A Semantic-based String Matching Library

Yue Han

Binghamton University

yhan1@binghamton.edu

Abstract String matching has been widely used in more and more web data applications. Although a lot of matching techniques have been proposed, most of them are based on evaluating the similarity of strings in terms of their lexical or syntactic structures. The fact is that the same entity can be denoted by totally different expressions for which the conventional syntactic-based string matching techniques will not be effective any more. In this paper, a semantic-based string matching library (SSML) is proposed and implemented to achieve the semantic level string matching by combining all possible patterns for specific domain and appropriate matching techniques together. Particularly, two widely used domains are analyzed in details and implemented, that is, "Date" and "Person Name". For each domain, a metric is proposed to quantify the potentiality of string matching. In general, under the framework of SSML, it is convenient and efficient to add new domains into the system.

1. Introduction

Data on the web is increasing exponentially, which drives a lot of emerging techniques to make users access the available information more effectively and efficiently, like metasearch engine, schema/ontology integration, data warehouse, ecommerce, query mediation, etc. The common problem to be issued in all these different applications is to find occurrence of a pattern string within another string or body of text, which is called string matching. Obviously, string-matching plays an important role in the wider domain of text processing.

Since string-matching is a conventional research topic in information retrieval, it has been extensively studied. While it being formulated differently in diverse communities, including statistics, database, artificial intelligence and so on, different techniques have been proposed to improve the matching performance in terms of both accuracy and efficiency. Some examples of string-based techniques which are extensively used in matching systems are prefix, suffix, edit distance, and n-gram. The matching is based on evaluating the similarity between the strings and most of the existing work focus on inventing new metrics to quantify this similarity. Although a practical threshold value based on experiments can be used to discriminate whether the strings should be matched or not, the original numerical value of similarity can always denote the possibility of matching even without introducing the new parameter.

These wildly used and studied approaches in string matching evaluate similarity in terms of the lexical and syntactic structure of strings. They are typically based on the following intuition: the more similar the strings, the more likely they denote the same concepts. However, the fact is strings can still represent the same entity even if they are very different in the lexical and syntactic structure. For example, Alexander. W. Street compared with Alex William Street, August 20th, 2008 compared with 08/8/20. By using the syntactic-based matching techniques, it tends to be concluded that they are not matched and should denote totally different concept. Obviously, the conclusion is not correct since

these two entities are just expressed in different formats of representations respectively. Therefore, the semantics of strings should also be taken into consideration in order to make the matching results more convincible. Two requirements should be satisfied while achieving the semantics-level string matching:

(1) Some prior knowledge should be used to ease the understanding of particular strings for systems, like domain names;

(2) There should be some definite patterns to represent a certain category of entities.

Obviously, the first requirement will speed up the matching process for strings. Otherwise, we cannot afford the time and space complexity to automatically form knowledge about the strings and also it could be very inaccurate. The second requirement underlies the necessity of doing string-matching from the aspect of semantics and the relative certainty of given patterns will also make it feasible. Without these two conditions satisfied, the development of semantic-level string matching approaches will be meaningless.

In this paper, a semantic-based string matching library (SSML) is proposed to implement the semantic-level matching structure by combining patterns, domains and matching techniques together. For each domain, the corresponding patterns in which the domain entities can be represented are analyzed and implemented by using appropriate matching techniques. These different domains associated with their patterns and matching techniques are named matching functions. On the user-level interface, the functions will accept a domain name and two strings as input and return one of the following values as output: No(not match), Yes(definite match), Maybe(potential match).

The patterns in different domains could be very complicated so that the regular expressions are not enough to express the pattern structures. However, the flexibility of programming languages makes it still feasible to be implemented under the framework of SSML. That's why we have tried to take into consideration the patterns as many as possible for these domains. In the program, two specific domains are implemented, Date and Person Name. The sifting of these two domains is based on the two requirements we discussed above. Another reason is that they are widely used in a lot of different applications.

Except for the cases of definite matching and not matching of strings, potential match will also be generated since there could be some degree of ambiguity while any string for evaluation is mapped into multiple patterns. In this case, a corresponding metric for each domain is put forward to quantify the potential match of strings. Obviously, a higher value would indicate a higher possibility of matching.

SSML provides a simple approach by looking up the appropriate patterns with the input domain knowledge to achieve the semantic level understanding of the strings. While being applied in the field of query searching, more relevant documents can be retrieved and the effectiveness of search engines will be greatly improved without limiting the evaluation on the similarity of lexical or syntactic structure.

