

PyOverflow - Final Report

Rohith Pudari & Vishal Kanna

Word Count: 1952

Penalty: 0

1. Introduction

Content on StackOverflow [5] often serves a complimentary role for official documentations with users providing more example usages and clarifying confusing concepts [2]. However, with the sheer volume of posts on Stack Overflow and no seemingly centralized structural organization, it is difficult to utilize StackOverflow as a systematic roadmap for learning new technologies [3]. A structured organization of StackOverflow posts can help readers branch out of their initial questions and explore other content, which is especially helpful when a developer is unaware of the specific items they are looking for. The project will focus on classifying questions on python into classes that are defined by the official standard library documentation of python [4]. For example, all the StackOverflow questions related to datatypes like lists, tuples, float.etc will be classified under the “Built-in Types” category same as the official python documentation.

This will be useful for cataloging the questions on StackOverflow into their respective categories extracted from the official documentation of python. This process in turn will help programmers find similar questions on the same topic and also help link key concepts from pythons to the questions if a user wants to explore in further detail.

2. Illustration of the working of the project

Input- The input to the model is different for training and testing. We also took some measures (described in data processing) to reduce the differences between the train and test data.

Training- sentences scrapped from the documentation + sampled questions from stack overflow

Testing- Questions from stack overflow

Model- CNN baseline/ GPT-2

Output- label from 0 to 3 representing a category in python



Figure - 1: The Illustration of the Project.

3. Background

Previous works has leveraged StackOverflow as a rich source of information by manually classifying and clustering questions to understand the current challenges [5] and trends of specific domains [6], such as mobile development [1], security [2,3], and cryptography [4]. These qualitative studies extract valuable insights into characteristics and motivations of Stack Overflow posts, potential improvements for existing software APIs, potential improvements of the overall Stack Overflow platform, and how individuals can use StackOverflow more effectively. Nevertheless, the process of manually classifying StackOverflow is a laborious task, which could be expedited through automated mechanisms.

4. Data Processing

We are using official documentation of python [4] as the training data and python questions from StackOverflow [5] as the test data.

4.1 Training data

4.1.1 Documentation scraping

We scraped official documentation of python for four categories: “Built-in Types” and “Built-in Exceptions”, “Regular Expressions(Regex)” and “Strings” . We used beautifulsoup and requests packages in python to perform scraping of documentation ([full code here](#)). We retrieved all the documentation content in the form of JSON files ([here](#))

Built-in Types

The following sections describe the standard types that are built into the interpreter.

The principal built-in types are numerics, sequences, mappings, classes, instances and exceptions.

Some collection classes are mutable. The methods that add, subtract, or rearrange their members in place, and don't return a specific item, never return the collection instance itself but `None`.

Some operations are supported by several object types; in particular, practically all objects can be compared for equality, tested for truth value, and converted to a string (with the `repr()` function or the slightly different `str()` function). The latter function is implicitly used when an object is written by the `print()` function.

Truth Value Testing

Any object can be tested for truth value, for use in an `if` or `while` condition or as operand of the Boolean operations below.

By default, an object is considered true unless its class defines either a `__bool__()` method that returns `False` or a `__len__()` method that returns zero, when called with the object. [1] Here are most of the built-in objects considered false:

- constants defined to be false: `None` and `False`.
- zero of any numeric type: `0`, `0.0`, `0j`, `Decimal(0)`, `Fraction(0, 1)`
- empty sequences and collections: `''`, `()`, `[]`, `{}`, `set()`, `range(0)`

Operations and built-in functions that have a Boolean result always return `0` or `False` for false and `1` or `True` for true, unless otherwise stated. (Important exception: the Boolean operations `or` and `and` always return one of their operands.)

Figure - 2: A snippet of a few sentences being scrapped for the class built in types.

