Recommender System

Exam Project of *“Advanced Information Management 2”*

Federico Motta

Università degli studi di Modena e Reggio Emilia

April 23, 2020

Outline

Problem definition

Input:

$\mathbf{k}$ = *number of items to suggest*

$\mathbf{U} = \{u_1, \ldots, u_M\}$ *users*

$\mathbf{I} = \{ i_1, \ldots, i_N\}$ *items*

$\mathbf{O} = \{\langle u, i, j \rangle : \text{“user } u \text{ prefers item } i \text{ over item } j\text{”}\}$ *observations*

Note: *all type of user preferences (e.g. rating, like, browsing history) can be converted to* $\langle u, i, j \rangle$ *observations*

Output:

given a user $\tilde{u}$ return k never seen items which he probably appreciates

Papers

- Shams, B., & Haratizadeh, S. (2016). Graph-based collaborative ranking. *arXiv preprint arXiv:1604.03147*. Available at https://arxiv.org/abs/1604.03147.

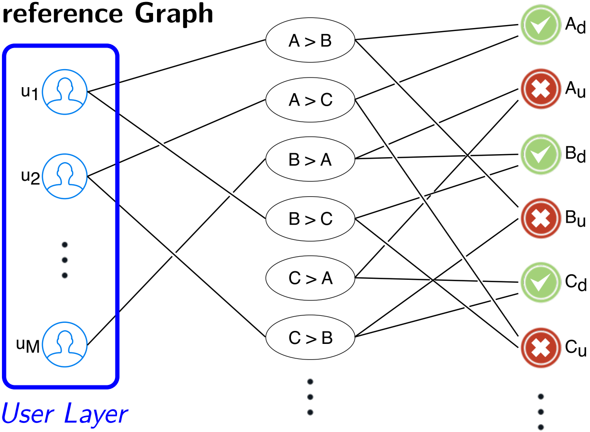
- Lops, P., de Gemmis, M., & Semeraro, G. (2011). Content-based recommender systems: State of the art and trends. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender systems handbook* (pp. 73–105). doi:10.1007/978-0-387-85820-3_3

- Basu, C., Hirsh, H., & Cohen, W. (1998). Recommendation as classification: Using social and content-based information in recommendation. In *Proceedings of the national conference on artificial intelligence (aaai '98)*. Available at http://cs.cmu.edu/~wcohen/postscript/aaai-98-collab.ps, AAAI Press.

- Baeza-Yates, R., Ribeiro-Neto, B. et al. (1999). Text operations. In *Modern information retrieval* (Chap. 7, Vol. 463, pp. 163–190, with Ziviani, Nivio). Available at https://homepages.dcc.ufmg.br/~nivio/papers/aw99.ps. Addison–Wesley.

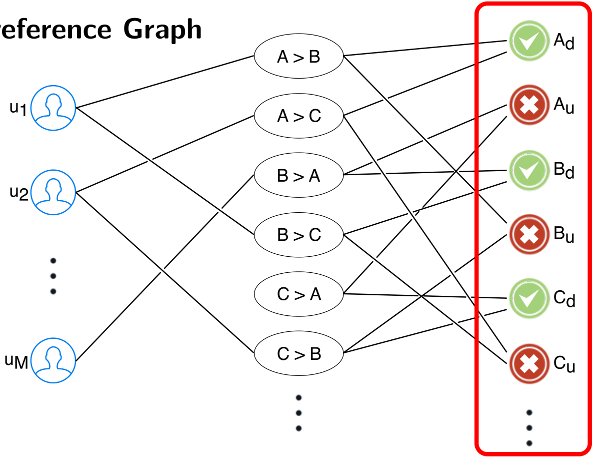
Collaborative–filtering (GRank) (Shams & Haratizadeh, 2016)

Tripartite Preference Graph



Collaborative–filtering (GRank) (Shams & Haratizadeh, 2016)

