Benchmark of approaches to NLP pipelines automation: AI plays words games

Mikhail Rozhkov

ml-repa.ru

AI-hat game

Let’s write down a few random words on separate pieces of paper and put them into a hat.

Player 1:

- select random word from a hat
- try to explain this word using not rooting words

Other players:

- get several guesses

Scoring

- the earlier a player get the correct guess, the more points he will receive as well as the Player 1

AI Hat competition on Raiffeisen bootcamp 2019

Task

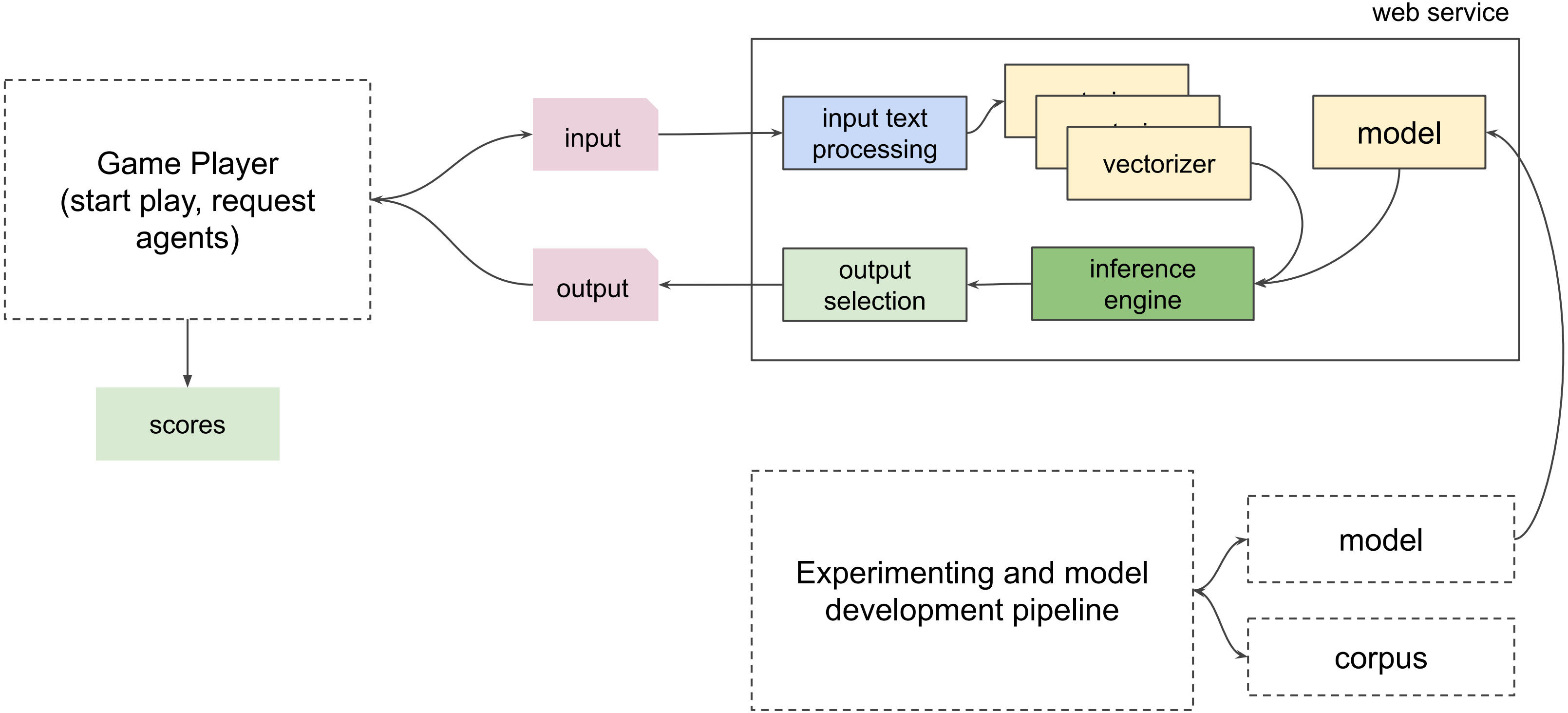
- teams develops their own model, which implement the logic of guessing and explaining
- model serving as a web service with REST API endpoints
  - /guess
  - /explain

