Master's Thesis

Automatic Data Generation for Multiple Choice Question Answering

Master's Thesis

Jorge Osés Grijalba
Master's Thesis for the

Master in Language Technologies

Universidad Nacional de Educación a Distancia

Directed by

Prof. Dr. D. Álvaro Rodrigo Yuste

September 2022

Resumen

En este trabajo exploramos el uso de datos artificiales para mejorar el comportamiento de modelos de respuesta a preguntas multirespuesta en contextos de escasez de datos generados por humanos, con especial foco en los distractores de dichas preguntas. Primero proponemos un proceso a seguir para generar este tipo de datos para posteriormente medir de forma precisa y cuantificable el aporte de estos datos al entrenamiento. Finalizamos el trabajo con un caso concreto en el que mejoramos el rendimiento de nuestro modelo mediante este proceso, y finalizamos explorando las limitaciones que se nos presentan.

Abstract

In this work we explore the use of artificially generated data to improve the behaviour of Question Answering models in the context of Multiple Choice Questions, focusing specially on the distractors of said questions. We propose a general course of action to take the original data, generate new data from it and measure the gain in learning that our model is able to extract from it. We then carry on with a specific example where we show our model indeed gets better with this process, to then spend some time studying the limits of this proposed approach.

Index

Figure Index

Table Index

Chapter 1

Introduction

In this work we'll be focusing in improving model performance in the task of answering Multiple Choice questions from a given text.

This task usually provides the model with a sample text and a question whose answer can be inferred from reading the text. The task of answering this question is usually given the name of *Reading Comprehension*.

Multiple Choice versions of this task involve presenting the reader (or model) with a number of closed options to choose from. The correct one we'll call the *answer* and the others will receive the name of *distractors*.

Most of the big advances in Machine Learning in general and Natural Language Processing in particular have come from training models in bigger and bigger datasets, which might make them difficult to use to our advantage in contexts where data scarcity is commonplace.

The most usual approaches that we find nowadays that tackle this problem somewhat succesfully are the ones based in transfer learning, where we make use of a model already trained on larger general purpose text collection and then transfer this knowledge by learning this specific task on a smaller collection.

Although this allows us to get better performances, Question Answering multiple choice collections remain smaller and less developed compared to other collections intended for tasks like simple reading comprehension or translation where large datasets can be taken straight from the web or other sources.

This is due to the fact that generating this data is usually relatively costly for a human, who has to first read the texts presented to them, then think of some questions that would judge sufficient comprehension of the text presented, and then think of other possible (yet wrong) answers so that finding the correct one presents enough of a challenge to the reader.

In turn, our main motivation is to propose methods for the generation of question and answers collections of the Multiple Choice kind that allow us to improve present day prediction models, providing them with larger datasets so they can learn more from them.

The other task for models that is important to this work is that of *Question-Answer Generation*, which can be split into *Question Generation* and *Answer Generation*, although they are tackled together in some cases. The goal of *Question Generation* is to automatically produce questions that can be answered with information present exclusively in the content of a certain given content. This content will be text in our case, but it can be found databases or even images in some recent developments. The answers and questions themselves can be classified into many different kinds. For example, in our case the answers might be of different lengths, ranging from words to named entities to entire sentences. We might also be asking for a concise, short answer or expecting a longer, more thorough answer.

One of the most obvious applications of this lies in education, where question generation helps in evaluating reading comprehension. It can also be used to create quizzes and online tests saving hundreds of hours of manual effort from educators, so our purpose then can be summed up as saving time for professors and every educator involved in the process of exam design. Another human application of the development of these models can be a serving as a spowerful tool for the practice of students preparing for a real test.

Often questions are generated from already available pre-selected answers in a process usually called *extractive*, although it's often the case we want to generate questions that query the text for additional information not contained within.

The main course of our work will involve generating new questions, answers and distractors from a corpus of given texts, that we will use to try and improve model performance in the task of Question Answering questions of the Multiple Choice kind, including generating the distractors involved so they are of sufficient quality to aid the model in training.

This would provide models with a cheap way of training themselves with an expanded corpus in a context of special scarcity, as well as potentially making a first step in generating these questions and distractions well enough so that they can be used to expand the corpus for the human task of writing exams.

In tackling this challenge, we've faced a number of issues that we elaborate on in greater detail throughout this essay. Also due to the experimental nature of this work, at some points it isn't clear if one strategy to follow is better than another. In those cases, when it was the case the decision was a matter of importance for our objectives like finding out if there is a better strategy for distractor generation than another, or if a certain type of question-answer pair yields inherently better results, we've opted for a branched approach, detailing the different experiment's results in different tables.

We've accompanied this with descriptions of each experiment, their results and either any difficulty faced during their realization or if they confirmed or rejected our initial hypotheses.

Before moving on with the rest of the work, we include a brief description of the aim of each chapter within this document.

We will first take a look at the related work in the field, glossing over a relevant collection of related works. After this we'll take a look at the data collections we'll be working with. Once this is done, we'll go into a couple of more general chapters that will explain the process followed to carry on our experiments, and after them the last chapters will go into them with more detail and analyze the results we've obtained.

The code developed to carry the experiments has been made public in this repository, as well as the instructions for the setup in a Google Colab environment.

1.1. Objectives

The main objectives of this work are to evaluate methods to automatically generate Multiple Choice collections and help evaluate their contribution to improve current systems. To this end we will work towards the following objectives:

1. Generate *Question-Answer* pairs from a given text.
2. Propose different methods of distractor generation for said pairs.
3. Evaluate how the quality and quantity of these tuples affects current systems.
4. Evaluate how the different types of distractors impact the results.

1.2. Structure

Before moving on we will delve deeper into the structure of this work. The main focus of each chapter remains as follows:

Chapter 1. Introduction. In this section we've stated our objective and laid out the structure of this document and the elements we'll be working with.

Chapter 2. Related Work. In this chapter we introduce the reader to the state of the art of the types of models we'll be working with, as well as question and answer generation.

Chapter 3. Overview. Through this one we will go over the general process followed and our logic behind it, without specifying any of the actual technologies or data used for our experiments.

Chapter 4. The Data. This section will analyze the datasets we'll be working with, their properties, format and preprocessing needed to work with them.

Chapter 5. Choice Of Technologies. In which we go over our requirements and our thoughts behind the choices of the different models we'll use throughout our experiments.

Chapter 6. Baselines. In this section we evaluate the baselines for our models, and get some base results with real data to compare our results with synthetic data against.

Chapter 7. Synthetic Data Generation. In this chapter we'll elaborate on the process followed to generate new synthetic samples we'll use later.

Chapter 8. Synthetic Data Analysis. This in turn analyzes the synthetic data generated in the previous version, making sure it makes sense and holds some predictive value.

Chapter 9. Synthetic Data Evaluation. This section contains the results of the main experiment, improving model performance with synthetic data generated from previously-seen texts.

Chapter 10. Limits Of Synthetic Data Generation. In this section we'll explore some of the limits encountered while following this process.

Chapter 11. Conclusions and Future Work. This chapter compiles the different conclusions reached throughout this work proposing how to expand upon it.

Chapter 2

Related Work

Now we'll move on to explore relevant the state of the art of several fields which are relevant to our work. We'll first go through the data collections available, to then move on with the *Question-Answer Generation*.

2.1. Datasets

As we mentioned in the introduction, the multiple choice collections are costly to generate and hard to evaluate. Some of the relevant collections that have been commonly employed to train Question Answering systems are the SQuAD(Rajpurkar et al., 2016) and the QuAIL(Rogers et al., 2020) datasets.

The Stanford Question Answering Dataset (SQuAD) was initially thought out for the reading comprehension task, and its composition process illustrates the difficulties faced by earlier efforts to create large scale collections.

For the process of generating 100,000 questions from a set of Wikipedia articles, a crowd of human workers was put to work to generate questions where the answer either lied within the text as a fragment or span, leaving some questions to be unanswerable. As one can imagine, this effort was very costly from the human side, and it was so even more when they wanted to expand it in the 2.0 version, with 50,000 more unanswerable questions that were written by other human workers in an adversarial process to look very similar to the 100,000 ones that were written before.

This large efforts required to create big collections meant that a lot of them presented poor diversity of data.

This was further diagnosed as an issue by a variety of studies. One very illustrative example can be found in (Gururangan et al., 2018), where the word *never* is found to be very predictive of the contradiction label in the Stanford Natural Language Inference corpus.

When the workers were faced with the task of creating a large number of questions, turns out negating was a cost-effective strategy for the workers follow to generate contradictory statements than the models later are able to pick up, in detriment to facing more diverse questions.

Some of the knowledge they gathered in the process illustrates certain aspects of crowd-sourcing that are hard beyond what we've already identified. First, not every text is fitted to produce all types of questions, so the crowd could be tasked for example with generating true-or-false questions from an opinion piece that is lacking in this kind of information.

Secondly, if instead of specifying the question type the worker is left free, most of the times the question generated are simple factoids or too repetitive what questions. For example, in collections created this way like (Kočiský et al., 2018) the workers generated a corpus where 40 % of the questions belong to the simplest factoid type.

The Question Answering for Artificial Intelligence (QuAIL)(Rogers et al., 2020) dataset tries to avoid this by reducing the amount of spurious correlations with model predictions that these types of situations generate, balancing different types of questions within the same dataset.

Other approaches have taken the adversarial route further, using models in their processes that automatically reject the easy questions provided, like in(Dua et al., 2019b). This serves as a way to weed out the weakest questions provided by the workers, but as a result makes the resulting sample smaller.

Also approaches on multi-dataset training have been attempted in (Dua et al., 2019a), but althoguh a model trained in one dataset does not necessarily generalize to another, training on multiple datasets might create more general-purpose systems. Special care has to be taken in this case in order to prevent the model identifying which kinds of questions are usually paired with each kind of text.

2.2. Question-Answer Generation

Historically Question Generation consisted on rule-based approaches like (Heilman y Smith, 2010), (Lindberg et al., 2013); (Labutov, Basu, y Vanderwende, 2015). In 2017 we started to see neural approaches like ((Du, Shao, y Cardie, 2017), (Zhou et al., 2017)) that have ended up outperforming these early rule based approaches. This use of rule based question generation often employed over-generation and ranking questions with regression models over manually tagged datasets of linguistic features.