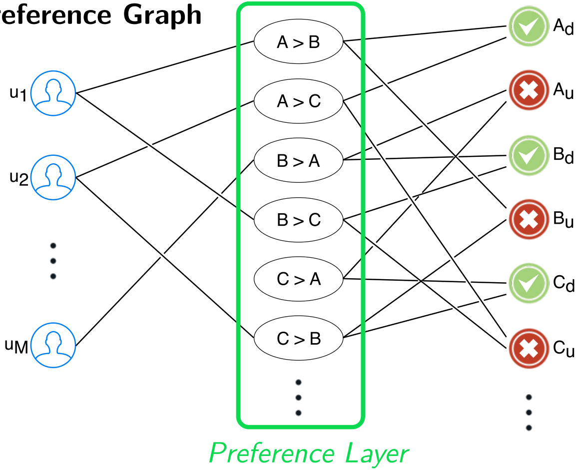
Tripartite Preference Graph



Representative Layer

Collaborative–filtering (GRank) (Shams & Haratizadeh, 2016)

Tripartite Preference Graph



Notation:

$\mathbf{P} = \{\ p_1,\ \ldots,\ p_{N\cdot(N-1)}\ \}$ *preferences*

$p = \langle\, i, j\,\rangle \in \mathbf{P} =$ " $i > j$ " " item i is preferred to item j "

Ranking Algorithm

$$\mathbf{GR} : U \times I \rightarrow \mathbb{R}_{\geq 0}$$

$$(1) \quad GR(i) = \frac{\overline{PPR}_{[i_d]}}{\overline{PPR}_{[i_d]} + \overline{PPR}_{[i_u]}}$$

"goodness" of unseen item i

Author's note: overlined symbols are vectors

Ranking Algorithm

$$\mathbf{GR} : U \times I \rightarrow \mathbb{R}_{\geq 0}$$

(1) $GR(i) = \dfrac{\overline{PPR}_{[i_d]}}{\overline{PPR}_{[i_d]} + \overline{PPR}_{[i_u]}}$ "goodness" of unseen item i

(2) $\overline{PPR}_t = \alpha \cdot T \cdot \overline{PPR}_{t-1} + (1 - \alpha) \cdot \overline{PV}$ Personalized Page Rank

Author's note: overlined symbols are vectors

Ranking Algorithm

$$\mathbf{GR} : U \times I \to \mathbb{R}_{\geq 0}$$

(1) $$GR(i) = \frac{\overline{PPR}_{[i_d]}}{\overline{PPR}_{[i_d]} + \overline{PPR}_{[i_u]}}$$ "goodness" of unseen item i

(2) $$\overline{PPR}_t = \alpha \cdot T \cdot \overline{PPR}_{t-1} + (1 - \alpha) \cdot \overline{PV}$$ Personalized Page Rank

(3) $$T_{[i\,j]} = \begin{cases} \frac{1}{degree(i)} & \text{if } (i, j) \in E \\ 0 & otherwise \end{cases}$$ Transition Matrix

(4) $$PV_{[j]} = \begin{cases} 1 & \text{if } j = \tilde{u} \\ 0 & \text{otherwise} \end{cases}$$ Personalized Vector

Author's note: overlined symbols are vectors

Grank Complexity

- $|V| = \underbrace{M}_{\text{users}} + \overbrace{N \cdot (N-1)}^{\text{preferences}} + \underbrace{2 \cdot N}_{\text{representatives}}$
- $|E| = \underbrace{S}_{\text{user–preference}} + \overbrace{2 \cdot N \cdot (N-1)}^{\text{preference–representative}}$ $\qquad S \simeq c \cdot N^2 \quad c \approx 2.48$

