General reminder: There will inevitably be some difference in your implementations. The grading will be done accordingly. Anyways, minor changes will not change the top 20 collocation candidates, except perhaps a few and maybe some differences in the ordering.

Should we do our calculations on all bigrams and words, or only collocation candidates?

You should do your calculations on all of the corpus (getting counts etc). Your collocation candidates are only a very small subset of all the dataset, and may not yield accurate results by themselves. After tokenizing and lemmatizing the text, find all bigrams and calculate all counts. Then, construct a list of collocation candidates that satisfy the criteria in part 1.f. You can then do the tests only on these collocation candidates.

Should we report POS-tagged collocation candidates?

The assignment and the answer sheet do not make a distinction between ADJ-NOUN and NOUN-NOUN candidates. For instance, at one point, you are asked to report the scores for (“good”, “one”), but not (“good” ADJ, “one” NOUN). You should therefore not report POS-tagged collocation candidates, nor have them in the first place.

So how do you find the counts of collocation candidates? Say we are looking for the collocation “good one”. It may appear as ADJ-NOUN or NOUN-NOUN in different places. When finding the count, your answers will be accepted if you use one of the following two implementations, as there is apparently some ambiguity in the instructions:

1 Add “good one” to the list of your collocation candidates and then count all appearances of “good one” regardless of POS-tags (even those that are not ADJ-NOUN or NOUN-NOUN).

2 Count the occurrences of (“good” (ADJ), “one” (NOUN)) and (“good” (NOUN), “one” (NOUN)) and add them up to get the count of “good one”.

How should I get the count of a word that appears in a collocation candidate, but appears elsewhere with a different POS-tag?

Say we are considering the bigram “fast horse”. The word “fast” may appear elsewhere as an adverb (as in “go fast”). These occurrences also contribute to the probability p(“fast”) and should be accounted for. However, if you implemented it in a different way, please mention that in your report, and I will check your answers accordingly.

What happens to the counts for window size 3?

When making probability and expectation calculations on window size 3, you should keep in mind that a given word is three times more likely to appear in a bigram, and the total bigram count is three times more (~3N). Make the necessary adjustments when calculating probabilities etc. For example, the expected number of a bigram will be three times the expectation for window size one. And, for example, when calculating the probability of a bigram empirically, you should divide the count of the bigram by the total number of bigrams, which is three times the count for window size one.

You can handle this issue in two ways in your code:

1 Multiply N and all c(w) by the window size (3) when making calculations.

or equivalently

2 Simply get the word counts from the list of all bigrams (not from the tokenized corpus), and choose N as the total number of bigrams (not the token size). Be careful not to count words 6 times instead of 3 (or 2 instead of 1) if you get the count from bigrams (each bigram consists of two words, and the total number of words in bigrams is ~6N, not ~3N, which is not what you are looking for).

Should I avoid punctuation marks when constructing bigrams?

The assignment does not ask you to do any further processing on the corpus other than tokenization and lemmatization. For example, in “… lincoln's inn hall. implacable november weather …”, if (“hall”, “implacable”) is not considered a bigram for window size 1, that is natural and ok. If you implemented it otherwise, it will also be accepted.

Are my scores too large?

Probably not. Chi-square test and likelihood-ratio test scores are naturally much larger than t-scores. The likelihood-score for example is asymptotically chi-square distributed only if the null hypothesis is true.

Should we construct 4-grams?

No. For window size 3, you are expected to find two word-collocation candidates, possibly separated by other words, over windows of size 3.

I get overflow errors when calculating binomial probabilities, what should I do?

Please use scipy or another appropriate library for binomial calculations. If you code it manually, it will likely be massively slow, and will cause overflow errors. Please note that you may still get a zero output for very low probabilities. In that case, avoid zero-division or log(0) by replacing the zero with some very small ε>0.

The nltk tokenizer/lemmatizer does not work very well. Should I be worried?

You are fine as long as you stick with the instructions. However, if you want to make an improvement by using alternative tokenizers/lemmatizers, you are more than welcome to do so. Just mention the difference in your report.