The rest of the paper is organized as follows. Section 2 introduces some well-know string matching techniques and related work in semantic matching. Section 3 describes the general framework of SSML and the process of SSML development. Section 4 has a thorough analysis on the patterns, matching techniques and evaluation metrics in the domain of date, while some details and problems in the implementation are also included. Similarly, Section 5 also analyze these aspects in details in the domain of person name. Section 6 lists some evaluation results based on the implementation and shows that the accuracy and efficiency of SSML proposed and implemented in this paper. Section 7 concludes all the work which has been done and put forward some future research directions in this field.

2. Related Work

The task of matching entity names has been explored by a number of communities [2]. They are typically based on the following intuition: the more similar the strings, the more likely they denote the same concepts. A lot of matching techniques have been proposed to tackle this problem. Distance functions map a pair of strings *s* and *t* to a real number *r*, where a smaller value of *r* indicates greater similarity between *s* and *t*. Similarity functions are analogous, except that larger values indicate greater similarity. One important class of distance functions are edit distances, in which distance is the cost of best sequence of edit operations that convert *s* to *t*. Typical edit operations are character insertion, deletion, substitution, and each operation must be assigned a cost. Two strings *s* and *t* can also be considered as multisets of words. Token-based distance metrics will be used to evaluate the similarity between them. Besides, some techniques are designed for long string comparison. Such as hybrid distance functions which apply the recursive matching scheme for comparing two long strings, blocking or pruning methods which focus on the comparison of the keywords by using priority knowledge.

The string-based techniques above consider strings as sequences of letters in an alphabet and doing comparisons with respect to the lexical structures. Language-based techniques consider entity names as words in some natural languages. They are based on natural Language processing techniques exploiting morphological properties of the input strings, including tokenization, lemmatization and elimination. Constraint-based techniques deal with the internal constraints being applied to the definitions of entities, such as type, cardinality of attributes, and keys; linguistic resources such as common knowledge or domain specific thesauri care used in order to match words based on linguistic relations between them, like WordNet, per-compiled thesaurus. In fact, the latter two categories of techniques have taken into consideration the semantics of strings in a naive way. However, the formats used for expressing the entity name could be much more complicated. That's why more sophisticated techniques need to be developed. Comparatively, matching is considered as an operator that takes two graph-like structures and produces a mapping between elements of the two graphs that correspond semantically to each other in the paper [3]. Two levels of semantic matching are proposed. Element-level semantic matching includes analysis of strings, data types, soundex, precompiled thesaurus and WordNet. The structure-level semantic matching is to translate the matching problem, namely the two graphs and our mapping queries into a propositional formula and then to check it for its validity. It is based on the element-level semantic matching while the relations among different nodes are considered.

The concept of two-level matching structure will be introduced into the design of SSML in this paper. Instead of analyzing the complicate relations among different nodes in the graph, a simple approach will be adopted to achieve the semantic-level matching structure which combines the three different attributes of a function entry together, including pattern, domain and element-level matching techniques. On the element-level matching structure, the traditional matching techniques will also be used for the implementation, like prefix, suffix, precompiled thesaurus, WordNet, etc.

3. A Framework of SSML

SSML can be thought as a collection of matching functions while each of them will provide all possible definite patterns for the corresponding domain. This is the higher level of semantic-matching which is implemented in a simple way that the domain list is looked up and the corresponding patterns will be further evaluated. For a specific domain, many existing syntactic or syntactic-like matching techniques can be used to construct the string-based patterns. This is called the element-level matching structure.

The framework of the implementation of SSML is depicted as figure 1. In fact, there are always different types of patterns existing even for the strings in the same domain. In order to map the input string into a specific pattern and then do the matching more efficiently, the different types of patterns in the same domain will be separated in multiple entries while each of them will be associated with the most appropriate matching techniques.

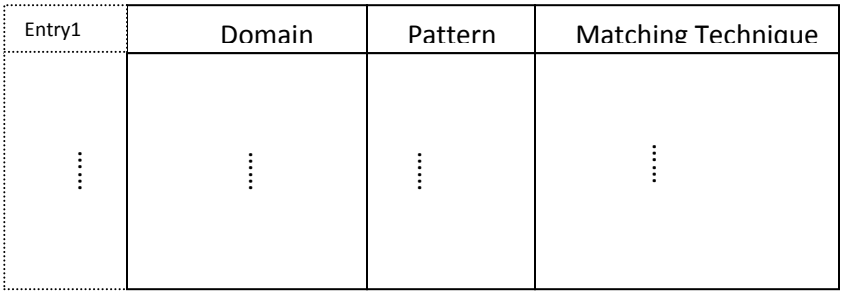

Figure 1 The framework of SSML

This framework is the most important component during the development of SSML. Each step during the process of SSML development will analyze the different attributes of matching functions and aim at constructing a complete framework finally. Figure 2 shows the process of SSML development with specific explanations for each step.