Grank Complexity

- $|V| = \underbrace{M}_{\text{users}} + \overbrace{N \cdot (N-1)}^{\text{preferences}} + \underbrace{2 \cdot N}_{\text{representatives}}$
- $|E| = \underbrace{S}_{\text{user–preference}} + \overbrace{2 \cdot N \cdot (N-1)}^{\text{preference–representative}}$ $\quad S \simeq c \cdot N^2 \quad c \approx 2.48$
- graph construction $\simeq O(\,N^2 + M\,)$ ***space complexity***
- recommendation $\simeq O(\,t \cdot E\,) = O(\,N^2\,)$ ***time complexity*** $\quad t \leq 20$

Grank Complexity

- $|V| = \underbrace{M}_{\text{users}} + \overbrace{N \cdot (N-1)}^{\text{preferences}} + \underbrace{2 \cdot N}_{\text{representatives}}$
- $|E| = \underbrace{S}_{\text{user–preference}} + \overbrace{2 \cdot N \cdot (N-1)}^{\text{preference–representative}}$ $\qquad S \simeq c \cdot N^2 \quad c \approx 2.48$
- graph construction $\simeq O(\, N^2 + M \,)$ ***space complexity***
- recommendation $\simeq O(\, t \cdot E \,) = O(\, N^2 \,)$ ***time complexity*** $\qquad t \leq 20$

Datasets built with $\mathbb{T} = 30, 40, 50, 60$ reviews[a] for each user ($M = \mathit{100}, \mathit{300}, \mathit{1000}$).

[a] using a ***stratified random sampling*** as suggested in *"Basu, Hirsh, and Cohen, 1998"*

Transition Matrix

Alert: building the transition matrix with with $\mathbb{T} = 30$ and $M \simeq 2\,000$ takes **26 hours** and **11.4 GB** of RAM (using a CSR matrix)

Transition Matrix

Alert: building the transition matrix with with $\mathbb{T} = 30$ and $M \simeq 2\,000$ takes **26 hours** and **11.4 GB** of RAM (using a CSR matrix) **that is too much !**

Transition Matrix

Alert: building the transition matrix with with $\mathbb{T} = 30$ and $M \simeq 2\,000$ takes **26 hours** and **11.4 GB** of RAM (using a CSR matrix) **that is too much !**

$$T = \begin{array}{c} \\ \text{users} \\ \text{preferences} \\ \\ \text{representatives} \end{array} \begin{array}{c} \begin{array}{cccc} \overbrace{\quad}^{\text{users}} & \multicolumn{2}{c}{\overbrace{\quad\quad}^{\text{preferences}}} & \overbrace{\quad}^{\text{representatives}} \end{array} \\ \left(\begin{array}{c|c|c|c} & 1 & & \\ \hline 2 & & & 3 \\ \hline & & & 4 \\ \hline & 5 & 6 & \end{array}\right) \end{array}$$

- density of T is $\approx 2.4 \cdot 10^{-8}$

Transition Matrix

Alert: building the transition matrix with with $\mathbb{T} = 30$ and $M \simeq 2\,000$ takes **26 hours** and **11.4 GB** of RAM (using a CSR matrix) **that is too much !**

$$T = \begin{array}{rc} & \begin{array}{c|cc|c} \text{users} & \text{preferences} & & \text{representatives} \end{array} \\ \begin{array}{r}\text{users}\\ \text{preferences}\\ \\ \text{representatives}\end{array} & \left(\begin{array}{c|c|c|c} & 1 & & \\ \hline 2 & & & 3 \\ \hline & & & 4 \\ \hline & 5 & 6 & \end{array}\right)\end{array}$$

- density of T is $\approx 2.4 \cdot 10^{-8}$
- this is the **largest** submatrix
- all elements in block 4 are $^1/_2$
- all elements in block 6 are $^1/_{(n-1)}$

