

Architecture and Design - News Article Category Prediction (Week 3 - Milestones)

Objective

This document is about classifying News Articles into categories - With information overload today users are inundated with news articles of all topics, even the ones which may not be relevant to users. Design a system which can classify incoming news articles and appropriately tag the corresponding category.

Main Software Components

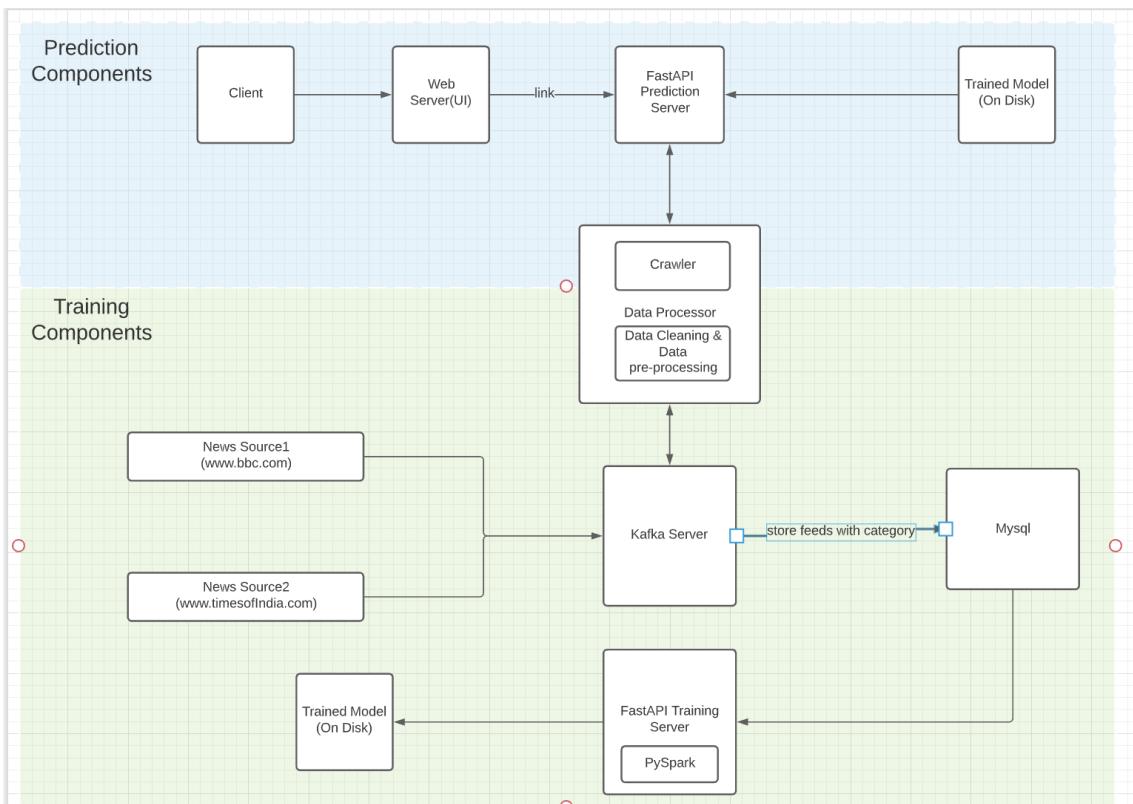
- html, css, js
- django
- python FastAPI
- Crawler(Beautiful Soup)
- pyspark
- Kafka
- hadoop
- mysql
- NLP libraries

Project Setup

Name	Date Modified	Size	Kind
> crawler	18-Oct-2021 at 1:34 PM	--	Folder
> Kafka Consumer	Today at 12:50 PM	--	Folder
> Kafka Producer	Today at 12:50 PM	--	Folder
> model	19-Oct-2021 at 12:05 PM	--	Folder
> mysql	Today at 12:50 PM	--	Folder
< newarticle	20-Oct-2021 at 10:22 AM	--	Folder
categoryPrediction	16-Oct-2021 at 1:50 PM	--	Folder
db.sqlite3	16-Oct-2021 at 11:30 AM	Zero bytes	Document
manage.py	16-Oct-2021 at 11:30 AM	667 bytes	Python Script
newarticle	16-Oct-2021 at 1:50 PM	--	Folder
static	16-Oct-2021 at 1:21 PM	--	Folder
templates	16-Oct-2021 at 1:13 PM	--	Folder
Predictor	19-Oct-2021 at 12:25 PM	--	Folder
Trainer	19-Oct-2021 at 11:53 AM	--	Folder

- All the packages and services are accumulated in the docker-compose(yaml) file.

