Published by :
http://www.ijert.org

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

Auto Code Generation for ML Deployment

Annapurna Ramesh
Computer Science and Engineering
Rashtreeya Vidyalaya College of Engineering ,RVCE
Bangalore, India

***Abstract*— Auto code generation is a machine learning service that helps automate the build of the ML pipelines and the ML deployment. It help reduce the effort for developers and helps them utilize that in building the model. First we'll go through the Introduction to Auto code gen and what we expect as the end result. To understand the tool we need dive into the concepts of ML deployment and how its is done and the most robust way of ml deployment is also described. Later, we dive into the details of the auto code generation and how it works as a service for the build of the ML deployment. Through this paper we will describe the requirements of the components and how the integration is done in the section call methodology. In methodology we'll highlight the High level design that is necessary to make the application work.**

I. INTRODUCTION

Auto Code Generation Tool is a fully managed machine learning service. With Auto Code Gen, data scientists and developers can quickly and easily build and create a deployment folder which can be deployed later by them into a production environment. This avoids the data scientists from the task duplication. They can focus on the model rather than the deployment aspects of scripts. This automation tool helps us to analyze the the components required for the model based on user-preference and then helps us build and visualize the files generated and download the files ready for production. This whole process of auto code generation is an iterative process and they can change the requirements of the project to be created on runtime. Auto Code Generation is a web app which uses Vue JS as front-end and Django as backend. This web app provides all the tools you need to take your models from experimentation to production while boosting your productivity. In a single unified visual interface, customers can help you to do the following things. First being, building a Docker file. It helps you add the required configurations that are required for deployment. Next, it helps inference through flask in the form of APIs. We can also choose whether to serve the model or not, this describes the necessity for flask and inference. If we decide not to serve the model then we can test the model. This is also an option given in the Auto Code Generation tool. Through swagger and other tools we can choose to add API documentation this is to test the inference of the model. So the whole is a collective combination of an ml service for deployment. We require the whole folder structure as the end result which consists of all the chosen configurations

II. REQUIREMENTS FOR ML DEPLOYMENT

In this section we will define the terminologies in the ML deployment and why they are used in the real world. There benefits and use cases.

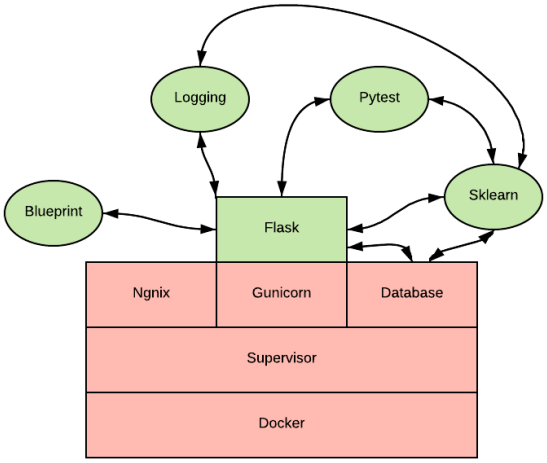


Fig 1. ML Deployment stack

A. Gunicorn

Gunicorn also coined as Green Unicorn is a WSGI or a web server gateway interface implemented so as to run python web apps. Gunicorn is one of many WSGI server implementations, but it's particularly important because it is a stable and is used in applications such as Instagram. Gunicorn implements the PEP3333 WSGI server standard specification so that it can run Python web applications that have an interface. Gunicorn is a pre-fork worker model this means that a master thread spins up workers to handle requests but does not control how those workers perform the request handling. Here, each worker is independent of the controller.

B. Nginx

Nginx comes under the BSD clause 2 type of license and it is free and open sourced. Nginx is mainly a web server which supports reverse proxy, load balancing, mail proxy along with HTTP cache support. A load balancer is nothing but distribution of resources to make things more efficient and faster. Nginx also solves the C10K problem,which is handling number of clients through network sockets. C10K means handling 10K connections at the same time. Nginx unlike the traditional servers do not handle requests through threads but they do so with a more scalable asynchronous event-driven architecture. This kind of architecture uses small but adequate amount of memory in a load. If you don't expect many connections as thousands of them you can still benefit from Nginx's small memory footprint and high performance. Nginx is a software that scales from as small as VPS to large clusters.