4.1.2 Code removal

We removed all the code from our training data (documentation of python) for simplicity and also because the GPT-2 model is not pretrained for code. This step was largely done while scrapping but also some removal took place while cleaning.

```
[
  {
    "Topic": "Built-in Types",
    "content": [
      "The following sections describe the standard types that are built into the\ninterpreter.",
      "The principal built-in types are numerics, sequences, mappings, classes,\ninstances and exceptions.",
      "Some collection classes are mutable. The methods that add, subtract, or\nrearrange their members in place, and don\u2019t return a specific item, never re
    ],
    "functions": [],
    "code_demo": [],
    "Subtopics": [
      {
        "Topic": "Truth Value Testing",
        "content": [
          "Any object can be tested for truth value, for use in an if or\nwhile condition or as operand of the Boolean operations below.",
          "By default, an object is considered true unless its class defines either a\n__bool__() method that returns False or a __len__() method that\nreturn
        ],
        "functions": [],
        "code_demo": [],
        "Subtopics": []
      }
    ],
    {
      "Topic": "Boolean Operations and, or, not",
      "content": [
        "These are the Boolean operations, ordered by ascending priority:",
        "Notes:",
        "This is a short-circuit operator, so it only evaluates the second\nargument if the first one is false.",
        "This is a short-circuit operator, so it only evaluates the second\nargument if the first one is true.",
        "not has a lower priority than non-Boolean operators, so not a == b is\ninterpreted as not (a == b), and a == not b is a syntax error."
      ],
      "functions": []
    }
  }
]
```

Figure 3: A snippet from built-in types JSON file.

4.1.3 Cleaning/ labeling

The JSON files are converted to pandas dataframes. The content under each major topic was stripped of special characters such as (‘\n’, ‘\u’,etc) and split into sentences. Each sentence was given a label based on where the sentence was scrapped from. All the classes are combined together to create a training dataset. Built-in exceptions had 42 sentences, Built-in types had 153 sentences, Strings had 88 sentences and regex had 34 sentences in the documentation.

| | content | topic | label |
|-----|---|---------------------|-------|
| 0 | In Python, all exceptions must be instances of... | Built-in Exceptions | 0 |
| 1 | In a try statement with an except clause tha... | Built-in Exceptions | 0 |
| 2 | Two exception classes that are not related v... | Built-in Exceptions | 0 |
| 3 | The built-in exceptions listed below can be g... | Built-in Exceptions | 0 |
| 4 | Except where mentioned, they have an "associ... | Built-in Exceptions | 0 |
| ... | ... | ... | ... |
| 293 | The module defines several functions, constan... | regex | 3 |
| 294 | Some of the functions are simplified versions... | regex | 3 |
| 295 | Most non-trivial applications always use the... | regex | 3 |
| 296 | Compiled regular expression objects support t... | regex | 3 |
| 297 | Since match() and search() return None when t... | regex | 3 |

298 rows x 3 columns

Figure 4: Final Training dataset.

4.1.3 Variations in training data

In the examples below the blue rectangle boxes represent individual sentences in the python documentation and the ellipse represent a single training example that the model will be fed for training.

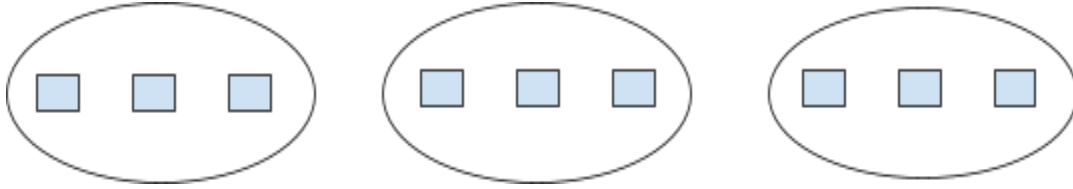
Case 1 (Sentences)- separate sentences

Each sentence in the documentation is one training example in the training set.



