FAKE NEWS DETECTION USING NLP

SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

THE AWARD OF

BACHELOR OF ENGINEERING DEGREE IN COMPUTER SCIENCE AND ENGINEERING

BY

Ezhilkaviya.k

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

STAR LION COLLEGE OF ENGINEERING AND TECHNOLOGY

MANANKORAI,THANJAVUR

-614206

OCTOBER-2023

IBM

IBM Watson AIOps

Abstract

Machine Learning (ML) becomes increasingly popular; industry spends billions of dollars building ML systems. Data scientists have come up with many good algorithms and trained models. However, putting those ML models into production is still in the early stage. The deployment process is distinct from that for traditional software applications; it is not yet well understood among data scientists and IT engineers in their roles and responsibilities, resulting in many anti-pattern practices. The key issues identified by researchers at Google include lack of production-like prototyping stack for data scientists, monolithic programs not fitted for component based ML system orchestration, and lack of best practices in system design. To find solutions, teams need to understand the inherent structure of ML systems and to find ML engineering best practices. This paper presents an abstraction of ML system design process, a design pattern named Model-Service-Client + Retraining (MSC/R) consisting of four main components: Model (data and trained model), Service (model serving infrastructure), Client (user interface), and Retraining (model monitoring and retraining). Data scientists and engineers can use this pattern as a discipline in designing and deploying ML pipelines methodically. They can separate concerns, modularize ML systems, and work in parallel.

List of Abbreviations

AWS: Amazon Web Service

EC2: Elastic Cloud Computing Service

ECS: Elastic Container Service

ECR: Elastic Container Registry

ELB: Elastic Load Balancing

S3: Simple Storage Service

IAM role: Identity and Access Management

GCP: Google Cloud Platform GCE: Google Computing Engine

GKE: Google Kubernetes Engine

REST: REpresentational State Transfer

API: Application Programming Interface

IoT: Internet of Things

MVC: Model-View-Controller

ML: Machine Learning

AI: Artificial Intelligence

DevOps: Development and Operations

YOLO: You Only Look Once. A new object detection approach

COCO dataset: Common Objects in Context

OpenCV: Open Computer Vision

LSTM: Long Short-Term Memory

RNN: Recurrent Neural Network

PaaS: Platform as a Service

ROI: Return on Investment

Introduction

According to the recently updated International Data Corporation (IDC) Worldwide Artificial Intelligence Systems Spending Guide, spending on AI systems will reach $97.9 billion in 2023, more than two and one half times the $37.5 billion that was spent in 2019. The compound annual growth rate (CAGR) for the 2018-2023 forecast period will be 28.4%. However, reports show that a majority (up to 87%) of corporate AI initiatives are struggling to move beyond test stages. Early evidence suggests that the technology can deliver real value to serious adopters. Those organizations that actually put AI and machine learning into production saw a 3-15% profit margin increase.

Commercial clouds provide Platform as a Service (PaaS) that offer end-to-end services to develop, design and run business applications in an agile and scalable environment. This ability is good to host ML models to a large number of users and large volumes of data sets. However, to effectively navigate and configure the complex cloud services and find the optimum deployment architecture remains a challenge. No matter how well the model is, data scientists continue to have issues deploying to production often resulting in crippled projects.

One of the major obstacles preventing businesses from gaining returns on their investment (ROI) is that ML infrastructure and operations are different from those designed for traditional software applications; they are much more complex and dynamic. As explained in, both traditional and ML applications perform actions in response to inputs. But the way actions are codified differs greatly. Traditional software codifies actions as explicit rules. ML does not codify explicitly. Instead rules are indirectly set by capturing patterns from data. As a result, IT teams are ill prepared to deploy and operationalize the trained models as usable business applications, and data scientists are diverting their talent to sorting out infrastructure and operational issues.

The fact is ML applications introduce a new culture; their deployment and operations require new discipline and processes different from the existing DevOps practices. The large investment with 87% failure to bring ML applications to production shows companies tried to solve the same design problems over and over again at great time and expense.

