UNIVERSITÀ DI PISA

Artificial Intelligence and Data Engineering

Business and Project Management

Project documentation

TEAM MEMBERS:
Fabiano Pilia
Emanuele Tinghi
Matteo Dal Zotto

Contents

Chapter 1

Introduction

1.1 Aim of the project

The goal of the project is to create a system capable of summarizing the many reviews referring to a single product so as to be able to give an overview of it without necessarily having to scroll through them all. This also makes it possible to have a textual correspondent to the commonly present star rating system. In this way it is possible to understand the pros and cons of the product instead of having only a general representation of the quality of the object.

1.2 Dataset

The dataset used for this project contains over 3 million reviews belonging to 185774 products sold on amazon.com . To conduct the various tests it was decided to use a reduced subset, containing only 5 products and around 55k reviews from the electronics section of the site. This was done to reduce the computation time and simplify the analysis of the results. The number of review per product can be seen in figure 1.1.
The dataset is composed by 14 attributes, but for our goal we need only a subset of them:

- **product_id**, the id of the product;
- **review_body**, the body of the review;
- **review_headline**, used as label during the fine tuning in some tests;
- **star_rating**, the rating, in terms of stars from 1 to 5, of the product;
- **helpful_votes**, the number of users that voted the review as *helpful*;
- **review_date**, when the review was submitted.

The second and the third are used during the model selection, while the other two are used during the final test to evaluate the summaries. For this purpose a *derived* attribute was introduced, **review_weight**, obtained as follows:

$$w_i = \log(helpful\_votes_i + 1) * star\_rating_i$$

The log is used to dampen the number of helpful votes and the *+1* to avoid to compute the log of 0 in case of no helpful votes.

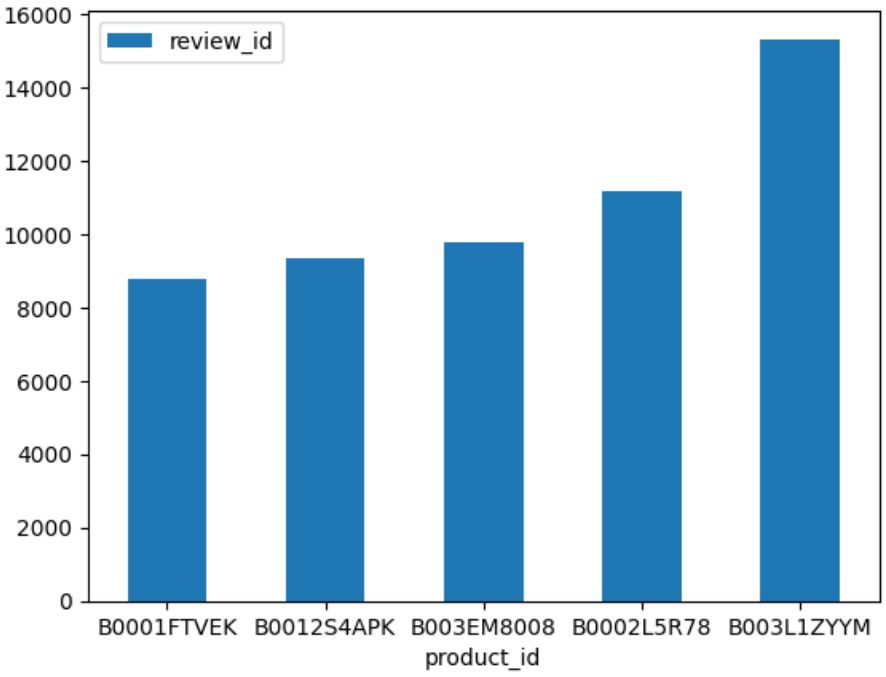

Figure 1.1: Number of reviews for each product

In the following figure1.2 we can see some entries of the resulting dataset:

	review_body	review_headline	star_rating	helpful_votes	review_date	review_weight
24	This cord works very well. A lot of other chea...	5 Stars	5	1	2015-08-30	3.465736
86	These HDMI cables are strong and durable. The...	High Quality	5	2	2015-08-27	5.493061
88	It works as intended. Have owned it for over a...	Just buy it. It works and is reliable. 1+ year...	5	1	2015-08-27	3.465736
184	Where should you always buy HDMI cables? Amazo...	Don't go to BestBuy and don't gamble with EBay...	5	1	2015-08-20	3.465736
201	We have always had great success with Amazon B...	We have always had great success with Amazon B...	5	1	2015-08-20	3.465736
...	...	...	...	...	...	...
15328	This is the HDMI cable to get right now: an HD...	A Basic Cable at a Good Price	5	297	2010-11-01	28.485467
15329	I connected my Blu Ray disc player to my HDTV ...	Easy and lightweight HDMI cable does it all.	5	5	2010-11-01	8.958797
15330	Just unpacked my cable,it is really very high ...	High quality	5	1	2010-11-01	3.465736
15331	I have purchased quite a few AmazonBasics item...	It works.	5	12	2010-11-01	12.824747
15332	This is a basic cable at a good price. I didn'...	No problems here	5	3	2010-10-29	6.931472

Figure 1.2: Some elements of the resulting dataset

1.2.1 Text preprocessing

It is then necessary to apply some preprocessing to the dataset in order to make it usable correctly. In particular the following actions are performed:

1. The text is trasformed in lowercase;
2. HTML tags are removed using the regex;
3. The text is tokenized using the word tokenizer of the Natural Language Toolkit;
4. Stopwords and punctuation marks are removed.

The tests are performed not only on the preprocessed text, but also on the full text exploiting in some cases the tokenizer embedded inside the model.

1.2.2 Additional information

In the following figure 1.3 we can see how much reviews are present for each number of tokens, in the second image we don't have considered the extremes (the reviews with 1, 2 and 614 words).

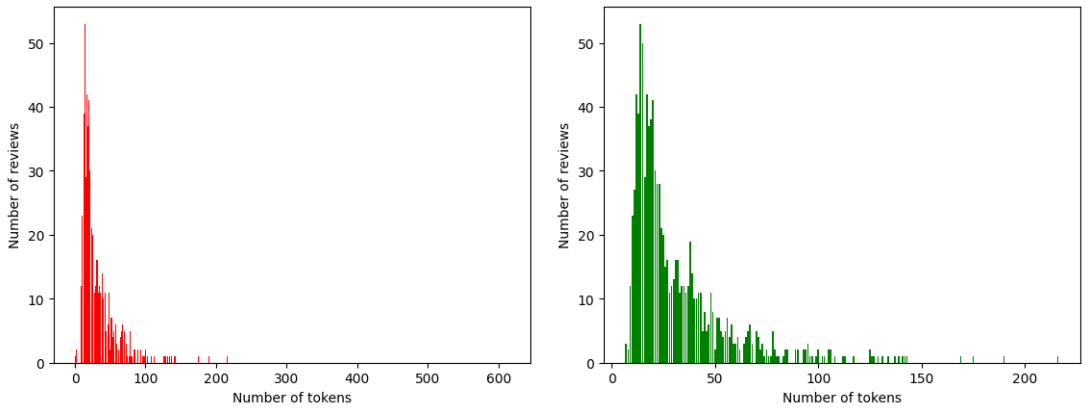

Figure 1.3: Distribution of the reviews among the number of words

We can note that the maximum is at **14** tokens, in fact **53** reviews contain 14 tokens after the preprocessing.
The average number of words (rounded) is **33**.

Chapter 2

Experiments

In this chapter we'll present the experiments and the results obtained using different **transformers** models. Before presenting the tests we need to highlight how the input to the models are built. The first type of input that we've tested was composed only by a **single review**, but performed poorly and we've adopted a batched solution. The following is the pseudocode of the batches creation:

```
def get_batches(reviews_list):
  count = 0
  batches = []
  i = 0
  num_samples = len(reviews)

  for i in range(0,num_samples):
    if count + len(reviews[i].split(" ")) < 1024:
      input = input + reviews[i] + " . "
      count = count + len(reviews[i].split(" "))
    else:
      batches.append(input)
      input = "summarize: " + reviews[i] + " . "
      count = len(reviews[i].split(" "))

  if(count != 0):
    batches.append(input)
```

We can see that it creates several strings, each one consisting obtained concatenating different reviews until it exceed the maximum number of tokens allowed (that varies model by model). The string *summarize:* is the prompt used in models like **T5**. Each string will be the input of the model at each step.

Our approach consists in different iteration of summarization as follows:

1. Compute N batches;

2. Obtain for each batch a summary;

3. Clean the text, since some models add puctuation marks or words to highlight that it's a summmary;

4. If we've more reviews than a determined threshold, start again from the point 1, but now the batches are obtained starting from the summaries obtained from the previous stage.

