Part 1: Unsmoothed ngrams:
Our goal is to compute unsmoothed unigram and bigram probabilities from our corpuses, one negative and one positive. The strategy is to move along the entire corpus with a window of 2, add the current element count as the unigram-count, and then the count of the window as the bigram-count. We can then compute the probability for both on demand.

Preprocessing decisions:
We chose to use the EOS character '\n' to separate documents from each other. After separating documents in this way, we add an explicit start of string character and end of string character - <S> and </S>. We chose not to use '-' as a delimiter; we wanted to treat hyphenated words as a single word. We did, however, use ‘’ as a delimiter; there were too many edge cases in the corpus to deal with adequately. We did plan on having ‘’ when used as a conjuction retained as part of the token, but this did not seem to make much of a difference. The words we did choose as delimiters were ' ', ';', ':' ‘,’ ‘.’. In addition, we experimented with using ',' and '.' as tokens, as they were important parts of speech we did not want to drop. But this created less intelligible sentences and also created additional problems such as unwanted characters like “. . .” getting included from the corpus. Based on this, we decided to treat ‘.’ and ‘,’ as delimiters as well.

Part 2: Random sentence generation

Goal: To generate sentences at random, given the knowledge of unigram and bigram models, developed from the corpus

Data: Do for both corpuses and both models

When we looked at our sentence generator, we realized that there was a design choice we needed to make - **Limit sentence to n words**
Because we drew the end-of-sentence symbol randomly, we were finding that our sentences were very long. To combat this, we chose to enforce a maximum of n words, with fewer possible if the end-of-sentence symbol is drawn. We used numpy to draw words from our probability distribution
For bigram generation, we maintain a data structure of next possible words that can follow a given word, along with their bigram probabilities.

Unigram sentence generated from pos.txt:
1. of culture advocacy no its are to in is an
2. both all sense so theme to a illustrated theory by
3. story that images irony viewer dark best inventive snaps of

Bigram sentence generated from pos.txt:
1. <s> neither as i love may also a good fight scenes
2. <s> this is the right questions of the multi-layers of a
3. <s> seen him </s> (stopped at less than 10 words because a </s> token was encountered)

Unigram sentence generated from neg.txt:
1. as brown right you of the just drama just lacks
2. detached -- have exploitive burger disaster of of anyone mr
3. hugh surrounding director be haplessness as still over ve leniency

Bigram sentence generated from neg.txt:
1. <s> the most visually flashy camera movements than us only way
2. <s> the rock s not only fun of disguise falls short
3. <s> starts and in the genre </s> (stopped at less than 10 words because a </s> token was encountered)

The unigram sentences appear to be a random selection of words from the corpus, as expected. The bigram sentence more closely resembles a well-formed English sentence. This is because we are able to draw on context when choosing the next word, and so from word to word, the output is closer to the well-formed sentences in the corpus.

Experimentation with seeding:

We also used seeding to see if better sentences could be generated. We provide a function that will take in the seeded words, and will then append n extra words, using the seeded words as context for the bigram model we have. We have not done this for unigram, as the extra context is not going to be used. Even for bigram, only the last word will be useful, as it is what we use to randomly choose a value.

We notice that with seeding, the sentence structure is still around the same as what the original bigram models provide in terms of English sentences that make sense. This is likely because the word that is used for context to feed into the bigram model is not sufficient to really change the probability of what is returned. If we were to adopt a larger window of context, we would likely see a more dramatic effect between seeding and no seeding on sentence generation.

Bigram generated sentence with seed [simply radiates star-power potential]: *simply radiates star-power potential success </s>*

Bigram generated sentence with seed [simply radiates star-power potential]: *simply radiates star-power potential for something one of the charisma make the dragons !*