Recommendation Systems for Décathlon - Discussion

Machine Learning I
MATH60629

Case developed by Jérémi DeBlois-Beaucage supervised by Laurent Charlin & Renaud Legoux

Presentation of the case

Q1: Which model(s) for a recommendation system would the data science team need to choose, and why?

Item-based vs. User-based

- |Users| >> |Items|
- Item-based
  - (+) Stability
  - (+) Better performance (not always)
  - (+) Explainability
- User-based
  - (+) Diversity

Plan

- Choose the right model for a recommendation task
- Models chosen by Décathlon
  - Basic models, as reference
  - Model 1: Based on similarity between product images
  - Model 2: Collaborative filtering based on products
  - Model 3: Matrix factorization
  - Model 4: Recurrent neural networks
  - Chosen metrics
  - Results and final choice
- Limits of the current models and the next steps being considered by Décathlon

How to Choose the Right Model?

- Somewhat subjective task
- Imperative: the model must be able to process a large amount of data
- Started with simpler models and then moved on to more complex ones
- Final choice according to performance on chosen metrics and logistical considerations
- 4 models have been selected

(Basic) Model 0: Reference Point

- Random recommendation
- Recommending the same 10 most popular items to every user

Model 1: Based on Similarity Between Product Images

Take the vector that represents the image of each product (representation from a CNN model)

1. **For each user, a list of all the products with which they have *interacted* is extracted.**

2. **For each *interacted product*, the 10 most "similar" products are chosen, based on the cosine distance between their image vectors.**

3. **The most similar product gets 10 "points", the second one gets 9, and so on. Points are added up, and the 5 products with the highest scores get recommended.**

Model 1: Based on Similarity Between Product Images

Take the vector that represents the image of each product (representation from a CNN model)

1. For each user, a list of all the products with which they have *interacted* is extracted.

2. For each *interacted product*, the 10 most “similar” products are chosen, based on the cosine distance between their image vectors.

3. The most similar product gets 10 “points”, the second one gets 9, and so on. Points are added up, and the 5 products with the highest scores get recommended.

User A has interacted in the past with items **3** and **5**

For each item, we identify the 10 items in the rest of the catalog that are most similar. We give 10 pts to the most similar one, 9 pts to the second most, and so on

3	5	
7	11	10 pts
12	7	9 pts
37	22	8 pts
11	30	7 pts
22	1	6 pts
6	26	5 pts
1	27	4 pts
28	12	3 pts
29	15	2 pts
30	9	1 pts

We sum all the scores, and recommend the top five items to our user

Recommendations to user **A**:

Item **7** (19 pts)
Item **11** (17 pts)
Item **22** (14 pts)
Item **12** (12 pts)
Item **1** (10 pts)

User A has interacted in the past with items **3** and **5**

For each item, we identify the 10 items in the rest of the catalog that are most similar. We give 10 pts to the most similar one, 9 pts to the second most, and so on

Item 3	Item 5	
7	11	10 pts
12	7	9 pts
37	22	8 pts
11	30	7 pts
22	1	6 pts
6	26	5 pts
1	27	4 pts
28	12	3 pts
29	15	2 pts
30	9	1 pts

We sum all the scores, and recommend the top five items to our user

Recommendations to user **A**:
Item **7** (19 pts)
Item **11** (17 pts)
Item **22** (14 pts)
Item **12** (12 pts)
Item **1** (10 pts)

Pros and Cons

- (+) Even new or unpopular products can get recommended
- (+) Explanations can be offered (*because you bought product X, you could find these products interesting*)
- (+) Easy to implement

- (-) The recommended products are very similar to previously purchased items
- (-) *Cold-start* problem for the users

Model 2: Collaborative Filtering Based on Products

Very similar model to the previous one. Instead of calculating the similarity of image vectors, similarity is calculated according to a user-product interaction matrix.

1. For each user, a list of all the products with which they have interacted is extracted.

2. For each product, the 10 most “similar” products are chosen, based on a cosine distance between their respective lines in the user-product interaction matrix.

