An evaluation of string similarity measures on pricelists of computer components

R. Jiroušek, V. Kratochvíl, T. Kroupa, R. Lněnička,
M. Studený, J. Vomlel, P. Hampl, and H. Hamplová

Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic (AV ČR)

Empo, s.r.o., Praha

Liblice, September 15–18, 2007

Matching euivalent components from pricelists

Definition

The task is to find a computer component described by partially structured text in different pricelists of computer components.

Matching euivalent components from pricelists

Definition

The task is to find a computer component described by partially structured text in different pricelists of computer components.

Example (1)

```
category IS printers OR category IS UNKNOWN
AND producer IS hp OR producer IS UNKNOWN
description IS SIMILAR TO
Toner Cartridge pro LJ4/M/+/4M+/5/5M/5N 92298X
```

Matching euivalent components from pricelists

Definition

The task is to find a computer component described by partially structured text in different pricelists of computer components.

Example (1)

```
category IS printers OR category IS UNKNOWN
AND producer IS hp OR producer IS UNKNOWN
description IS SIMILAR TO
Toner Cartridge pro LJ4/M/+/4M+/5/5M/5N 92298X

Toner pro LaserJet 4/4M, 4/4M Plus, 5/5N/5M (8800)
```

Matching euivalent components from pricelists

Definition

The task is to find a computer component described by partially structured text in different pricelists of computer components.

Example (2)

```
category IS accesories OR category IS UNKNOWN
AND producer IS logitech OR producer IS UNKNOWN
description IS SIMILAR TO
Pilot Optical Mouse, USB+PS/2, 3 tlačítka, černá
```

Matching euivalent components from pricelists

Definition

The task is to find a computer component described by partially structured text in different pricelists of computer components.

Example (2)

```
category IS accesories OR category IS UNKNOWN
AND producer IS logitech OR producer IS UNKNOWN
description IS SIMILAR TO
Pilot Optical Mouse, USB+PS/2, 3 tlačítka, černá

Logitech myš Pilot Optical Mouse Black, USB/PS/2, retail
```

Problem description

- We have pricelists of computer components from seven different resellers - some with more than 30,000 components.

Problem description

- We have pricelists of computer components from seven different resellers - some with more than 30,000 components.
- Most pricelists are partially structured, with `producer`, `product category`, `price`, and `product description`.

Problem description

- We have pricelists of computer components from seven different resellers - some with more than 30,000 components.
- Most pricelists are partially structured, with `producer`, `product category`, `price`, and `product description`.
- We use one additional category of unclassified components.

Problem description

- We have pricelists of computer components from seven different resellers - some with more than 30,000 components.
- Most pricelists are partially structured, with `producer`, `product category`, `price`, and `product description`.
- We use one additional category of unclassified components.
- Some suppliers provide also `part number` for some components. It should be unique.

Problem description

- We have pricelists of computer components from seven different resellers - some with more than 30,000 components.
- Most pricelists are partially structured, with `producer`, `product category`, `price`, and `product description`.
- We use one additional category of unclassified components.
- Some suppliers provide also `part number` for some components. It should be unique.
- Part numbers provide a very reliable matching.

Problem description

- We have pricelists of computer components from seven different resellers - some with more than 30,000 components.
- Most pricelists are partially structured, with `producer`, `product category`, `price`, and `product description`.
- We use one additional category of unclassified components.
- Some suppliers provide also `part number` for some components. It should be unique.
- Part numbers provide a very reliable matching.
- Unfortunatelly, many items in pricelists do not have any part number assigned.

A fulltext search method

- As a reference method we used the fulltext search of MySQL: `http://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html`

A fulltext search method

- As a reference method we used the fulltext search of MySQL: `http://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html`
- The search string is treated as a phrase in free text.

A fulltext search method

- As a reference method we used the fulltext search of MySQL: `http://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html`
- The search string is treated as a phrase in free text.
- The MySQL stopword list was applied.

A fulltext search method

- As a reference method we used the fulltext search of MySQL: `http://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html`
- The search string is treated as a phrase in free text.
- The MySQL stopword list was applied.
- Words present in more than 50% of the records were considered common and were not matched.

A fulltext search method

- As a reference method we used the fulltext search of MySQL: `http://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html`
- The search string is treated as a phrase in free text.
- The MySQL stopword list was applied.
- Words present in more than 50% of the records were considered common and were not matched.
- Also words shorter than four characters were not matched.

A fulltext search method

- As a reference method we used the fulltext search of MySQL: `http://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html`
- The search string is treated as a phrase in free text.
- The MySQL stopword list was applied.
- Words present in more than 50% of the records were considered common and were not matched.
- Also words shorter than four characters were not matched.
- We denote the similarity value of two strings S_1 and S_2 provided by this fulltext search method as $Sim_1(S_1, S_2)$.

A string edit distance measure

- This method is described in detail in our previous paper on this topic, which is part of the proceedings of the Eighth Czech-Japan Seminar in 2005.

