AUTOMATIC QUESTION GENERATION FOR IMPROVING LOW RESOURCE QUESTION ANSWERING PERFORMANCE

by

Yusufcan Manav

B.S., Computer Engineering, Boğaziçi University, 2018

Submitted to the Institute for Graduate Studies in
Science and Engineering in partial fulfillment of
the requirements for the degree of
Master of Science

Graduate Program in Computer Engineering

Boğaziçi University

2023

ACKNOWLEDGEMENTS

I express my deepest gratitude to my thesis supervisors, Assoc. Prof. Arzucan Özgür and Assist. Prof. Ebru Arısoy Saraçlar, for their unwavering assistance and valuable mentorship throughout my academic endeavor. Collaborating with them has been an honor, and their guidance has contributed significantly to my growth and knowledge. I am sincerely grateful for their dedicated time and efforts invested in my development.

I would like to thank Merve Ünlü for the remarkable collaboration and support throughout my thesis journey. Her dedication, expertise, and willingness to share their knowledge have greatly influenced my learning and growth.

I extend my appreciation to Assist. Prof. Şeniz Demir for her invaluable guidance in the processing and utilization of Wikipedia data for the purpose of this study.

This work is supported by Boğaziçi University Research Fund under the Grant Number 16903.

ABSTRACT

AUTOMATIC QUESTION GENERATION FOR IMPROVING LOW RESOURCE QUESTION ANSWERING PERFORMANCE

This thesis focuses on employing a question-generation system to improve the performance of question-answering models. We propose a multitask-trained question-generation module that is built on a multilingual encoder-decoder architecture and can produce question-answer pairs over plain text passages. We were able to adapt the question-generation system to several languages by using a multilingual model. First, we created a Turkish Question Answering dataset utilizing the Turkish Wikipedia pages and this question-generation system. Our experiments revealed that the performance on the Turkish XQuAD set was enhanced by 3% when the generated dataset was combined with the human-annotated dataset for question-answering model training.

Second we also extensively test our model in many languages and low-resource environments. We used limited annotated data from the question-answering datasets from different languages like English, German, French, and Turkish; to train the question generation model. We then utilized this model to create artificial question-answer pairs from the unannotated paragraphs. Our experiments revealed that, especially in the lower data settings, our augmentation strategy consistently outperformed the baseline question-answering models that are trained on human-annotated data across a range of dataset sizes and languages.

ÖZET

DÜŞÜK KAYNAKLI SORU CEVAPLAMA PERFORMANSINI ARTIRMAK İÇİN OTOMATIK SORU ÜRETİMİ

Bu tez, soru-cevap modellerinin performansını artırmak için bir soru üretim sistemi kullanma üzerine odaklanmaktadır. Önerdiğimiz model çok dilli Transformer tabanlı bir kodlayıcı-çözücü mimarisi üzerine inşa edilerek, birden fazla görev için aynı anda eğitilmiştir. Bu model sayesinde yalnız paragraflar girdi olarak alınarak soru cevap ikilileri üretilebilir. Çok dilli bir model kullanmamız sayesinde soru üretim sistemini çeşitli dillere uyarlayabildik. İlk olarak, Türkçe Vikipedi sayfalarını ve bu soru üretim sistemi kullanarak Türkçe Soru Yanıt veri kümesini oluşturduk. Deneylerimiz, üretilen veri kümesinin insan tarafından işaretlenmiş Soru Yanıt veri kümesiyle birleştirildiğinde, Türkçe XQuAD setindeki performansın %3 arttığını ortaya çıkardı.

İkinci olarak, modelimizi birçok dil ve düşük kaynak ortamında kapsamlı bir şekilde test ettik. Soru üretim modelini eğitmek için İngilizce, Almanca, Fransızca ve Türkçe gibi farklı dillerden soru-yanıt veri kümesinden sınırlı sayıda işaretlenmiş veri kullandık. Daha sonra bu modeli işaretlenmemiş paragraflardan yapay soru-cevap çiftleri oluşturarak; soru cevaplama modelinin eğitimine ek veri olarak kattık. Deneylerimiz, özellikle düşük veri ayarlarında, arttırma stratejimizin, insan tarafından işaretlenmiş veriye dayalı temel soru-yanıt modellerinin farklı boyutta ve dilde veri kümeleri üzerinde daha iyi performans gösterdiğini ortaya koydu.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS . . . iii
ABSTRACT . . . iv
ÖZET . . . v
LIST OF FIGURES . . . ix
LIST OF TABLES . . . x
LIST OF ACRONYMS/ABBREVIATIONS . . . xiii
1. INTRODUCTION . . . 1
2. RELATED WORK . . . 4
2.1. Question Generation and Filtering . . . 4
2.2. Question Answering . . . 9
3. METHODOLOGY . . . 11
3.1. Datasets . . . 11
3.1.1. SQuAD . . . 12
3.1.2. Spoken-SQuAD . . . 12
3.1.3. TQuAD . . . 13
3.1.4. THQuAD . . . 13
3.1.5. SQuAD-TR . . . 13
3.1.6. FQuAD . . . 14
3.1.7. GermanQuAD . . . 14
3.1.8. English Lectures . . . 14
3.1.9. Turkish Lectures . . . 15
3.1.10. XQuAD . . . 15
3.1.11. Turkish Wikipedia Cleaned Dump . . . 16
3.2. Models . . . 16
3.2.1. mT5 . . . 17
3.2.2. BERT . . . 17
3.2.3. ELECTRA . . . 18
3.2.4. XLM-RoBERTa . . . 18

3.3. Architecture . . . 19
3.3.1. Question Generation . . . 19
3.3.1.1. Training Tasks . . . 21
3.3.1.2. Model Architecture and Training Data Preparation . . . 21
3.3.1.3. Generating Question-Answer Pairs . . . 23
3.3.2. Question Quality Filters . . . 23
3.3.2.1. Heuristics Based Filter . . . 24
3.3.2.2. Deep Learning Model Based Filter . . . 25
3.3.3. Question Annotation Tool . . . 26
3.3.4. Question Answering Models . . . 26
3.3.4.1. Augmented Training . . . 27
3.3.4.2. 2 Step Training . . . 27
4. EXPERIMENTS AND RESULTS . . . 29
4.1. Question Answering Dataset Generation . . . 29
4.1.1. Experimental Setup . . . 29
4.2. Filtering the Generated Question-Answer Pairs . . . 34
4.2.1. Experimental Setup . . . 34
4.2.2. Results . . . 35
4.3. Data Augmentation Method For Question Answering Task . . . 38
4.3.1. Experimental Setup . . . 38
4.3.2. Results . . . 43
4.4. Domain Adaptation . . . 56
4.4.1. Experimental Setup . . . 56
4.4.2. Results . . . 57
4.4.2.1. SQuAD to English Lectures . . . 57
4.4.2.2. ThQuAD to Turkish Lectures . . . 58
5. CONCLUSION . . . 61
REFERENCES . . . 63
APPENDIX A: SQUAD EXPERIMENT RESULTS . . . 71
APPENDIX B: SPOKEN SQUAD EXPERIMENT RESULTS . . . 72
APPENDIX C: FQUAD EXPERIMENT RESULTS . . . 73

APPENDIX D: GERMANQUAD EXPERIMENT RESULTS 74
APPENDIX E: TURKISH LECTURES EXPERIMENT RESULTS 75
APPENDIX F: ENGLISH LECTURES EXPERIMENT RESULTS 76

LIST OF FIGURES

Figure 3.1. Overall architecture of the proposed framework. 20

Figure 4.1. The EM scores of different question-answering model settings on SQuAD development set. 44

Figure 4.2. The F1 scores of different question-answering model settings on SQuAD development set. 45

Figure 4.3. The EM scores of different question-answering model settings on Spoken SQuAD development set. 46

Figure 4.4. The F1 scores of different question-answering model settings on Spoken SQuAD development set. 47

Figure 4.5. The EM and F1 scores of question-answering models that augmented with generated data in SQuAD development set. 49

Figure 4.6. The EM and F1 scores of question-answering models that augmented with generated data in FQuAD development set. 50

Figure 4.7. The EM and F1 scores of question-answering models that augmented with generated data in GermanQuAD development set. . . 51

Figure 4.8. The EM and F1 scores of question-answering models that augmented with generated data in Spoken SQuAD development set. . 55

LIST OF TABLES

Table 3.1. Dataset statistics: number of articles, paragraphs, and question-answer pairs for each dataset used in the experiments. 12

Table 4.1. Question generation model evaluation with automatic metrics. . . 30

Table 4.2. Scores of different question-answering setups on ThQuAD test set and XQuAD. 31

Table 4.3. SQA performance of the BERTurk model. 33

Table 4.4. Performance of various filter methods over the test set of the question quality dataset. 36

Table 4.5. True positive, true negative, false positive, false negative filter status counts of our different filter models over the test set of the question quality dataset. This also contains the breakdown of each individual heuristic used in the Heuristics Based Filter. 37

Table 4.6. QA results of the BERT model trained with filtered and unfiltered versions of the generated dataset. Both versions of this dataset also omit the questions that are annotated for the filter model training. 38

Table 4.7. The number of question-answer pairs generated from the question-generation models trained on various data subsets. 43

Table 4.8. Comparison of question-generation models trained on varying size annotated data (1K - 5K) for English, French, and German reading comprehension datasets. 48

Table 4.9. Performances of question-answering and SQA models fine-tuned using only annotated data and using both annotated and generated data on English and Turkish lecture datasets. 53

Table 4.10. The performance of the question-answering and SQA models on spoken SQuAD and SQuAD. 54

Table 4.11. QA performance over the English Lectures dataset with different augmentation methods. 58

Table 4.12. QA performance over the Turkish Lectures dataset with different augmentation methods. 59

Table A.1. QA results of various models that are trained with SQuAD 1K-5K settings. 71

Table B.1. QA results of various models that are trained with Spoken SQuAD 1K-5K settings. 72

Table C.1. QA results of various models that are trained with FQuAD 1K-5K settings. 73

Table D.1. QA results of various models that are trained with GermanQuAD 1K-5K settings. 74

Table E.1. QA results of various models that are trained with Turkish Lectures Reading Comprehension dataset. 75

Table E.2. QA results of various models that are trained with Turkish Lectures Listening Comprehension dataset. 75

Table F.1. QA results of various models that are trained with English Lectures Reading Comprehension dataset. 76

Table F.2. QA results of various models that are trained with English Lectures Listening Comprehension dataset. 76

LIST OF ACRONYMS/ABBREVIATIONS

ASR	Automatic Speech Recognition
BLEU	Bilingual Evaluation Understudy
EM	Exact Match
LM	Language Model
LSTM	Long Short-Term Memory
MLM	Masked Language Modeling
NLP	Natural Language Processing
POS	Part of Speech
QA	Question Answering
QG	Question Generation
RNN	Recurrent Neural Networks
ROUGE	Recall-Oriented Understudy for Gisting Evaluation
SQA	Spoken Question Answering
TTS	Text to Speech
XLM	Cross-lingual Language Modeling
XML	Extensible Markup Language

1. INTRODUCTION

Question-answering (QA) is the task of finding answers for questions, from given text or spoken documents. A specific subtask of QA, known as machine reading comprehension, aims to find answers from a related text document. The answer is directly extracted from the given passage as a sequence of words. Listening comprehension is another subtask of the question answering. This task aims to find answers to the question from spoken documents.

Question answering models are trained using the question answer document triplets. During the training, the model takes a document and question and learns to predict the answer span from the document. In recent years many question-answering datasets and model architectures are proposed. There are multiple large-scale datasets like SQuAD [1, 2], Natural Questions [3], which become the benchmarks for this task. Also, multilingual datasets like TyDiQA [4], or datasets like FQuAD [5] for French, GermanQuAD [6] for German, and ThQuad [7] for the Turkish are generated, but they have limited data compared to English datasets. Because of that most of the work done in the question-answering problem is over the English.

Publicly available question-answering datasets for languages other than English are fairly limited. The size of the dataset is important to exploit the power of neural networks for downstream tasks. For example, training general purpose question answering models in Turkish requires a large-scale question-answering dataset in Turkish. Since generating a large-scale dataset from scratch is a challenging process, the generative language models can be used to automatically generate question-answer pairs to ease the process for the human annotators or they can be used directly for training a question-answering system.

In an effort of closing the gap between English and other languages in dataset size, we propose a framework to generate question-answering data using a generative

model and unannotated passages to generate a new dataset or augment the question-answering datasets in a given language. The generation framework consists of two parts. The first one is an encoder-decoder transformer-based question-answer pair generator which uses the text paragraphs as input. The second one is a filtering layer which is utilized to filter the generated questions to improve their quality.

The proposed question-generation model utilizes the mT5 [8] model which is a multilingual model. Being a multilingual model allows us to use same the pre-trained model while fine-tuning the model in the language of interest. This helps with the applicability of this method to multiple languages. We train our model in a multi-task fashion, using question-answering, question-generation, and answer-extraction tasks. Since we utilized an encoder-decoder architecture we were able to train the model in multiple tasks simultaneously without any architecture or loss change. The generation module is fine-tuned using an available question-answering dataset. The dataset is manipulated to generate data for 3 different tasks question generation, question answering, and answer extraction, which are used in training.

Our filtering model uses multiple approaches. The first approach is a heuristic-based approach in which we use language-specific rules together with pre-trained models to determine if the question-answer pair is of high quality. In our second approach, we trained a classifier model based on ELECTRA [9] to classify the good and bad questions. This approach was dependent on the annotation of generated questions, so we used our annotation tool to annotate a random set from the generated question-answer pair.s Because of the language-specific efforts, even though it was helpful, we omit the filter layer in most of our experiments.

Using the proposed question-generation model, we first generated a Turkish question-answering dataset over Wikipedia articles, similar to SQuAD [1] dataset. The largest human-annotated question-answering dataset in Turkish is the ThQuAD dataset [7], which contains questions about Islamic Science and Ottoman History. There is also a Turkish machine-translated version of the SQuAD dataset [10] which

contains articles from Wikipedia. We wanted to introduce a third dataset, that is generated over the articles from diverse topics from Turkish Wikipedia automatically. The datasets we discussed beforehand had questions from 750 and 500 different articles respectively, with our work we introduced a dataset generated with over 20K different articles. We think that this would help to improve the generalization performance of Turkish question-answering models across different domains. We train the question-generation model using the ThQuAD dataset and machine-translated version of the SQuAD [10]. The input for this question-generation model in generation time was the paragraphs that were crawled from Wikipedia.

We also utilize our proposed approach as a data augmentation method in low-resource question-answering settings. In order to simulate limited data settings, we sampled various sizes of small partitions from SQuAD, Spoken SQuAD, FQuAD, and GermanQuAD datasets. These small partitions were used to train the question-generation models and the remaining articles in these datasets were used as input to our question-generation model, and it generate the question-answer pairs using the paragraphs in these articles. Afterward, we trained various question-answering models using these subsets and generated questions to evaluate the impact of the generated questions as input to these models. Contributions of this thesis are:

- We propose a question-answer pair generation model to generate question-answer pairs from given text passages.
- We propose two filtering methods for filtering out the erroneous question-answer pairs generated by the question-generation model.
- We generate a synthetic dataset using paragraphs from Wikipedia in Turkish containing 83.6K question-answer pairs using our framework.
- We extensively test our generation model as a data augmentation method in low resource question answering settings and show that adding the generated question to the question-answering training improved the performance.

2. RELATED WORK

2.1. Question Generation and Filtering

The question generation research revolved around recurrent neural networks before 2019. Many RNN-based encoder-decoder networks were proposed for this task. One such work is Learning to Ask [11] which uses a hierarchical Bi-LSTM encoder for passages. The first layer is a Bi-LSTM-based sentence encoder for sentences. These sentence representations were utilized to encode the paragraphs with another Bi-LSTM. Then an LSTM-based decoder was used to generate questions using the encoded passage information.

Du [12] proposes a 2 step approach. First, the possible answer spans are identified using a Bi-LSTM-CRF model. Following that, a Bi-LSTM encoder and LSTM decoder are utilized. The feature-enriched input, containing word embeddings, answer position features, and coreference features extracted by a coreference resolution model, is taken by the encoder. Then the encoder output is fed to the decoder for generating the questions.

"Answer-focused and Position-aware Neural Question Generation" [13] utilizes a Bi-LSTM encoder that takes word embeddings, answer position, and lexical features like named-entity and POS tags as input when training. A unidirectional LSTM is used as the decoder. This decoder was based on a pointer-generator network that can generate a word from a given vocabulary or copy words directly from the input sequences. Position embeddings are also added to attention calculation to incentivize the selection of text closer to the answers. Another mechanism that detects possible question words from answers is also introduced to encode answer information and pass it to the generation process.

"Self-Training for Jointly Learning to Ask and Answer Questions" [14] utilizes a similar Bi-LSTM encoder and unidirectional-LSTM decoder for question generation. There is also another Bi-LSTM-based network for the question-answering task. Those two models are trained in a self-training fashion. At the first iteration of this training scheme, the training of both models starts with the annotated dataset. After the first pass, new questions are generated by the question-generation model and introduced to the system and those are answered by the question-answering model. Then an oracle model makes a selection among the generated question-answer pairs using the confidences of the QA and QG models. The selected examples are added to the training set of those models. This iteration process continues until the performance on the development set plateaus.

