NLP games и автоматизация пайплайнов с Kubeflow

Mikhail Rozhkov

web: ml-repa.ru
telegram: t.me/mlrepa

AI-hat game

Игра на загадывание и отгадывание слов.

Player 1:

- пытается объяснить вытянутое из шляпы слово с помощью набора неоднокоренных слов, которые называются по очереди

Other players:

- предлагают несколько вариантов отгадок

Scoring

- guess score: чем раньше игрок из Other players угадает вытянутое из шляпы слово, тем больше очков он получит.
- explain score: чем больше Other players угадают вытянутое из шляпы слово и чем раньше они это сделают, тем больше очков Player 1

AI Hat competition on Raiffeisen bootcamp 2019

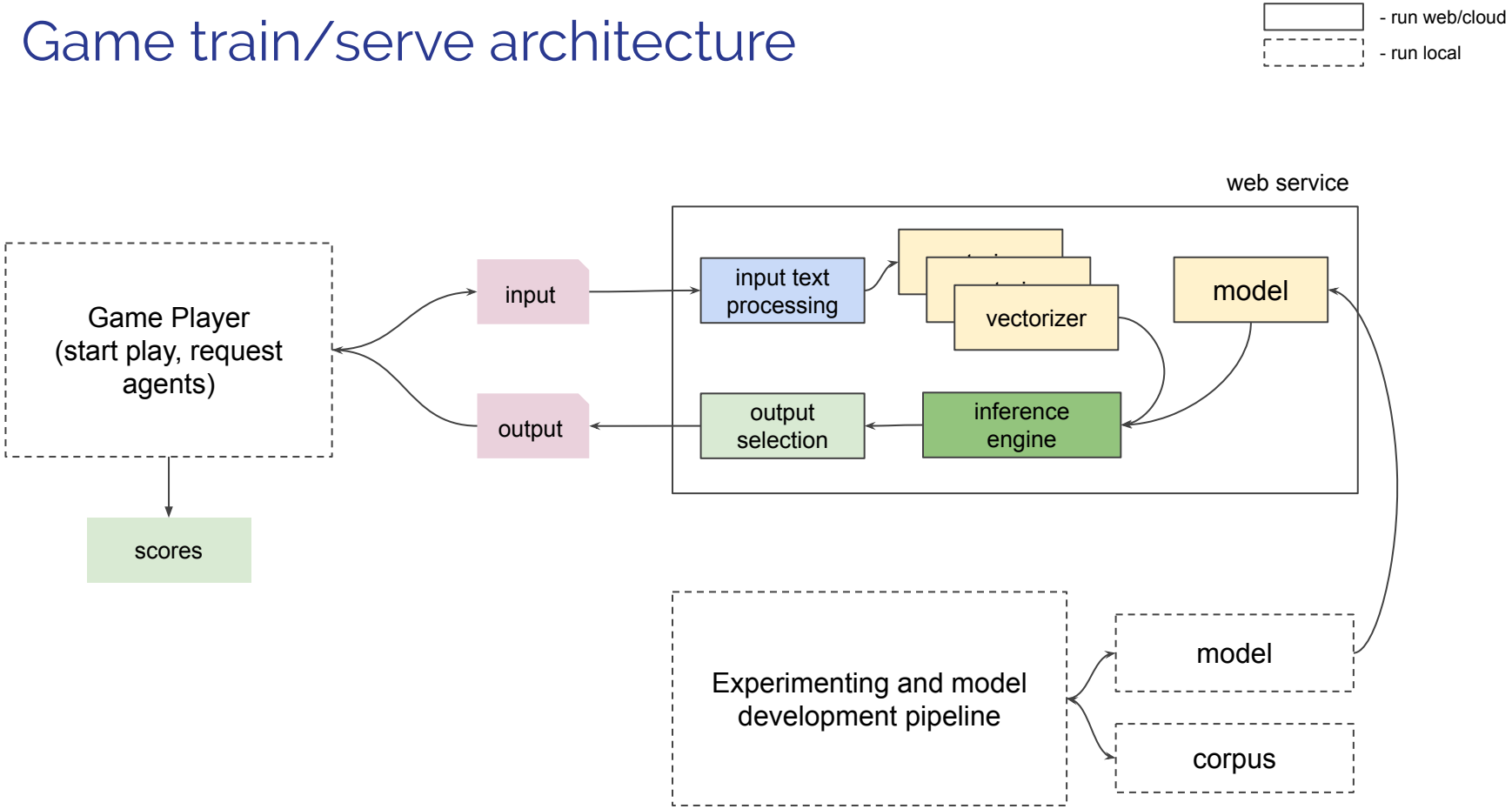
Task

- teams develops their own model, which implement the logic of guessing and explaining
- model serving as a web service with REST API endpoints
  - /guess
  - /explain