3. The final score is similar to the one produced by model 1. However, instead of attributing arbitrary points (10 pts for the 1st, 9 for the 2nd, etc.), the similarity scores are used directly. Points are added, and the 5 products with the highest scores get recommended.

Example: similarity between two products

- In this matrix, lines represent users and columns represent products. A value of 1 indicates an interest, and 0 means no interaction.
  - For example, the first column shows that only User 4 has interacted with Product 1.
- The cosine similarity between the first (a =[0 0 0 1]) and the second product (b=[0 1 0 0]) would be 0. No user has interacted with both products.
- The similarity between the third (c=[1 0 1 0]) and the fifth product (e=[1 0 1 1]) is 0.82. The closer the value is to 1, the greater the similarity between the products.

Products

```
[0, 0, 1, 0, 1, 0]
[0, 1, 0, 0, 0, 0]
[0, 0, 1, 0, 1, 0]
[1, 0, 0, 0, 1, 0]
```

Users

Pros and Cons

- (+) Explanations can be offered (because you bought product X, you could find these products interesting)

- (+) Easy and quick to implement

- (+) Collaborative filtering usually yields good results

- (-) Cold-start problem for the users and the products

Model 3 : Matrix Factorization

- Completion of the user-product interaction matrix through matrix factorization
  - Several methodologies have been used, like Singular Value Decomposition and Non-Negative Matrix Factorization.
- It predicts the probability of an interaction with each product. Recommended products are the ones with the highest probabilities of interaction.

