Software Engineering for Artificial Intelligence

Software Architecture of AI-enabled Systems

Marcel de Boer, Zlatko Kolev

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Motivation & Antipatterns

Architectural Patterns

Design Patterns

Where should AI live?

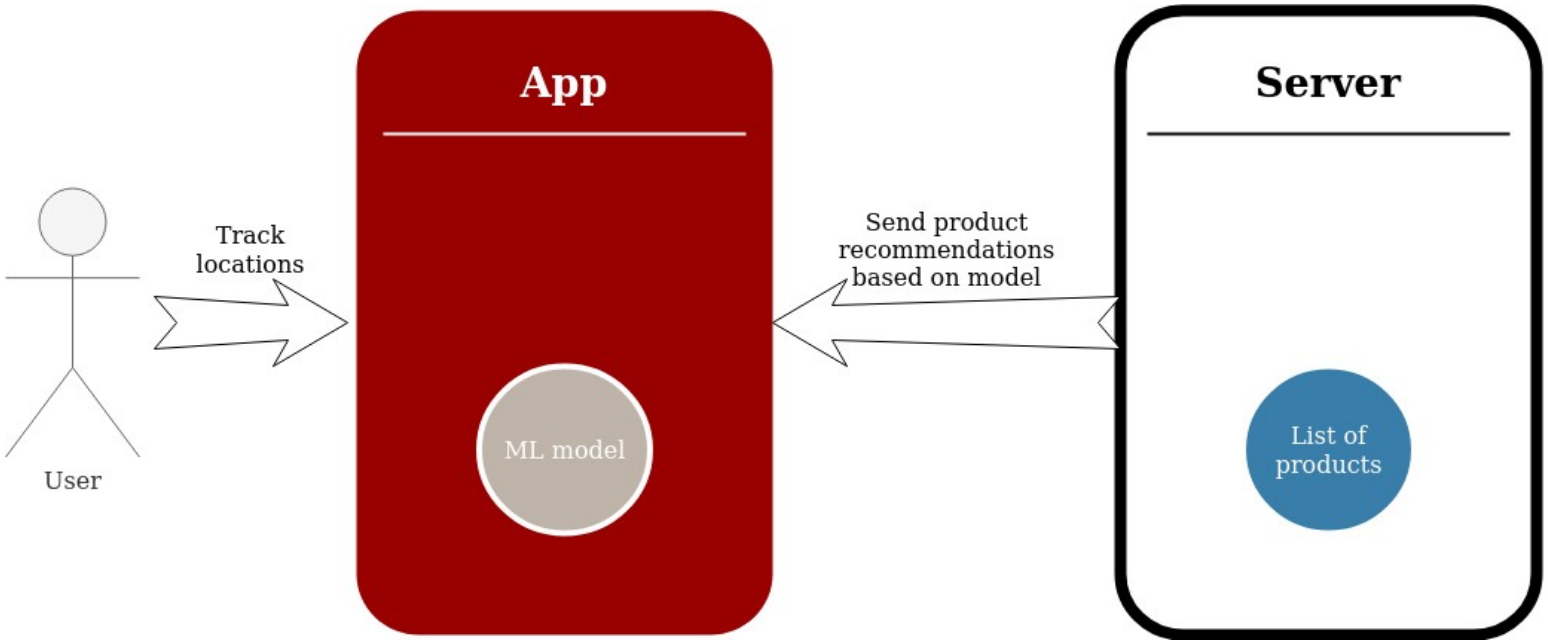
AI Platforms & Organizing Intelligence

Motivation

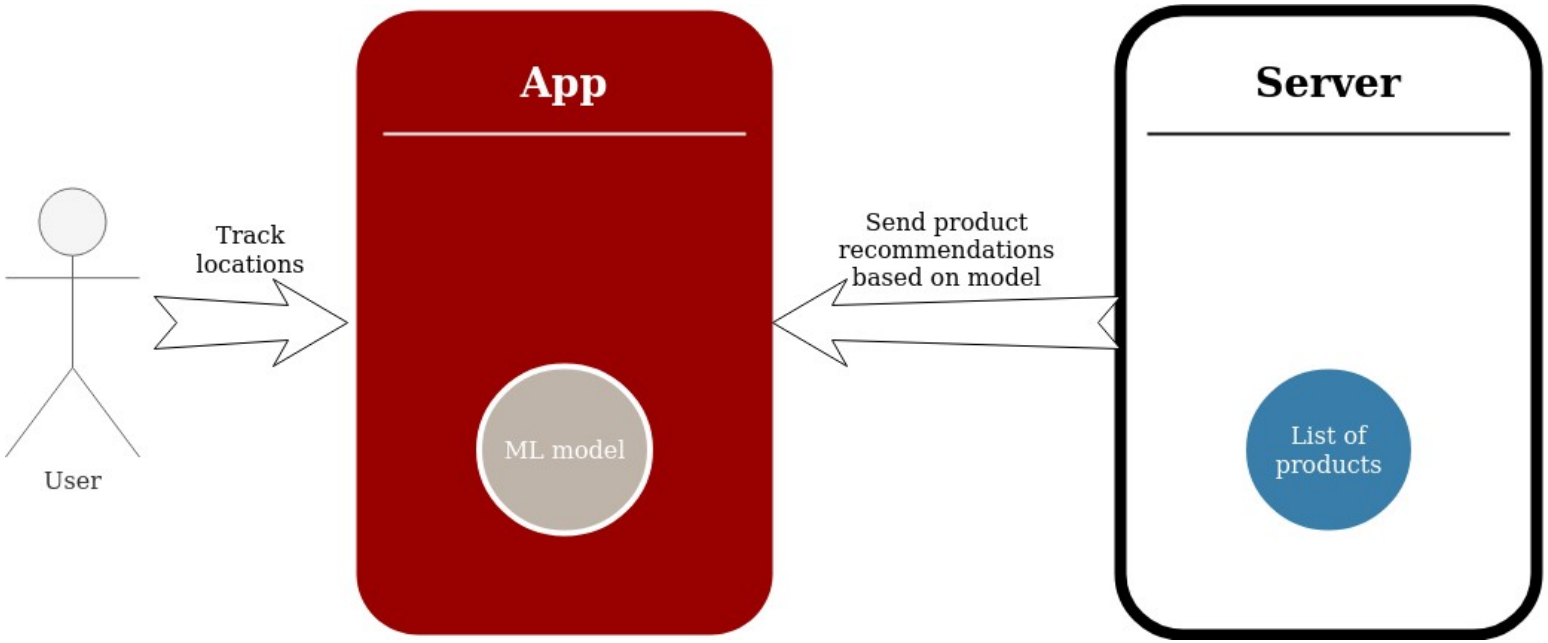
TECHNISCHE UNIVERSITÄT DARMSTADT

- Imagine that you want to implement an app which recommends products based on a user's activity...

Team A


App
Server
Track
locations
Send product
recommendations
based on model
User
ML model
List of
products

Team A



What are your thoughts about this architecture?