The failed ROI in the industry opened opportunities for finding best practices in the field. The concept of MLOps, short for Machine Learning and Operations, was introduced in recent years as a new discipline and process of collaboration among data scientists, system engineers and business analysts to work together and build ML business applications effectively. The main success factors are outlined as follows

1) Reduced time and difficulty to deploy ML models to production.

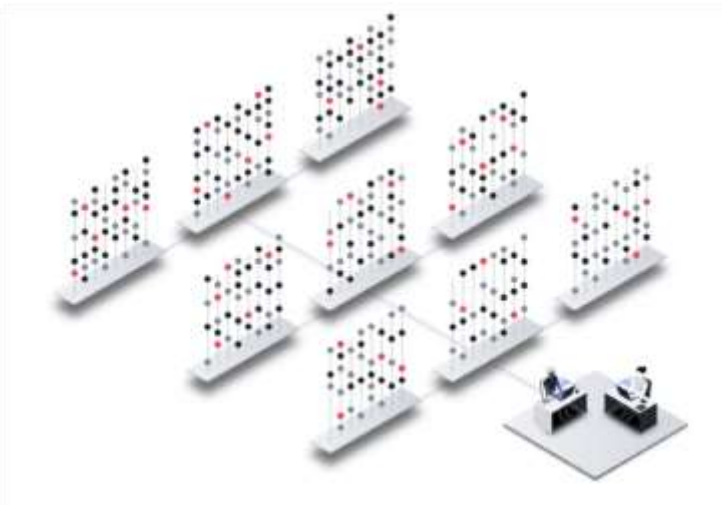
2)Capability to scale up/down horizontally and automatically.

3)Live model monitoring, tracking and retraining

An AI-powered IT incident resolution application, fueled by your own data

The solution enables you to:

- **Remove guesswork and preemptively resolve your next IT outage**
- **Merge structured and unstructured data, on any cloud**
- **Manage complex multi-cloud and multi-vendor environments to more easily align to your IT practices**
- **Surface insights within your existing ChatOps environment**
- **Develop insights quickly with pre-trained models, accelerating time-to-value**
- **Explain the decisions and recommendations provided**



Single (data) source of truth

IT infrastructure is the central nervous system of an enterprise.
Managing large annual investments in corporate technology, CIOs are under more pressure than ever to deliver innovation and emerging technologies to their enterprise while keeping costs down.

However, in today's technology-driven landscape even the smallest of IT outages can cause massive economic impact. IT leaders indicate that the costliest aspects of downtime are lost revenue (53%), lost productivity (47%), and negative impact on corporate reputation (41%).

IT teams sift through data from topology, logs, tickets, alerts and more, to better predict and reactively solve IT outages through static communication. Through these disparate data sources, teams still lack a singular view to aggregate these insights together into one application. Unfortunately, the typical IT Operations manager must sift through several data sources to monitor the status and outage likeliness of their IT stack. Even if an issue is detected, workers use multiple platforms to notify team members and diagnose issues through a network's standard runbook. Teams often work with tools that are time-stamped and don't summarize an outage into one cohesive conversation within their existing ChatOps environment.

IBM® Watson AIOps connects to any collaboration platform and delivers insights where professionals already work. High-growth firms consider AI key to business competitiveness today: 38% say it is the most important factor vs. just 20% of lower-growth firms. Almost 60% of firms say AI and machine learning are one of their highest, or the highest, investment area

Unleash your IT Team

IT professionals around the world are overwhelmed with data, but often have difficulty focusing on the insights that truly matter. Site Reliability Engineers (SREs) can spend the majority of their time sifting through multiple data sources, but they need to shift to mission-critical workloads.

