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

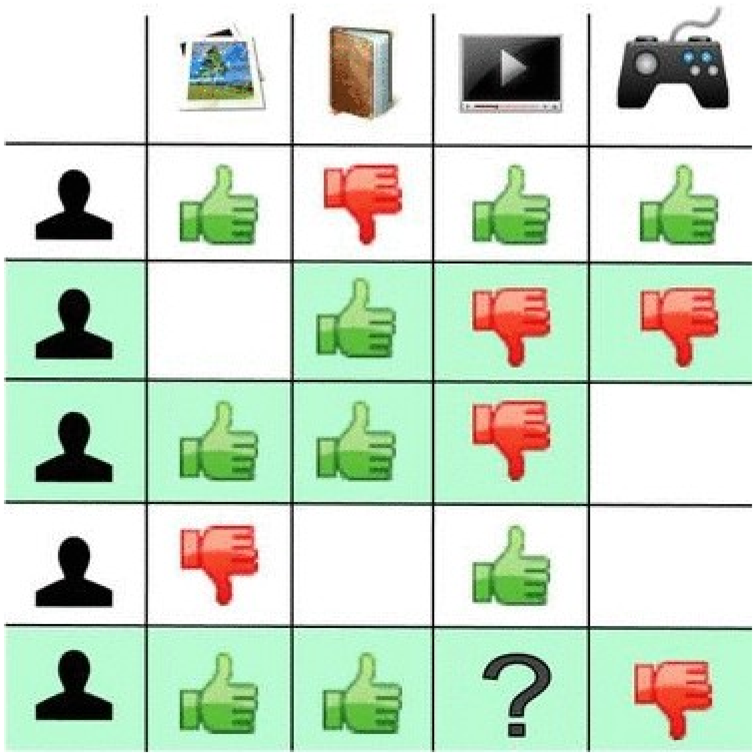
- **Bring Artificial Intelligence to Paxport**
  - Travel industry
    - Back-end service for searches and bookings of flights and accommodations
  - 3 years of stored bookings data
  - Improve holiday searches relevance/performance

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**
  - 840,030 bookings (2014-2016), 371,540 searches (2015-2016)
  - 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 83705x17508x64 full memory footprint
  - Re run model and evaluate rankings for over 100 weeks
    - Pandas dataframes key for easy data manipulation

