Movies Search Engine and Recommender System

Nassim Habbash 808292
Ricardo Matamoros 807450
Giacomo Villa 807462

CONTENTS

- Tools and Data
- Search Engine
- Recommender System
  - Personalized search recommendations
  - Content-based
  - Neural collaborative filtering
- Conclusions
- System demo

Tools

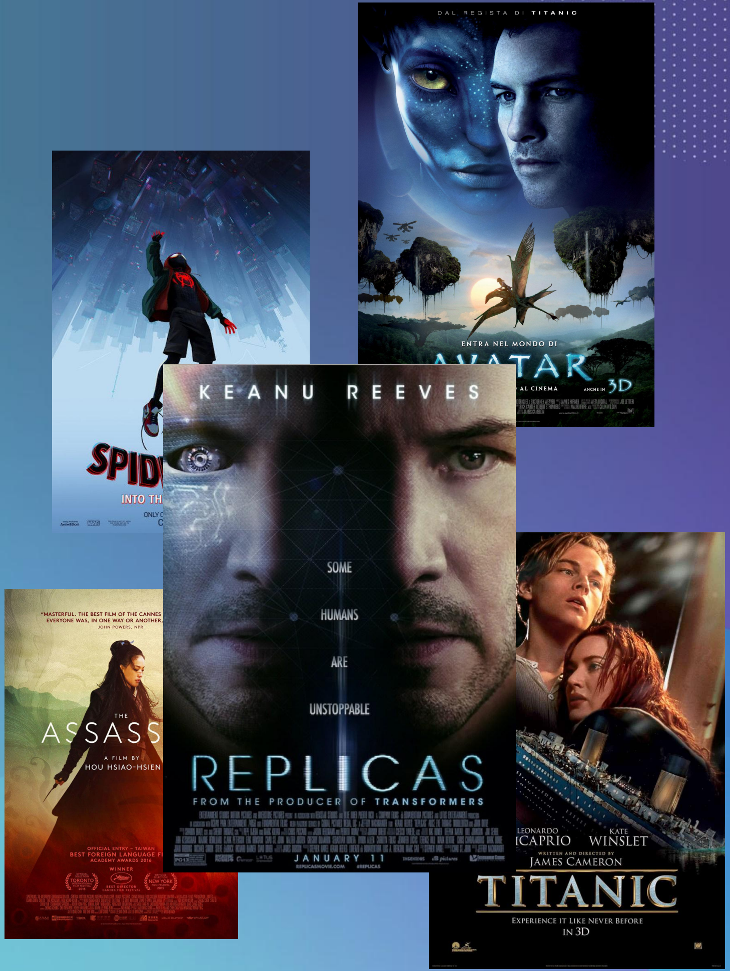
Libraries/Languages/Tools used:

- **ElasticSearch+Kibana** (Search Engine)
- **Keras** (Recommender System)
- **Django** (ORM/Web Framework)
- **Python** (Programming language)

Data - Movies

The dataset is MovieLens with approximately 45’000 movies textual information

- Title
- Overview
- Original Language
- Spoken Language
- Genres
- ...



Data - Ratings

The MovieLens data also contains about 270’000 user ratings on a scale from 1-5 obtained from the GroupLens website.

Data - Users

The user dataset contains 10 different profiles:

- Username
- Location
- Genres Preference
- Languages
- Film ratings
- ...