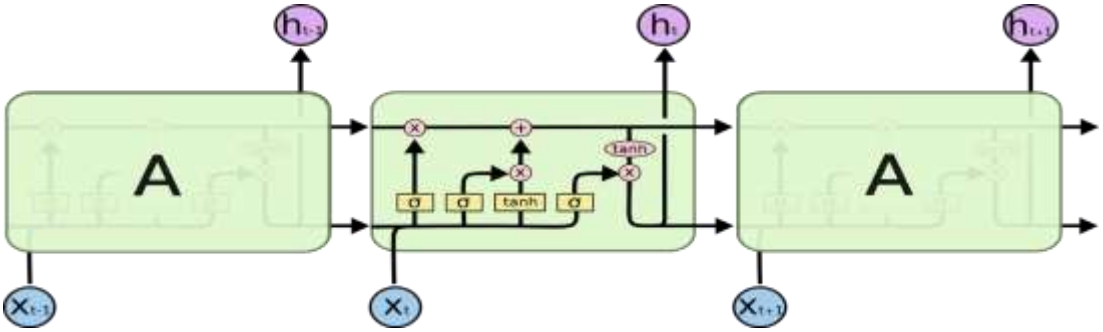
Teams must be equipped with acute agility to manage their capacities and operate across multiple departments in a timely manner. Nonetheless, the power of artificial intelligence has truly unlocked the ways in which it can be applied to the IT domain.

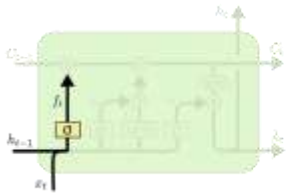
CIO offices require quick results from their investment in technology and seamless integration with a variety of technology. Watson AIOps provides out-of-the-box root cause ontology that targets frequently occurring problems in the IT operations domain with cloud-native applications. The technology integrates with your existing IT Ops tools, allows for model customization and runs on any cloud. With this product, enterprises have the ability to assess, diagnose, and react to anomalies surfaced in real time across disparate systems to improve stability, reliability and availability for their company.

By leveraging the advantages of Watson AI and a modern information architecture, we give the CIO organization new superpowers. Offering holistic, real-time insights and an interactive and consultative engagement experience, we provide peace of mind and instil ability into business workflows. With IBM AIOps, teams get insights, not just more data, enabling operations to keep pace with faster dev cycles, reducing skill requirements and the ability to exceed financial targets Built on IBM Cloud Pak® for Data, Watson AIOps aims to equip enterprises to execute an AI-first strategy to operationalize their data to transform and modernize business workflows. CIOs of the future will be able to use IBM's data fabric and AI-powered insights platform across their IT landscapes and dev toolchains to transform internal experiences, reduce risk, exceed client expectations, deliver external value, and accelerate business outcomes. This technology of tomorrow is available today.

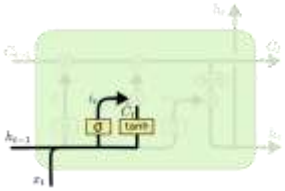
Stock Predictor Using Long Short-Term Memory (LSTM)

Another machine learning model used is a stock prediction model using Long Short-Term Memory (LSTM) designed by Dr. Sung Kim. LSTM, first published in 1997, is a Recurrent Neural Network (RNN) algorithm designed to avoid the long-term dependency problem in standard RNN. A LSTM has the form of a chain of repeating modules of neural networks which is very powerful in sequence prediction problems. The details of LSTM are in Figure1 and Figure2. In the Figures, Xt is the input, ht is the output, Ct is the cell state, ft is the forget value.



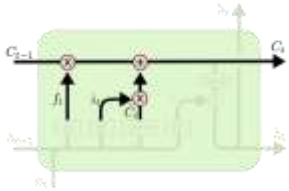


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

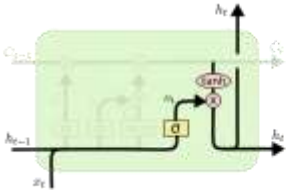


$$i_t = \sigma\left(W_i \cdot [h_{t-1}, x_t] \; + \; b_i\right)$$
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

(a) removing information **(b) create an update to the state**



$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$



$$o_t = \sigma\left(W_o \; [h_{t-1}, x_t] \; + \; b_o\right)$$
$$h_t = o_t * \tanh\left(C_t\right)$$

(c) update the cell state **(d) generate the output**

In the case study, historical stock price of Alphabet Inc. (GOOG) is used to train and generate the stock prediction model. The model can be used to predict the next day's Close Price based on the data from the previous seven days.