Text

- financial news
- words: financial terms

Game train/serve architecture

- run web/cloud
- run local



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

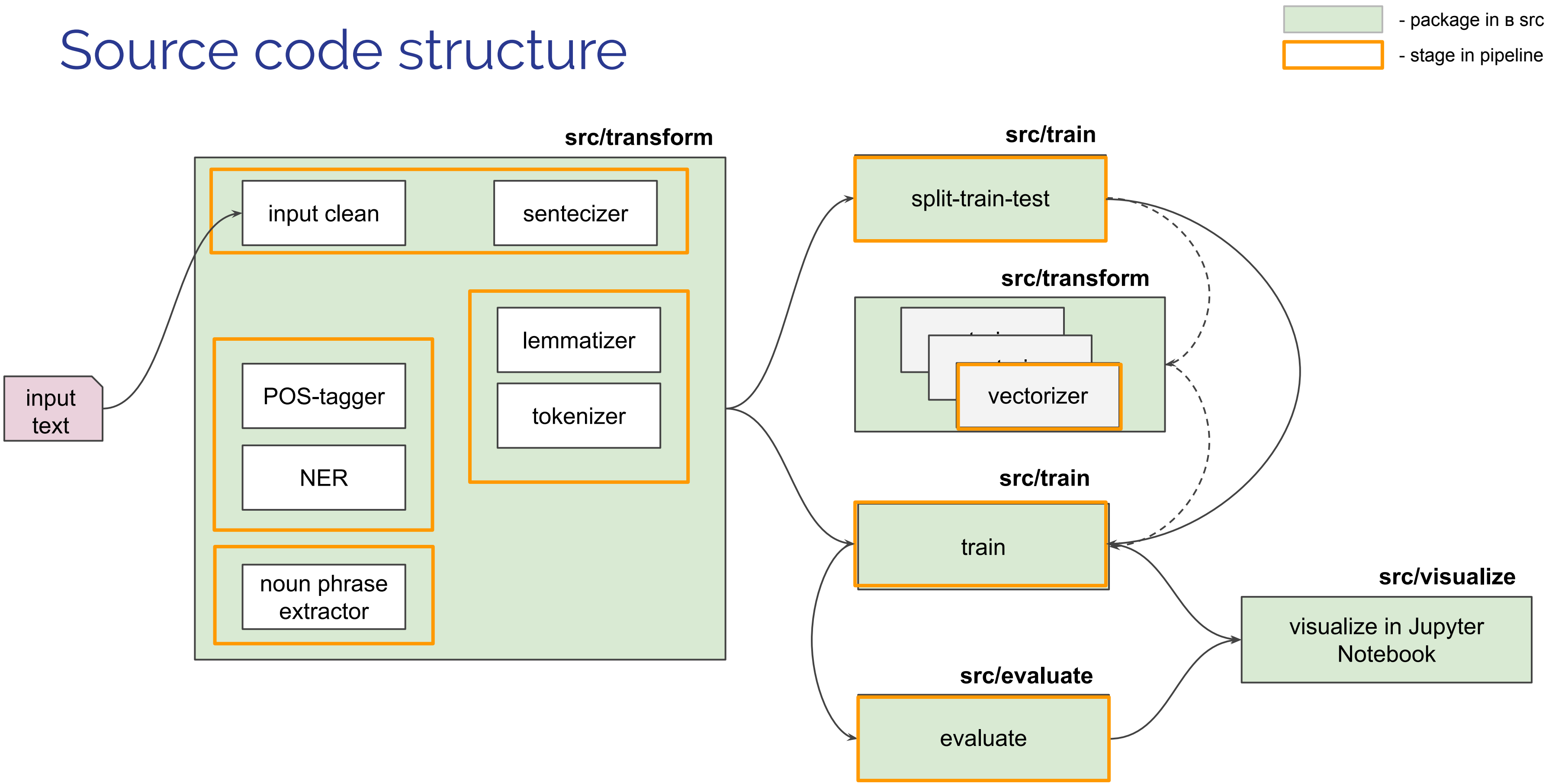
- dim=50, epoch=140, charNgram=(4, 6), wordNgram=3

