

UNIVERSITY OF NAMUR

SEARCH ENGINE FOR DEEP CODE SEARCH

PROGRAMMER'S GUIDE

SO Crawler

Aton KAMANDA

Last update May 2021



Contents

1	Preface	2
2	Development	3
3	Objective	3
4	Reproducibility	5
5	Server deployment	6
	5.1 GPU capabilities	7
6	Testing	9
7	Contact	10
8	Acknowledgments	10

1 Preface

To implement a program efficiently, developers can reuse previously written code snippets by searching through the web with natural language queries. Reusability is a fundamental concept in programming, most of the applications web service, devices we use tomorrow wouldn't have been possible without this practice.

Hence multiple search engines and websites have been created to deliver a user interface for programmers by providing them with the code they need as accurately as possible. But unfortunately, the task is complicated, the code is a language on its own it is difficult to make the machine understand what we are looking for with classical algorithms. most approaches mainly rely on the textual similarity between source code and natural language query which can lead to pretty poor results.

In the same way that if you wanted to translate a message from French to English, a program that took every French word in the sentence and transformed it into English would end up in poor-quality translations. (Although they are very close languages, imagine French - Japanese) the machine needs to have a semantic understanding of the sentences to perform well. It cannot just limit itself to basic word-to-word translations.

Such a thing as semantic understanding would have seemed far from reach a few years ago, but the recent revolution of artificial intelligence, and more particularly of deep learning in the field of natural language processing, has led to breakthroughs. The software I am to introduce you with this user guide aims to provide you with a conceptual and practical understanding of how you can use the research's latest advancements to provide efficient code faster and more accurately than ever before.

I would like to thank everyone who will read this hopefully well-written guide and wish you with all my heart that it can help you build great projects in the future and make you win precious time.

2 Developement

SO crawler is an application using NLP to retrieve code snippets based on natural language queries. The project is inspired by the recent research paper

[CodeBERT: A Pre-Trained Model for Programming and Natural Languages](#) which demonstrates that a deep learning model achieves state-of-the-art performance on both natural language code search and code documentation generation tasks.

The model was trained on [CodeSearchNet](#) dataset.

As mentioned in the paper the result obtained here on python can be applied to every programming language with appropriate fine-tuning. SOcrawler has been made to be adaptable to web application usage, as well as a visual code extension.

3 Objective

Code search is a very common activity in software development, to implement some functionalities, for example, render a certain web page developers usually search and reuse previously written code. As mentioned before the difficulty in code search is to take into account the high-level intent reflected in the natural language. Like mentionned in the preface just like replacing words sequence to sequence without any semantic processing doesn't provide a useful translation. Translating French to Japanese by just translating each word to one another won't work because they don't share the same semantic order. Similarly, Source code and natural language queries don't share common structures. But they can be semantically linked to one another for example a relevant snippet for the query.

"Read an object from an XML " could be

```
public static < S > S deserialize(Class c, File xml) {  
    try {  
        JAXBContext context = JAXBContext.newInstance(c);  
        Unmarshaller unmarshaller = context.createUnmarshaller();  
        S deserialized = (S) unmarshaller.unmarshal(xml);  
        return deserialized;  
    } catch (JAXBException ex) {  
        log.error("Error-deserializing-object-from-XML", ex);  
        return null;  
    }  
}
```

But non-deep learning approaches may not be able to return this code because it doesn't contain the keywords of the queries. Nevertheless an effective code search engine based on state-of-the-art deep learning can achieve that by semantic mapping. Semantic understanding can solve a lot of problems of information retrieval. We have already seen how useful this could be in cases where the user could find useful bits of code that do not contain the terms of his search, but it can also happen that the user simply makes a typing or spelling mistake, the semantic understanding algorithm is extremely efficient to solve these problems. Let's imagine that the user wants to build a sudoku solver, but without wanting to he writes "suku soler", SO Crawler has no trouble understanding what the user meant and keeps assigning a high probability to the fact that the sudoku solver's code is a good proposal, where tools often used for code searches such as Google or Stackoverflow return results without any link to the query.