Transition Matrix

Alert: building the transition matrix with with $\mathbb{T} = 30$ and $M \simeq 2\,000$ takes **26 hours** and **11.4 GB** of RAM (using a CSR matrix) **that is too much !**

$$T = \begin{array}{r} \\ \text{users} \\ \text{preferences} \\ \\ \text{representatives} \end{array} \begin{array}{c} \begin{array}{cccc} \overbrace{\quad}^{\text{users}} & \multicolumn{2}{c}{\overbrace{\qquad\quad}^{\text{preferences}}} & \overbrace{\quad}^{\text{representatives}} \end{array} \\ \left(\begin{array}{c|c|c|c} & 1 & & \\ \hline 2 & & & 3 \\ \hline & & & 4 \\ \hline & 5 & 6 & \end{array}\right) \end{array}$$

- density of T is $\approx 2.4 \cdot 10^{-8}$
- this is the **largest** submatrix
- all elements in block 4 are $^1/_2$
- all elements in block 6 are $^1/_{(n-1)}$

Thus, T has been split in:

T_1 = blocks 1, 2, 3, 5

T_2 = **blocks 4 and 6** (**generated row by row** with the *yield* statement)

Since $T = T_1 + T_2$ nothing changes a part performances ✓

Keyword–based (Naïve Bayes) (Lops, de Gemmis, & Semeraro, 2011)

Vector Space Model:

$\mathbf{D} = \{\ d_1,\ \ldots,\ \ d_N\}$ *corpus* *(of documents)*

$\mathbf{T} = \{\ \ t_1,\ \ldots,\ \ t_Z\}$ *dictionary* *(of keywords/terms)*

$\mathbf{d_j} = \{w_{1\,j},\ \ldots,\ w_{Z\,j}\} \in D$ *weights of terms* $t_1 \cdots t_Z$ *in document* j

Keyword–based (Naïve Bayes) (Lops et al., 2011)

Vector Space Model:

$\mathbf{D} = \{\ d_1,\ \ldots,\ d_N\}$ *corpus (of documents)*

$\mathbf{T} = \{\ t_1,\ \ldots,\ t_Z\}$ *dictionary (of keywords/terms)*

$\mathbf{d_j} = \{w_{1\,j},\ \ldots,\ w_{Z\,j}\} \in D$ *weights of terms $t_1 \cdots t_Z$ in document j*

Weighting the terms:

TF-IDF handles well $\overbrace{\text{single/multiple occurrences,}}^{\textit{TF assumption}}$ $\overbrace{\text{rare/frequent terms}}^{\textit{IDF assumption}}$

$$\mathrm{TF\text{–}IDF}(t_k, d_j) = \overbrace{\frac{f_{k,j}}{\max_z f_{z,j}}}^{\textit{Term Frequency}} \cdot \underbrace{\log \frac{N}{n_k}}_{\textit{Inverse Document Frequency}}$$

$n_k =$ *$d \in D$ with at least one t_k*

$f_{k,j} =$ *frequency of t_k in d_j*

Keyword–based (Naïve Bayes) (Lops et al., 2011)

Vector Space Model:

$\mathbf{D} = \{\ d_1,\ \dots,\ d_N\}$ *corpus (of documents)*
$\mathbf{T} = \{\ t_1,\ \dots,\ t_Z\}$ *dictionary (of keywords/terms)*
$\mathbf{d_j} = \{w_{1\,j},\ \dots,\ w_{Z\,j}\} \in D$ *weights of terms $t_1 \cdots t_Z$ in document j*

Weighting the terms:

TF-IDF handles well $\overbrace{\text{single/multiple occurrences}}^{\textit{TF assumption}}$, $\overbrace{\text{rare/frequent terms}}^{\textit{IDF assumption}}$, $\overbrace{\text{long/short docs}}^{\textit{normalization assumption}}$.

$$\text{TF–IDF}(t_k, d_j) = \overbrace{\frac{f_{k,j}}{\max_z f_{z,j}}}^{\textit{Term Frequency}} \cdot \underbrace{\log \frac{N}{n_k}}_{\textit{Inverse Document Frequency}}$$

