Music Predictive Analysis Project based on Lyrics

LEONARD MWANGI[1,*]

[1] School of Informatics and Computing, Bloomington, IN 47408, U.S.A.
[*] Corresponding authors: lmwangi@iu.edu

project-01, May 1, 2017

Being certain that lyrics of your song will lead to the next greatest hit would boost confidence to a lot of amateur artists who are faced with fears of never making it thus never attempting to make good their creativity. With Machine Learning (ML) this can be a thing of the past, these artists would have the ability to let ML models determine the viability of their lyrics becoming the next hit based on history of other songs that have made it to top. Through training, the model can certainly determine the outcome of different songs which will be depicted in this project.

Keywords: Cloud, I524

https://github.com/lmundia/sp17-i524/tree/master/project/S17-IO-3013/report/report.pdf

INTRODUCTION

When faced with the decision to forward their song to a recording company, amateur artists find it daunting due to uncertainty of whether their song would be recorded and if it is if it will make them wealthy. Having ability to run the lyrics through a predictive analysis process that would determine the viability of the song making it would be a huge win and confidence booster to many artists. That prediction is achievable by use of machine learning and creating a model that takes already greatest his and trains it to determine what makes the song successful. This would be done by analyzing the lyrics, the locality and time of release.

In this project, we will utilize machine learning to help determine the viability of a song becoming the next greatest hit based on the lyrics, time of production, locality and the artist. The project will utilize the greatest hits of all time [1] to train a model which will then be used to analyze larger dataset of random songs [2] and provide an in-depth analysis of the next possible hit. The project will utilize a Hadoop cluster deployed on Chameleon Cloud using CloudMesh to accomplish this analysis.

ENVIRONMENT

To achieve the analysis, Cloudmesh was used in order to allow the end users ability to choose a cloud of their liking. Chameleon Cloud and Jetstream were used for initial testing but changing the default cloud on CloudMesh would allow deployment to other clouds.

Cloudmesh

Cloudmesh is an open source component aimed at delivering software-defined system. It's a collection of virtualized bare-metal infrastructure, networks, application, systems and platform software unifying different clouds as a Cloud Testbeds as a Service (CTaas) [1]. Cloudmesh was developed by Professor Gregor von Laszewski in collaboration with his team at Indiana University [2]. In this environment, cloudmesh is used to automate installation of a four-node cluster.

Ansible

Ansible is also an open source platform used in automation of deployments and tasks executions [3] . It has the ability to execute scripts to a remote system thus making it easy to administer and manage hosts farm. Ansible is written in Python thus it requires python to be installed in the remote host for successful executions.

Ansible Playboks

Deemed as the real strength of Ansible, a playbook is a collection of instructions that defines what Ansible will do when it connects to each host [4] . Playbooks are written in YAML an easy to read data serialization language [5] . In this project, Ansible is used to deploy the required components to successfully accomplished the deployment, configuration and analysis of the defined files.

Apache Spark

Apache Spark is an open-source data processing and analytics engine designed to handle large scale dataset. Spark was originally developed by UC Berkeley but has been adopted in the Apache software stack since 2013 [6]. Spark has the ability to run as part of Hadoop, Mesos, Standalone or in a cloud environment [7]. In this project, Spark is deployed as a standalone environment and primarily used to analyze music data to in order to

identify patterns and make predictions.

MongoDB

MongoDB is a NoSQL, open source database program aimed at hosting documents in JSON-like format. MongoDB is favorable in storing unstructured data since it does not follow the traditional RDBMS structure. Its performance is also paramount when coupling related unstructured data [8]. For this project, MongoDB become the ideal database application because the source data structure is unknow and it can span in many directions where one dataset doesn't resemble the next. Putting such data in a structured database program would be cumbersome.

Python

Python is used as the programming of choice in this project due to its extensibility and also as a requirement for Ansible to execute successfully.

DEPLOYMENT

Environment deployment is automated as mentioned earlier using Cloudmesh and Ansible scripts. The environment is deployed on a 4-node cluster with the following resources.

Resources Table

Table 1 resources.

Table 1. resources dedicated to the environment

Node	Role	Flavor
*Node*1	*Spark*	*m*1.*small*
*Node*2	*Mesos*	*m*1.*small*
*Node*3	*MongoDB*	*m*1.*medium*
*Node*4	*Analytics*	*m*1.*small*

Cloudmesh

Code snippet below generates 4 node cluster that will be used in the inventory file for the Ansible roles. In order to use Cloudmesh Cluster, the following package levels or higher must be installed

- Cloudmesh client 4.7.2
- Pip 8.1.2
- Python 2.7.12

Algorithm 1. cloudmesh code

```
\$ cm cluster define --count 4 --image CC-Ubuntu16.04
\$ cm cluster use cluster-001
\$ cm cluster allocate
\$ cm cluster nodes > inventory.txt
```

format inventory file

Define roles in the inventory file and also remove the resolved name. After formatting the inventory file, the server roles and their corresponding public IP address should look similar to the output below with the right IP address.

