Problems regarding fuzzy string processing

Author: Arevik Harmandayan
Instructor: Armen Kostanyan

Project goals

The project aimed to investigate:

Fuzzyfied string dotted matching

- Matching a string with a fuzzy pattern
- Fuzzy matching a string with a pattern

Fuzzyfied string distance

- Determining the distance between a string and a fuzzy pattern
- Determining the distance between 2 strings using fuzzy matching

Remarks

- All of the algorithms presented in this thesis use the **dynamic programming** approach.
- These problems can be viewed as specific cases of approximate string matching.

Plan of presentation

Introduction

Fuzzy Logic, Linguistic variables

Fuzzified string dotted matching

Fuzzified string distance

Conclusion

Introduction

Applications

- Text searching
- Text editing
- Spell checking
- Context search
- Querying specific rows from a database
- Data compression
- Spam filtering
- Matching of nucleotide sequences

State of art

Distance based

- Levenshtein distance
- Damerau-Levenshtein distance
- Jaro–Winkler distance
- Hamming distance

Expression based

- Needleman–Wunsch algorithm
- Smith–Waterman algorithm

Fuzzy string matching with finite automata

Fuzzy Logic and Linguistic variables

Fuzzy logic

Fuzzy logic is a form of many-valued logic in which the truth values of the variables may be any real number between 0 and 1.

Fuzzy set

A fuzzy set is a set whose elements have ***grades of membership***. It is characterized by a ***membership function*** which assigns each element a grade of membership ranging between 0 and 1.

Fuzzy logic

Example: Let the universe of discourse be the interval (-∞, ∞), with ***u*** interpreted as ***temperature***. A fuzzy subset of ***U*** labeled ***high*** may be defined by a membership function such as

$\mu_A(u)=0$	for $u < 20$
$\mu_A(u)=(u-20)/20$	for $20 \leq u \leq 40$
$\mu_A(u)=1$	for $u > 40$

Linguistic variables

Linguistic variables are variables whose values are words or sentences in a natural or artificial language.

For example, **temperature** is a linguistic variable if its values are linguistic, i.e., high, very high, low, not low, extremely high, not very low, etc., rather than numeric, 40, 46, -10, 7, 78, 2, etc.

Linguistic variable is characterized by a quintuple **(L, T(L), U, G, M)**

Linguistic variables

(L, T(L), U, G, M)

L is the name of the variable

T*(*L*)** is the term-set of ***L , that is, the collection of its linguistic values;

U is a universe of discourse;

G is a syntactic rule which generates the terms in ***T*(*L*)**;

M is a semantic rule which associates with each linguistic value **X** its meaning, ***M*(X)**, where ***M*(X)** denotes a fuzzy subset of ***U***.

So the compatibility of 28 degrees with ***high*** might be 0,4, while that of 34 might 0,7.

Fuzzy patterns and fuzzy matching

In the first case of each of our problems we are going to match the string with a fuzzy pattern (pattern consisting of values of linguistic variables).

In the second case of each of our problems we are going to use fuzzy matching between the elements of the string with the elements of the pattern.

We are going to use the features of fuzzy logic and linguistic variables.

Fuzzified string dotted matching

Pattern dotted matching

String: abbdcaabbdccabcaabc

Pattern: bcabbcacab

This is a special case of **longest common subsequence** problem, when the length of the LCS should be equal to the length of the given pattern.