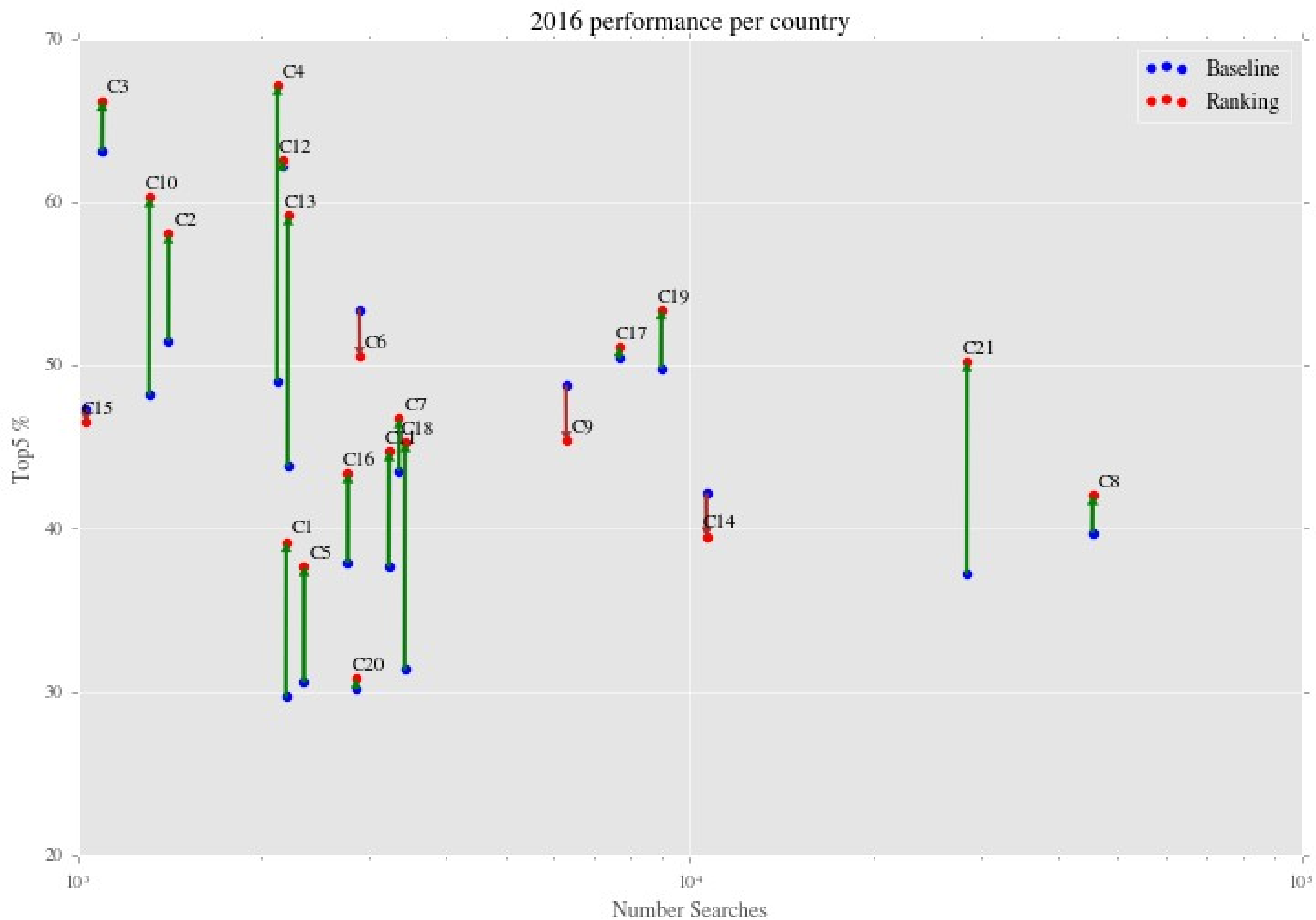
Results

Overall performance highlighting

	2015				2016			
	First Half		Second Half		First Half		Second Half	
Model	Top1 %	Top5 %	Top1 %	Top5 %	Top1 %	Top5 %	Top1 %	Top5 %
Baseline	7.947	26.935	11.261	33.219	14.759	40.387	16.377	44.581
SU1_Base	14.197	42.874	15.458	44.312	14.527	43.376	14.400	43.055
SU1_Base_Temporal	14.860	43.506	16.302	46.341	15.866	46.337	16.919	48.215
SU1_TFIDF_Temporal	14.659	42.753	15.912	45.447	15.351	44.365	16.121	45.835
SU1_BM25	14.466	42.516	15.352	44.879	14.473	42.958	15.103	44.211
SU1_BM25_Temporal	15.425	44.508	**16.564**	**47.129**	15.681	45.669	16.412	47.477
SU2_Base	14.310	42.697	15.375	44.667	14.650	43.602	14.478	43.394
SU2_Base_Temporal	15.091	44.352	16.401	46.939	**15.899**	**46.395**	**16.920**	**48.243**
SU2_TFIDF_Temporal	14.491	43.094	14.622	44.563	14.285	42.821	14.578	43.888
SU2_BM25	13.434	41.368	13.471	42.427	14.426	41.112	13.336	41.296
SU2_BM25_Temporal	**15.436**	**44.754**	16.046	46.499	15.067	45.074	15.774	47.103

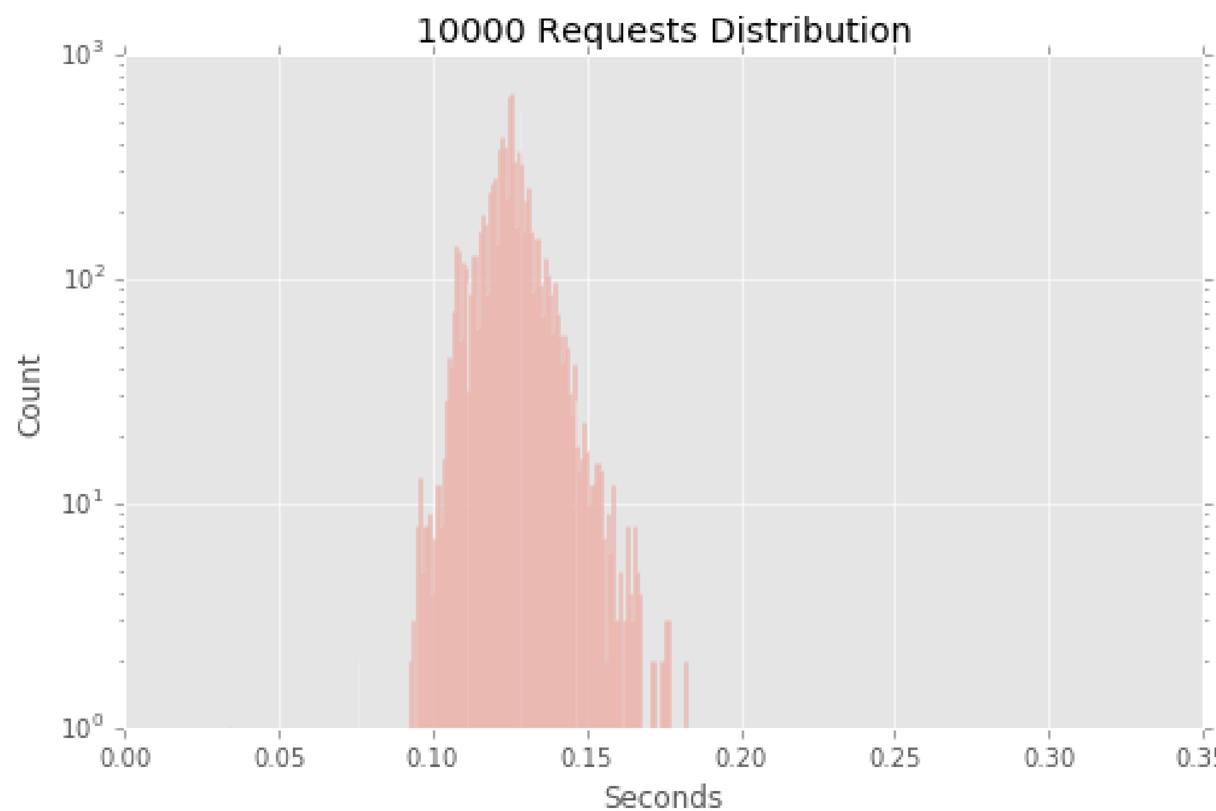
Results

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