Multilanguage Transformer Models for Answer-Aware Question Generation

Salvatore Correnti
s.correnti@studenti.unipi.it

September 8, 2023

Contents

Abstract

This report describes the experiments conducted over the MT5 and IT5 models on the Automatic Answer-Aware Question Generation task. The aim of the project is both to evaluate the performances of the two models in their "native" languages (English for MT5 and Italian for IT5) and to compare their performances on Question Generation in Italian. We also experimented with `Text-DaVinci-003`-powered Data Augmentation for MT5 on a subset of LMQG Squad dataset to see its impact on the quality of the finally trained model and with input "textual prompt tuning" giving additional information to the modle at training time both with and without usage of a Support Classifier for extracting features from the answer and the sentence in which it is contained.

1 Introduction

In Natural Language Processing, the task of question generation plays a pivotal role in various applications like question-answering systems and educational platforms. This task can be furtherly characterized by having or not a given answer in input to the model:

- **Answer-Aware Question Generation**, where the model receives in input both the context in which the question is formulated and one or more answers to the objective question. In this case, the model is expected to produce equivalent questions to the given one such that they are correctly answered by the given input answers;

- **End-to-End Question Generation**, where the model *does not* receive **any input answer**, but still has one or more objective questions whose answers may be different. In this case, the model is expected to generate questions that are correct and coherent with the context.

This project is focused on the **Answer-Aware Question Generation**, and in particular the objectives are to compare the performances of a multilingual model trained on an English dataset and then finetuned on an Italian one, and of a "native" Italian model finetuned on the same Italian one, and to design and test different techniques to improve the performance on the first model, aiming to transferability of the same to the second one.

We used two key models: **MT5 (Multilingual T5)**, trained on the **LMQG (Leveraging Monolingual Question Generation) Squad** dataset and **IT5 (Italian Translation Transformer 5)** fine-tuned on the **Squad-IT** dataset. To further improve the quality of the generated questions, we explored the integration of data augmentation techniques and the implementation of a support classifier model to suggest to the main model both at training and test time the most likely formulation of the given answer, represented by the start of the question (see section 5.5).
In this report we will discuss the methodologies employed, the datasets used for training and evaluation, the performance metrics applied, and the results of the experiments.

2 Related Works

Early Question Generation research focused on rule-based approaches that relied on syntactic and grammatical patterns to generate questions from given text. Subsequent studies explored machine learning techniques, such as supervised learning, to generate questions by training models on large question-answer pairs datasets [1].
In the last decase Question Generation saw significant progress thanks to the adoption of Deep Learning approaches and the increased computing power availability for experimenting with larger and deeper architectures. Examples of new approaches include the use of sequence-to-sequence models [2] and transformer-based models like `T5` [3] and `BERT` [4]. Models like `BERT` also made possible to adopt new evaluation metrics like `BERTScore`, which are based on semantic similarity of token embeddings in predicted and reference questions, thus exploiting a sort of "transfer learning" process [5].
Finally, the adoption of multilingual models made possible to extend the applicability of these models to over 100 different languages ([6]), addressing the issue of a significantly smaller amount of available training data for languages other than English.
Typically, success in this field is measured by achieving high results in metrics like BLEU, ROUGE, NIST-MT and BERTScore [7]. For instance, recent results on English by using `T5` showed the following results (table 1) on the Squad dataset [7]:

Learning Rate	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-L SUM	BERTScore
$5 \cdot 10^{-5}$	20.01	50.71	28.38	46.59	46.61	45.46
$3 \cdot 10^{-5}$	22.63	54.90	32.22	50.97	50.99	48.98
$2.5 \cdot 10^{-5}$	22.50	54.36	31.93	50.49	50.50	48.64
10^{-5}	20.17	50.46	28.38	46.79	46.81	44.97

Table 1: Results of the experiments with T5-Base on Squad dataset as reported in [7].

3 Data

3.1 Datasets

We have used two different datasets, both being variants of the **Squad** benchmark ([8]): **LMQG (Leveraging Monolingual Question Generation) Squad** [9] and **Squad-IT** [10], available respectively at the following links: `https://huggingface.co/datasets/lmqg/qg_squad` and `https://huggingface.co/datasets/squad_it`. Due to the long time needed for training the `MT5-Base` and `IT5-Base` models on the whole datasets even on V100 GPUs, the majority of the experiments is done on subsets of these two datasets, in particular for what concerns Data Augmentation (section 3.2). For each experiments the actual size of the used datasets is reported.