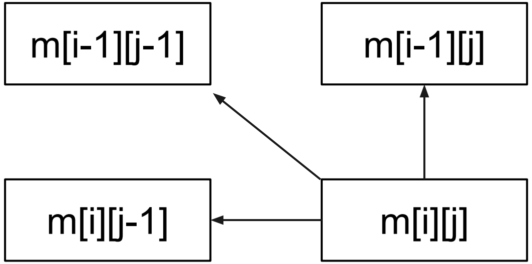
Longest Common Subsequence

```
LCS-LENGTH(X, Y)
m=X.length
n=Y.length
let b[1..m,1..n] and c[0..m,0..n] be new tables
for i = 1 to m
    c[i,0] = 0
for j = 0 to n
    c[0,j] = 0
for i = 1 to m
    for j = 1 to n
        if x_i == y_j
            c[i,j] = c[i-1,j-1] + 1
            b[i,j] = "↗"
        elseif c[i-1,j] ≥ c[i,j-1]
            c[i,j] = c[i-1,j]
            b[i,j] = "↑"
        else c[i,j] = c[i,j-1]
            b[i,j] = "←"
return c and b
```

```
PRINT-LCS(b,X,i,j)
if i == 0 or j == 0
        return
if b[i,j] == "↗"
        PRINT-LCS(b,X,i-1,j-1)
        print x_i
elseif b[i,j] = "↑"
        PRINT-LCS(b,X,i-1,j)
else PRINT-LCS(b,X,i,j-1)
```

Longest Common Subsequence

i	j	0	1	2	3	4	5	6
	y_j		B	D	C	A	B	A
0	x_i	0	0	0	0	0	0	0
1	A	0	↑ 0	↑ 0	↑ 0	↖ 1	← 1	↖ 1
2	B	0	↖ 1	← 1	← 1	↑ 1	↖ 2	← 2
3	C	0	↑ 1	↑ 1	↖ 2	← 2	↑ 2	↑ 2
4	B	0	↖ 1	↑ 1	↑ 2	↑ 2	↖ 3	← 3
5	D	0	↑ 1	↖ 2	↑ 2	↑ 2	↑ 3	↑ 3
6	A	0	↑ 1	↑ 2	↑ 2	↖ 3	↑ 3	↖ 4
7	B	0	↖ 1	↑ 2	↑ 2	↑ 3	↖ 4	↑ 4



1. If $x_m = y_n$, then $z_k = x_m = y_n$ and Z_{k-1} is an LCS of X_{m-1} and Y_{n-1}.
2. If $x_m \neq y_n$, then $z_k \neq x_m$ implies that Z is an LCS of X_{m-1} and Y.
3. If $x_m \neq y_n$, then $z_k \neq y_n$ implies that Z is an LCS of X and Y_{n-1}.

Fuzzy string dotted matching

Given string of length **n** over input alphabet **Σ**

Pattern of length **m** over set of values of a linguistic variable

A threshold **T**

Find a dotted occurrence of **P** in **S**

Example: **S = 2241**

P=small large small.

Fuzzy string dotted matching

We take a matrix matrix[n][m]. Every element of the matrix holds a reference to an array. The length of the arrays are distributed like this:

1	1	1	1	1
1	2	2	2	2
1	2	3	3	3
...	...	...	...	...
1	2	3	...	m

Fuzzy string dotted matching

This is of how the arrays are filled:

$$
m[i][j][h].weight=\begin{cases}
1 & \text{if } i=0 \text{ and } h=0, \text{or } j=0 \text{ and } h=0 \\
m[i-1][j-1][h-1].w*weight(a[i],b[j]) & \text{if } i=j=h \text{ and } i,j>0 \\
max(m[i-1][j][h].w,m[i][j-1][h].w,m[i-1][j-1][h-1].w*weight(a[i],b[j])) & \\
 & \text{if } i<=j,h<j \text{ and } i,j<0 \text{ or } i>=j,h<i \text{ and } i,j<0 \\
max(m[i-1][j][h].w,m[i-1][j-1][h-1].w*weight(a[i],b[j])) & \\
 & \text{if } i>j,h>=i \text{ and } i,j<0 \\
max(m[i][j-1][h].w,m[i-1][j-1][h-1].w*weight(a[i],b[j])) & \\
 & \text{if } i<j,h>=j \text{ and } i,j<0
\end{cases}
$$

If we take m[i-1][j][h], m[i][j][h].dir = ‘⬆’, if we take m[i][j-1][h] m[i][j][h].dir = ‘⬅’, else ‘⬈’.

