On-line fuzzy pattern matching
on sequences
(Extended abstract)

V. A. OLESHCHUK
Department of Information and Communication Technology
Agder University College
and
Sentek, Agder Research Foundation
N-4876 Grimstad, NORWAY.

Abstract

We consider pattern matching problems where patterns are presented as sequences of fuzzy constraints on input elements. Given an infinite alphabet $A$, a pattern $P[\alpha, \beta]$ is a sequence $\langle \mu_1, \mu_2, \ldots, \mu_m \rangle$ of membership functions $\mu_i$ defined on $A$. The pattern $P$ fuzzy matches an input sequence $t \in A^*$ if $t = ux_1x_2 \cdots x_nv$ such that

$$\frac{1}{m} \sum_{j=1}^{m} \mu_j(x_j) \in [\alpha, \beta], j = 1, 2, \ldots, m, 0 \leq \alpha \leq \beta \leq 1$$

for some $u, v \in A^*$.

We address the following problem: given a pattern $P[\alpha, \beta]$ and an input sequence $t$, find all positions in $t$ where $P$ fuzzy matches $t$. We present an on-line algorithm that solves the problem in general and give an efficient linear-time algorithm for some classes of patterns. The proposed algorithms can be used for efficient on-line pattern matching in digitized presentation of continuous reality such as, for example, digitized images and sounds, or noisy telemetric data.

Key-words: fuzzy sets, fuzzy pattern matching on sequences, fuzzy algorithms

1 Introduction

The important characteristic of many real-time business, engineering and medical applications is that they manage unconventional data such as images, sounds or temporal data that are stored in time series. Such applications are characterized by the extremely large volumes of data processed. However, many objects of real world such as images, sounds, temperatures etc. that used in such applications are continuous by their nature. For example, prices of stocks, ECGs, star light curves (to classify stars), etc. can be presented as time series. Continuous, by their nature objects, need to be discretized in some way to be presented in computers due to the finite presentation restriction of computer representation. Discretizing is a one-to-many process, that is, the same continuous object in the real world corresponds to many discretized presentation due to, for example, limited equipment resolution, noises or precise measurement problems. Efficient use of such data for analyses, prediction, and data mining requires implementation of operations available for conventional database systems such as searching, selecting, updating data in databases, etc. Since digitized presentations are approximate presentations of continuous objects, the traditional algorithms developed for discrete objects, for example texts, cannot always be used or are inefficient. Therefore, it is important to develop new algorithms that are fast, require little memory and limited buffering and can operate in real-time on digitized presentations of continuous objects.

Pattern matching is the problem of locating a specific pattern inside raw data. This is an important component of many tasks, including text editing, data retrieval, data compression, data min-
ing. Another application area is real-time monitoring [4,10] and event detection in manufacturing processes by examining noisy sensor data [11]. Formally, the pattern matching consists of finding one, or more generally, all the occurrences of a pattern inside sequential raw data. Raw data can be seen as a sequence over some finite or even infinite alphabet. The patterns are usually a collection of sequences described in some formal language. In addition to approximate presentations of continuous real-world objects, uncertainty and vagueness are usual in the human knowledge and reasoning. Therefore there are needs to handle such approximate and fuzzy data.

When row data are texts over finite alphabets the pattern matching problem is known as string-matching. Many solutions based on the use of automata or combinatorial properties of strings over finite alphabets have been proposed in the literature [5,6]. These problems and proposed solutions heavily depend on a finiteness property of the underlying alphabets and cannot be applied directly to solve the problems in the continuous setting where alphabets are infinite or very large and precise matching, as in digitized presentations, often is not necessary or even possible [9]. Therefore different approaches to handle efficiently imprecise fuzzy data have been studied in literature [2,3,7,12,13].

In this paper we study fuzzy pattern matching problem applied to sequences of data objects representing, for example, times series of digitized presentation of monitoring systems data. Since under digitization both a single pattern and row data can map into one-of-many digitized patterns and digitized row sequences (due to problems mentioned above) it is unlikely that an exact match can always be achieved. We present an on-line algorithm that solves the problem based on fuzzy set technology.

The paper is organized as follows. In Section 2 we provide the necessary definitions and notations. The main algorithm is presented in Section 3. General discussion of complexity of proposed algorithms is given in Section 4. Finally, concluding remarks are made in the last section.

