**1. What are Corpora?**

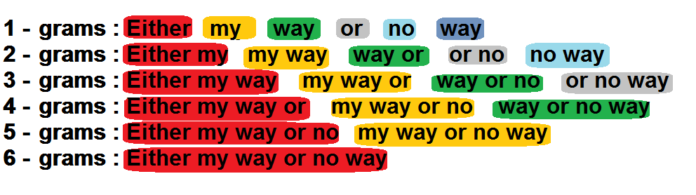
**Ans:** A corpus is a large and structured set of machine-readable texts that have been produced in a natural communicative setting. Its plural is **corpora**. They can be derived in different ways like text that was originally electronic, transcripts of spoken language and optical character recognition, etc.

**2. What are Tokens?**

**Ans:** **Tokens are the building blocks of Natural Language.** Tokenization is a way of separating a piece of text into smaller units called tokens. Here, tokens can be either words, characters, or subwords. Hence, tokenization can be broadly classified into 3 types – word, character, and subword (n-gram characters) tokenization.

**3. What are Unigrams, Bigrams, Trigrams?**

**Ans:** In Natural Language Processing n-gram is a contiguous sequence of n items generated from a given sample of text where the items can be characters or words and n can be any numbers like 1,2,3, etc.

An n-gram of size 1 is referred to as a **unigram** size 2 is a **bigram** size 3 is a **trigram**. When **N>3** this is usually referred to as four grams or five grams and so on.  

**4. How to generate n-grams from text?**

In [4]:

**from** nltk.util **import** ngrams, everygrams

**def** ngram\_convertor(sentence,n**=**3):

ngram\_sentence **=** ngrams(sentence**.**split(), n)

**for** item **in** ngram\_sentence:

print(item,end**=**',')

print()

sentence **=** "Life is either a daring adventure or nothing at all"

print('-'**\***25,'Unigram','-'**\***25)

ngram\_convertor(sentence,1)

print('-'**\***25,'Bigram','-'**\***25)

ngram\_convertor(sentence,2)

print('-'**\***25,'Trigram','-'**\***25)

ngram\_convertor(sentence,3)

print('-'**\***25,'Everygram','-'**\***25)

print(list(everygrams(sentence**.**split())))

------------------------- Unigram -------------------------

('Life',),('is',),('either',),('a',),('daring',),('adventure',),('or',),('nothing',),('at',),('all',),

------------------------- Bigram -------------------------

('Life', 'is'),('is', 'either'),('either', 'a'),('a', 'daring'),('daring', 'adventure'),('adventure', 'or'),('or', 'nothing'),('nothing', 'at'),('at', 'all'),

------------------------- Trigram -------------------------

('Life', 'is', 'either'),('is', 'either', 'a'),('either', 'a', 'daring'),('a', 'daring', 'adventure'),('daring', 'adventure', 'or'),('adventure', 'or', 'nothing'),('or', 'nothing', 'at'),('nothing', 'at', 'all'),

------------------------- Everygram -------------------------

[('Life',), ('Life', 'is'), ('Life', 'is', 'either'), ('Life', 'is', 'either', 'a'), ('Life', 'is', 'either', 'a', 'daring'), ('Life', 'is', 'either', 'a', 'daring', 'adventure'), ('Life', 'is', 'either', 'a', 'daring', 'adventure', 'or'), ('Life', 'is', 'either', 'a', 'daring', 'adventure', 'or', 'nothing'), ('Life', 'is', 'either', 'a', 'daring', 'adventure', 'or', 'nothing', 'at'), ('Life', 'is', 'either', 'a', 'daring', 'adventure', 'or', 'nothing', 'at', 'all'), ('is',), ('is', 'either'), ('is', 'either', 'a'), ('is', 'either', 'a', 'daring'), ('is', 'either', 'a', 'daring', 'adventure'), ('is', 'either', 'a', 'daring', 'adventure', 'or'), ('is', 'either', 'a', 'daring', 'adventure', 'or', 'nothing'), ('is', 'either', 'a', 'daring', 'adventure', 'or', 'nothing', 'at'), ('is', 'either', 'a', 'daring', 'adventure', 'or', 'nothing', 'at', 'all'), ('either',), ('either', 'a'), ('either', 'a', 'daring'), ('either', 'a', 'daring', 'adventure'), ('either', 'a', 'daring', 'adventure', 'or'), ('either', 'a', 'daring', 'adventure', 'or', 'nothing'), ('either', 'a', 'daring', 'adventure', 'or', 'nothing', 'at'), ('either', 'a', 'daring', 'adventure', 'or', 'nothing', 'at', 'all'), ('a',), ('a', 'daring'), ('a', 'daring', 'adventure'), ('a', 'daring', 'adventure', 'or'), ('a', 'daring', 'adventure', 'or', 'nothing'), ('a', 'daring', 'adventure', 'or', 'nothing', 'at'), ('a', 'daring', 'adventure', 'or', 'nothing', 'at', 'all'), ('daring',), ('daring', 'adventure'), ('daring', 'adventure', 'or'), ('daring', 'adventure', 'or', 'nothing'), ('daring', 'adventure', 'or', 'nothing', 'at'), ('daring', 'adventure', 'or', 'nothing', 'at', 'all'), ('adventure',), ('adventure', 'or'), ('adventure', 'or', 'nothing'), ('adventure', 'or', 'nothing', 'at'), ('adventure', 'or', 'nothing', 'at', 'all'), ('or',), ('or', 'nothing'), ('or', 'nothing', 'at'), ('or', 'nothing', 'at', 'all'), ('nothing',), ('nothing', 'at'), ('nothing', 'at', 'all'), ('at',), ('at', 'all'), ('all',)]

**5. Explain Lemmatization ?**

**Ans:** Stemming and Lemmatization are Text Normalization (or sometimes called Word Normalization) techniques in the field of Natural Language Processing that are used to prepare text, words, and documents for further processing.