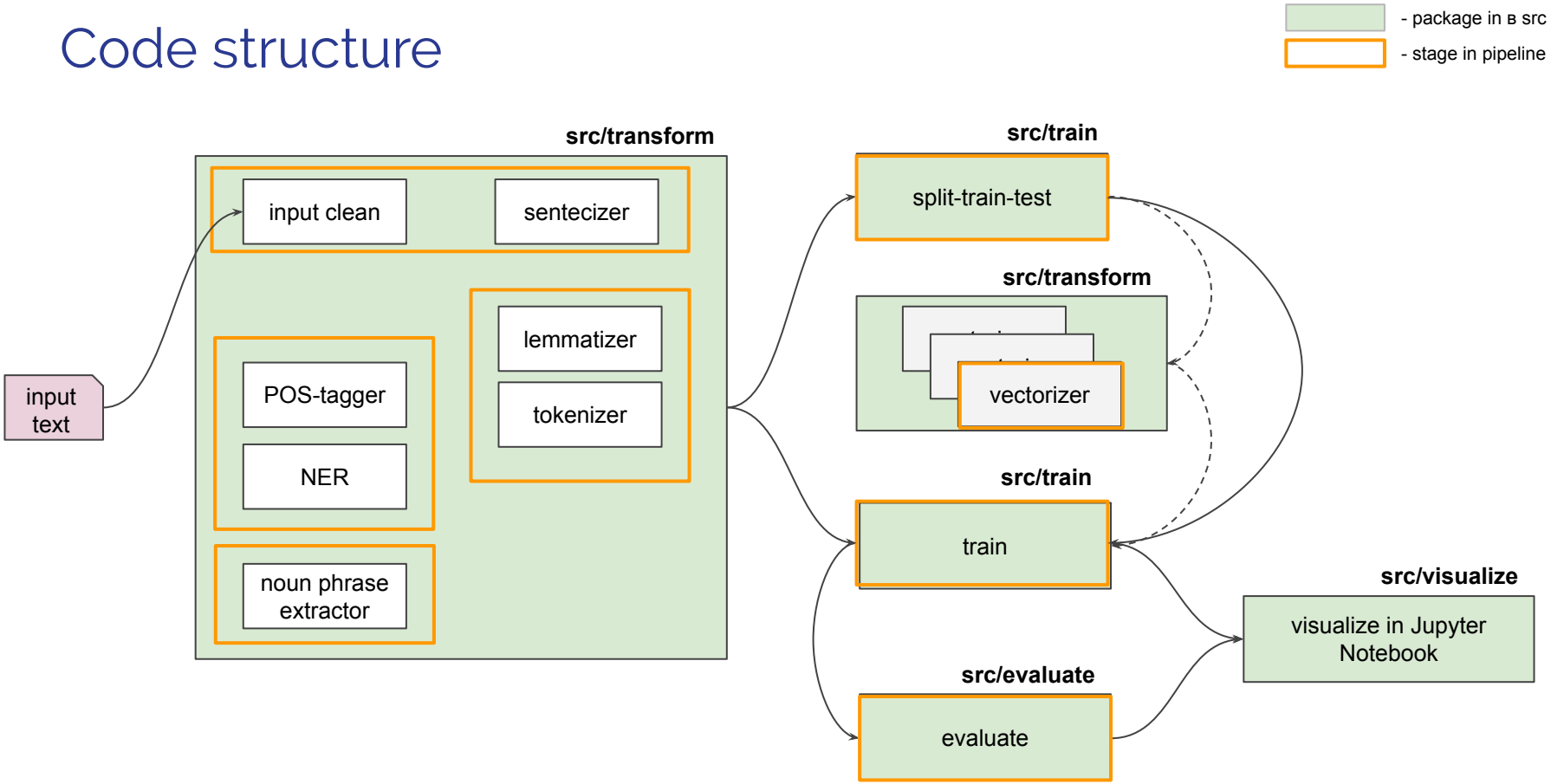
Text

- financial news
- words: financial terms

Game train/serve architecture



Code structure



Text processing

Методы:

1) Remove symbols and word reduction forms (- -> ‘ ‘, ‘ll -> will and etc.)
2) Remove links and tags (www, /, .com and etc.)
3) Lower-case
4) Lemmatization
5) Remove multiple spaces

Train embeddings (fasttext)

CBOW - for guessing

- dim=50, epoch=120, charNgram=(3, 4), wordNgram=3, window=5

SKIPGRAM - for explain

- dim=50, epoch=140, charNgram=(4, 6), wordNgram=3

Evaluation

Metrics

- total score = guess + explain
- separate validation set
- competition among models

Approach 1: dvc & mlflow

- dvc
  - pipelines automation
  - artifacts and models versioning
- mlflow
  - metrics tracking and experiment management

