Project Report

Moldir Koishybayeva
moldir.koishybayeva@univaq.it
University of L'Aquila — February 8, 2023

Introduction

The ability to condense a long text while maintaining the key points makes the summarization of extensive texts crucial. This can be helpful in applications such as text classification, natural language understanding, and information retrieval.

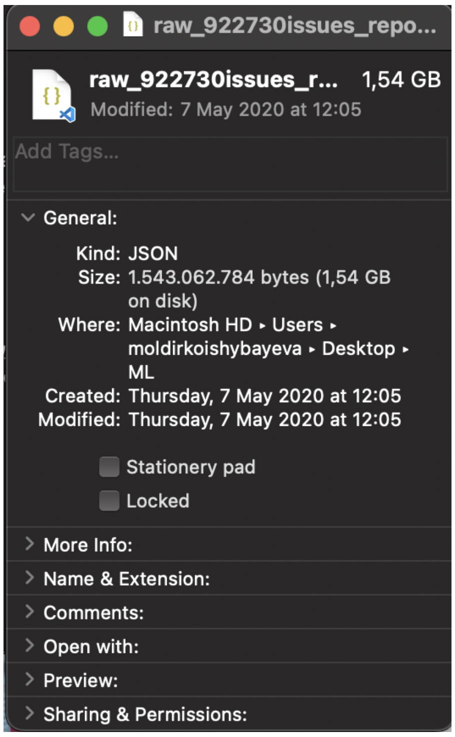
The proposed **project's objective** is to create a deep learning approach for summarizing long and complex texts, which refers to taking the most crucial information from a lengthy document and simplifying it into a clear, concise statement. Recurrent neural networks, transformers, and attention mechanisms are some of the methods used in the deep learning approach to analyze and comprehend the semantic content of the text.

The task contributes to providing text descriptions for GitHub issues. Three of the HuggingFace Transformer library's models, including BERT, BART, and T5, have been employed in this project. Project available at this URL

1 Dataset

The iTape project's publicly accessible dataset has been used, and the iTiger project(Zhang et al., 2022) presented as inspiration.

Number, title, body, and repository are the four columns that represent the dataset, which contains 922730 rows. Figure 1 depicts the data. (Zhang et al., 2022)



raw_922730issues_repo...

raw_922730issues_r... 1,54 GB
Modified: 7 May 2020 at 12:05

Add Tags...

General:

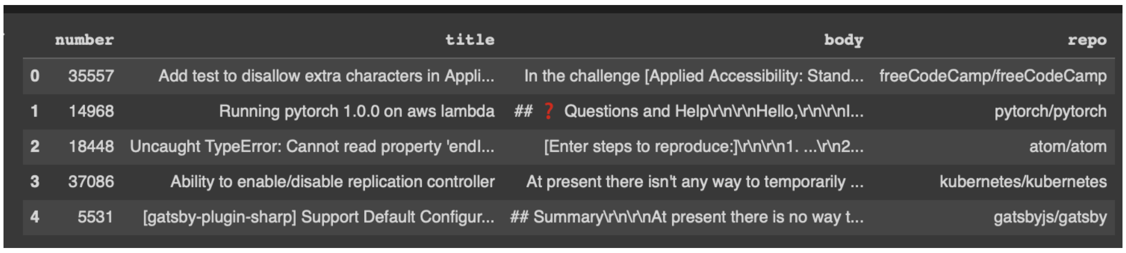
Kind: JSON
Size: 1.543.062.784 bytes (1,54 GB on disk)
Where: Macintosh HD ▸ Users ▸ moldirkoishybayeva ▸ Desktop ▸ ML
Created: Thursday, 7 May 2020 at 12:05
Modified: Thursday, 7 May 2020 at 12:05

Stationery pad
Locked

More Info:
Name & Extension:
Comments:
Open with:
Preview:
Sharing & Permissions:

Fig 1. Dataset

In order to automate the generation of GitHub issue titles, a dataset was gathered by the researchers of iTape project . The acquired dataset is comprehensive, extensive enough, and applicable to the project's purpose. Figure 2 illustrates overall format and content of the columns.



	number	title	body	repo
0	35557	Add test to disallow extra characters in Appli...	In the challenge [Applied Accessibility: Stand...	freeCodeCamp/freeCodeCamp
1	14968	Running pytorch 1.0.0 on aws lambda	## ❓ Questions and Help\r\n\r\nHello,\r\n\r\nI...	pytorch/pytorch
2	18448	Uncaught TypeError: Cannot read property 'endl...	[Enter steps to reproduce:]\r\n\r\n1. ...\r\n2...	atom/atom
3	37086	Ability to enable/disable replication controller	At present there isn't any way to temporarily ...	kubernetes/kubernetes
4	5531	[gatsby-plugin-sharp] Support Default Configur...	## Summary\r\n\r\nAt present there is no way t...	gatsbyjs/gatsby

Fig 2. The view of a dataset in a Pandas DataFrame

2 Data Preprocessing

Preparing the data for modeling is an essential stage in the data preparation process. Data preparation is fundamental to the machine learning process due to the frequent flaws, inconsistencies, and noise in raw data as well as its potential to contain irrelevant information, outliers, and missing values.