Brief Introduction of Cloud Technologies

Amazon Web Services (AWS)

Amazon Web Service (AWS) is a cloud computing service provided by Amazon. It offers reliable, scalable, and inexpensive cloud computing services, such as Elastic Cloud Computing Service (EC2), Simple Storage Service (S3), etc.

With its rich ecosystem, AWS allows users to develop, test, deploy, maintain their application without hardware worries.

Google Cloud Platform (GCP)

Google Cloud Platform (GCP) is a cloud computing service provided by Google. It is a suite of cloud computing services that runs on the same infrastructure Google uses internally for its enduser products, such as Google Search, Gmail and YouTube. The services on GCP such as Google Computing Engine (GCE), Google Kubernetes Engine (GKE), etc have less configurations compared to AWS, making it easy for users to develop and deploy their applications.
Other Technologies Used

Docker
Docker is an open platform for developing, shipping, and running applications. It provides the ability to package and run an application in a loosely isolated environment called a container. With container technology, applications can be portable, scalable, and lightweight running on different environments.

Related Work

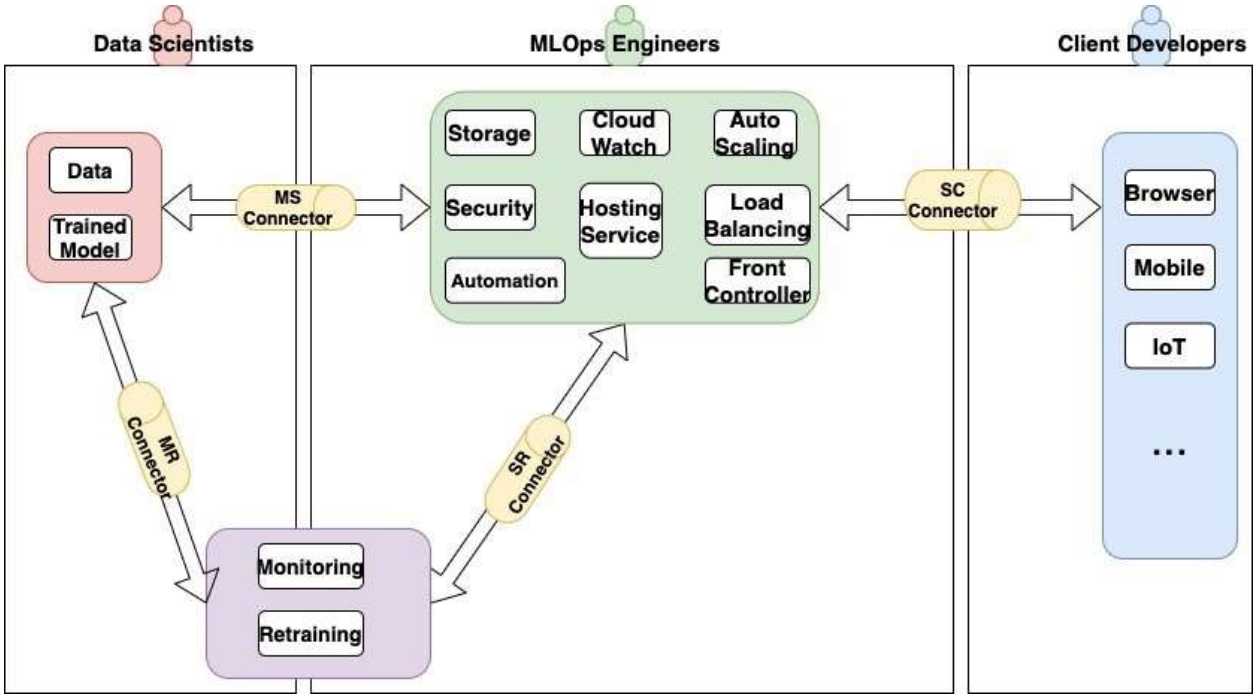
Deploying Machine Learning service into production is still new. Industry leaders such as Google, IBM, AWS, Microsoft spend a lot of resources to do research and try to capture the best practices that can be implemented widely. O'Leary and M. Uchida worked with over 100 participants in industry on their ML pipelines and identified three common problems as: 1) The environment for prototyping ML models should be designed to prevent the need to re-implement from scratch for production, 2) ML pipelines should provide a framework of pre-defined canonical unit of operations as components such that ML code can follow ML engineering best practices, as opposed to free-form flexibility, 3) Interfaces between components—both code and data—should be made explicit and simple enough so that implementing such interfaces is easy to use for ML code authors. It concludes that software developers are concerned by the complexity of ML systems and their lack of knowledge of the architecture and design (anti-)patterns that could help them.