LMQG Squad This dataset is a subset of **QG-Bench**, a unified question generation benchmark recently proposed [9]. Each entry in the dataset contains a short paragraph and a couple of *one* question and **one** answer that is a substring of exactly *one* sentence in the paragraph. Besides this, the dataset is already pre-processed for being used for Question Generation tasks, and in particular it contains the following fields:

- `paragraph_question`: the question followed by the paragraph, marked at the beginning with `question:` and `context:` and separated by a comma;
- `answer`. `question`, `sentence`, `paragraph`: the answer, the question, the sentence containing the answer and the context paragraph, respectively;
- `sentence_answer`: the sentence containing the answer, which is internally delimited by a pair of `<hl>` tags;

- `paragraph_answer`: the entire paragraph with the answer internally delimited by a pair of `<hl>` tags;
- `paragraph_sentence`: the paragraph with the *sentence* containing the answer internally delimited by the `<hl>` tags.

This dataset differs from Squad by providing *only* one answer for each question, while the original Squad may contain multiple answers. Similarly to Squad, there may be multiple entries with the same context, each one based on a different sentence in the context itself. The training, validation and test splits contain respectively 75722, 10570 and 11877 examples.

Squad-IT This dataset has been proposed recently [10] and is intended to be an equivalent of the Squad benchmark for assessing Question Generation models quality for the Italian language. The dataset is obtained by semi-automatic translation of the Squad dataset into Italian and is smaller than the original Squad, containing 54159 and 7609 example for training and testing respectively. In particular, about 90% of the original Squad dataset was translated, but due to imprecisions and to the fact that in many cases the translated answer was not contained in the translated paragraph anymore, only about 45% of the dataset was still feasible, and after further corrections of the text or reformulation of the answers, the final dataset contains about 62% of the original dataset examples [10]. Since there is no default validation set, we split the training set into 43327 training and 10832 validation examples. Each example contains the following fields: `id`, `context` with the source paragraph, `question` and `answers`, which is a dictionary field containing the text of each answer and its start in the paragraph as two different lists.

As a consequence of the translation, Squad-IT differs from LMQG-Squad by having one or more answers per question, thus allowing to split up examples with multiple answers to augment the actual size of the dataset for a Question Generation task.

3.2 Data Augmentation

After several initial experiments, we augmented a subset of the `LMQG-Squad` dataset with the aim to improve the performances of the trained models. We decided to focus on the augmentation *of the questions* rather than of the answers, that is we reformulated each question into one or more equivalent ones and then added a new example consisting of the original answer and context and the new question for each generated one. We then compared the results of the finetuning on a small subset (2000 examples) of the original LMQG-Squad without data augmentation and the finetuning on the augmented dataset with 4 questions for each original one for a total of $10,000$ examples. The objective of this choice of the size of the original dataset and the number of equivalent questions per example was to give as much as possible "importance" to the actual *augmentation* of the data. This means to have a dataset in which the variability (for instance in terms of contexts and structure of the questions) of the data is intentionally "low" (at least with respect to the whole LMQG-Squad dataset) and the model could quickly achieve a low loss since the same answer will generally have a low Cross Entropy with *all* the equivalent corresponding questions, but at the same time the different questions could still have a certain degree of variability in how they are structured in order to assess how *paraphrasing* a question could lead to a better generalization capability of the model and to be able to compare the results with the ones obtained by an indipendent finetuning on a subset of $10,000$ examples taken from the original LMQG-Squad with *no* augmentation.

The idea behind this augmentation was that a possible weakness of an Answer-Aware Question Generation model is that for a given answer there could be *different* ways to formulate an appropriate question, and these could also be not necessarily equivalent (for example, asking *when* something happened instead of *where*), and so forcing the model to have a single target for each answer (i.e., a single question for each answer) may potentially have the side effect to lead to a worse generalization capability, possibly especially if the dataset is small. Instead, by giving the model different targets whose similarity (as captured by BERTScore, see section 3.2.1) is high, one can expect the model to tend to an "average" of all given questions, and to be less different to the original one by being more similar to a small cluster of equivalent similar questions. However, this means also to give to the model different labels for the same input, and it is not trivial that providing multiple outputs for the same inputs leads to a better generalization performance. The results of the previously described experiments suggest that this data augmentation can give similar results to a finetuning with a dataset with the same number of examples as the augmented one but without augmentation.