Transformers [15] have become the mainstream neural models for QG after 2019. Recent research on QG has utilized the open source pre-trained models like BERT [16], GPT [17, 18] and T5 [19] architectures.

Lopez [20] proposed a GPT-2 [18] based model in which two different input models were experimented with for question generation. In the training of the first one, the passage is concatenated with all of the questions, using the "Question:" token before each of them. This is done for training the model to generate multiple questions in a single pass to speed up the process. This approach sometimes results in inputs longer than the context length of the GPT-2 model. For alleviating this problem a second method is introduced. In this approach, the passages are duplicated and each is concatenated with one of the questions to this passage in the same fashion. This led to multiple training examples for one passage. The generated questions are evaluated using automatic methods like the BLEU score.

Liu proposed [21] adding extra two inputs "clue" and "style" on top of the paragraph and answer to the GPT-2 model. Since question-answering datasets that are used to train the question-generation model do not contain those two features auxiliary methods are used to determine those from this dataset. The clue is copied or

paraphrased block of text from the passage in question. The clue block is determined by counting overlapping tokens and stems between questions and blocks of passage. The second extra information, "style", is the type of questions like "why", "when", "yes-no" questions, etc. Style is determined by making a string search using the question words like "why", "what" etc., and the possible start of yes-no questions like "is", "should" etc in the question. All of the features are concatenated with special tokens like $< clue >$, $< answer >$ in between. This input is then fed to the GPT-2 model to generate questions.

UNILM [22] proposed a new Transformer based model pre-trained by multiple language modeling objectives, that are sequence-to-sequence, bidirectional, left-to-right, and right-to-left jointly. This paper contains answer-aware question generation experiments using the sequence-to-sequence model. In which a concatenation of passage and answer span is given as input. The target sequence for fine-tuning is the question from the annotated dataset.

Yin [23] used two-stage architecture for question generation. The first stage was a BERT encoder which encodes context with answer spans highlighted with special tokens. The second stage was a BERT as an LM model and a transformer-based pointer generator network. Their experiments show that this two-step approach had better generalization performance to out-of-domain data.

In recent years question generation is also been utilized for data augmentation and domain adaptation for question-answering systems [24–28]. For making those models generate questions from passages without any answer annotation, different architectures have been proposed. Unlike the question generation models discussed before, they are primarily evaluated with the generated questions' impact on the question-answering performance.

Shakeri [24] proposed an end-to-end question-answer pair generator using the BART [29] model. The problem is modeled as a conditional generation task in which a

passage is fed to the model and then the model generates questions and answers consecutively. 3 different generation approaches are proposed. First one takes the input passages and then generates answers then the questions. The second approach is a similar model that takes passages as input to the encoder. Then it generates the questions and then answers in this order. The only difference between the first two approaches is the order of generation. The third one involves a 2-step approach. First, the input passages were passed through an encoder to obtain questions. Then the questions and the passages were fed to the encoder again to generate corresponding answers. Questions with low log-likelihood scores were eliminated. These approaches were evaluated for domain adaptation between different question-answering datasets and the best-performing approach was the 2-step one. The addition of the generated questions on top of the out-of-domain human-annotated data improved the performance by around 8-10% on average on various datasets.

Alberti [25] introduced two models based on BERT architecture. The first model takes passages and extracts candidate answer spans. The second model is fine-tuned as a left-to-right language model which is obtained by altering the attention mask of the BERT encoder. During prediction, it takes the context and answer as input and generates the question using iterative greedy decoding. This work also introduced Roundtrip Consistency for filtering the generated questions using another question-answering model. With this method, each generated question and the corresponding passage are fed to another question-answering model, and if the prediction of this question-answering model matches the generated answer then this question-answer pair is considered to satisfy roundtrip consistency and kept in the dataset, otherwise it is filtered out. Using this approach 3 million questions over SQuAD2.0 [2] passages and 4 million questions from Natural Questions Dataset [3] were generated. Automatically generated questions improve the EM score by 2% and the F1 score by 1.7% on the SQuAD 2.0 dataset.

Puri [26] proposed a 3-step approach. For the answer extraction a BERT model is used and the question generation model is replaced with decoder-only GPT-2 ar-

chitecture, which is more suited to generation tasks, finetuned on question generation using a question-answering dataset. Also, a BERT-based question-answering model is used for roundtrip consistency filtration. 20 million questions were generated using this model over Wikipedia articles. The experiments showed that at this scale training the model with only synthetic data outperforms the model trained with annotated data over the SQuAD dataset.

Another work [27] also proposed a 3-step approach using ELECTRA for the answer span extraction and the question-answering models for round trip filtration. A BART-based generator model is used to generate questions using passages and answers. In their approach, they use a self-training scheme to iteratively train the question generator and question-answering models to improve performance. Augmenting the generated questions for domain adaptation for various question-answering datasets improves the EM performance by around 11% on average and, 2% of this improvement is due to the proposed self-training approach.

Bartolo [28] proposed another 3-step model that involves using an answer candidate selection model to determine possible answers by taking passages as inputs and returning answer spans. A self-attention labeling method is proposed for determining candidate answer spans. This method employs a multi-label classification head for predicting the candidate start and end tokens simultaneously, generating a binary label for each span to determine whether it is a candidate answer or not. Subsequently, a question generation model based on BART, which takes a paragraph and answer span as input, is used to generate the question. Multiple techniques are proposed for filtering the question-answer pairs such as confidence measures of answer candidates and question generators, influence function, and Ensemble Roundtrip Consistency using six models. Finally, a self-training method is also introduced to relabel or eliminate question-answer pairs based on the outputs of the six models.

Our proposed method uses a two-step generation architecture, in which the first step was extracting the answer spans and the second one was using the passage and

answer spans to generate questions similar to some of the recent work [24–28]. In contrast, we only used a single encoder-decoder architecture mT5 [8], which is trained in multitask fashion to determine the answer spans and generate questions. We used QA filtering only in the Turkish question-answering dataset generation setting, because our experiments were mostly focused on low-resource settings and the question-answering performance was relatively low so the roundtrip filtration method which is widely used by different papers was not effective as their settings. We needed to implement language-specific rules and annotate data to train classification models to classify good and bad questions. Since it was a strenuous process we only select the Turkish for our filtering experiments, and focus our efforts to evaluate our question generation performance over various settings and languages.

2.2. Question Answering

Question answering in this thesis focuses on reading and listening comprehension tasks. The reading comprehension involves models working on textual input and the answers are extracted from textual documents. Listening comprehension is the task of extracting answers from spoken documents. Two prevalent approaches are used for listening comprehension tasks: the first involves a combination of automatic speech recognition (ASR) and reading comprehension models (cascade models), while the second approach entails end-to-end training of a model with spoken documents.

In recent years, reading comprehension models have been generally based on various transformer architectures. BERT [16] proposed a fine-tuning scheme with a question-answering head, which determines the start and the end span of answers on top of the transformer encoder embedding. Many other transformer encoder architectures like AlBERT [30], RoBERTa [31], and ELECTRA [9] also utilize a similar head architecture for the reading comprehension task.

The encoder-decoder architectures like T5 [19] and BART [29] are also used for reading comprehension tasks. Since they are sequence-to-sequence models they don't

need the task-specific head like the encoder-only models. The models are trained with the concatenation of questions and passages as input to generate target answer sequences using cross-entropy loss.

Large language models are also utilized for the reading comprehension task. Brown and Wei [32, 33] discuss the large language model's few-shot learning capabilities after pre-training. These models are not up to the performance of the fine-tuned models but their out-of-the-box capabilities are quite promising.

For the listening comprehension task, one of the most prominent approaches is the cascade model architecture. Cascade models first utilize an Automatic Speech Recognition (ASR) system to obtain transcriptions of spoken documents, then the question-answering task is performed on these erroneous transcriptions using either RNN-based or transformer-based models [34–39]. On the other hand, end-to-end models leverage acoustic and text data together to optimize question-answering performance [40, 41].

In this thesis, we use reading and listening comprehension models to evaluate the impact of the generated data in various languages and data settings. We used multiple transformers-based models for both reading and listening comprehension tasks such as BERT [16] and mT5 [8], and ELECTRA [9]. For the listening comprehension task, we focused on the second part of the cascade architecture which is the question-answering model for our experiments.

3. METHODOLOGY

In this thesis, we aim to improve the question-answering performance by using question generation over unannotated text data for low-resource domains and languages. Since data annotation for question-answering is costly, we generated a question-answering dataset over plain text without the need for human supervision. We used a multilingual text-to-text transformer model as our question generation model. Using multilingual models allows us to easily adapt to this approach to multiple languages. The generated questions are also used to augment the human-annotated data and fine-tune question-answering models for the evaluation of our question-generation performance.

3.1. Datasets

In our research, we explored the proposed data augmentation and fine-tuning framework for low-resource question-answering settings using reading and listening comprehension datasets in various languages. For reading comprehension, we utilized SQuAD [1] for English, TQuAD [42], THQuAD [7], and machine-translated SQuAD [10] datasets for Turkish; FQuAD [5] for French, GermanQuAD [6] for German, and two datasets collected and annotated from video lectures, one for English lectures [37] and one for Turkish lectures [37]. For listening comprehension, we utilize Spoken SQuAD [38] for English and spoken versions of the lecture datasets [37].

Our research mainly focuses on QA with limited annotated data. The lecture datasets have less than 1000 questions for training so we used the whole dataset in our experiments. The other datasets have more than 10K questions so to simulate low-resource settings we used various subsets from these datasets. The overall sizes of those datasets can be seen in Table 3.1. For SQuAD, Spoken SQuAD, FQuAD, and GermanQuAD some train data are set apart as the development set to tune the model hyperparameters. The official SQuAD development set is used as the test partition in

the experiments. English lectures dataset does not have a development set, so 2-fold cross-validation is applied on the test set to tune the model hyperparameters.

Table 3.1. Dataset statistics: number of articles, paragraphs, and question-answer pairs for each dataset used in the experiments.

Dataset	Train Set			Development Set			Test Set		
	Articles	Paragraphs	QAs	Articles	Paragraphs	QAs	Articles	Paragraphs	QAs
SQuAD [1]	397	17070	76065	45	1826	8534	48	2067	10570
FQuAD [5]	105	4449	18569	12	472	2162	18	768	3188
GermanQuAD [6]	2286	2286	10376	254	254	1142	474	474	2204
Spoken SQuAD [38]	397	17070	33589	45	1826	3522	48	2067	5351
English Lectures [37]	26	259	309	-	-	-	15	94	175
Turkish Lectures [37]	75	547	878	13	118	286	27	178	287
TQuAD [42]	681	2232	8308	-	-	-	72	275	892
THQuAD [7]	756	2400	14224	-	-	-	85	301	1330
SQuAD-TR [10]	442	18776	61293	-	-	-	-	-	-

3.1.1. SQuAD

SQuAD [1, 2] is an English reading comprehension dataset that consists of short passages from Wikipedia. It includes approximately 23K paragraphs extracted from 536 articles covering diverse topics, along with over 100K human-annotated question-answer pairs. While the training and development sets are available online, the test set is reserved exclusively for the challenge. For our research, we used the SQuAD v1.1 training and development splits. The official development set is employed to evaluate the performance of the proposed framework. As a result, we reserved 10% of the training set for hyperparameter tuning for both question-generation and question-answering models.

3.1.2. Spoken-SQuAD

The Spoken SQuAD dataset [38] is an English listening comprehension dataset that was created from the SQuAD dataset. The process involved converting the SQuAD passages into the spoken form using Google Text-to-Speech and then generating cor-

responding ASR transcriptions using CMU Sphinx. If the answer to a question was transcribed incorrectly, that question-answer pair was removed from the dataset. As a result, spoken SQuAD is a subset of SQuAD that includes around 37K and 5.3K question-answer pairs for the train and test sets, respectively. Additionally, a development set was created by setting apart 10% of the training set for tuning the hyperparameters of the question-generation and question-answering models.

3.1.3. TQuAD

TQuAD is a Turkish reading comprehension dataset containing questions on Turkish Islamic Science topics. In the dataset, there are 8.3K questions over 2.2K paragraphs from 680 articles for the training set. The development set contains 900 questions over 275 passages from 72 articles. We did not use this dataset directly in this work, we only used the extended version of this set, which is THQuAD.

3.1.4. THQuAD

THQuAD is a Turkish reading comprehension dataset that contains questions on Turkish Islamic Science and Ottoman History. It contains the TQuAD dataset and extends it with Ottoman History questions. In the dataset, there are 14K questions over 2.4K paragraphs from 756 articles for the training set. The development set contains 1330 questions over 301 passages from 85 articles. We used the training set for the question-generation and question-answering model training and the development set for testing the question-answering performance. We set apart 10% of the training set as a development set for hyperparameter tuning of the question-generation and question-answering models.

3.1.5. SQuAD-TR

SQuAD-TR [10] is a Turkish reading comprehension dataset. It is generated by translating the SQuAD 2.0 dataset using Amazon Translate API. It contains 61K

question-answer pairs over 18.8K paragraphs. We utilize this dataset while training our question-generation model for Turkish.

3.1.6. FQuAD

FQuAD is a French reading comprehension dataset containing passages from French Wikipedia and 20K human annotated question-answer pairs [5]. In our work, we used the train, and development splits of FQuAD v.1.0. The official development set is used to evaluate the performance of the proposed framework, so we set apart 10% of the train partition for hyperparameter tuning, both for question-generation and question-answering models.

3.1.7. GermanQuAD

GermanQuAD is a German reading comprehension dataset containing passages from Wikipedia and 13,000+ human annotated question-answer pairs [6]. It is based on the German equivalent of English articles in SQuAD, the corresponding questions were reformulated into German, so that the training dataset contains about 11K annotated question-answer pairs while the test set has about 2K pairs. In our work, we used the train and test splits of the GermanQuAD dataset. GermanQuAD does not have a development set, so we set apart 10% of the train partition as the development set for hyperparameter tuning, both for question-generation and question-answering models.

3.1.8. English Lectures

English Lectures is an English reading and listening comprehension dataset collected from lecture domain [37, 43]. The train partition of the English lecture data contains video recordings (11.4 hours) and their corresponding reference transcriptions collected from MIT OpenCourseWare Signals and Systems course [44]. The test partition of the English lecture data contains the video recordings (1.2 hours) and reference transcriptions of the same course taught at MEF University. The reference transcrip-

tions were first split into short passages and then manually annotated to generate the question-answer pairs, resulting in 309 and 175 question-answer pairs respectively for train and test partitions. To obtain the listening comprehension dataset, the recordings were decoded using Kaldi [45] resulting in 19.2% and 6.7% WERs respectively for the train and test partitions. Since this dataset contains very limited data, we applied 2-fold cross-validation on the test set instead of setting apart a development set from the train partition.

3.1.9. Turkish Lectures

Turkish Lectures is a Turkish reading and listening comprehension dataset collected from the lecture domain [37]. The dataset contains video recordings and their corresponding reference transcriptions collected from law courses offered at MEF University. The data were split into training (12 hours), development (2 hours), and test (4 hours) sets. The reference transcriptions of each set were split into short passages and then manually annotated to generate the question-answer pairs, resulting in 878, 286, and 287 question-answer pairs respectively for training, development, and test partitions. To obtain the listening comprehension dataset, the recordings were decoded using Kaldi [45] resulting in 14.7%, 12.8%, and 14.4% WERs for train, development, and test sets, respectively.

3.1.10. XQuAD

XQuAD (Cross-lingual Question Answering Dataset) is a benchmark dataset for question-answering models for various languages. The dataset consists of 1190 questions over 240 paragraphs. The dataset is generated by translating SQuAD v1.1 dataset into 11 languages (Spanish, German, Greek, Russian, Turkish, Arabic, Vietnamese, Thai, Chinese, Hindi, and Romanian). Translations are made by professional translators. We used the Turkish part of this dataset for evaluating our Turkish question-answering models.

3.1.11. Turkish Wikipedia Cleaned Dump

For generating this dataset, we first collected Turkish Wikipedia pages from Wikipedia Data Dump. The resulting number of articles was about 460K. We collected those pages using a Wikipedia XML parser [46]. With this parser, for each page, we generated a title, a subject, a table, and the preprocessed contents of the articles, which split into paragraphs. The subject was the class of the article type, like "athlete", "politician" "city" etc. found in the metadata of the articles. The table was the block that can be seen in the top right of the document just below the image which contains some factual information like birthday, age, location, etc. This table could also be utilized to determine extra answer spans on top of our answer extraction module, but we did not utilize this information at least as of now.

We filtered out articles with missing subject fields, which tend to be lower quality articles like containing one sentence etc. After that, we also remove the articles with non-Turkish characters to eliminate any possible different language text. We then select the first paragraphs of the remaining articles. The idea behind this is that the first paragraph is usually the summary containing much factual information about the topic, which is a good candidate to generate question-answer pairs. We further filter out these paragraphs if they have less than 40 words since the shorter ones were mostly because of either bad parsing, or just short intros that do not contain much question-answer material. The final number of paragraphs was 20.4K from 20.4K articles since we only used the first ones. The paragraphs of this dataset are used for generating Turkish question-answer pairs using the trained question-generation model.

3.2. Models

We employed mT5 model as our question generation architecture. For question-answering, we used BERT, ELECTRA, XLM-RoBERTa, and mT5 models. For the question filtering, we used ELECTRA-based models.

