A Query-Based Feature Store at OLX

OLX Brasil
Biggest C2C Marketplace in Brasil
6M Daily Active Users

5M Chat Messages Per Day

500K New Ads Per Day
Problems
How to minimize scam impact?
How to minimize scam impact?

We use Machine Learning!!
Chat Moderation → User Moderation → Ad Moderation → create → ML Model
Chat Moderation
Chat Moderation

Need to ban users ASAP, in minutes, based on chat behavior
Features → Online Model → Predictions
They needed a Transformation Engine to serve features in Real Time
Raw Data → Online Model → Predictions
Imagine if each model in production should replicate this infra?
Who you gonna call?
Who you gonna call?
The MLOps Platforms Team
This would be the start of our Feature Store
This would be the start of our Feature Store
FS Conception
Platform that centralizes processment, offers easy to create features and promotes reusability.
OLX Real time Data Sources
- Backbones/Queues
- Applications
- Database listener

Feature Store

Real time ML models
End user applications
OLX Real time Data Sources
- Backbones/Queues
- Applications
- Database listener

Data input interface
(Push) Topics

Transformer Engine
Runs user-registered transformers on data

Transformers
Transformer's state DB

Feature Store

Real time ML models
End user applications
FS Implementation
Main Concerns:

● Must be easy to create features
● Pricing
● Infrastructure Complexity
Messaging System

Kafka Managed by AWS
Integration with external components

Kafka Connect
What about the transformations?
Raw Data → Aggregations
Discovery with some Transformation Engines
<table>
<thead>
<tr>
<th>Feature</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Language</td>
<td>Java/SQL</td>
</tr>
<tr>
<td>Maturity</td>
<td>Good</td>
</tr>
<tr>
<td>Documentation</td>
<td>Bad</td>
</tr>
<tr>
<td>Feature Development</td>
<td>Regular</td>
</tr>
<tr>
<td>Beginner-friendly</td>
<td>Bad</td>
</tr>
<tr>
<td>Scalability</td>
<td>Good</td>
</tr>
<tr>
<td>Cost</td>
<td>Bad</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Good</td>
</tr>
<tr>
<td>Feature</td>
<td>Flink</td>
</tr>
<tr>
<td>--------------</td>
<td>-------</td>
</tr>
<tr>
<td>Language</td>
<td>Java/SQL</td>
</tr>
<tr>
<td>Maturity</td>
<td>Good</td>
</tr>
<tr>
<td>Documentation</td>
<td>Bad</td>
</tr>
<tr>
<td>Feature Development</td>
<td>Regular</td>
</tr>
<tr>
<td>Beginner-friendly</td>
<td>Bad</td>
</tr>
<tr>
<td>Scalability</td>
<td>Good</td>
</tr>
<tr>
<td>Cost</td>
<td>Bad</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Good</td>
</tr>
<tr>
<td></td>
<td>Flink</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------------</td>
</tr>
<tr>
<td>Language</td>
<td>Java/SQL</td>
</tr>
<tr>
<td>Maturity</td>
<td>Good</td>
</tr>
<tr>
<td>Documentation</td>
<td>Bad</td>
</tr>
<tr>
<td>Feature Development</td>
<td>Regular</td>
</tr>
<tr>
<td>Beginner-friendly</td>
<td>Bad</td>
</tr>
<tr>
<td>Scalability</td>
<td>Good</td>
</tr>
<tr>
<td>Cost</td>
<td>Bad</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Good</td>
</tr>
<tr>
<td></td>
<td>Flink</td>
</tr>
<tr>
<td>----------------------</td>
<td>----------</td>
</tr>
<tr>
<td>Language</td>
<td>Java/SQL</td>
</tr>
<tr>
<td>Maturity</td>
<td>Good</td>
</tr>
<tr>
<td>Documentation</td>
<td>Bad</td>
</tr>
<tr>
<td>Feature Development</td>
<td>Regular</td>
</tr>
<tr>
<td>Beginner-friendly</td>
<td>Bad</td>
</tr>
<tr>
<td>Scalability</td>
<td>Good</td>
</tr>
<tr>
<td>Cost</td>
<td>Bad</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Good</td>
</tr>
<tr>
<td></td>
<td>Flink</td>
</tr>
<tr>
<td>----------</td>
<td>-------</td>
</tr>
<tr>
<td>Language</td>
<td>Java/SQL</td>
</tr>
<tr>
<td>Maturity</td>
<td>Good</td>
</tr>
<tr>
<td>Documentation</td>
<td>Bad</td>
</tr>
<tr>
<td>Feature Development</td>
<td>Regular</td>
</tr>
<tr>
<td>Beginner-friendly</td>
<td>Bad</td>
</tr>
<tr>
<td>Scalability</td>
<td>Good</td>
</tr>
<tr>
<td>Cost</td>
<td>Bad</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Good</td>
</tr>
<tr>
<td>Feature</td>
<td>KSQLdb</td>
</tr>
<tr>
<td>-----------------</td>
<td>--------</td>
</tr>
<tr>
<td>Language</td>
<td>Good</td>
</tr>
<tr>
<td>Maturity</td>
<td>Good</td>
</tr>
<tr>
<td>Documentation</td>
<td>Good</td>
</tr>
<tr>
<td>Feature Development</td>
<td>Good</td>
</tr>
<tr>
<td>Beginner-friendly</td>
<td>Good</td>
</tr>
<tr>
<td>Scalability</td>
<td>Good</td>
</tr>
<tr>
<td>Cost</td>
<td>Regular</td>
</tr>
<tr>
<td>Flexibility</td>
<td>Regular</td>
</tr>
</tbody>
</table>
CREATE OR REPLACE STREAM test_stateless (  
adId VARCHAR,  
chatId VARCHAR) WITH (  
KAFKA_TOPIC='test_topic',  
VALUE_FORMAT='JSON',  
PARTITIONS=1)  

