Applied MLOps to Maintain Model Freshness on Kubernetes

2021 Berlin Buzzwords

About Us

Jeff Zemerick 

@jzonthemtn

Current Chair of Apache OpenNLP

Member of the Apache Software Foundation

Previously an AWS, GCP Engineer

jzemerick@opensourceconnections.com

David Smithbauer 

@dsmithbauer

Lead DevOps Engineer

linkedin.com/in/davidsmithbauer

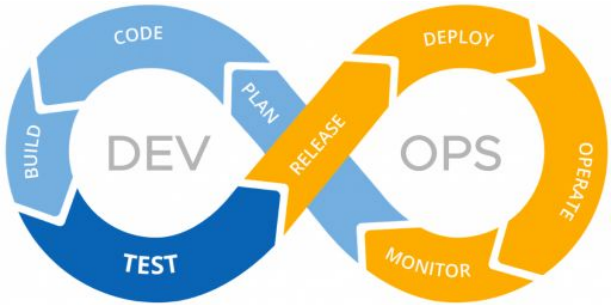
david.smithbauer@leidos.com

Introduction

- Illustrate how a containerized system with machine learning capabilities can evolve over time to increase performance and relevance
- Intended to describe concepts instead of a specific tooling “how-to” guide
- Describe MLOps, what it solves, and its challenges

What is MLOps?

- Intersections of DevOps and Machine Learning
- Getting a ML model into production



- Increased automation
- ML model lifecycle
  - SDLC, CI/CD
  - Deployment
  - Monitoring/metrics
- https://madewithml.com/courses/mlops/

https://blogs.vmware.com/management/2020/05/devops-where-are-we-and-how-did-we-get-here.html

Easy to deploy, hard to maintain

- ML applications are more complex because problems of DevOps plus ML issues
- Treatment of models as black boxes
- Changes in the data itself can affect how the system performs
- Good software engineering design practices don’t easily apply

D. Sculley, Gary Holt, Daniel Golovin, Eugene Davydov, Todd Phillips, Dietmar Ebner, Vinay Chaudhary, Michael Young, Jean-Francois Crespo, and Dan Dennison. 2015. ***Hidden technical debt in Machine learning systems****. In Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2 (NIPS'15). MIT Press, Cambridge, MA, USA, 2503-2511.*

ML Technical Debt

- Entanglement - *Changing Anything Changes Everything*, nothing is independent
- Pipeline management - ETL, training, validation, etc.
  - ETL is a whole thing of its own
- Hidden feedback loops
- Other tech debt
- Does it bring you joy? If not throw it out. (Spoiler - might not work out.)

Back to MLOps

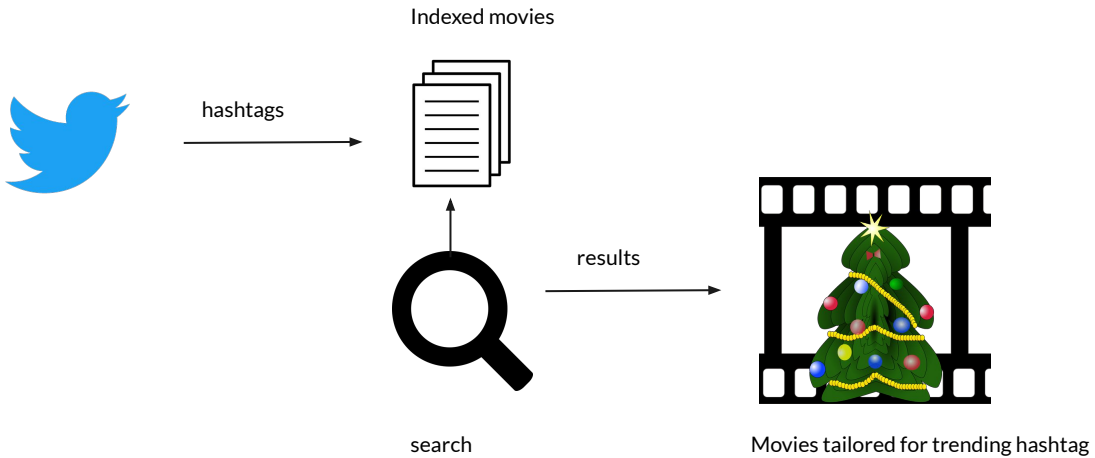
- Lots of AI projects don't make it to production
- Trying to apply processes to these challenges
- Kubeflow started in 2018
- Address the "Big 8" areas

https://www.socialg.tech/mlopsconcepts

Data Collection	Model Design
Data Processing	Model Training
Feature Engineering	Model Optimization
Data Labeling	Deployment and Monitoring

What We Are Building

A system to adjust movie search relevance based on current trends.