These either take into account the text as a whole, like the ones we just cited, or add a layer of importance paragraph-level information like (Du y Cardie, 2018); (Song et al., 2018).

In the neural models we've also seen success with a reinforcement learning like approach, using evaluation metrics ((Song et al., 2018)) or the accuracy (Ling, An, y Hasan, 2017) as rewards, and we've even seen others use (Alberti et al., 2019); (Shoemark et al., 2019) use pre-trained language models.

In that line, we can also encounter Variational AutoEncoders, which are probabilistic models used in a variety of other *Natural Language Processing* tasks. (Zhang, Yao, y Yan, 2018) use Variational Autoencoders for QG.

Up until these point, all of these works received a correct *answer* as input, for which a question must be generated. Another line of work generates Question-Answer pairs from context, although it is a relatively less developed task. Nonetheless we can encounter some recent works like(Alberti et al., 2019)

In other more recent works we can see that they propose a VAE for end-to-end QAG, resolving QA-pair consistency by maximizing their mutual information (Kim et al., 2019) (Yeh y Chen, 2019).

With the help of Question Generation models, it is also possible to train the Question Answering models in a semi-supervised learning manner to obtain improved performance.

Another interesting approach is that of (Sachan y Xing, 2018), where they show a curriculum learning process to supervise the Question Generation model, and in the process get more and more difficult questions for the Question Answering model.

(Dhingra, Danish, y Rajagopal, 2018) introduce a cloze-style QAG method to pretrain a QA model. Zhang and Bansal (2019) propose to filter out low-quality synthetic questions by the answer likelihood.

The models of the current state of the art make use of more modern *transformer* based language modeling, like we can see in (Alberti et al., 2019).

(Alberti et al., 2019) implements seq2seq models that generate questions, then enforce answer consistency on synthetic questions to filter out poorly generated questions. This technique receives the name of *roundtrip consistency*.

Other works unify this approach into a single transformer model like (Dong et al., 2019).

Some work uses NER or linguistic parsers to select passages for cloze translation as in (Lewis, Denoyer, y Riedel, 2019); (Dhingra, Danish, y Rajagopal, 2018).

To improve the quality of synthetic data generation and downstream QA models, improving language model quality is crucial.

In addition to pretraining task innovation, BERT(Devlin et al., 2019) has shown that increasing the size of available pretraining data directly improves performance in downstream tasks.

Other models like T5 (Raffel et al., 2020) or GPT-3, have shown that increasing language model scale improves the quality, coherency, and correctness of text generation.

The T5 model (Raffel et al., 2020) was originally published by Google in February 2020. Suited for a variety of tasks, it was developed after conducting a wide survey to pin down the techniques and data that worked best for the pretraining of this model, which resulted in the Colossal Clean Crawled Corpus, or C4 for short, which was made publicly available.

T5 reframes all NLP tasks into one unified text-to-text format where input and output are always strings. The following chart 2.1 taken from the previously cited paper illustrates its different capabilities.

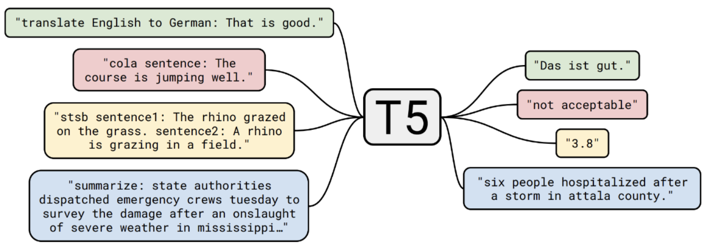
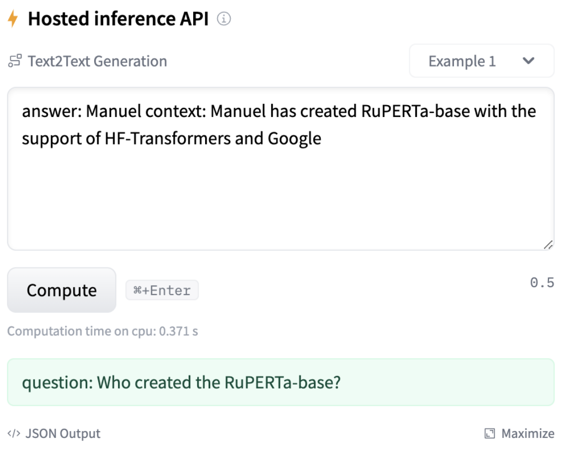

"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 2.1: Example of tasks T5 is capable of doing

In our case, we'll be making use of a finetuned version of T5 on the SQuAD dataset hosted publicly on HuggingFace here(Raffel et al., 2020), providing it with the intended markings for our task. In the following figure we can see an example of the T5 model generating a question, executed in the HuggingFace API.



answer: Manuel context: Manuel has created RuPERTa-base with the support of HF-Transformers and Google

Computation time on cpu: 0.371 s

question: Who created the RuPERTa-base?

Figure 2.2: Example of T5 Question Generation

2.3. Multiple Choice Questions and Distractor Generation

Multiple Choice Questions are widely used for a variety of student assessments and tests. They consist of a text, a question, the answer and a number of wrong yet plausible enough options that we'll call *distractors*.

Traditionally generated through human-based methods, automatic generation of distractors poses a number of challenges currently, and it's far from practical use.

These distractors might face a number of problems, given that we as humans tend to have a wider context of the words used in them compared to the models, which might lead to perfectly valid distractors the model might get fooled with are almost impossible to be chosen by humans.

Most of the methods are for single distraction generation as in (Chung, Chan, y Fan, 2020), but multiple distractors are usually required.

More complex like (Gao et al., 2018) try to generate longer and semanticrich distractors, which would be close to real distractors. They propose a hierarchical encoder-decoder framework with static and dynamic attention mechanisms to tackle this task.

Most of the work on this field so far has been focused on creating data good enough that humans could use, either for academic tests like in (Kalpakchi y Boye, 2021) or for quizzing about news stories (Lelkes, Tran, y Yu, 2021).

It could however be the case that relatively simple distractors that are not able to fool humans could be of use to train models that could learn something from them, expanding datasets in a context where data scarcity is prevalent and sourcing the task of generating the texts to humans has proven to present challenges. That is where our work comes into place.

Chapter 3

Overview

3.1. Synthetic Data Generation

We will now gloss over the general process we propose to carry this experiment, before specifying the actual tools we've used and the specific decisions we took.

Our general aim here is to expand a certain dataset so the model it's trained on yields better results compared to training it on the previously unexpanded dataset. Let's say a sample (or row) of our initial dataset is composed of a given *Text-Question-Answer-Distractors* tuple.

We start with a sample of a certain size whose results of the training we think could be improved with more data, as in one of the contexts of data scarcity we mentioned earlier. From this sample, we apply a process by which we generate synthetic data. For every text in the sample, we will first generate Question-Answer pairs through some heuristic. In general the easier kind would be *extractive*, where the answer is found within the text. Once we have a number of pairs for the text, it's time to generate approapiate distractors for each one, with our strategy of choice. The result would be several Question-Answer-Distractor tuples for each text in the real sample, that we will then join back with our dataset.

There are some smaller parts within this process that need to be catered to each specific case, and might benefit from a more in-detail analysis.

3.1.1. Question-Answer Generation

Most of the question generation models belong to the extractive category, although there are some that are able to generate answers that are not in the text. We have to be aware of the type of collection we're dealing with so our Question-Answer generation method creates Question-Answer pairs that are as similar as possible to the ones our model we'd be encountering.

For example, generating long answers when our model will mainly deal with short ones may lead to worse results than intended, and generating only answers present in the text might be detrimental if our model will deal with more abstract questions that cannot be found literally within the text.

This analysis of the texts has to be carried out beforehand. It's important to note that we can mix different strategies to generate different kinds of questions if we feel like we'd be facing some variety from it.

For example, if our method to generate *Question-Answer-Distractor* tuples from these texts is able to generate three distinct tuples for every text, our proportion will be at most one real sample for every three synthetic ones.

3.1.2. Distractor Generation

Several strategies in distractor generation might be followed. At this point, we have the *Text-Question-Answer* tuple generated. Distractors would ideally be similar enough to the answer so they cannot be inferred without the text, and at the same time be different enough so that they cannot be taken as a valid answer. Generating distractors is the most tricky part for the human side. We are usually able to generate these with human knowledge, that is, with some knowledge external to the text itself, so they pose some sort of challenge. There are some differences between what a valid distractor might be for some models compared to what a valid distractor might be to humans because of that external knowledge of the world.

In our experiments we propose a variety of methods for the generation of distractors, and test some of them. The most basic ones involve recycling other answers from other questions as distractors for the shorter answers, and switching around words for others of the same grammatical category for the longer answers. Other systems with more outside-world knowledge might be employed to this end, like ontologies, antonyms or sinonyms, if we want more elaborated answers.

If we intend to use this synthetic data for students to train for tests, more sophisticated methods would be required. However, as we see in the next sections, for the purpose of making models better more simple methods suffice.

3.2. Evaluation

3.2.1. Synthetic Data Validation

The first evaluation that needs to be carried out is the initial validation of the generated synthetic data. What we're asking here is not really if this data is useful to our model, which will be evaluated later, but whether these questions make sense from a human point of view, if the distractors are coherent and if the questions are challenging enough.

In general, the threshold for the synthetic data to be valid for human use would be higher, but we aim for it to be at least formally coherent and diverse enough to provide good training for our model later.

One way to validate if this data is good enough for our model is to try and train it only on this synthetic data, and seeing if it improves performance over completely random guesses. If that is the case, we can broadly judge it to have some predictive capabilities.

3.2.2. Model Evaluation

Once we have judged our synthetic data good enough in form, we have to evaluate it in function.

In the evaluation process we will be evaluating our newly generated data in two different ways. In the first one, we will be trying to answer the question *How does the real data compare to the expanded data?*, while in the second one we'll be asking *How would this compare to expanding our initial dataset with another real dataset?*

Our general aim here is to train (or finetune) two models of the same kind with two different samples, and compare the results of the two models with the same test set. Let's expand on the two processes.

3.2.3. Gains from Baseline

Our first sample will be composed of a *real* component, that is, a certain amount of data from an actual man-made dataset, and another *synthetic* component, that is, one generated from a Question-Answer-Distractor generation process of our choice from either the first collection of texts or the test set. The proportion between the two depends on the texts themselves and the method we'll be using to generate the questions.