3.2.1. mT5

mT5 [8] is based on the T5 [19] architecture, which stands for Text-to-Text Transfer Transformer and consists of a transformer-based encoder-decoder architecture that is trained on a large text corpus. Text-to-text format simplifies the training of a model for various downstream NLP tasks. This architecture allows for training each task by taking a text input and generating the corresponding target text. Consequently, it becomes feasible to use the same model, loss function, and hyperparameters across all tasks. mT5 model has the same architecture but was trained with multilingual data which contains text from 101 languages gathered from public Common Crawl web scrape. mT5 was trained using a supervised learning approach, where it was trained on a variety of tasks such as language modeling, machine translation, and summarization. mT5 has achieved state-of-the-art results on wide range of benchmark datasets, including those for natural language generation, summarization, and question-answering.

We used this model as the base of our question generation architecture. We also use this model for some question-answering experiments.

3.2.2. BERT

BERT is a bidirectional transformer encoder, which means that it takes into account the context of both the preceding and the following words when predicting the missing word in a sentence. BERT also incorporates a novel technique called masked language modeling (MLM), where random words are masked out and the model is trained to predict them based on the context of the surrounding words.

We used language-specific BERT models and also a multilingual BERT model trained in over 100 languages for our question-answering experiments.

3.2.3. ELECTRA

ELECTRA [9] is a pre-trained transformer-based model which contains a small generator and a discriminator. The key innovation of ELECTRA lies in its discriminator training objective, which allows it to achieve state-of-the-art performance on various NLP tasks while being more computationally efficient than other pre-trained models such as BERT.

In traditional pre-training methods, the model is trained to predict missing words or generate text based on a given prompt. However, ELECTRA's discriminator training approach involves training the model to distinguish between "real" and "fake" examples. The fake examples are created by replacing some of the words in the real examples with randomly generated ones.

We used ELECTRA in question answering and question filtering parts of our research.

3.2.4. XLM-RoBERTa

XLM-RoBERTa [47] is a pre-trained language model that is based on transformer architecture and is designed to handle multilingual NLP tasks. XLM-RoBERTa is an extension of the RoBERTa model, which was pre-trained on a large English corpus of using a masked language modeling (MLM).

XLM-RoBERTa was trained on diverse text data from over 100 languages. Pre-training was performed using two tasks: MLM, where random words in a sentence are masked out and the model is trained to predict them based on the context, and a new task called cross-lingual language modeling (XLM), where the model is trained to predict the language of the input text. The XLM task forces the model to learn a universal representation of language that can transfer knowledge across different languages.

We used XLM-RoBERTa as one question-answering benchmarking method in this work.

3.3. Architecture

This work consists of multiple parts. These parts are question generation, question filtering, question answering, and an annotation tool to annotate the generated questions as good and bad questions. The question-answering part is mostly used to benchmark whether the question generation and filtering methods work well since our aim is to improve question-answering performance using automatically generated data. Figure 3.1 shows the overall architecture of our framework. An annotated question-answering dataset is used to train the multitask model which can perform answer extraction and then question generation conditioned to those answers. In the inference time, unannotated paragraphs are used to generate a synthetic question-answering dataset using the trained multitask model. The generated question-answer pairs are then filtered by question quality filters. The automatically generated and human-annotated question-answering data are used to train the question-answering model which will be used to evaluate the effectiveness of the framework.

3.3.1. Question Generation

Given a piece of text, such as a paragraph or a document, the goal of question-generation is to automatically generate one or more questions that are relevant to the content of the text. The generated questions are expected to be grammatically well-formed and to be answered by the knowledge present in the text. In our framework for the question-generation model, we use mT5 [8], which is a multilingual transformer network with an encoder-decoder architecture. mT5 model is pre-trained on a large multilingual dataset for masked language modeling that also involves reconstructing the masked-out consecutive spans of input tokens.

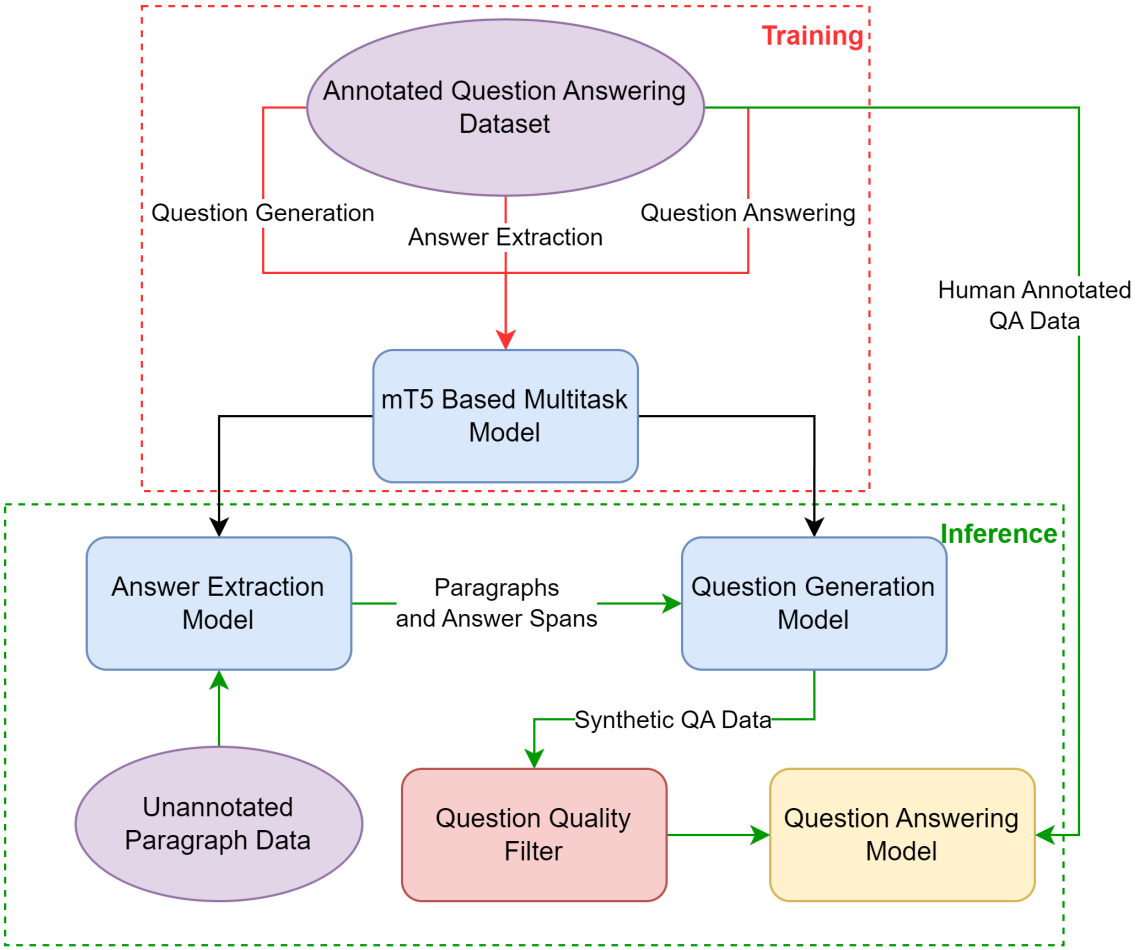

Figure 3.1. Overall architecture of the proposed framework.

For fine-tuning this question generation network, we utilized a multitask approach that contains answer span extraction, question generation, and question-answering data prepared from annotated QA datasets.

The relevance and quality of the generated questions are important for the performance of the tasks utilizing the generated questions, i.e., question answering. Relevance refers to how well the generated questions relate to the content of the input paragraph, and quality refers to how well-formed and meaningful the questions are. The metrics such as BLEU and ROUGE are also used to evaluate the question-generation model performance by measuring the degree of overlap between the generated questions and a set of reference questions.

3.3.1.1. Training Tasks. The question generation model is formulated as the joint conditional distribution $P(a, q|p)$ which can be factorized as $P(q, a|p) = P(a|p)P(q|a, p)$ where p is the input paragraph, q is the question and a is the answer span.

The question-generation model is fine-tuned using annotated data containing (a, p, q) triples with a multitask objective where the tasks are answer span extraction, question generation, and question answering. The answer span extraction task generates possible answer spans ($\hat{a}$) given the input paragraphs ($\hat{a} \sim P(a|p)$). The question generation task generates questions given the extracted answers and the input paragraphs ($\hat{q} \sim P(q|\hat{a}, p)$). The generated questions and input paragraphs are used in question-answering to predict the answers. Question answering is more like an auxiliary task, which helps the model learn the relation between questions and paragraphs.

3.3.1.2. Model Architecture and Training Data Preparation. The question generation architecture is founded on mT5 [8] architecture, which is a text-to-text transformer that takes a text input, encodes the input with an encoder, and produces a text output with a decoder. This makes the training of this model with multiple tasks and data sources easier since the same architecture can handle many tasks, rather than needing to add different layers on top of the transformers network.

For the implementation, we used the HuggingFace [48] implementation of the mT5 model and tokenizer. For the loss we used label smoothed negative log-likelihood loss over the generated output tokens.

We used a semi-structured input format for model training. For each task, we used key phrases to guide the model about the task and the input structure of this task. This semi-structured input helps the model to determine the parts of the input and task reliably and make better predictions.

For the answer span extraction task, we first split the paragraphs into sentences. After that, we determined which answer belong to which sentence using the sentence positions and answer positions in the paragraph. We got the mapping of answers and the sentences that they are found. For each group of answers that are extracted from the same sentence, we highlight their sentence in the paragraph with *<hl>* token. This marking tells the model to extract the answer spans only from this sentence. We also added *"extract answers:"* key phrase to tell the model that the task is answer span extraction. The target text for training the model was obtained by joining the answers of this sentence with a separator token. The input for this task is formatted like the following: *"extract answers: <start of paragraph> <hl> <highlighted sentence> <hl> <rest of paragraph>"*.The target text is like the following: *"<answer 1> <sep> <answer 2> <sep> .... <sep> <answer N>"*

For the question generation task, we used *"generate question:"* as the task identifier key phrase. We then highlight the answer span, that we want the model to generate questions in the paragraph by putting the span between the <hl> tokens. We also prepended this answer string before the whole highlighted paragraph to give the model a clear distinction between the answer span and the context. For the prepending of this answer, we use *"answer:"* key phrase, and for the paragraph with the highlighted answer we used *"context:"* key phrase as a prefix. The target text for this input was directly the question that we want to generate. We did not use any extra tokens or key phrases.The input for this task is formatted like the following: *"generate question: answer: <answer> context: <start of paragraph > <hl> <answer> <hl> <rest of paragraph>"*. The target text is like the following: *"<question>"*

For the question-answering task, we used *"answer question:"* as the task identifier key phrase. We then concatenate the question and paragraph using the *"question:"* and *"context:"* key phrases respectively. Since the question and paragraph are disjoint unlike the answer and paragraph we could not do any highlighting for the training of this task. The target text is directly the answer to the questions. We did not use any extra tokens or key phrases. The input for this task is formatted like the following:

"answer question: question: <question> context: <paragraph>". The target text is like the following: *"<answer>"*

3.3.1.3. Generating Question-Answer Pairs. After fine-tuning, the question generation model first predicts possible answer spans from the unannotated data (paragraphs). Then, the questions are automatically generated using extracted answers and the input paragraphs. It works like the following:

- Split the paragraph into sentences and generate multiple paragraphs where in each of them one of the sentences is highlighted with <*hl*> token. Add *"extract answers:"* task identifier key phrase at the beginning of each of those paragraphs.
- Pass those paragraphs through the model to get possible answers from each of the sentences and calculate the spans of those answers with respect to the paragraph.
- Using the predicted answers and the paragraph generate the question generation inputs. Highlight the answer span in the paragraph with <*hl*> token and also prepend the answer text to the paragraph passage. Add *"generate question:"* task identifier key phrase at the beginning of the input.
- Feed the inputs to the model to generate a question for each answer.
- Collect the question-answer pairs and format them as a SQuAD-like dataset to easily integrate them with other question-generation datasets.

3.3.2. Question Quality Filters

The question-generation model is generating question-answer pairs but not all of those question-answer pairs are of equal quality. As expected the model sometimes fails to generate complete questions or meaningful questions. We aim to eliminate those low-quality question-answer pairs with this filter model, to improve the performance and possibly shorten the training time, since we will have less but higher-quality data.

We tried two different methods to eliminate those questions. The first one is the heuristics-based filter, which doesn't need any data regarding the quality of the

question-answer pairs. We use some heuristics about the question structure in the language and also use some out-of-the-box models that can be found in the HuggingFace [48].

In the second method, we annotate the quality of the question-answer pairs using the question annotation tool and use this data to train a classifier that distinguishes the good and bad question-answer pairs. Since the annotating was taking a long time we only made experiments for the filter methods with the Turkish.

3.3.2.1. Heuristics Based Filter. This filter has consist of multiple checks which are both based on the structure of the questions in language and also some pre-trained models that can be found in the HuggingFace library or can be fine-tuned with the data that will be used for the question generation module. We did not need to make any extra annotation with this method, which shortens the overall development time. The checks used in this filter are:

- *Does question contain a question word?:* This filter has some predetermined list of words that points to that sentence that can be a question; like "when", "what" etc. Using this list of words the filter searches them in the questions to determine if that question contains any question word which points that this question can be valid. Since the questions in QA datasets are mostly "wh" questions, any candidate question should use one of those question words. If we can not find any of those words the question is most likely invalid.
- *Does the question end with a question mark?:* This checks if the question ends with a question mark. With this rule, we try to determine if the question is complete by having a punctuation mark at the end, which points to that the question is complete.
- *Does the question contain repetitive stems?:* This check tries to determine if the question contains words with the same stem for an unnatural amount (more than 30% of all stems are the same). This rule is added because sometimes the question-generation model tends to generate words with the same stem but differ-

ent suffixes consecutively. Those sentences were most of the time not meaningful. For getting the stems of the word we used different Zemberek [49] morphological analysis tools.

- *Sentence structure check using ELECTRA Discriminator:* Since the discriminator part of the ELECTRA model was trying to find the words which are replaced by the generator in the pre-training, we wanted to utilize this discriminator to run over the generated question and check if any of the tokens are not belonging to the sentence. For this task, we directly get ELECTRA Discriminator from the HuggingFace and did not fine-tune it over our data.
- *Roundtrip Consistency:* In this check we first fine-tune BERT based question-answering model with our question-generation training data. We then feed the generated question and paragraph to this model and get the predicted answer from this model. If the predicted answer is not the same as the original answer, we eliminated those questions. This method is proposed for the question generation system in this paper [25].

Those checks are run on the question-answer pairs and their results are then aggregated with "and" to determine the final verdict of the question-answer pair.

The heuristics-based classifier was faster to implement but it had its limitations. Except for the Roundtrip Consistency, no module would capture the relationships between the question-answer pair and paragraph. Even then this check is only using question and passage as the input which discards the possible intricate relation between the answer and other parts.

3.3.2.2. Deep Learning Model Based Filter. This model aims to capture more complex relationships between question-answer pairs and paragraphs which could not be captured by the heuristics that we can come up with. Since even though the question and answer are valid separately, this doesn't mean that they would be not meaningful in the context of this paragraph. We wanted to classify the validity of question-answer pairs using the relationship between them and the paragraph.

We used an ELECTRA discriminator-based sequence classifier, for this task and utilized the HuggingFace implementation of the *ElectraForSequenceClassification*. For the input of this model, we used our annotation tool to annotate high and low-quality questions among the generated questions. In our dataset, we label each question-answer pair as valid or not. We did not use more fine-grained labeling to simplify the annotation process. For the input of the model, we concatenated the question, answer, title, and paragraph. We used the *[SEP]* token to separate those different parts in the input. The input for this task is formatted like the following: *<question> [SEP] <answer> [SEP] <title> [SEP] <paragraph>*

We try to give all the possible text as input to our model to be able to capture the relationships between passage and question-answer pair, which would capture more intricate relationships than the heuristic-based classifier to have better performance.

3.3.3. Question Annotation Tool

After generating the question-answer pairs over paragraphs we also wanted to have a way to inject human supervision into the loop. So we implemented a basic annotation tool in which we can load a set of questions and it displays the passage and the generated question-answer pairs from this paragraph. Users then can select the question-answer pairs that are valid and eliminate the rest. They also can select a new answer span for the questions that are valid and answerable but the answer span is wrong.

3.3.4. Question Answering Models

We used various question-answering models to measure the performance improvements gained from the generated questions. We used metrics like ROUGE and BLEU to measure the question generation performance but in the end, we wanted to improve the question-answering performance using the question generation. For those models, we used the implementations from HuggingFace and fine-tune them with various

datasets both human-annotated and generated. We used language-specific models like BERT and ELECTRA when possible, and also made tests with multilingual models like XLM-RoBERTa, mBERT, and mT5.

For the BERT, ELECTRA, XLM-RoBERTa, and mBERT we used the question answering head from HuggingFace which tries to determine the start and end tokens for answers from the paragraph. For the mT5 we used a similar training scheme to our question generation model. The only difference is we discard the answer span extraction and question generation tasks in this training.