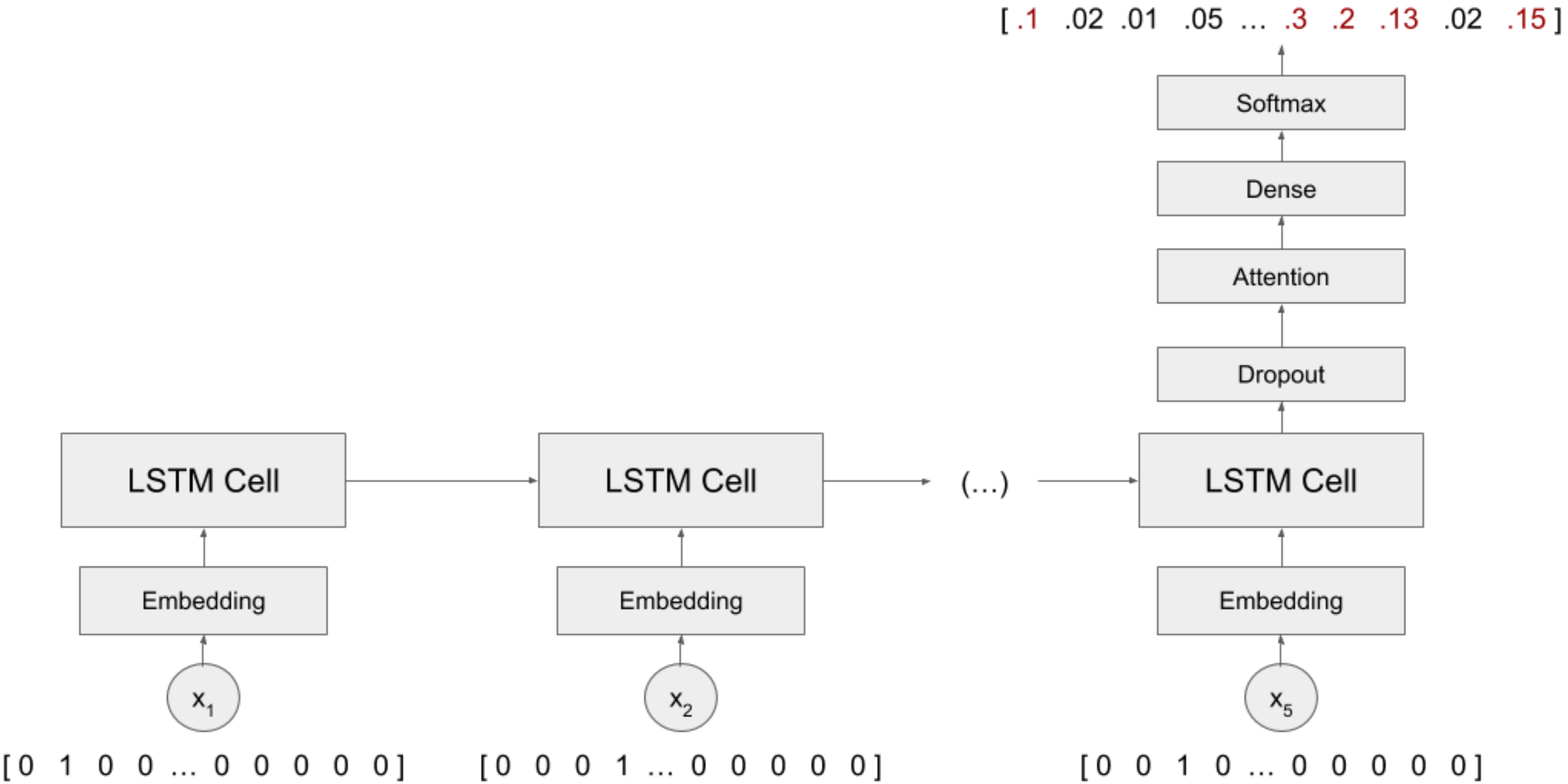
Pros and Cons

- (+) Matrix factorization usually yields good results
- (+) It can reveal interesting underlying characteristics

- (-) Cold-start problem for the users and the products
- (-) Important computational costs
- (-) Deploying the models requires a more complex virtual infrastructure*

Model 4: Recurrent Neural Networks

- This model tackles the problem as a sequence of products with which the user interacts, and tries to predict what the next one could be.
  - Each product is represented by a one-hot vector.
    - Products are entered into the network sequentially, and the network predicts the next one.
  - The network then outputs a probability distribution for every product in the catalogue.
  - Long Short-Term Memory (LSTM) neurons are used, with dropout-type regularization and attention principles.


[.1 .02 .01 .05 … .3 .2 .13 .02 .15]
Softmax
Dense
Attention
Dropout
LSTM Cell
LSTM Cell
(…)
LSTM Cell
Embedding
Embedding
Embedding
x_1
x_2
x_5
[0 1 0 0 … 0 0 0 0 0]
[0 0 0 1 … 0 0 0 0 0]
[0 0 1 0 … 0 0 0 0 0]

Pros and Cons

- (+) New users can be easily added
- (+) Recurrent networks usually give very good results

- (-) Cold-start problem for the products
- (-) Works better if a user has interacted with several products

Metrics

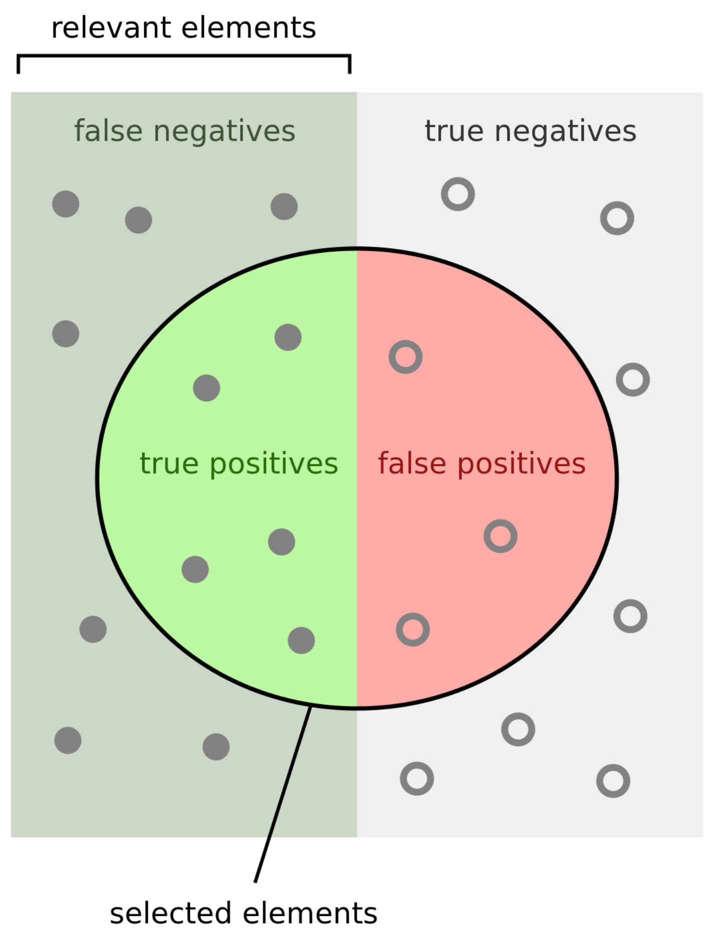
Only off-line metrics: the team does not have access to on-line evaluation or user studies