Example Search Results

Out-of-the-box “Family” movie results

1. The New Adventures of Pinocchio
2. Raising the Bar
3. A Christmas Star
4. LazyTown
5. Bling
6. Christmas Miracle
7. Soul Surfer
8. Air Bud: Golden Receiver
9. Patrick
10. Playing with Fire

“Family” movie results with “christmas” trending

1. A Christmas Carol
2. Unaccompanied Minors
3. Jingle All the Way
4. Jingle All the Way 2
5. The Grinch
6. A Christmas Snow
7. The 12 Dogs of Christmas
8. The Nutcracker
9. Saving Christmas
10. The Swan Princess Christmas

Components

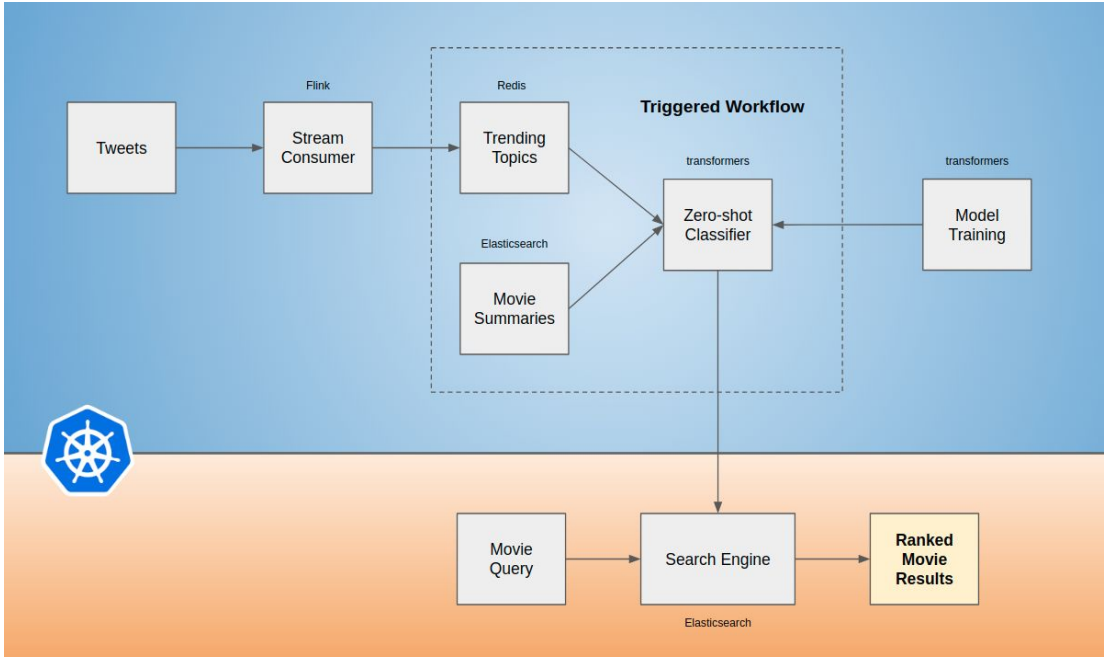
- A Twitter consumer (an Apache Flink application)

- A search index of movies (Elasticsearch)

- A natural language classifier (custom-trained zero-shot learning classifier)

- A search relevance tool (Quepid)
- A database and a cache (MySQL and Redis)

Architecture



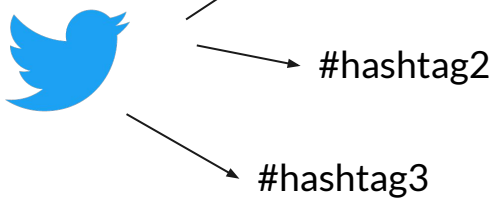
The Data Set

- A listing of movies and information such as title, genres, actors, and summary
- Subset of an export from The Movies Database
- Indexed movies into Elasticsearch


```
"562": {
   "genres": [
     {
       "id": 28,
       "name": "Action"
     },
     {
       "id": 53,
       "name": "Thriller"
     }
   ],
   "original_language": "en",
   "original_title": "Die Hard",
   "overview": "NYPD cop John McClane's plan to reconcile with his estranged wife is thrown for a serious loop when, minutes after he arrives at her office, the entire building is overtaken by a group of terrorists. With little help from the LAPD, wisecracking McClane sets out to single-handedly rescue the hostages and bring the bad guys down.",
```

