Conversational PEGASUS

Gonçalo Raposo
goncalo.cascalho.raposo@tecnico.ulisboa.pt

INESC-ID, Instituto Superior Técnico, Universidade de Lisboa

Lisbon, Portugal

May 17, 2021

Abstract

Keywords First keyword · Second keyword · More

1 Motivation

PEGASUS is an abstractive summarization model that was proposed in Zhang et al. [2020]. Given a question, if the passages relevant for the answer are available, then the answering task will be very similar to a summarization of those passages.

2 Relevant Passages Retrieval

A system that given a question, retrieves passages from a database/document/website that are related with the question and are relevant for a possible answer. Use work from Rita Costa.

3 Answer Generation

3.1 PEGASUS Model

PEGASUS is an abstractive summarization model that was proposed in Zhang et al. [2020]. Abstractive summarization differs from extractive summarization in the sense that it might output words or sentences that are not present in the input text. It has a Transformer encoder-decoder architecture, each with 16 layers, and was pre-trained with two objectives:

- Gap Sentences Generation (GSG): Masking whole sentences from a document and generating these gap-sentences from the rest of the document. This works well as a pre-training objective for summarization tasks.
- Masked Language Modeling (MSM): Select 15% tokens in the input text and mask 80%, replace 10% by a random token, and 10% leave unchanged. The decoder has to correct those masked words.

For pre-training, 2 large text corpus were used: C4 (text from 350M Web-pages), and HugeNews (1.5B articles). This model is able to adapt very quickly when fine-tuning with small numbers of supervised pairs (1000 examples).

Various checkpoints are provided: PEGASUS-large consists in the pre-trained model, and the others (e.g. PEGASUS-xsum) correspond to checkpoints that fine-tuned the model with specific datasets (e.g. XSum).

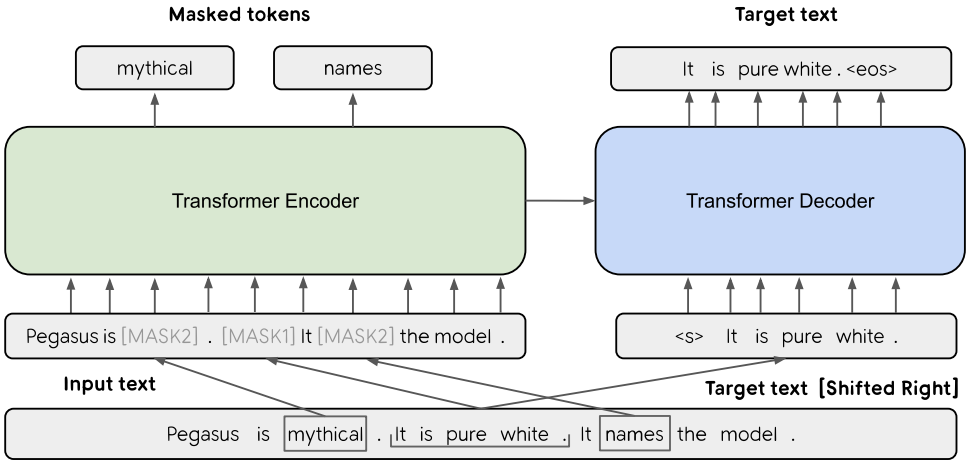


Figure 1: Transformer encoder-decoder architecture of PEGASUS model. Both GSG and MLM objectives are represented using the `[MASK1]` and `[MASK2]` tokens, respectively. Illustration obtained from Zhang et al. [2020].

3.2 TREC CAsT Dataset

The first idea was to use the dataset from TREC CAsT (Conversational Assistance Track) year 2, discussed in Dalton et al. [2020]. Bellow is an alleged example from this dataset:

```
{
  "number": 2,
  "turn": [
    {
      "number": 1,
      "raw_utterance": "What are the main breeds of goat?"
      "system_response": "The main types of goats include..."
      "response_document": "MARCO_5023599"
    },
    {
      "number": 2,
      "raw_utterance": "Tell me about boer goats."
      "system_response": "The development of the Boer goat in the early 1900s can be traced to the
           Dutch farmers of South Africa. Boer is a Dutch word meaning farmer."
      "response_source": "MARCO_2533087"
    },
    ...
  ],
}
```

As can be seen above, for each raw utterance, there is a system response and a response document/source. For the example presented, the response document/source, which would correspond to the relevant passage, points to a document from the MS MARCO (Microsoft Machine Reading Comprehension) dataset.