For the training of the models, we utilized two different approaches to integrate the generated data into the training scheme. The first one is augmented training. In which we combine human-annotated datasets and generated ones directly and shuffle them. The second one is a 2-step approach, in which we first fine-tune the model with generated set and then fine-tune with the human-annotated one.

3.3.4.1. Augmented Training. In the augmented training we directly merge the human annotated dataset with the generated dataset and shuffle them. That is the simpler method in which we expect the extra questions from the generated data to improve the performance. Also adding questions from other domains or sources generated with the question generation model would improve the dataset quality even though the individual questions might be suboptimal.

We expect this version to perform better especially if there is a domain drift between the human-annotated dataset. In this setting we can easily augment the train set with generated data from target domain paragraphs, using the question generation model trained with human-annotated question-answer pairs.

3.3.4.2. 2 Step Training. This method splits the training into 2 steps. In the first step, we train the model with only generated questions. The idea behind this is that gener-

ated questions are either easier or lower in quality with respect to the human-annotated dataset. It can be called a noisy dataset because of the errors in the generation. Because of that we first fine-tune the model up to a point using the only generated data and harvest some basic questions answering performance and extra coverage. Then fine-tune this model checkpoint with human-annotated questions, which are of higher quality and harder, to smoothen the possible problematic cases that would be introduced by the generated data.

We expect this version to perform better especially if we have a dataset in the target domain but we are not satisfied by the performance of the model which is trained with this data. Therefore we want to improve the question-answering performance of this model.

4. EXPERIMENTS AND RESULTS

4.1. Question Answering Dataset Generation

In this first experiment, we wanted to use the question-answer pair generation architecture to generate a Turkish Wikipedia dataset like SQuAD. The Turkish Question answering datasets TQuAD and ThQuAD are not open-domain datasets and their questions are in a confined domain. Because of that, we wanted to generate a more general domain dataset that can have more generalizable performance for out-of-domain data and would be beneficial to improve the Turkish question-answering performance in other datasets.

4.1.1. Experimental Setup

The question-generation module was implemented in Python using the HuggingFace library [48]. The pre-trained mT5-small model is used for our question-generation module. The model is trained with batch size 8, and 32 gradient accumulation steps. The effective batch size was 256 and the learning rate was $1e^{-4}$. We fine-tuned our question-generation model for 30 epochs with two Turkish question-answering datasets. The first one was ThQuAD [7] and we also used SQuAD-TR [10] which is an English to Turkish machine translated version of the SQuAD2.0 dataset. The first ThQuAD had 15.4K questions while SQuAD-TR had 61K questions. Since the model is trained in multitask fashion, we generate three data points from each question-answer-passage triplet for the training. You can refer to 3.3.1.2 for further information on the training data generation.

We used the paragraphs from the Turkish Wikipedia Dump discussed in Chapter 3.1.11 to generate question-answer pairs. Using the paragraphs and question-generation model we trained we generated a question-answering dataset over this dump.

We used automatic performance metrics and human evaluation for determining the question-generation model performance. We used BLEU and ROUGE metrics and used ThQuAD and XQuAD passages and answers and fed them through our generation module. We then calculate the ROUGE and BLEU scores between the original question and the generated question to assess the question-generation performance. We make those calculations for both the surface form of the words and lemmatized versions. The lemmatization is done with Zemberek [49] tool. We also conducted a human evaluation of the questions generated from the Wikipedia Passages by our question-generation model, to assess the quality of the questions and their relation with the answers.

The generated dataset is also evaluated using question-answering models. Three models are trained as question-answering models to evaluate our generated dataset's capability. We used BERT, ELECTRA, and mT5 models for this task. We implement the models in Python using the HuggingFace library. The BERT and ELECTRA question-answering models were fine-tuned with a batch size of 16 and a learning rate of $2e^{-5}$ for 20 epochs. The mT5 model used batch size 8 with 32 gradient accumulation steps and fine-tuned for 20 epochs with a learning rate of $1e^{-3}$. We fine-tuned 3 different models for each model type, using different data combinations. The models are fine-tuned with ThQuAD, generated dataset, and a combination of both. We evaluated the models on the ThQuAD test split and XQuAD Turkish split. The combined question-answering model which uses data questions generated by the question-generation model and ThQuAD is trained with the augmented training method in this section.

Table 4.1. Question generation model evaluation with automatic metrics.

		ROUGE L	**BLEU 1**	**BLEU 2**
ThQuAD	**Lemmatized**	0.478	0.485	0.297
	Surface	0.443	0.390	0.235
XQuAD	**Lemmatized**	0.397	0.425	0.192
	Surface	0.328	0.307	0.116

After fine-tuning, the question-generation model generated 83.6K question answer pairs from the 20K passages from the Turkish Wikipedia data. Each paragraph has an average of 4 questions, and the average question and answer lengths are around 7 and 3 words, respectively.

The question-generation performances in Table 4.1 show that generated questions were significantly better in terms of ROUGE and BLEU metrics. This is most probably caused by the ThQuAD data being used in the question generation and since it is a domain-specific dataset question-generation model was able to learn its language better than the XQuAD's question type. This also gives insight into why question-answering models trained with generated data perform better in ThQuAD than XQuAD.

Table 4.2. Scores of different question-answering setups on ThQuAD test set and XQuAD.

	Training Sets	**Test Sets**			
		ThQuAD		**XQuAD**	
		EM	**F1**	**EM**	**F1**
BERTurk	ThQuAD	0.57	0.76	0.47	0.64
	Generated Questions	0.46	0.67	0.43	0.58
	Combined	0.57	0.76	0.50	0.64
mT5	ThQuAD	0.45	0.64	0.33	0.51
	Generated Questions	0.32	0.52	0.32	0.47
	Combined	0.47	0.66	0.39	0.55
ELECTRA	ThQuAD	0.58	0.77	0.46	0.64
	Generated Questions	0.45	0.66	0.44	0.58
	Combined	0.57	0.77	0.52	0.66

The Exact Match (EM) and F1 scores for the question-answering experiments are given in Table 4.2. The fine-tuning with ThQuAD only outperforms the question-answering model which is only fine-tuned by generated data. That is expected in the

ThQuAD dataset because it contains passages and questions specifically from a domain in which the test set also resides naturally. On the contrary, we generated questions using Wikipedia passages that contain various questions from various domains. The XQuAD contains passages from multiple domains from Wikipedia which is much more similar to the generated question-answering dataset. The generated question-answering dataset still was not able to outperform the ThQuAD-only model. This can be due to the noisy nature of the generated questions. The generated model is more prone to semantic and grammatical errors than the human-annotated dataset. The ThQuAD-only model outperforms the generated model by 11% and 9% in EM and F1 scores respectively over the ThQuAD dataset. The performance gap closes to 4% EM and 6% F1 score for the XQuAD dataset. The scores are given for the BERTurk model but the other two models also have similar trends.

We then combined those two datasets for training a third set of models. This combination improves the results for the XQuAD dataset. The EM score of the model is increased by 3% in the BERTurk model, while both the mT5 and ELECTRA models show 6% improvement. The F1 scores here increased by 4% for mT5 and 2% for the ELECTRA-based question-answering model, while the BERTurk model did not see a significant increase in the F1 score. Adding the generated question-answering data on top of the ThQuAD data did not bring any improvements to the results over the ThQuAD test set. This can be because of the domain mismatch between the generated questions and the ThQuAD domain.

We also perform experiments for the listening comprehension tasks with the ASR outputs of the Wikipedia passages, ThQuAD passages, and XQuAD passages. Each passage is first converted to speech using a TTS system and then converted back to text using an ASR system, using the system proposed in [50]. This introduced ASR errors to the systems. We wanted to evaluate how the robustness of the model against the noise introduced by ASR errors, changes with the addition of generated data. We used only the BERTurk model as a question-answering system for those SQA experiments. We removed the data points if the answer is not found in the ASR output

of the passage. This is because since there are no ground truth spans to evaluate the question-answering model performance it was not easy to fairly evaluate the models since some chunk of the data could not be predicted correctly, because there is no ground truth answer span. Also, we could not use the question-answer pairs that lack ground truth answer span in the paragraph with our proposed architecture.

The SQA results are given in Table 4.3. For the ThQuAD test set the generated question dataset did perform the lowest similar to the textual question answering task. This might be again because of generated questions and the ThQuAD test set's mismatch. When we used the combined data the EM score improved by 2% in the ThQuAD dataset which points that generated questions are helping about the noise introduced by the ASR system. In the XQuAD dataset, we see that our generated questions outperform the question-answering model trained by ThQuAD data by 5% (from 0.35 to 0.40) in the EM score and 3% in the F1 score. The best model was again the one trained with a combined version of the data. Which improved the EM score further by 7% and the F1 score by 5%. This shows that our question generation model makes the models more robust to the noise introduced by ASR systems, which can be a helpful tool to improve SQA performance in other datasets.

Table 4.3. SQA performance of the BERTurk model.

QA Data Source	**ThQuAD**		**XQuAD**	
	EM	**F1**	**EM**	**F1**
ThQuAD	0.51	0.74	0.35	0.55
Generated Questions	0.47	0.69	0.40	0.58
Combined	0.53	0.74	0.47	0.63

We manually investigated 12.3K question-answer pairs from 2.8K paragraphs using the Question annotation tool. This is the 14% of the whole generated dataset. According to human evaluators, 55% of the questions were grammatically and semantically correct. 11% of those questions were having incorrect answer spans. Human

annotators also investigated further some of the question-answer pairs to determine the root causes of the errors. Randomly selected 168 question-answer passages are then investigated and the common errors as in the following:

- Factually incorrect(37%): The question-generation model generates questions, but it merges the various factual data from the passage to the point that it is incorrect.
- Semantically wrong (18%): The sentence is not fully meaningful.
- Grammatically incorrect (39%): It has grammatical errors like tenses, usage of suffixes, etc.
- Incorrectly formed entities (6%): The model generates questions with entities not found in the given passages.

4.2. Filtering the Generated Question-Answer Pairs

After generating the question-answer pairs using the generation model, we saw that some of the question-answer pairs are not grammatically or semantically correct. We wanted to test if we can eliminate these lower-quality invalid questions automatically and if question-answering model training benefits from this elimination process.

4.2.1. Experimental Setup

For this experiment, we first work on annotating the questions that the question-generation model generated as valid and invalid. This annotation is done while checking the question-generation results by hand for section 4.1. They are used to train the deep learning model-based question quality filter using our annotation tool. We then split these questions into training and development and test sets to train and evaluate the models. This dataset is generated from the training set of the generate Turkish Question Answering dataset so we omit the questions that are annotated for the question quality dataset from the question-answering dataset which dropped the question count in the training set of this dataset from 71178 to 58811. .

The generated question quality filter dataset had a total of 12367 question-answer pairs. We used 10022 of those questions as the train set, 1082 of them as the development set, and 1263 of them as the test set. In the dataset 55% of those questions were valid while 45% of them were marked as invalid.

Afterwards, we train the ELECTRA-based filter model with the prepared question quality dataset. The filter system was implemented in Python using the HuggingFace library with the pre-trained Turkish ELECTRA model. For training, we used a batch size of 32 with $2e^{-5}$ learning rate and trained the model for 10 epochs. In each epoch, we evaluated the filtering performance over the development split and selected the epoch with the best filtering performance over the development set. Then the filtering performance of this model is evaluated over the test set of the question quality dataset.

Afterwards, we used the ELECTRA-based filter model to filter the remaining questions of the generated dataset. The filtered and unfiltered version of this dataset were then used for training a question-answering model. The question-answering models generated with this dataset were also trained in the same fashion as the BERT question-answering model in Section 4.1.1

For the heuristic-based filter model, we did not use any training data since we handcrafted its rule set as explained in Section 3.3.2.1. Its performance is tested over the test set of the question quality dataset.

4.2.2. Results

The performance of the filters over the test set is listed in Table 4.4. The heuristics-based filter combines the filters in the first five rows of this table. The combined heuristics model had a precision of 0.654 and a recall of 0.693 for filter performance. Considering that 0.55 of the dataset is marked as valid in the ground truth, the filter improves the valid question-answer pairs ratio in the dataset by around 10%.

However, Table 4.5 shows that we are only passing 486+257=743 of the questions out of 1263 questions in the test set, which is around only 60% of the original dataset size. Even though overall question quality is higher the loss on the number of questions is much higher which can lead to a drop in question-answering performance. Because the model still can learn more coarse things even from wrong examples, especially in the earlier steps of the training, but also the model might learn wrong patterns from the generated examples.

Table 4.4. Performance of various filter methods over the test set of the question quality dataset.

	Precision	**Recall**	**F1**	**Accuracy**
ELECTRA Discriminator	0.606	0.874	0.716	0.614
Roundtrip Consistency	0.597	0.807	0.686	0.591
Question Mark Presence	0.568	1	0.725	0.578
Question Word Presence	0.562	1	0.72	0.568
Contains Repetitive Stems	0.559	0.994	0.716	0.561
Heuristics Based Filter (Combined)	0.654	0.693	0.673	0.626
ELECTRA Discriminator + Roundtrip Consistency	0.647	0.698	0.671	0.621
ELECTRA Based Filter Model	0.767	0.795	0.781	0.752
Heuristics and ELECTRA Filter Model Combined	0.802	0.602	0.688	0.697

ELECTRA discriminator and Roundtrip Consistency blocks of the combined heuristics models was the main contributor to the precision increase and the drop in the question counts. ELECTRA discriminator had 0.606 precision and 0.874 recall which was the biggest contributor to the combined filter performance. Roundtrip Consistency was close behind with 0.597 precision and 0.807 recall. The remaining 3 filter blocks had high recall rates because they let almost anything pass the filter eliminating only a small number of questions. So even though their F1 scores are higher than the other filters, they were not highly contributing to the result. As it can be seen from Table 4.4, the precision difference between ELECTRA discriminator + Roundtrip Consistency and Heuristics Based Filter (Combined) is 0.007 (less than 1%), which points to the other blocks' contribution.

The ELECTRA-based filter model has the best performance over the test set. It has a precision of 0.767 and a recall of 0.795. It can get 10% more of the high-quality questions with respect to the heuristics-based method. The model was also able to get those questions 11.3% more precisely. After this filter is used, only 557+169=726 questions passed the filter out of 1263 questions, which is around 57.5% of the dataset. This ratio is even less than the 60% ratio of the heuristics-based filter. Even though the overall filtered question quality is higher, the filtered-out questions can still be valuable in learning. So the loss of them might deteriorate the question-answering performance. On the other hand, eliminating those questions also prevents the model from learning wrong things so it may be beneficial to the performance of the model.

Table 4.5. True positive, true negative, false positive, false negative filter status counts of our different filter models over the test set of the question quality dataset. This also contains the breakdown of each individual heuristic used in the Heuristics Based Filter.

	TP	**TN**	**FP**	**FN**
ELECTRA Discriminator	613	163	399	88
Roundtrip Consistency	566	180	382	135
Question Mark Presence	701	29	533	0
Question Word Presence	701	16	546	0
Contains Repetitive Stems	697	12	550	4
Heuristics Based Filter (Combined)	486	305	257	215
ELECTRA Discriminator + Roundtrip Consistency	489	295	267	212
ELECTRA Based Filter Model	557	393	169	144
Heuristics and ELECTRA Filter Model Combined	484	430	132	217

We also conducted question-answering tests with the ELECTRA-based filter method. The results of this experiment are in Table 4.6. The unfiltered version of the dataset had 58811 questions, while the filtered version of this dataset had 34038 questions, which is 57.9% of the unfiltered version. Training with the filtered data

brought a 2.1% relative improvement in EM performance while bringing a 4.5% F1 score improvement over the unfiltered version of the dataset when we evaluate over the XQuAD. For the ThQUAD dataset, the improvement in EM was 1% and the improvement in F1 was 1.4%. This points out that even though we have fewer questions the increased overall quality of questions is more impactful in the performance of the question-answering model.

Table 4.6. QA results of the BERT model trained with filtered and unfiltered versions of the generated dataset. Both versions of this dataset also omit the questions that are annotated for the filter model training.

	Filtered Generated		**Unfiltered Generated**	
	EM	**F1**	**EM**	**F1**
ThQuAD	42.89	64.53	42.47	63.66
XQuAD	40.67	57.57	39.83	55.09

4.3. Data Augmentation Method For Question Answering Task

In this setting, we wanted to test if the data generation approach is an effective way to augment the question-answering training data in low-resource settings. This approach is tested on 5 different datasets from 4 different languages which are English, German, French, and Turkish. Various data scarcity levels were tested by subsampling the datasets for training the question-generation and question-answering models.

4.3.1. Experimental Setup

In our initial set of trials, we utilized the SQuAD and Spoken SQuAD datasets. The training set of SQuAD was first shuffled at the article level, and then 10% of the articles were designated as a development set to fine-tune the model's hyperparameters. The remaining articles were separated into two mutually exclusive, equally sized subsets referred to as Part-I and Part-II. We employed the paragraphs and question-

answer pairs in Part-I as the annotated data to fine-tune the question-generation and question-answering models in our experimentation. To examine the performance of question-generation and question-answering with limited amounts of annotated data, we subdivided the articles in the Part-I partition into six different-sized subsets, consisting of roughly 5%, 10%, 20%, 30%, 40%, and 50% of the original SQuAD train partition. It should be noted that the larger subsets include the smaller ones, and the largest subset, 50%, corresponds to the complete Part-I data. We regarded Part-II as an unannotated partition and only utilized the paragraphs, not the question-answer pairs, in question-generation to automatically generate question-answer pairs. For the reading comprehension experiments, we trained six question-generation models, one for each subset derived from Part-I. Using these question-generation models, automatically generated question-answer pairs were produced for the paragraphs in Part-II.