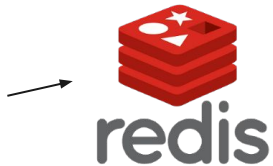
Twitter Stream Consumer

- Apache Flink job that maintains a map of hashtags to their count of occurrences.
- Sort the map by value to get a list of the top N trending hashtags.
- Persist that map to Redis.



Hashtag map of counts

hashtag	count
hashtag1	4
hashtag2	16
hashtag*N*	*xx*



How to classify movie overviews based on hashtags?

- We don’t know what might be trending tomorrow
- Hard to train a classification model using unknown labels
- We need something else… a zero-shot classifier

Natural Language Inference (NLI)

- Given a premise, determine if a hypothesis is:
  - true (entailment)
  - false (contradiction)
  - underdetermined (neutral)
- Also referred to as Recognizing Textual Entailment (RTE)

Premise	**Hypothesis**	**Label**
A soccer game with multiple males playing.	Some men are playing a sport.	entailment
A man inspects the uniform of a figure in some East Asian country.	The man is sleeping.	contradiction

http://nlpprogress.com/english/natural_language_inference.html

NLI Training Data

Sentence pairs manually labeled for entailment, contradiction, and neutral

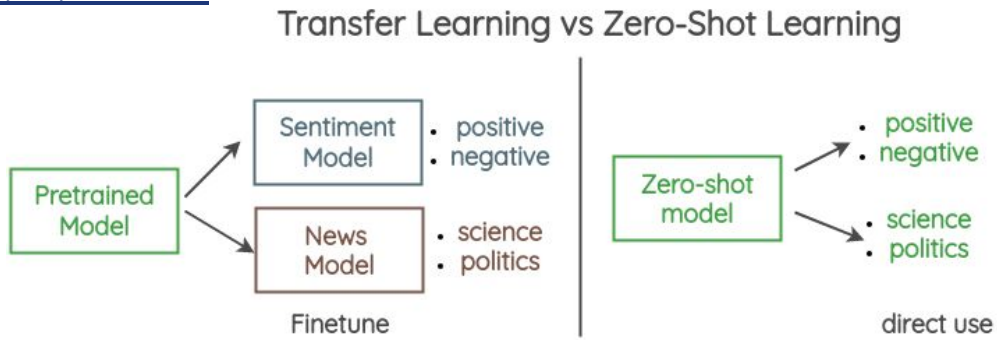

```
{
  "premise":"Two women are embracing while holding to go packages."
  "hypothesis":"The sisters are hugging goodbye while holding to go packages after just eating lunch."
  "label":1
}
```


SNLI available on Hugging Face Datasets

NLI Model as a Zero-shot Learning Classifier

- Classify text into one or more categories
- Categories do not have to be known at training time
  - Hashtags come and go!

- https://joeddav.github.io/blog/2020/05/29/ZSL.html



https://amitness.com/2020/05/zero-shot-text-classification/

Classifying Movie Overviews

Input

Hypothesis

This text is about _____.

hashtag(s)

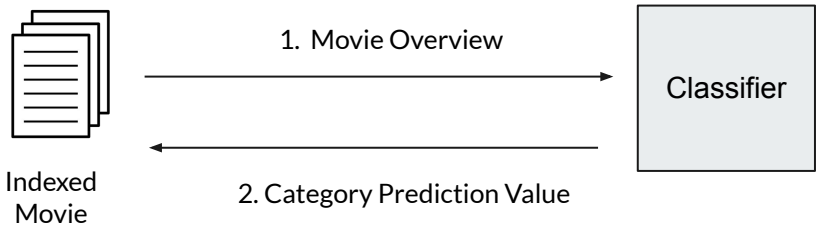
- Sequence (movie overview)
- Candidate labels (hashtags)