2 Preliminaries

Let $A$ be an infinite set of elements, called an alphabet, and $A^*$ denote the set of all finite-length sequences of elements from $A$. The length of a finite sequence $t_n = a_1a_2\cdots a_n$ is denoted as $|t_n|$, that is, $|t_n| = n$. We assume that $t_0 = \varepsilon$, where $\varepsilon$ denotes the empty sequence. Let $A^m$ denote the set of all sequences of length $m$ formed using elements from $A$. We say that a sequence $w$ is a prefix of a sequence $t$ if $t = uwv$ for some sequence $v \in A^*$. Similarly, we say that a sequence $w$ is a suffix of a sequence $t$ if $t = uvw$ for some sequence $u \in A^*$. A sequence $w$ is a subsequence of a sequence $t$ if $t = uvw$ for some $u, v \in A^*$.

A fuzzy set $F, F \subseteq A$, is characterized by a membership function $\mu_F : A \rightarrow [0,1]$ where $\mu_F(a)$ represents the grade of membership of a from $A$ in the fuzzy set $F$.

Let $\mathcal{M}_A = \{\mu_1, \mu_2, \ldots\}$ be a set of membership functions defined on $A$. A pattern $P$ of length $m$ is a sequence $\langle \mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_m} \rangle$ of membership functions from $\mathcal{M}_A$. The pattern $P = \langle \mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_m} \rangle$ represents a fuzzy set $L_P$ of sequences from $A^m$, $L_P \subseteq A^m$, characterized by a membership function $\mu_{L_P} : A^m \rightarrow [0,1]$, with

$\mu_{L_P}(a_1a_2\cdots a_m) = \frac{1}{m} \sum_{j=1}^{m} \mu_{i_j}(a_j)$, where $a_j \in A$

If $P = \langle \mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_m} \rangle$ is a pattern of length $m$ and $0 \leq \alpha \leq \beta \leq 1$ then $P[\alpha, \beta]$ denotes a pattern that represents a fuzzy set $Q_P[\alpha, \beta]$, $Q_P[\alpha, \beta] \subseteq L_P$, defined as following:

$Q_P[\alpha, \beta] = \{a_1 \cdots a_m | \mu_{L_P}(a_1 \cdots a_m) \in [\alpha, \beta]\}$

where $a_{ij} \in A$.

We assume that $P$ denotes pattern $P[\alpha, \beta]$ where $\alpha = 0$ and $\beta = 1$.

We say that a pattern $P[\alpha, \beta] = \langle \mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_m} \rangle$ fuzzy matches a sequence $t_n = x_1x_2\cdots x_n$ from $A^*$, if a sequence $y_1y_2\cdots y_m$ from $Q_P[\alpha, \beta]$ occurs as a subsequence of $t$, that is, if $t = uy_1y_2\cdots y_mv$ for some $u, v \in A^*$ and $\mu_{L_P}(y_1y_2\cdots y_m) \in [\alpha, \beta]$. The pattern $P[\alpha, \beta]$ occurs beginning at position $k + 1$ in sequence $t_n = x_1x_2\cdots x_kx_{k+1}\cdots x_{k+m}\cdots x_n$ or matches $t$ in position $k + 1$ if

$x_{k+1} \cdots x_{k+m} \in Q_P[\alpha, \beta]$ where $0 \leq k \leq n - m$
In this paper we address the following fuzzy sequence matching problem:

Given a pattern $P [\alpha, \beta]$ from $\mathcal{M}_A$ and a sequence $t$ from $A^*$, find all positions in $t$ where $P [\alpha, \beta]$ fuzzy matches $t$.

Pattern recognition with fuzzy pattern $P [\alpha, \beta]$ from $\mathcal{M}_A$ on sequence $t$ can be performed by calculating the membership function $\mu_{L_P}$ for every subsequence $w$, $|w| = m$ of $t$ where $\mu_{L_P} (w) \leq \alpha$ means that $w$ does not match the fuzzy pattern $P [\alpha, \beta]$ to any degree, and $\mu_{L_P} (w) = \beta$ means that $w$ fully matches the fuzzy pattern $P [\alpha, \beta]$. However, we can design an efficient matching algorithm by optimizing calculation of $\mu_{L_P} (w)$.

3 Fuzzy pattern matching algorithm

In this section we present a general on-line algorithm for fuzzy pattern matching on sequences.