3.2.1 Data Augmentation Process

We used OpenAI `Text-DaVinci-003` model [11] for generating the reformulated questions. In particular, after a few preliminary trials, we used the following prompt:

```
"Propose <num_examples> equivalent questions to the given one.
You may use information in context. Prefer shorter questions.
Prepend each question with <eoq> and append <eoq> at the end
of each question. Do not include the text between <hl> in the
questions and do not change named entities in the given question."
```

with num_examples being the number of desired equivalent questions, `<hl>` the tag used in LMQG-Squad for delimiting the answer (see section 3.1) and `<eoq>` is a special tag for delimiting each question and simplifying the extraction of the questions from the generated output. We also used a temperature of 0.2 to get questions substantially similar to the original ones and a maximum number of tokens of 200.

For our experiment, we used num_examples = 4 and we then checked the generated questions to be as equivalent as possible to the original ones by using the BERTScore metric [5] as available on HuggingFace (`https://huggingface.co/spaces/evaluate-metric/bertscore`).
This metric uses pretrained contextual embeddings from existing `BERT` models to match the words in predicted and reference sentences by the *cosine similarity* of their embeddings, and returns the `precision`, `recall` and `F1` scores for the predicted sentence. This metric has been proposed in 2019 [5] and it has been shown to correlate well with human judgement about the semantic similarity between two sentences. In our case, we used `Roberta-Large` (`https://huggingface.co/roberta-large`) as model from which to take the embedddings and we verified that all generated questions have an `F1` score of at least 80%, with (...) over 90%.
We decided to not prune the lowest-ranked questions both for time and cost reasons and for possibly maintaining a slightly higher variability in the formulation of the questions. However, we limited our experiments on the already cited 2000-examples subset of LMQG-Squad with 8000 generated questions due to some inconsistencies in the generated text when asking equivalent questions for a batch of more than 1 question, and to the associated costs for OpenAI API usage.

3.3 Support Classifier

Besides the other experiments already cited, we also tried to improve the performances of the `MT5-base` model on LMQG-Squad by training a sort of ensemble of the actually finetuned model and a *support classifier* whose purpose is to suggest the other model at validation/test time what is the most likely initial prepositional phrase of the target question that "qualifies *what* the question should ask". For instance, given the answer:

```
"foreign protesters"
```

from the sentence:

```
"Thai authorities threatened to arrest <hl> foreign protesters
<hl> and ban them from future entry into Thailand."
```

we can reasonably infer that the most probable starting phrase would be:

```
"Who <verb> <body of the question>?"
```

indicating that we "qualify" the question as asking for *who is someone*.

We defined the following 16 classes:

1. 0–5 → `"<prep-who>"/"<prep-what>"/"<prep-which>"/"<prep-where>"/ "<prep-when>"`: a preposition followed by "who"/"what"/"which"/"where"/ "when"/"why" (e.g. "From who ...?", "In what ...?", "In which ...?", "From where ...?", "From when ...?");

2. $6-11$ → `"who"/"what"/"which"/"where"/"when"/"why"`: the given pronoun at the beginning of the question (e.g. "Why ...?");

3. 12 → `"<how-adv>"`: "how" followed by an adverb (e.g. how much/how many/how long);

4. 13 → `"how"`: only "How" at the beginning of the question;

5. 14 → `"<verb>"`: a question starting with a verb (e.g. "Did you ...?", "Were they ...?", "Is it true that ...?");

6. 15 → `"<unknown>"`: all other possibilities.

It is clear that this is not an exhaustive list of all possible phrases and that there may be some overlapping between different classes (e.g. "what" and "which"). We however tried to at least keep separate the most common pronouns (who, what etc.) when used "alone" from when their meaning is modified by a nearby preposition (e.g. "in", "to", "from"), and we used the `"<unknown>"` class to represent all other cases.

