

**Department of Computer Science and Engineering(UG Studies)**

**PES University, Bangalore-560085**

|  |  |
| --- | --- |
| **Session :** Aug - Dec 2017  **Credits :** 0-0-2-0-1 | UE14CS405 : Machine Learning Lab |
| **Lab # :** 06 | Build a Unigram, Bigram and trigram model for a given text document |

**Data sets :** 1) Twitter data set (Pre-processed)

2) The 20 Newsgroups data set

## 3) nltk corpus

## 4) Blogs Information

## 5) The IMDB Movies Dataset

**Theory :**

In the fields of computational linguistics and probability, an ***n*-gram** is a contiguous sequence of *n* items from a given sequence of text or speech. The items can be [phonemes](https://en.wikipedia.org/wiki/Phoneme), [syllables](https://en.wikipedia.org/wiki/Syllable), [letters](https://en.wikipedia.org/wiki/Letter_(alphabet)), words or base pairs according to the application. The *n*-grams typically are collected from a [text](https://en.wikipedia.org/wiki/Text_corpus) or [speech corpus](https://en.wikipedia.org/wiki/Speech_corpus). When the items are words, *n*-grams may also be called **shingles**.

An *n*-gram of size 1 is referred to as a "**unigram**"; size 2 is a "[**bigram**](https://en.wikipedia.org/wiki/Bigram)" (or, less commonly, a "digram"); size 3 is a "[**trigram**](https://en.wikipedia.org/wiki/Trigram)". Larger sizes are sometimes referred to by the value of *n* in modern language, e.g., "four-gram", "five-gram", and so on.

If X=Num of words in a given sentence K, the number of N-grams for sentence K would be:

http://latex.codecogs.com/gif.latex?Ngrams_K=X-(N-1)

N-grams of texts are extensively used in text mining and natural language processing tasks. They are basically **a set of co-occurring words** within a given window and when computing the n-grams you typically move one word forward (although you can move X words forward in more advanced scenarios).

For example, for the sentence **"The cow jumps over the moon"**. If N=2 (known as bigrams), then the ngrams would be:

* **the cow**
* **cow jumps**
* **jumps over**
* **over the**
* **the moon**

So you have 5 n-grams in this case. Notice that we moved from the->cow to cow->jumps to jumps->over, etc, essentially moving one word forward to generate the next bigram.  
  
If N=3, the n-grams would be:

* **the cow jumps**
* **cow jumps over**
* **jumps over the**
* **over the moon**

**What are N-grams used for?**

N-grams are used for a variety of different task. For example, when developing a language model, n-grams are used to develop not just unigram models but also bigram and trigram models. Google and Microsoft have developed web scale n-gram models that can be used in a variety of tasks such as spelling correction, word breaking and text summarization.

