Web Information Retrieval

Dipl.-Inf. Christoph Carl Kling
Exercises WebIR

ask questions!

WebIR@c-kling.de
Clustering Coefficient (≠ Transitivity ratio)

Definitions:

global clustering coefficient:

\[ C^g = \frac{\sum_{i=1}^{N} \# \text{connected friends of node } i}{\sum_{i=1}^{N} \# \text{possible connections between friends of node } i} \]

local clustering coefficient:

\[ C^l = \frac{1}{N} \sum_{i=1}^{N} \frac{\# \text{connected friends of node } i}{\# \text{possible connections between friends of node } i} \]
Clustering Coefficient (≠ Transitivity ratio)

Example

\[ C^{(1)} = \frac{3}{1+1+6} = \frac{3}{8} \]
Clustering Coefficient (≠ Transitivity ratio)

Example

\[ C^{(2)} = \frac{1}{5} (1 + 1 + 1/6) = \frac{13}{30} \]

\[ C^{(1)} = \frac{3}{8} \]

The two clustering coefficients give different measures.

\( C^{(2)} \) increases with nodes with low degree.
Document collection

d1: Marcus try assassin Caesar.
d2: Marcus Rome.
d3: Caesar rule. Rome loyal Caesar hate.
d4: loyal. People try assassin rule loyal.

Terms:
marcus, try, assassin, caesar, rome, rule, loyal, hate, people

TF-IDF vectors:

d1: (1,1,1,1,0,0,0,0,0,0)
d2: (1,0,0,0,1,0,0,0,0,0)
d3: (0,0,0,1,.5,.5,.5,1,0,0)
d4: (0,.5,.5,0,0,.5,1,0,1,0)

\[ tf(t, d) = \frac{f(t, d)}{\max\{f(w, d) : w \in d\}} \]

\[ idf_i = \log \frac{N}{n_i} \]

\[ w_{i,j} = tf_{i,j} \cdot idf_i \]
Exercise:

(Re)calculate the tf-idf scores for d1-d5. We use the log to base 2 (actually any log works).
Vector Space Model

Exercise:

Prove that cosine-similarity and Euclidean distance yield an identical relative distance measure (e.g. between documents) when all document and query vectors are normalised to be of length 1.
Vector Space Model

Exercise:

Prove that cosine-similarity and Eukldean distance yield an identical relative distance measure (e.g., between documents) when all document and query vectors are normalised to be of length 1.
Vector Space Model

\[ \vec{a} = (2, 3) \]

\[ |\vec{a}| = \sqrt{2 \cdot 2 + 3 \cdot 3} = \sqrt{9} = 3 \]

\[ \vec{a}_{norm} = \frac{\vec{a}}{|\vec{a}|} = \left( \frac{2}{\sqrt{9}}, \frac{3}{\sqrt{9}} \right) \]
Vector Space Model

\[
\cos(\vec{a}, \vec{b}) = \frac{a_x \cdot b_x + a_y \cdot b_y}{\sqrt{(a_x)^2 + (a_y)^2} \cdot \sqrt{(b_x)^2 + (b_y)^2}}
\]

\[\equiv a_x \cdot b_x + a_y \cdot b_y\]

\[
d(\vec{a}, \vec{b}) = \sqrt{(a_x - b_x)^2 + (a_y - b_y)^2}
\]

\[
= (a_x - b_x)^2 + (a_y - b_y)^2
\]

\[
= a_x^2 - 2(a_x \cdot b_x) + b_x^2 + a_y^2 - 2(a_y \cdot b_y) + b_y^2
\]

\[
= a_x^2 + a_y^2 + b_x^2 + b_y^2 - 2(a_x \cdot b_x + a_y \cdot b_y)
\]

\[
= 2 - 2(a_x \cdot b_x + a_y \cdot b_y) \propto -(a_x \cdot b_x + a_y \cdot b_y) \equiv -\cos(\vec{a}, \vec{b})
\]

* for normalised vectors of length 1
Given the search request

“Assassinate people”

The query is a binary vector again:

\[ q: (0,0,1,0,0,0,0,0,0,1) \]
**Vector Space Model**

\( q: (0,0,1,0,0,0,0,0,1) \)

This time, we don't know which docs are relevant. Instead, we look at which documents have both a high score for the terms “assassin” and “people”.