However, the TREC CAsT dataset turned out not to include the system responses, therefore, it could not be used to train the proposed model.

3.3 MS MARCO Dataset

MS MARCO (Microsoft Machine Reading Comprehension), presented in Nguyen et al. [2016], is a large scale dataset. Since its release, currently has $> 10^6$ unique real queries from Bing and focus not only in QA but many more problems related to search. Each entry of the MS MARCO dataset has the following structure:

```
{
  "answers": {
    "[]": "string"
```

```
  },
  "passages": {
    "[]": {
      "is_selected": "int32",
      "passage_text": "string",
      "url": "string"
    }
  },
  "query": "string",
  "query_id": "int32",
  "query_type": "string",
  "wellFormedAnswers": {
    "[]": "string"
  }
}
```

Given that each sample as a user query, relevant passages, and an answer, maybe this dataset could be used to train the proposed model. However, there is no conversational information.

3.4 CoQA Dataset

CoQA (Conversational Question Answering) is a large-scale dataset that was proposed in Reddy et al. [2019]. This dataset consists of $> 10^5$ question-answer regarding multiple text passages. What distinguishes this dataset is that the questions are conversational and the answers can be free-form text. This dataset has the following structure:

```
{
  "data": [
    {
      "source": "mctest",
      "id": "3dr23u6we5exclen4th8uq9rb42tel",
      "filename": "mc160.test.41",
      "story": "Once upon a time, in a barn near a farm house, there lived a little white kitten
          named Cotton. Cotton ...",
      "questions": [
        {
          "input_text": "What color was Cotton?",
          "turn_id": 1
        },
        {
          "input_text": "Where did she live?",
          "turn_id": 2
        },
        ...
      ],
      "answers": [
        {
          "span_start": 59,
          "span_end": 93,
          "span_text": "a little white kitten named Cotton",
          "input_text": "white",
          "turn_id": 1
        },
        {
          "span_start": 18,
          "span_end": 80,
          "span_text": "in a barn near a farm house, there lived a little white kitten",
          "input_text": "in a barn",
          "turn_id": 2
        },
        ...
    ],
    ...
```

3.5 Preparing the Data

Given the proposed task, the input of the model will correspond to a question concatenated with the corresponding story. As an example, a sample from the CoQA dataset would be constructed as:

Input: "What color was Cotton?
Once upon a time, in a barn near a farm house, there lived a little white kitten named Cotton. Cotton ..."

Output: "white"

With the input and output defined, the next step is the tokenization, which corresponds to assign a number/id to each token of the text. For this step, there are two memory and performance optimizations that can be performed, namely, choosing an appropriate integer data type and reducing the padding introduced in the batch tensor.

3.5.1 Data Type

To implement the pre-trained PEGASUS model, the Transformers library from Hugging Face was used. More detail can be found in Wolf et al. [2020]. Referring to its documentation[1], one can observe that `torch.LongTensor` is used for a token ids tensor, which uses long integers for each token id. A long integer occupies 8 bytes = 64 bits, which allows to represent about 1.8×10^{19} different tokens. Considering that the vocabulary size of PEGASUS defaults to 50265 tokens, a long integer is unnecessarily large to store PEGASUS tokens. Apart from the `MASK` token, the other tokens are a positive integer, therefore, a 32-bit integer is enough for this scenario, since it can represent about 2.1×10^9 positive values. One might wonder if a 16-bit integer would still be enough, but it is only able to represent 32767 positive values, which is insufficient.

Replacing 64-bit integers (long) by 32-bit integers is, therefore, a costless way to reduce the memory utilization of the input and output tensors by half. The task of choosing the appropriate data type considering the representation requirements is an important and beneficial one. However, the model expects the output/labels tensor to be a `torch.LongTensor` and raises an error otherwise. This could probably be fixed, but for the current application that tensor was kept as a `torch.LongTensor` while the input tensor is reduced to a `torch.IntTensor`.