These classes can be translated in Italian as: `"<prep-chi>"`, `"<prep-cosa>"`, `"<prep-quale>"`, `"<prep-dove>"`, `"<prep-quando>"`, `"<prep-perché>"`, `"chi"`, `"cosa"`, `"quale"`, `"dove"`, `"quando"`, `"perché"`, `"quanto/quanti/per quanto tempo/..."`, `"come"`, `"è vero che/..."`, `"<unknown>"`, hence the trained classifier should be usable also together with a model finetuned on a dataset in Italian, although we did not experimented with this.

For generating the dataset we used LMQG-Squad, and in particular we took the first occurrence of a pattern as described for the labels (e.g. the first occurrence of "who") to label each example; although there may be cases in which this gives a (partially) wrong result, e.g. "Because of what ...?" mapped to "What...?", although this can be paraphrased as "What is the cause of ...?", we observed that by filtering the LMQG dataset according to this set of rules, only 2708, 272 and 414 of the training, validation and test examples belonged to the `<unknown>` class, being respectively 3.58%, 2.57%, 3.49% of their "source" datasets, and that about 80% of the training examples can be identified by simply looking at the *first two words in the question*, meaning that there is only a relatively small percentage of questions not captured by any of the other 15 classes and that, if present, the error caused by examples like "Because of ...?" should not affect at least the majority of the generated examples.

After training, we used this classifier at evaluation and test time to modify the initial part of the textual input given to the "main" model, and in particular we used both `top-1` and `top-2 accuracy` for modifying the input, as we describe in section 3.4. We did not use it at training time since the feature can be extracted from training questions with 100% accuracy by definition.

3.3.1 A slightly different approach

Together with the support classifier described before, we also experimented with a slightly different variant in which there is no explicit classifier trained before the main model, and instead there is *only* the input instruction modification as following for the training examples, while evaluation and test examples were not modified:

```
generate questions that start with <label>: ...
```

In this case we have used different classes, by mentioning explicitly the preposition *before* what, when and other pronouns instead of grouping them together in a single `<prep-...>` class. We have used the following classes:

`who`, `in what`, `at what`, `what`, `when`, `why`, `how many`, `how`, `where`, `in which`, `which`, `<unknown>`, with the `<unknown>` class not modifying the input. We have observed that on the same dataset this approach led to better results when compared to the support classifier above described (see section 5).

3.4 Data Preparation

Both datasets were preprocessed in order to have only the columns `question` and `answer_context` for the concatenation of the answer text and the context paragraph. We opted to add two special tokens `<answer>` and `<context>` for delimiting the answer and the context respectively instead of using the two markers *"answer: "* and *"context: "* and a comma for separating. We also kept the ¡hl¿ tag for highlighting the answer *inside* the paragraph and we prepended each answer-context couple with the instruction `"generate questions: "`. The following shows an example item from LMQG-Squad:

```
{
    "question": "What is heresy mainly at odds with?",
    "answer_context": "generate questions: <answer> establi-
    shed beliefs or customs <answer> <context> Heresy is any
    provocative belief or theory that is strongly at variance
    with <hl> established beliefs or customs <hl>. A heretic
    is a proponent of such claims or beliefs. [...] <context>"
}
```

For the Squad-IT dataset we created a separate example for each answer contained in the `answers` field.

For the Support Classifier (see section 5.5) we used as input only the sentence containing the answer with the latter delimited with ¡hl¿ *inside* the sentence, and as label the preposition associated with the question, as showed in the following:

```
{
    "label": 1,
    "sentence_answer": "Heresy is any provocative belief
    or theory that is strongly at variance with <hl>
    established beliefs or customs <hl>.",
    "answer": "established beliefs or customs"
}
```

with 1 corresponding to the `"what"` label.
After having trained the classifier and before starting the training for validation set and at test time for the test set, we modify the textual input by replacing the instruction prompt in different ways according to the confidence level of the classifier to each label. In particular, we considered the first two most likely labels and we set a threshold of 0.5 over the fraction: $\frac{p_2}{p_1}$, where p_1 and p_2 are the first and second highest probability values for a given example.