To interface your API REST with the world, you need Gunicorn and Nginx. Docker, Gunicorn, Nginx and the

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

swagger documentation have a weird interaction in this project,
Gunicorn launches the python flask server entry point with the command Gunicorn app:app -b 0.0.0.0:5000 . Gunicorn listen on the port 5000.Nginx configuration is as follows:

```
server {
listen 5001;location / {
proxy_pass http://0.0.0.0:5000;
proxy_set_header Host $host:5000;
proxy_set_header X-Real-IP $remote_addr;
proxy_set_header X-Forwarded-For
$proxy_add_x_forwarded_for;
}
}
```

Nginx is listening on the port 5001 and proxy to 5000, to Gunicorn server. Docker listens the port 5000 from outside the docker and sends it to 5001 and everything work fine:
world --5000--docker--5001--nginx--5000--gunicorn
Nginx is made to listen on the 5001 and not 5000 port because Gunicorn was redirecting everything to his proper port and it was not possible to use the nice swagger documentation.

C. Flask (WSGI server)

Flask is a light WSGI web application framework. Flask is written in such a way that it is quick and easy to get started with a simple application and scale up to a complex application. It is a simple wrapper around Werkzeug and Jinja and has become one of the most popular Python web application frameworks. Flask offers suggestion but doesn't make anything mandatory its up to the user to choose which library to implement. There are also many extensions provided to make development a lot easier and simpler.

D. Supervisor

Supervisor is a client/server system that allows its users to control a number of processes on UNIX-like operating systems. It has two main components. The client and the server.
The Server piece is called the supervisord. Supervisord is responsible for starting the child processes or programs when it is invoked. It is also responsible for responding to client commands, restarting exited or crashed subprocess,logging the output of the subprocess using stdout and stderr and handling and generating events corresponding to the points in the lifetime of the subprocess. This server process uses a configuration file usually located in /etc/suppervisor.conf.
The client piece is called the supervisorctl. It provides access to the supervisord through a shell like interface. From supervisorctl a user can connect to different process of supervisord(only one a time), get access to the status of the supervisord subprocess along with the lists of the running process of a supervisord.
Through supervisor, we will launch 4 processes:

- Gunicorn
- nginx
- database-server

You can easily restart all of them with the command: supervisorctl restart all

E. Docker

Developing apps today requires developers to do something more than just writing code. Those being multiple architecture, frameworks, languages all these components gives way for enormous complexity. Docker is built to simplify and accelerate the work flow while giving a choice and freedom to developer to pick the tools, stack and deployment environment of their choice. Docker is a program for virtualization, it permits to manage "containers", that pack together virtual OS, software and its dependencies. Containers are easy to set up and deploy. There are two steps in the "dockerization", the installation and the setup. These two steps can be separated in two files: the Dockerfile and the docker-compose.yml. Dockerfile are the instruction to the docker to build an image. A Dockerfile consists of all the commands a user would call on the command line to assemble an image. Using the docker build one could build the image by executing several commands on the command line.

F. Loggers

In reality, logging is important. When you transfer money, there are transfer records. When an airplane is flying, black box (flight data recorder) is recording everything. If something goes wrong, people can read the log and have a chance to figure out what happened. Likewise, logging is important for system developing, debugging and running. When a program crashes, if there is no logging record, you have little chance to understand what happened. For example, when you are writing a server, logging is necessary log any data in python for debugging we need to add the following lines:

```
import logging
logging.config.fileConfig('logging.conf')
log = logging.getLogger(__name__)
```

Where logging.conf is a configuration file. There are 5 logging levels defined in the logging.conf. Those being:

```
log.critical('text')
log.error('text')
log.warning('text')
log.info('text')
log.debug('text')
```