Approach 2: kubeflow

- pipeline configuration
- metrics tracking
- artifacts

Kubeflow: a platform for building ML products

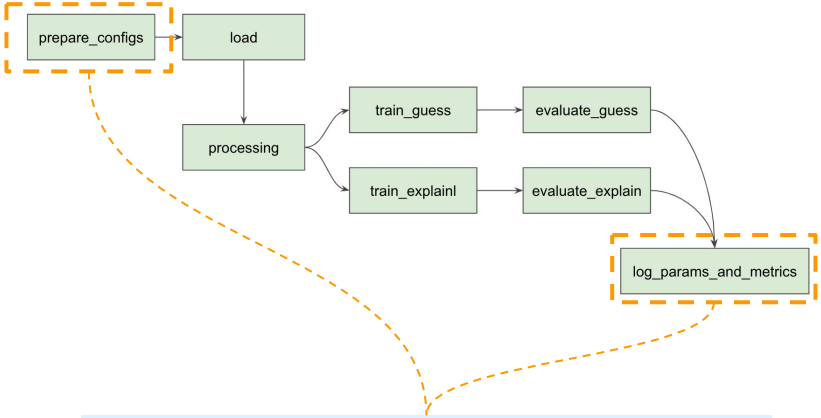
- *A curated set of compatible tools and artifacts that lays a foundation for running production ML apps*
- Run containers on Kubernetes cluster
  - Kubernetes runs everywhere
  - Enterprises can adopt shared infrastructure and patterns for ML and non ML services
- Key features
  - Easy, repeatable, portable deployments on a diverse infrastructure
  - Deploying and managing loosely-coupled microservices
  - Scaling based on demand

- Pipelines
- Notebooks
- TensorFlow model training
- Model serving
- Multi-framework

Kubeflow

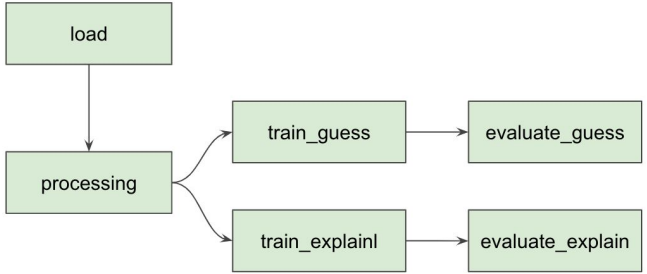
Pipeline

dvc + mlflow



DS решает вопросы с конфигом и логированием метрик/параметров*

kubeflow



kubeflow предлагает UI для конфига, логирует все inputs/outputs этапов**

* можно не выделять в отдельные этапы

** это не всегда удобно

Code structure

dvc + mlflow

```
config/
docker/ - Dockerfile
dvc/
src/
     game/
     pipelines/
          load.py
          processing.py
          ...
```

kubeflow

```
docker/
     base/ - Dockerfile
     load/ - Dockerfile
     processing/ - Dockerfile
     ...
src/
     full-pipeline/
          components/ - .yaml for each stage
          pipeline.py
     pipelines/
          load.py
          processing.py
          ...
```

Code structure

dvc + mlflow

```
config/
docker/ - Dockerfile
dvc/
      .dvc files
src/
      game/
      pipelines/
            load.py
            processing.py
            ...
```

kubeflow

```
docker/
      base/ - Dockerfile
      load/ - Dockerfile
      processing/ - Dockerfile
      ...
src/
      full-pipeline/
            components/ - .yaml for each stage
            pipeline.py
      pipelines/
            load.py
            processing.py
            ...
```

different approach to config pipeline

Code structure

dvc + mlflow

```
config/
docker/ - Dockerfile
dvc/
src/
    game/
    pipelines/
        load.py
        processing.py
        ...
```

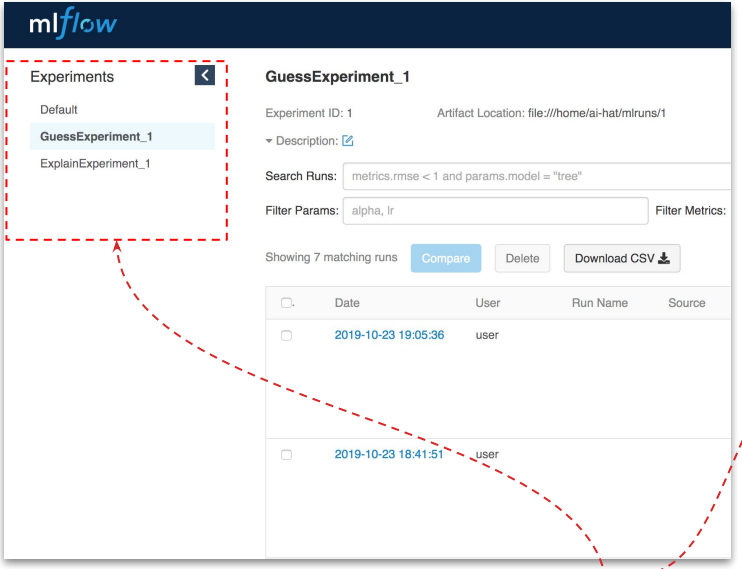
kubeflow

```
docker/
    base/ - Dockerfile
    load/ - Dockerfile
    processing/ - Dockerfile
    ...
src/
    full-pipeline/
        components/ - .yaml for each stage
        pipeline.py
    pipelines/
        load.py
        processing.py
        ...
```