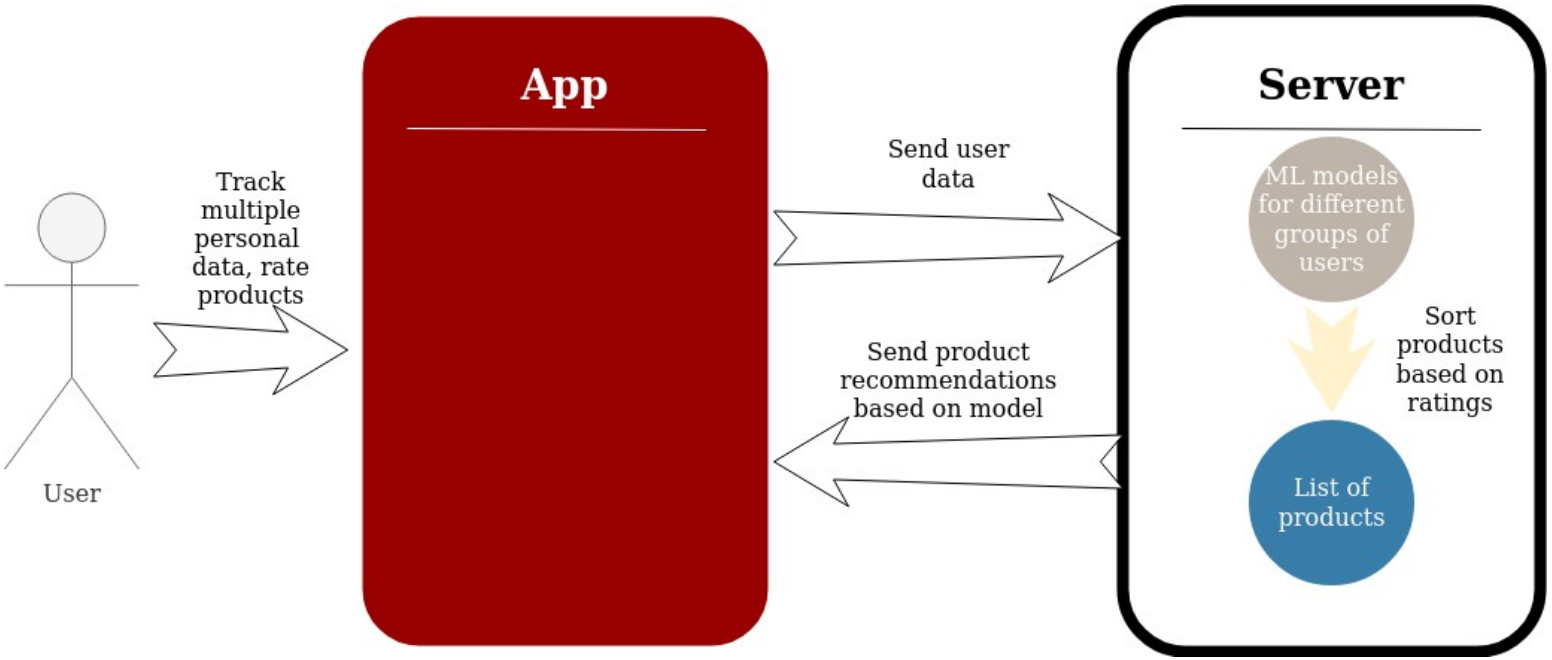
Team A: Possible Problems

TECHNISCHE UNIVERSITÄT DARMSTADT

- Data Collection and new data types
  → **Pipeline Jungle**
- Using third-party ML code
  → **Glue Code**
- ML on the smartphone
  → **Big Ass Script, Experimental Codepaths**

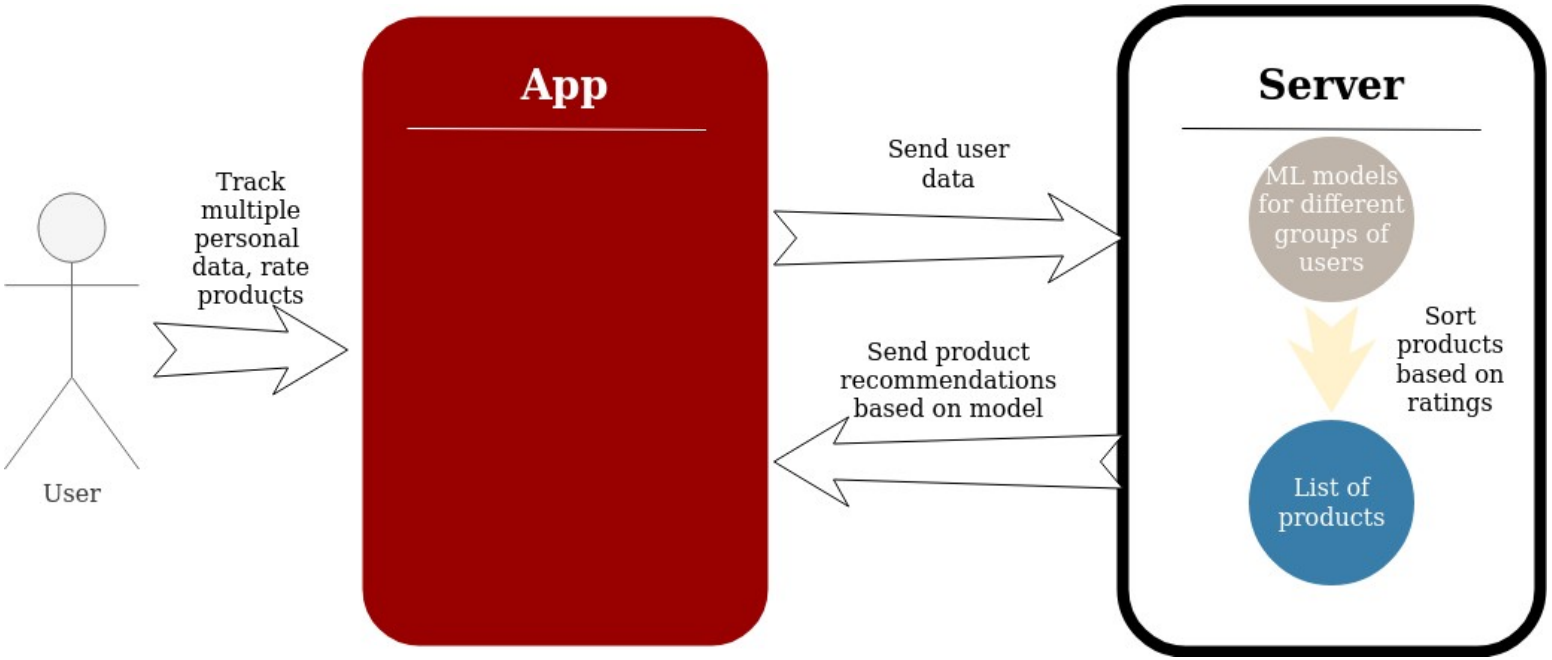
Team B

TECHNISCHE UNIVERSITÄT DARMSTADT



Team B

TECHNISCHE UNIVERSITÄT DARMSTADT



What are your thoughts about this architecture?

Team B: Possible Problems

TECHNISCHE UNIVERSITÄT DARMSTADT

- Use different 3rd party ML Models
  → **Glue code, Multiple-language smell**

- Distributed system
  → **Abstraction debt**

- Different models might sort the products differently
  → **Undeclared consumers**

Technical Debt

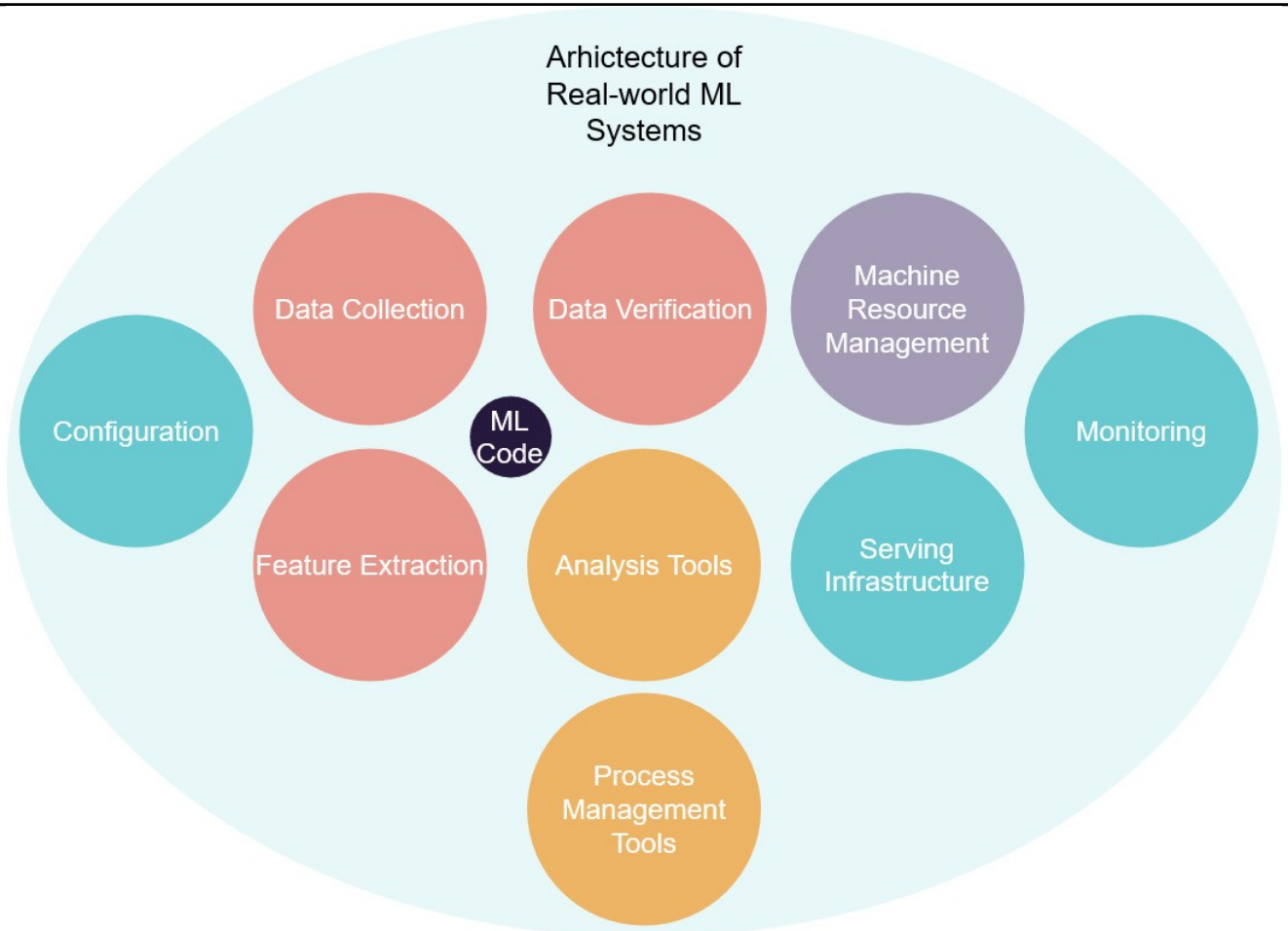
TECHNISCHE UNIVERSITÄT DARMSTADT

- Machine Learning doesn't come for free!