**Stemming** is the process of reducing inflection in words to their root forms such as mapping a group of words to the same stem even if the stem itself is not a valid word in the Language.

**Lemmatization**, unlike Stemming, reduces the inflected words properly ensuring that the root word belongs to the language. In Lemmatization root word is called Lemma. A lemma (plural lemmas or lemmata) is the canonical form, dictionary form, or citation form of a set of words.

**6. Explain Stemming ?**

**Ans:** Stemming is the process of reducing a word to its word stem that affixes to suffixes and prefixes or to the roots of words known as a lemma. Stemming is important in natural language understanding (NLU) and natural language processing (NLP).

**7. Explain Part-of-speech (POS) tagging ?**

**Ans:** Part-of-speech (POS) tagging may be defined as the process of converting a sentence in the form of a list of words, into a list of tuples. Here, the tuples are in the form of (word, tag). We can also call POS tagging a process of assigning one of the parts of speech to the given word.

In simple words, we can say that POS tagging is a task of labelling each word in a sentence with its appropriate part of speech. We already know that parts of speech include nouns, verb, adverbs, adjectives, pronouns, conjunction and their sub-categories.

Most of the POS tagging falls under **Rule Base POS tagging**, **Stochastic POS tagging** and **Transformation based tagging**.

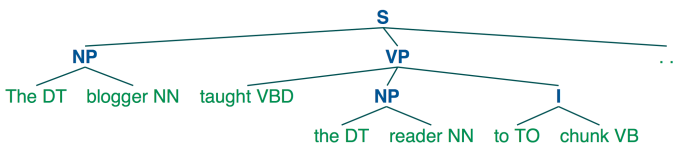
**8. Explain Chunking or shallow parsing ?**

**Ans:** Chunking is somewhere between part of speech (POS) tagging and full language parsing, hence the name shallow parsing. If chunkers are an inbetween stage then why are they relevant? The answer comes down to utility and speed.

POS tagging is very fast but often doesn’t provide a ton of utility for information extraction. It’s helpful to know the POS tags, but when we try to derive information about our text we’re still swimming within the unstructured soup of words in a sentence. Knowing that word 1, 4 and 7 in our sentence are nouns won’t often won’t prove useful enough to help us reliably gain knowledge about what our sentence is actually saying; there’s too much room for mistake.

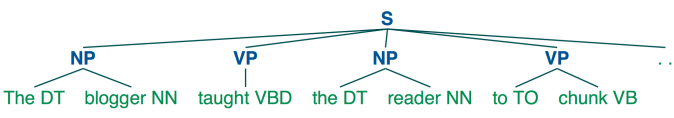
**POS Tags:** 

On the other hand, full parsing is extremely useful: we’re able to understand the syntactic relationship details between the words in our text, and information extraction becomes much easier to define. However, full parsing takes a very long time and will often give you information you don’t necessarily need. Some degree of parsing helps structure our text, but knowing that the determiner in the middle of our sentence is four branches down from the root and part of a nested prepositional clause within a NP clause within the main VP clause…might be overkill, as is the other parse tree produced for our sentence because the syntactic ambiguity in the prepositional clause lends itself to two interpretations: the subject in pajamas shooting an elephant, or the subject shooting the elephant that is wearing his pajamas.

**(Sort of correct) Parsing:** 

And let’s be honest, you’re only here because you want to understand what Twitter is saying about your company’s new line of designer sandwiches or whatever, so all of this extra information is unnecessary and confusing.

Chunking is the happy middle ground that gives you enough information about the syntactic structure to reliably extract meaning from language without burdening your system with unnecessary information.

**Chunking:** 

**9. Explain Noun Phrase (NP) chunking ?**

**Ans:** Text chunking is dividing sentences into non-overlapping phrases. Noun phrase chunking deals with extracting the noun phrases from a sentence. While NP chunking is much simpler than parsing, it is still a challenging task to build a accurate and very efficient NP chunker. The importance of NP chunking derives from the fact that it is used in many applications.

**10. Explain Named Entity Recognition ?**

**Ans:** Named entity recognition (NER) — sometimes referred to as entity chunking, extraction, or identification — is the task of identifying and categorizing key information (entities) in text. An entity can be any word or series of words that consistently refers to the same thing. Every detected entity is classified into a predetermined category. For example, an NER machine learning (ML) model might detect the word “super.AI” in a text and classify it as a “Company”.

NER is a form of natural language processing (NLP), a subfield of artificial intelligence. NLP is concerned with computers processing and analyzing natural language, i.e., any language that has developed naturally, rather than artificially, such as with computer coding languages.