Essentially, we're trying to see how big is the performance gain observed when adding the synthetic data to the previously used train data said synthetic data has been generated from. To this end we will use the comparison between the accuracy after training the model real data and after training it with both our initial data and the synthetic data generated from it.

3.2.4. Gains versus Real Data

Assuming a certain amount of information can be extracted from every text, the quantity of synthetic data we'll be able to generate will inevitably be constrained by an upper bound depending on the number of texts and the amount of information that we're able to employ from each text to generate distinct *Question-Answer-Distractor* tuples.

The second sample we'll be using will include a copy of the real part of the first sample, but will be expanded with more real tuples, preferably from outside the test set, until it meets the size of the first sample.

These two-sample approach means to replicate two different situations. In the first one, we're replicating a situation of scarcity where no more data is available, and is partly subdued by our approach.

In the second one, we aim to replicate a situation where an equivalent amount of human-made data was provided to our system.

Chapter 4

Data

We'll be working with two bodies of data, the *RACE*(Lai et al., 2017) dataset and the *EntranceExams* English dataset. Both of them consist of a collection of exams and tests collected and treated in various ways. In this section we'll aim to describe them and cover their caveats.

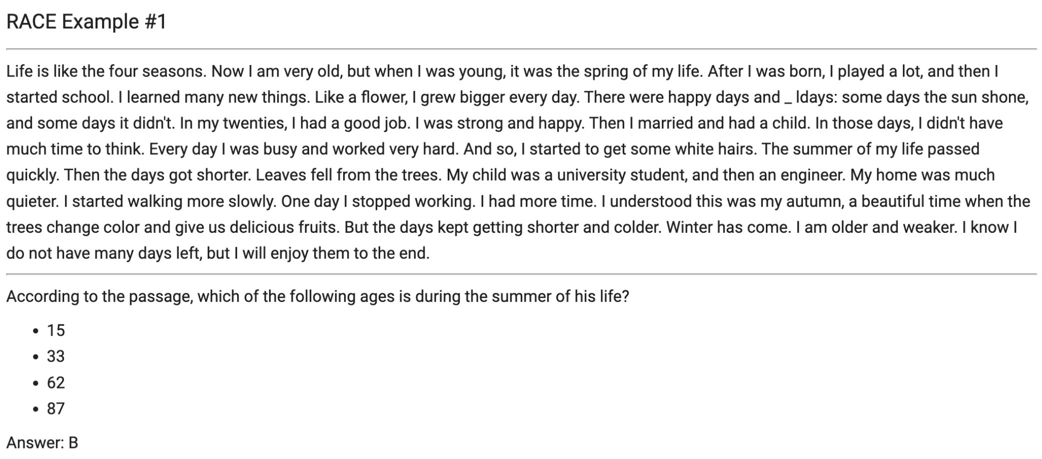
4.1. Choosing Our Data

The data we'll be choosing will be based on a series of criteria that we think will make our experiment easier.

First, we don't want a dataset that is too small. To simulate data scarcity we might just work with a subsample of it if it is too large, but if it is too small we might not be able to generate sufficient data. It ideally has to be one of the commonly used collections, which are more tested and better made and thus give us better benchmarks to our approach.

Keeping another collection of data on the side to test if the learning gains from synthetic data might be applied to another dataset is also a good idea, and we can use a small collection to this end to also test the limits of our approach for smaller collections. We end up settling with the *RACE*(Lai et al., 2017) collection, commonly used for a lot of NLP tasks as our main collection, while we will be using the *EntranceExams* collection as an smaller collection for some side experiments.

Let's see them in detail.


RACE Example #1

Life is like the four seasons. Now I am very old, but when I was young, it was the spring of my life. After I was born, I played a lot, and then I started school. I learned many new things. Like a flower, I grew bigger every day. There were happy days and _ ldays: some days the sun shone, and some days it didn't. In my twenties, I had a good job. I was strong and happy. Then I married and had a child. In those days, I didn't have much time to think. Every day I was busy and worked very hard. And so, I started to get some white hairs. The summer of my life passed quickly. Then the days got shorter. Leaves fell from the trees. My child was a university student, and then an engineer. My home was much quieter. I started walking more slowly. One day I stopped working. I had more time. I understood this was my autumn, a beautiful time when the trees change color and give us delicious fruits. But the days kept getting shorter and colder. Winter has come. I am older and weaker. I know I do not have many days left, but I will enjoy them to the end.

According to the passage, which of the following ages is during the summer of his life?

- 15
- 33
- 62
- 87

Answer: B

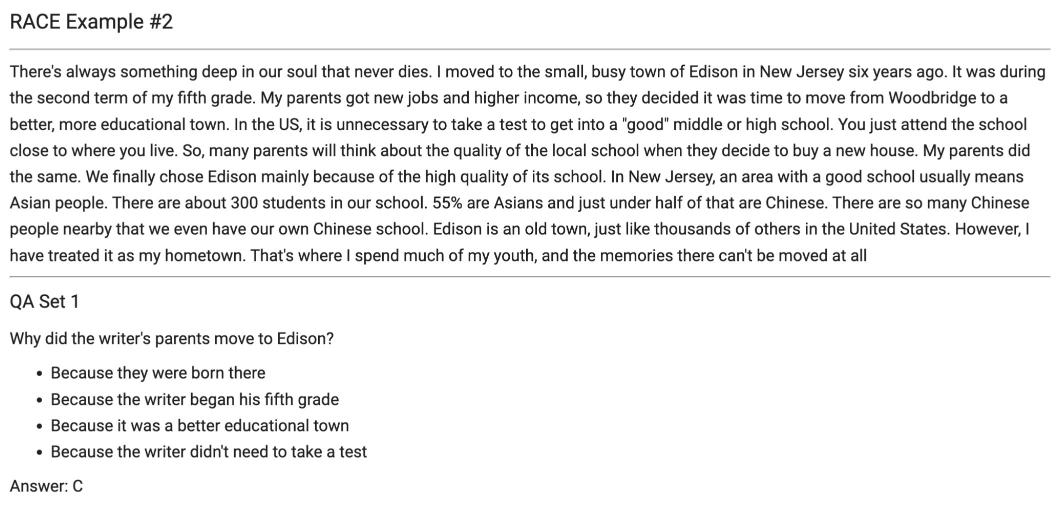
Figure 4.1: Sample from RACE

4.2. RACE

The ReAding Comprehension Dataset From Examinations, RACE for short, is a dataset for benchmark evaluation for the task of reading comprehension. It was introduced by researchers of the Language Technologies Institute at Carnegie Mellon University. Its texts, questions and answers are sourced from exams intended for Chinese students of English in the age range between 12 to 18(Lai et al., 2017). It consists of around 28,000 texts (that the authors refer to as passages) and 200,000 questions generated by human English teachers, and covers a variety of topics intended originally for evaluating human students. One important aspect of RACE that the authors note as the reason for introducing it is that it has a higher proportion of questions that require reasoning compared to previous collections. Another interesting aspect is that candidate options in RACE are human generated sentences which may not appear in the original passage(Lai et al., 2017), which is especially interesting for our task at hand. Finally, he dataset is also designed to cover different styles and domains of knowledge.

It is subdivided in two datasets, middle and high, for those questions intended for middle school and high school students. In our case we will be focusing on the middle school ones.

However, it is of special notice that since the articles and questions are selected and designed to test Chinese students learning English as a foreign



RACE Example #2

There's always something deep in our soul that never dies. I moved to the small, busy town of Edison in New Jersey six years ago. It was during the second term of my fifth grade. My parents got new jobs and higher income, so they decided it was time to move from Woodbridge to a better, more educational town. In the US, it is unnecessary to take a test to get into a "good" middle or high school. You just attend the school close to where you live. So, many parents will think about the quality of the local school when they decide to buy a new house. My parents did the same. We finally chose Edison mainly because of the high quality of its school. In New Jersey, an area with a good school usually means Asian people. There are about 300 students in our school. 55% are Asians and just under half of that are Chinese. There are so many Chinese people nearby that we even have our own Chinese school. Edison is an old town, just like thousands of others in the United States. However, I have treated it as my hometown. That's where I spend much of my youth, and the memories there can't be moved at all

QA Set 1

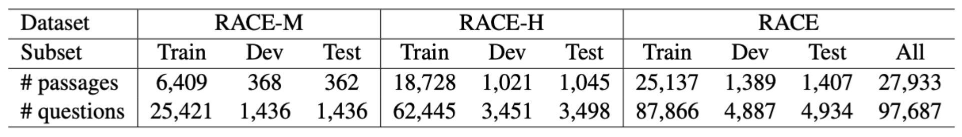
Why did the writer's parents move to Edison?

- Because they were born there
- Because the writer began his fifth grade
- Because it was a better educational town
- Because the writer didn't need to take a test

Answer: C

Figure 4.2: Sample from RACE

language, the vocabulary size and the complexity we might expect out of the language constructs are simpler than those we would find in news or Wikipedia articles commonly present in other related datasets.(Lai et al., 2017) We can see the exact composition of the RACE dataset in 4.3.



