Which car fits my life?

Mobile.de’s approach to recommendations

PyData Berlin, July 1st, 2017
Florian Wilhelm, Arnab Dutta
Introduction

Dr. Arnab Dutta
Data Scientist
mobile.de GmbH

@kopfhohen
kraktos

Dr. Florian Wilhelm
Data Scientist
inovex GmbH

@FlorianWilhelm
FlorianWilhelm
florianwilhelm.info
Outline

- Introduction
- Use-cases
- Theory
- Our Approach
- Example
- Outlook
Part of ebay Tech

MOBILE.DE
GERMAN MARKET LEADER

13.5 MIO
UNIQUE USER
PER MONTH

1.6 MIO
VEHICLES

290
EMPLOYEES

DREILINDEN / FRIEDRICHSHAIN BERLIN
HEADQUARTERS
IT-project house for digital transformation:

- Agile Development & Management
- Web · UI/UX · Replatforming · Microservices
- Mobile · Apps · Smart Devices · Robotics
- Big Data & Business Intelligence Platforms
- Data Science · Data Products · Search · Deep Learning
- Data Center Automation · DevOps · Cloud · Hosting
- Trainings & Coachings

Using technology to inspire our clients. And ourselves.

inovex offices in
Karlsruhe · Pforzheim · München · Köln · Hamburg.

www.inovex.de
Outline

- Introduction
- Use-cases
- Theory
- Our Approach
- Example
- Outlook
Why Recommendations?

Show width of offering

Inspiration

Engagement
Why Recommendations?

User Benefits
- engagement
- inspiration
- relevance

Business Benefits
- high click-through-rate
- small exit- & bounce-rates
Mobile.de Conversion Funnel
Recommendations on Home

Recommendations based on preferences of visiting users as an alternative entry point.
Recommendations on Search Results Page

Recommendations based on similar vehicle make and model id to present alternatives.
**Recommendations on View Item Page**

Recommendations based on the specific make and model a user is viewing to present alternatives.
Recommendations on your Wishlist

Recommendations based on the specific **make and model** of a **deleted ad** to provide almost identical recommendations.

Recommendations based on the **users car preferences** and the **parking lot items**.
§ Introduction
§ Use-cases
§ Theory
  § Collaborative Filtering
  § Content Based
§ Our Approach
§ Example
§ Outlook
Collaborative Filtering
Item-based Recommendations

Cosine Similarity

\[
\cos \theta = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}
\]

- Watched / Rated
- Unwatched

Item-Item Similarity
Item-based Recommendations

Wishlist

Recommendation
Summary of Collaborative Filtering

✓ Collective behaviour of users

✓ Standard-Method (it works, it’s reliable etc.)

**Cold Start Problem:** New listings need a certain number of clicks to be recommended.

**Sparsity problems:** Lot fewer interaction data points than total items and users.

Content agnostic

Only “batch-based” learning
### User Preferences

- **Looking For:** Used Car (100%)
- **Prefers (Make):** BMW (50%), Audi (50%)
- **Prefers (Model):** Audi A3 (25%), Audi A4 (25%), BMW 318 (50%)
- **Searching In:** lat 52.5206, lon 13.409
- **Search Radius:** 300km
- **Preferred Price:** 20 000€ ± 1500€
- **Preferred Mileage:** 10 000km ± 5000km
Content-based Filtering

- **<Price: 10K, Category: small>** interacted
- **<Price: 6K, Category: small>** similar
- **<Price: 10K, Category: small>** recommend
- **<Price: 90K, Category: sports>** less similar
Summary of Content-based

✓ Works even if there are no other users

✓ content-based preferences of users based on a weighted vector of item features

    Hard to do recommendations for new users (cold start problem)

    Non-applicable for heterogenous content types

    Low diversity, i.e. more of the same
Introduction
Use-cases
Theory
Our Approach
Example
Outlook
Find the car that perfectly fits your life

User’s Car Preferences

Car Pool + Attributes
(make, model, color, price, ...)