Efficient pattern recognition with fuzzy pattern $P [\alpha, \beta]$ from $\mathcal{M}_A$ can be done based on iterative calculations of a membership function $\mu_{L_P} (w)$ as explained in this section.

Consider a sequence $t_k$ of $k$ elements from $A$, that is $t_k = a_1 a_2 \cdots a_k \in A^*$, and a fuzzy pattern $P [\alpha, \beta] = \langle \mu_1, \mu_2, \ldots, \mu_m \rangle$ from $\mathcal{M}_A$. Let $S^P_k$ denote a set of lengths of all prefixes of $\langle \mu_1, \mu_2, \ldots, \mu_m \rangle$ that fuzzy matching some suffix of $t_k$, that is, $l \in S^P_k$ if and only if prefix $\langle \mu_1, \mu_2, \ldots, \mu_l \rangle$ of length $l \leq m$ fuzzy matches $u_l = a_{k-l+1} \cdots a_k$. We use $\mathcal{R}_l$ to denote the membership function corresponding to prefix $\langle \mu_1, \mu_2, \ldots, \mu_l \rangle$, that is,

$$\mathcal{R}_l (u_l) = \frac{1}{l} \sum_{r=1}^{l} \mu_{\vec{a}} (a_{k-l+r})$$

Let $\alpha_l$ and $\beta_l$ denote the lower and upper bounds of membership grades of $u_l$ such that within these bounds is still possible to find $v_{m-l} = a_{k+1} \cdots a_{k+m-l}$ such that $\mu_{L_P} (u_l v_{m-l}) \in [\alpha, \beta]$.

We say that the prefix $\langle \mu_1, \mu_2, \ldots, \mu_l \rangle$ of length $l \leq m$ fuzzy matches $u_l$ if $\mathcal{R}_l (u_l) \in [\alpha_l, \beta_l]$. Thus $P [\alpha, \beta] = \langle \mu_1, \mu_2, \ldots, \mu_m \rangle$ fuzzy matches an input $a_1 a_2 \cdots a_m$ if $\mathcal{R}_m (a_1 a_2 \cdots a_m) \in [\alpha_m, \beta_m]$ where $\alpha_m = \alpha$ and $\beta_m = \beta$.

Let us consider how to find $\alpha_l$ and $\beta_l$, $l = 1, 2, \ldots, m-1$ for given $P [\alpha, \beta]$. Suppose that $\mathcal{R}_l \in [\alpha_l, \beta_l]$ after analyzing $t = \cdots a_i a_{i+1} \cdots a_i$. Let $a_{i+1}$ be the next element of $t$ to be analyzed. Then, based on definition of $\mathcal{R}_l (u_l)$ we have

$$\mathcal{R}_{l+1} (a_i a_{i+1}) = \frac{l \mathcal{R}_l (a_i a_{i+1}) + \mu_{l+1} (a_{i+1})}{l+1} \quad \Rightarrow \quad (l+1) \mathcal{R}_{l+1} (a_i a_{i+1}) = l \mathcal{R}_l (a_i a_{i+1}) + \mu_{l+1} (a_{i+1})$$

and

$$m \mathcal{R}_m (a_i a_{i+1} \cdots a_m) = l \mathcal{R}_l (a_i a_{i+1} \cdots a_i) + \sum_{r=l+1}^{m} \mu_{l_r} (a_{i+r}) \quad \Rightarrow \quad \mathcal{R}_m (a_i a_{i+1} \cdots a_m) = \frac{l \mathcal{R}_l (a_i a_{i+1} \cdots a_i) + \sum_{r=l+1}^{m} \mu_{l_r} (a_{i+r})}{m}$$

It is no needs to continue fuzzy matching on the rest of the elements if even in the best case, that is with full match, $\mathcal{R}_m (a_i a_{i+1} \cdots a_m) \leq \alpha$ or even in the worst case, that is with full mismatch, $\mathcal{R}_m (a_i a_{i+1} \cdots a_m) \geq \beta$.