The authors identified a list of patterns, such as "ML Versioning'', "Daisy Architecture" loosely applied in the process of developing ML pipeline, but there remains work to identify design patterns for ML applications.

MSC/R Implementation Case Study

The Models

- **YOLOv3 for image detection using COCO dataset [8,10]. The COCO dataset contains 330K images with 80 categories of objects.**
- **Stock Prediction trained with the historical stock price of Alphabet Inc. (GOOG) from Aug 18,2004 to May 22,2020 to predict 20-day stock price. The data contains Open, High, Low, Volume and Close data.**



Deployment - ML model is ready to be deployed and tested.

MLOps is the major player in this stage. Its work includes packaging the code, deploying ML models, configuring and tuning the environment, testing, and working with other teams to reengineer as needed. Traditionally, this is the point at which many companies struggle.

Following MSC/R in the Development stage, the 3 main issues identified by can be greatly reduced or eliminated, thus, to increase success rate in this stage significantly.

Production - While the ML model is in production serving clients, MLOps is responsible for operational tasks such as monitoring, performance tuning, continuous integration and testing etc. This is not the focus of this project.

YOLO – api.py

```
import time from absl import flags,
logging from absl.flags import FLAGS
import cv2 import flask import numpy as
np import tensorflow as tf import boto3
from flask_restful import reqparse, Api, Resource
from yolov3_tf2.models import (
   YoloV3, YoloV3Tiny
)
from yolov3_tf2.dataset import transform_images, load_tfrecord_dataset from
yolov3_tf2.utils import draw_outputs
from data.py import getObject,uploadObject flags.DEFINE_string('classes',
'./data/coco.names', 'path to classes file') flags.DEFINE_string('weights',
'./checkpoints/yolov3.tf',
                'path to weights file')
flags.DEFINE_boolean('tiny', False, 'yolov3 or yolov3-tiny')
 flags.DEFINE_integer('size', 416, 'resize images to') flags.DEFINE_string('image',
'./data/girl.png', 'path to input image') flags.DEFINE_string('tfrecord', None,
'tfrecord instead of image') flags.DEFINE_string('output', './static/output.jpg',
'path to output image') flags.DEFINE_integer('num_classes', 80, 'number of
classes in the model') def detect(fileName):     physical_devices =
tf.config.experimental.list_physical_devices('GPU')     if len(physical_devices) > 0:
tf.config.experimental.set_memory_growth(physical_devices[0], True)

   #if FLAGS.tiny:
   #    yolo = YoloV3Tiny(classes=FLAGS.num_classes)
   #else:
   yolo = YoloV3(classes=80)

   yolo.load_weights('./checkpoints/yolov3.tf').expect_partial()
logging.info('weights loaded')

   class_names = [c.strip() for c in open('./data/coco.names').readlines()]
logging.info('classes loaded')

   #if FLAGS.tfrecord:
   #    dataset = load_tfrecord_dataset(
   #        FLAGS.tfrecord, FLAGS.classes, FLAGS.size)
   #    dataset = dataset.shuffle(512)     #    img_raw, _label
= next(iter(dataset.take(1)))     #else:
   img_raw = tf.image.decode_image(
      open(fileName, 'rb').read(), channels=3)
img = tf.expand_dims(img_raw, 0)
   img = transform_images(img, 416)
```

```
   t1 = time.time()
   boxes, scores, classes, nums = yolo(img)     t2 =
time.time()
   logging.info('time: {}'.format(t2 - t1))

   logging.info('detections:')     for i in range(nums[0]):
logging.info('\t{}, {}, {}'.format(class_names[int(classes[0][i])],
                                                np.array(scores[0][i]),
                                 np.array(boxes[0][i])))

   img = cv2.cvtColor(img_raw.numpy(), cv2.COLOR_RGB2BGR)     img =
draw_outputs(img, (boxes, scores, classes, nums), class_names)
   cv2.imwrite('./static/output.jpg', img)
   logging.info('output saved to: {}'.format('./static/output.jpg'))

app = flask.Flask(__name__) api = Api(app)

# argument parsing parser =
reqparse.RequestParser()
parser.add_argument('query', type=str)

class YOLODetection(Resource):     def
post(self):
      args = parser.parse_args()       user_query =
args['query']       file_name = user_query
      bucket = "elasticbeanstalk-us-west-1-019024743397"       object_name =
file_name
      getObject(bucket, file_name, object_name)
      detect(file_name)       image_name =
'./static/output.jpg'       upload_name =
'yolo/detection.png'
      url = uploadObject(bucket,image_name,upload_name)
output = {'Detection result': url}       return output


api.add_resource(YOLODetection,'/')

if __name__ == '__main__':
   app.run(host='0.0.0.0', port= 5000)
```