At the end of this process we have a list of summaries. In the final test we've taken into account also the dates.

2.1 BART

The first model used is **BART** (*Bidirectional and Auto-Regressive Transformers*), a sequence-to-sequence model trained as a denoising autoencoder, so the training data includes "corrupted" or "noisy" text, which would be mapped to clean or original text. Due to the way on which it was trained it's possible to perform different tasks on it such as *machine translation*, *question answering* or (our task) ***text summarization***.
BART follows the standard architecture of the **transformers**, already explained in the previous chapter. As we can see in the following figure 2.1 it is composed by a **Bidirectional Encoder** (**BERT**) and by an **Autoregressive Decoder** (**GPT**).

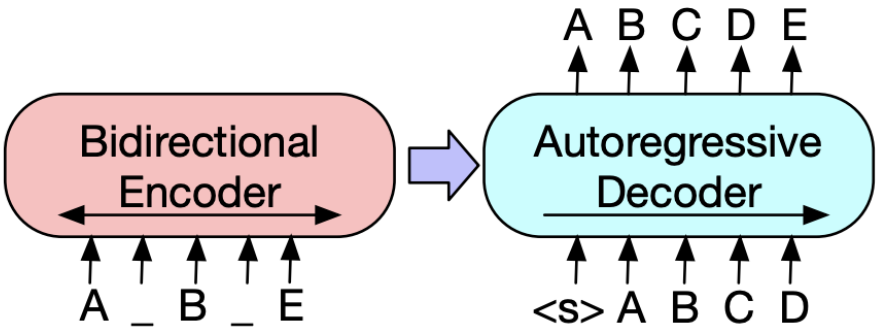

Figure 2.1: Schematic representation of the BART architecture

In figure 2.2 we can see a more detailed representation of the BART architecture: the first module uses the BERT encoder, that for every text sequence in its input, outputs an embedding vector for each token of the input sequence as well as an additional vector containing sentence-level information used by the decoder to learn information about the tokens and the sentence as a whole. The decoder used is **GPT** that starting from a special token generates iteratively the **next token** of the sequence based on what contains the current sentence and the information obtained

as output from the *encoder* module, then add the new token to the input that is given as input for the next iteration.

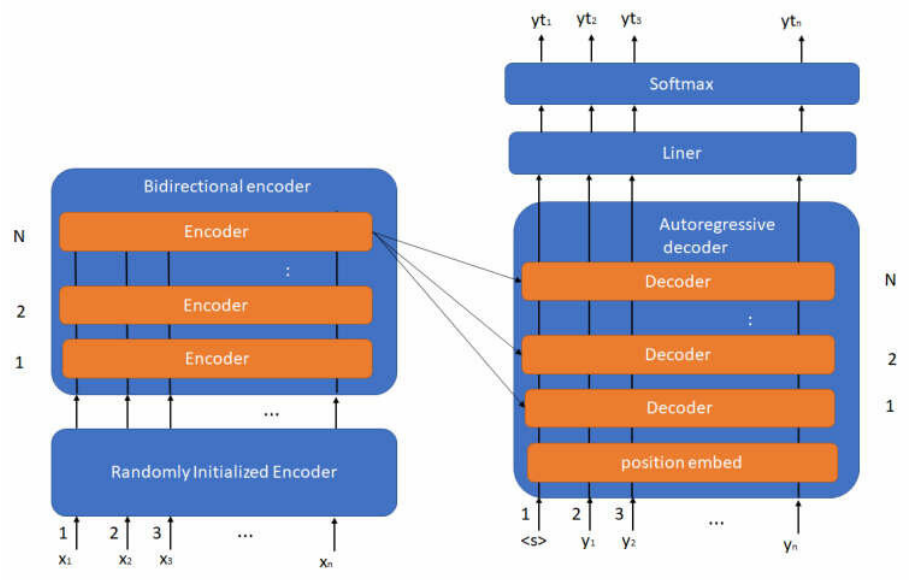

Figure 2.2: BART architecture

Depending on the complexity of the architecture we have different BART models that we can see in figure 2.3.