- Main challenges
  - Entanglement → CACE principle
  - Correction Cascades
  - Data Dependencies
  - Feedback Loops

- Anti-Patterns!

Architectures of AI-enabled Systems



Based on Figure 1 by D. Sculley et al. 2015. **Hidden technical debt in Machine learning systems**.
https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf, last access 01.06.20

Anti-Patterns

TECHNISCHE UNIVERSITÄT DARMSTADT

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Motivation & Antipatterns

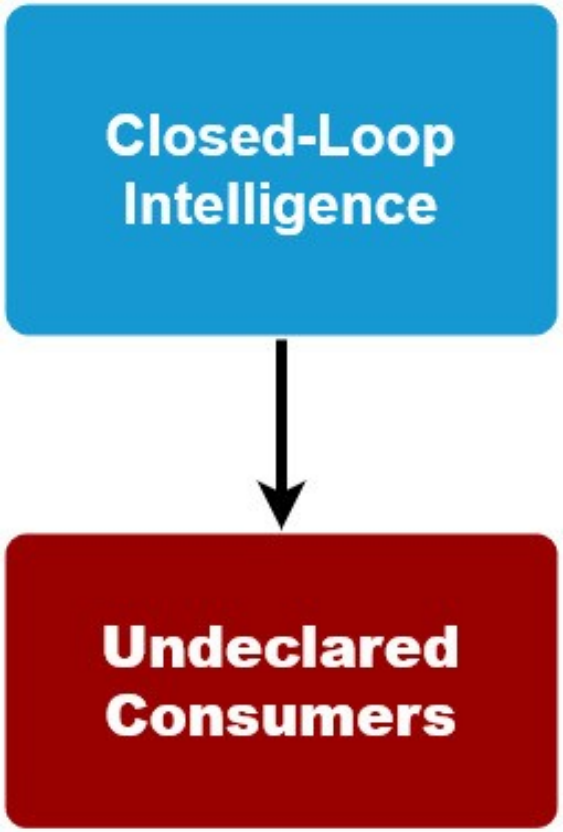
Architectural Patterns

Design Patterns

Where should AI live?

AI Platforms & Organizing Intelligence

Architecture Patterns: Closed-Loop Intelligence



- Connect machine learning to the user and close the loop
- Clear interactions with implicit and direct outputs
- Avoid hidden feedback loops

Architecture Patterns: Closed-Loop Intelligence

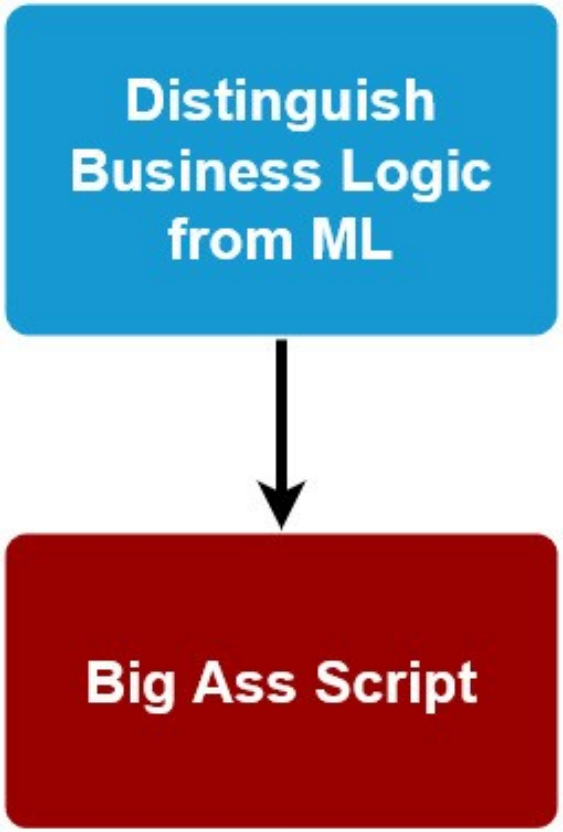
TECHNISCHE UNIVERSITÄT DARMSTADT

- Runtime for loading and executing models, and providing users with the right experiences
  → **Intelligence Runtime**

- Coordination between runtime, delivery, monitoring and creation
  → **Intelligence Creation Environment**

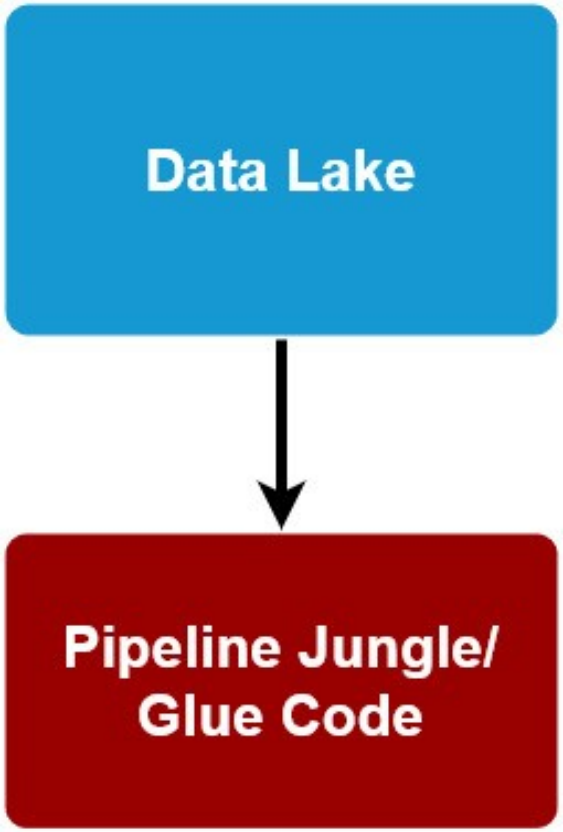
- Adapt to the changing world
  → **Intelligence Management, Telemetry**

Architecture Patterns: Distinguish Business Logic from ML



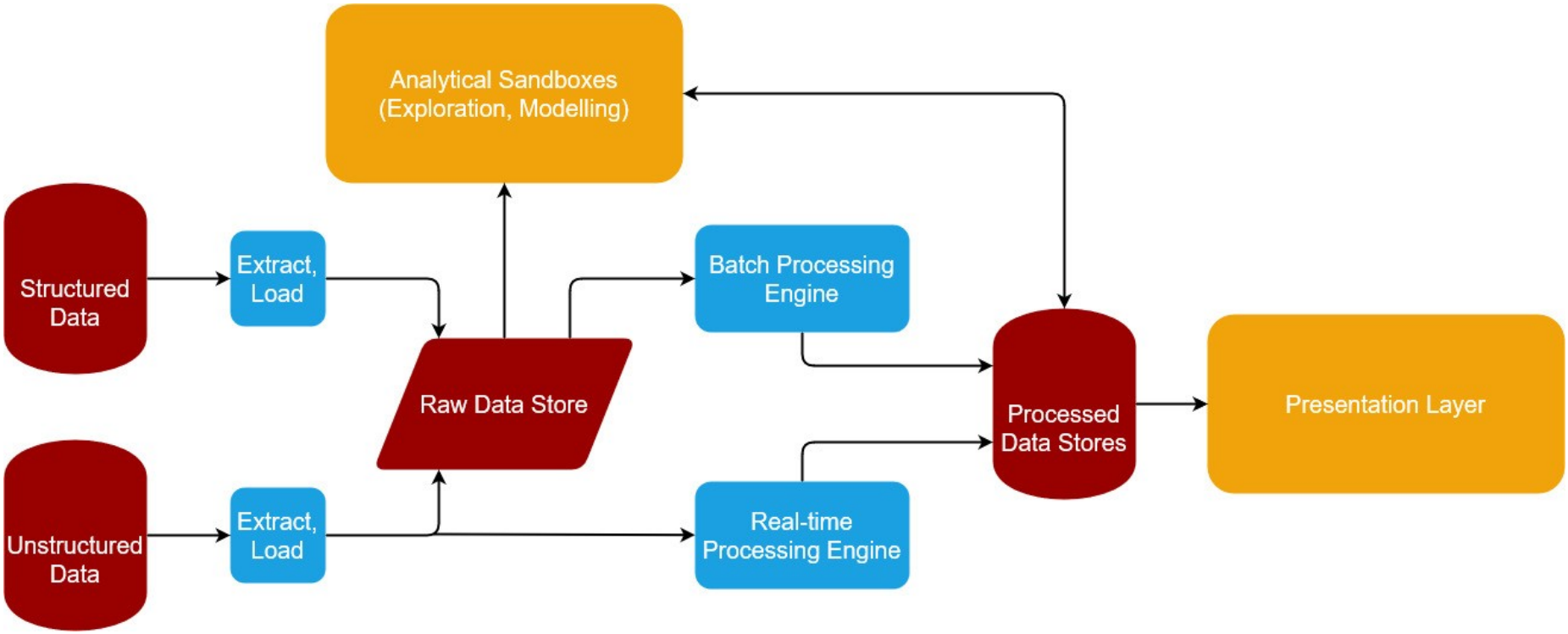
- Separate business logic and inference engine
- Install a gateway before a set of applications, services, or deployments
- In general: Separation of concerns