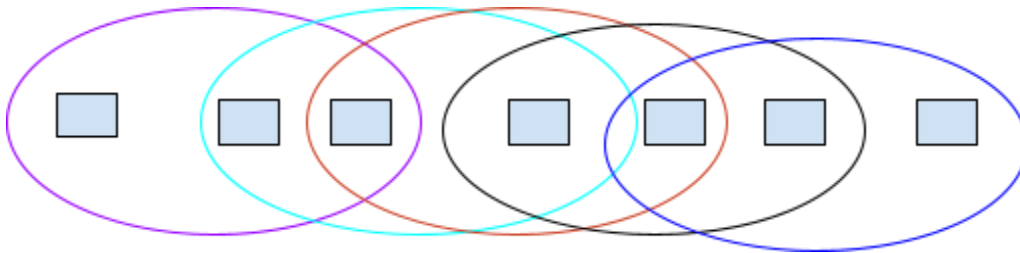
Case 2 (No Overlap)- 3 combined sentences with no overlap

Each training example contains 3 consecutive sentences from the documentation,



Case 3 (Overlap) - 3 combined sentences with overlap

Each training example contains 3 consecutive sentences from the documentation, with overlap, this is done to increase the context to the model without losing the size of the training data.



Case 4 (Stack overflow)- This is case 3 plus 10 stack overflow questions for each label

4.2 Test Data

4.2.1 SQL to retrieve training examples

Initially we retrieved all the questions in StackOverflow with the tag “python” using SQL. We used StackOverflow data explorer [11] to retrieve those questions. The questions are further filtered using tags such as “Built in types”, “exceptions”, “strings” and “regex”. From these filtered subsets we choose 100 questions for each tag except for built-in types which had only 34 questions.

| 1 | Id | PostTypeId | AcceptedAnswerId | ParentId | CreationDate | DeletionDate | Score | ViewCount | Body |
|---|----------|------------|------------------|----------|---------------------|--------------|-------|-----------|--|
| 2 | 2052390 | 1 | 24065533 | | 2010-01-12 21:07:40 | | 3002 | 2659297 | <p>How do I raise an exception in Python so that it can later be caught via an <code>except</code> |
| 3 | 13267912 | 1 | 13363934 | | 2012-11-07 10:36:32 | | 2 | 310 | <p>Where does boost python register from python converters for builtin types such as from <code>P |

Figure 5: Raw test data, with two example StackOverflow questions.

4.2.2 code removal

Similar to the training data, we removed all the code blocks from the test data for simplicity.

4.2.3 Cleaning/ labeling

The data is put through a similar cleaning process as the training data, to remove the special characters but we do not split the test data into sentences. We split the test data

into its title and body and add the respective label based on the tags it was filtered with Figure 6 shows the final cleaned test data set for all the labels. The test data file can be accessed [here](#)

| Id | Body | Title | Tags | text | label |
|---------|------------------------------------|---|---|---------------------------|-------|
| 166198 | I'm writing a little debug app for | How can I capture all exceptions from a wxPython application? | <python><exception><error-handling><wxwidgets><error-reporting> | How can I capture all ex | 0 |
| 1291438 | From this question, I'm now do | When I catch an exception, how do I get the type, file, and line | <python><exception><exception-handling><error-handling> | When I catch an excepti | 0 |
| 1350671 | My background is in C# and I'd | "Inner exception" (with traceback) in Python? | <python><exception><error-handling> | "Inner exception" (with t | 0 |
| 1483429 | How do I print the error/except | How do I print an exception in Python? | <python><exception><error-handling> | How do I print an excep | 0 |
| 1508467 | How can I log my Python exce | Log exception with traceback in Python | <python><exception><logging><error-handling> | Log exception with trac | 0 |
| 2739582 | When is exception handling m | Condition checking vs. Exception handling | <python><exception><error-handling><conditional-statements> | Condition checking vs. I | 0 |
| 3891804 | I am trying to raise a Warning | Raise warning in Python without interrupting program | <python><exception><error-handling><warnings> | Raise warning in Pythor | 0 |
| 4096087 | There is a list standard python | Where's the standard python exception list for programmes to | <python><exception><error-handling> | Where's the standard py | 0 |
| 4572362 | I'm trying to call a function. Or | Is there someway I can get specific details about an AttributeError | <python><exception><attributes><error-handling> | Is there someway I can | 0 |

Figure 6: A snippet of the cleaned test data.