The relevance now is just the sum of the tf-idf weights for every word in the query.
Vector Space Model

The relevance now is just the sum of the tf-idf weights for every word in the query.

For every document, we have to sum over its tf-idf scores for the words in the query.

The tf-idf weights are stored in inverted indexes.
The tf-idf weights are stored in inverted indexes.

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>doc13: 0.3</td>
<td>doc99: 0.27</td>
<td>doc767: 0.7</td>
</tr>
<tr>
<td>doc4: 0.1</td>
<td>doc6: 0.3</td>
<td>doc41: 0.2</td>
</tr>
<tr>
<td>doc99: 0.4</td>
<td>doc13: 0.98</td>
<td>doc52: 0.5</td>
</tr>
<tr>
<td>doc767: 0.8</td>
<td>doc4: 0.23</td>
<td>doc4: 0.9</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Score: sum of tfidf weights for search terms.
For the query “assassin people”:
doc13 has score 1.28
doc4 has score 0.33
doc99 has score 0.67
...
Vector Space Model

The tf-idf weights are stored in inverted indexes.

Score: sum of tfidf weights for search terms.

This means, we have to do a join in a database, where we look up the score of every document for all query terms and calculate the sum.

Sounds bad.
To speed up the merging of tf-idf weights, documents can be ordered and we jump to “buckets” of document IDs when searching for an ID to join. (This is just what you did when searching a name in a phone book.)

**assassinate:** 2 4 9 16 59 66 128 135 291 311 315 591 672 899 ...

**people:** 1 2 3 5 8 17 21 35 39 46 52 66 75 88 ...

![Diagram Example]
DBS-Style Top-k Query Processing

Given: query \( q = t_1 \ t_2 \ldots \ t_z \) with \( z \) (conjunctive) keywords

similarity scoring function \( \text{score}(q,d) \) for docs \( d \in D \), e.g.: \( \text{score}(q,d) = \text{aggr}\{s_i(d)\}\) (e.g.: \( \Sigma_{i \in q} s_i(d) \))

Find: top \( k \) results w.r.t. \( \text{score}(q,d) \)

Naive join\&sort QP algorithm:

\[
\text{top-k} \left( \sigma[\text{term}=t_1] \ \text{(index)} \times \ \text{DocId} \right) \\
\times \ 
\sigma[\text{term}=t_2] \ \text{(index)} \times \ \text{DocId} \\
\ldots \\
\sigma[\text{term}=t_z] \ \text{(index)} \times \ \text{DocId} \\
\text{order by } s \text{ desc}
\]
The tf-idf weights are stored in inverted indexes. Our query is now: “people assassin caesar”
By ordering the documents by their tfidf-score, we can speed up the process of calculating the K documents with the highest sum of tf-idf scores:

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b: 0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
<tr>
<td>c: 0.35</td>
<td>f: 0.2</td>
<td>b: 0.2</td>
</tr>
<tr>
<td>a: 0.3</td>
<td>g: 0.2</td>
<td>a: 0.1</td>
</tr>
<tr>
<td>h: 0.1</td>
<td>c: 0.1</td>
<td>c: 0.05</td>
</tr>
<tr>
<td>d: 0.1</td>
<td>...</td>
<td>f: 0.05</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Threshold Algorithm (TA, Quick-Combine, MinPro)
(Fagin'01; Günther/Balke/Kießling; Nepal/Ramakrishna)

