Team Members:

- Blake Oberlander (bho180000)
- Sharar Siddiqui (srs200003)

a) N-grams is a sliding window that goes over a corpus of text. N here would signify how many words at a time the window would look over. 1 would mean it takes 1 word at a time(Unigram), Bigram for 2 words, Trigram for 3. Using this method we can get a set of words that can be predicted to come after one specific word and the likeliness of it. And through that we can make a probabilistic model of prediction which can then be used to predict words, identify languages and more.

b) N-grams have applications in:
   i) Spelling and grammar checking and correction
   ii) Text compression
   iii) Dictionary look-up
   iv) Language identification
   v) Semantic analysis (for example, sentiment)

c) For any language model, there exists a probability distribution of n-grams. We could calculate the probability of a specific n-gram appearing in a language using a corpus of texts. The probability is given by (number of times the specific n-gram appears) / (total number of n-grams). For large values of n, however, this straightforward computation is infeasible. So we make use of the Markov assumption:

$$P(w_1, w_2, \cdots, w_n) = P(w_1, \cdots, w_{n-1})P(w_n|w_1, \cdots, w_{n-1})$$
$$\approx P(w_1, \cdots, w_{n-1})P(w_n|w_{n-1})$$

Applying that recursively…

$$\approx P(w_1)P(w_2|w_1)P(w_3|w_2) \cdots P(w_n|w_{n-1})$$

This gives us an easy way to quickly approximate the probability of any n-gram (/ sentence) programatically. Our program implements this formula but also incorporates smoothing, which is described below.

d) To get a good language model we need the source to encompass most if not all of the general terms in a language. If given a corpus of anatomy and we were given lines that referred to engineering it is likely that the prediction might fail to get low accuracy. So to get accurate results we will need a good source text that would likely have the Ngrams of the lines we may ask it to predict and/or recognize.

e) Without smoothing, we run into the **sparsity problem.** Some sequences of words never appear in a corpus yet still may appear in a valid sentence. For example, the probability of the 2-gram “amazingly quintessential” appearing in any given corpus is extremely small, yet it is still possible in the broader language. Without using the smoothing technique, though, the Markov evaluation of any n-gram containing this 2-gram would likely be 0, depending on the corpus. That’s because of how the probabilities multiply. This is undesirable, so ideally no factor in the product should ever be exactly equal to zero. In the worst case, a factor should only be very close to zero. A smoothing function,

such as Laplace, ensures that all factors are non-zero, and that no single 2-gram disqualifies the sentence.

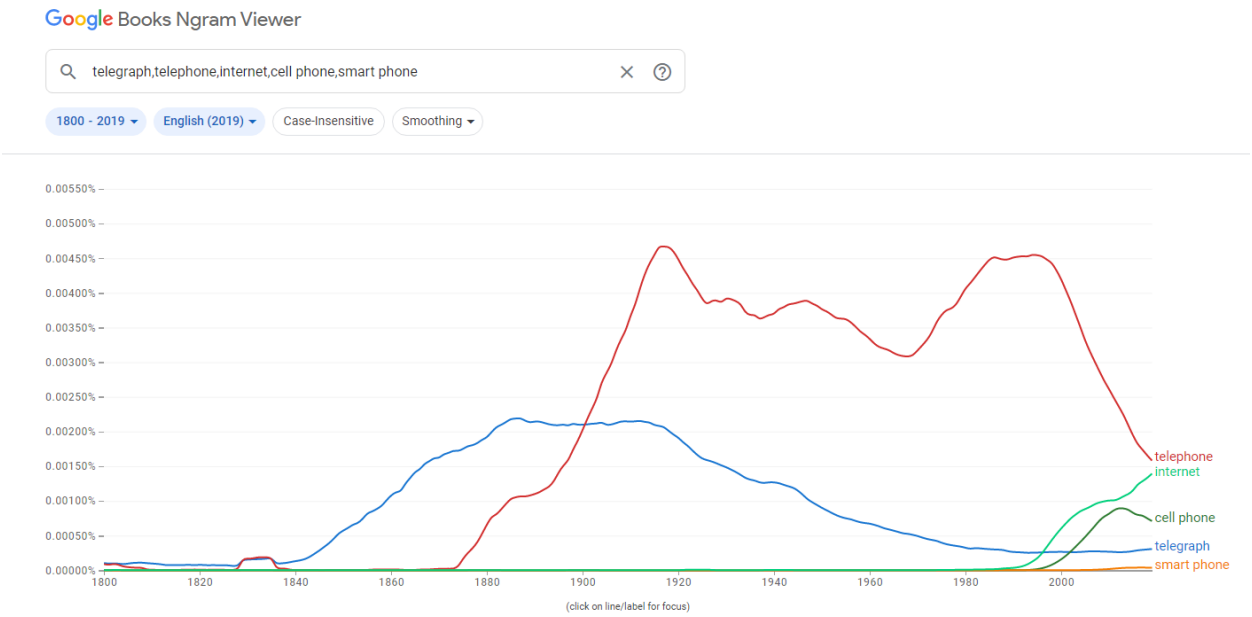
f) Text generation would signify a machine's ability to output human readable text. It would also encompass text prediction in it. Having a language model we can get the N-grams of the said model. As we have the N-grams we can also have the probabilities of words occurring next to each other. Using this feature of N-grams we can generate new text or predict text being written. The accuracy of the models generation increases with better corpuses and higher N-grams. A N-gram of 6 words would yield a better result than an N-gram of 2 words.
This has its caveats in what corpus is used to get the language model. We also are limited by the N-grams in the language models corpus. If the language modeled is a corpus about anatomy we can not use it for engineering language prediction.

g) There are 2 ways to evaluate NLP models: (1) extrinsic evaluation, and (2) intrinsic evaluation. Extrinsic evaluation is time consuming and requires a human to manually evaluate the quality of a model for some application. Intrinsic evaluation, on the other hand, uses math to evaluate models. A metric called **perplexity** is used to evaluate models intrinsically. Perplexity essentially represents the *entropy* of a data set. The less noisy and alike a model is to the actual language, the lower the entropy. Formally, perplexity is calculated as follows:

$$\sum_{test\ sentence} P(test\ sentence)^{-1/N}$$

… Where P is given by the model.

h) Googles ngram viewer(https://books.google.com/ngrams/) gives us the ability to put in words and it will give us the occurrences of the said words in the huge corpus of books. It also lets us put in a set of words(bigrams) or N-grams in general.



[Link to search results]

Using this we can see the occurrences of the ngrams (telegraph,telephone,internet,cell phone,smart phone) throughout time in the set of books and documents that are present in google's archive. This in a sense gives us the history of word usage through time. It can also give us interesting data similar to our output where we see the word telephone used much more today than cell phone or smart phone.