3.5.2 Dynamic Batches

The PyTorch Lightning framework, available in Falcon et al. [2019], was used to train and test the model. The typical pipeline of preparing the dataset consists in reading the text sentences and passing them to the tokenizer, which returns a tensor where each row will correspond to the token ids of a sentence. However, since sentences can have different lengths, the tokenizer pads every sentence to the length of the longest one, can be somewhat wasteful if in the dataset there is a much longer sentence than the rest. Although the padded values are masked by the model, they are still considered in the calculations, which spends computing and memory resources.

For this application, a technique I called dynamic batches, or referred in Benesty [2020] as dynamic/variable length padding, was used to decrease that unnecessary padding. It basically consists in delaying the creation of the tensor, so that, instead of creating a tensor for the entire dataset, only create tensors for the batches[2]. This way, instead of ending up with all sentences padded to the length of the longest sentence in the dataset, they are padded to the length of the longest sentence in the considered batch. Therefore, each batch tensor may have a different sequence length. This is particularly useful when there are only a few sentences that are much larger.

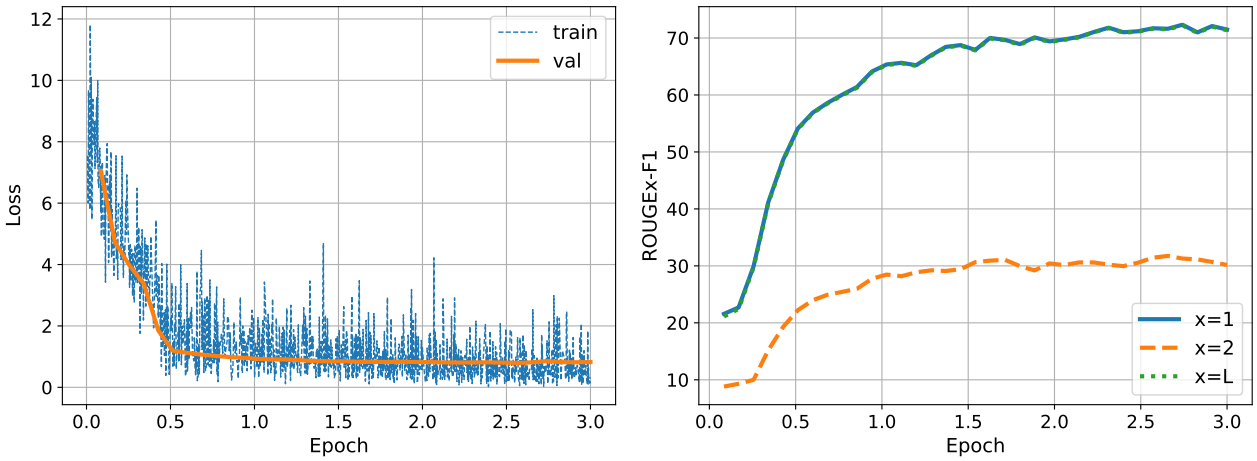
A thorough analysis was performed in Benesty [2020], which observed a 4.7 speedup when using dynamic batches of size 16 on the dataset X-NLI (Conneau et al. [2018]).

[1] https://huggingface.co/transformers/

[2] In this paper, the word batches actually refers to mini-batches.

4 Results

The PEGASUS model (checkpoint pegasus-large) was trained with the CoQA dataset for 3 epochs (early stopping while monitoring the validation loss). The maximum input sequence length was set to 1024 and the maximum output sequence length to 256. The batch size had to be only 2 due to insufficient GPU memory, but it accumulates the gradient of 128 batches during training, so that it is similar to a batch size of 256. The loss and ROUGE values obtained are presented in Figure 2.



(a) Train and validation loss.

(b) ROUGE metric in the validation dataset.

Figure 2: Results of training PEGASUS (large) in CoQA for 2 epochs.

4.1 Interacting with Streamlit

The best way to validate a conversational QA model is to ask it some questions. In that context, the framework Streamlit[3] was used to design and deploy a simple front-end that would allow to interact with the model by giving a context and questions and then obtaining an answer. The implemented interface is shown in Figure 3.A few examples are shown in Appendix A.