Evaluation

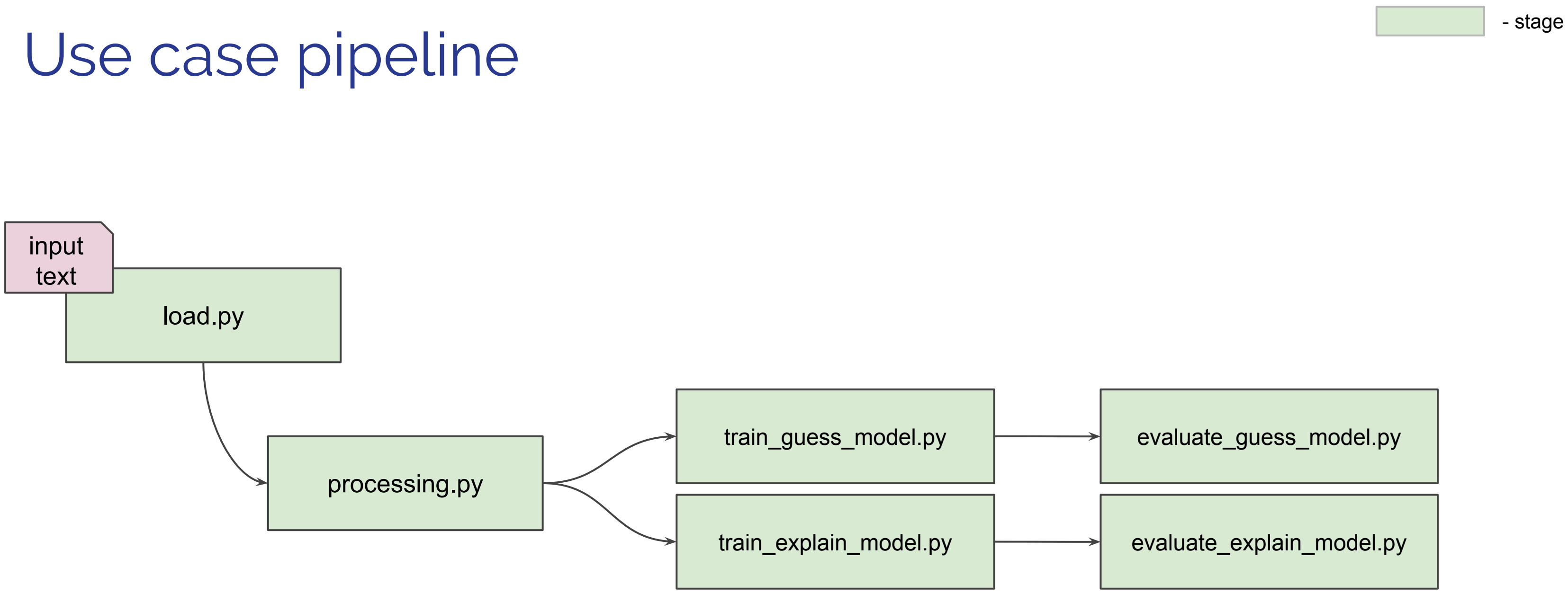
Metrics

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

Source code structure



Use case pipeline



Approach 1: dvc & mlflow

1. show code structure
2. show how to run
3. metrics tracking
4. models / artefacts
5. how to run new experiment

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

Approach 2: kubeflow

1. show code structure
2. show how to run
3. metrics tracking
4. models / artefacts
5. how to run new experiment

- 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

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

Thank you

Mikhail Rozhkov

ml-repa.ru

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.

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