Solving common problems

Problem-1

Lack of prototype stack mirroring production environment

With separation of concerns, data scientists can rely on MLOps to build the prototype stack that mirrors the production environment. MLOps is the domain experts in IT technology, system design and cloud technology. With proper requirements, standing up a ML development environment in public clouds in today's world is relatively simple and cost effective. Data scientists no longer need to struggle with using personal laptops, trying to piece together different tools and utilities to end up with "pipeline jungle."

Problem-2

Programming style conflict. Data scientists tend to develop models with a monolithic program, not following software engineering best practices.

The connector components in MSC/R provide guidelines for teams to follow best practice to create well defined interface, to design by contract, and to heed modularization. For instance, it enforces data scientists to compose code in the form of functions or microservices to be compatible with the production ecosystem.

Problem-3

System design anti-patterns. Glue code and pipeline jungles, causing integration issues. Following the discipline of adopting system design best practices, such as separation of concerns, design by contract to produce more modularized code. For example, in order to leverage container technology, ML code needs to be cleaned to get rid of glue code, and pipeline jungles. With close collaboration with MLOps, data scientists and teams can test more often, get rid of experimental code and dead code paths, and deal with technical debt and anti-pattern practice quickly to reduce integration issues.

Deployment Steps

1. **Create the ML service in Cloud Run, in this implementation, specify the capacity of CPU allocated to 2,**
2. **Memory allocated to 2 GB.**
3. **Other infrastructures such as Load Balancing, HTTP endpoints, auto scaling are full managed by Google**
4. **Wait for Google Cloud Run launches the service**

Test

1. **The accessible URL of the ML service can be gotten from Cloud Run → Service details:**
2. **Stock Prediction: https://stock-predict-jay-rakei3ugwq-uc.a.run.app**
3. **YOLO: https://yolo-detection-jay-rakei3ugwq-ue.a.run.app**
4. **The test details for both Stock Prediction and YOLO are similar to the description in Section**

Thoughts on using Serverless platform, Fully managed platform SageMaker

For ML systems running fully managed and serverless ML platforms, such as AWS SageMaker, and Google CloudRun, there are still separation of concerns and the interfaces among teams as described in the Connector pattern. Those platforms provide tools for better pipeline automation in production, and shorter development lifecycle, but the work and responsibility remain and have to be shared divided among teams.