4.2.4 Variations in test data

The test data has 3 variations on what part of the StackOverflow question is used as the input of the model:

Case 1 (Title): Title of the StackOverflow question.

Case 2 (Body): Body of the StackOverflow question

Case 3 (Combined): Title + Body of the StackOverflow question

5. Architecture/Software

We used the GPT-2 medium model from huggingface with a classification head of 4 classes. It has 24-layers and 345 Million parameters. We fine-tuned the model for our classification task. For this report, we used the default parameters for training. The code file can be accessed [here](#). The training arguments are as follows:

- Epochs - 3 for the initial testing.
- Epochs- 6 for StackOverflow case in training data.
- Batch size - 1
- Evaluation Strategy: Accuracy
- Training time- 4 mins per epoch on dual NVIDIA Tesla T4 GPUs

6. Baseline Model

The baseline model has 2 Convolutional layers that have kernels of different size, then go through max pooling. The output of the maxpool layers are combined and dropout is applied here. After the dropout there are 2 fully connected layers which output 4 logits which finally undergo softmax to give the final class probabilities for the 4 classes.

The training arguments are as follows-

- Epochs - 80
- Batch size - 1
- Optimizer is Adam with learning rate of 0.0001
- Loss- Cross entropy loss
- Evaluation Strategy: Accuracy

```
CNNModel(  
  (embedding): Embedding(400000, 100)  
  (conv1): Conv2d(1, 25, kernel_size=(100, 2), stride=(1, 1), bias=False)  
  (conv2): Conv2d(1, 25, kernel_size=(100, 3), stride=(1, 1), bias=False)  
  (activate): ReLU()  
  (maxpool): MaxPool2d(kernel_size=(1, 500), stride=(1, 500), padding=0, dilation=1, ceil_mode=True)  
  (dropout): Dropout(p=0.25, inplace=False)  
  (out1): Linear(in_features=50, out_features=50, bias=True)  
  (out2): Linear(in_features=50, out_features=1, bias=True)  
)
```

Figure 7: Architecture of Baseline Model.

7. Quantitative Results

The X-axis on the graph shows the various combinations of training data and test data both the models were trained and evaluated with. The Y-axis represents accuracy of the model on the test data

We can see that in all cases GPT-2 performed better in most cases or at par with the baseline in the worst case.

When we overlapped our sentences to train the models we saw an increase in our test accuracy over just using individual sentences in both models. It can also be seen that in this case when we use both the title and body of the questions for testing there is an increase in accuracy in both models than by using just one of them to classify the question.

Another surprising result was when we combined sentences but without overlap our accuracy for both models severely dropped- we feel this is due to the lack of training data as this process cuts our training data size to one third of the size of using individual sentences.

A final experiment was done to achieve an increase in accuracy by including 10 questions from each class into the training set with the highest performance (Combining sentences with overlap). We also increased our training epochs for the GPT-2 model to 6 as we felt the model could perform better with increased training while the baseline was already saturated at 80 epochs.

These changes achieved the highest accuracy in both models on the test set with both the body and the title of the question included.

