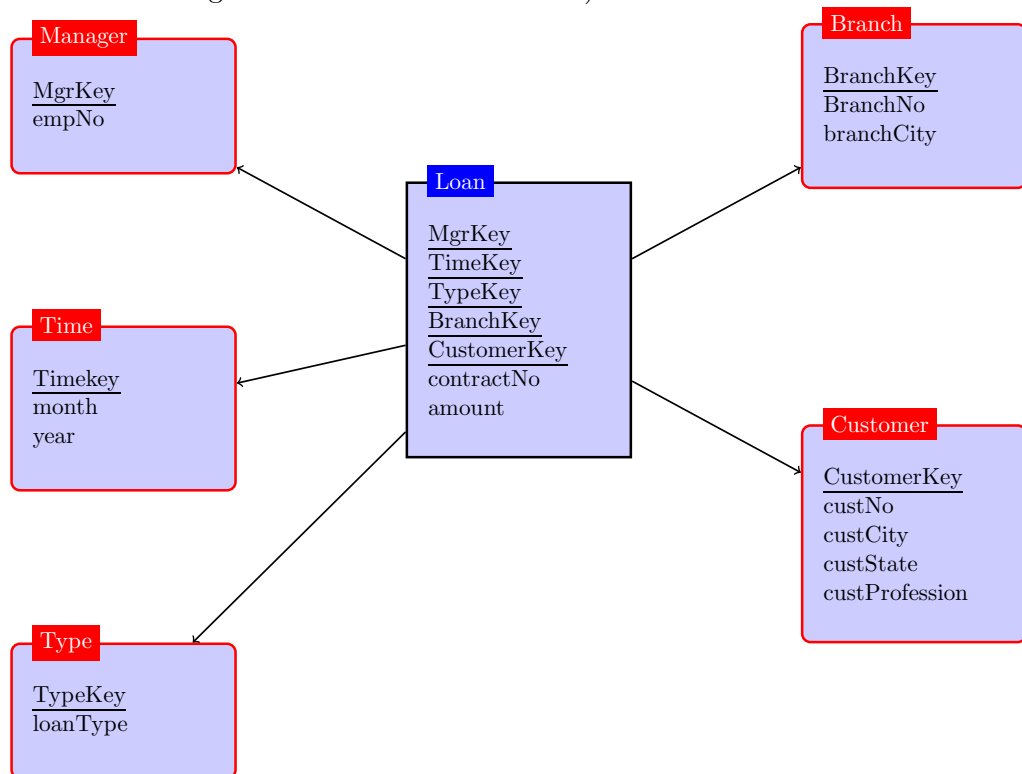


Solution to COMP9318 Assignment 1

Q1

(Note that there might be other correct solutions)



Q2

1

1	{ <i>aba</i> , <i>bab</i> , <i>aba</i> ₂ }
2	{ <i>abc</i> , <i>bca</i> , <i>cab</i> , <i>aba</i> }
3	{ <i>aaa</i> , <i>aac</i> , <i>aca</i> , <i>cab</i> , <i>abc</i> }
4	{ <i>abc</i> , <i>bcd</i> , <i>cde</i> , <i>def</i> , <i>efg</i> , <i>fgh</i> }

Note that we treat the same token x appearing multiple times in the same set as a (slightly) different token. c.f., the aba and aba_2 in the first string.

2 We calculate the document frequencies (df) for all the unique tokens (i.e., 3-grams here).

Token	<i>aaa</i>	<i>aac</i>	<i>aba</i>	<i>aba₂</i>	<i>abc</i>	<i>aca</i>	<i>bab</i>
<i>df</i>	1	1	2	1	3	1	1
Token	<i>bca</i>	<i>bcd</i>	<i>cab</i>	<i>cde</i>	<i>def</i>	<i>efg</i>	<i>fgh</i>
<i>df</i>	1	1	2	1	1	1	1

Hence, a possible global ordering (from most rare to the most frequent token) is

aaa, aac, aba₂, aca, bab, bca, bcd, cde, def, efg, fgh,
aba, cab,
abc

The prefix length should be $qd + 1 = 4$. Therefore, the prefixes for the strings are:

1	UNDERFLOW
2	{ <i>bca, aba, cab, abc</i> }
3	{ <i>aaa, aac, aca, cab</i> }
4	{ <i>bcd, cde, def, efg</i> }

Note that the first string does not have enough number of q-grams for its prefix. This means the string needs to be compared with **all** other strings.