Fuzzy string dotted matching

The string is **2241** and the pattern is **small large small**.

		small	large	small
	1	1	1	1
2	1	1 ↖ 0,75	1 ← 0,75	1 ↖ 0,75
2	1	1 ↖ 0,75	1 ← 0,75 2 ↖ 0,1875	1 ↖ 0,75 2 ↖ 0,5675
4	1	1 ↑ 0,75	1 ↖ 0,75 2 ↖ 0,5675	1 ← 0,75 2 ↑ 0,5675 3 ↖ 0,046875
1	1	1 ↖ 1	1 ← 1 2 ↑ 0,5675	1 ← 1 2 ↖ 0,75 3 ↖ 0,5675

Test Cases

```
Enter dimentions
4
3
Enter the string of numbers
2
2
4
1
Enter the string of linguistic variable string
s
l
s

0.5625
```

```
Enter dimentions
4
3
Enter the string of numbers
4
2
5
2
Enter the string of linguistic variable string
l
s
l

0.5625
```

```
Enter dimentions
5
4
Enter the string of numbers
2
5
4
1
3
Enter the string of linguistic variable string
s
l
s
s

0.375
```

```
Enter dimentions
5
3
Enter the string of numbers
5
3
2
4
4
Enter the string of linguistic variable string
s
s
l

0.28125
```

Exact String, Exact Pattern, fuzzy matching

Given string S of length **n** over input alphabet **Σ**

Pattern P of length **m** over input alphabet **Σ**

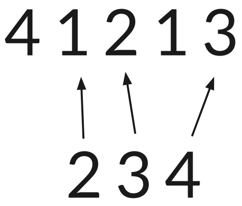
Every element of **Σ** has a corresponding threshold **T(s)**

Find a dotted, fuzzy matched to the P so that every element of the S is matched to P with the difference smaller than the corresponding T(s)

We need to mention that for this problem the elements of S and P must be from the the universe of discourse of the same linguistic variable.

Exact String, Exact Pattern, fuzzy matching

Example: S = <4,1 2,1,3 > and P = <2, 3, 4>, T = <0.25, 0.5, 0.5, 0.25, 0.25> . The linguistic variable is ***size***.



FLCS-LENGTH(X, Y)

```
n=S.length
m=P.length
let b[1..n,1..m] and c[0..n,0..m] be new tables
for i = 1 to n
      c[i,0] = 0
for j = 0 to m
      c[0,j] = 0
for i = 1 to n
      for j = 1 to m
         if FUZZY_MATCH(s_i, p_j,T(s_i))
               c[i,j] = c[i-1,j-1] + 1
               b[i,j] = "↗"
         elseif c[i-1,j] ≥ c[i,j-1]
               c[i,j] = c[i-1,j]
               b[i,j] = "↑"
         else c[i,j] = c[i,j-1]
               b[i,j] = "←"
return c and b
```

FUZZY_MATCH(s, p, d)

```
for i = 1 to |T(L)|   //T(L) is the the collection of linguistic values of L
      if (absolute(μ_i(a) - μ_i(b)) ≤ d)
      return true
return false
```

Exact String, Exact Pattern, fuzzy matching

Example: A = <4,1 2,1,3 > and B = <2, 3, 4>, T = <0.25, 0.5, 0.5, 0.25, 0.25> . The linguistic variable is ***size***.

	b_j	2	3	4
a_i	0	0	0	0
4	0	↑ 0	↖ 1	↖ 1
1	0	↖ 1	↑ 1	↑ 1
2	0	↖ 1	↖ 2	↖ 2
1	0	↖ 1	↑ 2	↑ 2
3	0	↖ 1	↖ 2	↖ 3

Fuzzified string distance

String distance algorithm

We can express the properties of d in terms of edit operations

performed on single (nonemply) letters λ and μ:

- (insert: replace ε by λ) $d(\varepsilon,\lambda) > 0$;
- (delete: replace λ by ε) $d(\lambda,\varepsilon) > 0$;
- (substitute: replace λ by μ) $d(\lambda,\mu) > 0$ iff $\lambda \neq \mu$;

For $d = d_L$ or d_E, $d(\lambda,\varepsilon) = d(\varepsilon,\lambda) = 1$ for all λ; while for $d = d_L$, $d(\lambda,\varepsilon) = 2$ and for
$d = d_E$, $d(\lambda,\mu) = 1$.