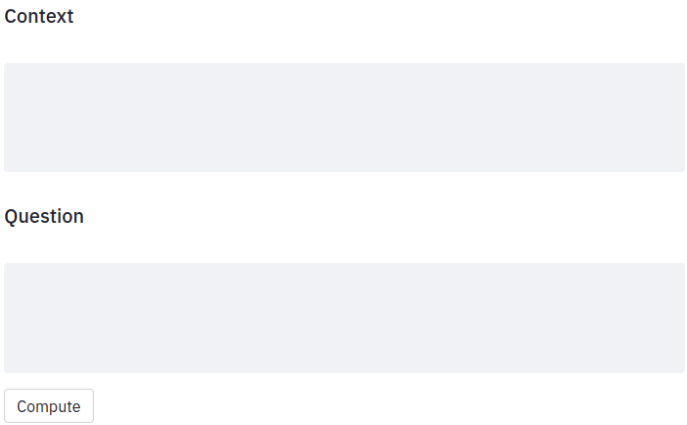


Figure 3: Designed interface using Streamlit.

[3]https://streamlit.io/

References

M. Benesty. Divide HuggingFace training time by 2 | Towards Data Science, 2020. URL https://towardsdatascience.com/divide-hugging-face-transformers-training-time-by-2-or-more-21bf7129db9q-21bf7129db9e. Accessed on 2021-05-13.

A. Conneau, R. Rinott, G. Lample, A. Williams, S. Bowman, H. Schwenk, and V. Stoyanov. XNLI: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2475–2485, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1269. URL https://www.aclweb.org/anthology/D18-1269.

J. Dalton, C. Xiong, and J. Callan. Cast 2020: The conversational assistance track overview. In *The Twenty-Ninth Text REtrieval Conference(TREC 2020) Proceedings*, 2020. URL https://trec.nist.gov/pubs/trec29/trec2020.html.

W. A. Falcon et al. PyTorch Lightning - GitHub, 2019. URL https://github.com/PyTorchLightning/pytorch-lightning. Accessed on 2021-05-13.

T. Nguyen, M. Rosenberg, X. Song, J. Gao, S. Tiwary, R. Majumder, and L. Deng. MS MARCO: A human generated machine reading comprehension dataset. In T. R. Besold, A. Bordes, A. S. d'Avila Garcez, and G. Wayne, editors, *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. URL http://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf.

S. Reddy, D. Chen, and C. D. Manning. CoQA: A conversational question answering challenge. *Transactions of the Association for Computational Linguistics*, 7:249–266, nov 2019. doi: 10.1162/tacl_a_00266.

T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, P. Cistac, T. Rault, R. Louf, M. Funtowicz, J. Davison, S. Shleifer, P. von Platen, C. Ma, Y. Jernite, J. Plu, C. Xu, T. Le Scao, S. Gugger, M. Drame, Q. Lhoest, and A. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, Oct. 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-demos.6. URL https://www.aclweb.org/anthology/2020.emnlp-demos.6.