G. Pytest

Testing is a tool which is essential for any developer. The Python community embraces testing, and also has the vaious libraries to support testing. In the larger Python ecosystem, there are a lot of testing tools. Pytest stands out because it handles most complex test case and is easy to write these scripts for the test cases. All functionalities of a main function should be tested in all the possible configurations and the python module Pytest has been made for it. This script is a very simplified example of testing and in real life each function should be tested separately. For the simplicity, here, we test train and predict in the same Pytest function. Pytest-flask is a python module to test flask application via Pytest. To do so, we just have to add a file conftest.py that initializes the app decorated with @pytest.fixture: Key elements for a robust predictive application will require to:

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

- use a real HTTP server instead of the testing server provided with flask
- put some testing in place to ensure your code is correctly working, even when you re-factor and tweak it
- document your code, and your API
- track the inner workings through logging
- handle multiple tasks with a basic queuing system

H. Sklearn

Scikit-learn (formerly scikits.learn and also known as sklearn) is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, *k*-means and DBSCAN, and is designed to inter-operate with the Python numerical and scientific libraries NumPy and SciPy. There are three parts of sklearn:

- Transformer in scikit-learn ar those that have fit and transform method, or fit_transform method.
- Predictor is a class that has fit and predict methods, or fit_predict method.
- Pipeline is just an abstract notion, it's not some existing ml algorithm. Often in ML tasks you need to perform sequence of different transformations of raw dataset before applying final estimator.

I. Blueprint (Flask Blueprint)

Both flask and flask blueprint have resources such as static fies, templates, and views that are associated with routes. But the only difference is Flask Blueprint is not an application and needs to be registered before use. So to register a blueprint we use register_blueprint(). In a blueprint we can define an URL prefix that routes all the URL in a blueprint. Apart from this there are cases where we use blueprints, those being:

- Factor an application especially big applications
- Register a blueprint with different URL rules
- Provide filters, templates through blue print

III. REQUIREMENTS FOR AUTO CODE GENERATION

After understanding the code required for the ml deployment we now move into the requirements of the actual application the Auto Code Gen.

A. Django (Python)

Django is a High-level python web framework that supports quick development along with a clean and pragmatic design. It is built by experienced developers and is hassle free to develop on this platform. It is also free and open source apart from being ridiculously fast. It follows the model-template-view (MTV) architectural pattern and is maintained by the Django Software Foundation (DSF), an independent non-profitable organization. Django eases the task of several websites that are database driven and has features such as re-usability and pluggability. This means that less coding effort is required and low coupling hence providing rapid development. The language Django used is Python and the entire set of files created by django is in python. Also the Django consists of the admin model where an interface is provided to insert, delete, update and view the users or admins of the application.

B. AST (Python)

AST parser is the one which compiles the python code into an AST or Abstract Syntax Tree format. This tree format can be easily traversed and are in th format of nodes. They work as a tree so each node has children or sub nodes. In AST we write a self calling function called as a Transformer and this helps us traverse the tree by calling its child nodes and also we can add code in this by using the append method. Finally we need to use the fix_missing method of the AST for perfect addition of code without any discrepancies that is not being in the AST format. We then use the AST Decompiler to get the original code and do a write back at the respective file which had to be changed. So the AST ensures seamless addition or modification of code in the python files. We use this to add additional APIs in the ML interface required as per user preference.

C. Vue (JavaScript)

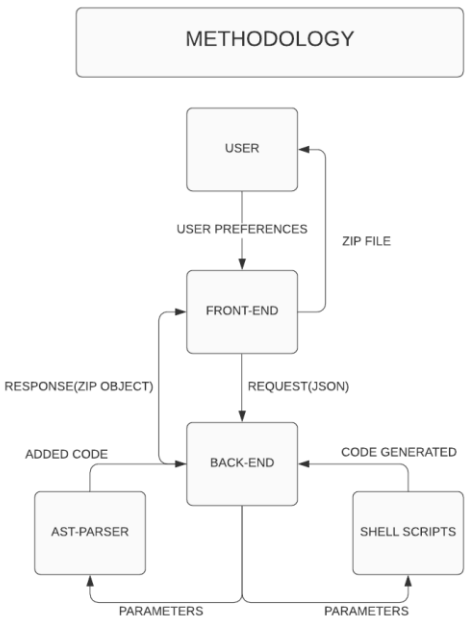
Vue JS is an open source framework used to build interfaces and single-page interfaces. Vue JS follows an incremental approach with a component base system, where the page is broken down into components. Advanced features like routing, state management are also a part of Vue JS. It also helps you to implement HTML features and attributes and are called directives. The directives come as pre-built features in Vue JS.