Initially, the number column from Figure 2 was omitted since it didn't provide any information for the generation of the title. Furthermore, re package was employed to prepare the title and body columns.

Python’s re package provides regular expression operations that can be employed to detect and replace patterns in text data. The module is used to match textual patterns and conduct a number of tasks, including searching for specific words, changing portions of strings, and validating data.

Further steps were performed such as:

1. **body column**

   - **replacing code snippets with word ‘code’**
     `data['body'].replace(to_replace=re.compile(r'(?:.|)+?'), value=' code ', regex=True)`
     Using the re.compile() and re.replace() methods, the following code substitutes all occurrences of code snippets in the "body" column of the "data" DataFrame with the word "code." Markdown’s code snippet syntax is compatible with the regular expression pattern r""(?:.|n)+?"".
   - **replacing added image link with word ‘image’**
     `data['body'].replace(to_replace=re.compile(r'\!\[(.*?)\]\(.+?)\)'), value=' image ', regex=True)`
     This code uses the re.compile() and re.replace() methods to replace all instances of image links with the word "image" in the "body" column of the "data" DataFrame. The markdown syntax for image links matches the regular expression pattern r'![(.*?)](.+?)’. The parenthesis capture the picture alt text, which can then be retrieved as a group in the resulting match object.
   - **replacing hyperlink text with word ‘link’**
     `data['body'].replace(to_replace=re.compile(r'(?<!\!)\[(.*?)\] \(.+?)\)'), value=' link ', regex=True)`
   - **replacing url text with word ‘url’**
     The regular expression pattern matching common URL syntax, including both HTTP and FTP protocols were replaced with word url.
   - **removing new lines**

2. **title column**

- **removing tags**
  A string of text enclosed in square brackets that is used to categorize or label the issue is called a tag. In order to identify the problem's nature, priority, status, and other pertinent details, tags are often appended to the beginning of an issue title. Therefore, tags were removed to simplify the issue title and evaluation.
- **removing emphasis**
  An emphasis is a technique used to highlight or emphasize a particular text. It is possible to emphasize text using a variety of methods, including bold, italics, underlining, or a mix of these. In order to reduce title consistency, emphasis was reduced.

3 Exploratory Data Analysis

From the illustration below, we can clearly observe from *almost 1 million GitHub issues* that coders usually mention words such as *support, work, fail, working, error, using, add, window, value*, while providing a title for the issue.

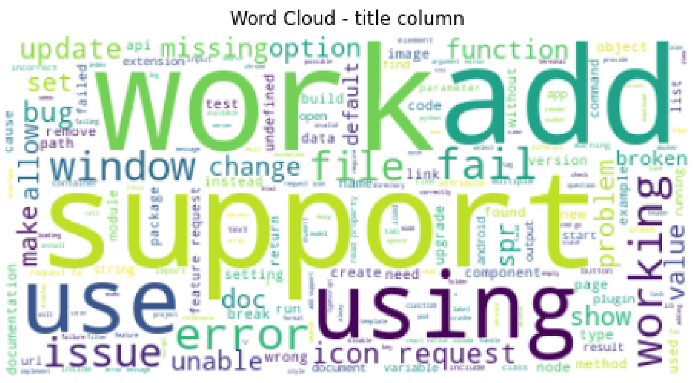


Fig 3. Word Cloud. Title column

From Figure 4 it can be noted that the most common repositories are *microsoft/vscode* (77796 issues), *golang/go* (34730 issues), and *kubernetes/kubernetes* (32704 issues).