Architecture



UI Server(django)

Technologies

- We are using django server as a frontend web server.
- HTML, CSS, bootstrap for rendering webpages.
- Javascript for input validations.

Functionality

- User can enter url or article in the input text box.
- Then he can click on Predict button for category predictions.
- It will get the article from the crawler and display it on the page.
- It will also call the predictor api which returns the predicted category.
- Retrain button is also provided to retrain the model, it will call the Trainer API for the same.

Most of the world's nations have signed up to a historic deal to ensure big companies pay a fairer share of tax. A hundred and thirty six countries agreed to enforce a corporate tax rate of at least 15% and a fairer system of taxing profits where they are earned. It follows concern that multinational companies are re-routing their profits through low tax jurisdictions. Countries including Ireland had opposed the deal but have now agreed to the policy. UK Chancellor Rishi Sunak said the deal would "upgrade the global tax system for the modern age". "We now have a clear path to a fairer tax system, where large global players pay their fair share wherever they do business," he said. The Organisation for Economic Cooperation and Development (OECD), an intergovernmental organisation, has led talks on a minimum rate for a decade. It said the deal could bring in an extra \$150bn (£108bn) of tax a year, bolstering economies as they recover from Covid. Yet it also said it did not seek to "eliminate" tax competition between countries, only to limit it. The floor under corporate tax will come in from 2023. Countries will also have more scope to tax multinational companies operating within their borders, even if they don't have a physical presence there. The move - which is expected to hit digital giants like Amazon and Facebook - will affect firms with global sales above 20 billion euros (£17bn) and profit margins above 10%. A quarter of any profits they make above the 10% threshold will be reallocated to the countries where they were earned and taxed there. "[This] is a far-reaching agreement which ensures our international tax system is fit for purpose in a digitalised and globalised world economy," said OECD Secretary-General Mathias Cormann. "We must now work swiftly and diligently to ensure the effective implementation of this major reform." This deal marks a sweeping change in approach when it comes to taxing big global companies. In the past, countries would frequently compete with one another to offer an attractive deal to multinationals. It made sense when those companies might come in, set up a factory and create jobs. They were, you could say, giving something back. But the new digital era giants have become adept at simply moving profits around, from the regions where they do business to those where they will pay the lowest taxes. Good news for tax havens, bad news for everyone else. The new system is meant to minimise opportunities for profit shifting, and ensure that the largest businesses pay at least some of their taxes where they do business, rather than where they choose to have their headquarters. Some 136 countries have signed up - an achievement in itself. But inevitably there will be,

Categories Supported

```
{'business':1,'computers':2,'covid':3,'entertainment':4,'health':5,'lifestyle':6
,'politics':7,'science':8,'sport':9,'technology':10,'trade':11,'travel':12}
```

Segregate the preprocessed data into training and test

```
X_train, X_test, y_train, y_test = train_test_split(
    news_df['article'],
    news_df['category'],
    random_state = 1
)
```

Complete model training and deployment (and model registry)

- We are using MLflow for model training and deployment.
- It will serialize the model and store it as the artifacts.

```
from sklearn.pipeline import Pipeline
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.naive_bayes import MultinomialNB
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.base import TransformerMixin, BaseEstimator, clone
import mlflow
import mlflow.sklearn

with mlflow.start_run(run_name='complaint_classifier_sklearn'):
```

```

# get mlflow run Id
run_id = mlflow.active_run().info.run_id

# Text preprocessing, tokenizing and filtering of stopwords are all included in
CountVectorizer
count_vect = CountVectorizer(ngram_range=(2,2), min_df=3)
tf_transformer = TfidfTransformer(use_idf=True)
clf = MultinomialNB()

# define pipeline
pipeline = Pipeline([
    ('vect', count_vect),
    ('tfidf', tf_transformer),
    ('clf', clf)
])

# Train pipeline
pipeline.fit(X_train, y_train)
y_pred = pipeline.predict(X_test)
accuracy = accuracy_score(y_pred, y_test)

# Log pipeline to mlflow
mlflow.sklearn.log_model(pipeline, "pipeline")
mlflow.log_metric("accuracy", accuracy)

# Log classification report
cls_report = pd.DataFrame(classification_report(y_test, y_pred, target_names=
[str(i) for i in range(1,13)], output_dict=True)).transpose()
cls_report.to_csv('classification_report.csv', index=True)
mlflow.log_artifact('classification_report.csv')

cls_report

```

Serialize the re-trained model.