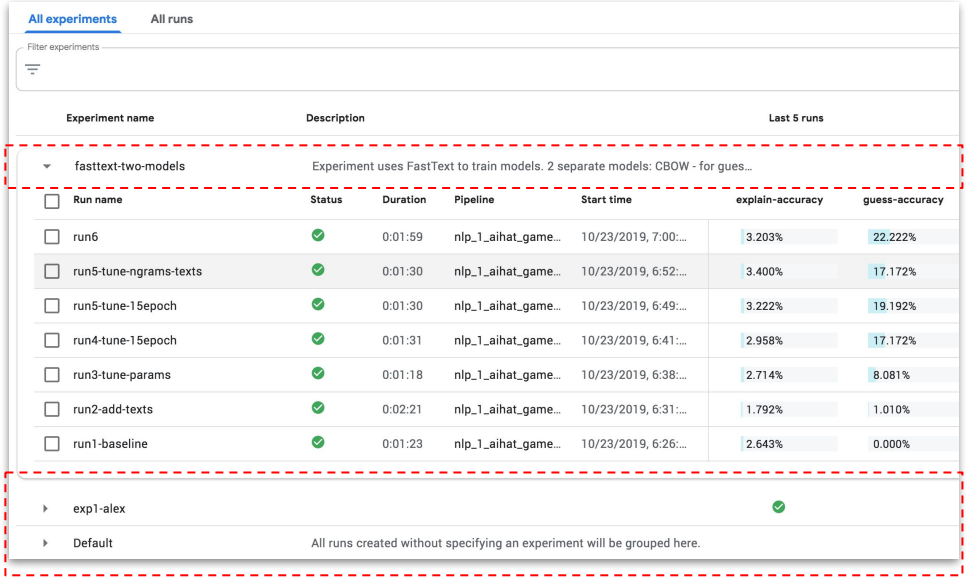
~ same .py for stages

Find experiments / runs

dvc + mlflow



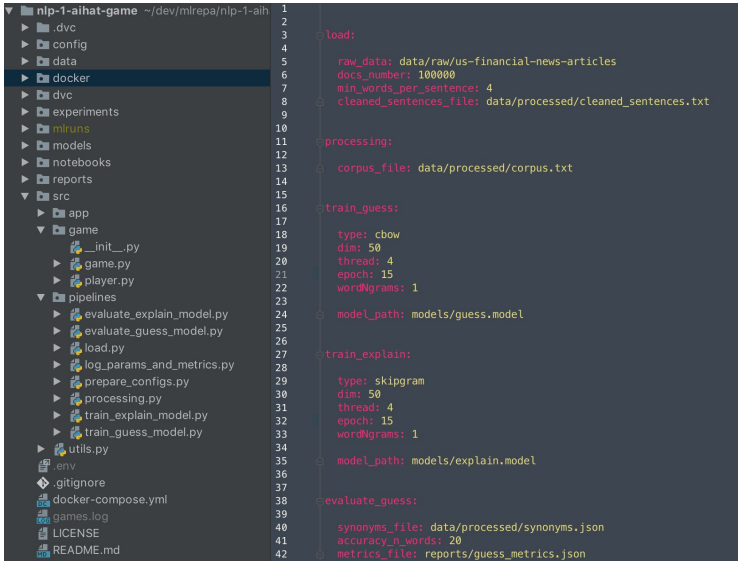
kubeflow



`list of experiments`

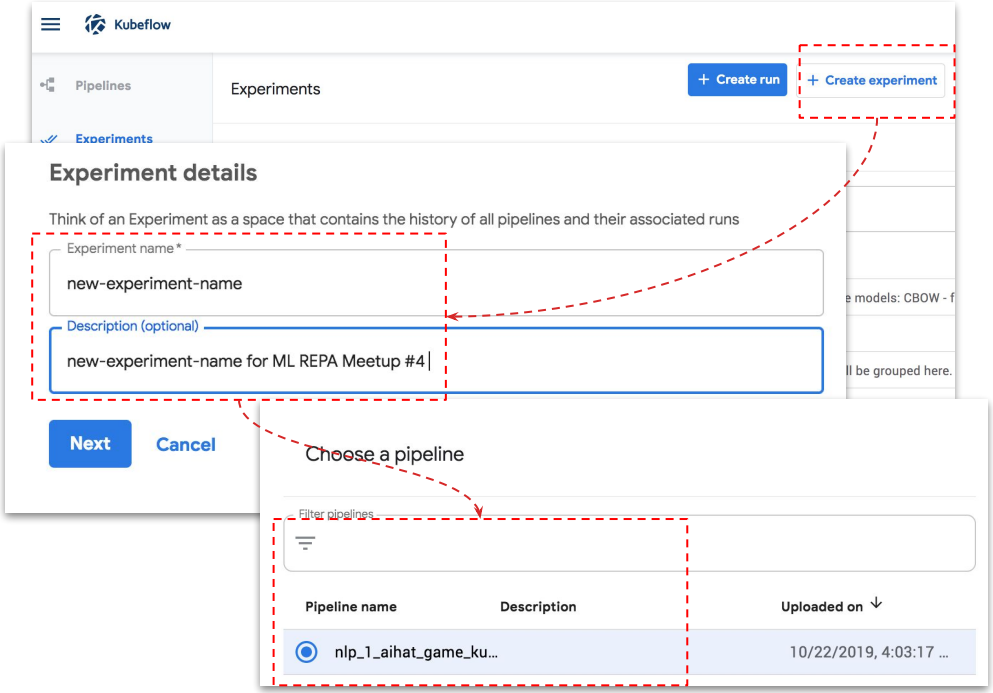
New experiment

dvc + mlflow




```
$ dvc repro last_pipeline_stage.dvc
```

kubeflow



Run experiment (kubeflow)

Experiments

+ Create run | + Create experiment | Compare runs | Clone run | Archive | Refresh

All experiments | All runs

Filter experiments

Experiment name	Description
new-experiment-name	new-experiment-name for ML REPA Mee
fasttext-two-models	Experiment uses FastText to train mode

Run name	Status	Duration	Pipeline	-accuracy
run6	✓	0:01:59	nlp_1_aihat_g	222%

Run parameters

Specify parameters required by the pipeline

project	vision-230607
bucket	vision-storage1
raw-data	nlp-1-aihat-game-data/short_dataset.zip
docs-number	100000
min-words-per-sentence	4
cleaned-sentences-file	nlp-1-aihat-game-data/cleaned_sentences_file.txt
corpus-file	nlp-1-aihat-game-data/corpus.txt
synonyms-file	nlp-1-aihat-game-data/synonyms.json
guess-model-type	cbow
guess-model-dim	70
guess-model-thread	

Metrics tracking

dvc + mlflow

```
log_experiment(
    experiment_name=current_experiment,
    client=mlflow_client,

    common_params={
        'docs_number': docs_number,
        'min_words_per_sentence': min_words_per_sentence
    },

    train_params={
        'model_type': train_explain_config.get('type'),
        'model_dim': train_explain_config.get('dim'),
        'model_epoch': train_explain_config.get('epoch'),
        'model_wordNgrams': train_explain_config.get('wordNgrams'),
    },

    metrics=json.load(open(explain_metrics_file))
)
```