String distance algorithm

Example: 2 7 6 5 8 4 2 6 7

1 7 5 8 3 2 6

We need 2 edit and 2 delete operations.

So the $d_E=4$

Lemma

For every $i \in 0..n_1, j \in 0..n_2$

$$c[i,j] = \min\{ c[i-1,j] + d(x_1[i],\varepsilon), c[i,j-1] + d(\varepsilon,x_2[h]), c[i-1,j-1] + d(x_1[i],x_2[h]) \}$$

Determining the distance between a string and a fuzzy pattern

Given string **S** of length **n** over input alphabet **Σ**

Pattern **P** of length **m** over set of values of a linguistic variable

A threshold **T**

Find the distance between **S** and **P**

No delete or insert will be needed when the element of the string matched the pattern with a degree of membership that is greater than a given threshold **T** so the distance will be **0**.

Determining the distance between a string and a fuzzy pattern

In this case also 3 operations can be done:

- (insert: replace ε by λ) $d(\varepsilon,\lambda) > 0$;
- (delete: replace λ by ε) $d(\lambda,\varepsilon) >0$;
- (substitute: replace λ by τ) $d(\lambda,\tau) > 0$ iff $\boldsymbol{\mu}_\tau(\lambda)<T$

Example: **2 2 3 1** **T = 0,75** **2 2 3 1**

small large small **1 5 1**

In this case 1 delete and 1 edit are needed, so the distance is 2.

Determining the distance between 2 strings using fuzzy matching

Given string **S** of length **n** over input alphabet **Σ**

Pattern **P** of length **m** over input alphabet **Σ**

Every element of **Σ** has a corresponding threshold **T(s)**

Find the distance between **S** and **P** using fuzzy matching

We need to mention that for this problem the elements of **S** and **P** must be from the the universe of discourse of the same linguistic variable.

Determining the distance between 2 strings using fuzzy matching

The operations that can be done are:

- (insert: replace ε by s_i) $d(\varepsilon, s_i) > 0$;
- (delete: replace s_i by ε) $d(s_i, \varepsilon) > 0$;
- (substitute: replace s_i by p_j) $d(s_i, p_j) > 0$ iff !FUZZY_MATCH(s_i, p_j, $T(s_i)$)

Example: S = 2 4 1 5 4 3 3 let's take for $\forall s\ T(s) = 0.25$

P = 3 1 4 3 1 2 5

In this case 1 insert, 1 delete and 1 edit are needed. Do the distance is 3.

In case of exact matching 2 insert, 2 delete and 2 edits would be needed.

Conclusion

Conclusion

This thesis presented the solutions to the problems:

Fuzzyfied string dotted matching

- Matching a string with a fuzzy pattern
- Fuzzy matching a string with a pattern

Fuzzyfied string distance

- Determining the distance between a string and a fuzzy pattern
- Determining the distance between 2 strings using fuzzy matching

All of the algorithms were using the **dynamic programming** approach.

References

[1] Cormen, Thomas H., Charles Eric Leiserson, and Ronald L. Rivest. Introduction to Algorithms. 3rd ed. Cambridge, Mass. : MIT Press, 2009.

[2] Smyth W. Computing Patterns in Strings. Pearson/Addison-Wesley, 2003

[3] L. A. Zadeh, "Fuzzy sets," *Information and Control*, vol. 8, no. 3, pp. 338-353, June 1965.

[4] L. A. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning - I," Information Sciences, vol. 8, no. 3, pp. 199-249, July 1975.

[5] Novák, V., Perfilieva, I. and Močkoř, J., “Mathematical Principles of Fuzzy Logic”, Kluwer Academic Publishers 1999.Print

[6] A. Kostanyan. Fuzzy String Matching with Finite Automata, in Proceedings on 2017 IEEE Conference "2017 Computer Science and Information Technologies" (CSIT), Yerevan, Armenia, 25-29 Sep., 2017, IEEE Press, USA, 2018, pp. 9 - 11.

Thank You