A string edit distance measure

- This method is described in detail in our previous paper on this topic, which is part of the proceedings of the Eighth Czech-Japan Seminar in 2005.
- We measure the similarity $Sim(S_1, S_2)$ of two strings S_1, S_2 by the total length of substrings of S_1 that are substrings of string S_2.

A string edit distance measure

- This method is described in detail in our previous paper on this topic, which is part of the proceedings of the Eighth Czech-Japan Seminar in 2005.
- We measure the similarity $Sim(S_1, S_2)$ of two strings S_1, S_2 by the total length of substrings of S_1 that are substrings of string S_2.
- We do not require the substrings of S_1 to be disjoint, which means that parts of substrings of S_1 longer than two are counted several times.

A string edit distance measure

- This method is described in detail in our previous paper on this topic, which is part of the proceedings of the Eighth Czech-Japan Seminar in 2005.
- We measure the similarity $Sim(S_1, S_2)$ of two strings S_1, S_2 by the total length of substrings of S_1 that are substrings of string S_2.
- We do not require the substrings of S_1 to be disjoint, which means that parts of substrings of S_1 longer than two are counted several times.
- In the experiments we used the relative string similarity defined as

$$Sim_2(S_1, S_2) = \frac{Sim(S_1, S_2)}{Sim(S_1, S_1)}$$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S		T	E	R	M

R_2

W	I	N		T	R	M	N	L

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S		T	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 1$

$Similarity(R_1, R_2) = 0$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S		T	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 1$

$R = "WI"$

$Length(R) = 2$

$Similarity(R_1, R_2) = 2$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S		T	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 1$

$R =$ "WIN"

$Length(R) = 3$

$Similarity(R_1, R_2) = 2 + 3$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S		T	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 2$

$Similarity(R_1, R_2) = 2 + 3$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S		T	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 2$

$R = "IN"$

$Length(R) = 2$

$Similarity(R_1, R_2) = 2 + 3 + 2$

The string edit distance measure

Example

R_1

W	I	**N**	D	O	W	S		T	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 3$

$Similarity(R_1, R_2) = 2 + 3 + 2$

The string edit distance measure

Example

R_1

W	I	N	**D**	O	W	S		T	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 4$

$Similarity(R_1, R_2) = 2 + 3 + 2$

The string edit distance measure

Example

R_1

W	I	N	D	**O**	W	S		T	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 5$

$Similarity(R_1, R_2) = 2 + 3 + 2$

The string edit distance measure

Example

R_1

W	I	N	D	O	**W**	S		T	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 6$

$Similarity(R_1, R_2) = 2 + 3 + 2$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	**S**		T	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 7$

$Similarity(R_1, R_2) = 2 + 3 + 2$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S	_	T	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 8$

$$Similarity(R_1, R_2) = 2 + 3 + 2$$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S	_	T	E	R	M

R_2

W	I	N	_	T	R	M	N	L

$k = 8$

$R ='' \ \text{T}''$

$Length(R) = 2$

$Similarity(R_1, R_2) = 2 + 3 + 2 + 2$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S		**T**	E	R	M

R_2

W	I	N		T	R	M	N	L

$k = 9$

$$Similarity(R_1, R_2) = 2 + 3 + 2 + 2$$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S		T	**E**	R	M

R_2

W	I	N		T	R	M	N	L

$k = 10$

$$Similarity(R_1, R_2) = 2 + 3 + 2 + 2$$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S		T	E	**R**	M

R_2

W	I	N		T	R	M	N	L

$k = 11$

$Similarity(R_1, R_2) = 2 + 3 + 2 + 2$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S		T	E	**R**	M

R_2

W	I	N		T	R	M	N	L

$k = 11$

$R =$ "RM"

$Length(R) = 2$

$Similarity(R_1, R_2) = 2 + 3 + 2 + 2 + 2$

The string edit distance measure

Example

R_1

W	I	N	D	O	W	S		T	E	R	M

R_2

W	I	N		T	R	M	N	L

$$Similarity(R_1, R_2) = 2 + 3 + 2 + 2 + 2 = 11$$

A vector based method

- Every string is encoded as a vector of real numbers whose components are formed by weights of individual *tokens* (groups of characters) presented in the string.

A vector based method

- Every string is encoded as a vector of real numbers whose components are formed by weights of individual *tokens* (groups of characters) presented in the string.
- The string is divided into tokens by special characters - tokens separators (e.g., space, comma, semicolon, etc.)

A vector based method

- Every string is encoded as a vector of real numbers whose components are formed by weights of individual *tokens* (groups of characters) presented in the string.
- The string is divided into tokens by special characters - tokens separators (e.g., space, comma, semicolon, etc.)
- A popular method for computing the weights is the TF-IDF method.