Architecture Patterns: Data Lake



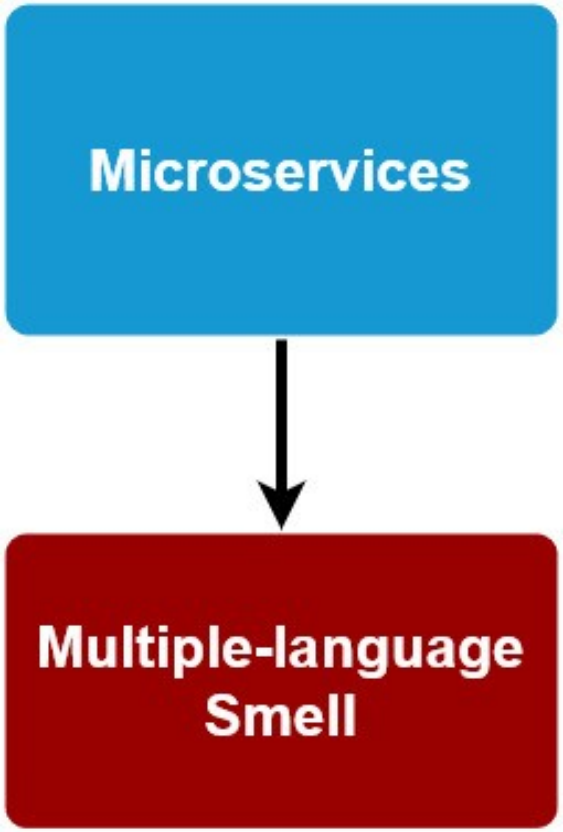
- Old architecture: **Data Warehouse** (extract data → transform → load)
- New challenges: understand the data first, make choices and compromises on which data to store and which data to discard
- Think holistically about data collection + feature extraction

Data Lake Architecture Pattern



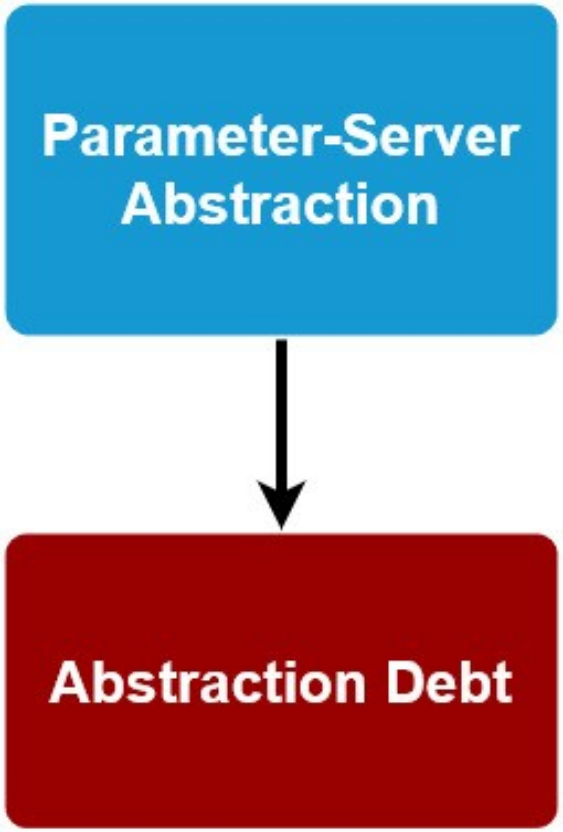
Based on Conceptual Data Lake Architecture by Pradeep Menon, 2017. **Demystifying Data Lake Architecture**. https://www.datasciencecentral.com/profiles/blogs/demystifying-data-lake-architecture, last access 01.06.20

Architecture Patterns: Microservices



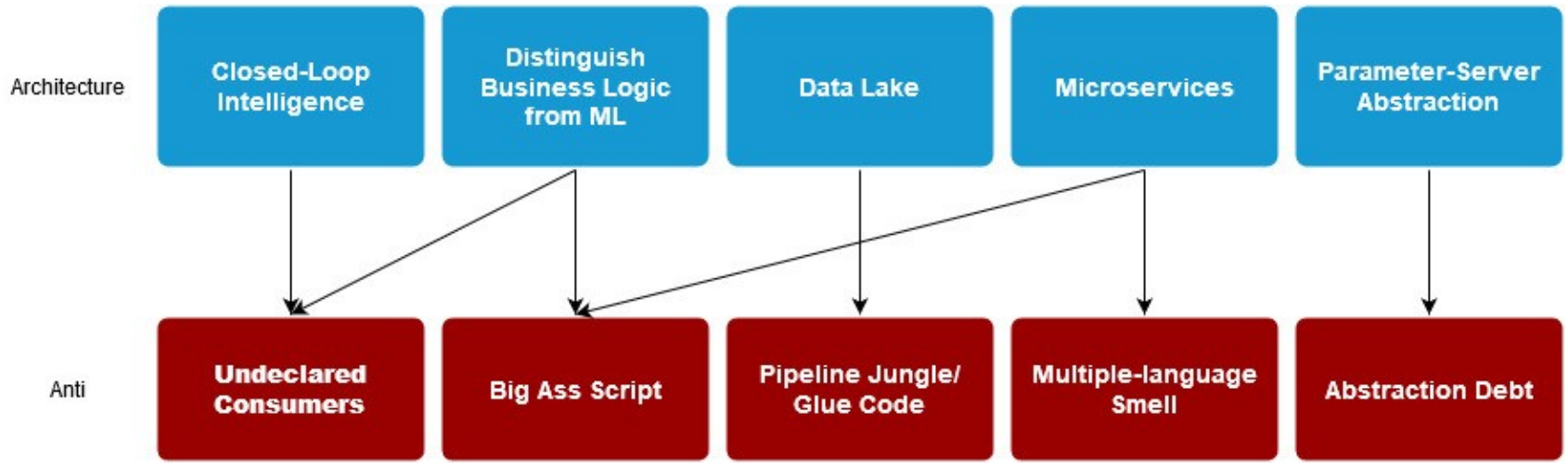
- Recombinable discrete units
- Before: push-based, manual, linear pipelines
- Now, with Microservices: pull-based, automated, on-demand content enrichment cycles

Architecture Patterns: Parameter-Server Abstraction



- Distribute data and workloads over worker nodes
- Server nodes maintain globally shared parameters (represented as vectors and matrices)
- **Federated Learning**: Device downloads the current model, improves it by learning from data, and sends update to the cloud. The update is averaged with other user updates to improve the shared model

Architecture Patterns



Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Motivation & Antipatterns

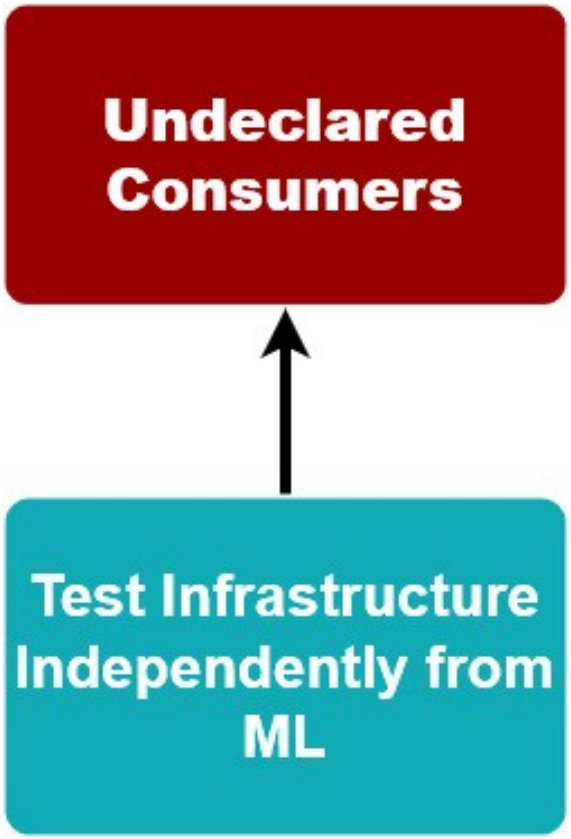
Architectural Patterns

Design Patterns

Where should AI live?

