Machine Learning with Python

Knowledge Transfer Partnership between University of West of England (UWE) and Paxport

by Pedro Ferreira

April 28, 2017
Outline

- Case Study
- Approach
- Implementation
- Results
Case Study

• Bring Artificial Intelligence to Paxport
  – Travel industry
    • Back-end service for searches and bookings of flights and accommodations
  – 3 years of stored bookings data
  – Improve holiday searches relevance/performance
Case Study

• **Challenges**
  - Scale, millions of daily searches
  - Seasonality, preferences change overtime
  - No user tracking

• **Main Tools**
  - Framework - Python (3.5.1) with Jupyter (4.0.6)
  - Data manipulation - Pandas (0.17.1)
  - Machine Learning resources - Scikit learn (0.16.1)
  - Supporting - Numpy (1.11), Scipy (0.16.0)
Approach

- Collaborative Filtering
  - Data organized in a User, Item, Preference matrix
  - Preference can be either *explicit* or *implicit*
  - Predict using the majority of similar users preferences for that particular item
Approach

- **Advantages**
  - Does not need extra data other than preferences to be effective
  - Very scalable (Matrix Factorization)

- **Disadvantages**
  - Needs a good amount of data as a starting point
  - Requires at least one observation for any given user/item before being able to make a prediction (*cold-start* problem)
Approach – Key Aspects

- "Super user" representation that utilizes search details as a way to group users (party info, dates, etc.)
  - i.e. 2 adults with no children for less than 3 days on a weekend (romantic trip?)
- Usage of implicit data (bookings)
- Matrix Factorization as the base algorithm (iALS *)
- Evaluation done by ranking searches from 2015-2016 in a weekly window and verifying the % of times the selected booking was in the Top 5 results provided

* http://yifanhu.net/PUB/cf.pdf
Implementation

• **Data overview**
  - Over 99.80% sparsity (preference matrix)

• **Model overview (iALS)**
  - Represents implicit feedback as *observations* and *confidence*
    - Confidence adapted to make the model robust to seasonality
  - Ranking obtained by multiplying the resulting Latent Factors
Implementation

- **Performance**
  - Python vs Cython (11 minutes and 45 seconds vs 7.65 seconds) build time per model
  - Sparse matrix representation vs $83705 \times 17508 \times 64$ full memory footprint
  - Re run model and evaluate rankings for over 100 weeks
    - Pandas dataframes key for easy data manipulation
## Results

### Overall performance highlighting

<table>
<thead>
<tr>
<th>Model</th>
<th>2015</th>
<th>2016</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>First Half</td>
<td>Second Half</td>
</tr>
<tr>
<td></td>
<td>Top1 %</td>
<td>Top5 %</td>
</tr>
<tr>
<td>SU1_Base</td>
<td>14.197</td>
<td>42.874</td>
</tr>
<tr>
<td>SU1_TFIDF.Temporal</td>
<td>14.659</td>
<td>42.753</td>
</tr>
<tr>
<td>SU1_BM25</td>
<td>14.466</td>
<td>42.516</td>
</tr>
<tr>
<td>SU1_BM25.Temporal</td>
<td>15.425</td>
<td>44.508</td>
</tr>
<tr>
<td>SU2_Base</td>
<td>14.310</td>
<td>42.697</td>
</tr>
<tr>
<td>SU2_Base.Temporal</td>
<td>15.091</td>
<td>44.352</td>
</tr>
<tr>
<td>SU2_BM25.Temporal</td>
<td>15.436</td>
<td>44.754</td>
</tr>
</tbody>
</table>
Results

Performance by regions (countries)
Results

- Proof of Concept deployed on a Virtual Machine
  - Single 2.20 GHz cpu
  - 4Gb ram
  - Hosted in France
  - 10,000 requests over 15 threads (83 seconds total)
Takeaway

• Global model
• Necessity for adaptability
  – Use of super users
  – Seasonality
• Notebooks are great for exploration
• Pandas is awesome!
Questions

Pedro Ferreira
Ped.j.ferreira@gmail.com

Chris Simons
Chris.Simons@uwe.ac.uk