$n_k =$ *$d \in D$ with at least one t_k*
$f_{k,j} =$ *frequency of t_k in d_j*

$$\mathbf{w_{k,j}} = \text{TF–IDF}(t_k, d_j) \Big/ \sqrt{\sum_{s=1}^{|T|} \text{TF–IDF}(t_s, d_j)^2}$$ *(cosine normalization)*

Ranking Algorithm

$$\mathbf{c_{MAP}} = \underbrace{\arg\max_{c_j} P(c_j/d_i)}_{\substack{\text{maximum a posteriori probability} \\ \text{of document } d_i \text{ belonging to class } c_j \\ \text{(most likely class)}}} = \arg\max_{c_j} \frac{P(c_j) \cdot P(d_i/c_j)}{\cancel{P(d_i)}}$$

Ranking Algorithm

$$\mathbf{c_{MAP}} = \underbrace{\arg\max_{c_j} P(c_j/d_i)}_{\substack{\text{maximum a posteriori probability} \\ \text{of document } d_i \text{ belonging to class } c_j \\ \text{(most likely class)}}} = \arg\max_{c_j} \frac{P(c_j) \cdot P(d_i/c_j)}{\cancel{P(d_i)}}$$

Assumptions:

- bag of words
- conditional independence

Ranking Algorithm

$$c_{MAP} = \underbrace{\arg\max_{c_j} P(c_j/d_i)}_{\substack{\text{maximum a posteriori probability} \\ \text{of document } d_i \text{ belonging to class } c_j \\ \text{(most likely class)}}} = \arg\max_{c_j} \frac{P(c_j) \cdot P(d_i/c_j)}{\cancel{P(d_i)}}$$

Assumptions:

- bag of words
- conditional independence

Multinomial Naïve Bayes Model:

$$P(c_j/d_i) = P(c_j) \cdot \prod_{w \in d_i} P(w/c_j)$$

Descriptions Preprocessing [b]

- unescape HTML entities (“&” → “&”)
- substitute punctuation with spaces (“requirements:minimum” → “requirements minimum”)
- remove multiple spaces (“Washington   DC” → “Washington” “DC”)
- split camelCase words (“GoldenEye” → “Golden Eye”)
- lowercase (“Xbox” → “xbox”)

[b] Baeza-Yates, Ribeiro-Neto, et al., 1999 ∼ Chapter 7, “Text Operations”

Descriptions Preprocessing [b]

- unescape HTML entities ("&" → "&")
- substitute punctuation with spaces ("requirements:minimum" → "requirements minimum")
- remove multiple spaces ("Washington   DC" → "Washington" "DC")
- split camelCase words ("GoldenEye" → "Golden Eye")
- lowercase ("Xbox" → "xbox")
- tokenization ("Frisco and LA" → "Frisco" "and" "LA")

[b] Baeza-Yates, Ribeiro-Neto, et al., 1999 ∼ Chapter 7, "Text Operations"

Descriptions Preprocessing [b]

- unescape HTML entities (“&” → “&”)
- substitute punctuation with spaces (“requirements:minimum” → “requirements minimum”)
- remove multiple spaces (“Washington   DC” → “Washington” “DC”)
- split camelCase words (“GoldenEye” → “Golden Eye”)
- lowercase (“Xbox” → “xbox”)
- tokenization (“Frisco and LA” → “Frisco” “and” “LA”)
- stop words removal (“drive” “a” “train” → “drive” “train”)

[b] Baeza-Yates, Ribeiro-Neto, et al., 1999 ∼ Chapter 7, “Text Operations”

Descriptions Preprocessing [b]

- unescape HTML entities ("&" → "&")
- substitute punctuation with spaces ("requirements:minimum" → "requirements minimum")
- remove multiple spaces ("Washington   DC" → "Washington" "DC")
- split camelCase words ("GoldenEye" → "Golden Eye")
- lowercase ("Xbox" → "xbox")
- tokenization ("Frisco and LA" → "Frisco" "and" "LA")
- stop words removal ("drive" "a" "train" → "drive" "train")
- stemming (with Porter) ("playable" → "play")

