Automatically Mining And Suggesting Stack Overflow Answers For Compilation And Syntax Errors In Python Programming Language

Ayşegül Rana Erdemli
dept. of Computer Science and Engineering
Sabanci University
Istanbul, Turkey
aysegulrana@sabanciuniv.edu

Dilara Tekinoğlu
dept. of Computer Science and Engineering
Sabanci University
Istanbul, Turkey
dilaramemis@sabanciuniv.edu

Mustafa Göktürk Yandım
dept. of Computer Science and Engineering
Sabanci University
Istanbul, Turkey
mgokturk@sabanciuniv.edu

Nisa Defne Aksu
dept. of Computer Science and Engineering
Sabanci University
Istanbul, Turkey
nisadefne@sabanciuniv.edu

***Abstract*—Developing software free from any syntax and/or compilation errors is a crucial task. It is well-known in the software community that this is not always easy to accomplish considering how hard finding and solving errors can be. As a fact, most developers face errors while coding and search for solutions on the Internet. This process, of course, requires manual human-effort and delays the production time. Our motivation is to reduce this development effort by automating the process of finding and suggesting existing solutions for compilation and syntax errors; thus saving time. To do so, we used 298.000 Stack Overflow Python question-solution set to train a Doc2Vec model and created 200 manual test data to evaluate our model. Finally, our model provided good suggestions with 59% rate with a 79% similarity score.**

***Index Terms*—Stack Exchange API, Stack Overflow, Python, Doc2Vec, error**

I. Introduction

Developing a software or writing a code is not always straightforward. There are variety of tools such as programming languages, Integrated Development Environments (IDE's), libraries, source control mechanisms, packages etc. that aims to help out new and experienced developers during this process.

Unfortunately, using these tools alone or together do not always provide the expected output(s) or behavior(s). Compilation errors are one great example for such a case. Compilation errors are errors that emerge when a program is compiled before it is actually being executed [1]. These errors are usually prompted as cryptic messages to developers in a mixture of natural language and specific error codes. In fact, these error outputs can be challenging to interpret that they can discourage newbies (and even professionals) [2].

Luckily there are some tools that developers can use to deal with these compilation errors to come up with a valid solution. Among different options, most famous ones include firstly (and obviously) Google and secondly Stack Overflow [3]. A usual scenario to find a solution to an output error message is just to copy the message and search within one of these services. While this method usually works and gives correct solutions to corresponding error, it can be still challenging to find an exact match. In addition, this process can take time thus delay the development progress.

Problem statement. In this project, we attempt to create an automated tool specific for Python language to automatically extract a relevant solution for an output error message given by the compiler.

Contribution. We hope that our project can be used as a reference for creating an automatic tool such as an extension to a specific IDE that automatically provides relevant solutions to developers and make their development journey more pleasable and effective.

Availability. Data and methods used during this project are publicly available at *StackOverflowSuggestBot*

II. Method

A. Definitions

Our first step was to determine the datasets that we will be using for our project. Firstly we wanted to look at ready datasets available in GitHub and other platforms. Even though we found a nice dataset called StaQC, which was a dataset consisting question-code pairs from Stack Overflow, we later decided not to use it as we found a better way to collect data. By looking at different literature works, we found out that we can collect data by using Stack Exchange API and this data

was perfectly related to our idea of a dataset, including enough questions and their related information regarding Python. After preprocessing this data and making it ready for model, we focused on creating our "Doc2Vec" model. Then we continued with training, retrieving answers, evaluating what we had as a result and showing how well our model did. Down below are some definitions regarding this project and to understand its context better:

Definition 1:
Stack Overflow. Stack Overflow is a programming question and answer platform for professionals and hobbyists. It contains questions and answers on a variety of computer programming topics.

Definition 2:
Doc2Vec. Another extensively used technique is Doc2Vec, which makes an embedding of a document regardless of its length. Doc2Vec computes a feature vector for each document in the corpus, whereas Word2Vec computes a feature vector for each word in the corpus. The Doc2vec model is built on Word2Vec.

Definition 3:
Stack Exchange API. Stack Exchange is a rapidly expanding network of question and answer websites covering a wide range of topics, from software engineering to other fields. Stack Exchange creates libraries of high-quality questions and answers centered on the most significant topics in each field. Answers, comments, badges, events, questions, revisions, suggested modifications, user information, and tags may all be retrieved via the Stack Exchange API.