Model	Description	# params
`bart.base`	BART model with 6 encoder and decoder layers	140M
`bart.large`	BART model with 12 encoder and decoder layers	400M
`bart.large.mnli`	`bart.large` finetuned on `MNLI`	400M
`bart.large.cnn`	`bart.large` finetuned on `CNN-DM`	400M
`bart.large.xsum`	`bart.large` finetuned on `Xsum`	400M

Figure 2.3: BART models

2.1.1 First test: BART-base

The first test on BART has been performed using its **base** configuration and in particular we've selected a sample of 1000 reviews to tests the model (it will be the same for all the other tests).

We've compared the results using both the preprocessed text and the full text.

The result of the first stage for the first batch is the following:

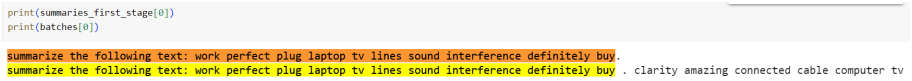

Figure 2.4: BART-base first stage summary

As we can see the behaviour of the model **extractive**, consisting in selecting among all the reviews present in the batch the one that is considered the most representative. It's clear that it isn't what we're looking for since we need a model that behaves in an **abstractive** manner, in order to capture the general thought relative to the product.
We've applied a second stage obtaining the same behavior extracting exactly the same sentence from the batch.

The second part of this test was to use the full sentence without preprocessing, but the results are exactly the same: only the sentence that the model recognize as most significant was extracted.

To solve this problem we've used increasingly complex models, in order have more abstraction.

2.1.2 Second test: BART-base with fine tuning

In the second test we've fine tuned **BART-base** using the *review_headline* as label for very few epochs.
We have used a split of **0.8** for the training set and **0.2** for the test set.
What happens in this case is that when we give a batch of reviews the model tries to associate a **label** (review headline) that best fit it. It works well, but unfortunately is not what we need, since the review headline don't represent a good summary. In fact by default the review headline is associated to the number of stars assigned to the product, therefore the most common headline are *Five Stars*, *Four Stars* and so on as we can see in the following figure 2.6.

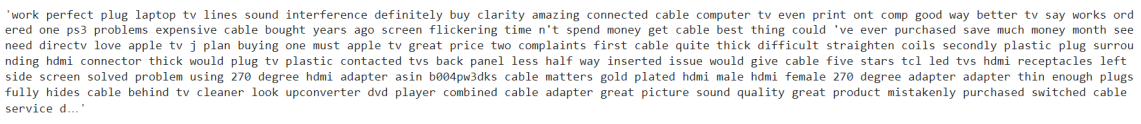

Figure 2.5: Example of a BART-base input used on the fine tuned model

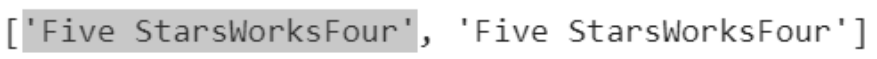

Figure 2.6: Example of a BART-base fine tuned result

2.1.3 Third test: BART-large-xsum

For the third model we've used **BART-large-xsum** that is a version of *BART-large* fine tuned on the Extreme Summarization (XSum) dataset, that is a dataset for evaluation of abstractive single-document summarization systems. We've exploited the pipeline for the summarization task provided by Hugging Face, so we've plugged in the pipeline directly the model and the associated tokenizer:

```
summarizer = pipeline("summarization",
                      model="facebook/bart-large-xsum",
                      tokenizer="facebook/bart-large-xsum")
```

The process to create the input and the stages are the same ones of the previous tests. The results obtained are more abstract than before, the batch relative to the following result is the same used in figure 2.5:

Hdmi cables work great provide high quality video blue ray player tv definitely fit bill work good ordered several pairs great product price normally order cables monoprice great quality great price happy cable purchased go new roku box would recommend work higher priced cables.

We can see that using this approach we've a model that abstracts the input text, so performing different stages the degree of abstraction increases, in fact the result of the second stage is the following:

Hdmi cable is one of the best buys ever made.

To evaluate if the sentiment of the starting batches of review is the same obtained after the second stage of summarization, we've performed a **sentiment analysis** using the following pipeline:

```
summarizer = pipeline(
model="nlptown/bert-base-multilingual-uncased-sentiment")
```

The expected ranking is obtained as the average star rating normalized between **[0, 1]** of the starting list of reviews.
The results are presented in the following and we can see that the **POSITIVE** sentiment is of the starting list is preserved in the final stage.