GPT-2- 0.8

Baseline- 0.49

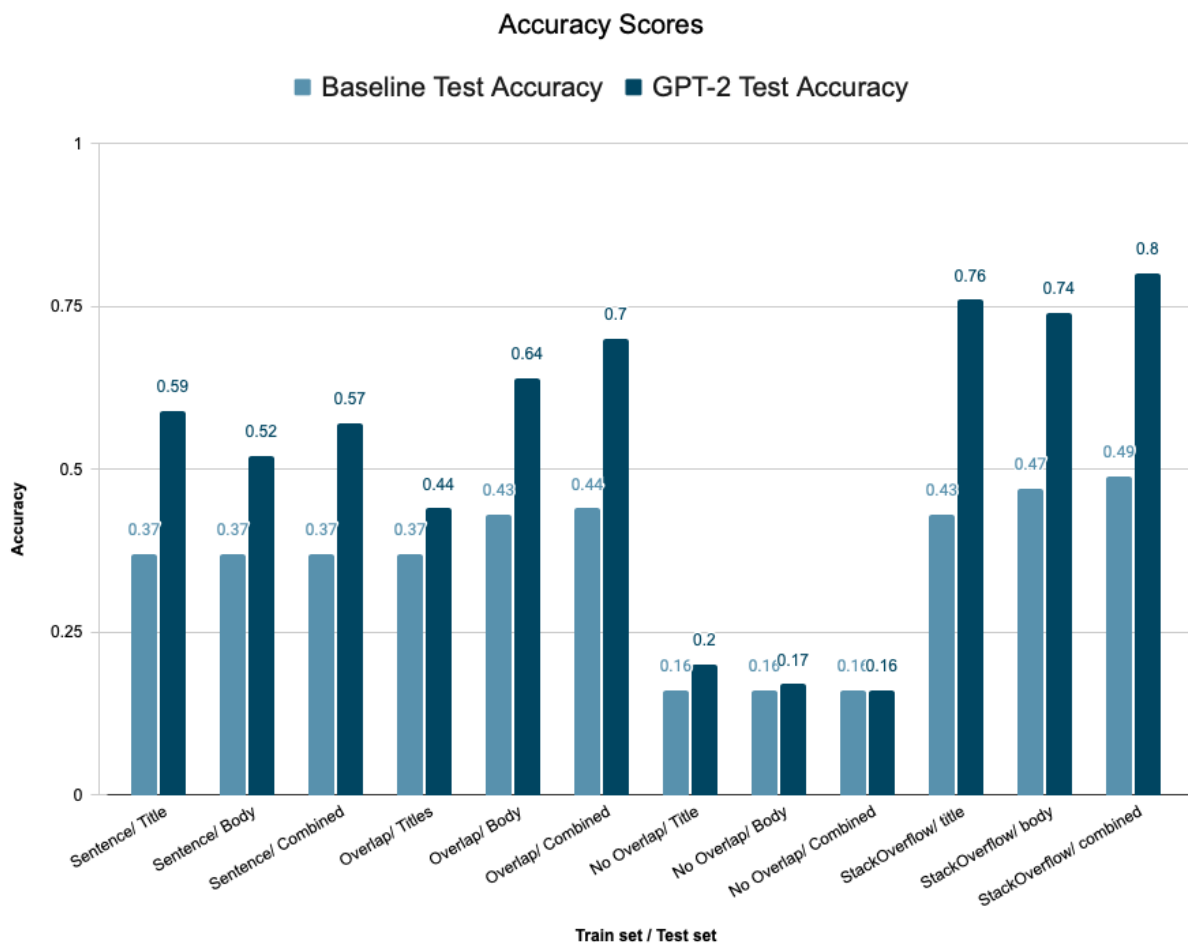
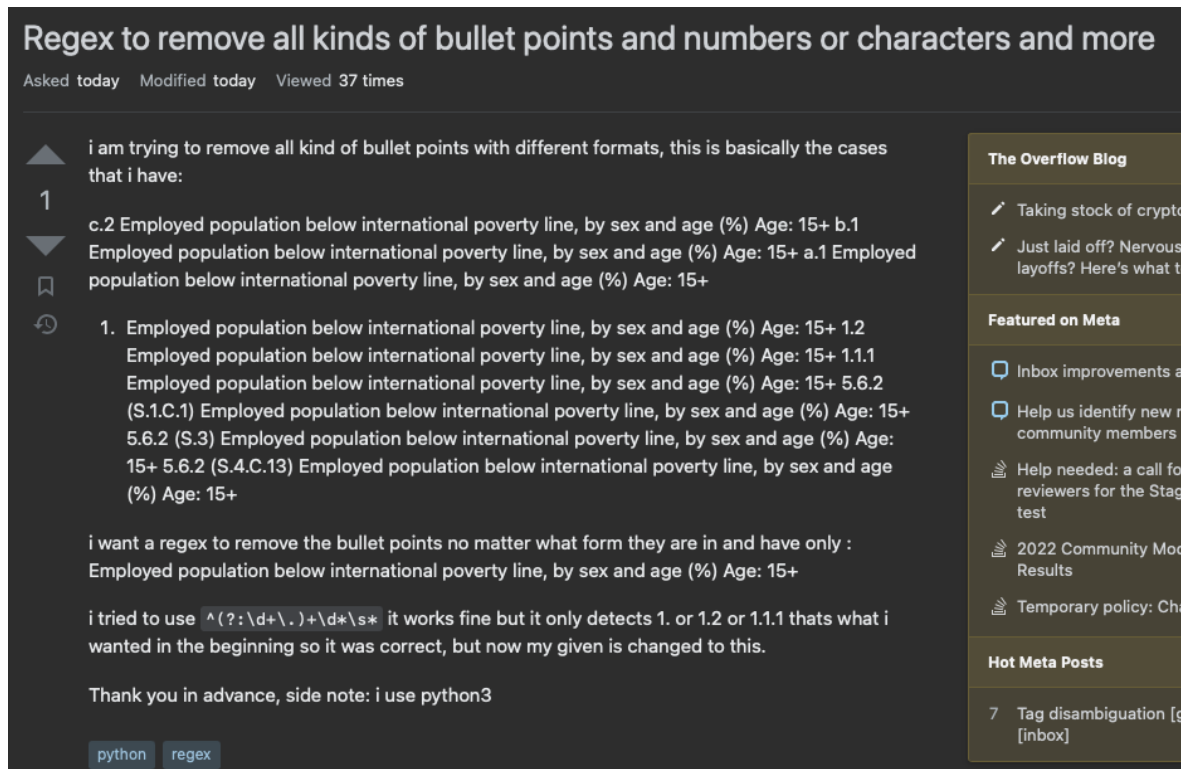


