Natural Language Processing
Lab 1.2

This lab sheet is to practice the concepts taught so far with a focus on byte pair encoding and minimum edit distance.

1. What is tokenization in the context of natural language processing?

Ans: Tokenization is broadly the process of breaking down text into smaller units, such as words or subwords.

2. Explain Byte Pair Encoding with an example.

Ans: BPE is a method for creating a subword vocabulary from a corpus. It repeatedly merges the most frequent pair of characters or character sequences e.g:

corpus='aaabdaaabac'

After several iterations of BPE we might get tokens like: a, b, c, d, aa, aaa, ab

3. Create a basic Byte Pair Encoding algorithm in Python that will work with a test corpus.

Ans:

```python
import re, collections

def get_stats(vocab):
    pairs = collections.defaultdict(int)
    for word, freq in vocab.items():
        symbols = word.split()
        for i in range(len(symbols)-1):
            pairs[symbols[i],symbols[i+1]] += freq
    return pairs

def merge_vocab(pair, v_in):
    v_out = {}
    bigram = re.escape(' '.join(pair))
    p = re.compile(r'(?<!\S)' + bigram + r'(?!\S)')
    for word in v_in:
        w_out = p.sub(''.join(pair), word)
        v_out[w_out] = v_in[word]
    return v_out

vocab = {'l o w </w>' : 5, 'l o w e r </w>' : 2,
'n e w e s t </w>':6, 'w i d e s t </w>':3}
num_merges = 10

for i in range(num_merges):
    pairs = get_stats(vocab)
    best = max(pairs, key=pairs.get)
    vocab = merge_vocab(best, vocab)
    print(best)
```

4. What is a minimum edit distance and its significance in NLP?

Ans: Minimum edit distance is the minimum number of operations (insertions, deletions, or substitutions) required to change one word into another. It is used in NLP for tasks like spell checking, speech recognition, and machine translation to find the closest word(s) to a given input

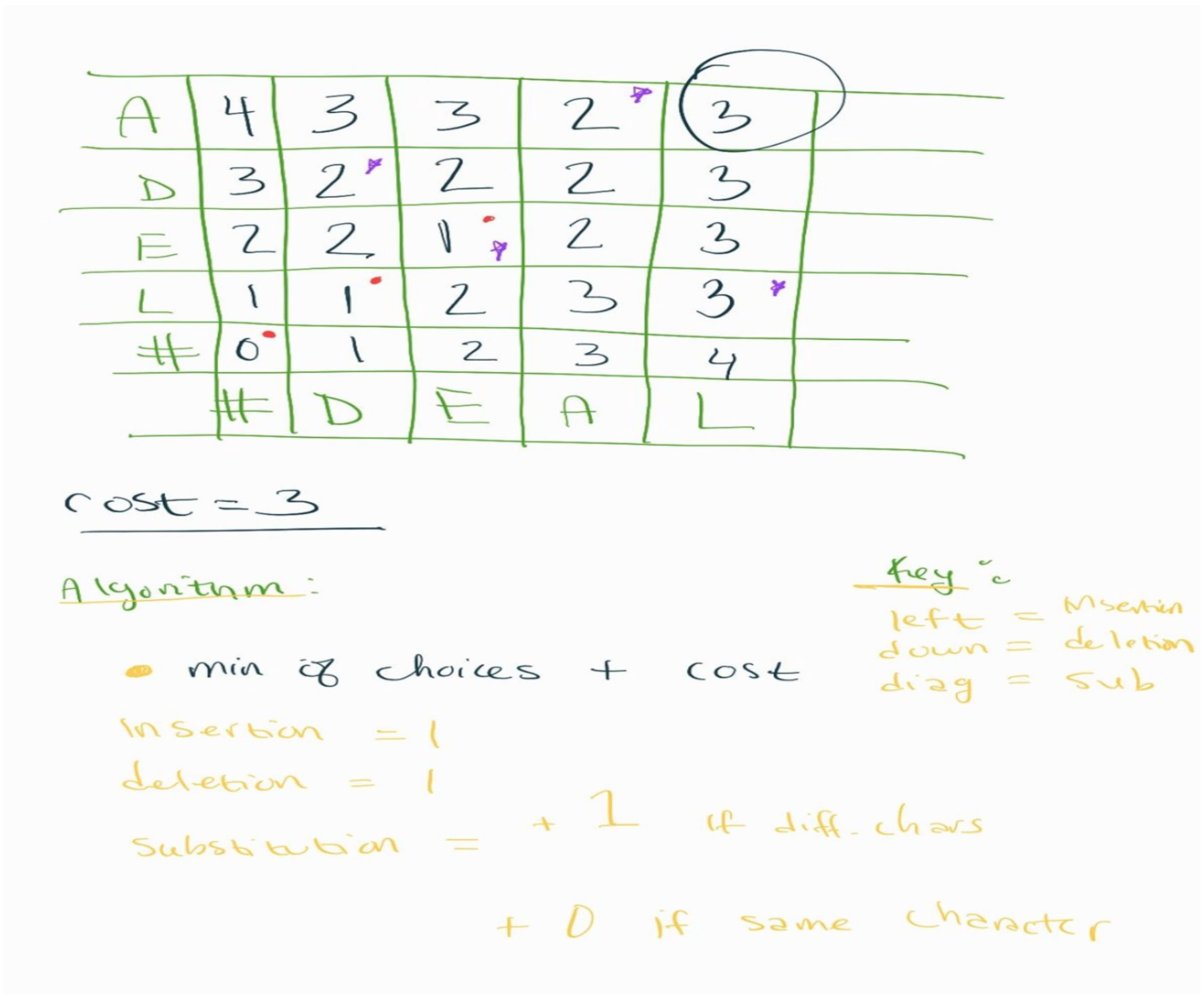
5. Compute the edit distance (using insertion cost 1, deletion cost 1, substitution cost 1) of *leda* to *deal*. Show your work (using the edit distance grid).