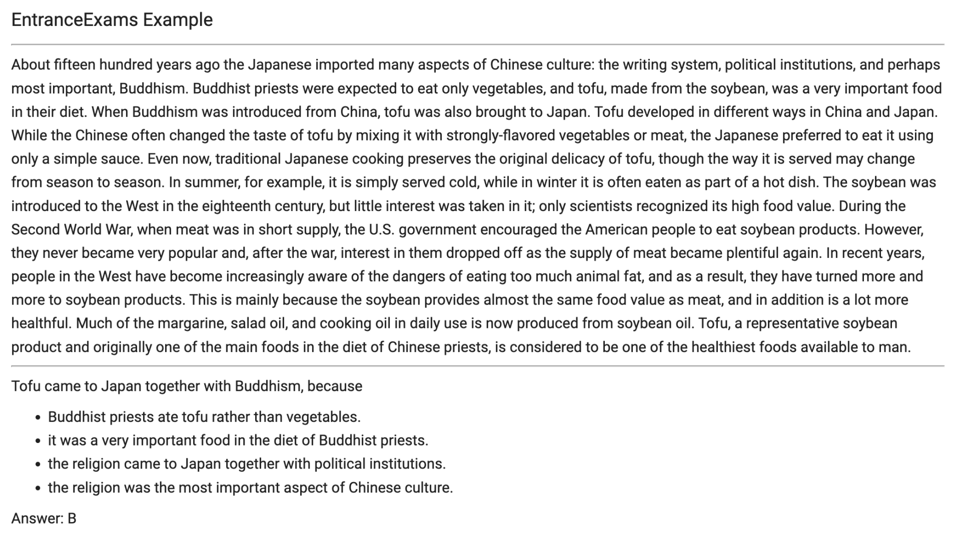
Dataset	RACE-M			RACE-H			RACE			
Subset	Train	Dev	Test	Train	Dev	Test	Train	Dev	Test	All
# passages	6,409	368	362	18,728	1,021	1,045	25,137	1,389	1,407	27,933
# questions	25,421	1,436	1,436	62,445	3,451	3,498	87,866	4,887	4,934	97,687

Figure 4.3: RACE dataset composition

The kinds of questions that are found within this dataset have been classified by the authors into several categories.(Lai et al., 2017) The first of those is *Word matching*, where the question exactly matches a part of the article. The answer is self-evident. When it is almost the same but with different wording it's called *paraphrasing*, it will be called *Single-sentence reasoning* if it can be inferred from a single sentence of the article, or *Multi-sentence reasoning* if it requires more. Finally, a number of them have been flagged as Ambiguous by human readers, but not so many to be statistically significant(Lai et al., 2017).

There are some questions where external, real-world knowledge is required, but according to the authors it should almost entirely be comprised of arithmetic knowledge of the basic kind.

4.3. EntranceExams


EntranceExams Example

About fifteen hundred years ago the Japanese imported many aspects of Chinese culture: the writing system, political institutions, and perhaps most important, Buddhism. Buddhist priests were expected to eat only vegetables, and tofu, made from the soybean, was a very important food in their diet. When Buddhism was introduced from China, tofu was also brought to Japan. Tofu developed in different ways in China and Japan. While the Chinese often changed the taste of tofu by mixing it with strongly-flavored vegetables or meat, the Japanese preferred to eat it using only a simple sauce. Even now, traditional Japanese cooking preserves the original delicacy of tofu, though the way it is served may change from season to season. In summer, for example, it is simply served cold, while in winter it is often eaten as part of a hot dish. The soybean was introduced to the West in the eighteenth century, but little interest was taken in it; only scientists recognized its high food value. During the Second World War, when meat was in short supply, the U.S. government encouraged the American people to eat soybean products. However, they never became very popular and, after the war, interest in them dropped off as the supply of meat became plentiful again. In recent years, people in the West have become increasingly aware of the dangers of eating too much animal fat, and as a result, they have turned more and more to soybean products. This is mainly because the soybean provides almost the same food value as meat, and in addition is a lot more healthful. Much of the margarine, salad oil, and cooking oil in daily use is now produced from soybean oil. Tofu, a representative soybean product and originally one of the main foods in the diet of Chinese priests, is considered to be one of the healthiest foods available to man.

Tofu came to Japan together with Buddhism, because

- Buddhist priests ate tofu rather than vegetables.
- it was a very important food in the diet of Buddhist priests.
- the religion came to Japan together with political institutions.
- the religion was the most important aspect of Chinese culture.

Answer: B

Figure 4.4: Sample from EntranceExams

Our second collection, the Japanese University Entrance Exams or *EntraceExams*, *EE* for short, consist on a series of questions with several levels of complexity and test a widee range of competences, although we'll be focusing on the ones intended for reading comprehension in English. In this case we'll be trying to evaluate our systems with conditions similar to which humans are evaluated to enter the University. This dataset has an associated challenge that has been carried throughout a number of years, so our EntranceExams sample will be the result of the aggregation of the datasets spanning the years 2014 to 2016 for the associated *Entrance Exams* task at NTCIR, where the exams are created by the Japanese National Center for University Admissions Tests.

Authors point out that an important difference from previous exercises is that question types are not restricted, so we encounter the same variety we do in the RACE dataset.

In our case, we will be making use of a joint collection resulting from the 2014, 2015 and 2016 English exams.

In total, they make up to 289 samples, each composed of a tuple of *Text-Question-Answer-Distractors*.

As a final note, we transform this data to match the dataset structure of the RACE sample we selected, and make sure the answers are equally distributed between the different options, so as not to introduce a bias in the process of training the model that might influence the scores.

Chapter 5

Choice of Technologies

Once we've chosen our data, in this chapter we will go over our choice of technologies for the particular implementation we'll be using for the experiments.

5.1. Question Answering Model

Our criteria followed for the choice of a *Question Answering* model aimed to select a series of models that had proven learning capability for the task at hand over the already selected dataset. If the model is able to learn a great deal from the dataset and offer good results, it follows that there is a large range of learning where we can experiment with samples of different sizes and be able to see the improvements with suficient statisticall significance. If this weren't the case, the gains from the expanded dataset could be too small to be significantly measured.

Another important aspect to take into account is if the model is able to learn enough with samples not so big as the whole dataset, that is, some significant gains are improved at smaller sample levels. This will allow us to cater to a wider range of experiments with smaller samples, to test out the limits of our data generation approach, that is, how far we can get with only a very limited number of small samples.

These two coupled with a general need for smaller resource requirements, and fast enough to allow for iteration of very different experiments. If our model were to be very slow, this would greatly impact our capabilities for experimentation.

We would also hope for a good out-of-the-box behaviour on the dataset, without having to tune the parameters too much. This made us focus on modern approaches like those of *transfer learning*, mainly those based in *BERT*.

To choose one kind of transfer learning model over the others, we performed the first part of our experiment, the benchmark for a small sample of real data, and compared the results and the time involved to get them. This process can be read about in detail in the next chapter.

5.2. Question-Answer-Distractor Generation Process

The first requirement for this model would be that it wouldn't be trained on the *RACE* collection, so as to avoid leakage of information.

Ideally it would have to allow us to generate diverse Question-Answer pairs, and be able to extract as many questions as needed from each text. This lead us to explore an *extractive* QA Generation Model. This would allow us to measure the quantity of information that we can extract from each text as the number of answers that can be selected within the text that the model can generate questions from, as well as allow us to pass passages to it and get questions for longer answers. This however poses a fallback, in the sense that our questions will be less abstract and present within the text.

Keep in mind that nothing in our theoretical approach so far limits us to only one of these models, we could mix them together to generate increased amounts of synthetic data. However, for the sake of simplicity we will be making use of only one.

As for Distractor Generation, we selected a number of strategies with varying degrees of complexity. These are developed in more detail in the next sections, but involve either selecting other question candidates, switching around words in other passages or selecting words from a valid grammatical category present within the texts.

5.3. Experiment Strategies

It is now that we introduce the concept of a *strategy* for us, that will mainly refer to a procedure chosen to generate the question-answer-distractor pairs from a given text, from the ones we've exposed in the previous section. Each strategy will generate a different *Synthetic Sample* from the same real sample from the *RACE* collection, whose size will be specified in the next chapter Baselines.

The first collection of synthetic data we'll name *Strategy I*, and will consist of selecting all Nouns from a text as answer candidates, and selecting the distractors for each question for the pool of nouns generated from said text.

In turn, *Strategy II* will consist of longer answers, selected from grammatically coherent phrases or parts of phrases within the text. For the distractors, we will use the other candidate phrases.

Finally, *Strategy III* will use a mix of nouns and other grammatical categories, but otherwise be the same as Strategy I, with short answers and distractors. Inthis strategy, the distractors need not be of the same gramatical category as the correct answer.

Without further delay, let's move to the actual experiment, starting with the baselines.

Chapter 6

Baselines

In the first phase of our experiments we will set the benchmark baselines our results will be compared against. These will serve as ceilings and floors of our experiments'results, and allow us to grasp a sense of how well our tests are doing versus the best and worst case scenarios.

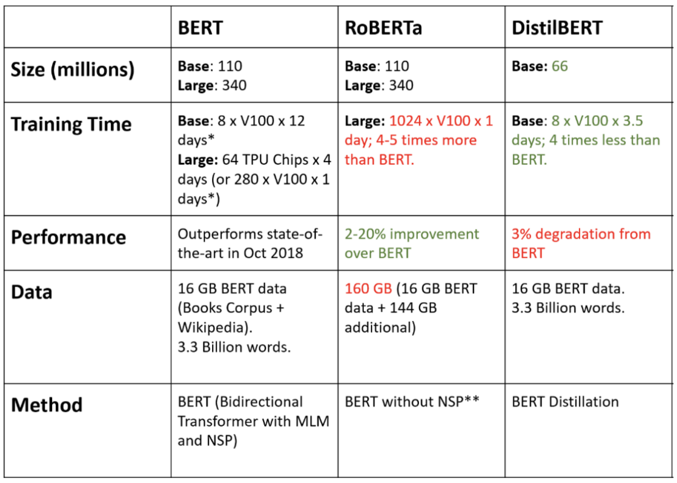
We'll be testing a number of pretrained models to see if they show any big advantages for later use. What we'll be looking for here is a balance between the model gaining enough accuracy through finetuning so the experiment can be carried with a degree of confidence and utility. Another requirement would be that the finetuning process doesn't take too long for our desired samples, since our experiments will present a certain degree of iteration, that is, they do not take too long to train a single sample.

The main differences between the models we'll be testing are explained in the following table6.1, and will be detailed later on.

In parallel to the decision of model, we have one remaining decision influenced by the choice of model, the one having to do with the dataset sizes to be used in this experiment.

At this point we'll be experimenting with different sample sizes and finding out that 10,000 samples are enough to carry out our experiment with confidence without taking too much to finetune the models, while at the same time providing enough texts so our synthetic data can grow up to a big enough size.

First we'll be taking a look at the baselines without finetuning for different variants of BERT. Given our previous experience, we expect the model to require some finetuning to be useful for this task, as it does with most downstream QA tasks.