[b] Baeza-Yates, Ribeiro-Neto, et al., 1999 ∼ Chapter 7, "Text Operations"

Descriptions Preprocessing [b]

- unescape HTML entities (“&” → “&”)
- substitute punctuation with spaces (“requirements:minimum” → “requirements minimum”)
- remove multiple spaces (“Washington   DC” → “Washington” “DC”)
- split camelCase words (“GoldenEye” → “Golden Eye”)
- lowercase (“Xbox” → “xbox”)
- tokenization (“Frisco and LA” → “Frisco” “and” “LA”)
- stop words removal (“drive” “a” “train” → “drive” “train”)
- stemming (with Porter) (“playable” → “play”)
- lemmatization (with WordNet) (optional)

[b] Baeza-Yates, Ribeiro-Neto, et al., 1999 ∼ Chapter 7, “Text Operations”

Performances

Normalized Discounted Cumulative Gain is a standard strategy used to assess recommender systems by comparing the quality of their top–k suggestions.

Performances

Normalized Discounted Cumulative Gain is a standard strategy used to assess recommender systems by comparing the quality of their top–k suggestions.

$$\textbf{NDCG@k} = \sum_{i=1}^{k} \frac{\overbrace{2^{r_i^u}}^{\text{rating of user } u \text{ to i-th item}} - 1}{\log_2(i+1)} \Bigg/ \underbrace{\sum_{i=1}^{k} \frac{\overbrace{2^5}^{\text{max rating is 5} \bigstar} - 1}{\log_2(i+1)}}_{\text{normalization coefficient} = \alpha_u}$$

Performances

Normalized Discounted Cumulative Gain is a standard strategy used to assess recommender systems by comparing the quality of their top–k suggestions.

$$\textbf{NDCG@k} = \sum_{i=1}^{k} \frac{\overbrace{2^{r_i^u}}^{\text{rating of user } u \text{ to i-th item}} - 1}{\log_2(i+1)} \Bigg/ \underbrace{\sum_{i=1}^{k} \frac{\overbrace{2^5}^{\text{max rating is } 5 \bigstar} - 1}{\log_2(i+1)}}_{\text{normalization coefficient } = \alpha_u}$$

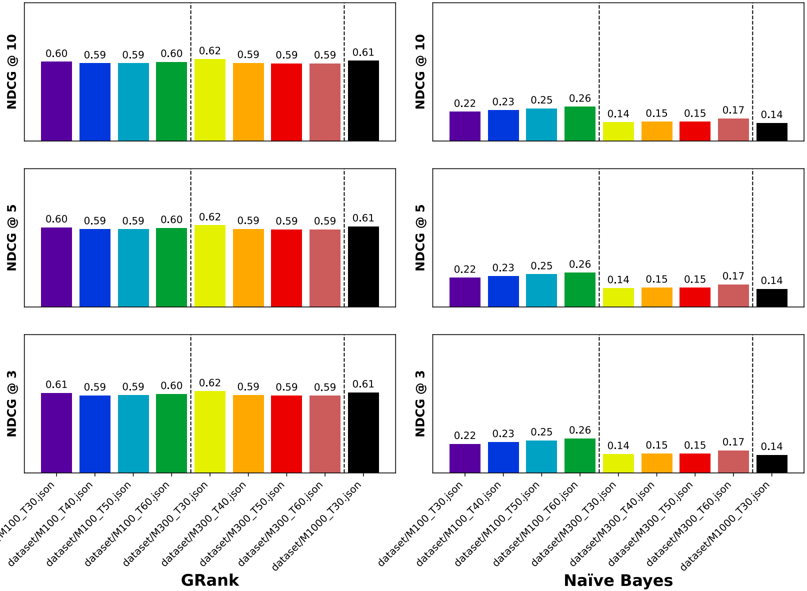
Other useful metrics are:

- accuracy $= \frac{\mathrm{TP}+\mathrm{TN}}{\mathrm{TP}+\mathrm{FP}+\mathrm{FN}+\mathrm{TN}}$
- precision $= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FP}}$
- recall $= \frac{\mathrm{TP}}{\mathrm{TP}+\mathrm{FN}}$
- F_1 score $= \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision}+\text{recall}}$

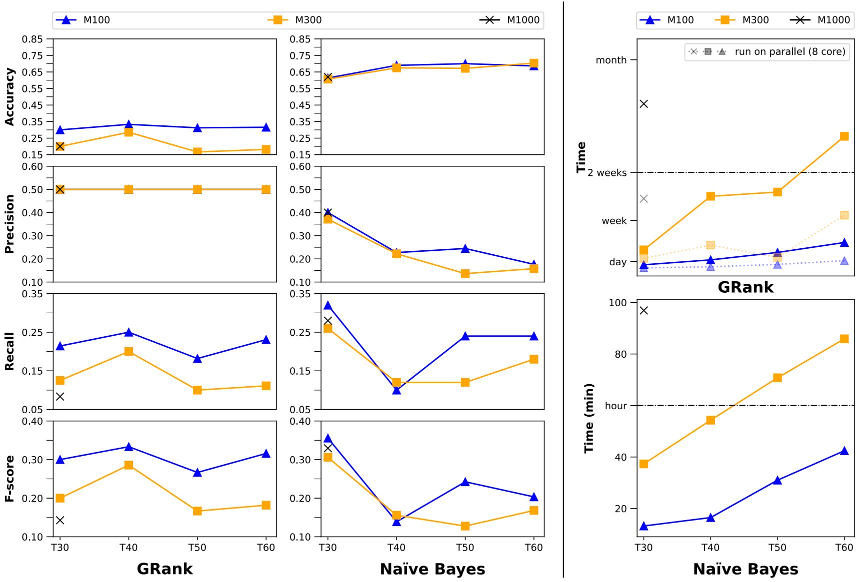
$$\begin{array}{c|cc} & \text{real positive} & \text{real negative} \\ \hline \text{predicted positive} & \mathrm{TP} & \mathrm{FP} \\ \text{predicted negative} & \mathrm{FN} & \mathrm{TN} \end{array}$$

confusion matrix

NDCG performance comparison



Other performance comparison



Development choices (source code available at https://github.com/TheJena/RecommenderSystem)

- Shrink MongoDB size *(instructions in: **docs/how_to_create_test_db_dump.pdf**)*

Development choices (source code available at https://github.com/TheJena/RecommenderSystem)

- Shrink MongoDB size *(instructions in: **docs/how_to_create_test_db_dump.pdf**)*

- Modular Architecture

$$\begin{tikzcd}
 & \text{grank.py} \arrow[dr] & \\
\text{extract_dataset.py} \arrow[ur] \arrow[dr] & & \text{merge_results.py} \arrow[d] \\
 & \text{naïve_bayes.py} \arrow[ur] & \text{plot.py}
\end{tikzcd}$$

- Extract data from MongoDB once, then load a JSON file

Development choices (source code available at https://github.com/TheJena/RecommenderSystem)

- Shrink MongoDB size *(instructions in: **docs/how_to_create_test_db_dump.pdf**)*

- Modular Architecture

$$
\begin{tikzcd}
 & \text{grank.py} \arrow[dr] & & \\
\text{extract\_dataset.py} \arrow[ur] \arrow[dr] & & \text{merge\_results.py} \arrow[d] & \\
 & \text{naïve\_bayes.py} \arrow[ur] & \text{plot.py} &
\end{tikzcd}
$$

- Extract data from MongoDB once, then load a JSON file

- ~~NetworkX~~ → SciPy + generator–iterators[c] + multiprocessing[d]

[c] https://docs.python.org/3/glossary.html#term-generator-iterator
[d] https://docs.python.org/3/library/multiprocessing.html

Development choices (source code available at https://github.com/TheJena/RecommenderSystem)