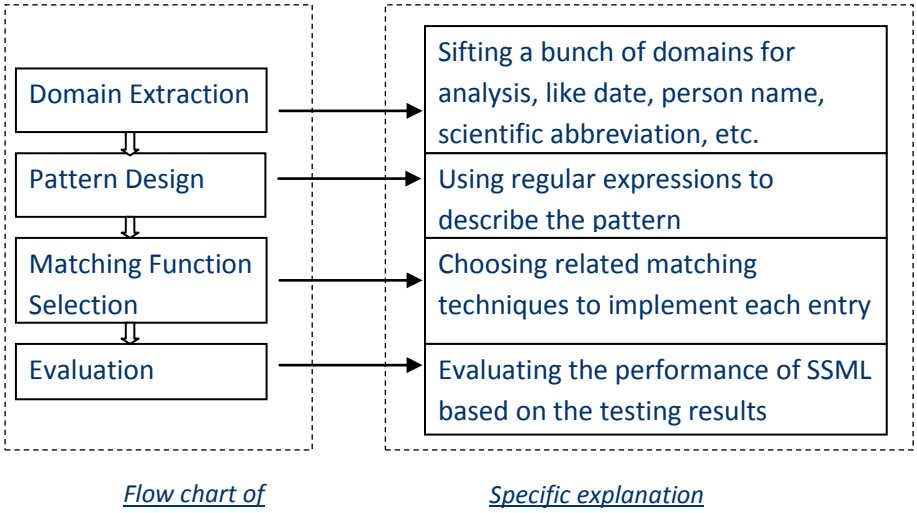

Figure 2 The process of SSML development

On the user-level interface, the functions will accept a domain name and two strings as input and return one of the following values as output: No(not match), Yes(definite match), Maybe(potential match). It is depicted in Figure 3.

Figure 3 The functionality of SSML

4. Analysis on Domain — "Date"

4.1 Pattern Design

The date usually indicates an absolute date point with the format containing the components including year, month and day. First, a thorough description of the different representations of a date component pattern is presented. For the component of year, it could be a two-digit number which is between the integer intervals from 00 to 99. There must be a zero ahead while only one digit is effective to indicate the number of year, for example 08 while the "0" here cannot be ignored. It could also be a four-digit number, like 1985. For the component of month, it can be a number from 1 to 12; it also could be a string which indicates the corresponding month, including both of full names, like January and abbreviations, like Jan. For the component of day, it can be a number from 1 to 31; it also could be number with a suffix, like 31st. Table 1 gives an overall description of the time component pattern representation.

Letter Rep.	Time Component	Presentation Format	Example
yy	Year	2-digit number	08
yyyy	Year	4-digit number	1985
mmm	Month in year	Text	Jan, January
mm	Month in year	Number	1
dd	Day in Month	Number	31,31st

Table 1. Time component pattern representation

Secondly, whether a term sequence can be parsed into a time is determined by a set of pre-defined rules. For example, if a rule says "a date is a month in text form followed by a day in number form and a year in 4-digit form", the term sequence "October 27, 2008" will satisfy the rule and will be parsed into a date. Such rules can be called patterns. Based on the different representations of the time component, the following patterns can be introduced and also they are legally used in linguistics.

(a) yy/mm/dd For example: 08/10/27
(b) yyyy/mm/dd For example: 2008/10/27
(c) mm/dd/yy For example: 10/27/08
(d) mm/dd/yyyy For example: 10/27/2008
(e) dd/mm/yy For example: 27/10/08
(f) dd/mm/yyyy For example: 27/10/2008

The slash "/" can be replaced by "-" or ".".

When the moth is represented by text format and the day can be with or without suffix, the patterns accepted are:

(g) mmm dd, yyyy For example: Oct 27, 2008 or Oct 27th, 2008
(h) mmm dd, yy For example: Oct 27, 08 or Oct 27th, 08

Sometimes, the input string may contain only a subset of time components. This is called incomplete date pattern. It could be part of any pattern listed above.