Search Engine

Indexing

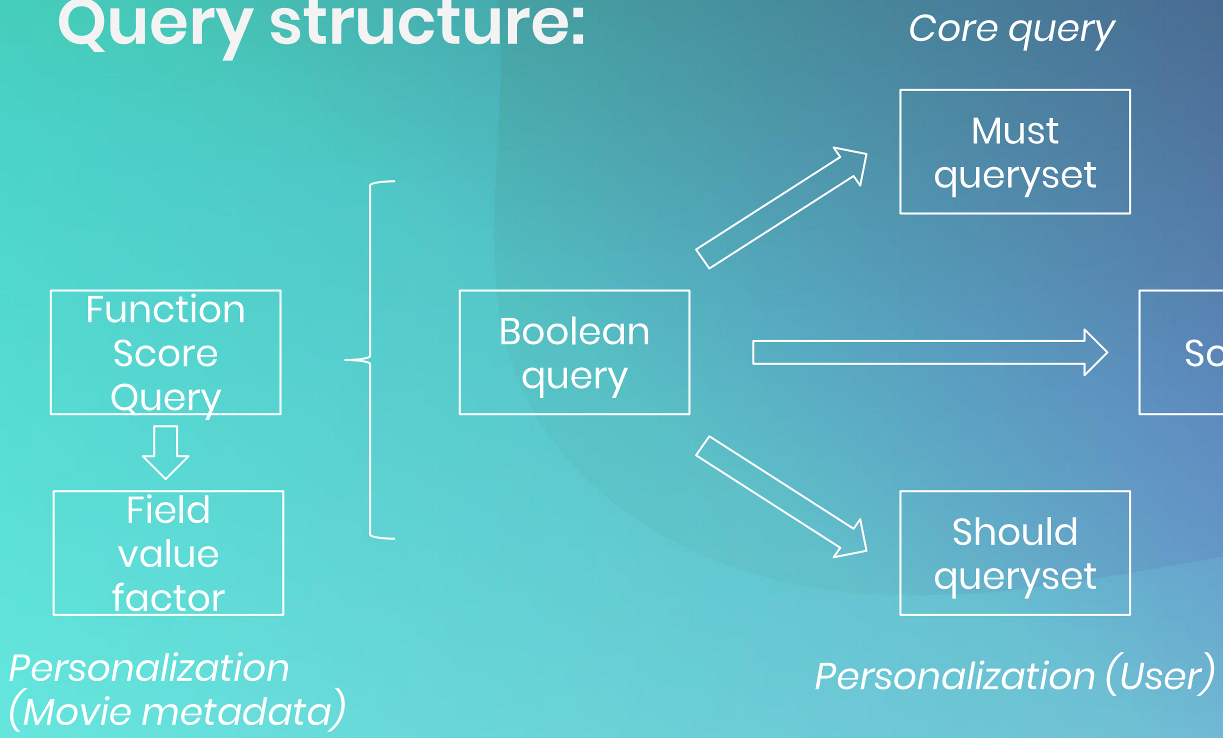
Defined analyzers:

- Standard
- No-stem

Analyzers specifications:

- Standard tokenizer, lowercase token filters, stopwords removal, Snowball stemmer
- Standard tokenizer, lowercase token filters, stopwords removal

Query structure:



Querying

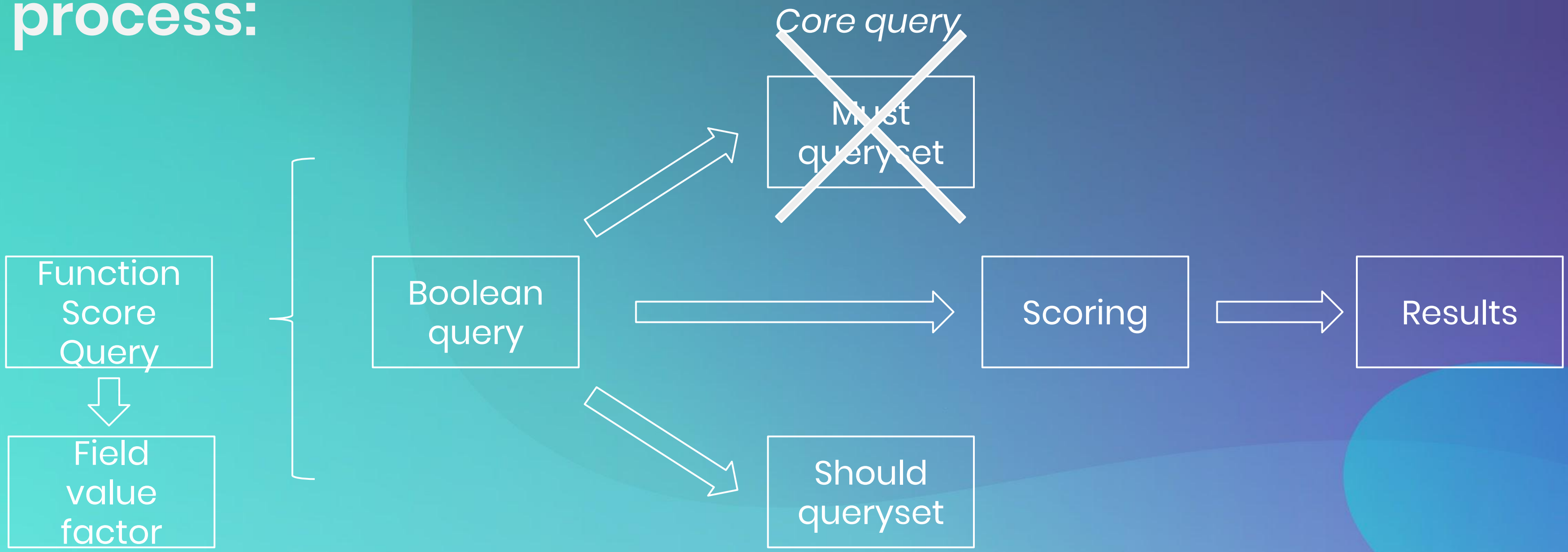
Query parameters:

- **Personalized:** based on user profile, using should queryset
- **Fuzzy:** allows inexact fuzzy matching
- **Synonyms:** creates multi-terms synonyms expansion
- **Popularity:** based on movie metadata, using function score query
- **Weighted vote:** based on movie metadata, using function score query
- **Phrasal:** allows exact search

Recommender Systems

PERSONALIZED SEARCH RECOMMENDATIONS

Recommendation process:

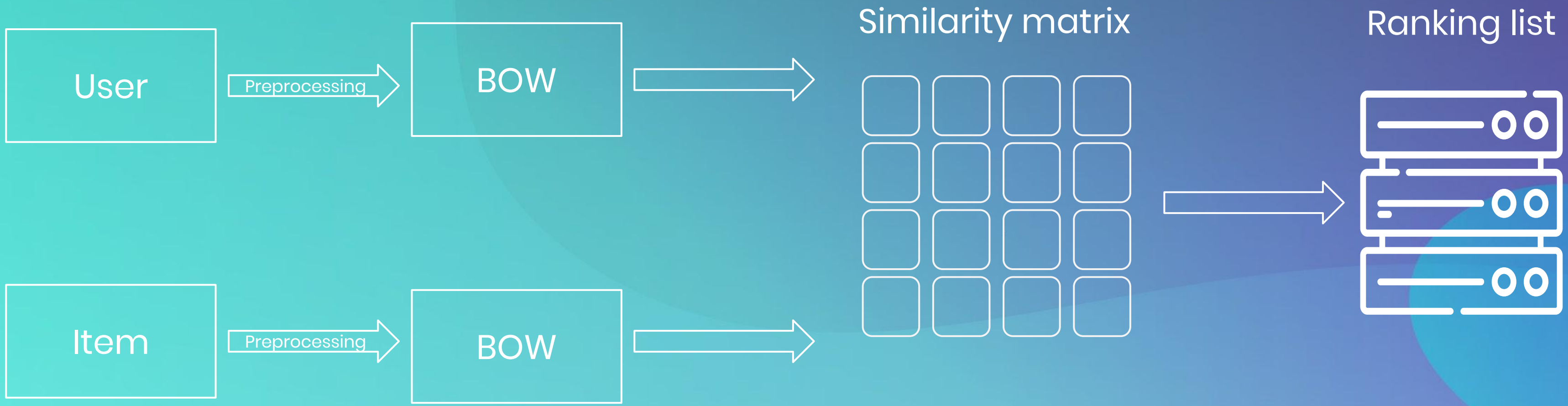


Personalization (Movie metadata)

Personalization (User)

CONTENT BASED

Recommendation process:

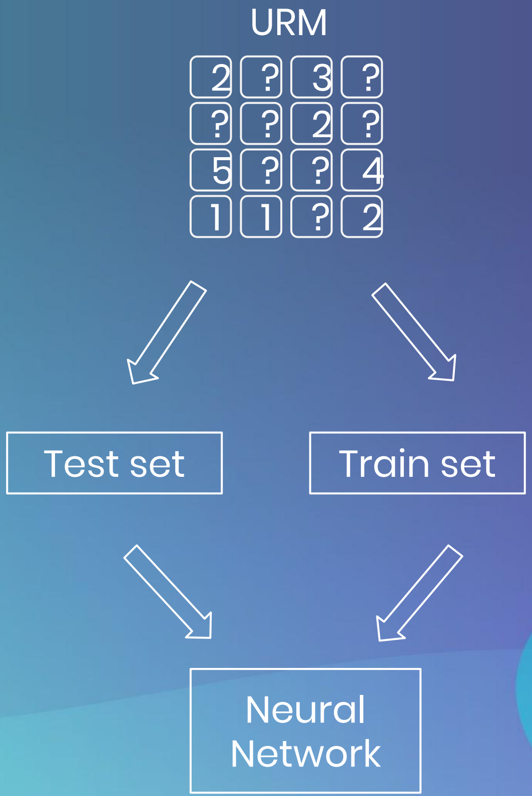


COLLABORATIVE FILTERING

We used deep learning to predict the missing ratings.

The steps to generate the recommendations are :

- Consider the URM matrix
- Split dataset in train and test set
- Train a neural network
- Save model to predict ratings



URM

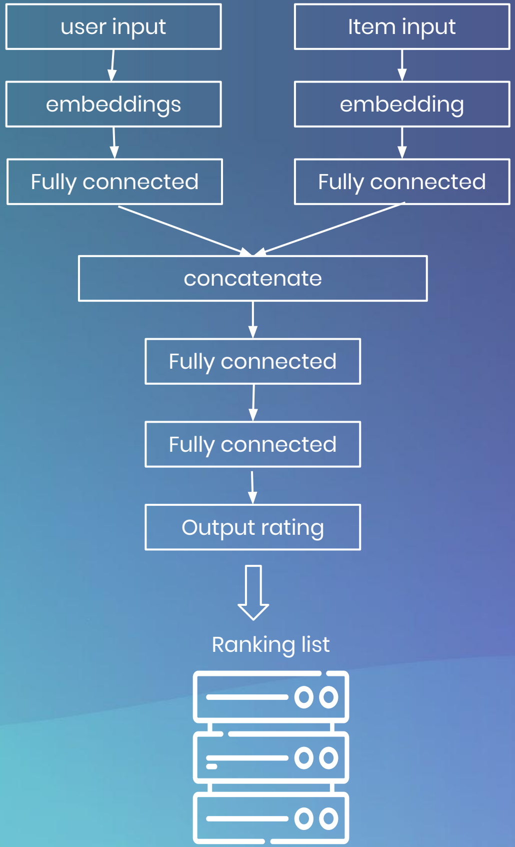
2	?	3	?
?	?	2	?
5	?	?	4
1	1	?	2

COLLABORATIVE FILTERING

Get the trained model (NN-FF), it is used to predict the ratings of films not yet seen by a user.

The steps to generate the recommendations are :

- Choose whether to applied filter or not
- Get user's Id, we compute the movies set not yet seen
- Generate list of couple , ex. (user_id, movie_id)
- Compute the missing ratings with model
- Order the list of films according to the predicted ratings
- Return k recommendations



user input

Item input

embeddings

embedding

Fully connected

Fully connected

concatenate

Fully connected

Fully connected

Output rating

Ranking list

ADVANTAGES AND DISADVANTAGES

Advantages :

- CB provides independence by users
- CB provides transparency by unknown users
- CB avoids cold start
- CF improves with increasing data
- CF is well suited to differents domains
- CF improves serendipity

Disadvantages:

- CB has problems of scalability with big datasets.
- CB has problems with filter bubbles.
- CF has problems with unbalanced datasets.
- CF with DL requires to retrain NN for new users and new items.

Conclusions

- Additional relevance dimensions (Search Engine)
- Custom similarities and mappings in Elasticsearch (Search Engine)
- Formal performance measures
- Denser items and users representations (Recommender System)
- Recommender hybridization (Recommender System)

Thanks for the attention 🍿🥤