Algorithm 2. Inventory file

```
[spark\_server]
<ip address>

[mesos\_server]
<ip address>

[mongo\_server]
<ip address>

[data\_node]
<ip address>
```

Ansible

After configuring deployment of virtual machines and configuring the inventory.txt, Ansible playbooks are used to automate deployment of the required components to the nodes defined above. The following Ansible files are included: ansible.cfg – used to define default configurations for Ansible including role_path and remote user.

Algorithm 3. ansible config file

```
[defaults]
roles\_path = roles
host\_key_checking=false
callback\_whitelist = profile_tasks
remote\_user=cc
```

Playbooks

The following playbooks are utilized for deployment

1. predict.yml – main file for the playbooks, used to call all the other playbooks and define their locations. Role-based playbooks are contained in scripts directory of the environment.
2. spark.yml – used to define the host, roles location and execution of Spark installation playbook. The playbooks redirect the folder structure to /roles/ansible-role-spark.
    - Tasks directory - contains main.yml playbook that's used to define all the tasks for Spark installation.
    - Defaults directory – contains main.yml that defines the default requirements for Spark installation including the spark version and download site for the distribution.
3. mesos.yml – used to define the host, roles location and execution of MongoDB installation playbook. The playbooks redirect the folder structure to /roles/ansible-role-mesos.
    - tasks directory – contains main.yml playbook that checks the operating system to determine the flavor of mesos that would be deployed. The environment contains playbooks for Debian and Redhat flavors.
    - defaults directory – contains main.yml which defines the version of mesos that will be downloaded and installed.

4. mongo.yml – used to define the host, roles location and execution of MongoDB installation playbook. The playbooks redirect the folder structure to /roles/ansible-role-mongo.
    - tasks directory - contains main.yml playbook that's used to define all the tasks for MongoDB installation. The file also includes task to create a database and initial user for the Mongo environment.
    - defaults directory – contains main.yml that defines the default requirements for MongoDB installation including the MongoD version and download site for the distribution.
    - vars directory – contains main.yml that defines the variables for the environment like data path, admin account and password.
    - template directory – contains mongod_config.j2 file which is used to define MongoDB configurations.
5. data.yml - used to define the host, roles location and execution of dataset playbook. The playbooks redirect the folder structure to /roles/ansible-role-dataset.
    - tasks directory - contains main.yml playbook that's downloads the MillionSongs dataset and copies into to MongoDB. The file also contains task to create collection and load data into MongoDB by executing a remote file myfiles.py

CONCLUSION

Ability for amateur artists, artists and record labels to quick determine the viability of a hit is paramount to their success and missed chances due to inexperience, fear of unknows, bad song or acting when time is not ripe can be costly. Machine learning has the ability to change these outcomes, a well-trained model can help determine with high accuracy where the song will end up.

REFERENCES

[1] G. von Laszewski, "Cloudmesh for beginners," WebPage, 2015. [Online]. Available: https://cloudmesh.github.io/introduction_to_cloud_computing/class/lesson/iaas/cloudmesh.html#cloudmesh-for-beginners

[2] H. L. H. C. G. C. F. Gregor von Laszewski, Fugang Wang, "Accessing multiple clouds with cloudmesh," Indiana University, School of Informatics and Computing, 919 E. 10th Street Bloomington IN 47408, U.S.A., techreport, Jun. 2014. [Online]. Available: https://www.researchgate.net/publication/266659258_Accessing_multiple_clouds_with_Cloudmesh

[3] networklore, "What is ansible?" Webpage, Apr. 2014. [Online]. Available: https://networklore.com/ansible/

[4] Red Hat, Inc., "Playbooks," Webpage, Apr. 2017. [Online]. Available: http://docs.ansible.com/ansible/playbooks.html

[5] O. Ben-Kiki, "Yaml ain't markup language (yaml™) version 1.2," Webpage, Oct. 2009. [Online]. Available: http://www.yaml.org/spec/1.2/spec.html

[6] A. Kattt, "Spark – lightning-fast cluster computing," Webpage, Nov. 2011. [Online]. Available: https://amplab.cs.berkeley.edu/projects/spark-lightning-fast-cluster-computing

[7] Apache Spark, "Lightning-fast cluster computing," Webpage. [Online]. Available: http://spark.apache.org/

[8] MongoDB Inc., "The mongodb 3.4 manual," Webpage. [Online]. Available: https://docs.mongodb.com/manual/

[9] labrosa, "Million song dataset," WebPage, 2012. [Online]. Available: https://labrosa.ee.columbia.edu/millionsong/

[10] "http://www.billboard.com/charts," WebPage, 2017. [Online]. Available: http://www.billboard.com/charts