- Shrink MongoDB size *(instructions in: **docs/how_to_create_test_db_dump.pdf**)*
- Modular Architecture

$$\text{extract\_dataset.py} \nearrow \text{grank.py} \searrow \quad \text{extract\_dataset.py} \searrow \text{naïve\_bayes.py} \nearrow \text{merge\_results.py} \downarrow \text{plot.py}$$

- Extract data from MongoDB once, then load a JSON file
- ~~NetworkX~~ → SciPy + generator–iterators [c] + multiprocessing [d]
- Natural Language Tool Kit + Sci Kit Learn + WordNet Expansion
- (in content-based) rating normalization, scaling (e.g. × 1 000) and rounding, fit, sort, use top–quartile [e] as a threshold to update confusion matrix

[c] https://docs.python.org/3/glossary.html#term-generator-iterator
[d] https://docs.python.org/3/library/multiprocessing.html
[e] *"Basu et al., 1998"*

Output Example

How to run, from a BASH shell:

```
./extract_dataset.py --category "Books" --category "Video Games" --output  dataset/M100_T30.json
                     --users     100    --reviews   30                                            --threads 8

./grank.py       dataset/M100_T30.json --stop-after 50 -k 10 --output results/grank.yaml       --parallel 8
./naive_bayes.py dataset/M100_T30.json --stop-after 50 -k 10 --output results/naive_bayes.yaml

./merge_results.py --content-based results/naive_bayes.yaml  --stop-after  1
                   --graph-based   results/grank.yaml        --top-k      10
```

Sample output:

```
================== Recommendations for user "A11L3YX5WIDKJ" ==================

                    naive bayes weight coefficient: 0.097       <~ NDCG@k
                          grank weight coefficient: 0.490       <~ NDCG@k

                 position |        item        | weighted rating
                ----------+--------------------+-----------------
                    1     |     B005FYJA52     |      0.430
                    2     |     0060788380     |      0.364
                    3     |     0007147295     |      0.335
                    4     |     B0038MTE7C     |      0.333
                    5     |     0006550436     |      0.331
                   ...    |     ... .. ...     |       ...
```

Conclusions

GRank

✓ NDCG similar to paper results

Multinomial Naïve Bayes

✗ worst NDCG performance

Conclusions

GRank

- ✓ NDCG similar to paper results
- ✓ higher F_1 score

Multinomial Naïve Bayes

- ✗ worst NDCG performance
- ✗ lower precision

Conclusions

GRank

- ✓ NDCG similar to paper results
- ✓ higher F_1 score
- ✗ slower (days)

Multinomial Naïve Bayes

- ✗ worst NDCG performance
- ✗ lower precision
- ✓ much faster (hours)

Conclusions

GRank

- ✓ NDCG similar to paper results
- ✓ higher F_1 score
- ✗ slower (days)
- ✗ scales $\propto N^2$ (n° items)

Multinomial Naïve Bayes

- ✗ worst NDCG performance
- ✗ lower precision
- ✓ much faster (hours)
- ✓ scales $\propto N$ (n° items)

Conclusions

GRank

- ✓ NDCG similar to paper results
- ✓ higher F_1 score
- ✗ slower (days)
- ✗ scales $\propto N^2$ (n° items)
- ✓ scales $\propto M$ (n° users)

Multinomial Naïve Bayes

- ✗ worst NDCG performance
- ✗ lower precision
- ✓ much faster (hours)
- ✓ scales $\propto N$ (n° items)

Conclusions

GRank

- ✓ NDCG similar to paper results
- ✓ higher F_1 score
- ✗ slower (days)
- ✗ scales $\propto N^2$ (n° items)
- ✓ scales $\propto M$ (n° users)
- ✓ scales $\propto S$ (n° user's preferences)

Multinomial Naïve Bayes

- ✗ worst NDCG performance
- ✗ lower precision
- ✓ much faster (hours)
- ✓ scales $\propto N$ (n° items)