Figure - 8: The Accuracy scores of both models on all cases of training and test data.

8. Qualitative Results

After some analysis on the predictions of the model, we found that the model performs well when the StackOverflow question is well explained and belongs to one particular class, as opposed to having multiple labels. Figure 9 shows the best case scenario for the model to generate a correct prediction. This question is well explained and belongs to one particular class/label.



Regex to remove all kinds of bullet points and numbers or characters and more
Asked today Modified today Viewed 37 times

i am trying to remove all kind of bullet points with different formats, this is basically the cases that i have:

- 1

c.2 Employed population below international poverty line, by sex and age (%) Age: 15+ b.1 Employed population below international poverty line, by sex and age (%) Age: 15+ a.1 Employed population below international poverty line, by sex and age (%) Age: 15+

1. Employed population below international poverty line, by sex and age (%) Age: 15+ 1.2 Employed population below international poverty line, by sex and age (%) Age: 15+ 1.1.1 Employed population below international poverty line, by sex and age (%) Age: 15+ 5.6.2 (S.1.C.1) Employed population below international poverty line, by sex and age (%) Age: 15+ 5.6.2 (S.3) Employed population below international poverty line, by sex and age (%) Age: 15+ 5.6.2 (S.4.C.13) Employed population below international poverty line, by sex and age (%) Age: 15+

i want a regex to remove the bullet points no matter what form they are in and have only :
Employed population below international poverty line, by sex and age (%) Age: 15+

i tried to use `^(?:\d+\.)+\d*\s*` it works fine but it only detects 1. or 1.2 or 1.1.1 thats what i wanted in the beginning so it was correct, but now my given is changed to this.

