the following areas could be explored in the future as an extension to this project

- **Fine-tuning + Retrieval-Augmented Generation (RAG):** RAG-based architectures [6] can be used to improve factual grounding. After fine-tuning the model as done in this work, a RAG pipeline can be added where the system first retrieves the most relevant NCERT passages and then combines them with the prompt before generation. This approach may help produce more accurate, context-aware, and robust questions and answers.

- **Reinforcement Learning from Human Feedback (RLHF):** RLHF [8] can be used to align the model with teacher preferences by rewarding high-quality questions and penalizing incorrect ones. A reward model trained on teacher ratings could be incorporated to reinforce educational value.

- **Model Scaling and Hyperparameter Tuning:** Another direct way to finetuen `t5-large` or `bart-large`, which may generate more coherent questions. [10, 5].

- **Evaluation by humans:** In addition to the mathematical matrix evaluations, we also need human evaluation (teachers and students), with ratings that will be more robust.

- **Application Deployment:**. The best-performing T5-Base model, we can integrate with a web application with a very simple UX-UI design, where teachers and students can generate questions and answers.

- **Joint Multi-Task Learning (QG + AG:** Here we trained the model separately for QG and AG, but we can also try to train a model that does both QG and AG.
- **Evaluation Beyond BLEU:** BLEU is limited for educational text. In future work could be use modern metrics like BLEURT, ROUGE-L, METEOR, BARTscore, or GPT-based semantic graders that will be better evaluation.

References

[1] 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 (NAACL-HLT)*, pages 4171–4186, 2019.

[2] Michael Heilman and Noah A. Smith. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 1011–1019, Los Angeles, California, 2010.

[3] KadamParth. NCERT Dataset (CSV) — HuggingFace Dataset Entry. https://huggingface.co/datasets/KadamParth/NCert_dataset, 2023.

[4] Taku Kudo and John Richardson. Subword regularization: Improving neural network translation models with multiple subword candidates. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 66–75, 2018.

[5] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 7871–7880, 2020.

[6] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 2020.

[7] Quentin Lhoest et al. Datasets: A community library for natural language processing. https://github.com/huggingface/datasets, 2021.

[8] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, et al. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[9] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 311–318, 2002.

[10] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[11] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Joe Brew. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2020.

[12] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*, 2019.