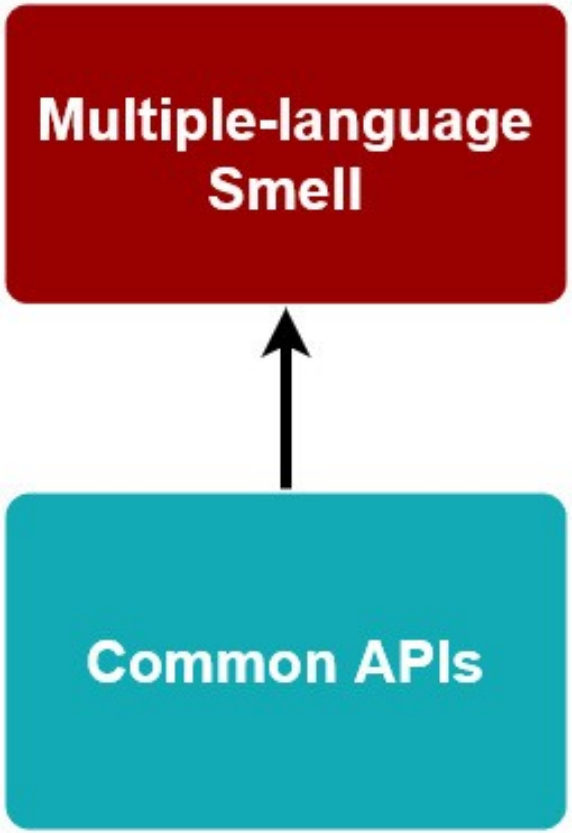
AI Platforms & Organizing Intelligence

Design Patterns: Test Infrastructure Independently from ML



- Ensure that infrastructure is testable and learning parts are encapsulated
- Check statistics in pipeline in comparison to statistics for the same data processed elsewhere
- Make sure that the model gives the same score in training and serving environment

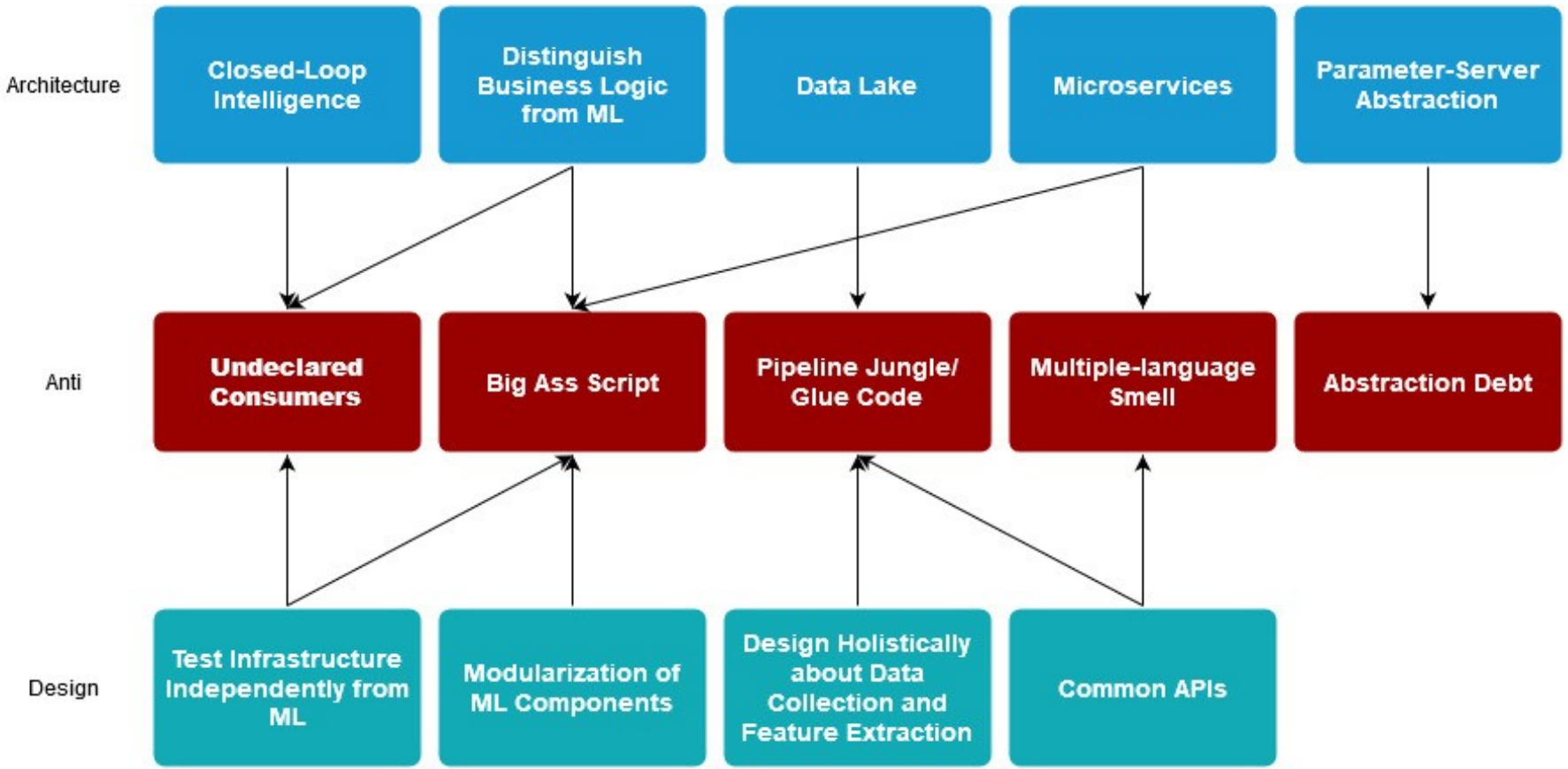
Design Patterns: Common APIs



- Using generic, independent ML frameworks results in lots of glue code
- Common APIs make supporting infrastructure more reusable and reduce the cost of changing packages

SE Architectures for ML-enabled systems: General Overview

TECHNISCHE UNIVERSITÄT DARMSTADT



Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Motivation & Antipatterns

Architectural Patterns

Design Patterns

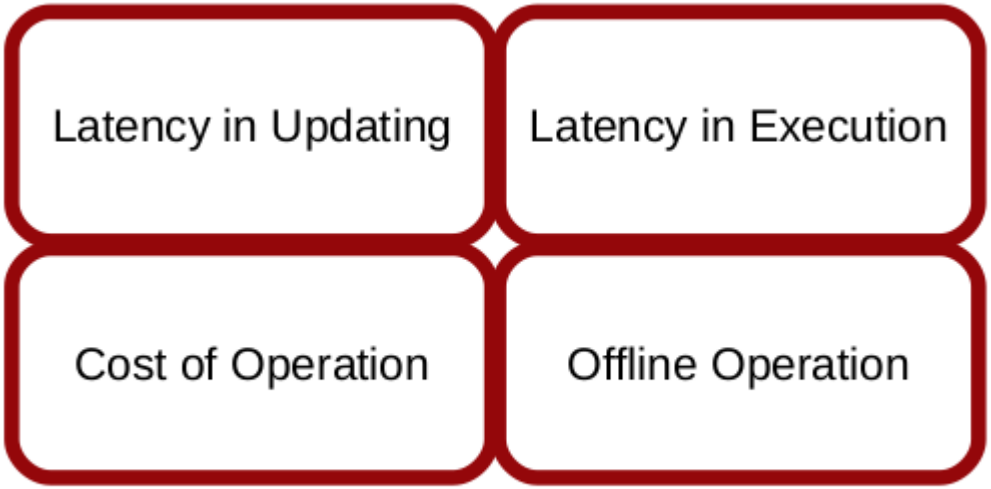
Where should AI live?

AI Platforms & Organizing Intelligence

Where should AI live?

TECHNISCHE UNIVERSITÄT DARMSTADT

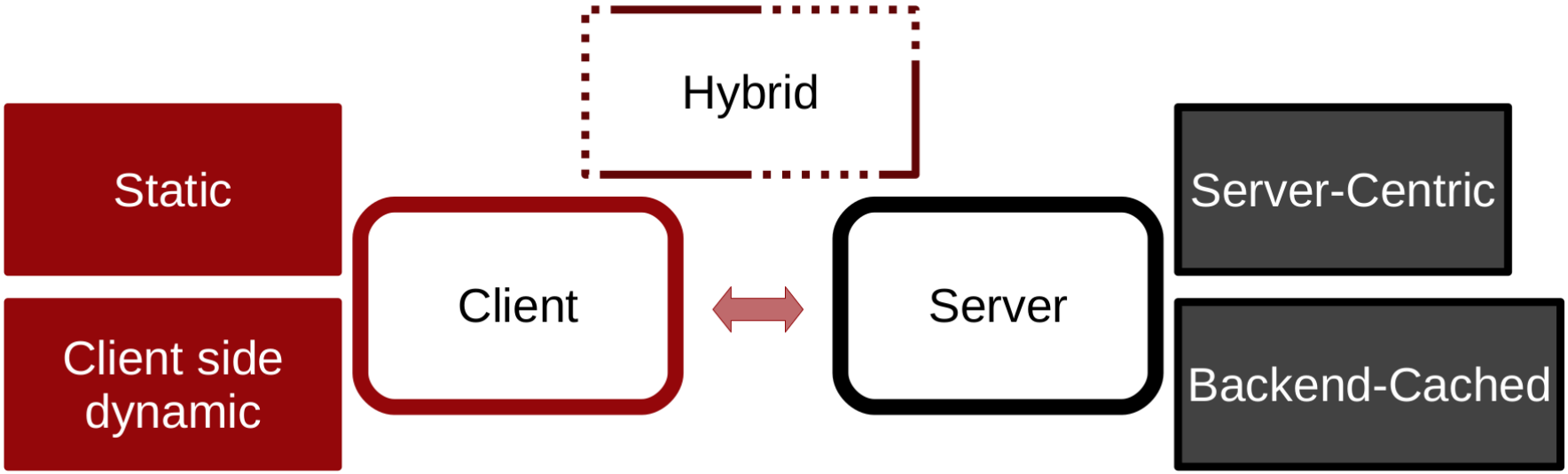
- The location of the AI can affect:



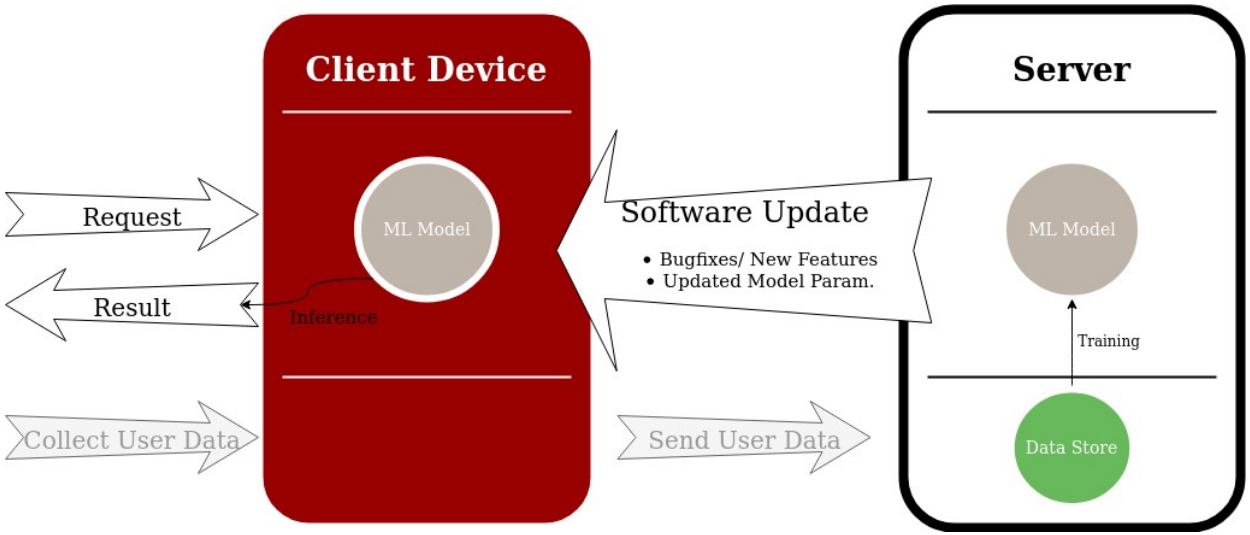
Places for AI to live

from Geoff Hulton, 2018. Building Intelligent Systems

TECHNISCHE UNIVERSITÄT DARMSTADT

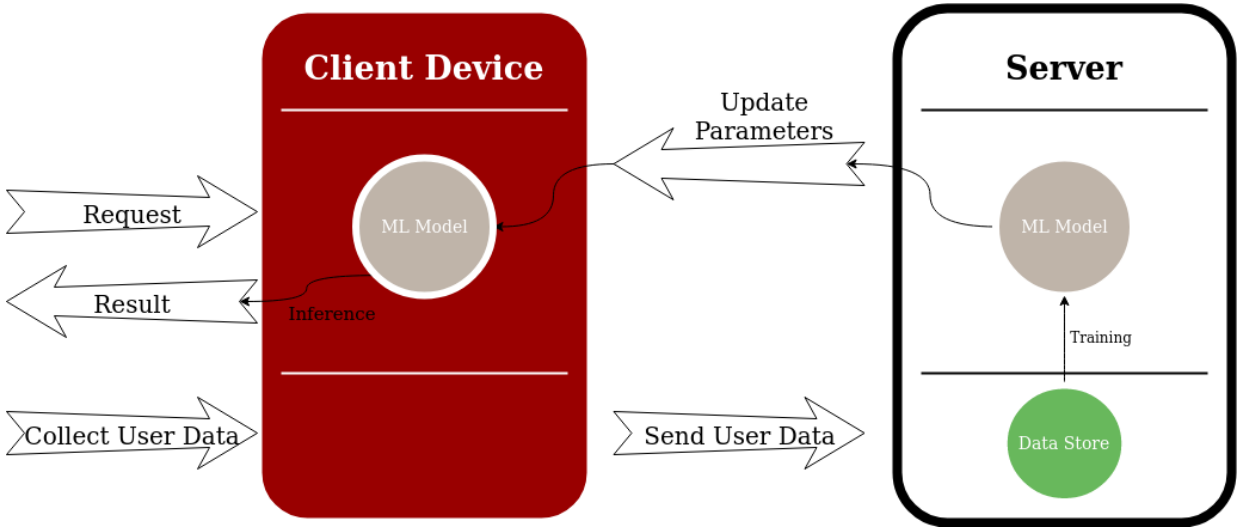


Static Intelligence on the Client



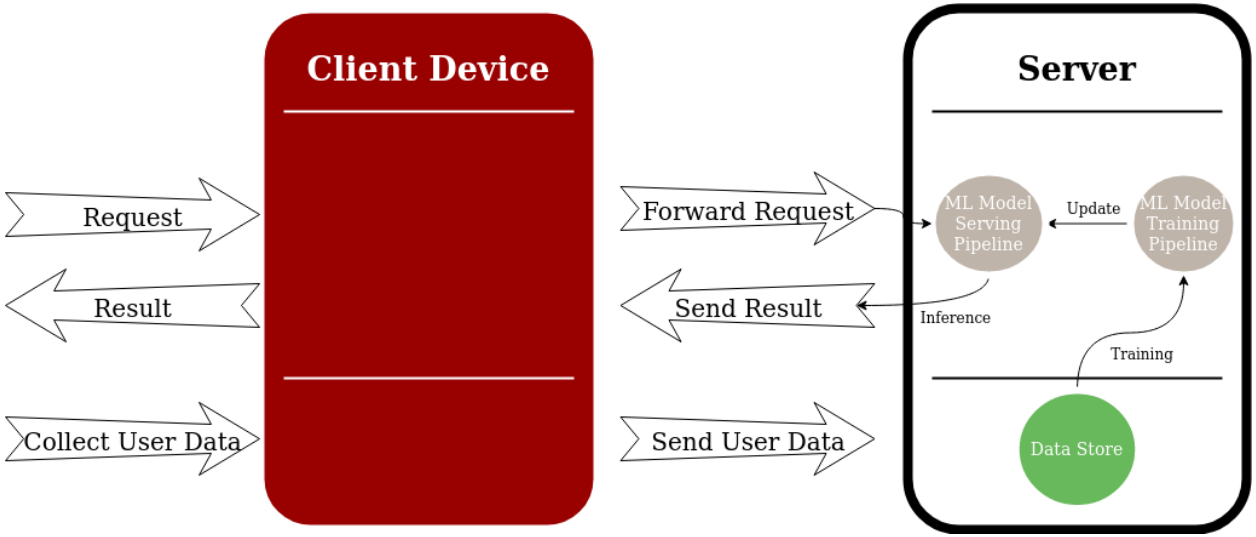
- Model updates in low frequency via Software Updates
  → Low Latency in updating
- Low latency in execution
- Cost of operation low
- Offline operation by default
- Only applicable for slowly changing problems

Client Side Dynamic



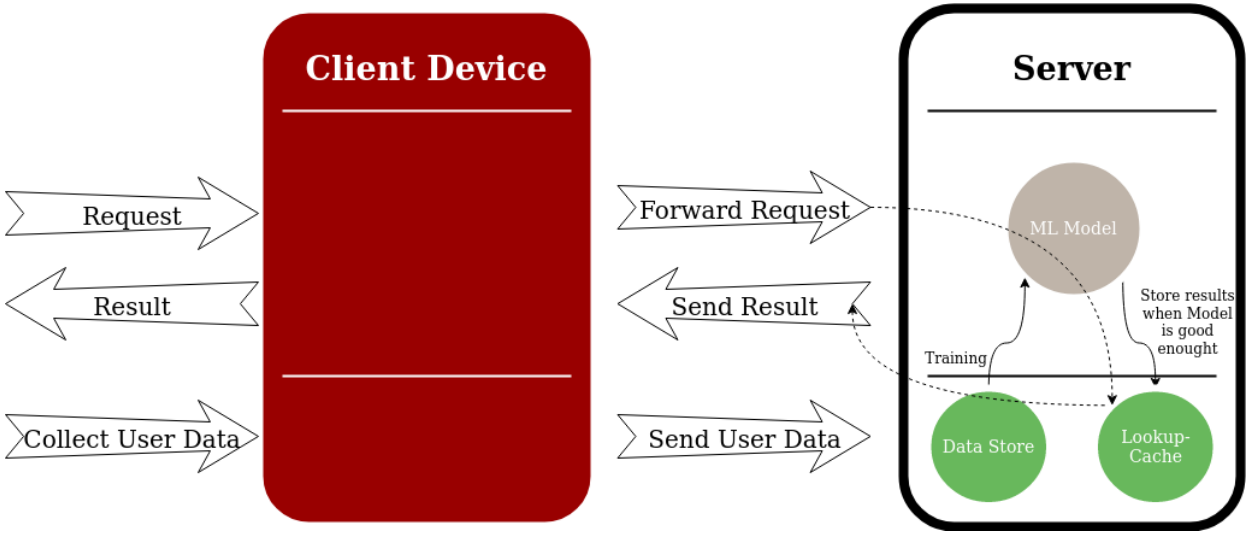
- Low latency in execution
- Potentially high latency in updating
- Cost of operation distributed between client and server
- Offline operation unproblematic for some amount of time