Ans: See Figure 1.

6. Figure out whether *drive* is closer to *brief* or to *divers* and what the edit distance is to each. Use Levehnstein distance.

Ans: See Figure 2.

7. Now implement a minimum edit distance algorithm and use your hand-computed results to check your code.



A	4	3	3	2	3
D	3	2	2	2	3
E	2	2	1	2	3
L	1	1	2	3	3
#	0	1	2	3	4
	#	D	E	A	L

cost = 3

Algorithm:

- min of choices + cost

insertion = 1

deletion = 1

substitution = + 1 if diff. chars

+ 0 if same character

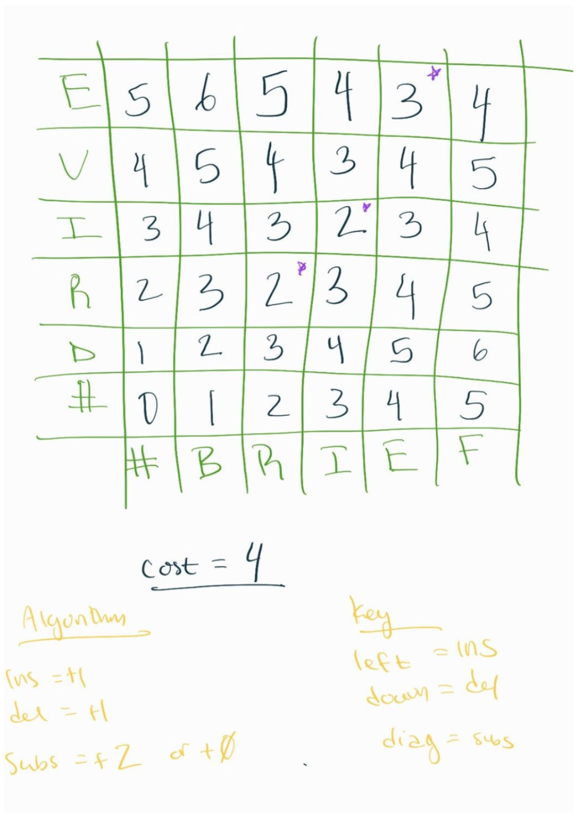
Key:

left = insertion

down = deletion

diag = sub

Figure 1: Q5



E	5	6	5	4	3*	4
V	4	5	4	3	4	5
I	3	4	3	2*	3	4
R	2	3	2*	3	4	5
D	1	2	3	4	5	6
#	0	1	2	3	4	5
	#	B	R	I	E	F

cost = 4

Algorithm

ins = +1

del = +1

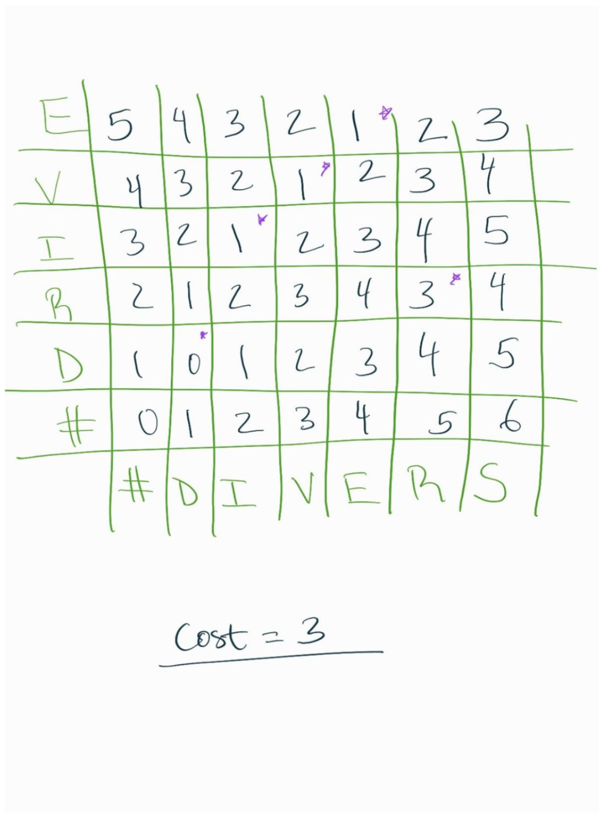
subs = +2 or +∅

Key

left = ins

down = del

diag = subs



E	5	4	3	2	1*	2	3
V	4	3	2	1*	2	3	4
I	3	2	1*	2	3	4	5
R	2	1	2	3	4	3*	4
D	1	0*	1	2	3	4	5
#	0	1	2	3	4	5	6
	#	D	I	V	E	R	S

Cost = 3

Figure 2: Q6