Thank you in advance, side note: i use python3

python regex

The Overflow Blog

- Taking stock of crypto
- Just laid off? Nervous layoffs? Here's what to

Featured on Meta

- Inbox improvements an
- Help us identify new re community members
- Help needed: a call for reviewers for the Stagi test
- 2022 Community Mod Results
- Temporary policy: Cha

Hot Meta Posts

- 7 Tag disambiguation [g [inbox]

Figure - 9: An Example of the best case scenario for the model.

The model was struggling with StackOverflow questions which had multiple topics in them, for example, in figure 10, the question talks about exceptions in regular expressions, the model struggled to predict the right label for these kinds of questions.

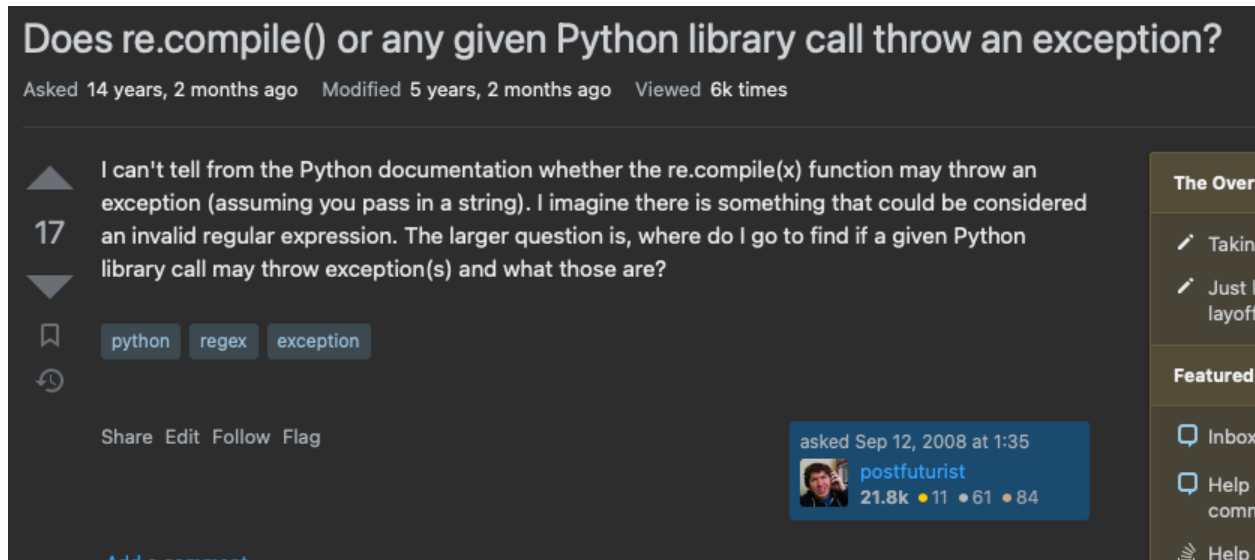


Figure - 10: An Example of a StackOverflow question where the models struggle.

9. Discussion & Learnings

From both our quantitative and qualitative results it can be seen that the main GPT-2 model performs quite well for the given problem. In the final training with the overlapped training set with the addition of stack overflow questions the GPT-2 model was able to achieve an accuracy of 80% which was well over the baseline. We would consider this good performance keeping in mind that this task did not have a large pre-collected data source and all the data used for training was manually scrapped and labeled. Another decision that was made was to use slightly different training data compared to the test data. This decision was made because if we fed a large sample of questions from stack overflow to the model there would be too much noise as the questions are not bound by any conditions and can have large overlap between labels. Hence the training on the documentation data alongside some questions we felt was the best approach for this problem as the documentation data had more structure and contained several key words.

The model's behavior to the variations of test data falls in line with our thinking- providing more information about the question gives us a higher accuracy on the test set.

