School of Computing and Information Systems
The University of Melbourne
COMP90049
Knowledge Technologies (Semester 1, 2018)
Workshop sample solutions: Week 4

Suppose that we have observed the token `lended`, and we have a dictionary as follows:

```
addendum
blenders
commodity
deaden
end
leader
leant
lent
lemonade
pleading
```

1. With respect to the input string `lended` and the dictionary entry `deaden`, calculate the following:

   (a) the Global Edit Distance, using the parameter $[m, i, d, r] = [+1, -1, -1, -1]$

(a)	ε		l		e		n		d		e		d
ε	0	←	-1	←	-2	←	-3	←	-4	←	-5	←	-6
	↑	↖		↖		↖		↖				↖	
d	-1		-1	←	-2	←	-3		-2	←	-3	←	-4
	↑	↖	↑	↖						↖			
e	-2		-2		0	←	-1	←	-2		-1	←	-2
	↑	↖	↑		↑	↖		↖			↑	↖	
a	-3		-3		-1		-1	←	-2		-2		-2
	↑	↖	↑		↑	↖	↑	↖				↖	
d	-4		-4		-2		-2		0	←	-1		-1
	↑	↖	↑	↖	↑	↖	↑		↑	↖			
e	-5		-5		-3		-3		-1		1	←	0
	↑	↖	↑		↑	↖			↑		↑	↖	
n	-6		-6		-4		-2		-2		0		0

   - From the table above, we can observe that the Global Edit Distance is 0, corresponding to the following sequence of operations: Replace, Match, Replace, Match, Match, Replace, which I will abbreviate as `rmrmmr`. (You can follow along with the highlighted back-pointers.)

   (b) the Local Edit Distance, using the parameter $[m, i, d, r] = [+1, -1, -1, -1]$

   - From the table overleaf, we can observe that the Local Edit Distance is 2 (highlighted); there are five equivalent-scoring substring matches that it corresponds to:
     - Align `-de-` in `lended` with the first `de-` in `deaden`: `mm`
     - Align `-ded` with `dead-`: `mmim`
     - Align `-de-` in `lended` with the second `-de-` in `deaden`: `mm`
     - Align `-ende-` with `-eade-`: `mrmm`
     - Align `-en-` with `-en`: `mm`

(b)	ε	l	e	n	d	e	d
ε	0	0	0	0	0	0	0
d	0	0	0	0	1	0	1
e	0	0	1	0	0	2	1
a	0	0	0	0	0	1	1
d	0	0	0	0	1	0	2
e	0	0	1	0	0	2	1
n	0	0	0	2	1	1	1

(c) the N-Gram Distance, using $n = 2$

- We begin by generating the 2-grams of the two strings; I will use the terminal marker (#) here:
  - `lended`: `#l`, `le`, `en`, `nd`, `de`, `ed`, `d#`
  - `deaden`: `#d`, `de`, `ea`, `ad`, `de`, `en`, `n#`
- Recall that the N-Gram Distance is defined as follows:

$$D(s,t) = \mid G_n(s) \mid + \mid G_n(t) \mid -2\times \mid G_n(s) \cap G_n(t) \mid$$

- Here we have 7 2-grams in `lended`, as well as 7 in `deaden`. Also, the two sets share 2 2-grams: `de` and `en`. (Note that we don't double-count the `de`s in `deaden`, because there is only a single `de` in `lended`)
- Consequently, the 2-gram Distance is $7 + 7 - 2 \times 2 = 10$

2. Find the best approximate match (or matches, if there are ties) in the dictionary for the string `lended`, based on the following methods; consider different parameters where necessary:

(a) the Global Edit Distance

- Using the above scoring parameter, the closest matches are `blenders` (+2) and `leader` (+2)
- You might like to try some other parameter setting(s), to see if they give different results.

(b) the Local Edit Distance

- Using the above scoring parameter, the closest match is `blenders` (+5)
- In this case, changing the parameter is unlikely to result in a different answer. (Why?)

(c) the N-Gram Distance

- If we are using n is 2 and padding with #, the best dictionary entry is `end`, with a 2-Gram Distance of 5.
- You might find that removing the padding characters or changing n will give different results.

(d) Soundex

- The Soundex code of `lended` is `l533`.
- None of the dictionary entries have this exact code; however, if we permit mismatches in the Soundex code, then the best matches are `commodity` (`c533`), `leant` (`l53`), `lent` (`l53`), and `lemonade` (`l553`)

3. Assuming that the "correct" (intended) dictionary entry was `lent`, calculate the precision of the following methods of finding approximate entries from the dictionary.

   (a) Neighbourhood search, with a neighbourhood of 1

   - There we're any results returned from the dictionary, so precision isn't well-defined ($\frac{0}{0}$)

   (b) Neighbourhood search, with a neighbourhood of 2

   - There was one entry returned from the dictionary (`leader`), but it wasn't `lent`, so the precision is $\frac{0}{1} = 0$.

   (c) Neighbourhood search, with a neighbourhood of 3

   - There were five entries returned from the dictionary, and `lent` was one of them. The precision of this system is the number of correct responses (1) out of the total number of attempted responses (5), $\frac{1}{5} = 20\%$

   (d) Global Edit Distance, with a parameter $[m, i, d, r] = [1, -1, -1, -1]$

   - There were two (tied) results from the dictionary (`blenders` and `leader`), but no `lent`, so the precision is $\frac{0}{2} = 0$

   (e) Local Edit Distance, with a parameter $[m, i, d, r] = [1, -1, -1, -1]$

   - There was just a single result (`blenders`) which wasn't `lent`, so the precision is 0

   (f) N-gram Distance, where n is 2 (and padding with terminals)

   - There was a single result which wasn't `lent`, so the precision is $\frac{0}{1}$

   (g) Using the Soundex transformation, and then looking for exact matches

   (h) Using the Soundex transformation, and then permitting a 1-neighbourhood

   - There weren't any exact matches with the Soundex code of `lended`, so precision isn't well defined
   - Allowing approximate matches of the Soundex code meant that there were four results, including `lent`, so the precision is $\frac{1}{4} = 25\%$