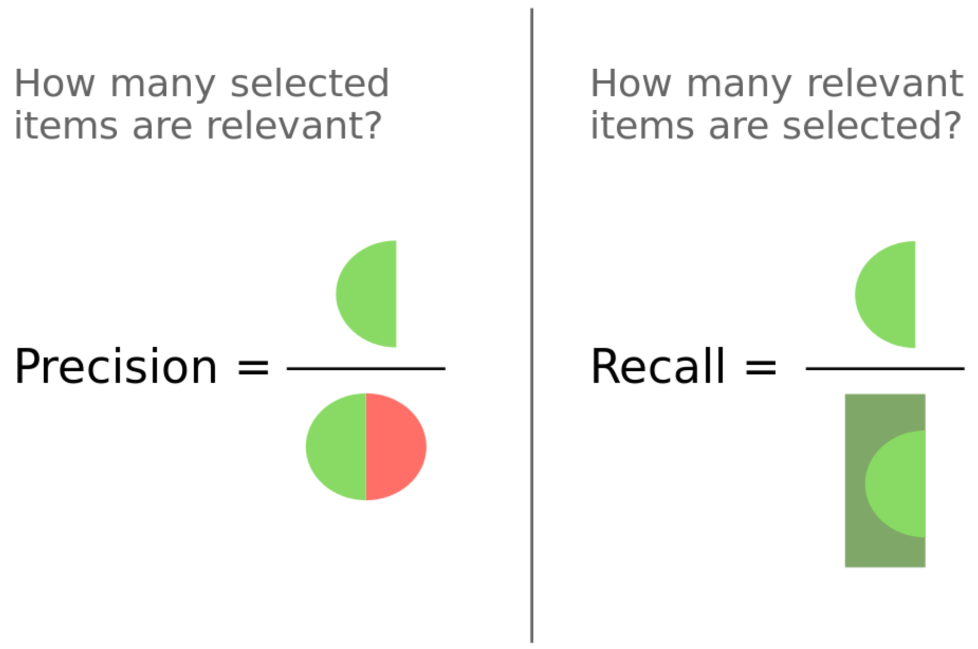
A. **Accuracy:** proportion of the recommended products that actually get bought by the users