mlflow.log_metric(‘name’, value)
в коде

https://www.mlflow.org/docs/latest/tracking.html

kubeflow

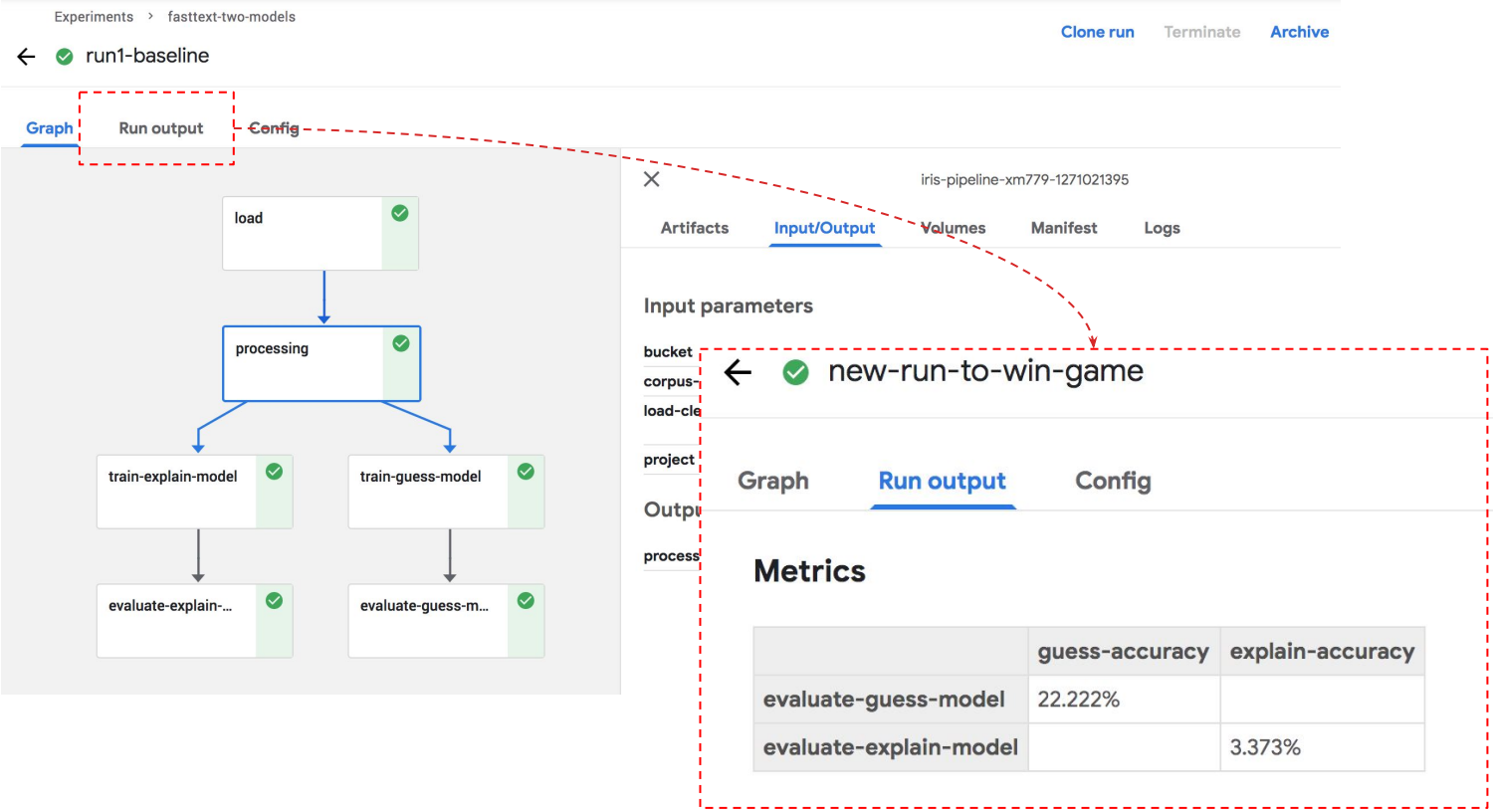
```
name: evaluate guess model
description: evaluate guess model
inputs:
  - {name: model_path, type: String, description: 'path to guess model'}
  - {name: synonyms_file, type: String, description: 'path to synonyms json'}
  - {name: accuracy_n_words, type: Integer, description: 'word count to get from model to calculate accuracy'}
  - {name: metrics_file, type: String, description: 'path to model'}
outputs:
  - {name: met
  - {name: MLP

implementation:
  container:
    image: 'gc
    args: [
      '--model
      '--synon
      '--accur
      '--metri

      '--metri
    ]
    fileOutput
      MLPipe
```

```
logger.debug('write metrics')
metrics = {
    'metrics': [{
        'name': 'guess-accuracy',
        'numberValue': accuracy,
        'format': 'PERCENTAGE',
    }]
}
with open('/mlpipeline-metrics.json', 'w') as f:
    json.dump(metrics, f)
```

определяем в конфиге MLPipeline Metrics
+ сохраняем .json файл в коде

https://www.kubeflow.org/docs/pipelines/sdk/pipelines-metrics/

Metrics tracking for a Run (kubeflow)



Metrics tracking for a Run (mlflow)

GuessExperiment_1 > **Run 4f6051f36e98433d890888c362a7923f**

Date: 2019-10-23 19:05:36

Run ID: 4f6051f36e98433d890888c362a7923f

Source:

User: user

Notes

None

Parameters

Name	Value
docs_number	100000
min_words_per_sentence	4
model_dim	50
model_epoch	15
model_type	cbow
model_wordNgrams	1