Output

Candidate Label	Probability
hashtag1	0.xx
hashtag2	0.xx

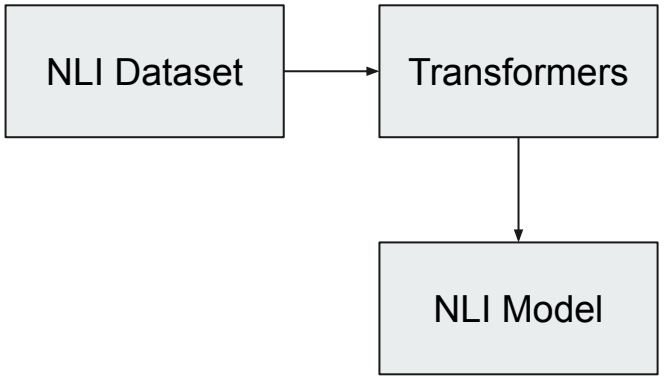
Using the Model

- Apply the zero-shot learning classifier for each of the movie overviews against the trending hashtag(s)
- Create a new field(s) for each movie that contains the value of the classification(s)
- Use the field(s) for ordering search results



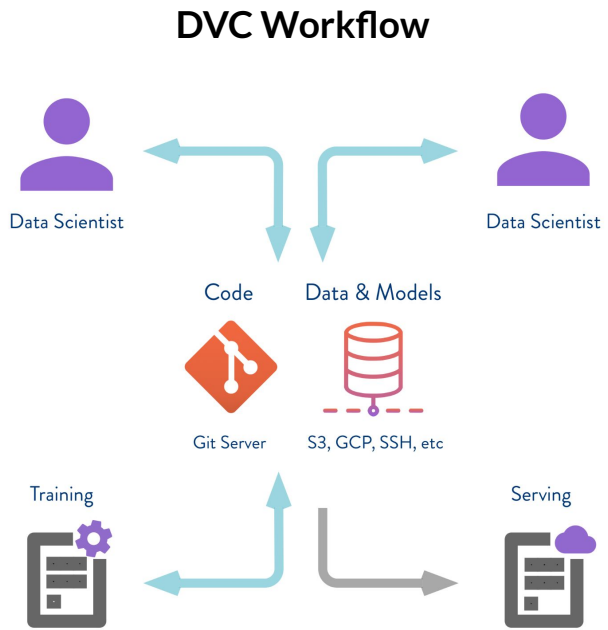
Model Training

- Use Hugging Face transformers and an NLI dataset
- Models stored alongside source control using DVC
- Training runs in a container



Model Versioning

- Models stored alongside source control using DVC
- Stores metadata about the models in Git while the models are stored in a backend store
- S3, GCP, SSH, NFS, etc.



https://dvc.org/doc/use-cases/sharing-data-and-model-files

Evaluating the Models

- Models must be updated
  - Trends come and go
  - New movie releases
  - Performance improvements

} stale model

- Made human judgments of movies against a few categories
- Persisted judgments to the database (`movie ID -> judgment value`)
- A search can compute the relevance score for the model search and compare it against the human-scored search
- Gives baseline performance

Evaluation Example

Human-judged movies

4 - A Christmas Carol
3 - Unaccompanied Minors
4 - Jingle All the Way
4 - Jingle All the Way 2
4 - The Grinch
3 - A Christmas Snow
4 - The 12 Dogs of Christmas
4 - The Nutcracker
4 - Saving Christmas
4 - The Swan Princess Christmas

“Family” movie results with “christmas”

4 - A Christmas Carol
0 - Predator
4 - Jingle All the Way
1 - Die Hard
1 - You've Got Mail
3 - A Christmas Snow
4 - The 12 Dogs of Christmas
4 - The Nutcracker
4 - Saving Christmas
4 - The Swan Princess Christmas