In our listening comprehension experiments, we explored two distinct configurations for QG. The first approach entailed employing the automatically generated question-answer pairs generated during the reading comprehension experiments with SQuAD; however, we replaced the original paragraphs with the corresponding ASR transcriptions. Then, we eliminated question-answer pairs where the answer to this question was not present in the transcription. This methodology is referred to as "Eliminated SQuAD" in question-generation experiments. It is important to note that a comparable method was employed when producing the Spoken SQuAD dataset from the SQuAD dataset.

The second approach involved generating Part-I and its subsets, as well as Part-II, from the Spoken SQuAD by substituting the reference paragraphs with their corresponding ASR transcriptions. Part-I and its subsets had fewer instances than those obtained from SQuAD owing to the absence of answers in the ASR transcriptions. Since Part-II only comprised the transcriptions, it had the same amount of data as the reading comprehension setup. We developed six distinct question-generation models, one for each subset derived from Part-I. Then, we generated automatically generated question-answer pairs for the transcriptions in Part-II using these question-generation

models. This technique is known as “ASR SQuAD” in question-generation experiments.

The question-generation system was implemented in Python using the HuggingFace library [48] with the pre-trained mT5-small model. We used a batch size of 8 with gradient accumulation steps of 32 to achieve an effective batch size of 256. Based on our preliminary experiments on our development set, the learning rate was chosen as 1e-3 and all question-generation models were fine-tuned for 10 epochs. We used a pre-trained BERT-based model for question answering systems. We fine-tuned the model with 16 batch size with $3e^{-5}$ learning rate for 20 epochs and select the model with the best F1 score over the development set as our final model.

We also aimed to investigate setups with more limited amounts of data, as even the smallest subset, which comprised approximately 5% of the data in the initial setup, contained nearly 4,000 questions in the SQuAD. Nonetheless, acquiring sufficient data for certain languages might still prove challenging.

In this part of our question generation experiments, we used the SQuAD, FQuAD, GermanQuAD, Spoken SQuAD, English Lectures, and Turkish Lectures datasets. Since SQuAD, FQuAD, GermanQuAD, and Spoken SQuAD datasets have 20.000+ questions, we only used subsets from these datasets to simulate low-resource question-answering settings. We first split the dataset similar to the experiments before. We set apart 10% as a development set and then split the rest into two sets Part-I and Part-II. After that, we shuffled Part-I and then selected 1K, 2K, 3K, 4K, and 5K question-answer pairs, and these subsets were used to train the question-generation models. We make the question selection at the article level so the question-answer counts are not exact but close to the given numbers. The 5K was the whole of Part-I. The bigger subset contained the smaller subset also in these experiments too.

We maintain this approach to ensure that the data remains as consistent as possible across sets, to prevent any performance fluctuations that may be caused by

random subsets that are more or less aligned with the development set. The motivation behind using varying size limited annotated data (1K-5K question-answer pairs) is to explore the effect of increasing data size for question-generation model fine-tuning. Also to keep the number of passages that question-answer generation will be made close between those 4 datasets, we used all of the passages from GermanQuAD which was around 1K passages and we randomly sample articles from Part-II split of other datasets to have around 1K passages in the end. We did it to keep our setups similar as possible between datasets. Even though finding unannotated passages is much simpler than finding or generating annotated data, keeping the passage count was also beneficial to this method's ease of use.

We also make experiments to see how our framework improves the question-answering performance with an increased amount of unannotated paragraphs. Even though we work on low resource settings getting unannotated paragraph data is much cheaper than generating annotated question-answering dataset. Because of that we also test the effects of increasing the number of passages from 1K to 8.5K in SQuAD and Spoken-SQuAD datasets. The Part-II of SQuAD and Spoken-SQuAD datasets had a maximum of 8.5K passages so we select randomly 1K, 2K, 4K, and 8.5K passages. We make this selection similar to the question-answer selection from Part-I which the bigger set also contains the smaller set inside. While fine-tuning the question-generation model with spoken SQuAD, we also explored generating the question-answer pairs from both the reference and ASR transcriptions of the unannotated data.

Since English and Turkish lecture datasets have very limited annotated question-answering data, whole train partitions were used to train the question-generation models, we did not use any sampling method. As the unannotated data for these datasets, we used paragraphs collected from English and Turkish lecture domains. For English, lecture text data were collected from the reference transcriptions of MIT OpenCourseWare Digital Signal Processing course [51]. The transcription document has paragraph information. For Turkish, lecture text data were collected from reference transcriptions of some other law courses' video lectures offered at MEF University. However, these

reference transcriptions do not have either the sentence or the paragraph information. We automatically obtained the sentence boundaries using morphological analysis. In Turkish, regular sentences typically end with verbs, and the language is agglutinative, meaning that verbs often have common suffixes. We used regular expressions to identify these suffixes and split the text into sentences based on these suffixes. We then chunked these sentences to create pseudo paragraphs of around 100 words. These pseudo-paragraphs served as our unannotated Turkish lecture data. Note that the lecture data only contain the reference transcriptions as the unannotated data, so in contrast to spoken SQuAD, question-generation fine-tuning was performed only on these reference transcriptions.

The question generation model was implemented using HuggingFace [48] with the pre-trained mT5-small [8] model containing 60 million parameters. We used batch size 8 with gradient accumulation steps of 32. The learning rate was $1e^{-3}$ for all models and they were fine-tuned for 30 epochs. Since mT5 is multilingual, we used the same base pre-trained model for starting point of training English, French, German, and Turkish question-generation models, but fine-tune the models for each of the languages separately. We also evaluated the performances of different question-generation models on the respective test sets of the datasets using ROUGE and BLEU metrics by comparing the automatically generated questions with the reference questions associated with the same paragraphs.

For evaluating our framework's effects on the question-answering performance, we fine-tuned question-answering models with different transformers models as a backbone. We used, various BERT models pre-trained on target languages. We also used multilingual models like mBERT and XLM-RoBERTa. Models are trained with 16 batch size and a learning rate of $2e^{-5}$ for 30 epochs. At each epoch, their performance is evaluated on the development set of the target dataset and the epoch with the best F1 score is selected.

4.3.2. Results

For 5-50% experiments, it can be seen that the number of question-answer pairs generated by fine-tuned question-generation model in Table 4.7. We generally generate more questions when we train the model with SQuAD data except the 10% setting compared to using ASR paragraphs of the Spoken-SQuAD. The reason would be that the question-generation model trained with SQuAD data had about 2 times the question-answer pairs for training. This would make this model trained better for all of the settings. Also, the noise introduced by ASR paragraphs might be interfering with our answer extraction process which decreases the number of questions that we can generate with this pipeline. Also increasing the number of questions in the ASR SQuAD setting did not bring any increase to the question count so the noise of the paragraphs might be our limiting factor. Eliminated SQuAD setting has the lowest of the question counts since we are eliminating any question-answer pair if an ASR error collides with an answer span.

Table 4.7. The number of question-answer pairs generated from the question-generation models trained on various data subsets.

Data Subsets	**SQuAD**	**ASR SQuAD**	**Eliminated SQuAD**
5%	36,591	24,163	8,537
10%	37,744	39,127	7,704
20%	41,799	39,024	8,192
30%	42,952	39,827	8,458
40%	42,795	39,794	7,989
50%	44,096	39,931	8,189

We evaluated the performance of the reading comprehension models with an F1 score on the official SQuAD dev set. The EM scores of the question-answering models are given in Figure 4.1 while F1 scores are given in Figure 4.2. The X-axis shows the percentage of the annotated data used in question-generation training. First of all,

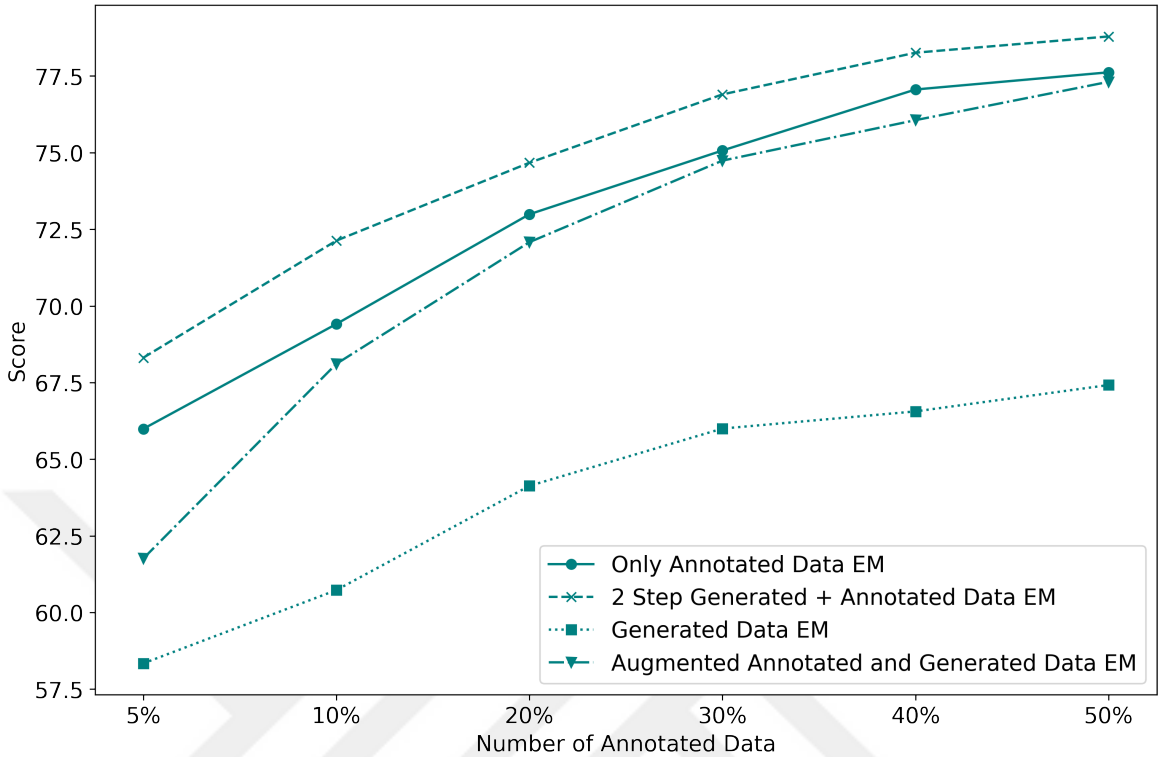

Figure 4.1. The EM scores of different question-answering model settings on SQuAD development set.

we see a performance improvement in all of the settings with the increased number of annotated data. The amount of increase is getting smaller with the increased amount of annotated data but it still is increasing. It is also seen from this Figure 4.2, that the 2-step approach outperforms the question-answering model only trained with human-annotated data. The data gap between the two increases if data is more limited (for example 5%, 10%), while the lead of 2 step model decreases for the higher data settings it is still outperforming the baseline by almost 1%. 10% subset yields the best result, 2.3% F1 improvement (from 79.47 to 81.77) in reading comprehension. Note that fine-tuning the question-answering model with the whole SQuAD train data (100% used in fine-tuning) yields 88.51 (F1 score). These scores are 76.36 and 86.16 when 5% and 50% subsets were used in fine-tuning.

F1 scores of the question-answering models trained only with the automatically generated data are also given in Fig. 4.2. This setting gave the lowest scores in all

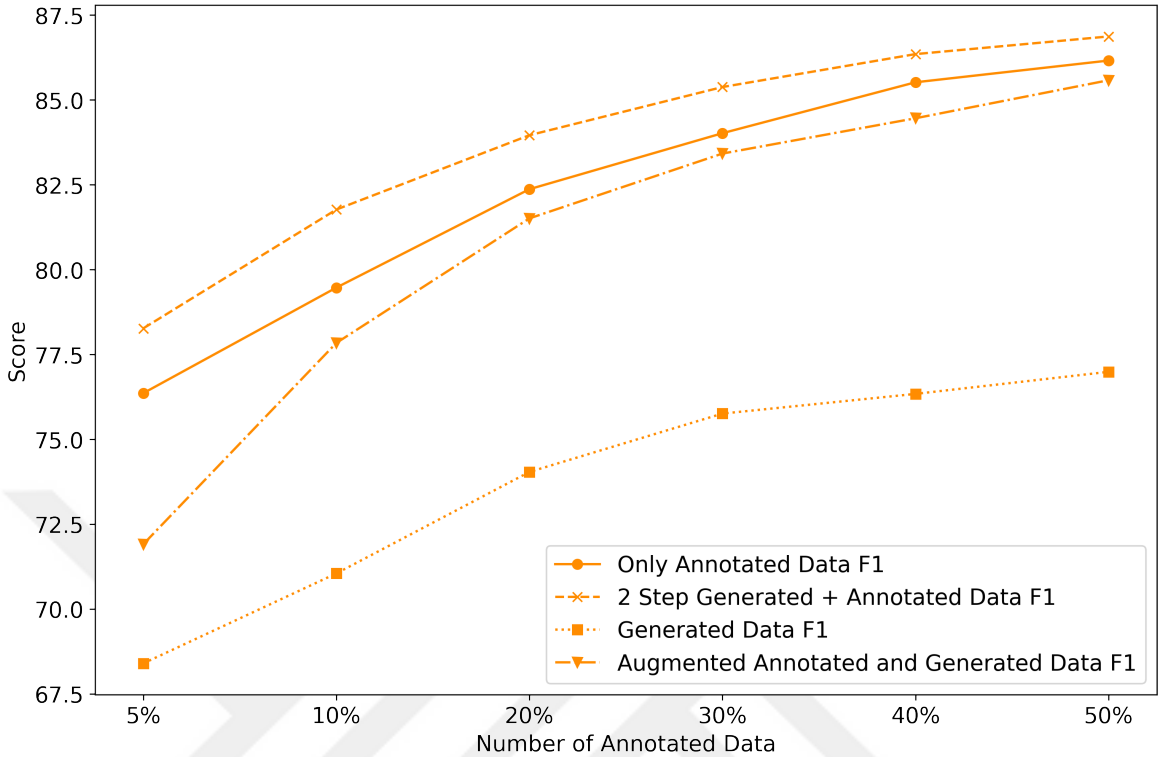

Figure 4.2. The F1 scores of different question-answering model settings on SQuAD development set.

of the settings, which points out that even a small number of human-annotated data is needed to perform well in this task. Comparing the augmented data and 2-step fine-tuning results indicates that the 2-step fine-tuning approach is more effective than the augmentation approach in our framework. The augmented data approach does not outperform the baseline models, whereas the 2-step fine-tuning approach significantly outperforms the baseline models.

Figure 4.3 presents the EM and Figure 4.4 presents the F1 scores of the question-answering models in the listening comprehension experiments. The subsets of the Spoken SQuAD train set were used to train the question-generation models, and their performance was evaluated on the development set of Spoken-SQuAD. The results show similar trends to those observed in the reading comprehension experiments. The models trained only with automatically generated data have the lowest performance, presumably due to noise in the question-answer pairs. The 2-step fine-tuning approach

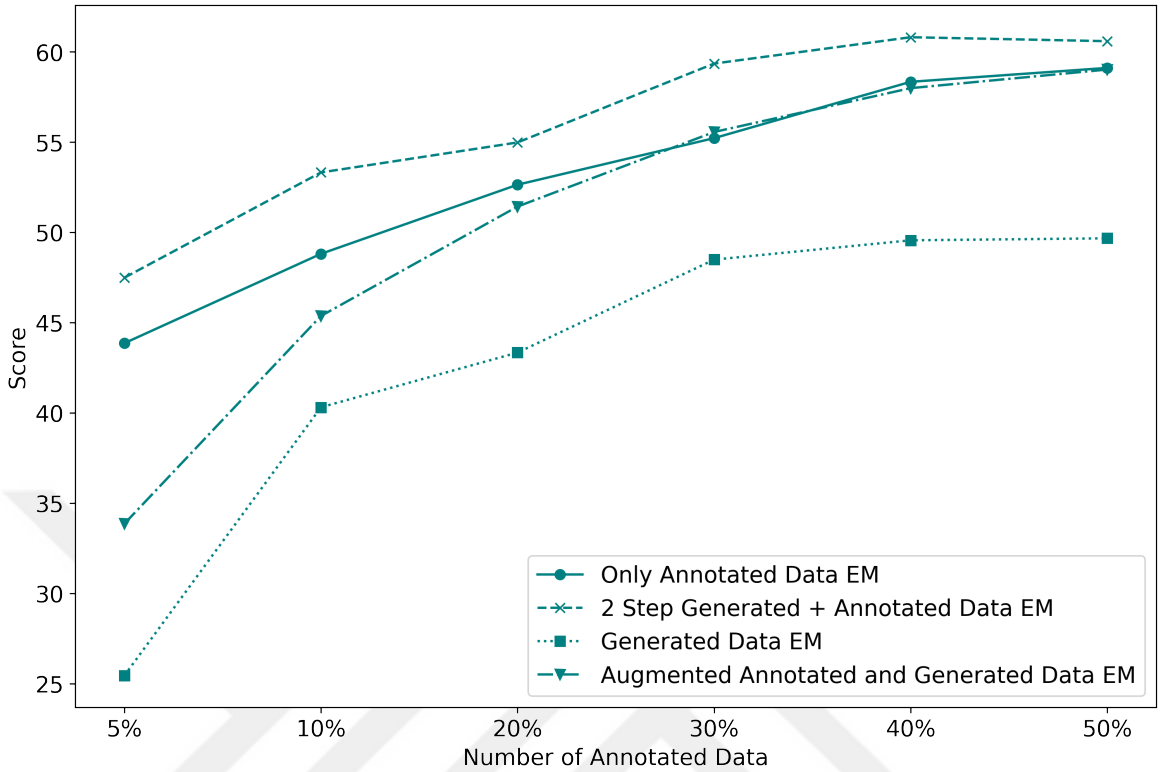

Figure 4.3. The EM scores of different question-answering model settings on Spoken SQuAD development set.

outperforms both the augmented data approach and the baseline models, highlighting the importance of a carefully chosen fine-tuning strategy. The performance improvements are more significant in limited data scenarios (i.e., 5%, 10%), with the 10% subset showing the highest gain in the listening comprehension experiments. However, with larger data, the gap between the question-answering models trained with only annotated data and the question-answering models trained with both generated and annotated data gets smaller. It is worth noting that fine-tuning the question-answering model with the entire Spoken-SQuAD train data (100%) yields an F1 score of 74.2. The scores are 55.91 and 70.16 when 5% and 50% annotated data subsets are used in fine-tuning, respectively.