Interactions of other users
(views, parkings, contacts)

Flexible
(cold-start, uncertainty, real-time, ...)

mobile.de
Hybrid Recommender

- no cold-start problem of new items
- integrate new user events in real-time
- robust and reliable concepts
- easy to tune for different use-cases
- comprehensible and debuggable
User Preference+ Recommendation Architecture

- All Listings
- Recommendation Engine
- User Preference Computation & Storage
- User Event Tracking & Storage
- Recommendation
- User Preference
- Service API
Hybrid Recommender Concept

Looking For: Used Car (100%)
Prefers (Make): BMW (50%), Audi (50%)
Prefers (Model): Audi A3 (25%), Audi A4 (25%), BMW 318 (50%)

Searching In: lat 52.5206, lon 13.409
Search Radius: 300km
Preferred Price: 20 000€ ± 1500€
Preferred Mileage: 10 000km ± 5000km

User Profile
Buyer
Last Action: Yesterday
Frequent User
User 12345
Likelihood to buy: 88 %

Elastic Search Query

<table>
<thead>
<tr>
<th>Score</th>
<th>Elastic Search Query</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.8</td>
<td>6.402 € (3.469 MVA)</td>
</tr>
<tr>
<td>0.3</td>
<td>6.008 €</td>
</tr>
<tr>
<td>1.7</td>
<td>16.000 €</td>
</tr>
<tr>
<td>3.2</td>
<td>12.000 €</td>
</tr>
<tr>
<td>1.1</td>
<td>3.499 €</td>
</tr>
<tr>
<td>0.9</td>
<td>6.999 €</td>
</tr>
</tbody>
</table>

...
- Introduction
- Use-cases
- Theory
- Our Approach
- Example
- Outlook
Finding similar make/models

- **Users** are often **uncertain** with their choices in orientation phase

- Help users **explore similar** models to make informed decisions

- Exploit the **collaborative** aspect in defining the concept of similarity
Make/Model Recommender with LightFM

LightFM:
- Matured and well documented Python package
- Optimized and parallelized with Cython
- Hybrid recommender based on matrix factorisation
- Supports Learning-to-Rank objectives (BPR, WARP)
Non-negative Matrix Factorisation (NMF)

\[ M (|U| \times |I|) = L (|U| \times |LF|) \times R (|LF| \times |I|) \]
**NMF: Embeddings to Similarities**

<table>
<thead>
<tr>
<th>LF₁</th>
<th>LF₂</th>
</tr>
</thead>
<tbody>
<tr>
<td>![Car image]</td>
<td>![Car image]</td>
</tr>
</tbody>
</table>

- car is represented as an **item embedding**
- Given 2 embeddings, \( LF_{item_i} \) and \( LF_{item_j} \)
- Compute \( \text{sim}(LF_{item_i}, LF_{item_j}) \)
- Find pairwise values for all item pairs (|I| x |I|)
NMF: Persisting Item similarities

```python
def model_fitting(matrix, loss, no_comp, epochs, num_threads, item_alpha):
    ......

Args:
    matrix: sparse matrix
    loss: loss function to be employed
    no_comp: dimension of the embeddings
    epochs: #passes over the data
    num_threads: parallel threads to run
    item_alpha: L2 regularization

Returns:
    fitted Matrix factorization (MF) model
    ......

model = LightFM(loss=loss,
                item_alpha=item_alpha,
                no_components=no_comp)

return model.fit(matrix, epochs=epochs,
                  num_threads=num_threads)
```

```json
"_id" : ObjectId("588f1a8ca2418eb98367fb8"),
"itemName" : "bmw-645",
"makeId" : 3500,
"makeModelId" : "3500-31",
"topKResults" : [
    {
        "itemName" : "audi-a4",
        "makeId" : 1900,
        "makeModelId" : "1900-9",
        "score" : 0.67208,
        "modelId" : 9
    },
    {
        "itemName" : "mercedes-benz-cla 180",
        "makeId" : 17200,
        "makeModelId" : "17200-225",
        "score" : 0.65011,
        "modelId" : 225
    }
]
```
Method
- best of both the worlds
- robust

Business
- higher CTR
- lesser exits rates

User
- engagement
- diversity
- Introduction
- Motivation & Use-cases
- Theory
- Implementation
- Outlook
### Deep Learning

#### Recent Breakthroughs in Deep Learning

**Dermatologist-level classification of skin cancer with deep neural networks**
(Nature 542, 115-118, February 2017)

**DeepStack: Expert-level artificial intelligence in heads-up no-limit poker**
(Science, March 2017)

#### Reasons for Deep Learning

- captures nonlinear relations
- holistic approach, i.e. reduces number of components possibly
- less feature engineering
- possibly improved quality
Approach: Wide and Deep Model

Probability that user X likes vehicle Y
Recommendations support users to find the perfect vehicle based on their preferences and by collaboration.
Any questions?