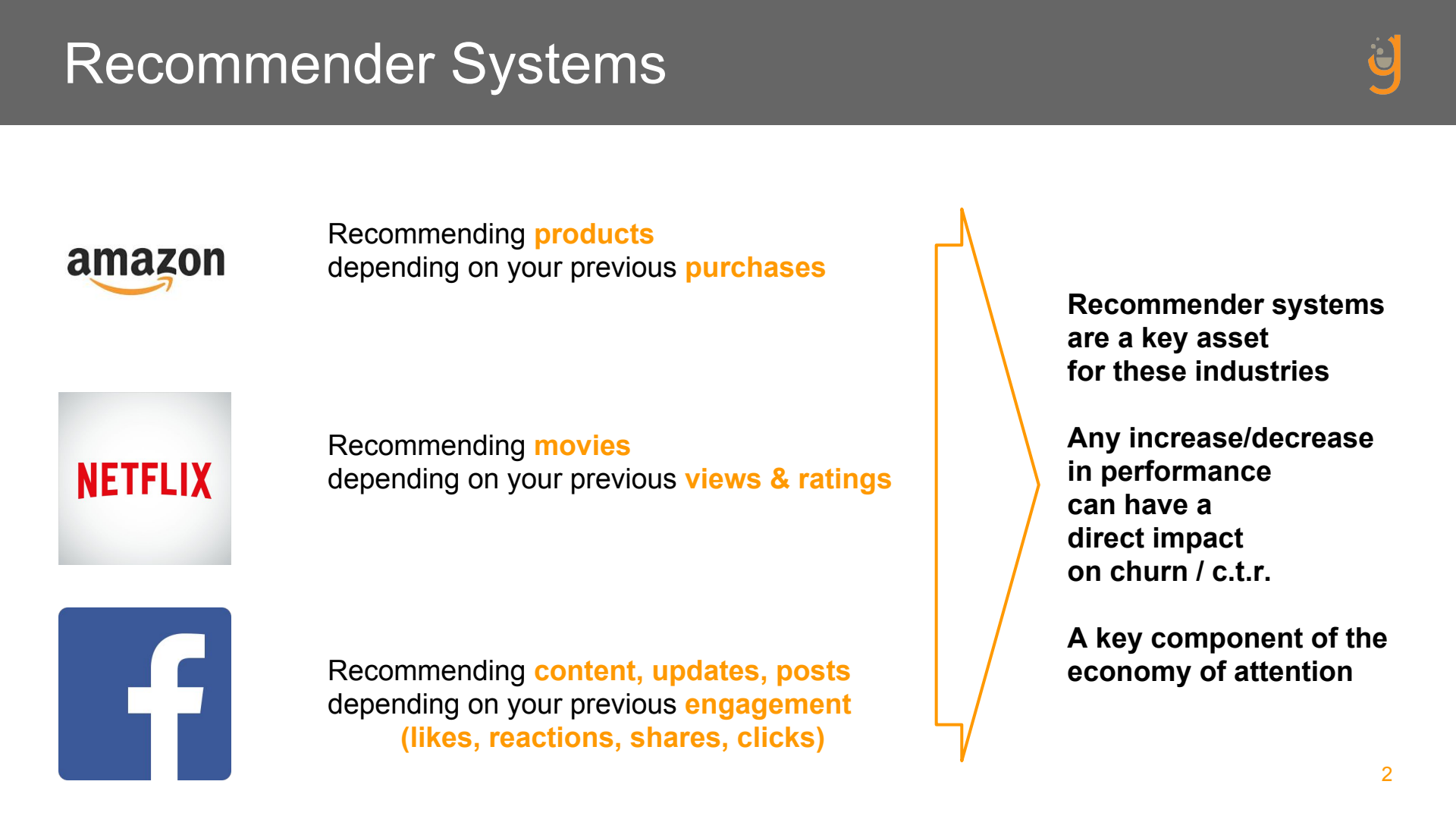
Building Recommendation Systems in Python using Apache Spark

December 3, 2016
Galvanize, Seattle

Recommender Systems

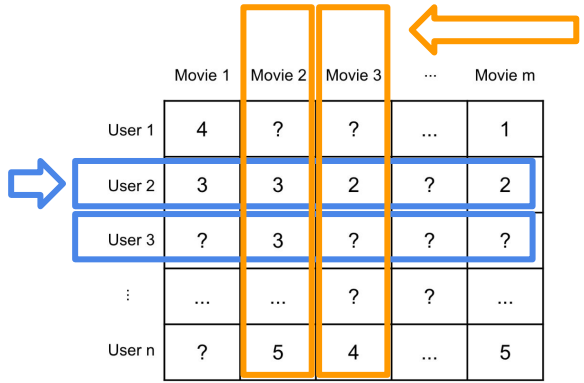


General Principles of Recommendation

User-User similarity

Recommend content that has been liked/bought/viewed by people similar to you.