B. Data Collection

The data collection for training set of Stack Overflow Python Error Titles and the answer data has done by Python scripts that communicated with the Stack Exchange API [4].

Stack Overflow Question Data with tag *"Python"* included 298.024 distinct questions. After filtering with the word "error" we got left with 25.068 titles. The dataset had columns: [tags, is answered: True, view count, answer count (bigger than 0), score, last activity date, creation date, question id, link, title, accepted answer id]

The answer data was also retrieved by the StackEchange API [4]. It consisted of columns: [is accepted, score, creation date, answer id, question id, body, link]

C. Data Preprocessing

The dataset we obtained from Stack Exchange API was initially rich in information containing many columns which were not necessary for our research goals. We first, dropped those columns. In order to train our Doc2Vec model, we only apply preprocessing on "question title" field of our dataset.

We removed stop words in English as well as commonly used words in our questions such as "Python". We converted all characters of titles to lowercase. We also applied stemming for each title. As a last step, we filtered out the titles which do not contain the word "error".

D. Doc2Vec Model

As the model that will help us find the semantically close questions to developer test questions from Stack Overflow,we chose to use gensim's pretrained Doc2Vec model. Since we have specific domain (Python Error Titles) we did not use a gensim corpus but our Stack Overflow titles corpus which we have gotten with the Stack Exchange API [4].

```
train_corpus = pd.DataFrame()
train_corpus["title"] = list(read_corpus(train["title"]))
train_corpus["question_id"] = train["question_id"]
train_corpus[:2]
```

Output:

	title	question_id
0	([error,instal,geopanda,gdal,api,...]	54734667
1	([unicod,error,unicodeescap,decod,...]	37400974

```
model = gensim.models.Doc2Vec.Doc2Vec(epochs=20)
model.build_vocab(train_corpus["title"])
model.train(train_corpus["title"],
    total_examples=model.corpus_count,
    epochs=model.epochs)
```

Fig. 1: Model training code

After preprocessing (as described in II-C) of the training data, we trained our model with this preprocessed data. Finding a test dataset was challenging.This is because, test dataset needed to be consisted of natural language Python questions since we would not want a language gap between the test and the training sets. As four developers and Computer Science students in our team, we have formed about 200 test queries inspired by the actual Stackoverflow queries, each can be considered as a real and meaningful query coming from a developer. Table I shows several examples to these queries.

Pycharm: Unresolved reference error when I open a project
ImportError: 'setuptools.build meta' is not a module
sqlalchemy.exc.InvalidRequestError: Mapper without property
MacOS Big Sur NPM Error Can't find Python executable
Visual studio code error while fetching extensions

TABLE I: Sample developer queries

For finding the closest match from the training set, we used embeddings created by Doc2Vec model. We vectorized our test instances and found the closest embeddings from our model by applying cosine similarity metric.

E. Retrieving the Answers

After finding a good suggestion to the question, the next step is to find the answers for that from Stack Overflow. With the help of the dataset generated by using Stack Exchange API's

Answers option, which had up to best 3 solutions (determined from the highest voted answers) a direct link to the answers are given to the user. Other than the direct link, without the need of clicking the link and changing the window by going to Stack Overflow, the body of the answer was also converted to string from HTML using BeautifulSoup and printed for the user to benefit from. Figure 2 shows an example to retrieval of the answers.

Enter your question about python:
Google-trans-new throws an error previously working code in Python.
This question sounds similar to yours:
python google trans new throwing error code wo...
Links and explanations to answers to this question:
1. https://stackoverflow.com/a/68270197
It seems to be an error from the package google-trans-new that is known and already corriged. (Check this discussion for more information).
A new version of the module with the bugfix hasn't been released to pip yet. So you have to manually do the modification or wait for the newt version to be released.
2. https://stackoverflow.com/a/69302211

```
from bs4 import BeautifulSoup
from bs4.formatter import HTMLFormatter
from googletrans import Translator
import requests

translator = Translator()

class UnsortedAttributes(HTMLFormatter):
    def attributes(self, tag):
        for k, v in tag.attrs.items():
            yield k, v
```

Check this 2 alternatives for Python transtator code:
https://neculaifantanaru.com/en/python-code-text-google-translate-website-translation-beautifulsoup-library.html
or here:
https://neculaifantanaru.com/en/example-google-translate-api-key-python-code-beautifulsoup.html