4.2 Metrics for quantifying the potential match

If both of the two input strings can match a specific pattern separately and denote the fixed date without any ambiguity, the output of the matching function will be "Match" or "Not match". If any of the two input strings can be assigned to multiple patterns and cannot represent a clear date, the output of the matching function will be "Potential match" or "Not match". For example, 11/31/2008 and 08/11/31 are evaluated. Both of them can denote a date which is formed by the year of 2008, the month of November and the day of 31st without any ambiguity. If we consider 11/12/10 and 2010/11/12, according to the patterns we have listed above, 11/12/10 can denote yy/mm/dd, mm/dd/yy or dd/mm/yy; although 2010/11/12 can only denote yyyy/mm/dd, we cannot conclude that they are matched. When 11/12/10 denotes mm/dd/yy, they could be matched. So the output of the matching function for these two input strings is "Potential match".

In order to make the strings match a specific pattern without any ambiguity, more conditions can be observed from the characteristics of different components of date. These conditions can be used to discriminate day, month and year. For example, if a component consists of 4 digits, it can only denote the year; if a component consists of 2 digits and the first digit is 0, it can only denote the year; if a component consists of 2 digits and this 2-digit number is greater than 31, it can only denote the year. If we have known that a 2-digit number doesn't denote the year, it must denote the day while it is greater than 12. In my implementation, I have taken all these conditions into consideration.

Instead of using a general "potential match" term to indicate the likelihood of matching between two date strings, we can assign a specific value as the potential match result. Obviously, this value should be able to show us how likely the two strings will be matched. In order to measure this potentiality, we introduce the following metric

$$Match(S_1, S_2) = \frac{P_{s1,s2}}{P_{s1} * P_{s2}}$$

$P_{s1,s2}$ represents the number of patterns which can be associated with s1 and s2 when they are matched; P_{s1} represents the number of patterns which can associated with s1; P_{s2} represents the number of patterns can be associated with s2. Obviously, the more patterns an input string could be associated with, the more ambiguous the input string will be. For the simplest case, both s1 and s2 can be associated with a fixed pattern respectively, P_{s1}=1 and P_{s2}=1 and the number of associated pattern when they are matched is also equal to 1. So $Match(S_1, S_2)$ = 1 indicates the two input strings can definitely match. If s1 and s2 can never be matched no matter which pattern is adopted, $Match(S_1, S_2)$ = 0. If two strings may be matched while some specific patterns are assigned, $Match(S_1, S_2)$ will be a value in the interval (0, 1). Higher value indicates higher possibility of matching, vice versa. For example, both of 02/03/04 and 03/04/02 can be associated with yy/mm/dd, mm/dd/yy, dd/mm/yy respectively. And also for each of the three patterns, they can be matched. So $Match(S_1, S_2)$ = 3/(3*3)=1/3 indicates the potentiality of matching.

4.3 Matching Techniques

The matching techniques related in the implementation of the domain of date include:

(1) Prefix: One token is just the initial part of another token with a non-modified meaning. For example, January and Jan.
(2) Suffix: One token is an affix which is placed at the end of another token. For example, 2008 and 08.
(3) Rearrangement: In this situation, a representation is formed by changing the relative positions of tokens in another representation, without modifying the original meaning. For example, 1/31/2008 and 2008/1/31.
(4) Synonymy: If two representations with different words have similar or identical meaning, they would be considered synonymous. For example, February and 2. Usually the precompiled thesaurus or WordNet can be used to tackle this problem. In this implementation, the number of synonymy is limited so we can simply enumerate all the cases.

Obviously, (1) and (2) are character-level matching techniques, (3) is the token-level matching technique and (4) is the semantic-level matching technique.

5. Analysis on Domain — "Person Name"

5.1 Pattern Design

Unlike the time components of "date" domain can be represented in different formats in terms of the number of digits, number or text, suffix and abbreviations, the format of name components is less complicated and is consisted of a sequence of letters(including the case of only one first capital letter). The primary components of "name" domain are first name, middle name and last name. Sometimes, a nick name may exist which is listed in the parentheses. The table below describes the formats for these name components.

Letter Rep.	Name Component	Presentation Format	Example
fff	First name	Multiple letters	George
F	First name	Single capital letter	G.
mmm	Middle name	Multiple letters	William
M	Middle name	Single capital letter	W.
lll	Last name	Multiple letters	Bush
nnn	Nick name	Multiple letters	Mark

The most widely used patterns are

(a) “fff mmm lll” or “fff lll” or “lll, fff”

While taking into consideration the first initials, the patterns below are also common

(b) “F. M. lll” or “fff M. lll”

Sometimes, the “/” will be used to separate the first name and last name

(c) “lll/fff”

It is not necessary to match the exact order of name components and also the using habit on the initials of which name components are adopted, so the following patterns are also acceptable

(d) “fff/lll” or “fff/mmm/lll”
(e) “F. mmm lll”

As to the nick name, it is usually put into the parentheses after the first name

(f) “fff(nnn) lll”

Sometimes, some variations to the same name may exist, like Mike and Michael. A thesaurus can be embedded in the library to tackle this problem. There may have other patterns for the expression on person names. Based on the extraction of the name components from the “name” domain, other patterns can always be added into the pattern library and easily associated with the input name string.