D. Shell Scripts

These are bash scripts which contain commands that are to be executed on the command line. These commands are written to fulfill the purpose to generate code. They are also used to copy files into the zip file. These commands run using the subprocess pipeline which is available in python. They are used to execute commands in a pipelined manner in the command line by a spanned sub process.

IV. METHODOLOGY

After understanding the code required for the ml deployment and the requirements of the actual application the Auto Code Gen. Here we highlight on the methodology we implemen
t



International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

1) The user using the interactive front-end gives the preferences and on completion of the preferences to view the code so far generated he/she clicks on the view button. On click an event in generated

2) On recording or storing the user preferences on interaction by the user with the front-end(Vue js) it gives the displays required. Only when the user wants to view the code he/she triggers an event on button click. This button click collects all the Information so far stored in the Vue js using state management and then makes a post request to the back end.

3) On receiving the post request the back-end extracts the required fields from the request body of the post request made using axios in the front-end(Vue js). Then a conditional check is done on the type of docker configurations and the requirements to deploy the ML model all the details are derived from the JSON sent by an axios request from the front end.

4) While addressing the conditional checks from request body parameters. The back-end triggers the shell scripts which generate the code . These bash files generates the files required for the ML deployment.

5) The code from the shell scripts comes back to the back-end where code can be added or modified using the ast-parser.

6) After all the code alterations are done the code is then made into zip object that are sent to the front-end and then here it is rendered as cards in the front end

7) The user on receiving these cards can download the zip or make changes and then download the modified zip to the local desktop

V. CONCLUSION AND FUTURE WORK

By this project we can concluded that our expected output will always be a zip file with all the code generated and also it is very flexible to edit this code or our selections before downloading and deploying. So we conclude that the auto code generation tool worked as required and can improve. The future enhancements could be the changes that can build upon this objective of auto code generation. The enhancements are , Extensible to training modules automation, that is building the train modules along with the existing deployment scripts. Hence making the entire ML Development an autonomous task. This is one of the major and most required task to make a fully fledged ML development task autonomous.

ACKNOWLEDGMENT

We are indebted to our guide**, Dr. Krishnappa H.K,** designation**, Dept of CSE** for his/her wholehearted support, suggestions and invaluable advice throughout our project work and also helped in the preparation of this thesis. We would also like to extend our gratitude to **Shivaram K.R, CEO, Curl Analytics and Utkarsh Vardhan, Data Scientist, Curl Analytics** for their guidance in the mechanics of the project.

We also express our gratitude to our panel members **Dr.Vinay Hegde (Associate Professor), Dr. Rajashree Shettar (Professor), Dr. Poonam G (Associate Professor),** Department of Computer Science and Engineering for their valuable comments and suggestions.

Our sincere thanks to **Dr. Ramakanth Kumar P.**, Professor and Head, Department of Computer Science and Engineering, RVCE for her support and encouragement.

We express sincere gratitude to our beloved Principal, **Dr. K. N. Subramanya** for his appreciation towards this project work.

We thank all the **teaching staff and technical staff** of Computer Science and Engineering department, RVCE for their help.

Lastly, we take this opportunity to thank our **family** members and **friends** who provided all the backup support throughout the project work.

REFERENCES

[1] Di Yang, Aftab Hussain, Cristina Videira Lopes “From Query to Usable Code: An Analysis of Stack Overflow Code Snippets ”, 2016 IEEE/ACM 13th Working Conference on Mining Software Repositories.