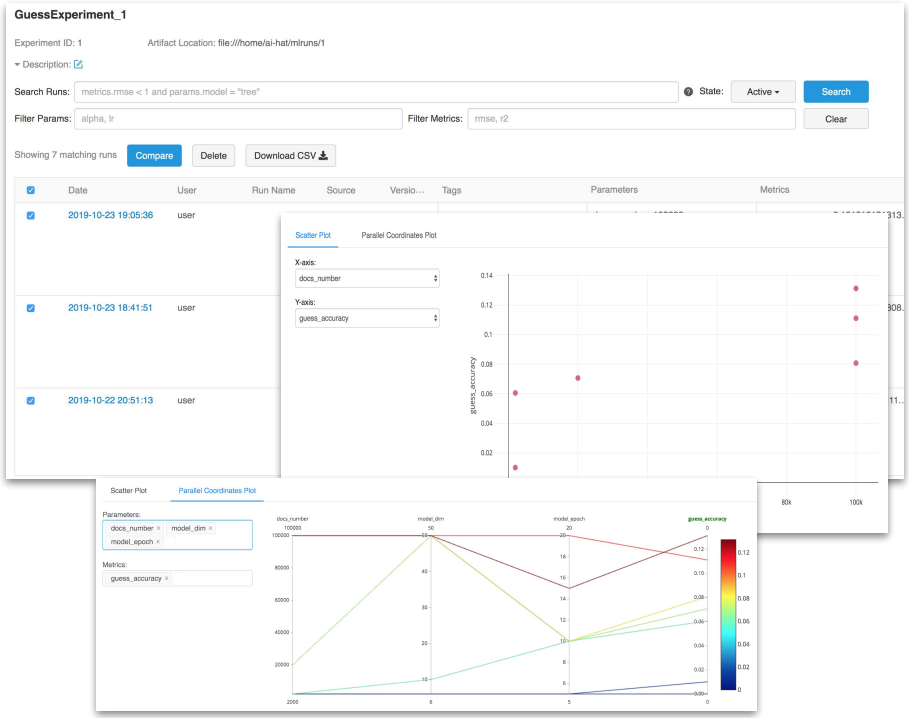
Metrics

Name	Value
guess_accuracy	0.131

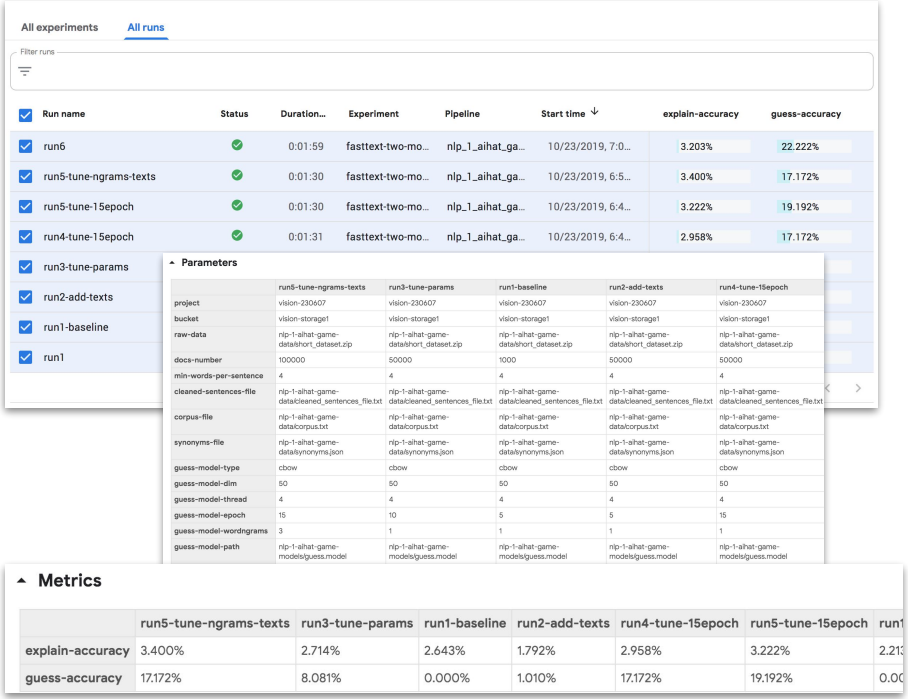
guess_accuracy и explain_accuracy можно логировать в одном RUN