For the lower resource settings (1K-5K) we first evaluated the question-generation performance using the automatic evaluation metrics, BLEU and ROUGE, before training the question-answering models and evaluating them with the question-answering

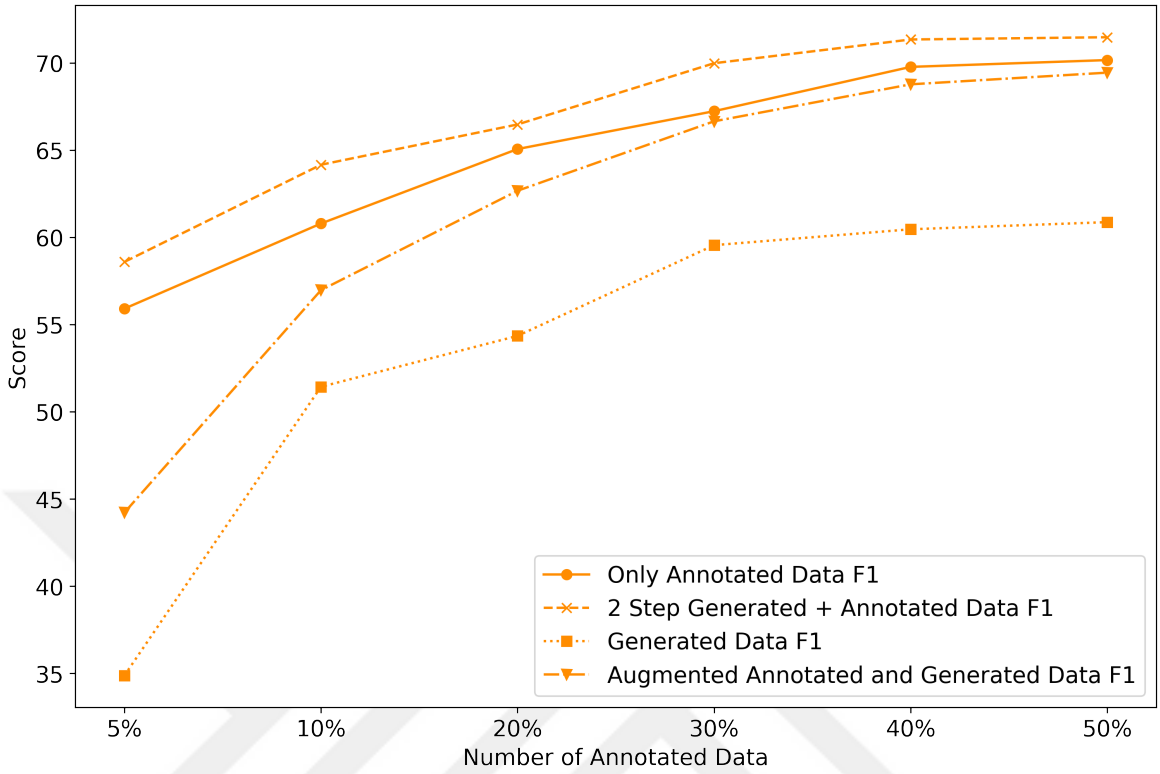

Figure 4.4. The F1 scores of different question-answering model settings on Spoken SQuAD development set.

models. The results obtained on the test sets are presented in Table 4.8 for question-generation models that were trained on varying sizes of annotated data (ranging from 1K to 5K) using SQuAD, FQuAD, and GermanQuAD datasets. The results demonstrate that an increase in the amount of data used in question-generation fine-tuning results in improved performance, particularly when the amount of data is increased from 1K to 2K, and the increase in performance is observed to slow down when the amount of data is further increased.

For the question-answering experiments, we first assessed the effectiveness of the automatically generated data on the reading comprehension datasets in limited annotated data and multilingual settings.

Figure 4.5, Figure 4.6, and Figure 4.7 show the question-answering performances (EM and F1 scores) on SQuAD, FQuAD, and GermanQuAD test sets, respectively.

The question-answering models were fine-tuned using both annotated and unannotated data, and their performances were compared against the baseline models, which were only fine-tuned with annotated data. The baseline scores are shown with solid lines while models trained with both data sources are shown with dashed lines in the figures. For each dataset, the annotated data size used for question-answering and question-generation model training is varied from 1K to 5K question-answer pairs and the unannotated data size is approximately 1K paragraphs. On average, the question-generation models generate 4950, 4850, and 5850 question-answer pairs for SQuAD, FQuAD, and GermanQuAD, respectively over those paragraphs.

Table 4.8. Comparison of question-generation models trained on varying size annotated data (1K - 5K) for English, French, and German reading comprehension datasets.

	Train subsets	**ROUGE**	**BLEU 1**	**BLEU 2**
SQuAD	1K	0.335	0.308	0.142
	2K	0.363	0.331	0.156
	3K	0.377	0.346	0.170
	4K	0.384	0.355	0.174
	5K	0.388	0.360	0.177
FQuAD	1K	0.240	0.219	0.087
	2K	0.311	0.279	0.134
	3K	0.335	0.304	0.150
	4K	0.346	0.313	0.157
	5K	0.345	0.311	0.160
GermanQuAD	1K	0.209	0.188	0.072
	2K	0.260	0.235	0.093
	3K	0.281	0.256	0.103
	4K	0.286	0.260	0.109
	5K	0.288	0.262	0.110

We did not train question-answering models in 1K-5K settings with the augmented training approach. Because Figure 4.3 and Figure 4.4 show that the augmented approach is performing lower than the question-answering models only trained with human-annotated data in the 5-50% experiments. We also saw that the performance difference was much higher at the lower resource settings like 5%. Since the 1K-5K settings have fewer data, while we training the combined data we only used the 2-step approach.

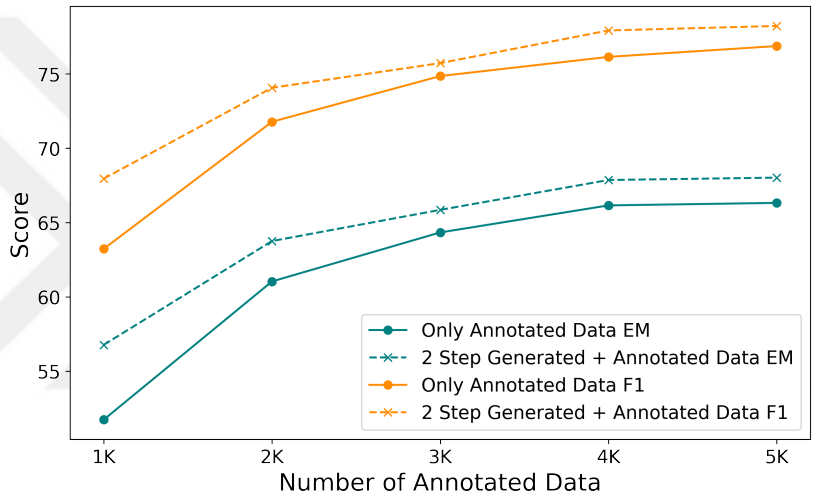

Figure 4.5. The EM and F1 scores of question-answering models that augmented with generated data in SQuAD development set.

In the case of SQuAD (Figure 4.5), we observed that increasing the amount of annotated data resulted in improvements in both EM and F1 scores for the baseline models, as evidenced by the increasing trend in solid lines. The best baseline result was achieved with the most annotated data, the 5K setting. The upper bound of EM/F1 scores is 81.31/88.51 when the entire annotated training data (comprising over 80K question-answer pairs) was used for fine-tuning the question-generation model.

The models fine-tuned with both annotated and generated data outperformed the baseline models for all training subsets. However, when the amount of annotated data is more constrained, such as in the 1K, 2K, and 3K scenarios, the performance improvements are more subtle. On average, the improvement in EM is 2.5%, and in F1

is 2.2%. The most significant gains were observed with the 1K subset, which showed a 5% improvement in EM (from 51.72 to 56.76) and 4.6% in F1 (from 63.22 to 67.95).

We also train mBERT, XLM-RoBERTa, and ELECTRA-based question answering models for the 1K-5K SQuAD settings. The results of those experiments can be seen in Appendix D. On average over all models, the best-performing set was 1K and it showed on average 2.6% EM improvement and 3% F1 improvement. The lowest improvements happened in the 5K setting, which had 0.6% and 0.6% improvement in EM and F1 scores respectively.

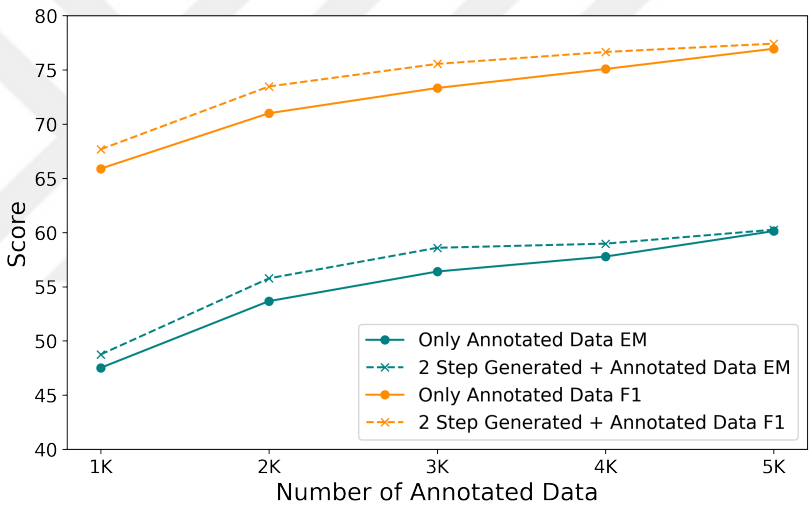

Figure 4.6. The EM and F1 scores of question-answering models that augmented with generated data in FQuAD development set.

Regarding FQuAD (as shown in Figure 4.6), we observed a similar trend to the SQuAD experiments, for both the baseline models and the models fine-tuned with the annotated and generated data. On average, the EM score improved by 1.3%, while the F1 score improved by 1.7%. The most substantial improvements were noticed in the 2K subset, with a 2% improvement in EM (from 53.67 to 55.77) and a 2.5% improvement in F1 (from 71.00 to 73.47). As shown in Figure 4.6, the difference between the solid and dashed lines diminishes with the increasing annotated data size. In the 5K setting, this results in EM scores of 60.13 and 60.26 and F1 scores of 76.95 and 77.42, respectively. The maximum possible EM/F1 scores are 64.02/81.03 when

the entire annotated training dataset (comprising over 20K question-answer pairs) was used for fine-tuning the question-generation model.

We also train mBERT and XLM-RoBERTa-based question-answering models for the 1K-5K FQuAD settings. The results of those experiments can be seen in Appendix D. On average over all models, the best-performing set was 1K, showing an average 4.4% EM improvement and 5.7% F1 improvement. The lowest improvements happened in the 5K setting, which had 0.8% and 0.6% improvement in EM and F1 scores respectively.

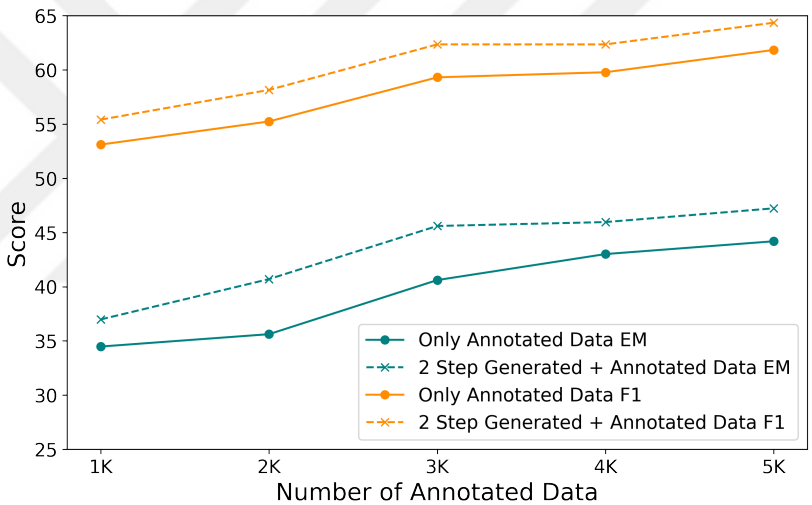

Figure 4.7. The EM and F1 scores of question-answering models that augmented with generated data in GermanQuAD development set.

Regarding GermanQuAD (Figure 4.7), we have also noted improvements both with the baseline models and the models fine-tuned with the annotated and the generated data when the annotated data size was increased. The 2-step process training process was also more efficient in the lower end of the data size, but the improvements in the 5K settings were larger than the FQuAD and SQuAD settings. The average improvement in EM was 3.7%, and in F1, it was 2.7%. The highest gains were observed with the 2K subset, with a 5% improvement in EM (from 35.62 to 40.72) and 3% in F1 (from 55.24 to 58.15). The improvement in the 5K set was 3% and 2.5% for the EM and F1 scores respectively. When the whole annotated training data (more than

11K question-answer pairs) were used for fine-tuning the question-generation model, the upper bound of EM/F1 scores was 51.18/69.52.

We also train mBERT and XLM-RoBERTa-based question-answering models for the 1K-5K GermanQuAD settings. The results of those experiments can be seen in Appendix D. On average over all models, the best-performing set was 2K, and the 2K set experiments showed on average 5% EM improvement and 3.9% F1 improvement. The lowest improvements happened in the 4K setting, which had 1.1% and 1.2% improvement in EM and F1 scores respectively.

Table 4.9 presents the reading comprehension results for both English and Turkish lectures. Because the annotated question-answering data is quite limited in both the English and Turkish lecture datasets, the whole train partitions were used to fine-tune the question-generation models. The first column of the table indicates the task, while the second column identifies the dataset used.

Similar to the previous findings, the models fine-tuned with annotated and generated data outperform the baseline models. In the case of English lectures, the improvements were substantial (from 41.15 to 45.19 in EM and 68.16 to 73.41 in F1), whereas, in Turkish lectures, the gains are more subtle (from 54.90 to 54.89 in EM and from 80.86 to 82.70 in F1). It is worth noting that the gap between the EM and F1 scores is larger in the lectures datasets than in the SQuAD dataset, likely due to the longer answer lengths in the lectures datasets (which average 19 words versus 3 words in the SQuAD dataset).

We also train mBERT and XLM-RoBERTa-based question-answering models for the lecture datasets. The results of those experiments can be seen in Appendix E for Turkish and in Appendix F for English. In the English Lectures dataset, on average 2 step training improved the EM score by 2% and F1 scores by 1.6%. For the Turkish Lectures dataset, EM scores increased by 2.6% and F1 scores increased by 4%.

Secondly, we examined the impact of increasing the number of unannotated passages used for automatically generating question-answer pairs. To perform this experiment, we used the SQuAD dataset with a 1K annotated subset, as this subset produced the greatest improvement when fine-tuning the question-answering model with both annotated and generated data. The number of paragraphs in the question-generation model was increased from around 1K to 8.7K. Table 4.10 displays the EM and F1 scores of these experiments over SQuAD and Spoken SQuAD. For SQuAD, varying sizes of annotated data are used to fine-tune question-answering models. For spoken SQuAD, separate question-generation models using ASR and reference transcriptions were fine-tuned to generate questions from reference transcriptions. The baseline scores were the lowest (EM/F1 as 51.72/63.22) when no generated data was used in question-answering training, as shown in Figure 4.5. The results increased to 59.80/71.36 as EM/F1 when more passages were used in question generation, indicating the importance of the unannotated data size for the question-generation model. It's worth noting that collecting unannotated data is simpler than collecting annotated data, so increasing the unannotated data size can be much faster than annotating a question-answering dataset. The average number of generated question-answer pairs from the unannotated sets is 16755, with a range of 4496 to 35439 using around 1K and 8.7K paragraphs, respectively.