5.2 Metrics for quantifying the potential match

Because of the simplicity of patterns, most input strings must be associated with specific patterns defined in the pattern library without ambiguity. If all the name components of the two input strings are represented in the multiple-letter formats. The comparison between them tends to generate a definite answer, “match” or “not match”. If any of the two strings contains a name component represented by the only first capital letter, the output of matching function would be “potential match” or “not match”. The more one-capital-letter components it contains, the lower is the potentiality of the matching. So we can introduce another metric to quantifying the possibility of matching which is similar to the case in “date” domain.

$$Match(S_1, S_2) = 1 - \frac{C_{s1} + C_{s2}}{N_{s1} * N_{s2}}$$

C_{s1} and C_{s2} represents the number of name components which is in the one-capital-letter format for input string s1 and s2 respectively; N_{s1} and N_{s2} represents the number of name components contained in each input string without taking into consideration the format. If both of the two strings doesn’t contain one-capital-letter component, C_{s1}=0 and C_{s2} =0, we conclude $Match(S_1, S_2)$ =1. For example, B. H. Obama and Barack H. Obama, the value would be 1-(2+1)/(3*3) = 6/9.

5.3 Matching Techniques

The most used two matching techniques in the implementation of “name” domain are prefix and synonymy. A parameter of length can be introduced to match a certain number of prefix letters. Obviously in the patterns of name domain, the length is assigned to one.

Sometimes two different expressions can denote the same name. It can be a prefix, like Alex and Alexandar; it can be variations, like Abe and Abraham. So we need to build a synonym library to take the variations of names into consideration.

6. Experimental Results

Some of the experimental based on my implementation of SSML is listed as below and it shows that the correctness of system while taking into consideration all the patterns discussed in this paper for the domain of "Date" and "Person Name".

Input: "2008/8/9" "Aug 9th,08" "Date" Output: Match

Input: "1/31/01" "2001/1/31" "Date" Output: Match

Input: "08.5.25" "5-25-2008" "Date" Output: Match

Input: "February 23,08" "23/2/2008" "Date" Output: Match

Input: "11/12/13" "2013/11/12" "Date" Output: Potential match

Input: "08/11/13" "11/14/2008" "Date" Output: Not match

Input: "George Bush" "Bush, George" "Person Name" Output: Match

Input: "Nick W. Street" "Nick William Street" "Person Name" Output: Potential Match – 8/9

Input: "Zhongfei(mark) Zhang" "Zhongfei Zhang" "Person Name" Output: Match

Input: "Han/Yue" "Yue, Han" "Person Name" Output: Match

Input: "Ye Liu" "Liu/Yue" "Person Name" Output: Not match

Input: "Joe Li" "Joseph Li" "Person Name" Output: Match

Input: "B. H. Obama" "Barack H. Obama" "Person Name" Output: Potential Match – 6/9

7. Conclusion and Future work

In this paper, we build up a framework for the semantics-based string matching library under which it is convenient and effective to implement the matching functions in different domains. In our implementation, two widely used domains are considered, that is, "Date" and "Person Name". There is always a certain degree of ambiguity existing during the matching process when the strings could be mapped into multiple patterns. Therefore, it is necessary to quantify the potentiality of matching by using appropriate metrics proposed in this paper.

Although we have tried to take into consideration the patterns in each domain as many as possible, there could be some missing patterns as the limit of knowledge. In the future, the patterns in the developed domains could be further expanded and also new domains could be implemented into the library. Another interesting direction is to consider quantifying the importance of patterns. Even if there are a lot of patterns in a specific domain, some of them may not be widely used. In this case, we should discriminate the importance of the patterns.

8. References

1. Pavel Shvaiko, Jerome Euzenat. A Survey of Schema-based Matching Approaches. Journal on Data Semantics IV (2005), pp. 146-171.
2. WilliamW. Cohen, Pradeep Ravikumar, Stephen E. Fienberg. A Comparison of String Distance Metrics for Name-Matching Tasks. Proceedings of the IJCAI-2003 Workshop on Information Integration on the Web.
3. Fausto Giunchiglia, Pavel Shvaiko. Semantic matching. Technical Report # DIT-03-013.
4. Yiyao Lu, Weiyi Meng, Wanjing Zhang, King-Lup Liu, and Clement Yu. Automatic Extraction of Publication Time from News Search Results. WIRI2006.