In the case of perfect match on the rest of $(m-k)$ elements holds

$$\mathcal{R}_m (a_i a_{i+1} \cdots a_m) = \frac{l \mathcal{R}_l (a_i a_{i+1} \cdots a_i) + (m-l)}{m}$$

If we claim that

$$\frac{l \mathcal{R}_l (a_i a_{i+1} \cdots a_i) + (m-l)}{m} \geq \alpha$$

then $\mathcal{R}_l$ must conform the following inequality:

$$\mathcal{R}_l (a_i a_{i+1} \cdots a_i) \geq \frac{m}{l} (\alpha - 1) + 1$$

Therefore

$$\alpha_l = \frac{m}{l} (\alpha - 1) + 1$$

In the case when even of the rest of $(m-k)$ elements mismatch the pattern, the following holds

$$\mathcal{R}_m (a_i a_{i+1} \cdots a_m) = \frac{l \mathcal{R}_l (a_i a_{i+1} \cdots a_i)}{m} \leq \beta$$
Algorithm MATCH($t, P[α, β]$)
Input: a sequence of data $t$ and a pattern $P[α, β] = \langle μ_{k_1}, μ_{k_2}, ..., μ_{km} \rangle$
Output: positions in $t$ where fuzzy matching occurs

1. $S \leftarrow \{0, 0\}$;
2. $pos \leftarrow 0$;
3. while input is not empty do
4.   $x \leftarrow$ read next element from $t$;
5.   $pos \leftarrow pos + 1$;
6.   $S \leftarrow$ UPDATE$(S, x, P[α, β])$;
7.   if $(m, \mathcal{R}) \in S$ then
8.     matching at $(pos - m + 1)$
9.     $S \leftarrow S \setminus \{(m, \mathcal{R})\}$
10. end.

Figure 1: Algorithm MATCH

and therefore $R_i$ must conform the following inequality:

$$R_i(a_1a_2\cdots a_i) \leq \frac{m}{l} \beta$$

That is

$$β_i = \frac{m}{l} \beta$$

Thus, $P[α, β]$ fuzzy matches $t_k$ if and only if $m \in S_j^P$ for some $j \leq k$. If we want to perform fuzzy pattern matching based on this, we have to construct a sequence of sets $S_0^P, S_1^P, ..., S_k^P$ for a given pattern $P[α, β] = \langle μ_{i_1}, μ_{i_2}, ..., μ_{im} \rangle$ and a sequence $t = a_1a_2\cdots a_k \cdots$, and then check whether $m$ is in $S_k^P$ for some $k > 0$. For any $S_k^P$ such that $m \in S_k^P$, we report that $P[α, β]$ matches $t$ in position $(k - m + 1)$.

The algorithm MATCH, based on the above idea, is presented in Fig. 1. The output of the algorithm MATCH($t, P[α, β]$) is the set of all positions of $t$ where $P[α, β]$ fuzzy matches $t$. The algorithm uses the procedure UPDATE$(S^P, x, P[α, β])$ to construct $S_i^P$ based on $S_{i-1}^P$ and $x$, where $x$ is a next element of sequence $t$.

Let us consider how the algorithm UPDATE works. The main purpose of UPDATE is to construct $S_0^P, S_1^P, ..., S_k^P$ for the given pattern $P[α, β] = \langle μ_{i_1}, μ_{i_2}, ..., μ_{im} \rangle$ and input sequence $t = a_1a_2\cdots a_k \cdots$. The algorithm UPDATE is presented in Fig. 2. According to Fig. 2, UPDATE calculates $S_{i+1}^P$ based on $S_i^P$, the next input element $x$ and the length of the prefix of $P$ that has been already matched.

The difference between the algorithm UPDATE and a naïve brute-force algorithm is that in UPDATE some unnecessary computations are removed from consideration. For each input element $x$ of input $t$, UPDATE calculates only those membership functions of the pattern $P[α, β] = \langle μ_{i_1}, μ_{i_2}, ..., μ_{im} \rangle$ that still may lead to some fuzzy matching of $P[α, β]$ in $t$. When a sequence of length $i$ has been analyzed, UPDATE contains in $S_i^P$ references on all membership functions that need to be evaluated for the next input $x$, that is, $j \in S_i^P$ refers on $μ_j$ from $P$.

However, the time complexity of UPDATE depends on properties of the pattern $P[α, β]$, that is, on both the complexity of evaluation of the membership functions and their interdependencies. The interdependencies presents knowledge of how values the membership functions of $P$ relate on each others.

Properties of UPDATE are summarized in Lemma 1.