scan all lists \( L_i \) (i=1..m) in parallel:
consider \( dj \) at position \( pos_i \) in \( Li \);
\[ high_i := s_i(dj); \]
if \( dj \notin \text{top-k} \) then {
    look up \( s_v(dj) \) in all lists \( L_v \) with \( v \neq i \); // random access
    compute \( s(dj) := \text{aggr} \{ s_v(dj) \mid v=1..m \} ; \)
    if \( s(dj) > \text{min score among top-k} \) then
        add \( dj \) to top-k and remove min-score \( d \) from top-k;
    }
\[ \text{threshold} := \text{aggr} \{ high_v \mid v=1..m \}; \]
if \( \text{min score among top-k} \geq \text{threshold} \) then exit;

\[ m=3 \]
\[ \text{aggr: sum} \]
\[ k=2 \]

\[ \begin{array}{c}
\text{f: 0.5} \\
\text{b: 0.4} \\
\text{c: 0.35}
\end{array} \] \hspace{1cm}
\[ \begin{array}{c}
\text{a: 0.55} \\
\text{g: 0.2} \\
\text{e: 0.1}
\end{array} \] \hspace{1cm}
\[ \begin{array}{c}
\text{h: 0.35} \\
\text{d: 0.35} \\
\text{f: 0.2}
\end{array} \] \hspace{1cm}
\[ \begin{array}{c}
\text{a: 0.1} \\
\text{c: 0.05} \\
\text{f: 0.05}
\end{array} \] \hspace{1cm}
\[ \begin{array}{c}
\text{top-k:} \\
\text{f: 0.75} \\
\text{a: 0.95} \\
\text{b: 0.8}
\end{array} \]
## Vector Space Model

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b: 0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
<tr>
<td>c: 0.35</td>
<td>f: 0.2</td>
<td>b: 0.2</td>
</tr>
<tr>
<td>a: 0.3</td>
<td>g: 0.2</td>
<td>a: 0.1</td>
</tr>
<tr>
<td>h: 0.1</td>
<td>c: 0.1</td>
<td>c: 0.05</td>
</tr>
<tr>
<td>d: 0.1</td>
<td></td>
<td>f: 0.05</td>
</tr>
<tr>
<td>...</td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
### Vector Space Model

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b: 0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
<tr>
<td>c: 0.35</td>
<td>f: 0.2</td>
<td>b: 0.2</td>
</tr>
<tr>
<td>a: 0.3</td>
<td>g: 0.2</td>
<td>a: 0.1</td>
</tr>
<tr>
<td>h: 0.1</td>
<td>c: 0.1</td>
<td>c: 0.05</td>
</tr>
<tr>
<td>d: 0.1</td>
<td>...</td>
<td>f: 0.05</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
## Vector Space Model

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b: 0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
<tr>
<td>c: 0.35</td>
<td>f: 0.2</td>
<td>b: 0.2</td>
</tr>
<tr>
<td>a: 0.3</td>
<td>g: 0.2</td>
<td>a: 0.1</td>
</tr>
<tr>
<td>h: 0.1</td>
<td>c: 0.1</td>
<td>c: 0.05</td>
</tr>
<tr>
<td>d: 0.1</td>
<td>...</td>
<td>f: 0.05</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
### Vector Space Model

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b: 0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
<tr>
<td>c: 0.35</td>
<td>f: 0.2</td>
<td>b: 0.2</td>
</tr>
<tr>
<td>a: 0.3</td>
<td>g: 0.2</td>
<td>a: 0.1</td>
</tr>
<tr>
<td>h: 0.1</td>
<td>c: 0.1</td>
<td>c: 0.05</td>
</tr>
<tr>
<td>d: 0.1</td>
<td>...</td>
<td>f: 0.05</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
scan index lists in parallel:
consider \( dj \) at position \( pos_i \) in \( Li \);
\( E(dj) := E(dj) \cup \{i\} \); \( high_i := si(q,dj) \);
\( bestscore(dj) := aggr\{x_1, \ldots, x_m\} \)
with \( x_i := si(q,dj) \) for \( i \in E(dj) \), \( high_i \) for \( i \notin E(dj) \);
\( worstscore(dj) := aggr\{x_1, \ldots, x_m\} \)
with \( x_i := si(q,dj) \) for \( i \in E(dj) \), 0 for \( i \notin E(dj) \);
top-k := k docs with largest worstscore;
threshold := bestscore\{d | d not in top-k\};
if min worstscore among top-k \( \geq \) threshold then exit;