- If $\frac{p_2}{p_1} > 0.5$, we modify the instruction to be:

  ```
  generate questions that start either with <most_likely_label>
  or with <second_most_likely_label>: ...
  ```

  i.e. we suggest both possibilities to the main model;

- if $\frac{p_2}{p_1} < 0.5$, we modify the instruction to be:

  ```
  generate questions that start with <most_likely_label>: ...
  ```

  i.e. we suggest *only* the most likely possibility to the main model;

- in all cases, if p_2 corresponds to the `<unknown>` label we suggest either p_1 or nothing depending on the above rules, and if p_1 corresponds to `<unknown>` we do not suggest anything, i.e. we do not modify the prompt.

4 Models

We have used two pretrained models, both belonging to the `T5` "class" of models:

- **MT5-Base**: `MT5` is the multilingual version of Google's T5 model [6]. As T5, it is an encoder-decoder text-to-text model, meaning that its input and its outputs are always text strings, hence it is usable for different tasks at the same time, like text summarization, question generation and question answering. Similarly to T5, it has been trained on `mC4`, a Common Crawl-based corpus extracted from the Internet, although in this case the corpus contains texts in 101 different languages instead of only English for `T5`;

- **IT5-Base**: `IT5` is a version of `T5` pretrained on the **Thoroughly Cleaned Italian mC4 Corpus** [12], a corpus obtained by filtering `mC4` on Italian text and by further preprocessing for removing inappropriate words and inadequate sentences. Since it is pretrained on a dataset in Italian, it can be used for comparing the performance of `MT5` after being finetuned on both LMQG Squad and Squad-IT with the performance of a "native Italian" model finetuned only on Squad-IT.

For both the models we used the `base` versions, which contains about $300M$ parameters. The actual trained models are made up by the `MT5` and `IT5` models followed by a language modeling head, i.e. a linear layer of size $(768, sizeofthevocabulary)$ for Question Generation experiments, or by a classification head, i.e. a linear layer of size $(768, 16)$ for the support classifier.

5 Experiments

5.1 Experiments Overview

All experiments were run on Google Colab , using both `T4`, `V100` and occasionally `A100` GPUs to accelerate the computations. For all the experiments we used a learning rate of 10^{-4} with linear epoch decay, `AdamW` optimizer and a batch size of 4 due to limitations of training time and memory available on Google Colab. For training over the small subsets with 1000, 2000 or $10,000$ items we set a maximum of 10 or 20 epochs, while for training on the whole datasets we ran the experiments for a maximum of 2 epochs due to temporal and computing power requirements.

5.2 Evaluation Metrics

Aside from the standard CrossEntropy loss for training and evaluation, we used `BLEU` and `ROUGE` scores in its variants `ROUGE`, `ROUGE-L` and `ROUGE-L SUM` for evaluating the performances of our models. During evaluation and test we employed a beam search with 4 beams and selection of the best 4 candidates according to the loss for providing questions to evaluate. After this, we ranked the 4 candidates using `BERTScore` based on `roberta-large` and we took the one with the highest `F1` score as final candidate.

Both `BLEU` and `ROUGE` are standard metrics for evaluating Question Generation models, and in particular:

- BLEU operates by comparing the machine-generated text to one or more reference human-generated texts. It quantifies the similarity between the machine-generated text and the reference texts by counting matching n-grams and then calculating a precision score, which measures how many n-grams in the machine-generated text also appear in the reference texts. The BLEU score is calculated by considering precision at various n-gram levels (typically up to 4-grams), giving more weight to higher-order n-grams to capture longer linguistic patterns. The individual precision scores are then combined into a single BLEU score using a geometric mean [13]. The minimum and maximum values are respectively 0 and 1, with 1 indicating that machine-generated text and references are exactly equal. Following a convention already seen in other works [7], we report our BLEU scores in percentages;

- `ROUGE (Recall-Oriented Understudy for Gisting Evaluation)` evaluates the quality of machine-generated text summaries by comparing them to one or more reference human-generated summaries. It primarily focuses on measuring the recall of content overlap between the machine-generated summary and the reference summaries. ROUGE computes the recall at various levels like unigrams, bigrams, and other n-grams, usually referred to as ROUGE-1, ROUGE-2 and ROUGE-n, to capture different levels of similarity. Finally, these values are combined to calculate the final ROUGE score [14]. As for BLEU, scores are between 0 and 1, with 1 indicating a perfect match, and we will report those scores in percentages;