A vector based method

- Every string is encoded as a vector of real numbers whose components are formed by weights of individual *tokens* (groups of characters) presented in the string.
- The string is divided into tokens by special characters - tokens separators (e.g., space, comma, semicolon, etc.)
- A popular method for computing the weights is the TF-IDF method.
- Let $n(x, S)$ be the number of occurrences of token x in string S (often, it is 0 and 1),

A vector based method

- Every string is encoded as a vector of real numbers whose components are formed by weights of individual *tokens* (groups of characters) presented in the string.
- The string is divided into tokens by special characters - tokens separators (e.g., space, comma, semicolon, etc.)
- A popular method for computing the weights is the TF-IDF method.
- Let $n(x, S)$ be the number of occurrences of token x in string S (often, it is 0 and 1),
- $n(S)$ be the total number of tokens in string S,

A vector based method

- Every string is encoded as a vector of real numbers whose components are formed by weights of individual *tokens* (groups of characters) presented in the string.
- The string is divided into tokens by special characters - tokens separators (e.g., space, comma, semicolon, etc.)
- A popular method for computing the weights is the TF-IDF method.
- Let $n(x, S)$ be the number of occurrences of token x in string S (often, it is 0 and 1),
- $n(S)$ be the total number of tokens in string S,
- m be the total number of all strings in the data, and

A vector based method

- Every string is encoded as a vector of real numbers whose components are formed by weights of individual *tokens* (groups of characters) presented in the string.
- The string is divided into tokens by special characters - tokens separators (e.g., space, comma, semicolon, etc.)
- A popular method for computing the weights is the TF-IDF method.
- Let $n(x, S)$ be the number of occurrences of token x in string S (often, it is 0 and 1),
- $n(S)$ be the total number of tokens in string S,
- m be the total number of all strings in the data, and
- $m(x)$ be the number of strings containing token x.

A vector based method

- Every string is encoded as a vector of real numbers whose components are formed by weights of individual *tokens* (groups of characters) presented in the string.
- The string is divided into tokens by special characters - tokens separators (e.g., space, comma, semicolon, etc.)
- A popular method for computing the weights is the TF-IDF method.
- Let $n(x, S)$ be the number of occurrences of token x in string S (often, it is 0 and 1),
- $n(S)$ be the total number of tokens in string S,
- m be the total number of all strings in the data, and
- $m(x)$ be the number of strings containing token x.
- The weight of a token x in string S is defined as

$$w(x, S) = \frac{n(x, S)}{n(S)} \log \frac{m}{m(x)} .$$

A vector based method

- Let d be the total number of different tokens in the entire data.

A vector based method

- Let d be the total number of different tokens in the entire data.
- Then $\mathbf{w}(S) = (w(x_1, S), \dots w(x_d, S))^T$ is a vector that characterizes string S.

A vector based method

- Let d be the total number of different tokens in the entire data.
- Then $\mathbf{w}(S) = (w(x_1, S), \ldots w(x_d, S))^T$ is a vector that characterizes string S.
- By $\mathbf{v}(S)$ we will denote the normalized weight vector

$$\mathbf{v}(S) = \frac{\mathbf{w}(S)}{\sqrt{\sum_{i=1}^{d} w(x_i, S)^2}}$$

A vector based method

- Let d be the total number of different tokens in the entire data.
- Then $\mathbf{w}(S) = (w(x_1, S), \ldots w(x_d, S))^T$ is a vector that characterizes string S.
- By $\mathbf{v}(S)$ we will denote the normalized weight vector

$$\mathbf{v}(S) \quad = \quad \frac{\mathbf{w}(S)}{\sqrt{\sum_{i=1}^{d} w(x_i, S)^2}}$$

- Similarity of two strings S_1 and S_2 is then computed as the scalar product of normalized weight vectors $\mathbf{v}(S_1)$ and $\mathbf{v}(S_2)$

$$Sim_3(S_1, S_2) \quad = \quad \sum_{i=1}^{d} v(x_i, S_1) \cdot v(x_i, S_2) \quad = \quad \mathbf{v}(S_1)^T \cdot \mathbf{v}(S_2) \ .$$

A vector based method

- Let d be the total number of different tokens in the entire data.
- Then $\mathbf{w}(S) = (w(x_1, S), \ldots w(x_d, S))^T$ is a vector that characterizes string S.
- By $\mathbf{v}(S)$ we will denote the normalized weight vector

$$\mathbf{v}(S) = \frac{\mathbf{w}(S)}{\sqrt{\sum_{i=1}^{d} w(x_i, S)^2}}$$

- Similarity of two strings S_1 and S_2 is then computed as the scalar product of normalized weight vectors $\mathbf{v}(S_1)$ and $\mathbf{v}(S_2)$

$$Sim_3(S_1, S_2) = \sum_{i=1}^{d} v(x_i, S_1) \cdot v(x_i, S_2) = \mathbf{v}(S_1)^T \cdot \mathbf{v}(S_2) .$$

- Note that since both vectors are sparse the computation of the scalar product can be efficiently implemented.

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

- For simplicity assume tokens from these two strings only:

  `toner, magenta, pro, clp, 510, 510n, az, 5000, stran, samsung, clp510, n, 5000str`

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

- For simplicity assume tokens from these two strings only:

  `toner, magenta, pro, clp, 510, 510n, az, 5000, stran, samsung, clp510, n, 5000str`

- $w(\texttt{toner}, S_1) = \frac{1}{9} \log \frac{36478}{274} = 0.236$

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

- For simplicity assume tokens from these two strings only:

  `toner, magenta, pro, clp, 510, 510n, az, 5000, stran, samsung, clp510, n, 5000str`

- $w(\texttt{toner}, S_1) = \frac{1}{9} \log \frac{36478}{274} = 0.236$
- $w(\texttt{toner}, S_2) = \frac{1}{7} \log \frac{36478}{274} = 0.303$

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

- For simplicity assume tokens from these two strings only:

  `toner, magenta, pro, clp, 510, 510n, az, 5000, stran, samsung, clp510, n, 5000str`
- $w(\texttt{toner}, S_1) = \frac{1}{9} \log \frac{36478}{274} = 0.236$
- $w(\texttt{toner}, S_2) = \frac{1}{7} \log \frac{36478}{274} = 0.303$
- $w(\texttt{magenta}, S_1) = \frac{1}{9} \log \frac{36478}{59} = 0.310$

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

- For simplicity assume tokens from these two strings only:

  `toner, magenta, pro, clp, 510, 510n, az, 5000, stran, samsung, clp510, n, 5000str`

- $w(\texttt{toner}, S_1) = \frac{1}{9} \log \frac{36478}{274} = 0.236$
- $w(\texttt{toner}, S_2) = \frac{1}{7} \log \frac{36478}{274} = 0.303$
- $w(\texttt{magenta}, S_1) = \frac{1}{9} \log \frac{36478}{59} = 0.310$
- $w(\texttt{magenta}, S_2) = \frac{1}{7} \log \frac{36478}{59} = 0.399$

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

- For simplicity assume tokens from these two strings only:

  `toner, magenta, pro, clp, 510, 510n, az, 5000, stran, samsung, clp510, n, 5000str`

- $w(\texttt{toner}, S_1) = \frac{1}{9} \log \frac{36478}{274} = 0.236$
- $w(\texttt{toner}, S_2) = \frac{1}{7} \log \frac{36478}{274} = 0.303$
- $w(\texttt{magenta}, S_1) = \frac{1}{9} \log \frac{36478}{59} = 0.310$
- $w(\texttt{magenta}, S_2) = \frac{1}{7} \log \frac{36478}{59} = 0.399$
- $\mathbf{w}(S_1) = (0.236, 0.310, 0.285, 0.420, 0.235, 0.345, 0.034, 0.121, 0.097, 0.000, 0.000, 0.000, 0.000)$

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

- For simplicity assume tokens from these two strings only:

  `toner, magenta, pro, clp, 510, 510n, az, 5000, stran, samsung, clp510, n, 5000str`
- $w(\texttt{toner}, S_1) = \frac{1}{9} \log \frac{36478}{274} = 0.236$
- $w(\texttt{toner}, S_2) = \frac{1}{7} \log \frac{36478}{274} = 0.303$
- $w(\texttt{magenta}, S_1) = \frac{1}{9} \log \frac{36478}{59} = 0.310$
- $w(\texttt{magenta}, S_2) = \frac{1}{7} \log \frac{36478}{59} = 0.399$
- $\mathbf{w}(S_1) =$ (0.236, 0.310, 0.285, 0.420, 0.235, 0.345, 0.034, 0.121, 0.097, 0.000, 0.000, 0.000, 0.000)
- $\mathbf{w}(S_2) =$ (0.303, 0.399, 0.366, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.056, 0.451, 0.023, 0.456)

The vector based method

Example

S_1 toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran

S_2 samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)

- $\mathbf{v}(S_1) = \frac{\mathbf{w}(S_1)}{\sqrt{\sum_{i=1}^{d} w(x_i, S_1)^2}} = \frac{\mathbf{w}(S_1)}{0.780}$

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

- $\mathbf{v}(S_1) = \frac{\mathbf{w}(S_1)}{\sqrt{\sum_{i=1}^{d} w(x_i, S_1)^2}} = \frac{\mathbf{w}(S_1)}{0.780}$
- $\mathbf{v}(S_2) = \frac{\mathbf{w}(S_2)}{\sqrt{\sum_{i=1}^{d} w(x_i, S_2)^2}} = \frac{\mathbf{w}(S_2)}{0.794}$

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

- $\mathbf{v}(S_1) = \frac{\mathbf{w}(S_1)}{\sqrt{\sum_{i=1}^{d} w(x_i, S_1)^2}} = \frac{\mathbf{w}(S_1)}{0.780}$
- $\mathbf{v}(S_2) = \frac{\mathbf{w}(S_2)}{\sqrt{\sum_{i=1}^{d} w(x_i, S_2)^2}} = \frac{\mathbf{w}(S_2)}{0.794}$
- $\mathbf{v}(S_1) = (0.302, 0.397, 0.365, 0.538, 0.301, 0.442, 0.044, 0.155, 0.124, 0.000, 0.000, 0.000, 0.000)$

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

- $\mathbf{v}(S_1) = \dfrac{\mathbf{w}(S_1)}{\sqrt{\sum_{i=1}^{d} w(x_i, S_1)^2}} = \dfrac{\mathbf{w}(S_1)}{0.780}$
- $\mathbf{v}(S_2) = \dfrac{\mathbf{w}(S_2)}{\sqrt{\sum_{i=1}^{d} w(x_i, S_2)^2}} = \dfrac{\mathbf{w}(S_2)}{0.794}$
- $\mathbf{v}(S_1) = (0.302, 0.397, 0.365, 0.538, 0.301, 0.442, 0.044, 0.155, 0.124, 0.000, 0.000, 0.000, 0.000)$
- $\mathbf{v}(S_2) = (0.339, 0.446, 0.409, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.063, 0.504, 0.026, 0.510)$

The vector based method

Example

S_1 `toner␣magenta␣pro␣clp-510/510n,␣az␣5000␣stran`

S_2 `samsung␣toner␣magenta␣pro␣clp510/n␣(5000str␣)`

- $\mathbf{v}(S_1) = \frac{\mathbf{w}(S_1)}{\sqrt{\sum_{i=1}^{d} w(x_i, S_1)^2}} = \frac{\mathbf{w}(S_1)}{0.780}$
- $\mathbf{v}(S_2) = \frac{\mathbf{w}(S_2)}{\sqrt{\sum_{i=1}^{d} w(x_i, S_2)^2}} = \frac{\mathbf{w}(S_2)}{0.794}$
- $\mathbf{v}(S_1) = (0.302, 0.397, 0.365, 0.538, 0.301, 0.442, 0.044, 0.155, 0.124, 0.000, 0.000, 0.000, 0.000)$
- $\mathbf{v}(S_2) = (0.339, 0.446, 0.409, 0.000, 0.000, 0.000, 0.000, 0.000, 0.000, 0.063, 0.504, 0.026, 0.510)$

$$\begin{aligned} Sim_3(S_1, S_2) &= \mathbf{v}(S_1)^T \cdot \mathbf{v}(S_2) \\ &= 0.302 \cdot 0.339 + 0.397 \cdot 0.446 + 0.365 \cdot 0.409 = 0.429 \end{aligned}$$

A linear combination of methods

- Each method uses a different approach for finding equivalent components.

A linear combination of methods

- Each method uses a different approach for finding equivalent components.
- Therefore one can hope that their combination can provide better results.

A linear combination of methods

- Each method uses a different approach for finding equivalent components.
- Therefore one can hope that their combination can provide better results.
- We have tested linear combinations of

A linear combination of methods

- Each method uses a different approach for finding equivalent components.
- Therefore one can hope that their combination can provide better results.
- We have tested linear combinations of
    - the fulltext search Sim_1,

A linear combination of methods

- Each method uses a different approach for finding equivalent components.
- Therefore one can hope that their combination can provide better results.
- We have tested linear combinations of
  - the fulltext search Sim_1,
  - string similarity Sim_2, and

A linear combination of methods

- Each method uses a different approach for finding equivalent components.
- Therefore one can hope that their combination can provide better results.
- We have tested linear combinations of
  - the fulltext search Sim_1,
  - string similarity Sim_2, and
  - the vector based method Sim_3

A linear combination of methods

- Each method uses a different approach for finding equivalent components.
- Therefore one can hope that their combination can provide better results.
- We have tested linear combinations of
  - the fulltext search Sim_1,
  - string similarity Sim_2, and
  - the vector based method Sim_3

$$Sim_4(S_1, S_2) = c_1 \cdot Sim_1(S_1, S_2) + c_2 \cdot Sim_2(S_1, S_2) + c_3 \cdot Sim_3(S_1, S_2)$$

A linear combination of methods

- Each method uses a different approach for finding equivalent components.
- Therefore one can hope that their combination can provide better results.
- We have tested linear combinations of
  - the fulltext search Sim_1,
  - string similarity Sim_2, and
  - the vector based method Sim_3

$$Sim_4(S_1, S_2) = c_1 \cdot Sim_1(S_1, S_2) + c_2 \cdot Sim_2(S_1, S_2) + c_3 \cdot Sim_3(S_1, S_2)$$

where $\mathbf{c} = (c_1, c_2, c_3)$ was set to $(0.3, 1, 1)$, $(0, 1, 1)$, and $(0, 1, 2)$.

Experiments

- We selected two pricelists of computer components from two different suppliers.

Experiments

- We selected two pricelists of computer components from two different suppliers.
- They contained together 64566 components.

Experiments

- We selected two pricelists of computer components from two different suppliers.
- They contained together 64566 components.
- From these two pricelists we selected only those components that were given a part number in both pricelists - we have got 7060 different part numbers.

Experiments

- We selected two pricelists of computer components from two different suppliers.
- They contained together 64566 components.
- From these two pricelists we selected only those components that were given a part number in both pricelists - we have got 7060 different part numbers.
- From these we randomly selected 500 part numbers.

Experiments

- We selected two pricelists of computer components from two different suppliers.
- They contained together 64566 components.
- From these two pricelists we selected only those components that were given a part number in both pricelists - we have got 7060 different part numbers.
- From these we randomly selected 500 part numbers.
- These part numbers defined our test pairs of components.

Experiments

- We selected two pricelists of computer components from two different suppliers.
- They contained together 64566 components.
- From these two pricelists we selected only those components that were given a part number in both pricelists - we have got 7060 different part numbers.
- From these we randomly selected 500 part numbers.
- These part numbers defined our test pairs of components.
- For each of 500 components from the first pricelist we used the tested methods to find k ($k = 1, 2, \ldots, 15$) most similar components in the (complete) second pricelist.

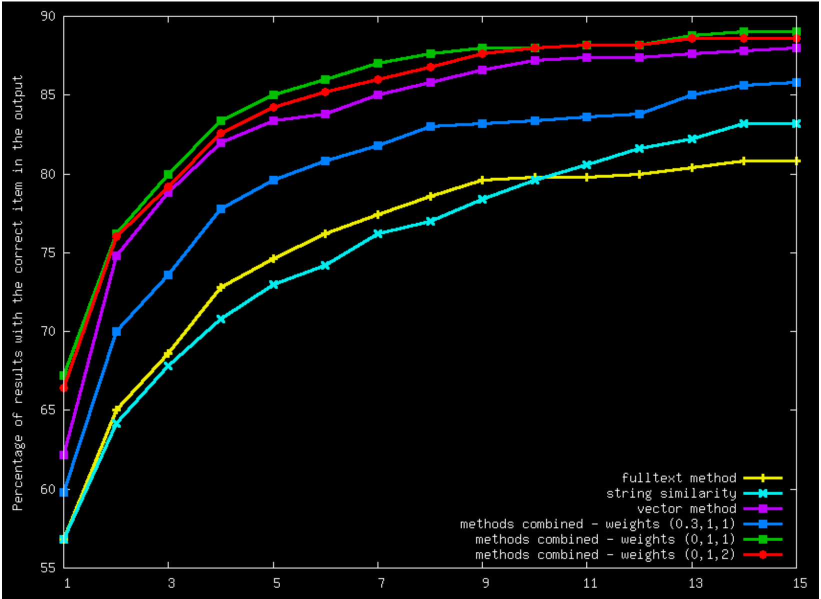
Experiments

- We selected two pricelists of computer components from two different suppliers.
- They contained together 64566 components.
- From these two pricelists we selected only those components that were given a part number in both pricelists - we have got 7060 different part numbers.
- From these we randomly selected 500 part numbers.
- These part numbers defined our test pairs of components.
- For each of 500 components from the first pricelist we used the tested methods to find k ($k = 1, 2, \ldots, 15$) most similar components in the (complete) second pricelist.
- Then we checked whether the component with the same part number is among those k selected ones.

Experiments

- We selected two pricelists of computer components from two different suppliers.
- They contained together 64566 components.
- From these two pricelists we selected only those components that were given a part number in both pricelists - we have got 7060 different part numbers.
- From these we randomly selected 500 part numbers.
- These part numbers defined our test pairs of components.
- For each of 500 components from the first pricelist we used the tested methods to find k ($k = 1, 2, \ldots, 15$) most similar components in the (complete) second pricelist.
- Then we checked whether the component with the same part number is among those k selected ones.
- We counted the number of these cases and computed the relative success rate for each method with respect to k.

Results of experiments



Examples of unmatched components

Example (Acer server)

```
AAG320 PD 940 (3.2 GHz, 2x 2MB, 800 MHz FSB),
1x 512 MB DDR2 533/16x DVD-ROM
Acer Altos G320-PD940 3.2GHz/2x2MB,800F/512MB/DVD/noHDD/noKB
```

Examples of unmatched components

Example (Acer server)

```
AAG320 PD 940 (3.2 GHz, 2x 2MB, 800 MHz FSB),
1x 512 MB DDR2 533/16x DVD-ROM
Acer Altos G320-PD940 3.2GHz/2x2MB,800F/512MB/DVD/noHDD/noKB
```

- `Acer Altos` is abbreviated to `AA`.

Examples of unmatched components

Example (Acer server)

```
AAG320 PD 940 (3.2 GHz, 2x 2MB, 800 MHz FSB),
1x 512 MB DDR2 533/16x DVD-ROM
Acer Altos G320-PD940 3.2GHz/2x2MB,800F/512MB/DVD/noHDD/noKB
```

- `Acer Altos` is abbreviated to `AA`.
- Different token separators (comma, space, slash, dash, braces) are used.