\( m=3 \)
aggr: sum
\( k=2 \)

\begin{align*}
m & = 3 \\
\text{aggr: sum} \\
k & = 2 \\
f & : 0.5 \\
b & : 0.4 \\
c & : 0.38 \\
a & : 0.3 \\
h & : 0.1 \\
d & : 0.1 \\
a & : 0.55 \\
b & : 0.2 \\
c & : 0.38 \\
a & : 0.3 \\
h & : 0.1 \\
d & : 0.1 \\
h & : 0.35 \\
d & : 0.35 \\
b & : 0.2 \\
\text{candidates:} \\
f & : 0.7 + ? \leq 0.7 + 0.1 \\
h & : 0.35 + ? \leq 0.35 + 0.5 \\
c & : 0.35 + ? \leq 0.35 + 0.3 \\
d & : 0.35 + ? \leq 0.35 + 0.5 \\
g & : 0.2 + ? \leq 0.2 + 0.4 \\
a & : 0.95 \\
b & : 0.8 \\
h & : 0.35 \\
d & : 0.35 \\
b & : 0.2 \\
a & : 0.1 \\
c & : 0.05 \\
f & : 0.05 \\
\end{align*}
Vector Space Model

Non-random-access algorithm:
We only look at the first i rows.

<table>
<thead>
<tr>
<th></th>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f</td>
<td>0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b</td>
<td>0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
<tr>
<td>c</td>
<td>0.35</td>
<td>f: 0.2</td>
<td>b: 0.2</td>
</tr>
<tr>
<td>a</td>
<td>0.3</td>
<td>g: 0.2</td>
<td>a: 0.1</td>
</tr>
<tr>
<td>h</td>
<td>0.1</td>
<td>c: 0.1</td>
<td>c: 0.05</td>
</tr>
<tr>
<td>d</td>
<td>0.1</td>
<td>...</td>
<td>f: 0.05</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
<td>...</td>
</tr>
</tbody>
</table>
### Vector Space Model

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
</tbody>
</table>

Top 2 documents:

Threshold:

Candidates:
<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
</tbody>
</table>

**Top 2 documents:**
- a: 0.55
- f: 0.5

**Threshold:**
- 1.4

**Candidates:**
- h: 0.35
**Vector Space Model**

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b: 0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
</tbody>
</table>

**Top 2 documents:**
- a: 0.55
- f: 0.5

**Threshold:**
- 1.4

**Candidates:**
- h: 0.35

...
### Vector Space Model

<table>
<thead>
<tr>
<th></th>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f:</td>
<td>0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b:</td>
<td>0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
</tbody>
</table>

**Top 2 documents:**
- b: 0.6
- a: 0.55

**Threshold:**
- 0.95

**Candidates:**
- f: 0.5
- h: 0.35
- d: 0.35
### Vector Space Model

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b: 0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
<tr>
<td>c: 0.35</td>
<td>f: 0.2</td>
<td>b: 0.2</td>
</tr>
</tbody>
</table>

... 

**Top 2 documents:**
- b: 0.6
- a: 0.55

**Threshold:** 0.95

**Candidates:**
- f: 0.5
- h: 0.35
- d: 0.35
**Vector Space Model**

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b: 0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
<tr>
<td>c: 0.35</td>
<td>f: 0.2</td>
<td>b: 0.2</td>
</tr>
</tbody>
</table>

...  

**Top 2 documents:**
- b: 0.8  
- f: 0.7  

**Threshold:**
- 0.75  

**Candidates:**
- a: 0.55  
- h: 0.35  
- d: 0.35  
- c: 0.35
### Vector Space Model

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b: 0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
<tr>
<td>c: 0.35</td>
<td>f: 0.2</td>
<td>b: 0.2</td>
</tr>
<tr>
<td>a: 0.3</td>
<td>g: 0.2</td>
<td>a: 0.1</td>
</tr>
</tbody>
</table>

...  

**Top 2 documents:**

- b: 0.8
- f: 0.7

**Threshold:**

- 0.75

**Candidates:**

- a: 0.55
- h: 0.35
- d: 0.35
- c: 0.35
### Vector Space Model

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>ceasar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b: 0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
<tr>
<td>c: 0.35</td>
<td>f: 0.2</td>
<td>b: 0.2</td>
</tr>
<tr>
<td>a: 0.3</td>
<td>g: 0.2</td>
<td>a: 0.1</td>
</tr>
</tbody>
</table>

... 

**Top 2 documents:**
- a: 0.95
- b: 0.8

**Candidates:**
- f: 0.7
- h: 0.35
- d: 0.35
- c: 0.35
- g: 0.2

**Threshold:**
- 0.6
## Vector Space Model

<table>
<thead>
<tr>
<th>assassin</th>
<th>people</th>
<th>caesar</th>
</tr>
</thead>
<tbody>
<tr>
<td>f: 0.5</td>
<td>a: 0.55</td>
<td>h: 0.35</td>
</tr>
<tr>
<td>b: 0.4</td>
<td>b: 0.2</td>
<td>d: 0.35</td>
</tr>
<tr>
<td>c: 0.35</td>
<td>f: 0.2</td>
<td>b: 0.2</td>
</tr>
<tr>
<td>a: 0.3</td>
<td>g: 0.2</td>
<td>a: 0.1</td>
</tr>
</tbody>
</table>

**Top 2 documents:**

- a: 0.95
- b: 0.8

**Threshold:**

- 0.6

**Candidates:**

- f: 0.7
- h: 0.35
- d: 0.35
- c: 0.35
- g: 0.2

0.8 (min score in top-K) is bigger than the threshold of 0.6!

We found the top-K documents!
Markov chain:

We are interested in ergodic Markov chains:

- homogeneous (transition probabilities fixed)
- irreducible (every state always reachable)
- aperiodic (greatest common divisor = 1 for recurrence)
Page Rank

Transition probability: 1/number of links
Exercise:

Is this Markov chain ergodic?
Random jump with $e = 0.3$
( normal jump with $(1-e) = 0.7$ )
→ the markov chain becomes ergodic!
Page Rank

Transition matrix $P$ (see board)

Initial state $\alpha = (0.25, 0.25, 0.25, 0.25)$
Exercise:

Calculate

\[ a_1 = a_0 P \]
\[ a_2 = a_1 P \]
\[ \quad \quad \quad \cdots \]
\[ a_6 = a_5 P \]

Will a converge? And why?
Markov Chain

Authority scores \( x = (0.25, 0.25, 0.25, 0.25) \)

Hub scores \( y = (0.25, 0.25, 0.25, 0.25) \)
Hubs and Authorities (or HITS) algorithm

Authority scores $x_1 = (0.25, 0.25, 0.25, 0.25)$

Hub scores $y_1 = (0.25, 0.25, 0.25, 0.25)$
Hubs and Authorities (or HITS) algorithm

Authority scores $x_2 = (0.5, 0.5, 0.5, 0)$

Hub scores $y_2 = (1.0, 0.5, 1.0, 0.5)$

These vectors have to be normalised to sum to 1!
Hubs and Authorities (or HITS) algorithm

Authority scores \( x_2 = (0.333, 0.333, 0.333, 0) \)

Hub scores \( y_2 = (0.333, 0.166, 0.333, 0.166) \)
Exercise:

Calculate

\[ x_2, y_2, \ldots x_4, y_4 \]

and normalise \( x \) and \( y \) after each step so that each vector sums to 1 (that is, the values can be interpreted as probabilities).
Exercise:

“Hubs and Authorities” - unlike Page Rank - does not take the whole web graph, but only a subset. Describe how the Hubs and Authority algorithm works!
Thank you!