- **Expected rating:** 4.7 stars
- **Observed rating:** 5 stars
- **Confidence:** 0.96

We've performed the same test using the full text obtaining the following result after the first stage:

```
'This is a great product for the price.',
'AmazonBasics HDMI cable is a good buy for the price.',
"Amazon's HDMI cable for the Apple TV is a good buy for the price.",
'This HDMI cable is just as high quality as any other high quality cable, the only difference is the price.',
'This HDMI cable is a great deal for the price, and the quality is top notch.',
'The Amazon High-Speed HDMI cable that I purchased along with my Roku has been a huge success!',
"AmazonBasics HDMI cables have been a huge success for me, and I've been using them for two years now.",
'AmazonBasics High-Speed HDMI Cable is a great value for the money.',
'An excellent cable for connecting a Blu-Ray player to a receiver at a fraction of the cost of bricks and mortar retailers.',
'AmazonBasics HDMI cables are a great value for the money.',
'AmazonBasics Digital Optical Audio Toslink Cable, 6 Feet [[ASIN:B001TH7GSW AmazonBasics]].']
```

Figure 2.7: Results obtained using the full text after the first stage

The final result is the following:

An excellent cable for connecting a Blu-Ray player to a receiver at a fraction of the cost of bricks and mortar retailers.

We've performed the sentiment analysis obtaining a worst result respect to the one obtained using the cleaned text, in fact using the cleaned text we can embed more reviews due to the compression obtained using the text preprocessing:

- **Expected rating:** 4.7 stars
- **Observed rating:** 5 stars
- **Confidence:** 0.84

2.1.4 Fourth test: BART-large-cnn

In this last test using the BART model we've performed exactly the same approach of the previous ones. For brevity we don't report the results as this model works as an extractive summarization model.

2.2 T5[1]

T5, or Text-to-Text Transfer Transformer, is a Transformer based architecture that uses a text-to-text approach. Every task – including translation, question answering, and classification – is cast as feeding the model text as input and training it to generate some target text. This allows for the use of the same model, loss function, hyperparameters, etc. across our diverse set of tasks. The changes compared to BERT include:

- adding a causal decoder to the bidirectional architecture.

[1] https://paperswithcode.com/method/t5

- replacing the fill-in-the-blank cloze task with a mix of alternative pre-training tasks.

All tests were performed in the same way as those of the Bart model. For brevity only a brief description of the test and the corresponding results and analysis will be provided.

2.2.1 First test: T5-base

The first test on T5 has been performed using the T5-base tokenizer and model on a reduced dataset containing only one product and around 15000 entries. The model was tested using 5 subsets of reviews which differ from each other only on the number of reviews:

- 1000 reviews: "i bought this cable for my ps3 and it works great."
- 2000 reviews: "i've bought 12 of these cables over the last couple years."
- 5000 reviews: "i bought this cable when i got my new gaming device."
- 10000 reviews: "i bought this cable when i got my new gaming device."
- 15000 reviews: "a friend sent me a gift of high-speed cable 6.5 feet."

As can be seen from the previous entries the results do not seem to improve by increasing the number of reviews taken by the model. This can be explained by the fact that the reviews, despite being about the same product, may vary a lot depending on which aspects of the product are analyzed.

2.2.2 Second test: T5-base-finetuned

The second test on T5 has been performed using again the T5-base tokenizer and model but this time it was added a finetuning step to try to improve the model correctness. the fine-tuning process was performed identically to that used for Bart.

The tests were done using different number of elements for training selected randomly from the training dataset to ensure that there was no bias. Changing the number of elements given in input for the training phase showed no visible change to the results.

The results in each test are one of the followings:

- "great value"
- "great product"
- "works great"

As can be seen the results are not optimal since they do not convey a summary of the product's reviews but an extremely generic summary which could be applied to any product.

2.2.3 Third test: T5-large

It was decided to test the T5-large model to see if there was any change in the results using a the -large version of the model. The results obtained by these tests showed that there is no visible difference in using T5-base or T5-large.

2.2.4 Fourth test: T5-large-finetuned

Fine-tuning the T5-large model yields the same results obtained by fine-tuning the T5-base model: the results will be one of three different possible elements, again due to the limits brought by the review_headline field which can not be used as the label optimally in the fine-tuning phase

2.3 Pegasus [2]

Pre-training with Extracted Gap-sentences for Abstractive SUmmarization Sequence-to-sequence models, or PEGASUS, uses self-supervised objective Gap Sentences Generation (GSG) to train a transformer encoder-decoder model.