Server Centric



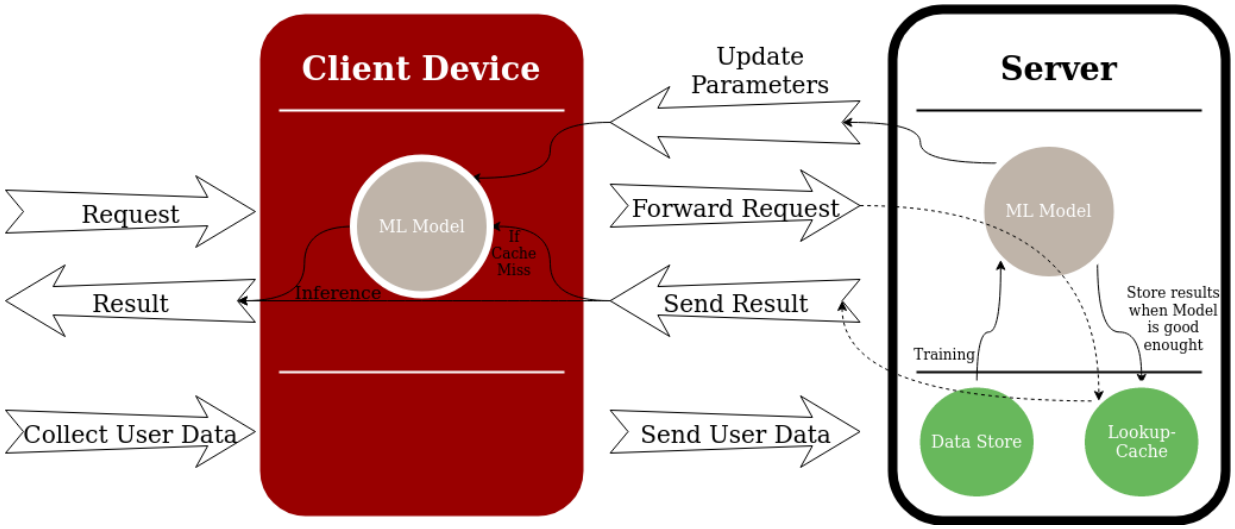
- Latency in execution increased by communication overhead
- Low latency in updating
- Cost of operation fully on server (provider)
- Offline operation requires to load the model to the client

Backend-Cached



- Low latency in execution, only lookup and request
- Low latency in updating
- Cost of operation based on usage volume
- Offline operation requires to load the model or parts of the cache to the client

Hybrid (Example)



- Store popular inputs in the cache, compute rest on client device
- Low latency in updating, but only required if server is unavailable
- Low latency in execution
- Offline operation unproblematic for some amount of time
- more complex

Outline

TECHNISCHE UNIVERSITÄT DARMSTADT

Motivation & Antipatterns

Architectural Patterns

Design Patterns

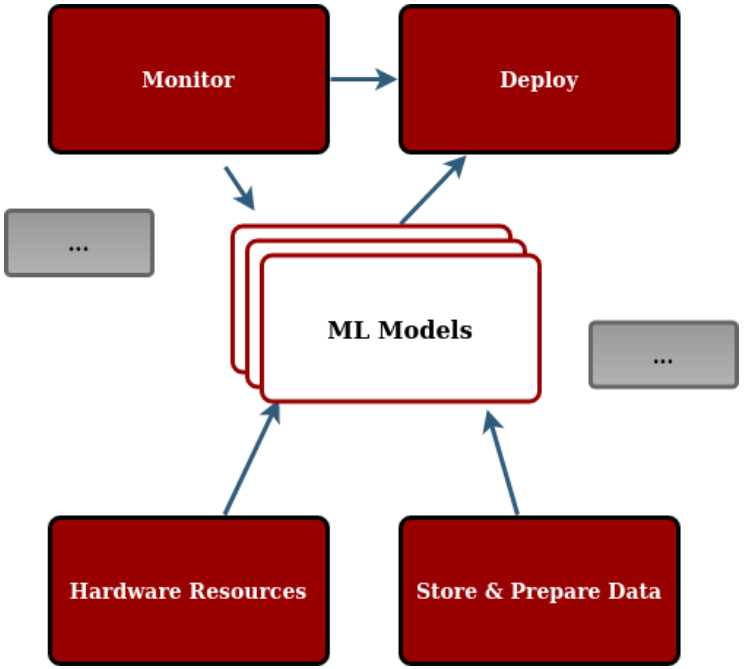
Where should AI live?

AI Platforms & Organizing Intelligence

AI Platforms

TECHNISCHE UNIVERSITÄT DARMSTADT

- Framework that automatically integrates different components of an AI system
- Often realized as microservices
- Accessible via API or GUI



Showcasing an Example
Azure Machine Learning
https://docs.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-ml

TECHNISCHE UNIVERSITÄT DARMSTADT

Create a workspace

```
ws = Workspace.create(name='TeamWorkspace',
                      subscription_id='1308a130-d549-44e1-ba66-ce8c487d76e3',
                      resource_group='amldemo',
                      location='eastus2',
                      exist_ok=True)
```

Upload data to datastore

```
ds = ws.get_default_datastore()
mnist_data = ds.upload(src_dir = './data/mnist', target_path = 'mnist', show_progress = True)
```

Allocate hardware resources

```
nc6_cluster = ComputeTarget.create(ws,
                                   name = "gpucluster",
                                   provisioning_configuration=provisioning_config)
```

Showcasing an Example 2
Azure Machine Learning
https://docs.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-ml

TECHNISCHE UNIVERSITÄT DARMSTADT

Create and submit an experiment

```
source_folder = './demo'
tf_estimator = TensorFlow(source_directory=source_folder,
                          compute_target=nc6_cluster,
                          entry_script='mnist_with_summaries.py')

run = mnist.submit(tf_estimator)
```

Monitor results

```
RunDetails(run).show()
```

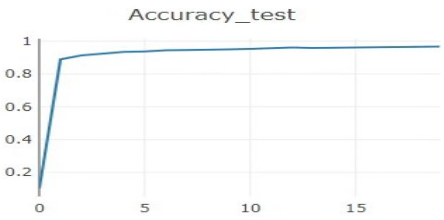
Run Properties

Status	Completed
Start Time	11/16/2018 10:39:39 AM
Duration	0:01:32
Run Id	MNIST_1542393578942
Arguments	N/A
learning_rate	0.001
dropout	0.9
Accuracy	0.9656000137329102

Output Logs

```
Streaming log file azureml-logs/60_control_log.txt
Streaming log file azureml-logs/80_driver_log.txt
Script process exited with code 0
Uploading driver log...
Finalizing run...

WARNING:tensorflow:From mnist_with_summaries.py:42: read_data_sets (from
tensorflow.contrib.learn.python.learn.datasets.mnist) is deprecated and will be removed in a
future version.
Instructions for updating:
Please use alternatives such as official/mnist/dataset.py from tensorflow/models.
WARNING:tensorflow:From /azureml-
envs/azureml_d830dfb29fc815326bc8cd6e3a484d0b/lib/python3.6/site-
packages/tensorflow/contrib/learn/python/learn/datasets/mnist.py:260: maybe_download (from
tensorflow.contrib.learn.python.learn.datasets.base) is deprecated and will be removed in a
future version.
```



Showcasing an Example 3
Azure Machine Learning
https://docs.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-ml

TECHNISCHE UNIVERSITÄT DARMSTADT

Automatic hyperparameter tuning

```
ps = RandomParameterSampling(
    {
        '--learning_rate': loguniform(-15, -3),
        '--dropout': uniform(0.5, 0.95)
    }
)

early_termination_policy = BanditPolicy(slack_factor = 0.15, evaluation_interval=2)

hyperdrive_run_config = HyperDriveRunConfig(estimator = tf_estimator,
                                            hyperparameter_sampling = ps,
                                            policy = early_termination_policy,
                                            primary_metric_name = "Accuracy_test",
                                            primary_metric_goal = PrimaryMetricGoal.MAXIMIZE,
                                            max_total_runs = 20,
                                            max_concurrent_runs = 5)

hd_run = mnist.submit(hyperdrive_run_config)
```

AutoML

```
automl_config = AutoMLConfig(task = 'classification',
                             path=source_folder,
                             compute_target = nc6_cluster,
                             data_script = source_folder + "/data.py",
                             max_time_sec = 600,
                             iterations = 20,
                             n_cross_validations = 5,
                             primary_metric = Metric.AUCWeighted,
                             concurrent_iterations = 4)

remote_run = mnist.submit(automl_config)
```

More Platforms

TECHNISCHE UNIVERSITÄT DARMSTADT

- *Amazon SageMaker*
  - Automatic Labeling via ML and Amazon Mechanical turk
  - Auto ML
- *Microsoft Azure ML*
  - Drag-n-drop interface
  - Auto ML
- *MLflow*
  - Open source platform/ no cost
  - Does not provide hardware resources
- *Google Cloud AI*
  - Auto ML
  - Many pre-trained models of Vision, Video, NLP etc.