Deploying a Model

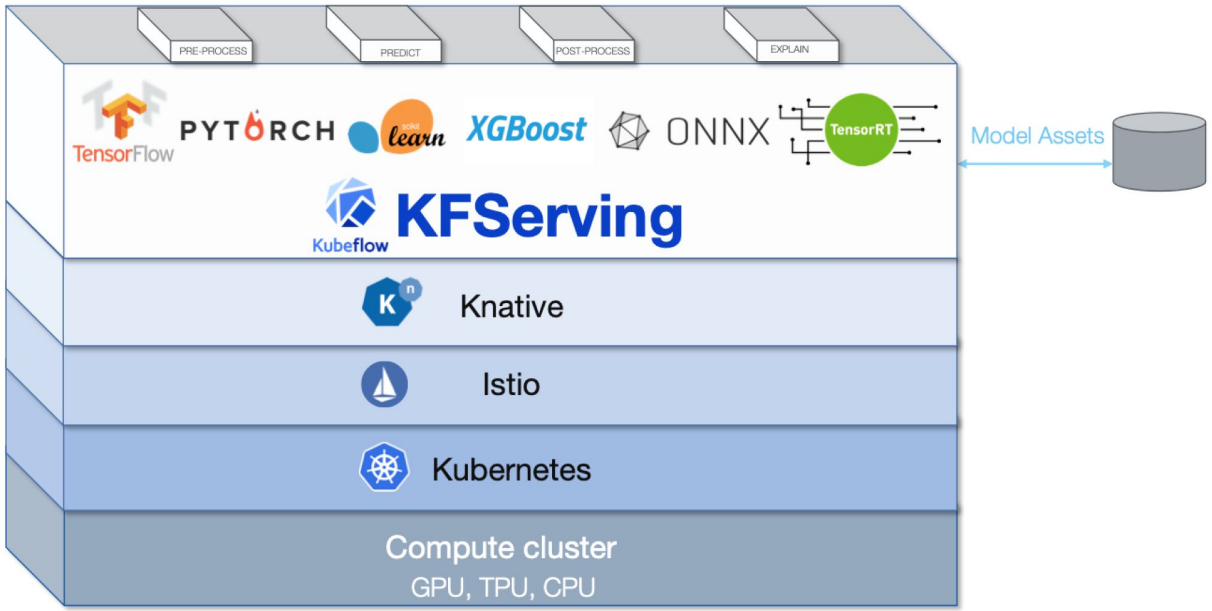
- Served using KFServing custom image
  - Encapsulates autoscaling, networking, health checks, etc.
  - Accessible over a REST interface
  - `curl -s http://localhost:8080/v1/models/zero-shot-classifier:predict -d @./input.json`
- Persisting the model
  - Can be pulled using DVC
  - Can be persisted to the Hugging Face model hub
- Our new classifier is ready to use!

KFServing

Inference on Kubernetes

KFServing supports popular ML frameworks

Custom image allows for using any model



https://github.com/kubeflow/kfserving

Serving Model with KFServing

Provide `init`, `load`, and `prediction` implementations

Run inside a Docker container

```
class KFServing_ZSC(kfserving.KFModel):
    def __init__(self, name: str):
        super().__init__(name)
        self.name = name
        self.ready = False

    def load(self):
        self.ready = True

    def predict(self, request: Dict) -> Dict:

        inputs = request["instances"]
        sequence = inputs[0]["sequence"]
        candidate_labels = inputs[0]["labels"]

        return classifier(sequence, candidate_labels, multi_class=True)
```

Deploying with KFServing

Deploy container to Kubernetes using custom resource

```
apiVersion: serving.kubeflow.org/v1alpha2
kind: InferenceService
metadata:
  labels:
    controller-tools.k8s.io: "1.0"
  name: zero-shot-learning
spec:
  default:
    predictor:
      custom:
        container:
          name: zero-shot-learning
          image: jzemerick/bbuzz-zero-shot:1.0
```

Monitoring

- All through the lifecycle:
  - Training
    - Monitor model training metrics
  - Deployment
    - Monitor model inference times
    - Inference not done at search time
  - Effectiveness
    - Monitor search relevance metrics

Code and Running

- Code: https://github.com/jzonthemtn/berlin-buzzwords-2021
- Follow the `README` using `docker-compose` to run everything locally

Why not use more of Kubeflow?

- You certainly can
- A bit heavier
- Good if you have multiple ML pipelines
- Intended to show principles

Summary

- MLOps may be applying DevOps concepts to ML but it brings many new challenges
- Important to address ML tech debt
  - Sculley, David, et al. "Machine learning: The high interest credit card of technical debt." (2014).
- A zero-shot classifier can help us label text for previously unseen categories
- Can use KFServing to deploy the model to Kubernetes
- With care we can get a ML project to production

Thanks!

Love to hear about similar projects!

Join the Relevance Slack - https://opensourceconnections.com/slack