A surprising insight in our project was using the same training data formatted in different ways we were able to achieve such high variation in accuracy. This shows us the importance of good data and also its formatting while training. We also realized the difficulty of scrapping, cleaning and labeling data, as this was the most time consuming part of the project.

If we were to do a similar project we would give more importance to data rather than the model. As we feel high quality data can have a larger impact than the pre trained model we choose.

10. Individual Contribution

Rohith- scraped the training data, created the stackoverflow case of the training data, Scraped and cleaned the test data, and trained the GPT-2 model. Performed code refactoring and added instructions in readme.

Vishal- processed the training data into a dataset(cleaning/ organizing/ labeling) and created the different variations, processed the cleaned test data to match the format of the training data and trained the baseline model.

11. References

[1] N. Imtiaz, A. Rahman, E. Farhana and L. Williams, "Challenges with Responding to Static Analysis Tool Alerts," *2019 IEEE/ACM 16th International Conference on Mining Software Repositories (MSR)*, 2019, pp. 245-249, doi: 10.1109/MSR.2019.00049.

[2] Gustavo H. Pinto and Fernando Kamei. 2013. *What programmers say about refactoring tools? an empirical investigation of stack overflow*. In *Proceedings of the 2013 ACM workshop on Workshop on refactoring tools (WRT '13)*. Association for Computing Machinery, New York, NY, USA, 33–36. <https://doi.org/10.1145/2541348.2541357>

[3] Tamara Lopez, Thein T. Tun, Arosha Bandara, Mark Levine, Bashar Nuseibeh, and Helen Sharp. 2018. *An investigation of security conversations in stack overflow: perceptions of security and community involvement*. In *Proceedings of the 1st International Workshop on Security Awareness from Design to Deployment (SEAD '18)*. Association for Computing Machinery, New York, NY, USA, 26–32. <https://doi.org/10.1145/3194707.3194713>

[4] T. Lopez, T. Tun, A. Bandara, L. Mark, B. Nuseibeh and H. Sharp, "An Anatomy of Security Conversations in Stack Overflow," *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 2019, pp. 31-40, doi: 10.1109/ICSE-SEIS.2019.00012.

[5] S. Nadi, S. Krüger, M. Mezini and E. Bodden, ""Jumping Through Hoops": Why do Java Developers Struggle with Cryptography APIs?," *2016 IEEE/ACM 38th International Conference on Software Engineering (ICSE)*, 2016, pp. 935-946, doi: 10.1145/2884781.2884790.

[6] Christoph Treude, Ohad Barzilay, and Margaret-Anne Storey. 2011. *How do programmers ask and answer questions on the web? (NIER track)*. In *Proceedings of the 33rd International Conference on Software Engineering (ICSE '11)*. Association for Computing Machinery, New York, NY, USA, 804–807. <https://doi.org/10.1145/1985793.1985907>

[7] Chris Parnin and Christoph Treude. 2011. *Measuring API documentation on the web*. In *Proceedings of the 2nd International Workshop on Web 2.0 for Software Engineering (Web2SE '11)*. Association for Computing Machinery, New York, NY, USA, 25–30. <https://doi.org/10.1145/1984701.1984706>

[8] Tseng, C.-H., & Lin, J.-R. (2022). A semi-hierarchical clustering method for constructing knowledge trees from stackoverflow. *Journal of Information Science*, 48(3), 393–405. <https://doi.org/10.1177/0165551520961035>

[9] Python official documentation, <https://docs.python.org/3/library/index.html>

[10] StackOverflow, <https://stackoverflow.com>

[11] StackOverflow data explorer, <https://data.stackexchange.com/>

Permissions:

| | Rohith Pudari | Vishal Kanna |
|--------------------------------|---------------|-------------------------|
| permission to post video: | Yes | Wait till video is seen |
| permission to post report: | Yes | yes |
| permission to post source code | Yes | yes |