[2] A. Bacchelli, L. Ponzanelli, and M. Lanza. Harnessing stack overflow for the ide. In Proceedings of the 2012 Third International Workshop on Recommendation Systems for Software Engineering (RSSE), pages 562–567. ACM, 2013.

[3] F. J. Budinsky, M. A. Finnie, J. M. Vlissides, and P. Yu. Automatic code generation from design patterns. IBM Systems Journal, 35:151–171, 1996.

[4] [4] G. Frederick, P. Bond, and S. Tilley. Vulcan: A tool for automatically generating code from design patterns. In Proceedings of the 2nd Annual IEEE Systems Conference, pages 1–4, 2008.

[5] R. E. Gallardo-Valencia and S. Elliott Sim. Internet-scale code search. In Proceedings of the 2009 ICSE Workshop on Search-Driven Development-Users, Infrastructure, Tools and Evaluation, SUITE ’09, pages 49–52, Washington, DC, USA, 2009. IEEE Computer Society.

[6] P. Morrison. Is programming knowledge related to age? an exploration of stack overflow. In Proceedings of the 10th IEEE Working Conference on Mining Software Repositories (MSR), 2013, pages 69–72. IEEE, 2013.

[7] S. M. Nasehi, J. Sillito, F. Maurer, and C. Burns. What makes a good code example? - a study of programming q and a in stackoverflow. In Proceedings of the 28th IEEE International Conference on Software Maintenance (ICSM), pages 102–111. ACM, 2012.

[8] J. Noble and R. Biddle. Patterns as signs. In Proceedings of the 16th European Conference on Object-Oriented Programming, ECOOP ’02, pages 368–391, London, UK, UK, 2002. Springer-Verlag.

[9] M. Ohtsuki, A. Makinouchi, and N. Yoshida. A source code generation support system using design pattern documents based on sgml. In Proceedings of the Sixth Asia Pacific Software Engineering Conference, APSEC ’99, pages 292–, Washington, DC, USA, 1999. IEEE Computer Society.

[10] K. Philip, M. Umarji, M. Agarwala, S. E. Sim, R. Gallardo-Valencia, C. V. Lopes, and S. Ratanotayanon. Software reuse through methodical component reuse and amethodical snippet remixing. In Proceedings of the ACM 2012 Conference on Computer Supported Cooperative Work, CSCW ’12, pages 1361–1370, New York, NY, USA, 2012. ACM.

[11] L. Ponzanelli, G. Bavota, M. D. Penta, R. Oliveto, and M. Lanza. Mining stackoverflow to turn the ide into a self-confident programming prompter. In Proceedings of the 10th IEEE Working Conference on Mining Software Repositories (MSR), 2014, pages 102–111. ACM, 2014.

[12] T. Suresh, C. Luigi, A. Lerina, and D. P. Massimiliano. An empirical study on the maintenance of source code clones. Empirical Software

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 9 Issue 05, May-2020

Engineering, 15(1):1–34, 2010. [13] W. Teitelman. PILOT: A Step towards Man-Computer Symbiosis. PhD thesis, September 1966.

[13] M. Umarji, S. Sim, and C. Lopes. Archetypal internet-scale source code searching. In B. Russo, E. Damiani, S. Hissam, B. Lundell, and G. Succi, editors, Open Source Development, Communities and Quality, volume 275 of IFIP – The International Federation for Information Processing, pages 257–263. Springer US, 2008.

[14] S. Wang, D. Lo, and L. Jiang. An empirical study on developer interactions in stackover Cow. In Proceedings of the 28th Annual ACM Symposium on Applied Computing), pages 1019–1024. ACM, 2013.

[15] E. Wong, J. Yang, and L. Tan. Autocomment: Mining question and answer sites for automatic comment generation. In Proceedings of the 2013 IEEE/ACM 28th International Conference on Automated Software Engineering (ASE), pages 562–567. ACM, 2013.

[16] Y. Zheng. 1.x-way architecture-implementation mapping. In Proceedings of the 33rd International Conference on Software Engineering, ICSE ’11, pages 1118–1121, New York, NY, USA, 2011. ACM.