- `ROUGE-L` and `ROUGE-L SUM` are both variants of `ROUGE`. In particular, ROUGE-L measures the longest common subsequence between the machine-generated text and the reference text, i.e. the longest sequence of words that appears in both the generated and reference texts. Instead, `ROUGE-L SUM` finds the longest common subsequence of words between the generated summary and each of the reference summaries and then computes the score based on these values [14];

- Finally, we already described `BERTScore` in section 3.2. Since both BLEU and ROUGE are metrics originally designed to be used in Machine Translation and Text Summarization tasks but are still widely used in Question Generation, we included also BERTScore as it is designed to be "semantic-aware" of the texts it compares by using BERT embeddings, and we also used it as metric for selecting best generated question during evaluation and test.

5.3 Baselines on MT5 and IT5

Our first baselines experiments consisted in finetuning MT5 with small subsets of the LMQG-Squad dataset, in particular with 1000 and 2000 examples, to see how they perform with access to only a restricted training set. After that, we finetuned both MT5 and IT5 respectively with a subset of $10,000$ examples from LMQG-Squad and a subset of $10,000$ examples from Squad-IT. For generating the training subsets, we always shuffled the original training dataset with a seed of 0 for replicability and then took the first n examples for making sure that bigger datasets were always supersets of smaller ones. Results are shown in table 2, in particular we report the epoch in which we got the best validation loss results (counting from 1) and metric values over the *test* set. For these base cases, we did not compute BERTScore, as performances on other metrics are already quite low. As

Model-Data	Epoch	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-L SUM	BERTScore (F1)
MT5-1000	9	7.09	30.86	11.15	28.7	28.7	n.a.
MT5-2000	9	10.048	33.73	13.70	30.90	30.91	n.a.
MT5-10000	5	**21.583**	**51.008**	**28.978**	**47.335**	**47.333**	**91.567**
IT5-10000	2	21.123	44.064	25.574	41.073	41.064	87.35

Table 2: Results on MT5 with 1000, 2000 and $10,000$ examples and on IT5 with $10,000$ examples.

expected, we can see that with the smallest datasets the performances on test set are significantly worse than with $10,000$ examples. We see instead that with $10,000$ examples `MT5` seems to out-perform `IT5` in their respective languages. It is also evident how the adoption of a larger training set can lead to a substantial improvement in the score results.

5.4 Data Augmentation

After that, we applied the data augmentation as described in section 3.2 to both the 1000 and 2000 datasets to bring them to actual sizes of 5000 and 10000, and compared the results with the original ones and with the $10,000$ dataset for MT5. Results are shown in table 3.

We did not tried out Data Augmentation for $10,000$ examples datasets with MT5 and IT5 for issues relating to generating equivalent questions for batches of examples and OpenAI API usage limits.

Data	Augmented	Epoch	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-L SUM	BERTScore (F1)
5000	Yes	1	15.235	42.625	20.517	38.766	38.770	n.a.
10000	Yes	9	19.703	48.426	26.006	44.799	44.798	91.012
10000	No	5	**21.583**	**51.008**	**28.978**	**47.335**	**47.333**	**91.567**

Table 3: Results on MT5 with augmented 1000 $\rightarrow$ 5000, 2000 $\rightarrow$ 10000 and 10000 examples on LMQG Squad.

As we can see from table 3, training with augmented data gives a lower performance compared with training with non-augmented ones with same size due to the lower variability of the augmented datasets w.r.t. answers and questions, but we still see a substantial improvement from the non-augmented 1000 and 2000 dataset, with BLEU score approximatively doubling, and significant improvements also for the other metrics. Considering that 80% of these datasets is made up by augmented examples, this results may suggest that data augmentation is actually effective at improving the results even for small datasets. In this case, we actually shuffled the whole augmented dataset, while it may be possible that similar results may be obtained also by keeping original and augmented examples together and calculating the actual loss by averaging over all augmented questions, thus maintaining the computational cost of a several times (5 in this case) smaller dataset but keeping benefits, though we did not test it.