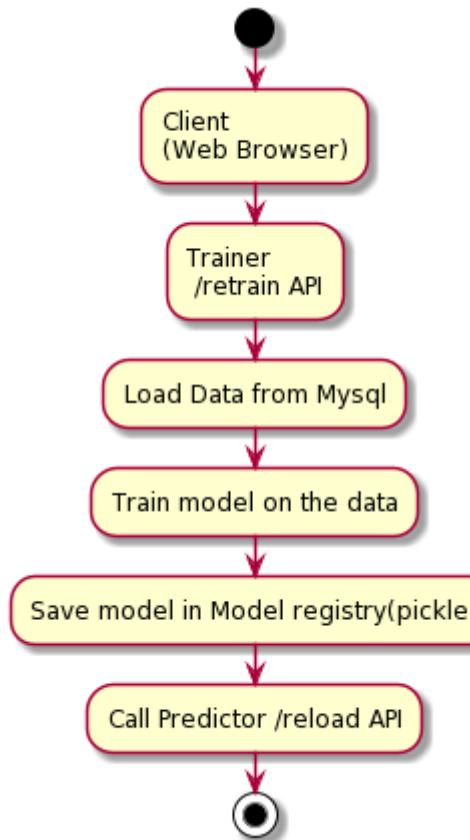
- Trained models are logged as MLflow model artifacts.

Push this model to model-registry (MLFlow) along with hyperparameters

Name	Date Modified	Size	Kind
0	Today at 1:05 PM	--	Folder
> Ocaaadaeeea54d0f9d4ce9e07c5d605f	Today at 10:20 AM	--	Folder
> 850634f2a76f4317aba9525e789060c0	Today at 10:18 AM	--	Folder
> b96b7ce6c0bf411d86d7313c1ea7b80b	Today at 1:05 PM	--	Folder
< artifacts	Today at 1:05 PM	--	Folder
< model	Today at 10:14 AM	--	Folder
conda.yaml	Today at 10:14 AM	215 bytes	YAML
MLmodel	Today at 10:14 AM	349 bytes	Document
model.pkl	Today at 10:14 AM	3.8 MB	Document
requirements.txt	Today at 10:14 AM	107 bytes	Plain Text
training_confusion_matrix.png	Today at 10:14 AM	31 KB	PNG image
meta.yaml	Today at 10:14 AM	403 bytes	YAML
metrics	Today at 10:14 AM	--	Folder
params	Today at 10:14 AM	--	Folder
tags	Today at 10:14 AM	--	Folder
> cb797b45cd1a44189fbc6ecb768264e5	Today at 10:22 AM	--	Folder
meta.yaml	Today at 10:14 AM	128 bytes	YAML

Convert the project to an On-Demand service, to be able to retrain the model as needed. This could be a flask API that could trigger the entire pipeline in model-training-service (loading data into PySpark, preprocessing, model training)

- Retrain API is created on the Trainer service.
- The service will get the articles from the mysql db and train the model.
- This model is then logged into the model registry by MLflow.
- Trainer will then inform the predictor to reload the model.



{.callout}

Crawler

- There are mainly three components of the Crawler:

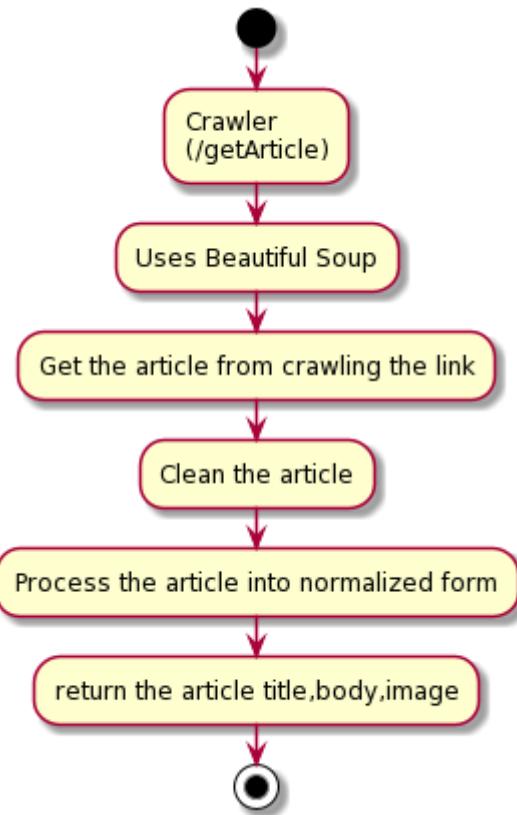
1. Crawler Scheduler

- Scheduler runs the cron job which will run once in a day.
- It will get the latest links from the google search web page.
- And then it pushes it to mysql db.



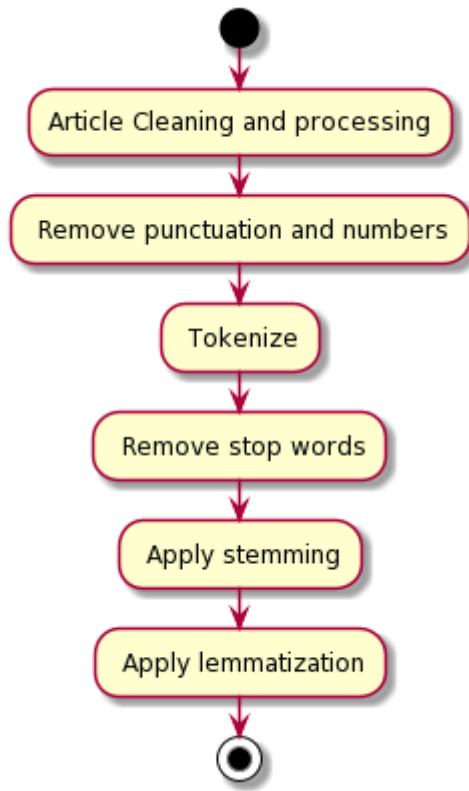
2. Crawler getArticle API

- This API will crawl the webpage, given the article link. and returns the article along with the image.



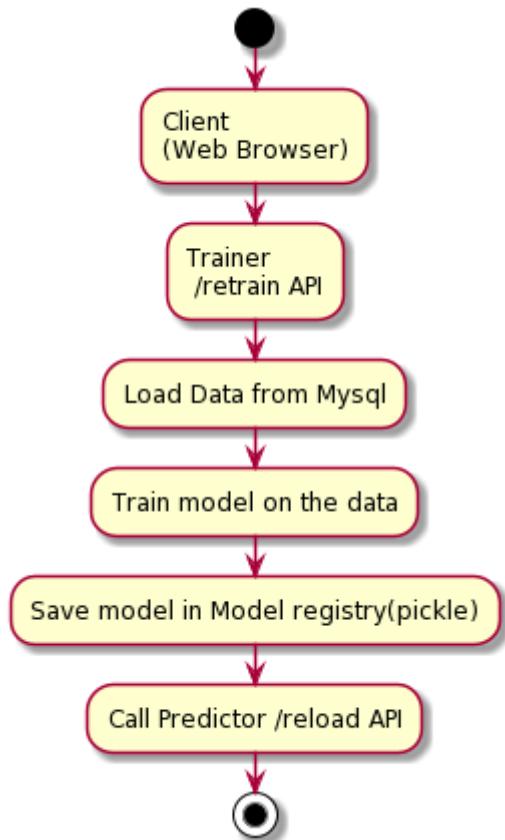
3. Preprocessing Article

- It involves text normalization, stop words removal, stemming, lemmatization etc.



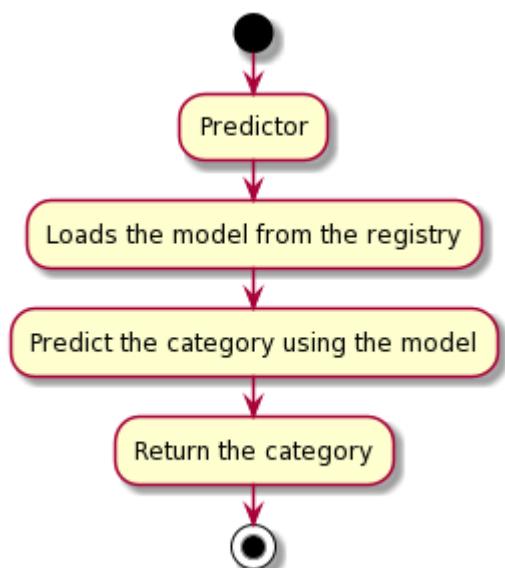
Trainer

- Trainer first loads the articles from mysql db.
- Then train the model using MLflow and pyspark.
- MLflow saves the model in the modele-registry as the artifacts.