Benchmark results

dvc + mlflow



kubeflow

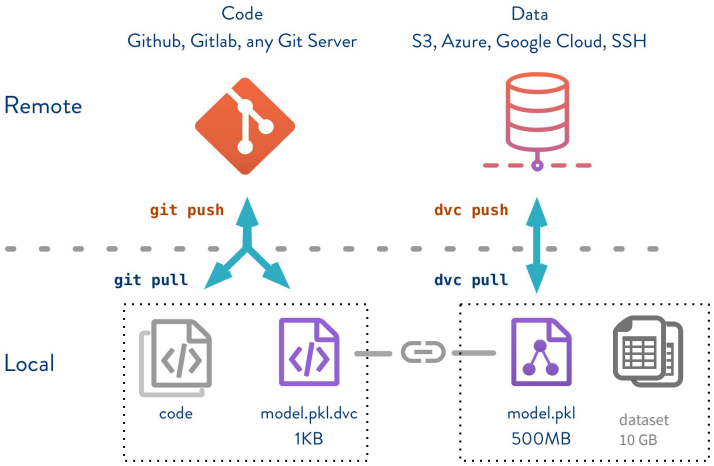


	run5-tune-ngrams-texts	run3-tune-params	run1-baseline	run2-add-texts	run4-tune-15epoch
project	vision-230607	vision-230607	vision-230607	vision-230607	vision-230607
bucket	vision-storage1	vision-storage1	vision-storage1	vision-storage1	vision-storage1
raw-data	nlp-1-aihat-game-data/short_dataset.zip	nlp-1-aihat-game-data/short_dataset.zip	nlp-1-aihat-game-data/short_dataset.zip	nlp-1-aihat-game-data/short_dataset.zip	nlp-1-aihat-game-data/short_dataset.zip
docs-number	100000	50000	1000	50000	50000
min-words-per-sentence	4	4	4	4	4
cleaned-sentences-file	nlp-1-aihat-game-data/cleaned_sentences_file.txt	nlp-1-aihat-game-data/cleaned_sentences_file.txt	nlp-1-aihat-game-data/cleaned_sentences_file.txt	nlp-1-aihat-game-data/cleaned_sentences_file.txt	nlp-1-aihat-game-data/cleaned_sentences_file.txt
corpus-file	nlp-1-aihat-game-data/corpus.txt	nlp-1-aihat-game-data/corpus.txt	nlp-1-aihat-game-data/corpus.txt	nlp-1-aihat-game-data/corpus.txt	nlp-1-aihat-game-data/corpus.txt
synonyms-file	nlp-1-aihat-game-data/synonyms.json	nlp-1-aihat-game-data/synonyms.json	nlp-1-aihat-game-data/synonyms.json	nlp-1-aihat-game-data/synonyms.json	nlp-1-aihat-game-data/synonyms.json
guess-model-type	cbow	cbow	cbow	cbow	cbow
guess-model-dim	50	50	50	50	50
guess-model-thread	4	4	4	4	4
guess-model-epoch	15	10	5	5	15
guess-model-wordngrams	3	1	1	1	1
guess-model-path	nlp-1-aihat-game-models/guess.model	nlp-1-aihat-game-models/guess.model	nlp-1-aihat-game-models/guess.model	nlp-1-aihat-game-models/guess.model	nlp-1-aihat-game-models/guess.model

	run5-tune-ngrams-texts	run3-tune-params	run1-baseline	run2-add-texts	run4-tune-15epoch	run5-tune-15epoch	run1
explain-accuracy	3.400%	2.714%	2.643%	1.792%	2.958%	3.222%	2.21
guess-accuracy	17.172%	8.081%	0.000%	1.010%	17.172%	19.192%	0.00

Models / artifacts

dvc + mlflow



models/artifact versioning with DVC

kubeflow

Artifacts | Input/Output | Volumes | Manifest | Logs

Input parameters

bucket	vision-storage1
explain-model-dim	70
explain-model-epoch	20
explain-model-path	nlp-1-aihat-game-models/explain.model
explain-model-thread	4
explain-model-type	skipgram
explain-model-wordngrams	2
processing-corpus-file-output	gs://vision-storage1/nlp-1-aihat-game-data/corpus.txt
project	vision-230607

Output parameters

train-explain-model-model-path-output	gs://vision-storage1/nlp-1-aihat-game-models/explain.model

need manual edit for model path each run

Original image source: https://dvc.org/doc/use-cases/data-and-model-files-versioning

Benchmark approaches of DVC, MLflow and kubeflow

	DVC	MLflow	kubeflow
Artifacts version control (models, datasets, etc.)	yes dvc run args	yes log_artifact()	yes* via metadata API
Pipeline execution DAG	yes	no*	yes
Caching of intermediate results	yes	no	no
Experiment management (tracking metrics, comparison, visualization)	yes-no*	yes	yes
Metadata	.dvc files	params, metrics, artifacts meta	kfmd library
Deployment/serving	no	yes	yes
Works locally	yes	yes	no*

* not out of the box or not flexible enough but possible to do/use/hack

Benchmark approaches of DVC, MLflow and kubeflow

	DVC	MLflow	kubeflow
Learning rate level (for DS)	**moderate** (git, shell cmd)	**low** (python only)	**high*** *k8s, GCP
Efforts for start with	**moderate**	**low**	**high***
Cost of money	**free**	**free**	**k8s cluster**
Maintainability / customization efforts	**low**	**low**	**high** 50+ microservices

* not out of the box or not flexible enough but possible to do/use/hack

Benchmark approaches of DVC, MLflow and kubeflow

	DVC	MLflow	kubeflow
pipelines	Complex pipeline with intermediate data saved into separate files. No duplicated computations and copy of artifacts.	Simple pipelines, one model. Serving model out of the box.	Pipelines with different resources requirements.
cool feature	Handful for experimentation local or collaboration (shared resources).	Nice UI for tracking metrics/params and experiment benchmark.	Reusable components, experiments benchmark and computation graph visualization
reproducibility	Reproducibility out of the box. Easy to checkout to previous version.	Need to save a copy for all data/code and artifacts to get reproducibility	Work in progress to versioning but still many drawbacks. Users' responsibility.

Thank you

Mikhail Rozhkov

ml-repa.ru

Some links

- DVC tutorials
- MLflow tracking
- Kubeflow pipelines quickstart
- Reproducibility in Machine Learning
- Kubeflow v0.6: support for artifact tracking, data versioning & multi-user
- The Data Science Bill of Rights
- KFServing