5.5 Support Classifier

Support Classifier was trained on the whole LMQG-Squad dataset pre-processed for the task as described in section 3.4 with a learning rate of 10^{-4}. Training lasted for 3 epochs ($\approx$ 160 min on an `A100` GPU), and we finally took the checkpoint after epoch 2 since it gave the best validation loss results. Its statistics are reported in the following table 4:

Train Loss	Validation Loss	Accuracy	Top-2 Accuracy	Top-3 Accuracy
0.1368	0.1155	65.9%	84.512%	92.511%

Table 4: Train and Validation Loss and Top-1/2/3 accuracy results on Test Set for the Support Classifier after epoch 2.

As we can see, Top-1 accuracy is relatively low with about 61%, while Top-2 and Top-3 accuracies are significantly higher. This led us to the idea of using Top-2 accuracy for suggesting 1 or 2 possible starts to the main model as described in section 3.4, though a similar experiment may be performed also with Top-3 Accuracy.

table 5 contains all the results for base 10000 dataset with *no* input modifications, for modifications of evaluation and test set using only Top-1 accuracy (i.e., suggesting only 1 start for each example) and with Top-2 accuracy with a threshold of 0.5 as described in section 5.5. We also include the results obtained for the slightly different experiment with *no* explicit classifier (i.e., only modifications of training inputs) as described in section 3.3.1 and the results obtained by a full training over the whole LMQG Squad training set of 75722 examples.

Train Type	Epoch	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-L SUM	BERTScore (F1)
Baseline	5	21.583	51.008	28.978	47.335	47.333	91.567
Top-1 Accuracy	5	22.260	51.075	29.226	47.338	47.347	91.447
Top-2 Accuracy	6	**22.264**	**51.248**	**29.422**	**47.557**	**47.562**	91.582
No Classifier	8	**24.427**	**56.006**	**32.816**	**52.080**	**52.109**	**92.133**
Full Training	2	**24.205**	**53.989**	**32.251**	**50.242**	**50.249**	**90.248**

Table 5: Results on MT5 with baseline 10000 dataset with no input changes, with Top-1 and Top-2 accuracy and with input modifications *without* classifier.

All three approaches improved the results on all metrics, though Top-1 Accuracy and Top-2 Accuracy approaches got almost same results, and in general we got improvements of about 3% for BLEU and 0.5/1.5/0.47/0.48% for ROUGE w.r.t. the baseline, hence it is difficult to actually assess the impact of the usage of the classifier. Further experiments may be done in future works to assess also the results by using Top-3 accuracy and by using thresholding also on confidence values for each labels.
We notice however how the approach without a classifier by only adding extra information about the question in the training set, as described in section 3.3.1, led instead to substantial improvements w.r.t. previous cases and on the same level as a full training after 2 epochs. Further experiments, e.g. on the whole dataset, may tell if this is due for instance to actually *suggesting* a preposition to the model (e.g. "In what" instead of "a preposition and what"), and see if these results may be replicated consistently also with higher subsets of LMQG Squad or other datasets.

5.6 MT5 over Squad-IT

The MT5 model checkpoint obtained after the full training on LMQG Squad and applied on Squad-IT *without* any finetuning on Italian datasets achieved very low results on the evaluation metrics: 6.83 for BLEU and

21.992/9.256/20.483/20.487 for ROUGE. Taking a look at a couple of examples:

- Original Questions:
  1. Quale percentuale del patrimonio globale nel 2000 era di proprietà di appena l' 1% degli adulti?
  2. Quali cambiamenti di condizioni possono rendere insostenibile la foresta pluviale amazzonica?
- Generated Questions:
  1. How much of the patrimonio globale does the più ricco 1% degli adulti possess in 2000?
  2. What causes a foresta pluviale amazzonica to become insostenibile?

It seems that there are "key" parts of the question, e.g. "quantifiers" like "How much" or "What" or verbs, that the model is not capable to translate in Italian, while it seems to perform well with nominal phrases.

We have then performed a finetuning of the model checkpoint on the whole Squad-IT dataset, getting the following results after 1 epoch (table 6):

Train Type	Epoch	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	ROUGE-L SUM	BERTScore (F1)
IT5-10000	2	21.123	44.064	25.574	41.073	41.064	87.35
Both Datasets	1	19.912	43.749	25.186	40.946	40.932	86.276
IT5-full training	2	21.719	44.676	26.202	41.612	41.607	87.612

Table 6: Comparison on Squad-IT test dataset between the MT5 model after training on both LMQG Squad and Squad-IT and IT5 model after training on 10000 examples from Squad-IT.