For example, once a prototype pipeline is configured, AWS SageMaker enables developers and data scientists to build and train models using Notebook, and immediately deploy the model for integration and testing. Therefore, it enhances the CI-CD-CT process (continuous integration, continuous delivery, continuous testing). It comes with cost, for example, the price comparison between SageMaker and EC2. The pick-and-choose approach used for this project provides a good opportunity to understand how different tools work together. It also provides flexibility for MLOps to customize and tune individual components, such as choosing different CPU/GPU, data storage on the pipeline and make the pipeline more cost effective.

Both AWS and GCP offer serverless options for deploying machine learning models

Design SC Connector

MLOps works with Client developers to determine the protocol between Front Controller and client endpoint. In this case, the ML model is exposed as a RESTful API, the client needs to invoke the service by sending HTTP requests and get responses in json file format. For testing purposes, I acted as a client developer and wrote a HTML web page where a user can submit a request to the REST API and the page then displays the result on the browser. In business projects, a client may be a streaming device, MLOps would need to use different protocols such as Real-Time Messaging Protocol (RTMP) in Service-Client connector.

Design MR Connector

This interface between Model and Retraining requires collaboration between data scientists and MLOps to define the rules of how retrained models are to be tested, versioned and released. For this project with limited resources, I chose to use AWS S3 to store a retrained model. The auto deploying script automatically picks up the latest model in the directory. In business, when a new version is produced and deposited to GitHub repository, it can trigger CI/CD process before the model is deployed to the cloud

Design SR Connector

This interface between Service and Retraining requires collaboration between data scientists to decide how to initiate retraining, and what parameters to dynamically set at runtime. For example, you can change the training data

path and get data from different sources. For this case study, I used a static data path pointing to an AWS S3 bucket.

I used a system scheduler as a trigger.

FAKE NEWS DETECTION IN SOCIAL MEDIA

Fake news and hoaxes have been there since before the advent of the Internet. The widely accepted definition of Internet fake news is: fictitious articles deliberately fabricated to deceive readers". Social media and news outlets publish fake news to increase readership or as part of psychological warfare. In general, the goal is profiting through click baits. Click baits lure users and entice curiosity with flashy headlines or designs to click links to increase advertisements revenues. This exposition analyse the prevalence of fake news in light of the advances in communication made possible by the emergence of social networking sites. The purpose of the work is to come up with a solutionthat can be utilized by users to detect and filter out sites containing false and misleading information. We use simple and carefully selected features of the title and post to accurately identify fake posts. The experimental results show a 99.4% accuracy using logistic classifier

Automatic Online Fake News Detection Combining Content and Social Signals

The proliferation and rapid diffusion of fake news on the Internet highlight the need of automatic hoax detection systems. In the context of social networks, machine learning (ML) methods can be used for this purpose. Fake news detection strategies are traditionally either based on content analysis (i.e.)

analyse the content of the news) or - more recently - on social context models, such as mapping the news‟ diffusion pattern. In this paper, we first propose a novel ML fake news detection method which, by combining news content and social context features, outperforms existing methods in the literature, increasing their already high accuracy by up to 4.8%. Second, we implement our method within a Facebook Messenger chat bot and validate it with a real-world application, obtaining a fake news detection accuracy of 81.7%.

THE SPREAD OF FAKE NEWS BY SOCIAL BOTS

The massive spread of fake news has been identified as a major global risk and has been alleged to influence elections and threaten democracies. Communication, cognitive, social, and computer scientists are engaged in efforts to study the complex causes for the viral diffusion of digital misinformation and to develop solutions, while search and social media platforms are beginning to deploy countermeasures. However, to date, these efforts have been mainly informed by anecdotal evidence rather than systematic data. Here we analyse 14 million messages spreading 400 thousand claims on Twitter during and following the 2016 U.S. presidential campaign and election. We find evidence that social bots play a key role in the spread of fake news.