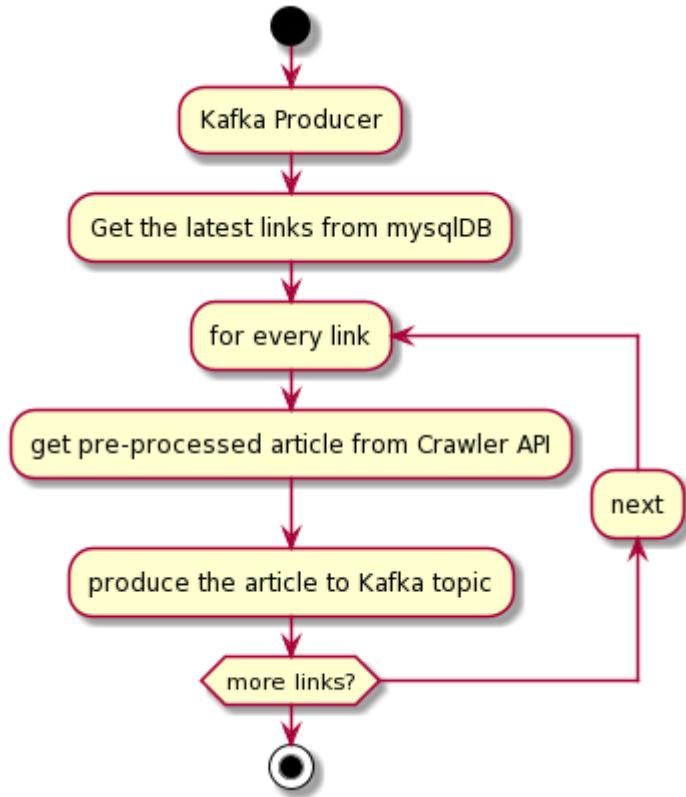
Predictor

- Predictor loads the model from the registry.
- Predicts the category, given the article.



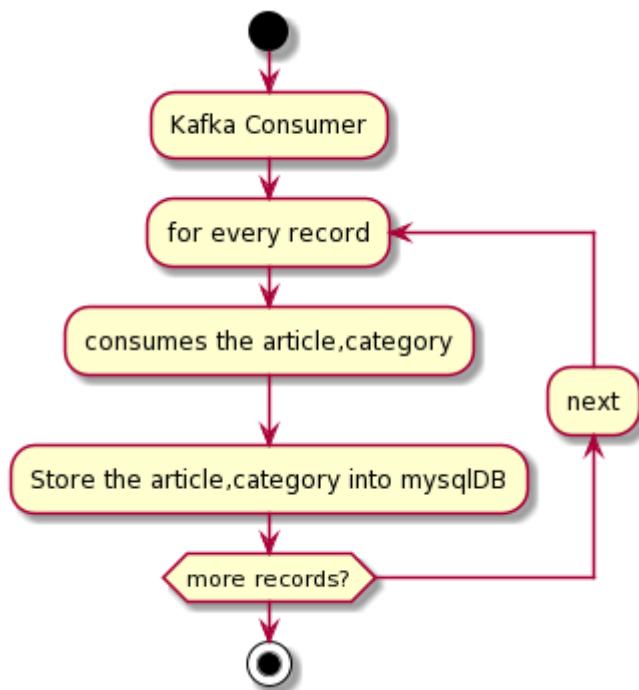
Kafka Producer

- Gets the latest links from the mysql db.
- For every link, gets the preprocessed article from the crawler and produce it to Kafka topic.



Kafka Consumer

- Kafka Consumer consumes the article and write the article into mysql.



Group Details

- Group-Name: VK Learners
- ◦ Member1: Vinod Adwani
- ◦ Member2: KSRC Murthy