	BERT	RoBERTa	DistilBERT
Size (millions)	**Base**: 110 **Large**: 340	**Base**: 110 **Large**: 340	**Base:** 66
Training Time	**Base**: 8 x V100 x 12 days* **Large:** 64 TPU Chips x 4 days (or 280 x V100 x 1 days*)	**Large:** 1024 x V100 x 1 day; 4-5 times more than BERT.	**Base:** 8 x V100 x 3.5 days; 4 times less than BERT.
Performance	Outperforms state-of-the-art in Oct 2018	2-20% improvement over BERT	3% degradation from BERT
Data	16 GB BERT data (Books Corpus + Wikipedia). 3.3 Billion words.	160 GB (16 GB BERT data + 144 GB additional)	16 GB BERT data. 3.3 Billion words.
Method	BERT (Bidirectional Transformer with MLM and NSP)	BERT without NSP**	BERT Distillation

Figure 6.1: Differences between BERT types as seen here

To make sure our preprocessing of the data is indeeed correct, we'll be making use of an already-finetuned ROBERTA variant, from which we expect high accuracies if we've preprocessed our data correctly for the models.

From now on, every accuracy is given as a floating point number with four decimals of precision, and a random seed has been set on the system for replicability. We will also be calculating the scores resulting of training on the whole EE dataset.

	distilbert	bert	roberta-large	finetuned
Accuracy for RACE	0.2541	0.2572	0.2585	0.8231
Accuracy for EE	0.2553	0.2524	0.2562	0.7629

Table 6.1: Baseline Accuracies without finetuning

As expected, the accuracies around 0.25 indicate no better than random performance, since we have to pick between four options for each answer. More important than that, we see that the large finetuned model is able to test our samples, which indicates that we're preprocessing them correctly for the model, and gives us peace of mind moving on. Since no finetuning has been carried on for these intiial benchmarks, the choice of model is not yet clear.

The small differences for each model can be accounted for the variety in

Accuracies	distilbert-base	bert-base-uncased	roberta-large
Accuracy for RACE	0.4451	0.4821	0.5232
Accuracy for EE	0.4096	0.4827	0.5328

Table 6.2: Baseline Accuracies training with 10,000 (random) samples of RACE

size and parameters. With BERT base we're making use of 110 million parameters, ROBERTA large accounts for 340 million parameters and Distilbert 66 million parameters in the base version. In the case of Distilbert and BERT they have been trained with 16GB of data, and ROBERTA 160GB of data. Of course, the base BERT model is a form of Bidirectional Transformer from which the other ones are distilled.

As for the sample size that we'll be using to finetune, we have the previous reference of finetuning with the whole dataset yielding accuracies of 0.82. What we're looking for here is to detect a bracket of increment in the sample that posseses a large enough bump in accuracies so that the results of generating synthetic data can be scaled within, that is, we can observe an smaller increment with some statistical significance.

To reach this end, we start with small samples of 1000 and increment them in brackets of 1000, until we get a baseline that we judge sufficient enough for the model to have learned some data, around 0.5.

After trying with incremental sizes, we arrive at a comfortable size of 10,000, which also allows for a somewhat managable finetuning time, is fast enough to generate a sample of that size in synthetic data, and a scaling of up to 20,000 yields a high enough accuracy to appreciate differences within.

Chapter 7

Synthetic Data Generation

Once our baselines have been stablished we move on to outline the process we'll follow for the generation of our data. This new data, of the same form of the real RACE and EE samples presented in Chapter 4 but generated from those through our own process, will receive the name of *synthetic* data.

This process will start by taking a text from a sample of the RACE or EE collection.

From this text some answer candidate will be selected following one of the strategies we presented earlier(5.3). After this candidate is selected, we will need to generate the corresponding question and some suitable distractors for it. The question will be generated through the *T5* model, and for the distractors a series of different strategies might be followed.

At this point it's worth mentioning that suitable distractors for humans must pass a must a rather high threshold compared to what we'll be doing here, but for the sake of generating them automatically and fast we'll be giving up on that for the moment.

Our intuition here said that even if these distractors would not be so at all for a human, for a model they might have value. An aspect that we'll be incorporating into our strategy that is taken from human-made distractors is to include distractors that are present throughout the text.

We think that this will help the model identify some parts within the texts better, instead of just checking whether the distractor is present, although it's difficult to test this hypothesis in a somewhat robust manner other than trial and error of generated samples.

These constraints identified, we can nonetheless undertake a procedure that goes as follows.

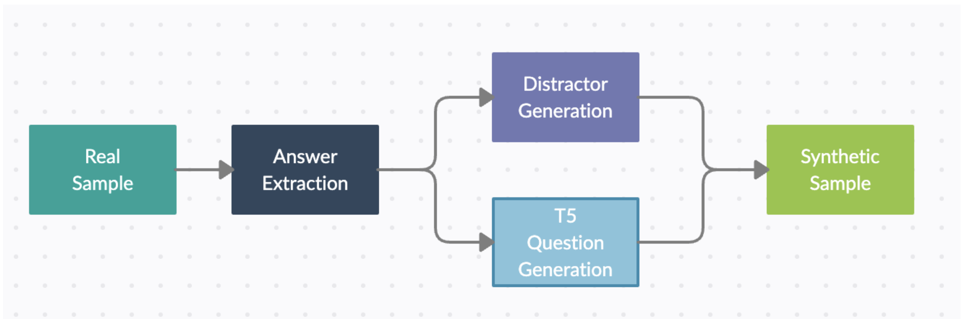


Figure 7.1: Procedure to generate a Synthetic Sample

7.1. Answer Extraction

The process of extracting the candidate answer from the text starts by parsing the text with *Spacy* and generating an structured document that contains the grammatical tags as well as the subdivision by phrases and the named entities recognized within.

Once that is done, we have followed a number of different strategies to choose the answer candidates that will impact in turn the question generation and the future generation of distractors.

On our most simple strategy, we select the *Nouns* as candidates. Each text will have as many candidates as unique nouns within it. Once they have been selected, we make sure to shuffle them so the order they appear in doesn't introduce a bias in the generation of the questions. We have tried some variants of this strategy like selecting different categories that might make sense like adjectives also.

In another strategy, we select phrases and parts of them within the text. These are detected as such by Spacy so they are grammatically meaningful, making sure there are no phrases cut in the middle or other analogous cases.

7.2. Question Generation

No matter the chosen strategy, each of the candidates is then passed as input along with the text to the *T5* model, that as we had seen is a sequence-to-sequence question generator which takes an answer and context as an input, and generates a question as an output.

This method for question generation will remain the same for the different extracted answers. We made sure that the version of T5 we were using had been finetuned on a collection other than RACE so as to avoid leaking information from the test set into our process.

7.3. Distractor Generation

For each of these Answer-Question generation methods, we'll be testing out several distractor strategies.

They will mainly be *extractive*, that is, we'll extract the distractors from the text itself, within the candidates for other questions.

For answers that are of a longer form, like phrases, we'll also try changing some words around to see if they provide better or worse distractors than those.

Once all of this has been done, we have generated a number of samples that will allow us to test how different forms of questions, answers and distractor impact the improvement this data will make to our models.

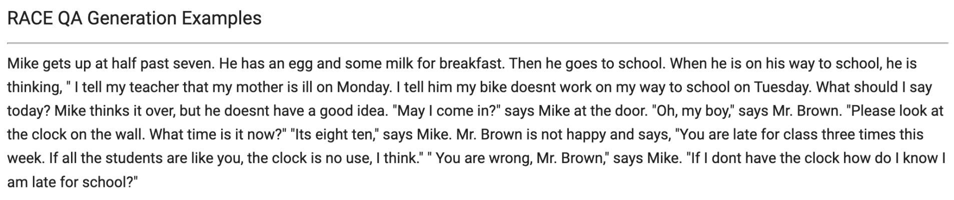
Chapter 8

Validating Synthetic Data

After going through the process of generating our *synthetic* data, we will take some time to illustrate some of the generated samples as well as talk about how we've made sure that these samples actually provide value.

To this end there's a number of checks to be made.

First, we've done a visual analysis of all the samples to make sure that they'd make sense to a human. For example, let's take a look at the following text present in the RACE *middle* collection



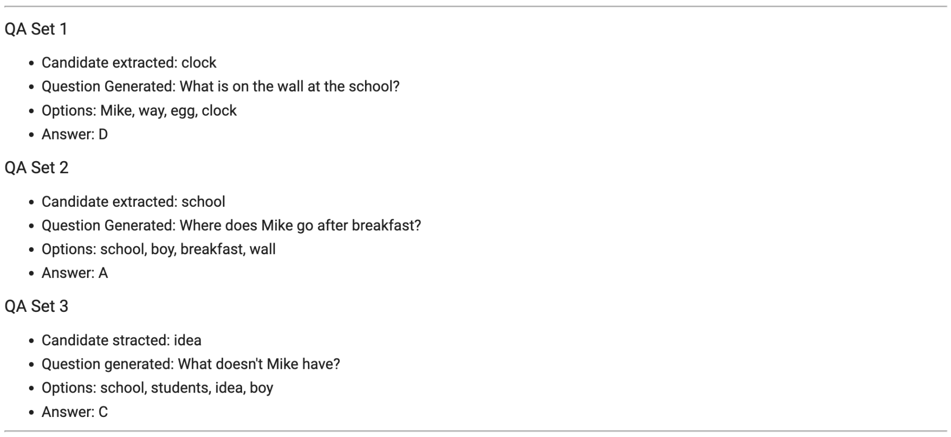
RACE QA Generation Examples

Mike gets up at half past seven. He has an egg and some milk for breakfast. Then he goes to school. When he is on his way to school, he is thinking, " I tell my teacher that my mother is ill on Monday. I tell him my bike doesnt work on my way to school on Tuesday. What should I say today? Mike thinks it over, but he doesnt have a good idea. "May I come in?" says Mike at the door. "Oh, my boy," says Mr. Brown. "Please look at the clock on the wall. What time is it now?" "Its eight ten," says Mike. Mr. Brown is not happy and says, "You are late for class three times this week. If all the students are like you, the clock is no use, I think." " You are wrong, Mr. Brown," says Mike. "If I dont have the clock how do I know I am late for school?"

Figure 8.1: Text 1, from the RACE collection

This is a typical text for the collection, not too long and not too short either. This text has been parsed for T5 and some candidates to be answers have been selected according to some strategy. The result of this process is something like the following, for a strategy that would select *nouns* as candidates and selecting the first three pairs generated, that is, our *Strategy* I(5.3).