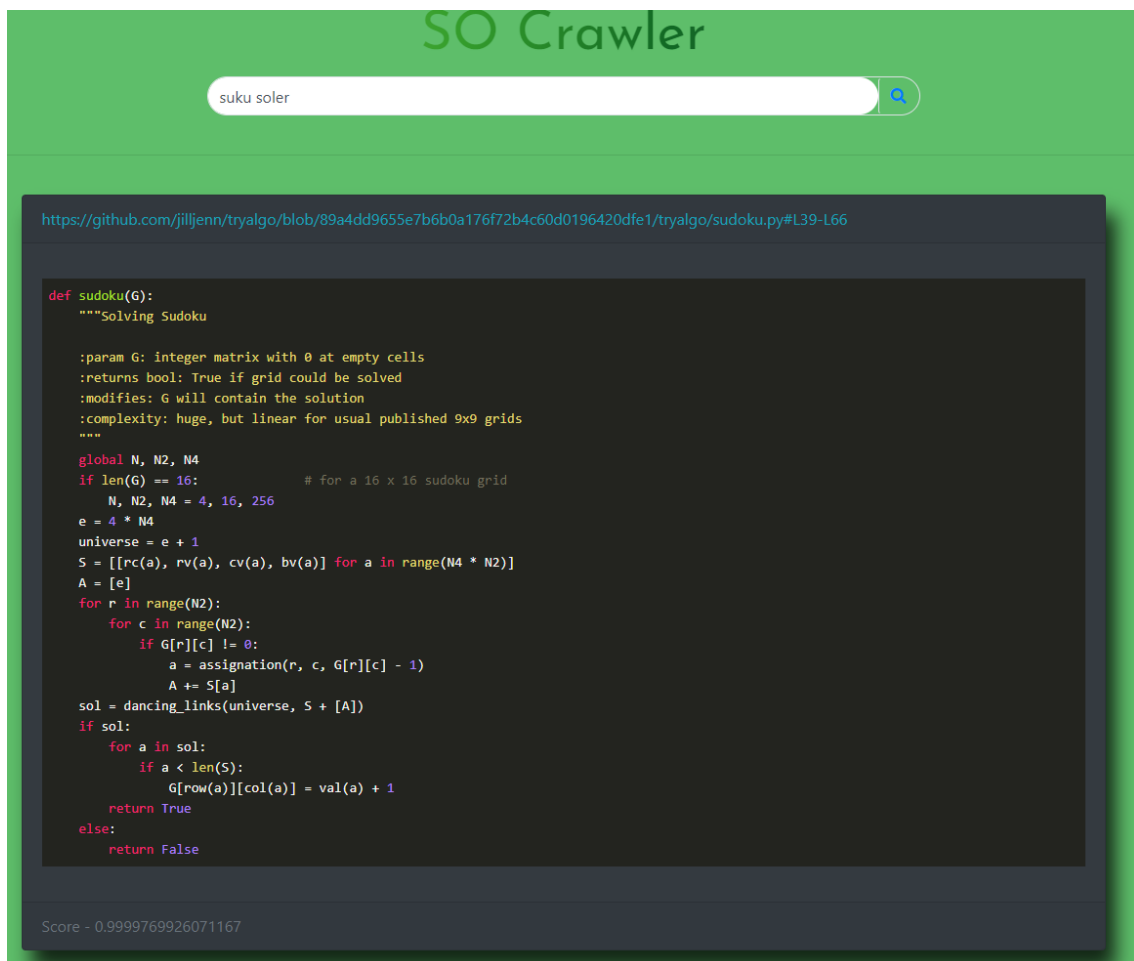


Figure 1: An example of the effectiveness of SO Crawler despite the misspelling of the user the search engine still manages to find the relevant piece of code.

4 Reproducibility

The project is entirely reproducible from the notebooks provided in the "notebooks" folder of the project, different resources also allow to reproduce the project and potentially reach similar results. The whole data cleaning, vectorization, training, and testing process has been resumed in the notebook.

Utilities such as data cleaning scripts, model checkpoints, or model hyperparameters can be found in utils, or are generated from the notebook. Transformers class used here can be found on <https://www.tensorflow.org/tutorials/text/transformer>. A detailed tutorial of how to link the trained models to a powerful web interface is available [here](#) and retraces the main steps to reproduce the project for ALBERT model. Before reproducing the project make sure your environments and libraries are in the right versions, a simple way to ensure this is to create an environment with all dependencies via [Anaconda](#). To create an environment with all dependencies do the following.

```
# global environnement containing torch,tensorflow,etc for the whole
project
conda env create -f environment.yml -n codesearch
conda activate codesearch
#tensorflow 1 for fine-tuning
conda env create -f tensorflow1.yml -n tensorflow1
conda activate tensorflow1
```

Multiple models have been tested and can be deployed according to your need. To fine-tune ALBERT you need to use tensorflow1.x or your program will crash and you also need to download the following https://tfhub.dev/google/albert_base/3 and proceed with the right paths.

```
python "./create_pretraining_data.py" \
  --input_file "./docstrings.txt" \
  --output_file "./tf_examples" \
  --vocab_file "./30k-clean.vocab" \
  --spm_model_file "./30k-clean.model" \
  --do_lower_case True \
  --max_seq_length 50 \
  --max_predictions_per_seq 8 \
  --random_seed 12345 \
  --dupe_factor 5
```

To fine-tune CodeBERT instructions are provided [here](#).