Table 4.9. Performances of question-answering and SQA models fine-tuned using only annotated data and using both annotated and generated data on English and Turkish lecture datasets.

Task	**Datasets**	**Annotated data**		**Annotated + Generated Data**	
		EM	**F1**	**EM**	**F1**
QA	English Lectures	41.15	68.16	45.19	73.42
	Turkish Lectures	54.90	80.86	54.89	82.70
SQA	English Lectures	41.72	68.51	41.72	70.48
	Turkish Lectures	48.95	78.36	49.30	79.66

We conducted SQA experiments on listening comprehension datasets (Spoken SQuAD and English and Turkish lecture datasets). The spoken SQuAD experiments used the same setup as the SQuAD experiments, and ASR transcriptions were used for both annotated and unannotated data. Figure 4.8 displays the EM and F1 scores of the question-answering models fine-tuned with spoken SQuAD subsets. Similar to the reading comprehension experiments, models fine-tuned on both annotated and generated data outperformed baseline models. The average improvement in EM and F1 was 3.4% and 3.5%, respectively. The highest gains were seen with the 1K subset, with a 6% improvement in EM (from 37.11 to 43.15) and a 6.8% improvement in F1 (from 48.82 to 55.62). When the entire annotated training data (consisting of over 37K question-answer pairs) was used for question-answering model fine-tuning, the upper bound of EM/F1 scores was 64.03/74.23.

Table 4.10. The performance of the question-answering and SQA models on spoken SQuAD and SQuAD.

Datasets	**Annotated data**	**Unannoated data size**							
	Question-Generation Train Set	**1100**		**2200**		**4400**		**8675**	
		EM	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
SQuAD	-	56.76	67.95	58.99	70.41	59.87	71.41	59.80	71.36
Spoken SQuAD	ASR Transcription	41.21	53.68	43.58	55.65	45.35	57.24	46.14	57.82
	Reference	44.18	55.97	46.20	58.07	47.22	58.88	47.73	59.95

We also train mBERT, XLM-RoBERTa, and ELECTRA-based question answering models for the 1K-5K Spoken SQuAD settings. The results of those experiments can be seen in Appendix D. On average over all models, the best-performing set was 1K and it showed on average 3.7% EM improvement and 4.3% F1 improvement. The lowest improvements happened in the 4K setting, which had 0.8% and 0.7% improvement in EM and F1 scores respectively.

We investigated the impact of using reference transcriptions as both annotated and unannotated data on spoken SQuAD. The results of question-answering models fine-tuned with data generated from ASR and reference transcriptions are shown

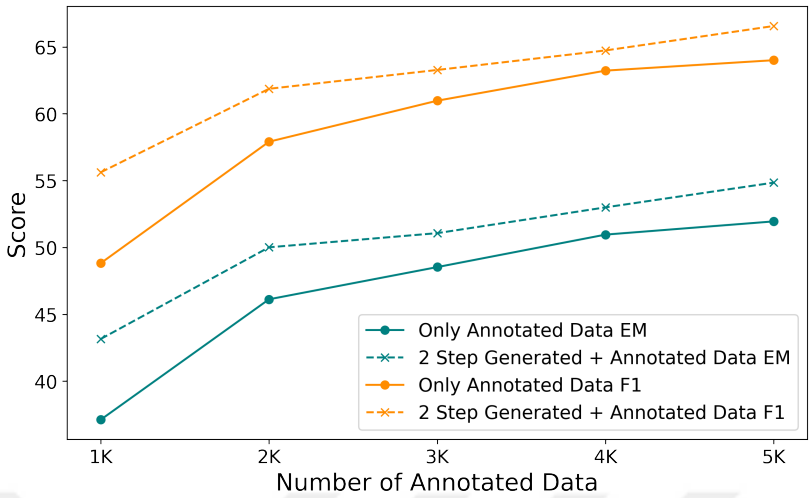

Figure 4.8. The EM and F1 scores of question-answering models that augmented with generated data in Spoken SQuAD development set.

in Table 4.10 under the "Spoken SQuAD" section. The question-generation models were fine-tuned on both ASR and reference transcriptions and questions were generated from various unannotated sets ranging in size from 1K to 8.7K, taken from reference transcriptions. The models surpassed the baseline, which was 37.11/48.82 as EM/F1 (as shown in Figure 4.8). Using reference transcriptions as both annotated and unannotated data led to better results than using ASR transcriptions. The average improvement in EM was 9.2% and in F1 was 9.4% with reference transcriptions, while it was 7% in EM and 7.3% in F1 with ASR transcriptions. The quality of the questions generated from the reference texts was higher, leading to an overall enhancement in the framework when using reference transcriptions.

We also train mBERT and XLM-RoBERTa-based question-answering models for the lecture datasets. The results of those experiments can be seen in Appendix E for Turkish and in Appendix F for English. In the English Lectures dataset, on average 2 step training improved the EM score by 1.2% and F1 scores by 2.5%. For the Turkish Lectures dataset, EM scores increased by 4.1% and F1 scores increased by 1.8%.

Table 4.9 displays the results of the listening comprehension experiments on both English and Turkish lectures under the "SQA" section. The annotated data in these

experiments was obtained from ASR transcriptions, while the unannotated data was sourced from reference transcriptions. It should be noted that the performance of the SQA models is generally lower compared to the question-answering models due to errors introduced by the ASR system. However, the integration of generated data into the fine-tuning process leads to an improvement in SQA performance. In the English lectures dataset, the EM score remained relatively unchanged at 41.72, but the F1 score increased by 2% (from 68.515 to 70.485). On the other hand, in the Turkish lectures dataset, the improvement in the EM score was 0.4% (from 48.95 to 49.30), while the F1 score increased by 1.3% (from 78.366 to 79.66).

4.4. Domain Adaptation

This section contains experiments to train a question-generation model in the larger dataset in another domain and then use this model to generate question-answer pairs on another more resource-constrained domain. We believe that the question-generation model can harvest the knowledge from other domains and using this knowledge, we can generate training data in the domain of interest which has low or no annotated data. The generated question-answer pairs can be used to train question-answering models in the domain of interest. This would greatly shorten the question-answering model training for the new domain since it will not require manually annotated data in the target domain.

4.4.1. Experimental Setup

The question-generation module was implemented in Python using the HuggingFace library [48]. We used the small pre-trained mT5 model with a batch size of 8 and 32 gradient accumulation steps to achieve an effective batch size of 256. The model was fine-tuned for 30 epochs with a learning rate of $1e^{-4}$ using SQuAD and ThQuAD for English and Turkish respectively. The SQuAD dataset is a general domain dataset that has Wikipedia passages but does not have any topic constraints. On the other hand, ThQuAD contains questions only from Turkish Islamic Science and Ottoman History

topics. This difference will also show us that a more general question-answering dataset helps with the domain adaptation problem.

Afterwards the trained question-generation models are used to generate question-answer pairs over the English Lectures and Turkish Lectures dataset passages. English Lectures and Turkish Lectures datasets are vastly different in terms of the topic from the bigger SQuAD and ThQuAD datasets which question-generation models trained. We then trained question-answering models using this generated data. For baselines, we first used question-answering models trained by SQuAD and ThQuAD datasets directly. Another baseline was using the handcrafted small amount of data in the domain.

We trained BERT-based question-answering models for the evaluation of the performance of the different datasets. The question-answering model was implemented in Python with Hugging Face [48]. English BERT-large [16] and Turkish BERT-base [52] pre-trained models are used for English and Turkish lecture datasets respectively. The batch size is fixed at 8 for all the models. They were fine-tuned for a maximum of 20 epochs with a learning rate of 3e-5. At the end of each epoch, the model is evaluated over the development set, and the model with the best F1 score on the development set is selected as the best model for testing.

4.4.2. Results

4.4.2.1. SQuAD to English Lectures. In this experiment, we saw that using only synthetic questions generated by the SQuAD question-generation model for question-answering training yields the worst results. Even though the English Lectures dataset has a much lower number of questions in the training set. This might be due to the differences in the dataset distribution. Because while the SQuAD dataset has short answers, the English Lectures dataset had long answers which would create a difference between the human-annotated and synthetic question-answer pairs.

We check our question-generation models' question generation performance using the ROUGE and BLEU scores. We saw that when we train the question-generation model with SQuAD data it had a ROUGE-L score of 0.304 while having 0.255 and 0.132 BLEU-1 and BLEU-2 scores. On the other hand question-generation model trained with the English Lectures dataset only had a ROUGE-L score of 0.230, while having 0.223 and 0.107 BLEU-1 and BLEU-2 scores. It shows that the question-generation model-trained SQuAD data was better to mimic the human-annotated questions in the English Lectures dataset.

Table 4.11. QA performance over the English Lectures dataset with different augmentation methods.

	EM	**F1**
QA trained on SQuAD	20.58	57.17
SQuAD question-generation Synthetic Questions	1.14	32.75
English Lectures question-generation Synthetic Questions	6.88	37.40
English Lectures Data	41.15	68.17
English Lectures question-generation + English Lectures Data	45.19	73.42
SQuAD question-generation + English Lectures Data	52.30	75.82

When we train our model in a 2-step fashion, first with synthetic questions, and then with the original dataset however gave different results than the synthetic question-only approach. In this approach using question-answer pairs generated by SQuAD question-generation improved scores further 7% in EM and 2.4% in the F1 scores, then using the English Lecture data as question-generation model. This points out that the increased number of questions even though in different domains is beneficial, we still need small hand-crafted data to fine-tune the model to the target domain, especially if we have a mismatch in data distribution.

4.4.2.2. ThQuAD to Turkish Lectures. The ThQuAD to Turkish Lectures experiment show us similar patterns to the English experiment. We got the worst scores when we

generate questions using the question-generation model trained with ThQuAD data. Its F1 score was 21% lower than its counterpart in which we use questions from the question-generation model trained with the Turkish Lectures dataset for question-answering training. The question-answering model directly trained on ThQuAD is also around 13.2% worse in F1 performance with respect to the question-answering model trained with questions from the question-generation model trained with the Turkish Lectures dataset. This worse performance than English experiments might be related to the limited domain of the ThQuAD dataset, it was only containing questions about Islamic Science and Ottoman History, while the SQuAD dataset had articles from various topics, which would improve the generalization performance of the model.

Table 4.12. QA performance over the Turkish Lectures dataset with different augmentation methods.

Setting	**EM**	**F1**
QA trained on ThQuAD	3.15	51.91
ThQuAD question-generation Synthetic Questions	0.70	44.30
Turkish Lectures question-generation Synthetic Questions	37.41	65.12
Turkish Lectures Data	54.90	80.86
Turkish Lectures question-generation + Turkish Lectures Data	54.89	82.70
ThQuAD question-generation + Turkish Lectures Data	57.34	83.00

We check our question-generation models' question-generation performance using the ROUGE and BLEU scores. We saw that when we train the question-generation model with ThQuAD data it had a ROUGE-L score of 0.233 while having 0.254 and 0.075 BLEU-1 and BLEU-2 scores. On the other hand question-generation model trained with the Turkish Lectures dataset only had a ROUGE-L score of 0.226, while having 0.248 and 0.073 BLEU-1 and BLEU-2 scores. It shows that the question-generation model trained with ThQuAD data was slightly better to mimic the human-annotated questions in the Turkish Lectures dataset.

In the 2-step model training settings, the model trained with questions generated by the ThQuAD question-generation model outperforms the setting in which the synthetic questions come from the Turkish Lectures question-generation model. The model performed 2.5% better in EM score, and 0.3% in F1 scores, which is less significant than the English experiments but especially EM score difference was still significant. This shows us that the more questions used in the question-generation model training benefit the overall 2-step performance.

5. CONCLUSION

In this thesis, we propose a framework to generate question-answer pairs using a question-generation model for improving the question-answering performance for low-resource languages. The proposed question generation model uses mT5 which has multilingual capabilities. Because of that, the same pre-trained model can be utilized for any language. We then use this framework to generate a Turkish Question Answering dataset from Turkish Wikipedia passages. We also experiment with different filtering techniques to filter out the generated questions for improving dataset quality. For testing the framework's low resource capabilities we extensively test our approach's validity as a data augmentation method in low resource settings in English, German, French, and Turkish datasets, with varying amounts of data subsets from the datasets.

The proposed generated Turkish Question Answering dataset has consist of 83.6K questions generated by this framework over 20K Wikipedia passages. Our experiments revealed that even though for most of the settings it was not able to surpass the human-annotated corpus but it was a beneficial data augmentation tool, improving performance on XQuAD Turkish dataset by 3%.

Our low-resource experiments in English, German, French, and Turkish datasets revealed that our framework was working efficiently even at around or less than 1000 questions. They were more impactful in the lower data settings. For the 1000 question settings our model improved the F1 score by 4.5% in SQuAD, 7% in Spoken SQuAD, 2.5% in GermanQuAD, and 1.8% in FQuAD which shows that this approach is suited to low resource settings over multiple languages.

For the future work, we can expand the Turkish Generated question answering dataset. In the generation of this dataset, we only used the first paragraphs. The main idea behind that was that the first paragraphs were generally a summary of the article and it had more objective information on which model would be more successful in the

generation process. We saw from the low resource experiments after that, the model was capable to generate questions from all paragraphs of the articles, so we can expand the Turkish Generated dataset to the other paragraphs.

In our experiments even though we use multiple languages, we did not use those languages together. We fine-tune the model for each language separately. In the future, we can exploit the model's multilingual capability by training the model with data from multiple languages. We can also experiment with adapting the question-generation model from a resourceful language to low resource language to distill some of the knowledge from the language with more data.

REFERENCES

1. Rajpurkar, P., J. Zhang, K. Lopyrev and P. Liang, "SQuAD: 100,000+ Questions for Machine Comprehension of Text", *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 2383–2392, Austin, Texas, Nov. 2016.

2. Rajpurkar, P., R. Jia and P. Liang, "Know What You Don't Know: Unanswerable Questions for SQuAD", *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 784–789, Melbourne, Australia, Jul. 2018.

3. Kwiatkowski, T., J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, J. Devlin, K. Lee, K. Toutanova, L. Jones, M. Kelcey, M.-W. Chang, A. M. Dai, J. Uszkoreit, Q. Le and S. Petrov, "Natural Questions: A Benchmark for Question Answering Research", *Transactions of the Association for Computational Linguistics*, Vol. 7, pp. 452–466, 2019.

4. Clark, J. H., E. Choi, M. Collins, D. Garrette, T. Kwiatkowski, V. Nikolaev and J. Palomaki, "TyDi QA: A Benchmark for Information-Seeking Question Answering in Typologically Diverse Languages", *Transactions of the Association for Computational Linguistics*, Vol. 8, pp. 454–470, 2020.

5. d'Hoffschmidt, M., W. Belblidia, Q. Heinrich, T. Brendlé and M. Vidal, "FQuAD: French Question Answering Dataset", *Findings of the Association for Computational Linguistics: Empirical Methods in Natural Language Processing*, pp. 1193–1208, Online, Nov. 2020.

6. Möller, T., J. Risch and M. Pietsch, "GermanQuAD and GermanDPR: Improving Non-English Question Answering and Passage Retrieval", *Proceedings of the 3rd Workshop on Machine Reading for Question Answering*, pp. 42–50, Punta Cana, Dominican Republic, Nov. 2021.

7. Soygazi, F., O. Çiftçi, U. Kök and S. Cengiz, "THQuAD: Turkish Historic Question Answering Dataset for Reading Comprehension", *2021 6th International Conference on Computer Science and Engineering (UBMK)*, pp. 215–220, Ankara, Turkey, 2021.

8. Xue, L., N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua and C. Raffel, "mT5: A Massively Multilingual Pre-trained Text-to-Text Transformer", *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 483–498, Online, Jun. 2021.

9. Clark, K., M.-T. Luong, Q. V. Le and C. D. Manning, "ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators", *International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

10. Budur, E., O. Khattab, Ö. Rıza, D. Soylu, T. Güngör and C. Potts, "Turkish Question Answering Dataset (SQuAD-TR).", TABILAB DIP Project, a Repository for NLP Projects, Tools and Corpora, Bogazici University, 2022.

11. Du, X., J. Shao and C. Cardie, "Learning to Ask: Neural Question Generation for Reading Comprehension", *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1342–1352, Vancouver, Canada, Jul. 2017.

12. Du, X. and C. Cardie, "Harvesting Paragraph-level Question-Answer Pairs from Wikipedia", *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1907–1917, Melbourne, Australia, Jul. 2018.

13. Sun, X., J. Liu, Y. Lyu, W. He, Y. Ma and S. Wang, "Answer-focused and Position-aware Neural Question Generation", *Proceedings of the The Conference on Empirical Methods in Natural Language Processing*, pp. 3930–3939, Brussels, Belgium, 2018.

14. Sachan, M. and E. Xing, "Self-Training for Jointly Learning to Ask and Answer Questions", *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pp. 629–640, New Orleans, Louisiana, Jun. 2018.

15. Vaswani, A., N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser and I. Polosukhin, "Attention is All you Need", I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan and R. Garnett (Editors), *Advances in Neural Information Processing Systems*, Vol. 30, Curran Associates, Inc., Red Hook, NY, USA, 2017.

