Student number:	2467273
Course title:	COMPSCI4074 Text as Data (H)
Questions answered:	ALL

1.

a)

Assumption: *similarity* is calculated, not relative distance. Assumption: the word vector embeds context (and by extension, meaning) of the word and is able to signify the outlying vector. Assumption: the programming language used has a working hash map implementation.

0. Keep a minimum similarity variable *min_sim* (initialise to positive infinity) and the outlying vector position variable *outlying_vec* (initialised to -1).
1. Go through all word vectors and again for each vector (a nested loop).
2. Build a hash map/dictionary with each word as key and a list of all similarities of the vector with all other vectors.
3. After all similarities have been calculated for a word, calculate the total similarity of the word for all other words.
4. Compare the total similarity with *min_sim*. If smaller, change *min_sim* to the current total similarity and *outlying_vec* to the current vector's position.
5. After the full loop ends, the vector at position *outlying_vec* is the outlier. Even if the difference between similarities is small, one word is always returned.

b) If k is too large, the context of a word is essentially overfitted – there are no (or very few) other words with similar contexts because it is too specific/rare. The opposite is true for too small of a value for k – the word would be close to context-less and would be similar to too many other words.

c)

1. Pass through all words. Only one pass through the words is needed by sliding a window of size k through the words.

2. The word in the middle of the context is the anchor word, which can be accessed from a hash map (necessary for fast (O(1)) lookup) and:

3. updated with its current context in the pass. This updating is performed essentially by conducting a set union with all of the contexts for every occurrence of the word. It would increase the number of each context word occurrence by 1.

The final, overall context vector for each word (the value in the hash map with the word as the key) should be a sparse representation to preserve memory, like an ordered list of <word, frequency> tuples to count the term frequency of all contextual words.

d) The contexts of these words can be wildly different, e.g., "I saw the bird" and "The king in the North". Function words do not have much of a meaning themselves and only act as additional semantics to a sentence/part of a sentence, and these vectors would be wildly ambiguous.

2.

a)

Trigram	Frequency
abc	3
eda	3
aed	2
dab	2

b) If the question is asking for non-unique n-grams:

n=1 (Unigrams): 593
n=2 (Bigrams): 592
n=3 (Trigrams): 591
n=4: 590
n=5: 589

c)
i) Every character ‘a’ is the end of a token.

ii) Every bigram ‘ab’ is the end of a token until the end of the sentence, when the last character is the end of the last token.

d)
i)
|A| = 5 (unique)
|B| = 5 (unique)
$|A \cap B|$ = 3
$|A \cup B|$ = 7

sim(A, B) (Sørensen–Dice Coefficient) = 2 * $|A \cap B|$ / (|A| + |B|) = 2 * 3 / 10 = 0.6

sim(A, B) (Jaccard Similarity) = $|A \cap B|$ / $|A \cup B|$ = 3 / 7 = 0.429

ii)
|C| = 5
$|A \cap C|$ = 1
$|B \cap C|$ = 1

sim(A, C) = 2 * 1 / (5 + 5) = 0.2
sim(B, C) = 2 * 1 / (5 + 5) = 0.2

Because sim(A, B) > sim(A, C) + sim(B, C) (0.6 > 0.2 + 0.2 = 0.4), it does not satisfy the triangle inequality (sim(A, B) $\leq$ sim(A, C) + sim(B, C)) and is thus only a semi-metric.

3.

a) A stemmer would be extremely important to normalise different usages of the same root of a word and help classify a text with those different usages. For example, the subject matter "Electricity" can be seen in words such as "electric", "electronic", "electrical", etc. Stemming would allow to normalise all of these (to something like "electr") and point towards section H with much more ease by standardising all of those different words to the same one with one meaning.

Case normalisation would achieve all of the same goals as stemming but by normalising the varied use of cases, such as sentences starting with capital letters ("Lighting is an important section.") and proper names for items, e.g., "The newly developed Lightbulb 3000", in contrast with non-capitalised uses of similar words ("I like lightbulbs." and "Much lighting, wow.").

b) One example I could come up is the word "dress". As a noun, it is a fancy item of clothing, which, when encountered in a patent, would seem, as a textile, to point towards section D as the document's class. However, it can be understood as a verb as well, and its usage as a verb might be encountered more in section A, since clothing is a human necessity and "to dress up in clothing" might be a more frequently encountered phrase in these kinds of patents. Another example would be the word "iron", which, used as a noun (a chemical element and metal), would be indicative of section C, or, used as a verb (to steam and un-wrinkle clothing), would be much more indicative of section D.

c) Because the POS tagger fails on specialised language, it shows that the tagger has been trained on general, widely spoken language, which has very different semantics and lexicon to specialised language. From the example in part (b), "dress" is more commonly used in the textile connotation, but it could be much more prevalent as a verb in section A, which would then be wrongly attributed to the widely used meaning. In order to fix this, the tagger needs to be trained with some already classified patents (keeping 20% as test data for validation), which would then ensure that this domain-specific language is kept in evaluation.

d) System A performs very marginally better than the baseline in both training and test data, which shows that either it is a poorly chosen system or that it is underfitted, i.e., not trained enough on the train set. It seems to be very good on generalisability then since it does well on the test set, but it could be trained more (but care should be taken not to overfit it by stopping when the test set metrics start to worsen and/or by using cross-validation).

System B performs better on the train set, but worse in the test set, which indicates that it is overfitted and not generalisable. It should be trained less on the train set, maybe by also using cross-validation, but System A seems to do better than System B on the test set, which is the more important set for further use of the system.