CREATE OR REPLACE TABLE test_stateful as  
SELECT adId, COUNT(*)  
FROM test_stateless  
GROUP BY adId  
EMIT CHANGES
Example of usage
We can create a stream to get the Raw Data

```
CREATE STREAM chatmod_message ( --noqa
    userIp VARCHAR,
    tipVisibleTo VARCHAR,
    tipValue VARCHAR,
    tipType VARCHAR,
    textMessage VARCHAR,
    senderAccountId VARCHAR,
    receiverAccountId VARCHAR,
    publicId VARCHAR,
    platform VARCHAR,
    messageType VARCHAR,
    messageTimestamp VARCHAR,
    messageId VARCHAR,
    listId VARCHAR,
    chatId VARCHAR,
    adId VARCHAR,
    senderId VARCHAR,
    senderType VARCHAR,
    isFirstMessage BOOLEAN
) WITH ( 
    KAFKA_TOPIC = 'chatmod_message',
    VALUE_FORMAT = 'json',
    PARTITIONS = 1,
    TIMESTAMP = 'messageTimestamp',
    TIMESTAMP_FORMAT = 'yyyy-MM-dd HH:mm:ss.SSS'
);
```
We can create a TABLE that aggregates data
CREATE TABLE chatmod_first_message_similarity_feature WITH (KAFKA_TOPIC = 'chatmod_first_message_similarity_feature') AS

SELECT
    buyer_id AS buyer_id_key,
    AS_VALUE(buyer_id) AS buyer_id,
    MIN_LEVENSHTEIN_DISTANCE(textMessage) AS first_message_similarity
FROM chatmod_buyer_message WINDOW SESSION (2 HOURS)
WHERE isFirstMessage = true
GROUP BY buyer_id Emit CHANGES;
We can generate a stream that reads from this Table and saves into a topic

```sql
CREATE OR REPLACE STREAM chatmod_first_message_similarity_feature_windowed ( --noqa
  buyer_id VARCHAR,
  buyer_id_key VARCHAR KEY,
  first_message_similarity INTEGER
) WITH (
  KAFKA_TOPIC='chatmod_first_message_similarity_feature',
  VALUE_FORMAT='JSON',
  WINDOW_TYPE='SESSION'
);
```
Raw Data Stream

Min Levenshtein distance in chats per user

Mean of number of characters in messages per user

Join of all features

Number of chats per user

Features
Declaring Features
product:
  name: test_name_6
  team: test_team
  description: This is the product description

connectors:
  connector_1:
    name: source_connector
    connector_type: source
    topic_name: topic_source_connector
  connector_2:
    name: sink_connector
    connector_type: sink
    topic_name: topic_sink_connector

features:
  feature_1:
    name: first_feature_6
    description: This is my first feature
    feature_streams:
      - stream_1
      - stream_2
    sql_file: example_sql_scripts/file_1.sql
    input_topic: topic_1
    output_topic: topic_2
    is_external: True
  feature_2:
    name: second_feature_6
    description: This is my second feature
    feature_streams:
      - stream_3
      - stream_4
    sql_file: example_sql_scripts/file_2.sql
    input_topic: topic_2
    output_topic: topic_3
    is_external: True
    dependencies:
      - feature_1
Feature Declaration Interface

`feature_store create-features example_config.yml`
Results
Messages Input per day
5 M

Features
8 created

Feature Output per day
1.5 M

Models using the Feature Store
2 models
Fraudulent messages detected per day: 300-400

Complaint reduced by: 22-26%
Roadmap
Creation of features in batch, from datalake
Feature Versioning
Feature Governance

Monitoring

Storage   Serving

Registry
Export features into Datalake, for training
Monitoring

Storage | Serving

Registry

Feature serving with database and API
Thank you!

Do you have any questions?

Augusto Acioli Pinho Vanderley - OLX Brasil

https://www.linkedin.com/in/augusto-vanderley/?locale=en_US