5 Server deployment

To deploy the server you just need to have the web folder, go inside, and do the following.

Windows powershell

```
$env:FLASK_APP = "app.py"
python -m flask run
```

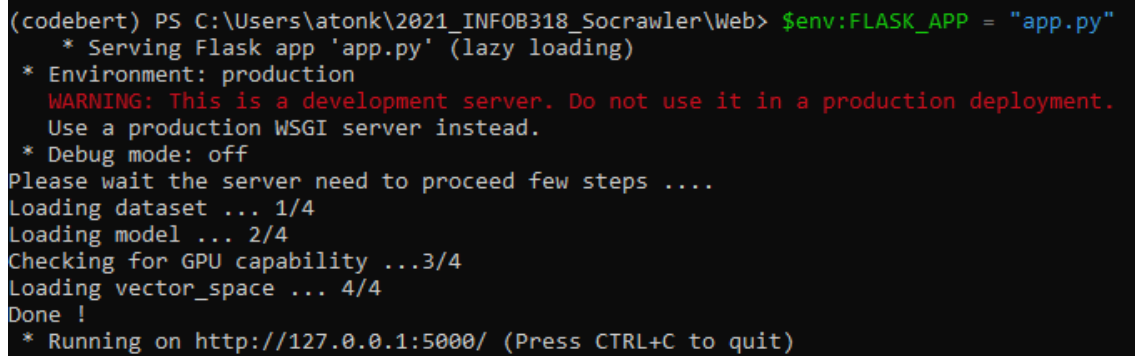
Linux/OSX bash

```
export FLASK_APP=app.py
flask run
```

Make sure that your folder structure looks like this, you need to have the .csv dataset, the vector space, and the model on your machine.

```
|  app.py
|  codegit.csv
|  vector_space_234853.pt
|
+---images
|     bigunamur.png
|     MEDIUM.png
|     unamur.png
|
+---python_model
|     config.json
|     pytorch_model.bin
|
+---static
|   +---css
|   |     main_page.css
|   |     results_page.css
|   |
|   \---img
|         unamur.png
|
+---templates
|     main_page.html
|     results_page.html
|
\---__pycache__
      app.cpython-38.pyc
```

Your terminal should indicate the progression of the server deployment so if something is missing you will notice it right away.



```
(codebert) PS C:\Users\atonk\2021_INF0B318_Socrawler\Web> $env:FLASK_APP = "app.py"
* Serving Flask app 'app.py' (lazy loading)
* Environment: production
  WARNING: This is a development server. Do not use it in a production deployment.
  Use a production WSGI server instead.
* Debug mode: off
Please wait the server need to proceed few steps ....
Loading dataset ... 1/4
Loading model ... 2/4
Checking for GPU capability ...3/4
Loading vector_space ... 4/4
Done !
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

Figure 2: The terminal during the execution of the flask server when it occurs correctly.

5.1 GPU capabilities

The project has been designed and thought to run on servers with GPUs to maximize efficiency, for this you will need to use CUDA available here <https://developer.nvidia.com/cuda-downloads> .

And then check that CUDA is well installed and that your GPU is well detected you can execute the following commands.