We thus run the prefix filtering-based similarity join algorithm (with slight modification) as follows:

1. We consider the first string, and add it to a special underflow set. Every string needs to be compared with strings in the underflow set.
2. We consider the second string. First it is compared with the underflow strings; calculating $ed(s_1, s_2) = 1$ and thus it is an answer. Since the current inverted index is empty, we do not need to find candidates for s_2 . We then index all the prefix 3-grams of s_2 , i.e., $idx[bca] = idx[aba] = idx[cab] = idx[abc] = s_2$.

```

      a b c a b a
0    1 2 3 4 5 6
a:  1 0 1 2 3 4 5
b:  2 1 0 1 2 3 4
a:  3 2 1 1 1 2 3
b:  4 3 2 2 2 1 2
a:  5 4 3 3 2 2 1

```

3. We consider the third string. First it is compared with the underflow strings; calculating $ed(s_1, s_3) = 4$ and it is not an answer. We then probe the inverted index for all s_3 's prefix, and obtain s_2 as the candidates (because $idx[cab] = s_2$). We then verify $ed(s_2, s_3) = 3$ and it is not an answer. We then index all the prefix 3-grams of s_3 , i.e., $idx[aaa] = idx[aac] = idx[aca] = s_3$ and $idx[cab] = \{s_2, s_3\}$.

```

      a a a c a b c
      0 1 2 3 4 5 6 7
a: 1 0 1 2 3 4 5 6
b: 2 1 1 2 3 4 4 5
a: 3 2 1 1 2 3 4 5
b: 4 3 2 2 2 3 3 4
a: 5 4 3 2 3 2 3 4

```

```

      a a a c a b c
      0 1 2 3 4 5 6 7
a: 1 0 1 2 3 4 5 6
b: 2 1 1 2 3 4 4 5
c: 3 2 2 2 2 3 4 4
a: 4 3 2 2 3 2 3 4
b: 5 4 3 3 3 3 2 3
a: 6 5 4 3 4 3 3 3

```

4. We consider the third string. First it is compared with the underflow strings; calculating $ed(s_1, s_4) = 6$ and it is not an answer. We then probe the inverted index for all s_4 's prefix, and the candidate set is empty.

```

      a b c d e f g h
      0 1 2 3 4 5 6 7 8
a: 1 0 1 2 3 4 5 6 7
b: 2 1 0 1 2 3 4 5 6
a: 3 2 1 1 2 3 4 5 6
b: 4 3 2 2 2 3 4 5 6
a: 5 4 3 3 3 3 4 5 6

```

Therefore, the final answer is (s_1, s_2) .

Note: if we use length filtering, we do not need to calculate edit distance for (s_1, s_3) and (s_1, s_4)

3

$$\begin{aligned}
 ned(s, t) &= \frac{ed(s, t)}{\max(|s|, |t|)} \\
 &= \frac{ed(s, t)}{|t|} && \text{(because } |s| \leq |t| \text{)} \\
 &\geq \frac{|t| - |s|}{|t|} && \text{(because } ed(s, t) \geq |t| - |s| \text{)}
 \end{aligned}$$

Note that the equality is achievable (e.g., when $t = s \circ w$).

Since we know $ned(s, t) \leq \alpha$, we must have

$$\frac{|t| - |s|}{|t|} \leq \alpha$$

The above is equivalent to $|t| \leq \frac{1}{1-\alpha}|s|$.

Q3

1. Original similarity matrix can be rewritten into

	p_1	p_2	p_3	p_4	p_5
p_1		0.10	0.41	0.55	0.35
p_2			0.64	0.47	0.98
p_3				0.44	0.85
p_4					0.76
p_5					

2. The largest similarity is 0.98. Thus we merge p_2 and p_5 into p_{25} . Update the matrix as

	p_1	p_{25}	p_3	p_4
p_1		0.225	0.410	0.550
p_{25}			0.745	0.615
p_3				0.440
p_4				

3. The largest similarity is 0.745. Thus we merge p_{25} and p_3 into p_{235} . Update the matrix as

	p_1	p_{235}	p_4
p_1		0.287	0.550
p_{235}			0.557
p_4			

4. The largest similarity is 0.557. Thus we merge p_{235} and p_4 into p_{2345} . Update the matrix as

	p_1	p_{2345}
p_1		0.352
p_{2345}		

(the matrix is optional here)

5. Finally, we merge all into one cluster.

Therefore the final result is:

