An evaluation of string similarity measures on pricelists of computer components*

Radim Jiroušek, Václav Kratochvíl, Tomáš Kroupa, Radim Lněnička, Milan Studený, and Jiří Vomlel

Institute of Information Theory and Automation
Academy of Sciences of the Czech Republic
http://www.utia.cz/

Petr Hampl and Helena Hamplová

Empo spol. s r. o.
Jeremenkova 88, Prague 4, Czech Republic
http://www.empo.cz/

Abstract

This paper presents new results of our experimental research on string similarity measures on pricelists of computer components. The first results were already presented at the Eighth Czech-Japan Seminar on Data Analysis and Decision Making under Uncertainty in Třešť [1] in 2005.

1 Introduction

The task we are interested in is to find a given computer component described by an unstructured text - a string S of characters - in different pricelists of computer components. Different suppliers describe the same component somewhat differently, therefore we cannot rely on a simple match between two components' descriptions. First, let us give three examples of components as they are described in two different pricelists.

```
Toner Cartridge pro LJ4/M/+/4M+/5/5M/5N 92298X
Toner pro LaserJet 4/4M, 4/4M Plus, 5/5N/5M (8800)

Pilot Optical Mouse, USB+PS/2, 3 tlačítka, černá
Logitech myš Pilot Optical Mouse Black, USB/PS/2, retail
```

*This work was supported by the Ministry of Education of the Czech Republic through grant nr. 1M0572 DAR.

```
AS 9804WKMi_RAID, T2500/2.0GHz, 2048MB, 2x120GB, DVD+/-RW SM, 20" WXGA,
WMC, ca
Aspire9804AWKMiDuo2,0GHz/2x1024MB/2x120HDD/DVD+-RWsm/20"/Ge7600/BT/1.3MB/
V/TV/WMC
```

Some of the pricelists we worked with contained more than 30,000 components. Fortunately, the computer components pricelists are partially structured. Therefore, in the current implementation of our system when searching an equivalent component in other pricelist we can restrict the search only to the same producer and the same product category[1].

Some suppliers provide together with the component description also its part number that should be unique. This can provide a very reliable matching between components from different pricelists. Unfortunately, we have observed that many items in pricelists do not have any part number assigned and some suppliers do not provide part numbers in their pricelists at all. Thus we use part numbers when they are available, but we have to have a method for finding equivalent components even if the part number is not available.

The rest of the paper is organized as follows. In the next section we review three methods we tested - a fulltext search, string similarity, and a vector based method plus a linear combination of these three methods. The third section describes results of experiments with the methods on a real data from two computer parts suppliers. We conclude the paper with conclusions and suggestions for future work.

2 Tested methods

The fulltext search method

As a reference method we have used the natural language search of MySQL Full-Text Search [2]. A natural language search interprets the search string S_1 as a phrase in natural human language (a phrase in free text). The MySQL stopword list applies. In addition, words that are present in more than 50% of the records are considered common and are not matched. Also words shorter than four characters are not matched. We denote this similarity measure between the searched string S_1 and a found string S_2 as $Sim_1(S_1, S_2)$.

String similarity

The first of proposed methods that we tested was inspired by the string edit distance [3]. It is described in detail in our previous paper on this topic [1]. 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

[1] Actually, not all components can be classified to a category therefore we need to use one additional category of unclassified components and we always search also in this category.

longer than two are counted several times. In the experiments we used the relative string similarity defined as

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

Vector based method

In vector based techniques [5] 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. In our context the string is divided into tokens by special characters (space, comma, semicolon, etc.) as tokens separators (separators are then omitted).

A popular method for computing the weights is the TF-IDF method [4], which we have also described in detail in our previous paper on this topic [1]. The weight of a token x in string S is defined as

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

where $n(x, S)$ is the number of occurrences of token x in string S (often, it is 0 and 1), $n(S)$ is the total number of tokens in string S, m is the total number of all strings in the data, and $m(x)$ is the number of strings containing token x. Let d denote 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 defined as

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

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 $\mathbf{v}(S_2)$

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

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

A combination of the methods

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

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

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

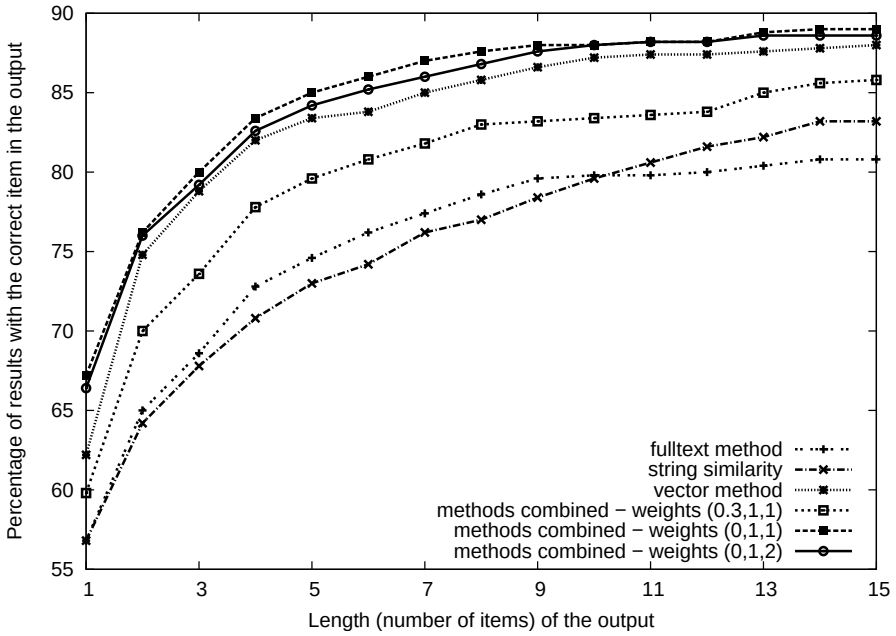

Figure 1: The comparison of the relative success rate of different methods with respect to the length of the output.

3 Experiments

First, we describe how we prepared a testing set. 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 and were present in both pricelists. Since only some components are listed in the pricelists with their part number we have got 7060 different part numbers. From these we randomly selected 500 part numbers. These defined our test pairs.

During the tests, for each component from the first pricelist (having the part number from those 500 selected ones) we used the tested methods to find k ($k = 1, 2, \ldots, 15$) most similar components in the second pricelist. Then we checked whether the component with the same part number is among those k selected ones (let us call k the length of the output). We counted the number of these cases and computed the relative success rate for each method with respect to k. The results for six different methods are presented in Figure 1.

From the figure we can see that from the three basic methods - the full-text search, string similarity, and the vector based method - by far the best performance was achieved with the vector based method. The fulltext method performance is rather poor since it even worsens the performance of the other methods when it is combined with them. From the tested combinations the

combination with weights $(0, 1, 1)$ appeared to be the best for the testing set. Thus, we can conclude that the vector based method can be slightly improved when it is combined with the string similarity.

Next, we provide four examples of the descriptions of component pairs that were not matched. For correct matching we would need to have the above methods extended so that they use additional information discussed in each of the examples.

Example 1 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 the same symbol (e.g., the space) is a separator (e.g., between `PD940` and `3.2GHz`) or it is not (e.g., `800 MHz` should be one token) depends on the context.

Example 2 Ink cartridge

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

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

Example 3 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`.

Example 4 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`.

4 Conclusions and future work

In this paper we performed experiments with three string similarity measures on real data - pricelists of computer components from two suppliers. We observed the best performance for the vector based method, which at about 62% of cases found the correct component as the first one and in 83% of cases it was among

the first five. This method was possible to slightly improve by combining it with the string similarity measure, which at about 67% of cases found the correct component as the first one and in 85% of cases it was among the first five.

Since the vector method appeared to perform best from the tested single methods it could serve as a basis for further improvements. One way to go is to work with a matrix $\mathbf{P}$ that would provide for all pairs of tokens their similarity. If this number would be in the interval $[0, 1]$ then it could be interpreted as a probability that the tokens are equivalent. Actually, we would not need to create explicitly the whole matrix $\mathbf{P}$ if we would assume that the matrix values 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 (pairs of synonyms would get value one), Czech-English dictionary (again, a pair from dictionary would get value one), a system of rules used for making common abbreviations, etc. This would lead to a natural generalization of the vector method where the formula (1) would be replaced by

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

Since the matrix $\mathbf{P}$ is sparse the computations can be efficiently implemented.

Additionally, Example 1 indicates that a smarter method for separating strings into tokens would be useful.

References

[1] H. Hamplová, J. Ivánek, R. Jiroušek, T. Kroupa, R. Lněnička, M. Studený, and J. Vomlel. Decision support system for comparison of price lists. In J. Vejnarová and T. Kroupa, editors, *Proceedings of the the Eighth Czech-Japan Seminar 2005 on Data Analysis and Decision Making under Uncertainty*, pages 32—38, Třešť, Czech Republic, 2005.

[2] MySQL full-text search. World wide web document: `http://dev.mysql.com/doc/refman/5.0/en/fulltext-search.html`.

[3] E. S. Ristad and P. N. Yianilos. Learning string edit distance. Research report CS-TR-532-96, Department of Computer Science, Princeton University, Princeton, NJ, 1997.

[4] G. Salton and C. Buckley. Term weighting approaches in automatic text retrieval. *Information Processing and Management*, 24(5):513—523, 1988.

[5] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communications of the ACM*, 18(11):613—620, 1975.