**Lemma 1** Let $P[α, β] = \langle μ_{i_1}, μ_{i_2}, ..., μ_{im} \rangle$ be a pattern. Let $S_0^P, S_1^P, ..., S_k^P$ be a sequence of sets generated by algorithm UPDATE based on input $t = x_1x_2\cdots x_k \cdots$ such that $S_0^P = \{(0, 0)\}$ and $S_k^P =$ UPDATE$(S_{k-1}^P, x_k, P)$. Then $S_k^P$ satisfies the
Algorithm MATCH finds on-line all positions in $t$ where $P[\alpha, \beta]$ fuzzy matches $t$.

Proof. Assume that MATCH processes input $t = x_1 x_2 \cdots x_{pos} \cdots$. We prove correctness by induction on $pos$.

First, the algorithm MATCH works correctly for the empty sequence $t_0$.

Suppose that MATCH has performed correctly for all $pos = 1, 2, \ldots, k$, that is, MATCH has found all positions in $t = x_1 x_2 \cdots x_k$ where $P[\alpha, \beta]$ fuzzy matches $t$. As follows from the algorithm (Fig. 1) (line 9), $S$ is equal to $S^P_k \setminus \{\langle m, \mathcal{R} \rangle \}$. Thus, according to Lemma 1, $S$ contains lengths of all proper prefixes of $P$ that fuzzy match some suffixes of $t_k$.

Let us show that MATCH works correctly for $pos = k + 1$.

Assume that MATCH has been processed an input $t_k = a_1 a_2 \cdots a_k$, $k > 0$. When the next element $x = a_{k+1}$ has been received (line 4), MATCH calls UPDATE($S, x, P[\alpha, \beta]$) in order to update $S$ with respect to $x$. By updating $S$ in line 6 we guarantee that, according to the inductive hypothesis and Lemma 1, $S$ contains lengths of all prefixes of $P$ that match some suffixes of $t_{k+1}$. Therefore, $m$ will be in $S$ if and only if $P[\alpha, \beta]$ matches $a_k + 1 a_{k+1} a_{k+2} a_{k+3} \cdots a_{k+1}$.

Thus, at any time when $m \in S$, the pattern is found (lines 7, 8). This concludes the proof. □

The time complexity of algorithm MATCH is linear in the length of input plus the time complexity of all UPDATE’s calls. We discuss the complexity of MATCH in Section 4. □

4 Complexity

In this section we consider briefly the complexity of the algorithm MATCH presented in Section 3.

The naive brute force algorithm consists of checking the input sequence $t$, $|t| = n$, at all positions of between 0 and $n - m$ whether an occurrence of the fuzzy pattern $P[\alpha, \beta] = \langle \mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_m} \rangle$ begins there or not. Then, after each attempt, it shifts the pattern exactly one position to the right. If membership functions can be evaluated in constant time, the complexity of such a brute force algorithm is $O(nm)$ both in the worst and average cases. However, by taking into consideration the history of matching to avoid unnecessary the actual time can be reduced, since not all membership functions of the pattern need to be evaluated after each shift. When membership functions $\mu_{i_1}, \mu_{i_2}, \ldots, \mu_{i_m}$
in a pattern \( P \) are interdependent, the time complexity can be improved even further. Collecting historical data in the course of matching we can use such interdependency to improve performance of such algorithms. The algorithm \textsc{Match} presented in this paper collects data dynamically and avoid evaluation of useless cases. The time complexity of the algorithm \textsc{Match} depends on properties of interdependencies of membership functions of \( P[\alpha, \beta] \).

5 Conclusion

We have presented a new algorithm for the fuzzy pattern matching problem over infinite alphabets where patterns are presented as sequences of fuzzy constraints defined on elements of some, generally, infinite alphabet. Our scheme permits on-line fuzzy matching on the input representing digitized continuous reality such as sounds, images or even noisy telemetric data without knowing \textit{a priori} either the whole sequence or the whole pattern. Another example when our algorithms can be efficiently applied is the problem of finding the longest fuzzy prefix of patterns that occurs within an input sequence. If the pattern is long, preprocessing can be costly and unnecessary if only relatively short prefixes will occur in input.

Further work should be done to find new classes of membership functions with sound applications for which linear-time algorithms exist. It would be interesting to analyze performance proposed algorithm and compare it with other known approaches when such exist.

References