For this project it was decided to test the Pegasus-xsum model and all tests were performed in the same way as those of the Bart model. For brevity only a brief description of the test and the corresponding results and analysis will be provided.

2.3.1 First test: Pegasus-xsum

The first test on pegasus-xsum has been performed using the pegasus-xsum tokenizer and model on a reduced dataset containing only one product and around 15000 entries. The model was tested using 5 subsets of reviews which differ from each other only on the number of reviews taken into account:

- 500 reviews: "Buying cable can be tricky, so we've put together a guide to help you get the best bang for your buck."
- 1000 reviews: 'Buying cables can be a tricky business, but here are a few tips to help you get the best bang for your buck.'
- 2000 reviews: "Hi, I've been using this HDMI cable for a few months and it works perfectly."

[2] https://arxiv.org/abs/1912.08777

- 5000 reviews: "I've been using this cable for a few months and it's working well."
- 10000 reviews: "I've been using this HDMI cable for a few months and it's working perfectly.", "Here is my review of the best HDMI cable I've ever bought."

As can be seen, in this case the results vary depending on the number of reviews taken into account but there is no sensible difference between the results obtained by summarizing 2000 or more reviews.

2.3.2 Second test: Pegasus-xsum-finetuned

The second test on Pegasus has been performed using again the Pegasus-xsum tokenizer and model but this time a fine-tuning step was added to try to improve the model accuracy.

Similarly to the results obtained in the t5-base-finetuned test, the results of the pegasus-xsum-finetuned test consist of few possible results:

- 'Does the job'
- 'Basic HDMI cable'

Again it is possible to see that the use of the review_headline as the label for the fine-tuning of the model leads to extremely synthetic results which are not optimal and are not sufficient to provide a good quality summary

Chapter 3

Results and conclusion

Given the results of our experiments, we've decided to use **BART-large-xsum** as the model for the final version.
We've exploited the same approach used during the tests, but now we're using the full list of reviews for each product, grouped by the year of publication. In the following we present the results for the 5 product considered at the beginning (chapter 1).
To validate the results we've performed a sentiment analysis as in the previous tests. Since the results are too many to be displayed here we decided to only show some of them chosen randomly.

3.1 Results of the final version

Product B003L1ZYYM

- **Year 2015:** ['As part of our series on the best cable products, we take a look at some of the best hdmi cable products on the market.']
  Expected rating: 4.6 stars **Observed rating:** 5 stars **Confidence:** 0.75

Product B0001FTVEK

- **year 2009**: ['Here are some of the best and worst-selling wireless headphones available on the market.']
  Expected rating: 3.9 stars **Observed rating:** 5 stars **Confidence:** 0.77

Product B0012S4APK

- **year 2008**: ['A cheetah mount has been rated "excellent" by the US Consumer Product Safety Commission (CPSC).']
  Expected rating: 4.0 stars **Observed rating:** 5 stars **Confidence:** 0.51

Product B003EM8008

- **year 2015**: ["As part of our series on the best headphones, we've been looking at some of the best earbuds."]
  Expected rating: 4.4 stars **Observed rating:** 5 stars **Confidence:** 0.65

Product B0002L5R78

- **year 2013**: ['As part of our series of product reviews, we take a look at some of the best cables we have ever used.']
  Expected rating: 4.4 stars **Observed rating:** 5 stars **Confidence:** 0.68

3.2 Conclusion and future works

In conclusion the models act better in some cases and not very good in some other, this is due to the variabilty that can be encountered respect to the product and the time period taken into account. An important factor is that the validation using the sentiment analysis does not provide a reliable instrument to evaluate the quality of the summary, since the reviews are general and it is not guaranteed that they express the same sentiment observed in the rating (a technical review).
The problem of the fine tuning are the labels, in fact they're are extremely synthetic and don't represent a good example to train the model for a summarization task.

In order to improve our work we need to use larger models that can improve the abstraction power and can handle larger inputs.
Another important improvement that can be done is to gather more accurate labels, that can represent in a better way the thought behind the reviews, but for the nature of the problem it is a bit complex.
Another test that can be done is to use the helpful votes, in particular the added review weight attribute, in order to use some reviews that are already validated by other users, this makes the review more reliable.