As we can see, to a human reader these questions have an obvious answer that can be reached without reading the text itself in most cases. However, to a computer system who doesn't know that humans cannot just go to *wall*


QA Set 1

- Candidate extracted: clock
- Question Generated: What is on the wall at the school?
- Options: Mike, way, egg, clock
- Answer: D

QA Set 2

- Candidate extracted: school
- Question Generated: Where does Mike go after breakfast?
- Options: school, boy, breakfast, wall
- Answer: A

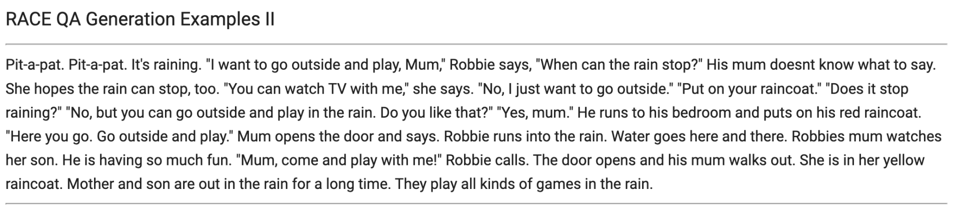
QA Set 3

- Candidate stracted: idea
- Question generated: What doesn't Mike have?
- Options: school, students, idea, boy
- Answer: C

Figure 8.2: QA pairs generated for Text 1

after breakfast or that *Mike* is not likely to be a thing that could be on the wall this might provide good enough information.

Let's now move on to take a look at the results of another strategy, *Strategy II*(5.3)this one taking phrases or chunks of them from the text as answers.


RACE QA Generation Examples II

Pit-a-pat. Pit-a-pat. It's raining. "I want to go outside and play, Mum," Robbie says, "When can the rain stop?" His mum doesnt know what to say. She hopes the rain can stop, too. "You can watch TV with me," she says. "No, I just want to go outside." "Put on your raincoat." "Does it stop raining?" "No, but you can go outside and play in the rain. Do you like that?" "Yes, mum." He runs to his bedroom and puts on his red raincoat. "Here you go. Go outside and play." Mum opens the door and says. Robbie runs into the rain. Water goes here and there. Robbies mum watches her son. He is having so much fun. "Mum, come and play with me!" Robbie calls. The door opens and his mum walks out. She is in her yellow raincoat. Mother and son are out in the rain for a long time. They play all kinds of games in the rain.

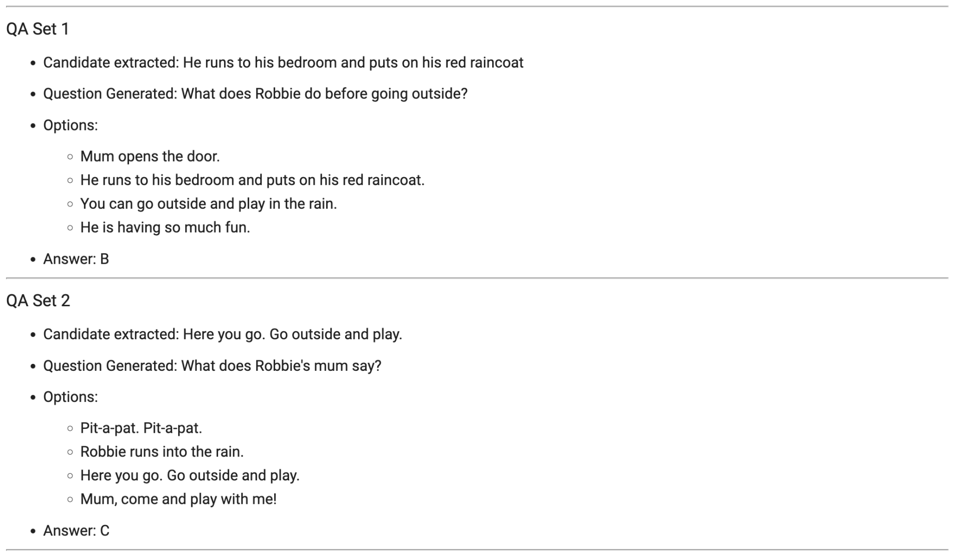
Figure 8.3: Text 2, from the RACE collection

Text 2 is another standard text as we can see, a short story without much special compared to other texts from RACE.

The collection that we've generated from this presents similar properties as the last one as we can see. A considerable number of these questions wouldn't even force a human to read the text, but they nonetheless present a good aid to computer systems that lack the human context.

This is the case too with the collections generated for the EntranceExams texts, that as we have seen have a lot in common with those of RACE.

Now, after the initial visual exploration of the different strategies follo-

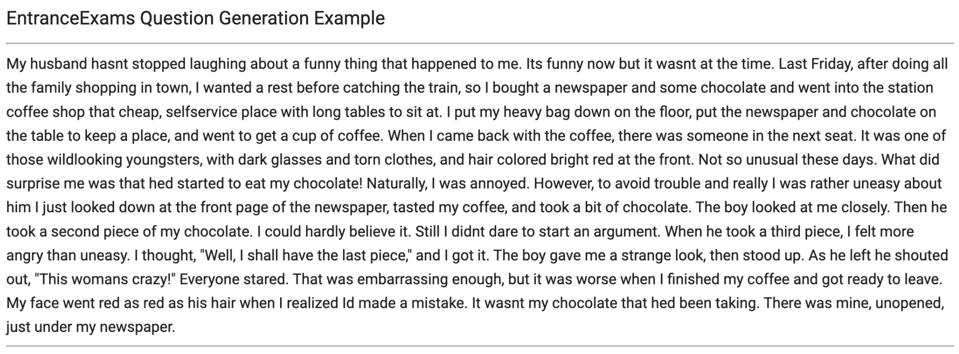

QA Set 1

- Candidate extracted: He runs to his bedroom and puts on his red raincoat
- Question Generated: What does Robbie do before going outside?
- Options:
  - Mum opens the door.
  - He runs to his bedroom and puts on his red raincoat.
  - You can go outside and play in the rain.
  - He is having so much fun.
- Answer: B

QA Set 2

- Candidate extracted: Here you go. Go outside and play.
- Question Generated: What does Robbie's mum say?
- Options:
  - Pit-a-pat. Pit-a-pat.
  - Robbie runs into the rain.
  - Here you go. Go outside and play.
  - Mum, come and play with me!
- Answer: C

Figure 8.4: QA pairs generated for Text 2


EntranceExams Question Generation Example

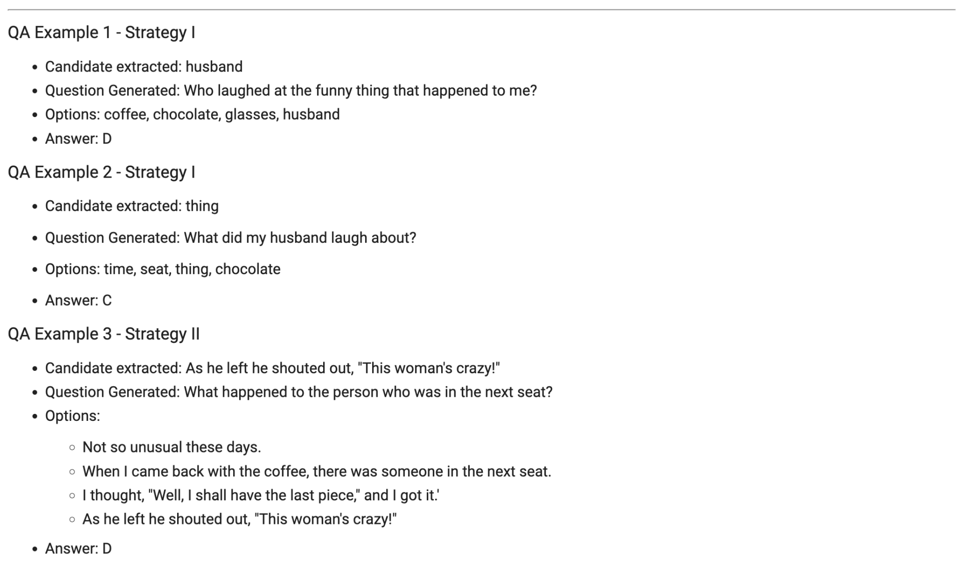
My husband hasnt stopped laughing about a funny thing that happened to me. Its funny now but it wasnt at the time. Last Friday, after doing all the family shopping in town, I wanted a rest before catching the train, so I bought a newspaper and some chocolate and went into the station coffee shop that cheap, selfservice place with long tables to sit at. I put my heavy bag down on the floor, put the newspaper and chocolate on the table to keep a place, and went to get a cup of coffee. When I came back with the coffee, there was someone in the next seat. It was one of those wildlooking youngsters, with dark glasses and torn clothes, and hair colored bright red at the front. Not so unusual these days. What did surprise me was that hed started to eat my chocolate! Naturally, I was annoyed. However, to avoid trouble and really I was rather uneasy about him I just looked down at the front page of the newspaper, tasted my coffee, and took a bit of chocolate. The boy looked at me closely. Then he took a second piece of my chocolate. I could hardly believe it. Still I didnt dare to start an argument. When he took a third piece, I felt more angry than uneasy. I thought, "Well, I shall have the last piece," and I got it. The boy gave me a strange look, then stood up. As he left he shouted out, "This womans crazy!" Everyone stared. That was embarrassing enough, but it was worse when I finished my coffee and got ready to leave. My face went red as red as his hair when I realized Id made a mistake. It wasnt my chocolate that hed been taking. There was mine, unopened, just under my newspaper.

Figure 8.5: Text 3, from the EntranceExams collection

wed for samples and distractor generation has been done, and we've visually checked that the samples therein are at least somewhat readable and might be useful, it's time to make further evaluations of their predictability.

Initially, what we're trying to see is if they hold any predictive value at all, what we called the stage of *validation* in our initial proposal.

We've already stablished the baselines in the previous chapter, and although we don't expect these synthetic samples to fare better than real data, one would expect that if we use *only synthetic data* to finetune our models we would end up somewhere between the baseline, no-finetuned accuracy and


QA Example 1 - Strategy I

- Candidate extracted: husband
- Question Generated: Who laughed at the funny thing that happened to me?
- Options: coffee, chocolate, glasses, husband
- Answer: D

QA Example 2 - Strategy I

- Candidate extracted: thing
- Question Generated: What did my husband laugh about?
- Options: time, seat, thing, chocolate
- Answer: C

QA Example 3 - Strategy II

- Candidate extracted: As he left he shouted out, "This woman's crazy!"
- Question Generated: What happened to the person who was in the next seat?
- Options:
    - Not so unusual these days.
    - When I came back with the coffee, there was someone in the next seat.
    - I thought, "Well, I shall have the last piece," and I got it.'
    - As he left he shouted out, "This woman's crazy!"
- Answer: D

the real data.