J. Zhang, Y. Zhao, M. Saleh, and P. J. Liu. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR, 2020.

A Examples from Streamlit

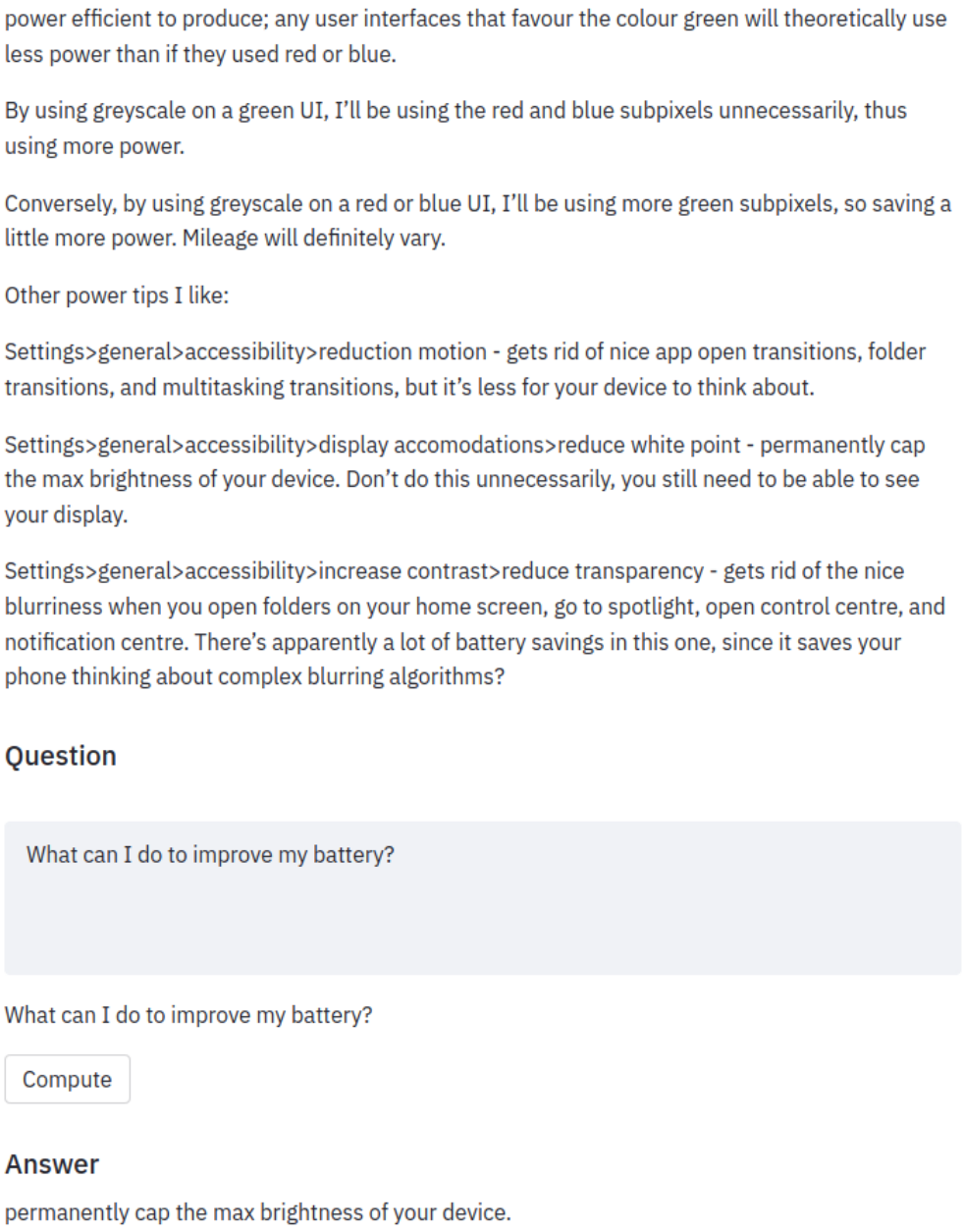


Figure 4: The context is a set of several tips about improving the battery life of an iPhone.

Dominic Cobb is the foremost practitioner of the artistic science of extraction, inserting oneself into a subject's dreams to obtain hidden information without the subject knowing, a concept taught to him by his professor father-in-law, Dr. Stephen Miles. Dom's associates are Miles' former students, who Dom requires as he has given up being the dream architect for reasons he won't disclose. Dom's primary associate, Arthur, believes it has something to do with Dom's deceased wife, Mal, who often figures prominently and violently in those dreams, or Dom's want to "go home" (get back to his own reality, which includes two young children). Dom's work is generally in corporate espionage. As the subjects don't want the information to get into the wrong hands, the clients have zero tolerance for failure. Dom is also a wanted man, as many of his past subjects have learned what Dom has done to them. One of those subjects, Mr. Saito, offers Dom a job he can't refuse: to take the concept one step further into inception, namely planting thoughts into the subject's dreams without them knowing. Inception can fundamentally alter that person as a being. Saito's target is Robert Michael Fischer, the heir to an energy business empire, which has the potential to rule the world if continued on the current trajectory. Beyond the complex logistics of the dream architecture of the case and some unknowns concerning Fischer, the biggest obstacles in success for the team become worrying about one aspect of inception which Cobb fails to disclose to the other team members prior to the job, and Cobb's newest associate Ariadne's belief that Cobb's own subconscious, especially as it relates to Mal, may be taking over what happens in the dreams.

