Dynamo Categorization

Powered by

Dynamo Categorization is an API capable of training data models that are used for predicting document categories. It consist of several authonomous parts:

- storage
- features extraction
- machine learning algorithm

Features

- Support creating one or more trained dataset for a tenant.
- Process tenant data and extract most important features
- Trained model is cached in the file system
- Predict document categories
- Evaluate success rate of prediction
- Visualise mismatched categories during the training

Trained datasets have their unique keys that are built from the tenant id and domain. This allows to have more than one model per tenant. So a model can be trained for different entities sepately. The different model for one tenant are distinguished by their domain identifier.

> During the training the most important part is feature extraction. Dynamo invested in creating an unified set of rules that support feature extraction for any client of the Categorization but for those willing to improve their results Dynamo can tailore a more precize version of these rules. All rules are configurable from a JSON. Once all data is processed against these rules it can be used for training

Random forest classifier is used for machine learning. It builds decision trees on different samples and takes their majority vote for classification and average in case of regression. After a model is trained it's cached on the file system for fast loading. All datasets can be retrained when a new set of data is accumulated in the storage.

Tech

Dynamo categorization is using a set of technologies that simplifies working with documents and predicting their categories:

- MongoDB - MongoDb is chosen for simplified database access with json documents
- Python - Python is responsible for the machine learning phase
- node.js - evented I/O for the backend
- Express - fast node.js network app framework @tjholowaychuk
- swagger - Simplified API development
- docker - set of PaaS products that use OS-level virtualization to deliver software in packages
- github - platform for code hosting and versioning

Installation

Dynamo Categorization requires Node.js v10+ to run.

Create new project clone from github. Open terminal and start docker containers

```
cd DynamoCategorization
docker compose up
```

Open second terminal for node.js backend

```
npm i
node app
```

For development environment...

```
node app development
```

or

```
npm start
```

Project orientation

Dynamo Categorization is located in DynamoCategorization folder and consists of the following subfolders.

Folder	Description
api	all javascript sources for nodejs backend
FileBank	static files related to training
FileBank/features	feature exctraction rules
FileBank/training	prepared csv for initial training
FileBank/trained	trained models

Folder	Description
MongoImage	MongoDb running in a container
pyserver	Python machine learning related code
swagger	swagger.yaml
tests	tests and db population for testing
uploads	stores temporary files during the file upload

Docker

Dynamo Categorization relies on 3 docker containers.

- storage - Official mongodb container

By default mongo is running on port 27017. Docker exposes port 27018 so it can coexist with another mongodb instance on the same machine.

- ai - Custom created container running python server and machine learning logic

Python machine learning API is running locally on port 3010. Flask is used on port 5000. When the API is started in a container for testing purposes docker compose remaps port 3311 to communicate with AI.

- app - Nodejs container that exposes the API.

Swagger is used to simplify building and consuming the API.

Docker-compose

Docker compose is used for containers orchestration. There are two yaml files.

- docker-compose-test.yaml - this one is used for testing. It runs only storage and ai container, while node app must be started manually
- docker-compose.yaml - this one is used in production. It creates all 3 contianers. Nginx is used for proxy the calls.

Set of commands used for working with compose follows

To stop all containers

```
docker compose down
```

To start all containers

```
docker compose up
```

To rebuild specific container

```
docker-compose build ai
```

To rebuild all containers

```
docker-compose up --build
```

build-categorization.sh

A simple script that automates deployment is created for linux environment.

```
sudo service monit stop
cd ~/Categorization && sudo docker-compose down
cd ~/Categorization && git reset --hard origin/main
cd ~/Categorization && git pull origin main
cd ~/Categorization && sudo docker-compose up --build -d
sudo service monit start
```

node.js

Node app is running locally on port 3030. Once the app is started you can test api with curl commands

```
curl --location --request GET 'http://localhost:3030/predict?
unique_id=MIT@HTTP://MITIMCO_STAGING/&domain=activity&title=Varde IX -
Distribution Notice - 2020-12-08.pdf' \
--header 'Authorization: Bearer 54F39463-64EE-40AB-8B4F-6A9B0C0F1360' \
--data-raw ''
```

node.js container is preconfigured to run on port 3100. So to execute commands against node.js container all commands needs to substitute the port in the above command.

Deployment

App is currently available on prometheus server in Dynamo Environment hosted in Boston. Detailed information about commands can be found on swaggerhub. Nginx is configurred to use the official url + segment cat. So the commands has the form

```
curl --location --request GET
'https://prometheus.dynamosoftware.com/cat/get_training_status?
unique_id=MIT@HTTP://MITIMCO_STAGING/&domain=activity' \
--header 'Authorization: Bearer 54F39463-64EE-40AB-8B4F-6A9B0C0F1360' \
--data-raw ''
```

Testing

A starting dataset is included in the project. It can be activated running the following command

```
npm test
```

License