This similarity is based on commonality of likes/buys/views btw two users.



Item-Item similarity

Recommend content that is similar to the content you've liked/bought/viewed.

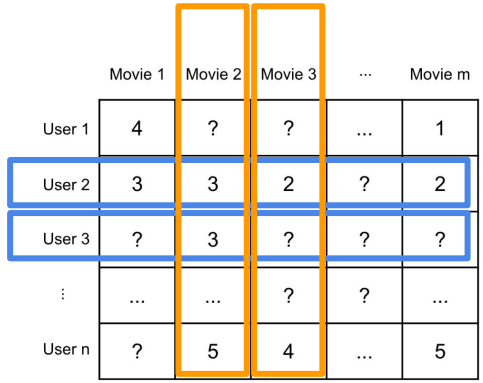
This similarity is based on commonality of likes/buys/views btw two items.

Pb: this is SPARSE and HUGE

Most items are unrated
(1-2% of ratings on Netflix)

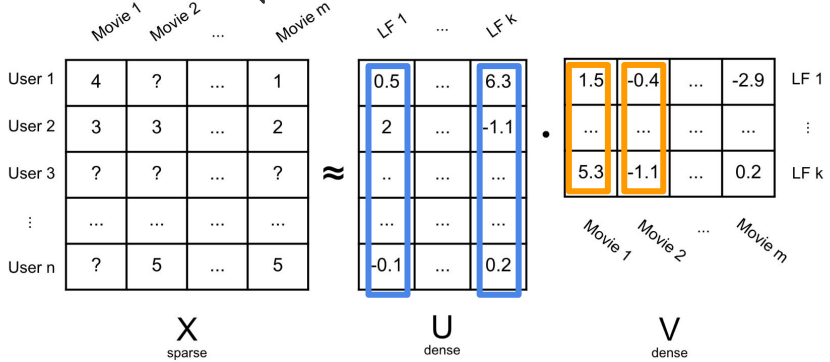
Matrix Factorization in a nutshell

Factorization: Finding “latent features” that let us factorize this sparse matrix as the product of two dense matrices



Factorization: finding U,V from X

Reconstruction: computing X from U,V



Matrix Factorization : Alternating Least Squares

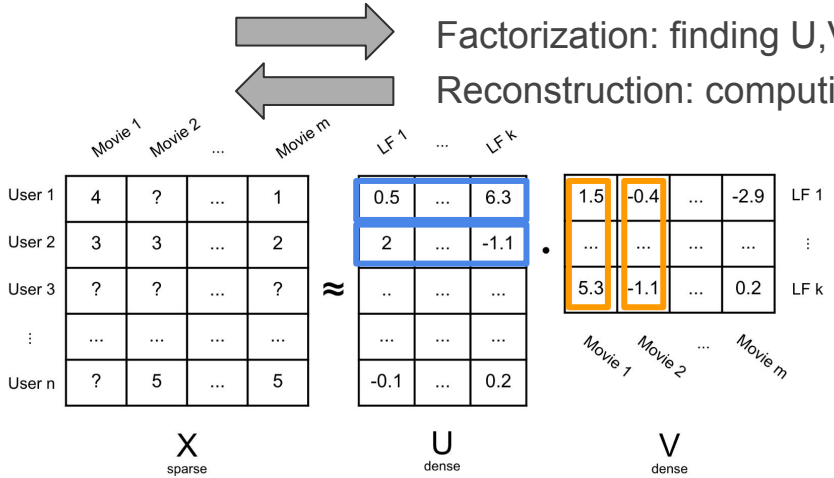
1) WHAT are Latent Vectors ?
Item vectors and **user vectors** used to decompose X as U.V

2) What are "GOOD" Latent Vectors ?
A couple of **item-vectors** **user-vectors** that minimize the error in reconstruction of X

3) HOW TO find "Good" Latent Vectors ?
Fix **item-vectors** and optimize **user-vectors** based on reconstruction error, then do the opposite

Factorization: Finding "latent features" that let us factorize this sparse matrix as the product of two dense matrices

Factorization: finding U,V from X
Reconstruction: computing X from U,V



Walkthrough

Building a recommender in Apache Spark