Question

What was their mission?

What was their mission?

Compute

Answer

To take the concept one step further into inception, namely planting thoughts into the subject's dreams without them knowing.

Figure 5: The context is a synopsis of the movie Inception.

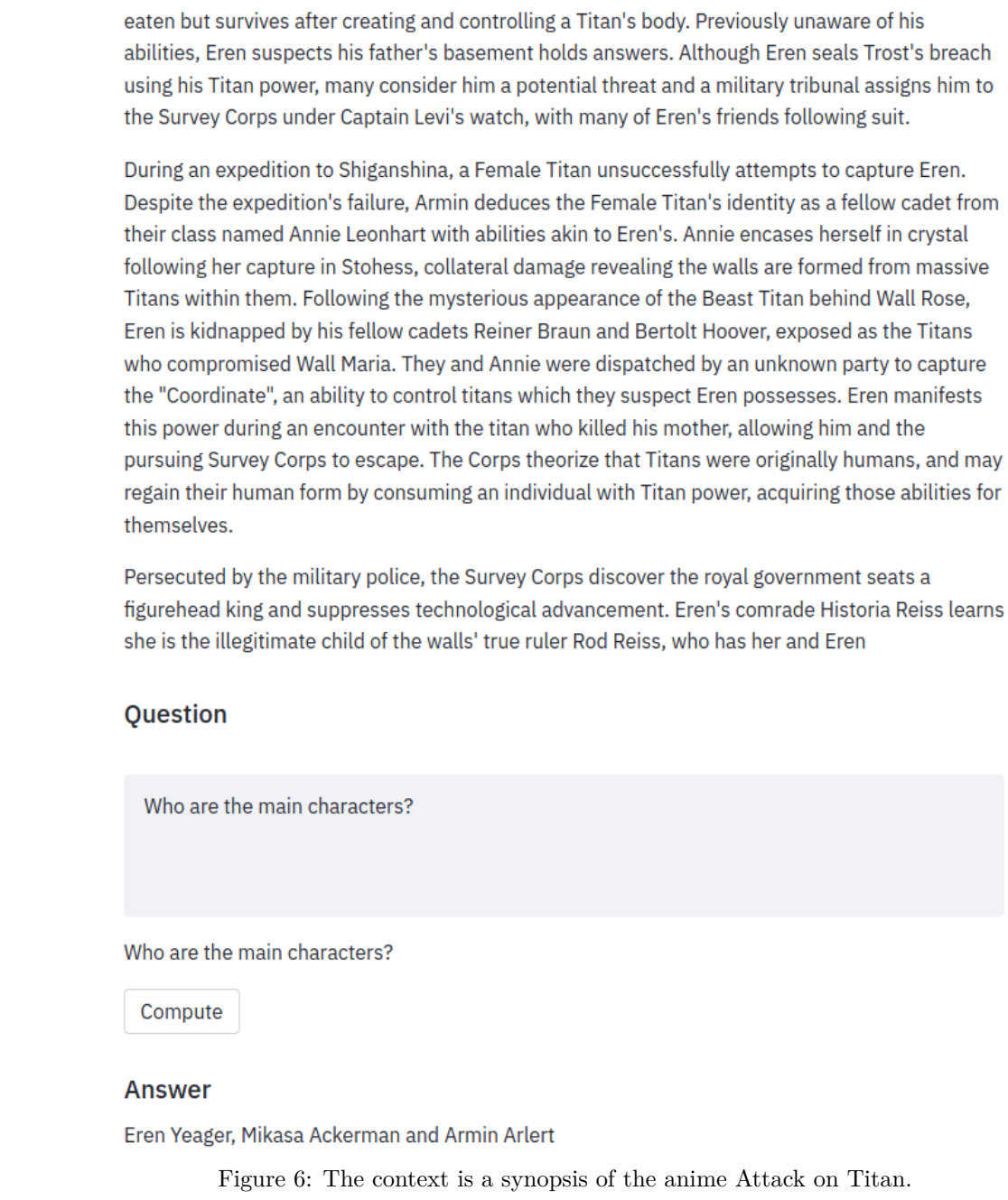


Figure 6: The context is a synopsis of the anime Attack on Titan.