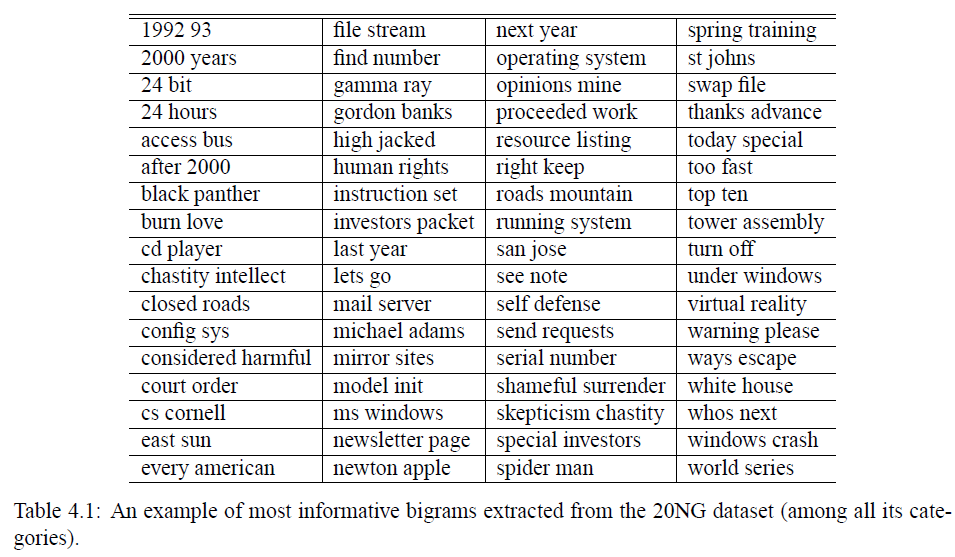
**Objectives**

* Frequent *n*-grams in English
* *n*-grams and statistical NLP
* *n*-grams and conditional probability
* Large *n*-gram resources

**Applications:**

* OCR / Voice recognition – resolve ambiguity
* Spelling correction
* Machine translation
* Confirming the author of a newly discovered work
* “Shannon game”: Predict the next word, given (*n-1)* previous words. Determine probability of different sequences by examining training corpus
* Text categorization
* Word clustering

**Most informative bigrams:**



**Sample Code:**

**Program-1: Python program that outputs a list of n-grams represented as list.**

**Input text file:** input\_text1.txt

Never stop fighting until you arrive at your destined place - that is, the unique you. Have an aim in life, continuously acquire knowledge, work hard, and have perseverance to realise the great life. A. P. J. Abdul Kalam

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

from nltk import word\_tokenize

file\_content = open("input\_text1.txt").read()

wordlist = word\_tokenize(file\_content)

print('\nTokens List:\n')

print(wordlist)

def getNGrams(input\_list, n):

print('\n',n,'\_Grams:\n')

return [input\_list[i:i+n] for i in range(len(input\_list)-(n-1))]

print(getNGrams(wordlist, 3))

**Program-1 output:**

Tokens List:

['Never', 'stop', 'fighting', 'until', 'you', 'arrive', 'at', 'your', 'destined', 'place', '-', 'that', 'is', ',', 'the', 'unique', 'you', '.', 'Have', 'an', 'aim', 'in', 'life', ',', 'continuously', 'acquire', 'knowledge', ',', 'work', 'hard', ',', 'and', 'have', 'perseverance', 'to', 'realise', 'the', 'great', 'life', '.', 'A.', 'P.', 'J.', 'Abdul', 'Kalam']

3 \_Grams:

[['Never', 'stop', 'fighting'], ['stop', 'fighting', 'until'], ['fighting', 'until', 'you'], ['until', 'you', 'arrive'], ['you', 'arrive', 'at'], ['arrive', 'at', 'your'], ['at', 'your', 'destined'], ['your', 'destined', 'place'],

.......

['realise', 'the', 'great'], ['the', 'great', 'life'], ['great', 'life', '.'], ['life', '.', 'A.'], ['.', 'A.', 'P.'], ['A.', 'P.', 'J.'], ['P.', 'J.', 'Abdul'], ['J.', 'Abdul', 'Kalam']]

**Program-2: Python program that outputs a list of n-grams represented as tuple.**

**Input text file:** input\_text2.txt

Be true to yourself, help others, make each day your masterpiece, make friendship a fine art, drink deeply from good books, build a shelter against a rainy day, give thanks for your blessings and pray for guidance every day.

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

from nltk import word\_tokenize

file\_content = open("input\_text2.txt").read()

wordlist = word\_tokenize(file\_content)

print('\nTokens List:\n')

print(wordlist)

def getNGrams(input\_list, n):

print('\n',n,'\_Grams:\n')

result=zip(\*[input\_list[i:] for i in range(n)])

return result

gram=getNGrams(wordlist, 2)

print(list(gram))

**Program-2 output:**

Tokens List:

['Be', 'true', 'to', 'yourself', ',', 'help', 'others', ',', 'make', 'each', 'day', 'your', 'masterpiece', ',', 'make', 'friendship', 'a', 'fine', 'art', ',', 'drink', 'deeply', 'from', 'good', 'books', ',', 'build', 'a', 'shelter', 'against', 'a', 'rainy', 'day', ',', 'give', 'thanks', 'for', 'your', 'blessings', 'and', 'pray', 'for', 'guidance', 'every', 'day', '.']