Accounts that actively spread misinformation are significantly more likely to be bots. Automated accounts are particularly active in the early spreading phases of viral claims, and tend to target influential users. Humans are vulnerable to this manipulation, retweeting bots who post false news. Successful sources of false and biased claims are heavily supported by social bots. These results suggests that curbing social bots may be an effective strategy for mitigating the spread of online misinformation. thousand claims on Twitter during and following the 2016 U.S. presidential campaign and election. We find evidence that social bots play a key role in the spread of fake news. Accounts that actively spread misinformation are significantly more likely to be bots. Automated accounts are particularly active in the early spreading phases of viral claims, and tend to target influential users. Humans are vulnerable to this manipulation, retweeting bots who post false news. Successful sources of false and biased claims are heavily supported by social bots. These results suggests that curbing social bots may be an effective strategy for mitigating the spread of online misinformation

MISLEADING ONLINE CONTENT

Tabloid journalism is often criticized for its propensity for exaggeration, sensationalise, scare-mongering, and otherwise producing misleading and low quality news. As the news has moved online, a new form of tabloidization has emerged: „click baiting.“ „Clickbait“ refers to “content whose main purpose is to attract attention and encourage visitors to click on a link to a particular web page” [„clickbait,“] and has been implicated in the rapid spread of misinformation online. This paper examines potential methods for the automatic detection of clickbait as a form of deception. Methods for recognizing both textual and non-textual click baiting cues are surveyed, leading to the suggestion that a hybrid approach may yield best results. Big Data Analytics and Deep Learning are two high-focus of data science. Big Data has become important as many organizations both public and private have been collecting massive amounts of domain-specific information, which can contain useful information about problems such as national intelligence, cyber security, fraud detection, marketing, and medical informatics. Companies such as Google and Microsoft are analyse large volumes of data for business analysis and decisions, impacting existing and future technology. Deep Learning algorithms extract high- level, complex abstractions as data representations through a hierarchical learning 7 process. Complex abstractions are learnt at a given level based on relatively simpler abstractions formulated in the preceding level in the hierarchy. A key benefit of Deep Learning is the analysis and learning of massive amounts of unsupervised data, making it a valuable tool for Big Data Analytics where raw data is largely un label and un-categorized

Conclusions and Future Work

Conclusions

Case study shows that by following MSC/R pattern, MLOps can quickly create a pipeline for data scientists to use for their prototyping and testing and replicate the pipeline as a template for designing large scale environments that mirror production quickly and effectively. The tasks can be better organized, and the collaboration is much easier with well-defined contracts and interfaces among teams. In the case study, a prototype ML stack can be configured and deployed in a matter of a few days in the public cloud. Load testing also shows the auto scaling and load balance technology in those clouds are very mature and reliable. The three success factors are very achievable if the process is guided by discipline and best practices. Leveraging infrastructure as code practice, the prototyping stack can be easily converted into a template; by upgrading the specification of certain components such as CPUs and memories, cluster size, it can be turned into production grade without huge effort; the template can be used to stand up more environment for testing staging with same architecture, significantly reduce the portability issue.

Using design patterns in software development helps address design complexity and quality. It can also adapt to ML service development. In this paper, an abstract Model-Service-Client + Retraining (MSC/R) is summarized from existing ML pipeline practices and simply implemented using 3 popular approaches. It is quite obvious that with the MSC/R pattern, the ML pipeline can be separated of concerns and reusable. This may provide a good suggestion in industry practising.

Future Work

End to End Pipeline Case Study

Collaborate with data scientists to conduct case study to create pipeline that start from data collection to model serving

More Patterns as Discipline and Best Practice

To further develop the MSC/R design pattern, more ML application patterns, classifying patterns and identifying anti-patterns will be collected. More detailed insight of each component in MSC/R and the connectors between them will be developed, such as Data infrastructure that produce live training data; Model Serving Infrastructure to differentiate Client requests, such as batch or real-time; Security infrastructure that protect the sensitive data and services.