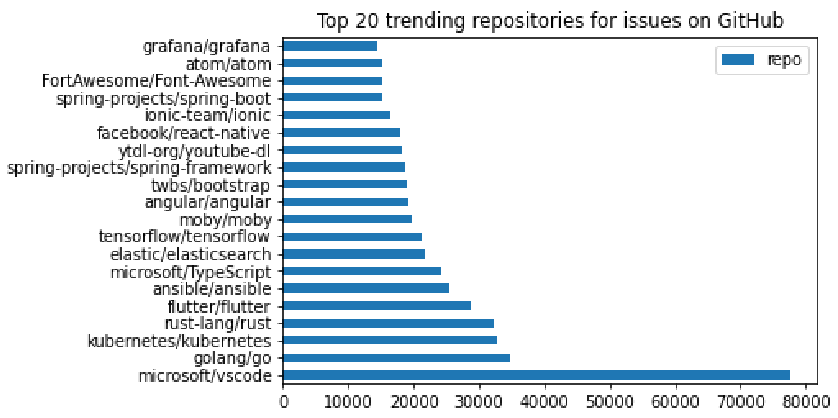


Fig 4. Top 20 trending repositories for issues on GitHub

After understanding our dataset, let's now proceed to modelling and evaluation.

4 Modelling

HuggingFace's Transformer provides pre-trained models for natural language processing purposes. Utilizing a pre-trained model for summarization purposes includes these steps:

- Loading a pre-trained model
- Fine-tuning the model
- Model evaluation

Functions that are provided by the Hugging Face's transformers library were included while implementation of the **iTiger project**.

- **AutoConfig** is an automated tuning of models based on transformers. Using AutoConfig, it can optimize the hyper-parameters of a transformer model.
- **AutoModelForSeq2SeqLM** is a function for automatically selecting and configuring models.
- **AutoTokenizer** is a function for automatically selecting tokenizer for a specified model
- **Seq2SeqTrainer, Seq2SeqTrainingArguments** are imported for the purpose of training sequence-sequence models.
- **MBartTokenizer** provides tokenization for the MBart model.
- **MBartTokenizerFast** provides tokenization for the MBart model.

For the models presented (BART, BERT, and T5), the `num_beams` parameter was set to 2, `max_length = 30`, and `min_length=10` to generate shorter text for the issue title. If the `num_beams` parameter is higher, it will generate more diversified text. Parameters `min_length` and `max_length` represent the minimum and maximum length of the generated text, respectively.

4.1 BART

```
model= "sshleifer/distilbart-cnn-6-6"
bart_tokenizer = AutoTokenizer.from_pretrained(model)
bart_model = BartForConditionalGeneration.from_pretrained(model)
```

distilbart-cnn-6-6 is a fast version of the cnn-based BART model. *AutoTokenizer* is a tokenizer that automatically tokenizes with appropriate tokenization method. *BartForConditionalGeneration* is a class for fine tuning the DistilBART model (AWS, n.d.).

4.2 BERT

BERT model training is computationally intensive, involving a robust GPU and a considerable volume of data.

```
tokenizer = AutoTokenizer.from_pretrained("bert-base-cased")
model = BertForMaskedLM.from_pretrained("bert-base-cased")
```

BertForMaskedLM is a class for fine-tuning the BERT model to the Masked Language Modeling task.Tunstall, L., Werra, L.von and Wolf, T., 2022

4.3 T5

```
model = T5ForConditionalGeneration.from_pretrained('t5-small')
tokenizer = T5Tokenizer.from_pretrained('t5-small')
```

T5-small is the text-to-text transformer model version. It is a large-scale pre-trained transformer model developed by Google Research.

5 Evaluation

The metrics for BLEU were chosen for analysis. A higher BLEU score indicates that the machine-generated translation and the reference translation are significantly more comparable when assessing the n-gram overlap between the machine-generated text and the reference. We can observe from the provided table below that the precision score derived from BLEU measures varies.

	Experiment 1	Experiment 2
BART	41.67	18.18
BERT	35.71	11.27
T5	23.81	23.53

Table 1: Precision Scores

According to the total number counts, it can be stated that BERT model is more applicable for paraphrasing larger text. Meanwhile, BART, T5 models provided shorter generated title.

6 Conclusions

Following the preceding stages, it can be stated that the titles produced by the BART and T5 transformer models were precise. Further steps are required:

- alternate evaluation metrics, as BLEU does not take meaning into account
- consider Pegasus transformer-based neural network architecture as possibility
- fine-tuning models on Amazon Sagemaker

References

AWS, A. (n.d.). *Set up a text summarization project with Hugging Face Transformers: Part 2*. Available from: `https://aws.amazon.com/blogs/machine-learning/part-2-set-up-a-text-summarization-project-with-hugging-face-transformers/`.

Tunstall, L., Werra, L.von and Wolf, T. (2022). *Natural language processing with transformers: Building language applications with hugging face*. O Reilly Media.

Zhang, T., Irsan, I. C., Thung, F., Han, D., Lo, D., and Jiang, L. (Nov. 2022). "iTiger: an automatic issue title generation tool". In: *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering*. ACM. DOI: `10.1145/3540250.3558934`. Available from: `https://doi.org/10.1145%2F3540250.3558934`.