```
import torch
torch.cuda.is_available()
>>> True

torch.cuda.current_device()
>>> 0

torch.cuda.device(0)
>>> <torch.cuda.device at 0x7efce0b03be0>

torch.cuda.device_count()
>>> 1

torch.cuda.get_device_name(0)
>>> 'GeForce GTX 950M'
```

Note that CUDA is even more important if you plan to train your models, Codebert's fine-tuning can take up to 5 hours on very powerful GPUs

In the case of SO Crawler the ALBERT model was trained on one GTX 1080 3 GB and the CodeBERT model on the following hardware 4.

Task	Dataset Name	Language	Training Cost	Inference Cost
Clone Detection	BigCloneBench	Java	3 hours training on P100 x2	2 hours on p100 x2
	POJ-104	C/C++	2 hours training on P100 x2	10 minutes on p100 x2
Defect Detection	Devign	C	1 hour on P100 x2	2 minutes on p100 x2
Cloze Test	CT-all	Python, Java, PHP, JavaScript, Ruby, Go	N/A	30 minutes on P100-16G x2
	CT-max/min	Python, Java, PHP, JavaScript, Ruby, Go	N/A	1 minute on P100-16G x2
Code Completion	PY150	Python	25 hours on P100 x2	30 minutes on P100 x2
	GitHub Java Corpus	Java	2 hours on P100 x2	10 minutes on P100 x2
Code Repair	Bugs2Fix	Java	24 hours on P100 x2	20 minutes on P100 x2
Code Translation	CodeTrans	Java-C#	20 hours on P100 x2	5 minutes on P100 x2
NL Code Search	CodeSearchnet, AdvTest	Python	5 hours on P100 x2	7 minutes on p100 x2
	CodeSearchNet, WebQueryTest	Python	5 hours on P100 x2	1 minute on P100 x2
Text-to-Code Generation	CONCODE	Java	30 hours on P100 x2	20 minutes on P100 x2
Code Summarization	CodeSearchNet	Python, Java, PHP, JavaScript, Ruby, Go	On average, 12 hours for each PL on P100 x2	On average, 1 hour for each PL on p100 x2
Documentation Translation	Microsoft Docs	English-Latvian/Danish/Norwegian/Chinese	30 hours on P100x2	55 minutes on P100x2

Figure 3: Execution time for each fine-tuning task in CodeBERT.

938676

3294

Netherlands

ssh4.vast.ai:18676

ROMED8

PCIe 3.0, 16x 12.0 GB/s

↑973.4 Mbps

↓455.3 Mbps

Age:

1h 41m

STOP...

DESTROY...

CONNECT

2x Tesla V100

26.4 TFLOPS

16.2 GB

AMD EPYC 7282 ...

Storage

23.2 DLPerf

Max CUDA: 11.2

659.3 GB/s

10.7/32 cores 43/129 GB

1925 MB/s 40.4 GB

17.7 DLP/\$/hr

GPU: 98.0%, 68.0C

Status: Successfully loaded pytorch/pytorch

\$1.308/hr

Figure 4: Hardware specifications on which SO Crawler last model was trained.

6 Testing

Most of the modules of SO crawler have been subject to tests via asserts and pre/post notably via the classes put in opensource by Tensorflow for example the following class to test our tokenizer

```
class TokenizationTest(tf.test.TestCase):

    def test_full_tokenizer(self):
        vocab_tokens = [
            "[UNK]", "[CLS]", "[SEP]", "want", "##want", "##ed", "wa", "un",
            "runn",
            "##ing", ",",
        ]
        with tempfile.NamedTemporaryFile(delete=False) as vocab_writer:
            if six.PY2:
                vocab_writer.write("".join([x + "\n" for x in vocab_tokens]))
            else:
                contents = "".join([six.ensure_str(x) + "\n" for x in vocab_tokens])
                vocab_writer.write(six.ensure_binary(contents, "utf-8"))

            vocab_file = vocab_writer.name

        tokenizer = tokenization.FullTokenizer(vocab_file)
        os.unlink(vocab_file)

        tokens = tokenizer.tokenize(u"UNwant\u00E9d,running")
        self.assertEqual(tokens, ["un", "##want", "##ed", ",", "runn",
            "##ing"])

        self.assertEqual(
            tokenizer.convert_tokens_to_ids(tokens), [7, 4, 5, 10, 8, 9])
```

The SO Crawler model was also tested on the mrr benchmark <https://github.com/microsoft/CodeBERT/blob/master/CodeBERT/codesearch/mrr.py> and gave a result of 0.27 which shows that as explained in the paper these methods are state-of-the-art.

7 Contact

If you have any further questions, problems with the application, or simply want to download the elements that allowed the creation of this project that are too large to be available on Github you can contact me via atonkamanda@hotmail.com.

Although it is possible to generate from the notebooks, I could provide you the weights of the different models used, the architecture of the model, the vector spaces, and the dataset after data cleaning. Tokens of the CodesearchNet dataset have been serialized via Pickles and can be provided too.

Don't hesitate to use the issues section of the Github repo. https://github.com/UNamurCSFaculty/2021_INF0B318_Socrawler/issues and feel free to contribute.

8 Acknowledgments

I would like to thank [Martin Weyso](#) PhD student at the University of Montreal and the University of Namur and [Vincent Englebert](#) leading professor at the University of Namur without whom this project would not have been possible.

Thanks to [Natassia Schutz](#) leading professor at the University of Namur for correcting the first version of this guide.

I would like to thank especially [Daya Guo](#) and [Feng Zhangyin](#) researchers at Microsoft research who helped me with CodeBERT when I had problems. As I want to become an AI researcher myself, the fact that I could help them to find bugs and that they are so open to the discussion was a real pleasure. A big thank you to [Nathan Cooper](#) PhD. at the College of William and Mary who also helped me better understand some elements of codebert and transformers in general.

And finally a big thank you to everyone who provided resources that were useful to me to complete this online project. I hope to have returned the favor to others who will follow with this guide.