# Task 1: [5 marks]

1. Complete the following code snippet to download [unigrams](https://norvig.com/ngrams/count_1w.txt) and [bigrams](https://norvig.com/ngrams/count_2w.txt) data, and load them into their corresponding dataframes, *unigrams\_df* **and** *bigrams\_df*

**YOUR CODE HERE**

print(f'Number of unigrams:totalUnigrams} Number of Bigrams: {totalBigrams}')

# display(unigrams\_df.head(100),bigrams\_df.head(100))

1. Complete the function below such that it computes the probability of a given sentence using the unigram and bigram counts from (a). Do not apply add-one smoothing [3 marks]

def probability(sentence):

**YOUR CODE HERE**

return sentenceProbability

probability("i love you")>probability('i hate you')

1. Copy the function from (b) to a new cell, and modify it such that it applies

add-one smoothing. Your new function should also address the problem of underflow

When computing the sentence Probability [2 marks]

**Expected output for (b) →**

# 

# 

# 

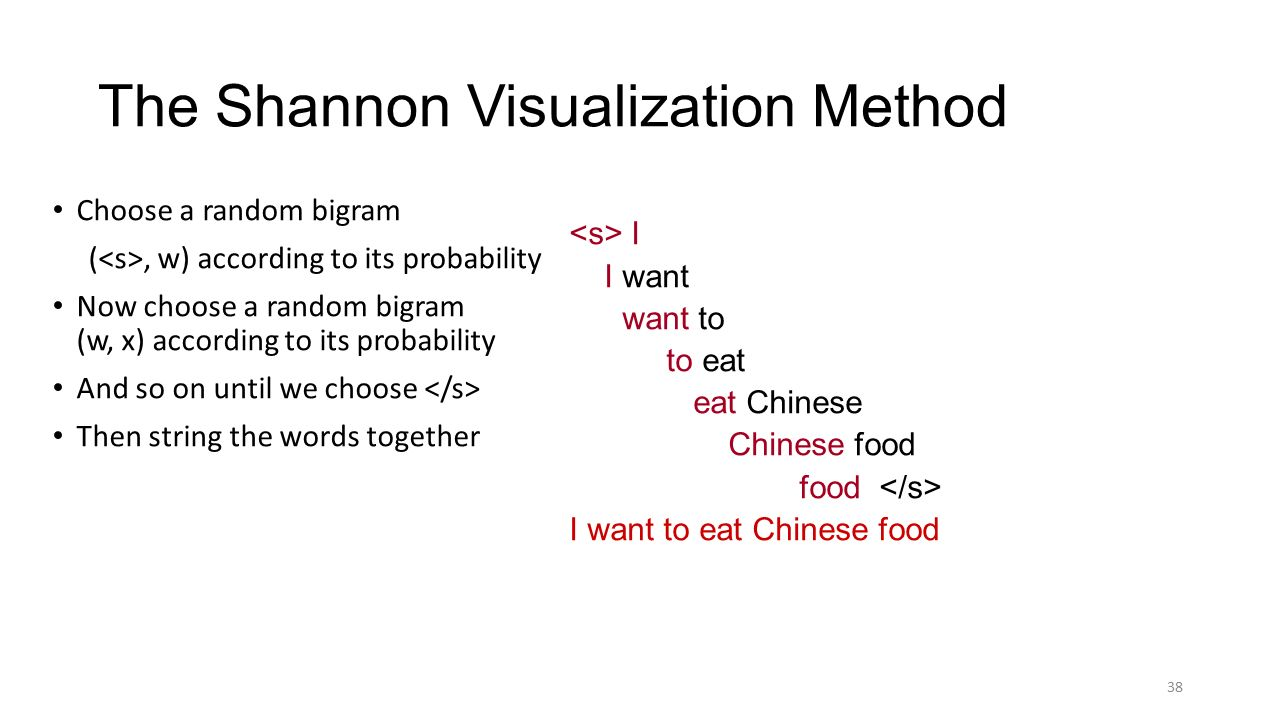
# 

# Task 3: [5 marks]

Using the [bigram counts](https://norvig.com/ngrams/count_2w.txt) (*bigrams\_df*), write a function that implements the Shannon visualization method:

def ShannonVisualization(seed=”<S>”):

**YOUR CODE HERE**



[Chapter 3, section 3.3](http://web.stanford.edu/~jurafsky/slp3/3.pdf) gives a detailed description of the Shannon Visualisation method; a screenshot of this section is provided below.

If you can follow the logic applied to unigrams, then you should be able to apply the same logic to bigrams.

The seed word can be any word, but let's stick to the default seed word <S> which indicates the beginning of the sentence.

First, you need to retrieve a list of all the bigrams starting with <S>, and then randomly select one of them according to its probability. Understanding this last part is key! You have a probability space between 0 and 1 and each of the candidate bigrams covers an interval in this space proportional to its frequency. For example, the candidate bigram “<S> and” has a total frequency of 261891475 which translates to a probability of 0.031513. If you consider this bigram to be your first candidate, then you can assign the probability interval 0.0 to 0.031513 to it. There are 8640 bigrams in the dataset which start with <S> and you need to computer the probability intervals for all of them.

Then Generate a random number between 0 and 1 and check to see which interval it falls into, and that would be your chosen bigram.

Pick the second word in the chosen bigram and that would be your new seed word.

This process should be repeated until you either reach </S> (indicating the end of the sentence) or come across a seed which does not exist in your list of bigrams (i.e. there is no bigram starting with this seed word).

**NOTE:** Given the large size of the bigrams data, the Shannon Viz game can continue indefinitely; therefore, for the purpose of testing you can terminate the game early whenever a common word (e.g. “the”, “of”, “in”) is produced as the next new seed.

## 

## </END OF ETIVITY>

Teaching team:

* Level of difficulty (easy/fair/hard)
* Task 4: use the Shakespeare corpus to create new unigram and bigram dataframes
* Task 5: deployment of a modern language model api/lib
* **TASK2.B print V, use loc and iloc**