We see that MT5 gets almost the same results as IT5 trained on 10,000 examples of Squad-IT, although this means that its performances are lower than a "native Italian" model.

6 Conclusions and Future Works

Our research had the objective to compare the performances of a finetuned multilanguage model and a "native Italian" model for the task of Question Generation in the Italian language, and moreover to experiment

with Question Generation by finetuning large language models. We chose the *T5* class of models in their `base` version since we considered them a good compromise between the size of the LLM and the computing power required for conducting experiments with it.
We started with a baseline given by small subsets of the LMQG Squad dataset and then we experimented with Data Augmentation of the 1000 and 2000 elements subsets and we introduced the support classifier based approach, obtaining generally good results, in particular for what concerns the Data Augmentation on small datasets and the modification of the training inputs according to the classification of the answers outlined in section 5.5 and section 3.4.

As already stated across the report, there is still significant room for improvement and future works, in particular for what it concerns the extension of Data Augmentation to the whole LMQG Squad and Squad-IT datasets, further experimenting with "textual prompt tuning" (with or without an explicit classifier) and how to actually extrapolate labels from the questions, and further experiments with the whole datasets to better compare the performances of a multilanguage model w.r.t. a single language-specific one as IT5. Other possible experiments may involve testing how Data Augmentation and Support Classifier impact on the performances of *small* models, e.g. `MT5-Small` and `IT5-Small`, as they reasonably will have worse performances in the same conditions but are much faster to train compared to their *base* or *large* versions.

References

[1] Michael Heilman and Noah A. Smith. "Good Question! Statistical Ranking for Question Generation". In: *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Los Angeles, California: Association for Computational Linguistics, June 2010, pp. 609–617. URL: `https://aclanthology.org/N10-1086`.

[2] Xinya Du, Junru Shao, and Claire Cardie. "Learning to ask: Neural question generation for reading comprehension". In: *arXiv preprint arXiv:1705.00106* (2017).

[3] Colin Raffel et al. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning*

Research 21.140 (2020), pp. 1–67. URL: `http://jmlr.org/papers/v21/20-074.html`.

[4] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[5] Tianyi Zhang et al. "Bertscore: Evaluating text generation with bert". In: *arXiv preprint arXiv:1904.09675* (2019).

[6] Linting Xue et al. "mT5: A massively multilingual pre-trained text-to-text transformer". In: *arXiv preprint arXiv:2010.11934* (2020).

[7] Cheng Zhang. "Automatic Generation of Multiple-Choice Questions". PhD thesis. University of Massachusetts Lowell, 2022.

[8] Pranav Rajpurkar et al. "Squad: 100,000+ questions for machine comprehension of text". In: *arXiv preprint arXiv:1606.05250* (2016).

[9] Asahi Ushio, Fernando Alva-Manchego, and Jose Camacho-Collados. "Generative language models for paragraph-level question generation". In: *arXiv preprint arXiv:2210.03992* (2022).

[10] Danilo Croce, Alexandra Zelenanska, and Roberto Basili. "Neural learning for question answering in italian". In: *AI* IA 2018–Advances in Artificial Intelligence: XVIIth International Conference of the Italian Association for Artificial Intelligence, Trento, Italy, November 20–23, 2018, Proceedings 17*. Springer. 2018, pp. 389–402.

[11] Tom Brown et al. "Language models are few-shot learners". In: *Advances in neural information processing systems* 33 (2020), pp. 1877–1901.

[12] Gabriele Sarti and Malvina Nissim. "It5: Large-scale text-to-text pretraining for italian language understanding and generation". In: *arXiv preprint arXiv:2203.03759* (2022).

[13] Kishore Papineni et al. "Bleu: a method for automatic evaluation of machine translation". In: *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*. 2002, pp. 311–318.

[14] Chin-Yew Lin. "Rouge: A package for automatic evaluation of summaries". In: *Text summarization branches out*. 2004, pp. 74–81.