2 \_Grams:

[('Be', 'true'), ('true', 'to'), ('to', 'yourself'), ('yourself', ','), (',', 'help'), ('help', 'others'), ('others', ','), (',', 'make'), ('make', 'each'), ('thanks', 'for'), ('for', 'your'), ('your', 'blessings'), ('blessings', 'and'),

.......

('and', 'pray'), ('pray', 'for'), ('for', 'guidance'), ('guidance', 'every'), ('every', 'day'), ('day', '.')]

**Program-3: Python program that outputs a list of n-grams by calling ngrams method in nltk.**

**Input text file:** input\_text.txt

Do you like to sing? My mom and I sing in a choir that meets every Tuesday. Last week, we sang Christmas carols at a nursing home. The residents of the nursing home enjoyed our visit! It was fun! Would you like to join us next Tuesday?

*\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_*

from nltk import word\_tokenize

from nltk.util import ngrams

file\_content = open("input\_text.txt").read()

tokens = word\_tokenize(file\_content)

print('\nTokens List:\n')

print(tokens)

for i in range(2,4):

print('\n',i,'\_Grams:\n')

gramslist=ngrams(tokens,i)

for gram in gramslist:

print(gram)

**Program-5 output:**

Tokens List:

['Do', 'you', 'like', 'to', 'sing', '?', 'My', 'mom', 'and', 'I', 'sing', 'in', 'a', 'choir', 'that', 'meets', 'every', 'Tuesday', '.', 'Last', 'week', ',', 'we', 'sang', 'Christmas', 'carols', 'at', 'a', 'nursing', 'home', '.', 'The', 'residents', 'of', 'the', 'nursing', 'home', 'enjoyed', 'our', 'visit', '!', 'It', 'was', 'fun', '!', 'Would', 'you', 'like', 'to', 'join', 'us', 'next', 'Tuesday', '?']

2 \_Grams:

('Do', 'you')

('you', 'like')

('like', 'to')

('to', 'sing')

('sing', '?')

('?', 'My')

......

('next', 'Tuesday')

('Tuesday', '?')

3 \_Grams:

('Do', 'you', 'like')

('you', 'like', 'to')

('like', 'to', 'sing')

('to', 'sing', '?')

('sing', '?', 'My')

('?', 'My', 'mom')

......

('us', 'next', 'Tuesday')

('next', 'Tuesday', '?')

**To do List :**

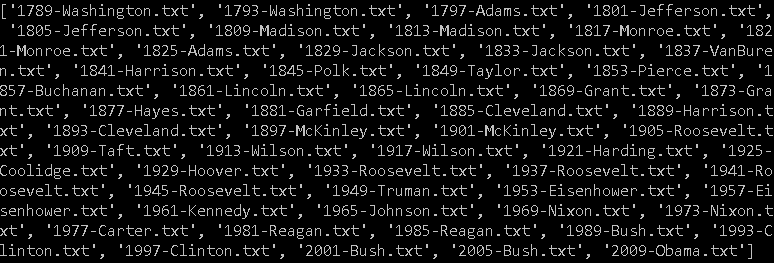
**Program-1:** Python program that outputs a list of n-grams represented as string and a dictionary containing n-gram and its count as key-value pairs.

**Program-2:** Python program that outputs a list of all n-grams(one, two, three, and four) represented as string and its probability.

**Data:**

from nltk.corpus import inaugural

print(inaugural.fields())



**Program 3**

Find the total number of words (tokens) in Obama's 2009 speech(i.e., 2009-Obama.txt). Find the total number of distinct words (word types) in the same speech.

**Program 4**

Compare the top 50 most frequent words in Barack Obama's 2009 speech with George Washington's 1789 speech.

What can knowing word frequencies tell us about different speeches at different times in history?

**Program 5**

Find the most common unigrams, bigrams and trigrams in Obama's 2009 speech.