Examples of unmatched components

Example (Acer server)

```
AAG320 PD 940 (3.2 GHz, 2x 2MB, 800 MHz FSB),
1x 512 MB DDR2 533/16x DVD-ROM
Acer Altos G320-PD940 3.2GHz/2x2MB,800F/512MB/DVD/noHDD/noKB
```

- `Acer Altos` is abbreviated to `AA`.
- Different token separators (comma, space, slash, dash, braces) are used.
- Whether a symbol is a separator depends on its context.

Examples of unmatched components

Example (Acer server)

```
AAG320 PD 940 (3.2 GHz, 2x 2MB, 800 MHz FSB),
1x 512 MB DDR2 533/16x DVD-ROM
Acer Altos G320-PD940 3.2GHz/2x2MB,800F/512MB/DVD/noHDD/noKB
```

- `Acer Altos` is abbreviated to `AA`.
- Different token separators (comma, space, slash, dash, braces) are used.
- Whether a symbol is a separator depends on its context.
- For example, the space symbol is a separator between `PD940` and `3.2 GHz` but "`3.2 GHz`" should be one token.

Examples of unmatched components

Example (Ink cartridge)

```
Ink. náplň No. 84 pro DesignJet 10PS/20PS/50PS
C5016A Black ink Cartridge pro DSJ x0ps
```

Examples of unmatched components

Example (Ink cartridge)

`Ink. náplň No. 84 pro DesignJet 10PS/20PS/50PS`
`C5016A Black ink Cartridge pro DSJ x0ps`

- `Cartridge` is `náplň` in Czech,

Examples of unmatched components

Example (Ink cartridge)

```
Ink. náplň No. 84 pro DesignJet 10PS/20PS/50PS
C5016A Black ink Cartridge pro DSJ x0ps
```

- `Cartridge` is `náplň` in Czech,
- `10PS/20PS/50PS` is abbreviated to `x0ps`, and

Examples of unmatched components

Example (Ink cartridge)

`Ink. náplň No. 84 pro DesignJet 10PS/20PS/50PS`
`C5016A Black ink Cartridge pro DSJ x0ps`

- `Cartridge` is `náplň` in Czech,
- `10PS/20PS/50PS` is abbreviated to `x0ps`, and
- `DesignJet` is abbreviated to `DSJ`.

Examples of unmatched components

Example (Cable)

```
Kabel Pure AV Blue series Firewire 4pin/6pin, 1.8m
PureAV kabel FireWire, 4/6 kolíků - 1,8 m - Řada Blue
```

Examples of unmatched components

Example (Cable)

`Kabel Pure AV Blue series Firewire 4pin/6pin, 1.8m`
`PureAV kabel FireWire, 4/6 kolíků - 1,8 m - Řada Blue`

- series is Řada in Czech,

Examples of unmatched components

Example (Cable)

`Kabel Pure AV Blue series Firewire 4pin/6pin, 1.8m`
`PureAV kabel FireWire, 4/6 kolíků - 1,8 m - Řada Blue`

- series is Řada in Czech,
- 4pin/6pin corresponds to 4/6 kolíků since pin is kolík in Czech, and

Examples of unmatched components

Example (Cable)

`Kabel Pure AV Blue series Firewire 4pin/6pin, 1.8m`
`PureAV kabel FireWire, 4/6 kolíků - 1,8 m - Řada Blue`

- `series` is `Řada` in Czech,
- `4pin/6pin` corresponds to `4/6 kolíků` since `pin` is `kolík` in Czech, and
- `1.8m` corresponds to `1,8m`.

Examples of unmatched components

Example (Mail antispam and antivirus)

```
SYMANTEC BRIGHTMAIL ANTISPAM + ANTIV 6.0 SUBS
+ GOLD MAINT 1YR IN VALUE BAND F(5
Sym. Bright.Antispam + Antivirus 6.0 IN F(500-999) + 1YR GM
```

Examples of unmatched components

Example (Mail antispam and antivirus)

`SYMANTEC BRIGHTMAIL ANTISPAM + ANTIV 6.0 SUBS`
`+ GOLD MAINT 1YR IN VALUE BAND F(5`
`Sym. Bright.Antispam + Antivirus 6.0 IN F(500-999) + 1YR GM`

- `Sym. Bright.Antispam + Antivirus` corresponds to `SYMANTEC BRIGHTMAIL ANTISPAM + ANTIV` and

Examples of unmatched components

Example (Mail antispam and antivirus)

```
SYMANTEC BRIGHTMAIL ANTISPAM + ANTIV 6.0 SUBS
+ GOLD MAINT 1YR IN VALUE BAND F(5
Sym. Bright.Antispam + Antivirus 6.0 IN F(500-999) + 1YR GM
```

- `Sym. Bright.Antispam + Antivirus` corresponds to `SYMANTEC BRIGHTMAIL ANTISPAM + ANTIV` and
- GM is an abbreviation for `GOLD MAINT`.

Conclusions

- We performed experiments with three string similarity measures on real data

Conclusions

- We performed experiments with three string similarity measures on real data
- We observed the best performance for the vector based method.

Conclusions

- We performed experiments with three string similarity measures on real data
- We observed the best performance for the vector based method.
- At 62% of cases found the correct component first and in 83% of cases it was among the first five.

Conclusions

- We performed experiments with three string similarity measures on real data
- We observed the best performance for the vector based method.
- At 62% of cases found the correct component first and in 83% of cases it was among the first five.
- It was slightly improved when combinined with the string similarity measure.

Conclusions

- We performed experiments with three string similarity measures on real data
- We observed the best performance for the vector based method.
- At 62% of cases found the correct component first and in 83% of cases it was among the first five.
- It was slightly improved when combinined with the string similarity measure.
- At 67% of cases found the correct component first and in 85% of cases it was among the first five.

Conclusions

- We performed experiments with three string similarity measures on real data
- We observed the best performance for the vector based method.
- At 62% of cases found the correct component first and in 83% of cases it was among the first five.
- It was slightly improved when combinined with the string similarity measure.
- At 67% of cases found the correct component first and in 85% of cases it was among the first five.
- a smarter method for separating strings into tokens

Conclusions

- We performed experiments with three string similarity measures on real data
- We observed the best performance for the vector based method.
- At 62% of cases found the correct component first and in 83% of cases it was among the first five.
- It was slightly improved when combinined with the string similarity measure.
- At 67% of cases found the correct component first and in 85% of cases it was among the first five.
- a smarter method for separating strings into tokens
- the vector method as a basis for further improvements

Future work

- One way to go is to work with a matrix $\mathbf{P}$ that would provide for all pairs of tokens their similarity.

Future work

- One way to go is to work with a matrix $\mathbf{P}$ that would provide for all pairs of tokens their similarity.
- We could assume that the values of matrix $\mathbf{P}$ are zero unless specified otherwise.

Future work

- One way to go is to work with a matrix **P** that would provide for all pairs of tokens their similarity.
- We could assume that the values of matrix **P** are zero unless specified otherwise.
- There are several ways of having the values different from zero and they could be combined together. We could use:

Future work

- One way to go is to work with a matrix $\mathbf{P}$ that would provide for all pairs of tokens their similarity.
- We could assume that the values of matrix $\mathbf{P}$ are zero unless specified otherwise.
- There are several ways of having the values different from zero and they could be combined together. We could use:
  - a dictionary of synonyms,

Future work

- One way to go is to work with a matrix **P** that would provide for all pairs of tokens their similarity.
- We could assume that the values of matrix **P** are zero unless specified otherwise.
- There are several ways of having the values different from zero and they could be combined together. We could use:
  - a dictionary of synonyms,
  - Czech-English dictionary,

Future work

- One way to go is to work with a matrix **P** that would provide for all pairs of tokens their similarity.
- We could assume that the values of matrix **P** are zero unless specified otherwise.
- There are several ways of having the values different from zero and they could be combined together. We could use:
  - a dictionary of synonyms,
  - Czech-English dictionary,
  - a system of rules used for making common abbreviations, etc.

Future work

- One way to go is to work with a matrix $\mathbf{P}$ that would provide for all pairs of tokens their similarity.
- We could assume that the values of matrix $\mathbf{P}$ are zero unless specified otherwise.
- There are several ways of having the values different from zero and they could be combined together. We could use:
  - a dictionary of synonyms,
  - Czech-English dictionary,
  - a system of rules used for making common abbreviations, etc.
- This leads to a natural generalization of the vector method:

$$\begin{aligned} Sim(S_1, S_2) &= \sum_{i=1}^{d} \sum_{j=1}^{d} v(x_i, S_1) \cdot \mathbf{P}_{i,j} \cdot v(x_j, S_2) \\ &= \mathbf{v}(S_1)^T \cdot \mathbf{P} \cdot \mathbf{v}(S_2) \ . \end{aligned}$$

Future work

- One way to go is to work with a matrix **P** that would provide for all pairs of tokens their similarity.
- We could assume that the values of matrix **P** are zero unless specified otherwise.
- There are several ways of having the values different from zero and they could be combined together. We could use:
  - a dictionary of synonyms,
  - Czech-English dictionary,
  - a system of rules used for making common abbreviations, etc.
- This leads to a natural generalization of the vector method:

$$
\begin{aligned}
Sim(S_1, S_2) &= \sum_{i=1}^{d} \sum_{j=1}^{d} v(x_i, S_1) \cdot \mathbf{P}_{i,j} \cdot v(x_j, S_2) \\
&= \mathbf{v}(S_1)^T \cdot \mathbf{P} \cdot \mathbf{v}(S_2) \ .
\end{aligned}
$$

- Since the matrix **P** and vectors $\mathbf{v}(S_1), \mathbf{v}(S_2)$ are sparse the computations can be efficiently implemented.