Finally, we would expect that samples generated from one collection fare better in that collection.

At this point we found the first obstacle, because we detected the EntranceExams collection didn't contain enough texts to generate a meaningful medium-sized synthetic data collection.

This problem is explored in Limits of Synthetic Data Generation in detail, but for now we'll focus on three different sets of 10,000 synthetic samples each, all generated from the same RACE collection of 1000 texts.

Other little variations were tested and iterated through, but in the end yielded similar performance. These iterations included small changes like mixing phrases and different grammatical categories as distractors and answers or giving priority to Named Entities as distractors and candidates. In the end, we opted for these three to illustrate the process and follow through with the analysis.

	Strategy I	Strategy II	Strategy III	Real Data
On RACE test set	0.3742	0.3834	0.3796	0.5362
On EE	0.3367	0.3452	0.3474	0.5139

Table 8.1: Accuracies for BERT-base-uncased finetuning with 10,000 synthetic data

As we can see, the three of them present similar predictability potential, improving the .25 baseline without finetuning, and remaining below the performance of real data as expected. Let's move on to see if we can combine them with real data to improve model performance.

Chapter 9

Evaluating the Use of Synthetic Data

Since we saw that the different strategies followed for synthetic data generation yielded similar predictability, we'll move on and use them in combination with real data to see if one of them proves better.

The main application of our research will be to improve model performance, so the most valuable results will come from comparing finetuning the model with an initial sample of real data to finetuning the model with real data expanded with synthetic data *generated from said data*, that is, without introducing synthetic data generated from texts not contained within said data. This collection will be called *Expanded w/ Synthetic Data.*

However, for the sake of experimenting, we'll also test the results of generating this synthetic data from previously unseen texts. We expect these to have better results than the others, since they have new data, but not quite as good as the real data equivalent. This we will call *Expanded w/ new Synthetic Data*, and for each table will be generated with the correspondant strategy.

In both cases we'll be working with samples of size 10,000 from RACE, which is equivalent to 1000 different texts. This is a good number because all of the RACE texts admit at least 5 questions, so we make sure that we can generate enough questions from each of them, while keeping the dataset sizes small enough to run in our local environment but at the same time large enough to demonstrate decent results and be able to see the difference between samples, as well as demonstrating use in a scarcity context.

We provide two baselines: one taken from finetuning BERT with the base 10,000 samples from RACE, and one taken from finetuning BERT with 20000 samples from RACE. We expect these to act as the theoretical minimum and maximum to reach with these texts.

In the process of experimenting with the data we found some natural limits to the size of the synthetic samples we were able to generate from a given real sample, which will be explored further in the next chapter.

For each experiment we will also test if this learning process is also improved while being tested on the *EntranceExams* dataset.

9.1. Experiment I: Short Answers

The first experiment is with synthetic data generated with nouns as answers and distractors, that we called *Strategy I*(5.3). We will do it for both previously seen texts, *Expanded w/ Synthetic Data*, and new texts *Expanded w/ new Synthetic Data.*

	Finetuning with 10,000 Real Data	Expanded w/ Synthetic Data	Expanded w/ new Synthetic Data	Expanded w/ Real Data
On RACE finetuned with 10,000 RACE samples	0.5362	0.5853	0.6258	0.6731
On EE finetuned with 10,000 RACE samples	0.5139	0.5571	0.5924	0.6428

Table 9.1: Accuracies for BERT-base finetuned with 10,000 RACE samples

We can see that some noticeable improvements have been detected with the use of synthetic data, although it is of the most simple form we have.

In the case of working with data from a new dataset, it seems that the effect of knowing the texts that it's going to be exposed to beforehand can help the system perform better in the new questions, and this is a constant that we'll see in the next experiments aswell.

Let's move on to other strategies with longer Answers and Distractors.

9.2. Experiment II: Longer Answers

For our second experiment we'll use samples generated with phrases as answers and distractors, that is, *Strategy II*(5.3).

	Finetuning with 10,000 Real Data	Expanded w/ Synthetic Data	Expanded w/ new Synthetic Data	Expanded w/ Real Data
On RACE finetuned with 10,000 RACE samples	0.5362	0.5712	0.6058	0.6731
On EE finetuned with 10,000 RACE samples	0.5139	0.5371	0.5724	0.6428

Table 9.2: Accuracies for BERT-basefinetuned with 10,000 RACE samples

9.3. Experiment III: Different Grammatical Categories

And now with short answers of different grammatical categories as answers and distractors, that is, following *Strategy III*(5.3).

	Finetuning with 10,000 Real Data	Expanded w/ Synthetic Data	Expanded w/ new Synthetic Data	Expanded w/ Real Data
On RACE finetuned with 10,000 RACE samples	0.5362	0.6053	0.6213	0.6731
On EE finetuned with 10,000 RACE samples	0.5139	0.5721	0.5904	0.6428

Table 9.3: Accuracies for BERT-base finetuned with 10,000 RACE samples

9.4. Experiment IV: Distractor Variation

As a final experiment we'll use a different method for the distractor generation in one of our strategies. We'll be modifying *Strategy II*(5.3), changing the words in our answer phrase for others of the same grammatical category, that is, introducing variance within the words in the answer instead of changing the phrase entirely. This is achieved by choosing words of the same grammatical category found within the text, changing all the nouns within each of them.

	Finetuning with 10,000 Real Data	Expanded w/ Synthetic Data	Expanded w/ new Synthetic Data	Expanded w/ Real Data
On RACE finetuned with 10,000 RACE samples	0.5362	0.5512	0.5818	0.6731
On EE finetuned with 10,000 RACE samples	0.5139	0.5295	0.5437	0.6428

Table 9.4: Accuracies for BERT-basefinetuned with 10,000 RACE samples

As we see the accuracy achieved is lower, maybe due to the answer now having the property of being the only text present literally in the text, and thus being easier to spot.

9.5. Conclusions

As we can see, the four experiments yield promising results. Small differences within the strategies can be found, although not significant enough to prove any of the three strategies a definite best for the general case. These experiments have demonstrated that synthetic data can indeed be used to improve model's performance, and that following even simple fast and automated procedures for distractor generation can be useful and improve performance.

In all experiments, the result of making these synthetic questions from new texts that the system was then to be evaluated on gave, as expected, better results overall. This might be useful for certain systems where answering new answers doesn't need to be immediate, and so a finetuning of the system with these synthetic questions can be carried on before answering the new questions. If our purpose is to produce a finished system that produces answers quickly, the approach in which we generate the synthetic data from the original train sample should be preferred.

These types of synthetic data might not present the best use for human use cases, but nonetheless are cheap and fast to make and give our models better performance.

Improving distractors so these questions and answers however might prove a different task altogether, since what is a good distractor for humans is completely different from which distractors might be good to use in the context of models.

It is also worth noting that the improvement in accuracy was also transfered to the results of testing on the *EntranceExams* dataset.

For now, we'll explore the limits of this approach we mentioned earlier in the next chapter as well as go into detail on some of the problems revealed by trying to generate relatively large collections with extremely small numbers of texts like those of EntranceExams.

Chapter 10

Limits of Synthetic Data Generation

Although as we have seen the synthetic data we've generated has aided in improving our models, there are some limits to this approach that we found in the previous experiments. We will explore some of them to see if more insights can be gained.

First and foremost there is a limited amount of information contained within each text, and in extension within the training sample we have access to.

Even with a wide array of strategies selecting the correct answer candidates, if they are to be found within the text, that is, following an *extractive* approach, we are limited to at most every combination of consecutive words within the text, with the additional requirement that they ought to make grammatical sense in isolation.

This theoretical limit to the extractive approach to question generation also applies to distractor generation, since there are only a few grammatically valid ones within the text.

It is also intuitive to us that if we were to simply replace different distractors in a synthetic sample and expand our sample in that way, the gains in predictibility would be less than with a newly made sample, but that would have to be investigated in future work.

In this section however we will explore the limits of the methods proposed within this work, to show that if a sample is small enough the quality of the data generated won't be sufficient to improve performance, and can even impact performance negatively in some cases.

10.1. Expanding Small Samples

The first of the experiments carried within this section has synthetic data generated with the *Strategy I* approach(5.3).

In the previous chapter our experiments measured performance on our small EE dataset, where the train split of which has in total no more than 30 unique texts, with several questions for each of them. Let's see how it behaves when we expand the sample through our previous means to larger sizes.

	Not Expanded	Exp w/ 200 synthetic	Exp w/ 300 Synthetic	Exp w/ 1000 Synthetic
On EE test subset	0.23	0.22	0.23	0.21

Table 10.1: Accuracies for BERT-base-uncased finetuning with data generated from EE - train

In this case we're seeing the model obviously overfitting to the train set in the overtuning process, seeing how the performance actually *decreases* relative to baseline.

This is a minimal example of a situation we'd encounter everytime we overproduce synthetic texts from a smaller source, to the point where no new information is being fed to the model but instead variations of the same questions with different distractors over and over, as a result of our question generation procedure being deterministic for an answe-text pair.

10.2. Beating real data

In our second experiment, our purpose is different.

Here we have a larger playground to play with, around 10,000 new data, and we're trying to beat the improvement to the model resulting from adding these 10,000 samples to the finetuning set with as many synthetic questions as we can generated from previously seen data.

Here, we've used a mix of the strategies discussed in the previous section (strategies I, II and III) to generate 20000 synthetic samples, that give a

	Not Expanded	Exp w/ 10,000 synthetic	Exp w/ 20,000 Synthetic	Exp w/ 30,000 Synthetic	Exp w/ 10,000 real
Accuracy	0.5362	0.5953	0.6552	0.6741	0.6732

Table 10.2: On RACE finetuned with 10,000 RACE samples

better result than the previous baseline. So, for our particular sample, 30000 synthetic data have beat 10,000 new data.

The question however remains if this is something that can be achieved for any dataset and any sample, given that we can see that the more synthetic data we generate from a given text with our strategies the smaller the gain results.

In any case the quality of the synthetic data generated will be a function of the number of texts available, their size, and the candidate answers.

This also shows that different strategies from the previous section can actually be combined to further improve model behaviour, although it is more expensive either time if it cannot be parallelized or resources nonetheless to do so.