16. Devlin, J., M.-W. Chang, K. Lee and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", *Proceedings of the Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pp. 4171–4186, Minneapolis, Minnesota, Jun. 2019.

17. Radford, A. and K. Narasimhan, "Improving Language Understanding by Generative Pre-Training", `https://openai.com/research/language-unsupervised`, 2018, accessed on March 15, 2022.

18. Radford, A., J. Wu, R. Child, D. Luan, D. Amodei and I. Sutskever, "Language Models are Unsupervised Multitask Learners", `https://github.com/openai/gpt-2`, 2019, accessed on March 15, 2022.

19. Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li and P. J. Liu, "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer", *Journal of Machine Learning Research*, Vol. 21, No. 140, pp. 1–67, 2020.

20. Lopez, L. E., D. K. Cruz, J. C. B. Cruz and C. Cheng, "Transformer-based End-to-end Question Generation", *arXiv preprint arXiv:2005.01107*, Vol. 4, 2020.

21. Liu, B., H. Wei, D. Niu, H. Chen and Y. He, "Asking Questions the Human Way:

Scalable Question-Answer Generation from Text Corpus", *Proceedings of The Web Conference 2020*, p. 2032–2043, New York, NY, USA, 2020.

22. Dong, L., N. Yang, W. Wang, F. Wei, X. Liu, Y. Wang, J. Gao, M. Zhou and H.-W. Hon, *Unified Language Model Pre-training for Natural Language Understanding and Generation*, Vol. 32, Curran Associates, Inc., Red Hook, NY, USA, 2019.

23. Yin, X., L. Zhou, K. Small and J. May, "Summary-Oriented Question Generation for Informational Queries", *Proceedings of the 1st Workshop on Document-grounded Dialogue and Conversational Question Answering (DialDoc 2021)*, pp. 81–97, Online, Aug. 2021.

24. Shakeri, S., C. Nogueira dos Santos, H. Zhu, P. Ng, F. Nan, Z. Wang, R. Nallapati and B. Xiang, "End-to-End Synthetic Data Generation for Domain Adaptation of Question Answering Systems", *Proceedings of the Empirical Methods in Natural Language Processing*, pp. 5445–5460, Online, Nov. 2020.

25. Alberti, C., D. Andor, E. Pitler, J. Devlin and M. Collins, "Synthetic QA Corpora Generation with Roundtrip Consistency", *Proceedings of the Association for Computational Linguistics*, pp. 6168–6173, Florence, Italy, Jul. 2019.

26. Puri, R., R. Spring, M. Shoeybi, M. Patwary and B. Catanzaro, "Training Question Answering Models From Synthetic Data", *Proceedings of the Empirical Methods in Natural Language Processing*, pp. 5811–5826, Online, Nov. 2020.

27. Luo, H., S.-W. Li, M. Gao, S. Yu and J. Glass, "Cooperative Self-training of Machine Reading Comprehension", *Proceedings of the North American Chapter of the Association for Computational Linguistics*, pp. 244–257, Seattle, United States, Jul. 2022.

28. Bartolo, M., T. Thrush, R. Jia, S. Riedel, P. Stenetorp and D. Kiela, "Improving Question Answering Model Robustness with Synthetic Adversarial Data Genera-

tion", *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 8830–8848, Online and Punta Cana, Dominican Republic, Nov. 2021.

29. Lewis, M., Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov and L. Zettlemoyer, "BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension", *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, Jul. 2020.

30. Lan, Z., M. Chen, S. Goodman, K. Gimpel, P. Sharma and R. Soricut, "Albert: A Lite BERT for Self-supervised Learning of Language Representations", *Proceedings of the International Conference on Learning Representations*, Addis Ababa, Ethiopia, 2020.

31. Liu, Y., M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer and V. Stoyanov, "RoBERTa: A Robustly Optimized BERT Pretraining Approach", *arXiv 1907.11692*, 2019.

32. Brown, T., B. Mann, N. Ryder, M. Subbiah, J. D. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language Models are Few-shot Learners", *Proceedings of the NeurIPS*, Vol. 33, pp. 1877–1901, Online, 2020.

33. Wei, J., M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai and Q. V. Le, "Finetuned Language Models are Zero-Shot Learners", *arXiv preprint arXiv:2109.01652*, 2021.

34. Tseng, B.-H., S.-S. Shen, H.-Y. Lee and L.-S. Lee, "Towards Machine Comprehension of Spoken Content: Initial TOEFL Listening Comprehension Test by Machine", *Proceedings of the Interspeech*, Vol. 27, pp. 2731–2735, San Francisco, CA, USA, 2016.

35. Ünlü, M., E. Arisoy and M. Saraçlar, "Question Answering for Spoken Lecture Processing", *International Conference on Acoustics, Speech and Signal Processing*, pp. 7365–7369, Brighton, United Kingdom, 2019.

36. Lee, C.-H., H.-y. Lee, S.-L. Wu, C.-L. Liu, W. Fang, J.-Y. Hsu and B.-H. Tseng, "Machine Comprehension of Spoken Content: TOEFL Listening Test and Spoken SQuAD", *Transactions on Audio, Speech, and Language Processing*, Vol. 27, No. 9, pp. 1469–1480, 2019.

37. Ünlü, M. and E. Arisoy, "Uncertainty-Aware Representations for Spoken Question Answering", *Spoken Language Technology Workshop (SLT)*, pp. 943–949, Shenzhen, China, 2021.

38. Li, C.-H., S.-L. Wu, C.-L. Liu and H.-y. Lee, "Spoken SQuAD: A Study of Mitigating the Impact of Speech Recognition Errors on Listening Comprehension", *Proceedings of Interspeech*, pp. 3459–3463, Hydebarad, India, 2018.

39. You, C., N. Chen and Y. Zou, "Knowledge Distillation for Improved Accuracy in Spoken Question Answering", *International Conference on Acoustics, Speech and Signal Processing*, pp. 7793–7797, Toronto, Ontario, Canada, 2021.

40. Chuang, Y.-S., C.-L. Liu, H.-Y. Lee and L.-s. Lee, "SpeechBERT: An Audio-and-Text Jointly Learned Language Model for End-to-end Spoken Question Answering", *Interspeech*, Graz, Austria, 2019.

41. Lin, G.-T., Y.-S. Chuang, H.-L. Chung, S.-w. Yang, H.-J. Chen, S. Dong, S.-W. Li, A. Mohamed, H.-y. Lee and L.-s. Lee, "DUAL: Discrete Spoken Unit Adaptive Learning for Textless Spoken Question Answering.", *Annual Conference of the International Speech Communication Association*, Korea, 2022.

42. TQuad, "TQuAD", `https://tquad.github.io/turkish-nlp-qa-dataset/`, accessed on May 31, 2022.

43. Ünlü, M., E. Arisoy and M. Saraçlar, "Question Answering for Spoken Lecture Processing", *Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pp. 7365–7369, Brighton, United Kingdom, 2019.

44. Oppenheim, A., "RES.6-007 Signals and Systems", MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA, Spring 2011.

45. Povey, D. *et al.*, "The Kaldi Speech Recognition Toolkit", *Proceedings of the Automatic Speech Recognition and Understanding Workshop*, pp. 1–4, Big Island, HI, US, 2011.

46. Vardar, U. F., İ. T. Devran and S. Demir, "An XML Parser for Turkish Wikipedia", *27th Signal Processing and Communications Applications Conference (SIU)*, pp. 1–4, Sivas, Turkey, 2019.

47. Conneau, A., K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer and V. Stoyanov, "Unsupervised Cross-lingual Representation Learning at Scale", *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 8440–8451, Online, Jul. 2020.

48. Wolf, T., L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest and A. Rush, "Transformers: State-of-the-Art Natural Language Processing", *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pp. 38–45, Online, Oct. 2020.

49. Akın, A. A., "Zemberek-NLP", `https://github.com/ahmetaa/zemberek-nlp`, accessed on July 20, 2022.

50. Ünlü Menevse, M., Y. Manav, E. Arisoy and A. Özgür, "A Framework for Automatic Generation of Spoken Question-Answering Data", *Findings of the Con-*

ference on Empirical Methods in Natural Language Processing, Abu Dhabi, Dec. 2022.

51. Oppenheim, A., "RES.6-008 Digital Signal Processing", MIT OpenCourseWare, https://ocw.mit.edu. License: Creative Commons BY-NC-SA, Spring 2011.

52. Schweter, S., "BERTurk - BERT Models for Turkish", `https://doi.org/10.5281/zenodo.3770924`, Apr. 2020, accessed on May 15, 2022.

APPENDIX A: SQUAD EXPERIMENT RESULTS

Table A.1. QA results of various models that are trained with SQuAD 1K-5K settings.

		BERT		**mBERT**		**XLM-RoBERTa**		**ELECTRA**	
Annotated Data Size	**Training Setting**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
1K	Original	51.72	63.22	59.73	70.76	59.59	69.93	64.72	75.10
	Generated	47.16	58.38	52.01	63.86	55.08	65.95	57.49	68.83
	2 Step	**56.76**	**67.95**	**60.38**	**71.87**	**63.41**	**73.58**	**65.78**	**77.59**
2K	Original	61.03	71.78	66.35	76.31	66.04	75.42	71.43	81.18
	Generated	53.66	64.47	57.68	68.53	59.04	69.52	63.49	73.44
	2 Step	**63.75**	**74.07**	**67.18**	**77.10**	**68.24**	**77.56**	**73.72**	**82.57**
3K	Original	64.33	74.86	67.97	77.58	69.56	78.40	74.48	82.95
	Generated	52.47	62.77	58.83	69.30	61.14	70.46	65.64	75.18
	2 Step	**65.85**	**75.73**	**68.99**	**78.60**	**70.47**	**78.85**	**75.40**	**84.21**
4K	Original	66.15	76.15	68.87	78.81	70.29	79.15	74.87	83.87
	Generated	56.87	66.95	59.63	69.64	60.70	70.06	65.35	75.12
	2 Step	**67.86**	**77.93**	**70.02**	**79.49**	**70.54**	**79.61**	**75.70**	**84.46**
5K	Original	66.32	76.88	69.00	79.36	70.72	80.03	75.17	84.30
	Generated	56.24	66.39	60.72	70.74	62.37	71.91	66.51	75.90
	2 Step	**68.02**	**78.23**	**69.17**	**79.73**	**70.87**	**80.28**	**75.65**	**84.78**

APPENDIX B: SPOKEN SQUAD EXPERIMENT RESULTS

Table B.1. QA results of various models that are trained with Spoken SQuAD 1K-5K settings.

		BERT		**mBERT**		**XLM-RoBERTa**		**ELECTRA**	
Annotated Data Size	**Training Setting**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
1K	Original	37.11	48.82	45.15	57.32	37.84	49.34	50.64	61.73
	Generated	32.52	44.08	37.39	48.20	34.83	46.08	42.07	53.29
	2 Step	**43.15**	**55.62**	**45.49**	**57.78**	**43.97**	**56.86**	**52.94**	**64.30**
2K	Original	46.12	57.91	50.33	61.43	49.60	60.51	57.60	68.03
	Generated	41.41	52.62	43.99	54.65	42.65	53.55	51.54	61.90
	2 Step	**50.01**	**61.87**	**50.93**	**61.81**	**50.91**	**61.80**	**59.63**	**69.73**
3K	Original	48.53	60.98	52.63	64.32	50.25	62.25	58.66	70.00
	Generated	40.53	52.87	42.09	54.28	43.15	55.13	50.89	61.70
	2 Step	**51.07**	**63.28**	**53.09**	**64.87**	**52.10**	**63.35**	**59.37**	**70.31**
4K	Original	50.96	63.23	53.86	65.76	53.06	64.76	61.48	72.03
	Generated	41.84	54.07	45.62	57.09	44.48	56.02	52.91	63.30
	2 Step	**53.00**	**64.74**	**54.31**	**65.83**	**53.34**	**65.13**	**61.97**	**72.68**
5K	Original	51.95	64.01	54.67	66.67	53.82	65.83	62.70	73.25
	Generated	44.44	56.37	47.34	59.06	45.00	56.83	52.59	64.10
	2 Step	**54.85**	**66.57**	**56.31**	**67.96**	**54.12**	**65.96**	**63.99**	**74.08**

APPENDIX C: FQUAD EXPERIMENT RESULTS

Table C.1. QA results of various models that are trained with FQuAD 1K-5K settings.

		BERT		**MBERT**		**XLM-ROBERTA**	
Annotated Data Size	**Training Setting**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
1K	Original	47.52	65.89	42.57	61.78	34.41	50.59
	Generated	33.28	50.9	30.11	48.22	28.33	45.33
	2 Step	**48.75**	**67.67**	**44.32**	**63.92**	**44.60**	**63.71**
2K	Original	53.67	71	48.06	66.83	48.68	66.15
	Generated	47.74	65.77	42.03	59.55	41.91	59.91
	2 Step	**55.77**	**73.47**	**49.84**	**68.25**	**51.57**	**69.09**
3K	Original	56.40	73.33	50.13	69.03	51.69	68.73
	Generated	49.75	67.24	46.02	63.33	46.49	64.49
	2 Step	**58.59**	**75.55**	**53.48**	**71.42**	**53.98**	**70.98**
4K	Original	57.78	75.08	54.08	71.42	52.98	70.77
	Generated	51.76	69.37	47.46	65.35	48.18	65.24
	2 Step	**58.97**	**76.65**	**54.77**	**72.40**	**54.86**	**72.14**
5K	Original	60.13	76.95	53.70	71.48	55.80	72.50
	Generated	52.13	69.89	46.68	64.49	48.43	66.66
	2 Step	**60.26**	**77.42**	**55.52**	**72.21**	**56.24**	**73.16**

APPENDIX D: GERMANQUAD EXPERIMENT RESULTS

Table D.1. QA results of various models that are trained with GermanQuAD 1K-5K settings.

		BERT		**MBERT**		**XLM-ROBERTA**	
Annotated Data Size	**Training Setting**	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
1K	Original	34.48	53.11	39.93	57.54	39.52	56.47
	Generated	25.32	41.77	28.45	44.02	31.81	45.77
	2 Step	**36.98**	**55.41**	**42.20**	**59.58**	**41.56**	**58.80**
2K	Original	35.62	55.24	42.47	60.43	40.43	56.87
	Generated	32.99	51.02	36.52	52.86	37.02	53.74
	2 Step	**40.70**	**58.15**	**45.10**	**62.49**	**47.60**	**63.58**
3K	Original	40.61	59.31	46.55	64.30	46.78	62.74
	Generated	37.11	53.48	41.06	56.22	40.25	55.33
	2 Step	**45.60**	**62.35**	**47.37**	**64.98**	**47.14**	**64.52**
4K	Original	43.01	59.78	48.32	66.21	50.14	67.09
	Generated	38.34	53.28	40.88	57.10	43.60	59.56
	2 Step	**45.96**	**62.35**	**48.50**	**66.35**	**50.27**	**67.94**
5K	Original	44.19	61.84	49.09	65.75	50.14	66.97
	Generated	39.34	55.14	42.83	57.34	41.79	57.31
	2 Step	**47.23**	**64.35**	**53.13**	**68.38**	**51.41**	**67.66**

APPENDIX E: TURKISH LECTURES EXPERIMENT RESULTS

Table E.1. QA results of various models that are trained with Turkish Lectures Reading Comprehension dataset.

	BERT		**MBERT**		**XLM-ROBERTA**	
Training Setting	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
Original	**54.90**	80.86	45.10	76.64	50.00	78.92
Generated	37.41	65.12	33.22	59.99	27.97	54.88
2 Step	54.89	**82.70**	**49.65**	**78.43**	**51.40**	**80.02**

Table E.2. QA results of various models that are trained with Turkish Lectures Listening Comprehension dataset.

	BERT		**MBERT**		**XLM-ROBERTA**	
Training Setting	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
Original	48.95	78.36	41.96	73.22	39.51	75.82
Generated	36.71	66.60	26.92	59.50	27.27	59.09
2 Step (question-generation training with ASR Data)	**49.30**	**79.66**	**47.55**	**76.55**	45.80	**76.53**
2 Step (question-generation training with Text Data)	48.60	79.33	44.76	75.68	**46.85**	76.42

APPENDIX F: ENGLISH LECTURES EXPERIMENT RESULTS

Table F.1. QA results of various models that are trained with English Lectures Reading Comprehension dataset.

	BERT		**MBERT**		**XLM-ROBERTA**	
Training Setting	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
Original	41.15	68.17	41.31	67.70	34.33	61.16
Generated	6.88	37.40	12.02	48.07	8.57	43.35
2 Step	**45.19**	**73.42**	**42.81**	**71.29**	**36.61**	**64.23**

Table F.2. QA results of various models that are trained with English Lectures Listening Comprehension dataset.

	BERT		**MBERT**		**XLM-ROBERTA**	
Training Setting	**EM**	**F1**	**EM**	**F1**	**EM**	**F1**
Original	41.72	68.52	39.75	68.07	30.86	61.83
Generated	8.57	42.27	6.87	46.14	6.30	39.24
2 Step (question-generation training with ASR Data)	**41.72**	**70.49**	**40.94**	**71.11**	**33.24**	**64.41**
2 Step (question-generation training with Text Data)	40.03	67.39	38.88	70.51	32.30	63.98