B. **Recall:** proportion of products that were actually bought which were recommended

C. **Coverage:** proportion of products that were recommended to at least 1 user

Option to not consider diversity or serendipity

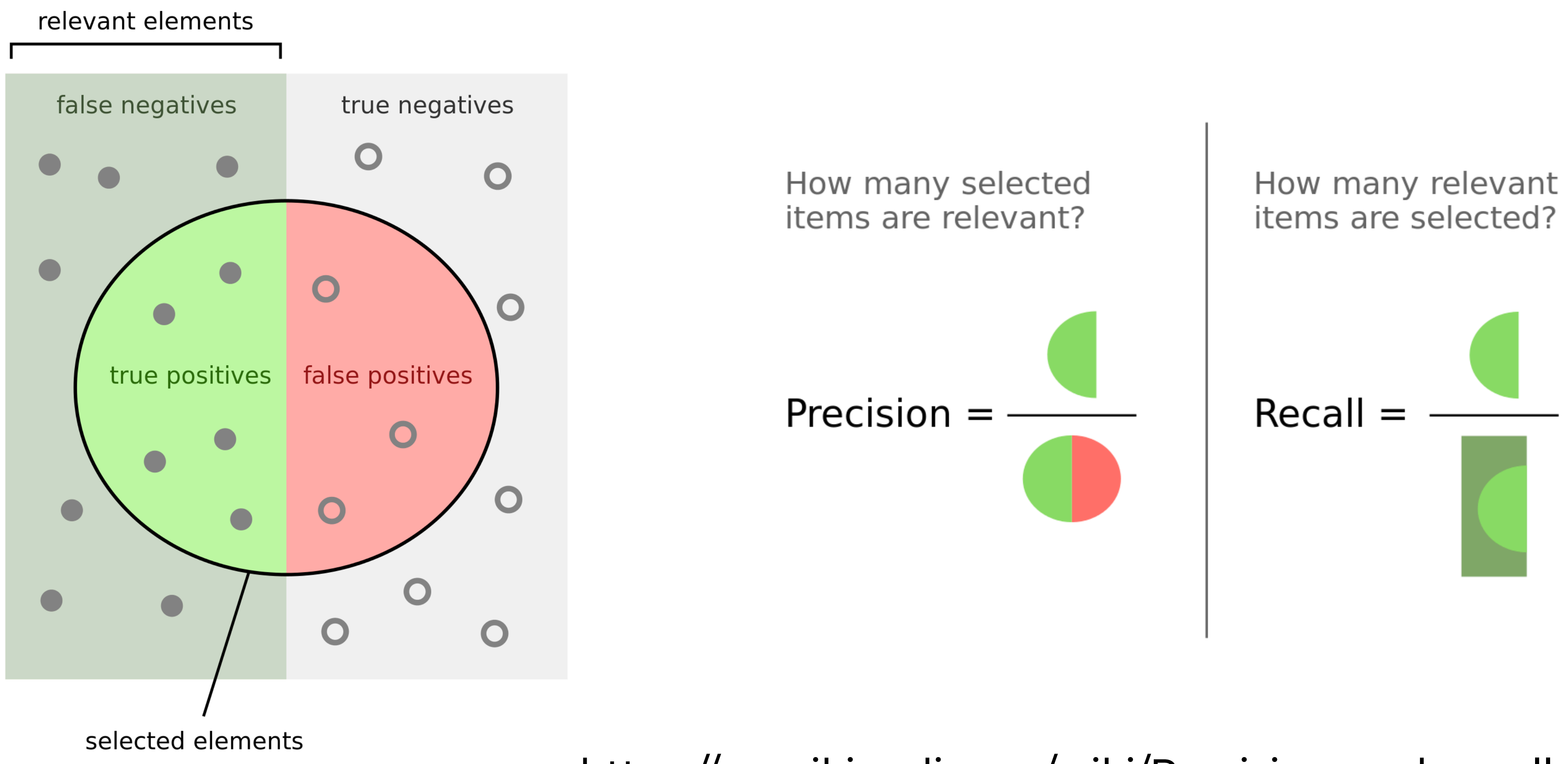




https://en.wikipedia.org/wiki/Precision_and_recall

Results

Model	Random	Most popular	1. Visual Similarity	2. Collaborative, based on products	3. Matrix factorization	4. Recurrent Neural Networks (RNNs)
Precision	0,06%	1,5%	1,9%	3,4%	3,9%	4%
Recall	0,07%	1,8%	2,3%	4,1%	5,3%	5,7%
Coverage	91%	0,07%	37,1%	74%	69%	57%



https://en.wikipedia.org/wiki/Precision_and_recall