Organization & Architecture

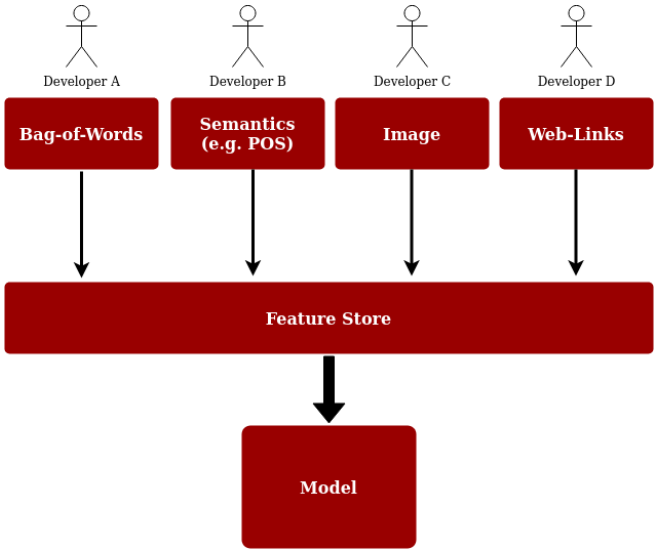
TECHNISCHE UNIVERSITÄT DARMSTADT

- Organize Intelligence in a way that supports:
  - High accuracy
  - Efficient collaboration
  - Lose coupling
  - Measureability
  - Easy Growth
- Some techniques on the following slides

Decoupled Feature Engineering

TECHNISCHE UNIVERSITÄT DARMSTADT

- Each developer works on a different part of the context to create features
  - Example: Develop features for classifying Websites
- Problem:
  - Features could be redundant



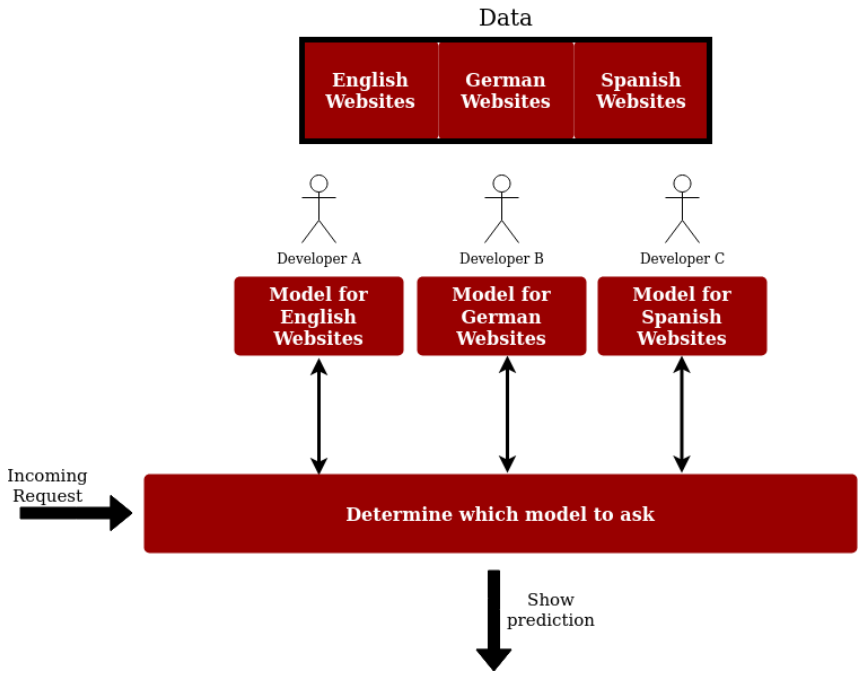
Context Partitioning

TECHNISCHE UNIVERSITÄT DARMSTADT

- Partition the data into subsets and create a model for each subset
  - Example: Classify websites with different languages

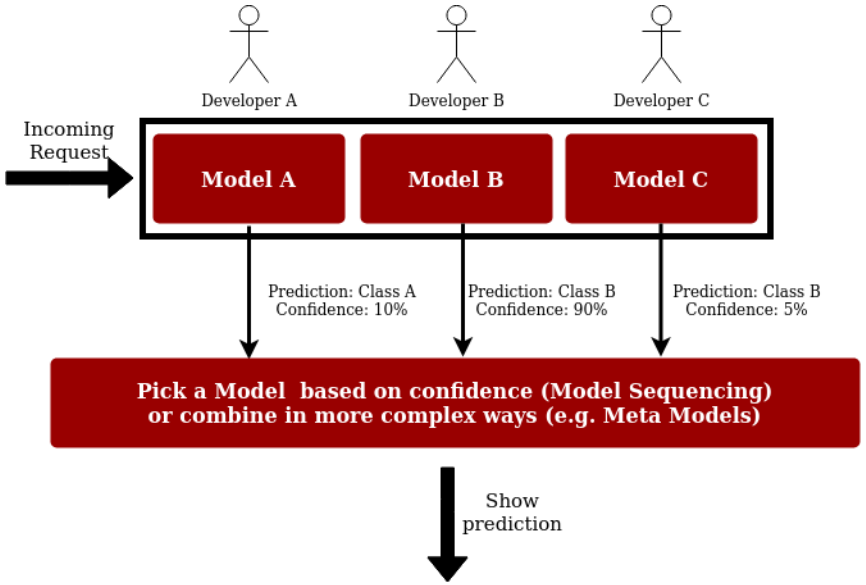
- Problems:
  - ML algorithms could instead use the partition criteria internally

    → manual partitioning might result in lower accuracy



Model Sequencing/Meta Models

- Develop different Models and pick the most confident prediction/combine their predictions
  - Each model has the chance to vote for the outcome
- Problems:
  - Meta models can have high accuracy but are harder to manage
  - Model sequencing might trade off some accuracy



Conclusion

TECHNISCHE UNIVERSITÄT DARMSTADT

- AI Systems architecture needs to consider new challenges that come with the characteristics of ML algorithms
- We learned several patters on different levels of abstraction in order to address these challenges
  - Where to put your model?
  - Architecture patters
  - Design patterns
- Currently most patterns are little concrete
- Microservices are one of the most prominent patterns in real world AI applications
- AI Platforms provide a ready-to-use architecture based on Microservices
- Some Architectural patterns support efficient organization of AI development

Questions?

Discussion

TECHNISCHE UNIVERSITÄT DARMSTADT

Trade-offs between different patterns:
e.g. Do Microservices lead to more glue code?

Discussion

TECHNISCHE UNIVERSITÄT DARMSTADT

In which cases would you put your AI on the client/server?

Discussion

TECHNISCHE UNIVERSITÄT DARMSTADT

Have you used any of those patterns in your own projects?

If no, what other patterns?

References

- D. Sculley et al. 2015. **Hidden technical debt in Machine learning systems**. https://papers.nips.cc/paper/5656-hidden-technical-debt-in-machine-learning-systems.pdf, last access 01.06.20
- H. Washizaki et al. 2019. **Studying Software Engineering Patterns for Designing Machine Learning Systems**. https://arxiv.org/abs/1910.04736, last access 26.05.20
- H. Washizaki et al. 2019. **Machine Learning Architecture and Design Patterns**. http://www.washi.cs.waseda.ac.jp/wp-content/uploads/2019/12/IEEE_Software_19__ML_Patterns.pdf, last access 01.06.20
- **Daniel Smith, 2017. Exploring Development Patterns in Data Science**. https://www.theorylane.com/2017/10/20/some-development-patterns-in-data-science/, last access 26.05.20
- Pradeep Menon, 2017. **Demystifying Data Lake Architecture**. https://www.datasciencecentral.com/profiles/blogs/demystifying-data-lake-architecture, last access 01.06.20
- B. McMahan and D. Ramage, 2017. **Federated Learning: Collaborative Machine Learning without Centralized Training Data**. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html, last access 26.05.20

References

- Mu Li et al. 2014. **Scaling Distributed Machine Learning with the Parameter Server**. https://www.cs.cmu.edu/~muli/file/parameter_server_osdi14.pdf, last access 01.06.20
- Md S. Rahman et al. 2019. **Machine Learning Software Engineering in Practice: An Industrial Case Study**. https://arxiv.org/abs/1906.07154, last access 01.06.20
- M. Zinkevich, 2019. **Rules of Machine Learning: Best Practices for ML Engineering**. https://developers.google.com/machine-learning/guides/rules-of-ml/#your_first_objective, last access 01.06.20
- Microsoft, 2019. **What is Azure Machine Learning?** https://docs.microsoft.com/en-us/azure/machine-learning/overview-what-is-azure-ml
- Geoff Hulton, 2018. Building Intelligent Systems: A Guide to Machine Learning Engineering Chapter 13 & Chapter 21
- Rahman et al., 2019. Machine Learning Software Engineering in Practice: An Industrial Case Study

Acknowledgments & License

- All Images are by the authors of these slides
- These slides are made available by the authors (Marcel de Boer, Zlatko Kolev) under CC BY 4.0