Fig. 2: Retrieving answers

III. Results

For each data instance in our test dataset, we vectorized this instance and determined top 3 closest vectors from our trained Doc2Vec model. Manual evaluations showed that the least close one among these three vectors was generally not a good answer for the given test question. Therefore, we decided to provide top-2 answers as the suggestions for the user's query. After this step, as a team of four software developers, we manually evaluated whether the suggested Stackoverflow answers were actually related to the given test question or not. Each test result was labeled as true (indicating that the suggested answer is indeed a proper answer to the question) or false (otherwise). Examples to this instances can be found in figures 3, 4 and 5.


```
"80": {
    "Test_Input": "When combining two df in pandas,
        ValueError: Shape of passed values is (6, 6), indices
        indicate (4, 6) error",
    "The_Closest": {
        "ID": "22878 40560165",
        "Words": "22878 strange errro valueerror shape passed
            values 7...",
        "isValid": false,
        "Similarity": 0.6902309656143188
    },
    "Second_Closest": {
        "ID": "18696 60747603",
        "Words": "18696 pandas join value error shape passed
            value inc...",
        "isValid": true,
        "Similarity": 0.6592617034912109
    }
}
```


Fig. 3: Example 1


```
"183": {
    "Test_Input": "Pycharm: Unresolved reference error when I
        open a project",
    "The_Closest": {
        "ID": "13123 20479696",
        "Words": "13123 pycharm unresolved reference error
            ide opening...",
        "isValid": true,
        "Similarity": 0.8324739933013916
    },
    "Second_Closest": {
        "ID": "7868 68532634",
        "Words": "7868 keep getting unresolved reference error
            use ki...",
        "isValid": true,
        "Similarity": 0.747882068157196
    }
}
```


Fig. 4: Example 2


```
"40": {
    "Test_Input": "Windows 10 pip install dotenv error code 1",
    "The_Closest": {
        "ID": "6576 49328525",
        "Words": "6576 pip install dotenv error code 1 windows
            10",
        "isValid": true,
        "Similarity": 0.9484830498695374
    },
    "Second_Closest": {
        "ID": "11020 66552328",
        "Words": "11020 error somebody tell error code",
        "isValid": true,
        "Similarity": 0.9121130704879761
    }
}
```


Fig. 5: Example 3

After our manual evaluation on 200 developer generated questions, the following results in Table II were obtained.

Percentage of times when the first suggestion is true:	0.05
Percentage of correct suggestions:	0.59
Average vector similarity for correct suggestions:	0.79
Average vector similarity for the wrong suggestions:	0.53

TABLE II: Results

Average similarity scores for the suggestions being high shows that the model agrees with the developers on how similar the sentences are.

It shows that the model is promising because it does not randomly find good results, it generally find the good matches with high similarity score while the weak suggestions are not claimed to be so similar by the model. The model is also not so confident with the bad suggestions.

IV. Discussion

We are aware that our approach has certain limitations.

Firstly, although promising results were obtained, since the test set is so small and manually created by the team members, it is not as completely reliable . With a test set that has more queries and better quality data, better results might be obtained.

For our problem, we believe that Doc2Vec model is a useful approach. However, some other models can be tried to see if they would make a difference and may be a better fit for the project objective.

Retrieving answers and demonstrating them to the user can be done successfully by a program, instead of solely printing the results to the console. An IDE with a good user interface for the queries and answers would be a good improvement.

Acknowledgment

We want to thank to our Professor Anıl Koyuncu and Teaching Assistant Genco Coşgun for their continuous support and guidance during this project. We really appreciate their time and responsiveness that help us pursue our goals.

References

[1] Mesbah, A., Rice, A., Johnston, E., Glorioso, N., Aftandilian, E. (2019). DeepDelta: Learning to repair compilation errors. Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. https://doi.org/10.1145/3338906.3340455

[2] Becker, B. A., Denny, P., Pettit, R., Bouchard, D., Bouvier, D. J., Harrington, B., Kamil, A., Karkare, A., McDonald, C., Osera, P., Pearce, J. L., Prather, J. (2019). Compiler error messages considered unhelpful. Proceedings of the Working Group Reports on Innovation and Technology in Computer Science Education. https://doi.org/10.1145/3344429.3372508

[3] Stack Overflow. (2022). Stack Overflow. https://Stack Overflow.com/

[4] StackEchange. (2022). Stack Exchange API. https://api.Stack Exchange.com/