10.3. Playing around with the distractors

In this experiment we try to feed the model with several different versions of the same *Text-Question-Answer* tuple where only the distractors are changed for other options preselected from the pool generated from the text as in *Strategy I.* By doing this, we expand our sample greatly at a lower cost.

	Not Expanded	Exp w/ 10,000 synthetic	Exp w/ 10,000 Synthetic + 10,000 Distractor Variations	Exp w/ 10,000 Synthetic + 20000 Distractor Variations
Accuracy	0.5362	0.5953	0.6132	0.6288

Table 10.3: Accuracies for BERT-base-uncased finetuning with 10,000 real data

As we can see the gains from it are lower than with the whole expanded collection. As expected, the amount of information that the model is able to learn is not neglilible, but nonetheless smaller. The positive side to this approach, generating distractor combinations is much easier than generating new question-answer pairs, so there's this remains nonetheless a cheap way to make our synthetic data help if ever a little bit more.

It however remains another line of work to see if for some cases where better distractors can be generated automatically, or even if this is feasible for other distractor generation strategies.

Chapter 11

Conclusions and Future Work

We'll finish our exposition with a compilation of all the results we've reached throughout this work as well as where more work is needed to reach conclusions or expand the experimentsútility.

As we have seen, we have achieved a fast and robust process to generate basic Question-Answer-Distractor tuples that have been pivotal in improving our models'performance.

In our strategies these tuples have taken the form of certain short words or passages extracted from the texts themselves, and nonetheless have proven to be useful in expanding the accuracy of the models.

For our other objective though, aiding in the human case, this generation strategy proves to be lacking. Most of the samples inspectioned failed to pose a real challenge, often not even requiring that the text be read to be answered correctly. This later problem may be addressed by asking humans to simply tweak the distractors manually, already a magnitude of work less than generating the whole question, answer and distractors, or trying to detect the ones that do not make sense through other means like converting the question to an affirmative sentence and checking the probability of being the next word for each distractor.

In the context of data scarcity, we've detected some limits to the amount of valid questions we're able to generate from a given text, some of the limits at least owing to our choice of a extractive QA generation procedure. These limits could be slightly expanded through the use of different distractors, as we've seen in the last section, but we run into a hard limit nonetheless.

For this case, we've illustrated the limits of our approach with the EntranceExams collection that we weren't able to expand substantially due to the number of text present being too low to extract sufficient samples from.

As a consequence of this limit to the information we're able to extract, we detected some overfitting that led to worse accuracies when trying to generate too many synthetic samples.

We were able to show decreasing gains for our particular sample showing a logarithmic approach to a theoretical ceiling in our particular case and sample, but more work is needed to make these results more robust and detect whether this is the case with other strategies.

In this case, intuition tells us that there is a limit of information to be extracted for each text, and thus no matter the strategy followed for the Question-Answer-Distractor generation we will run into one such limit to performance gains.

It could be interesting to compare whether different strategies for Question-Answer-Distractor generation have different ceilings, thus some being particularly good for smaller samples in the context of data scarcity we've dealt with throughout this work.

BIBLIOGRAPHY

Bibliography

[Alberti et al.2019] Alberti, Chris, Daniel Andor, Emily Pitler, Jacob Devlin, y Michael Collins. 2019. Synthetic QA corpora generation with roundtrip consistency. En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, páginas 6168–6173, Florence, Italy, Julio. Association for Computational Linguistics.

[Chung, Chan, y Fan2020] Chung, Ho-Lam, Ying-Hong Chan, y Yao-Chung Fan. 2020. A bert-based distractor generation scheme with multi-tasking and negative answer training strategies. 10.

[Devlin et al.2019] Devlin, Jacob, Ming-Wei Chang, Kenton Lee, y Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. En *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, páginas 4171–4186, Minneapolis, Minnesota, Junio. Association for Computational Linguistics.

[Dhingra, Danish, y Rajagopal2018] Dhingra, Bhuwan, Danish Danish, y Dheeraj Rajagopal. 2018. Simple and effective semi-supervised question answering. En *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, páginas 582–587, New Orleans, Louisiana, Junio. Association for Computational Linguistics.

[Du y Cardie2018] Du, Xinya y Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from Wikipedia. En *Proceedings of the 56th*

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), páginas 1907–1917, Melbourne, Australia, Julio. Association for Computational Linguistics.

[Du, Shao, y Cardie2017] Du, Xinya, Junru Shao, y Claire Cardie. 2017. Learning to ask: Neural question generation for reading comprehension. En *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, páginas 1342–1352, Vancouver, Canada, Julio. Association for Computational Linguistics.

[Dua et al.2019a] Dua, Dheeru, Ananth Gottumukkala, Alon Talmor, Sameer Singh, y Matt Gardner. 2019a. Comprehensive multi-dataset evaluation of reading comprehension. En *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, páginas 147–153, Hong Kong, China, Noviembre. Association for Computational Linguistics.

[Dua et al.2019b] Dua, Dheeru, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, y Matt Gardner. 2019b. DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. En *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, páginas 2368–2378, Minneapolis, Minnesota, Junio. Association for Computational Linguistics.

[Gao et al.2018] Gao, Yifan, Lidong Bing, Piji Li, Irwin King, y Michael R. Lyu. 2018. Generating distractors for reading comprehension questions from real examinations. *CoRR*, abs/1809.02768.

[Gururangan et al.2018] Gururangan, Suchin, Swabha Swayamdipta, Omer Levy, Roy Schwartz, Samuel Bowman, y Noah A. Smith. 2018. Annotation artifacts in natural language inference data. En *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, páginas 107–112, New Orleans, Louisiana, Junio. Association for Computational Linguistics.

[Heilman y Smith2010] Heilman, Michael y Noah A. Smith. 2010. Good question! statistical ranking for question generation. En *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, páginas

609–617, Los Angeles, California, Junio. Association for Computational Linguistics.

[Kalpakchi y Boye2021] Kalpakchi, Dmytro y Johan Boye. 2021. Bert-based distractor generation for swedish reading comprehension questions using a small-scale dataset. *CoRR*, abs/2108.03973.

[Kim et al.2019] Kim, Yunsu, Petre Petrov, Pavel Petrushkov, Shahram Khadivi, y Hermann Ney. 2019. Pivot-based transfer learning for neural machine translation between non-English languages. En *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, páginas 866–876, Hong Kong, China, Noviembre. Association for Computational Linguistics.

[Kočiský et al.2018] Kočiský, Tomáš, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, y Edward Grefenstette. 2018. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328.

[Labutov, Basu, y Vanderwende2015] Labutov, Igor, Sumit Basu, y Lucy Vanderwende. 2015. Deep questions without deep understanding. En *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, páginas 889–898, Beijing, China, Julio. Association for Computational Linguistics.

[Lai et al.2017] Lai, Guokun, Qizhe Xie, Hanxiao Liu, Yiming Yang, y Eduard Hovy. 2017. Race: Large-scale reading comprehension dataset from examinations. *arXiv preprint arXiv:1704.04683*.

[Lelkes, Tran, y Yu2021] Lelkes, Ádám D., Vinh Q. Tran, y Cong Yu. 2021. Quiz-style question generation for news stories. *CoRR*, abs/2102.09094.

[Lewis, Denoyer, y Riedel2019] Lewis, Patrick, Ludovic Denoyer, y Sebastian Riedel. 2019. Unsupervised question answering by cloze translation. En *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, páginas 4896–4910, Florence, Italy, Julio. Association for Computational Linguistics.

[Lindberg et al.2013] Lindberg, David, Fred Popowich, John Nesbit, y Phil Winne. 2013. Generating natural language questions to support learning on-line. En *Proceedings of the 14th European Workshop on Natural Language Generation*, páginas 105–114, Sofia, Bulgaria, Agosto. Association for Computational Linguistics.

[Ling, An, y Hasan2017] Ling, Yuan, Yuan An, y Sadid Hasan. 2017. Improving clinical diagnosis inference through integration of structured and unstructured knowledge. En *Proceedings of the 1st Workshop on Sense, Concept and Entity Representations and their Applications*, páginas 31–36, Valencia, Spain, Abril. Association for Computational Linguistics.

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

[Rajpurkar et al.2016] Rajpurkar, Pranav, Jian Zhang, Konstantin Lopyrev, y Percy Liang. 2016. SQuAD: 100,000+ questions for machine comprehension of text. En *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, páginas 2383–2392, Austin, Texas, Noviembre. Association for Computational Linguistics.

[Rogers et al.2020] Rogers, Anna, Olga Kovaleva, Matthew Downey, y Anna Rumshisky. 2020. Getting closer to ai complete question answering: A set of prerequisite real tasks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8722–8731, Apr.

[Sachan y Xing2018] Sachan, Mrinmaya y Eric Xing. 2018. Self-training for jointly learning to ask and answer questions. En *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, páginas 629–640, New Orleans, Louisiana, Junio. Association for Computational Linguistics.

[Shoemark et al.2019] Shoemark, Philippa, Farhana Ferdousi Liza, Dong Nguyen, Scott Hale, y Barbara McGillivray. 2019. Room to Glo: A systematic comparison of semantic change detection approaches with word embeddings. En *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International*

Joint Conference on Natural Language Processing (EMNLP-IJCNLP), páginas 66–76, Hong Kong, China, Noviembre. Association for Computational Linguistics.

[Song et al.2018] Song, Linfeng, Zhiguo Wang, Wael Hamza, Yue Zhang, y Daniel Gildea. 2018. Leveraging context information for natural question generation. En *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, páginas 569–574, New Orleans, Louisiana, Junio. Association for Computational Linguistics.

[Yeh y Chen2019] Yeh, Yi-Ting y Yun-Nung Chen. 2019. QAInfomax: Learning robust question answering system by mutual information maximization. En *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, páginas 3370–3375, Hong Kong, China, Noviembre. Association for Computational Linguistics.

[Zhang, Yao, y Yan2018] Zhang, Fangfang, Jin-ge Yao, y Rui Yan. 2018. On the abstractiveness of neural document summarization. En *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, páginas 785–790, Brussels, Belgium, Octubre-Noviembre. Association for Computational Linguistics.

[Zhou et al.2017] Zhou, Qingyu, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, y M. Zhou. 2017. Neural question generation from text: A preliminary study. En *NLPCC*.