УДК 81'322
https://doi.org/10.28925/2311-2425.2021.151

NLP 'RECIPES' FOR TEXT CORPORA: APPROACHES TO COMPUTING THE PROBABILITY OF A SEQUENCE OF TOKENS

Monika Porwoł,
State University of Applied Sciences in Racibórz,
Institute of Modern Language Studies (IMLS)
ul. Słowackiego 55, 47-400, Racibórz, Poland
monika-porwol@wp.pl
ORCID iD 0000-0003-1094-3910

Investigation in the hybrid architectures for Natural Language Processing (NLP) requires overcoming complexity in various intellectual traditions pertaining to computer science, formal linguistics, logic, digital humanities, ethical issues, and so on. NLP as a subfield of computer science and artificial intelligence is concerned with interactions between computers and human (natural) languages. It is used to apply machine learning algorithms to text (and speech) in order to create systems, such as machine translation (converting from text in a source language to text in a target language), document summarization (converting from long texts into short texts), named entity recognition, predictive typing, etc. Undoubtedly, NLP phenomena have been implanted in our daily lives, for instance, automatic Machine Translation (MT) is omnipresent in social media (or on the world wide web), virtual assistants (Siri, Cortana, Alexa, and so on) can recognize a natural voice or e-mail services use detection systems to filter out some spam messages. The purpose of this paper, however, is to outline the linguistic and NLP methods for textual processing. Therefore, the bag-of-n-grams concept will be discussed here as an approach to extract more details about the textual data in a string of a grouped words. The n-gram language model presented in this paper (that assigns probabilities to sequences of words in text corpora) is based on findings compiled in Sketch Engine, as well as samples of language data processed by means of NLTK library for Python. Why would one want to compute the probability of a word sequence? The answer is quite obvious — in various systems for performing tasks, the goal is to generate more fluent texts. Therefore, a particular component is required, which computes the probability of the output text. The idea is to collect information on how frequently the n-grams occur in a large text corpus and use it to predict the next word. Counting the number of occurrences can also envisage certain drawbacks, for instance, there are sometimes problems with sparsity or storage. Nonetheless, the language models and specific computing 'recipes' described in this paper can be used in many applications, such as machine translation, summarization, even dialogue systems, etc. Lastly, it has to be pointed out that this piece of writing is a part of an ongoing work tentatively termed LADDER (Linguistic Analysis of Data in the Digital Era of Research) that touches upon the process of datacization[1] that might help to create an intelligent system of interdisciplinary information.

***Key words*:** *linguistics, Natural Language Processing, language modeling, tokenization, term frequency, bag-of-words, probabilistic classification, bag-of-n-grams, n-gram, Sketch Engine, Python, NLTK.*

[1] The term pertains to "the process of transforming information resources accessed by humans into information resources accessed by machines" and it was formulated by an American lexicographer — Erin McKean — and a founder of 'Reverb' that compiles the online dictionary 'Wordnik'.

Порвол М.

Прийоми обробки природних мов для корпусів текстів: підходи до калькуляції вірогідності послідовності лем

Дослідження штучних надбудов для обробки природних мов (ОПМ) вимагає подолання низки проблем у багатьох напрямах традиційних досліджень, пов'язаних із комп'ютерними науками, формальною лінгвістикою, логікою, цифровою гуманітаристикою, етичними традиціями тощо. Як напрям всередині комп'ютерних наук ОПМ вивчає взаємодію між мовами програмування та людськими (природними) мовами. Завдяки застосуванню машинних алгоритмів навчання до текстів (письмових і усних) утворюються такі системи, як машинний переклад (міжмовне накладання з однієї мови на іншу), реферування документів (накладання довгого тексту та скороченого відповідника), розпізнавання номінацій, інтелектуального уведення тексту і таке інше. Безсумнівно, технології ОПМ глибоко інкорпоровані у наше повсякдення. Наприклад, машинний переклад (МП) вбудовано в соціальні мережі та Інтернет, віртуальні помічники (Сірі, Кортана, Алекса тощо) розпізнають голос або диференціюють текст електронної пошти для відфільтровування спаму. Однак мета цієї розвідки — окреслити лінгвістичні й ОПМ методи та підходи до обробки текстів. У зв'язку з цим розглядаються поняття N-грамних кластерів як один із підходів до деталізації текстуальних даних у потоці певних послідовностей лем. Представлена в цій розвідці мовна N-грамна модель (що приписує вірогідність певним послідовностям лем у текстових корпусах), базується на даних, отриманих за допомогою Sketch Engine, а також прикладах мовних даних, опрацьованих у бібілотечних пакетах природних мов мови програмування Python. Питання обчислення вірогідностей послідовностей лексем вирішує очевидні труднощі: виконання команд у різних системах потребує природних формулювань тексту. Відповідно, необхідно мати вузол, який обчислює вірогідність тексту на виході. Опрацювавши інформацію про частоту вживань мовних N-грам у текстах великого корпусу, можна передбачати наступне слово. Калькуляція окремих слововживань може мати свої недоліки, наприклад виникають проблеми з обмеженнями збереження. Проте викладені в статті мовні моделі та окремі прийоми обчислень мають широкий спектр застосування, наприклад у машинному перекладі, реферуванні, лінійних діалогових системах тощо. Важливо зазначити, що ця розвідка є частиною тривалого проєкту LADDER — мовний аналіз даних в епоху цифрових досліджень, який стосується параметрування даних[2] і допомагає створити інтелектуальну систему міждисциплінарної інформації.

Ключові слова: *лінгвістика, обробка природних мов, мовне моделювання, лексемізація, частотність термінів, N-грамний кластер, N-грамна модель, Скетч Енджін, Python, бібілотечний пакет природніх мов.*

Introduction

Unquestionably, Artificial Intelligence (AI) is continuously managing (super)human level of performing various tasks with the usage of available technologies. Nonetheless, AI is not able to deal with complex phenomena regarding decision making, problem-solving, creative assignments yet.

Therefore, a kind of synergy between humans and machines has been treated as the so-called hybrid intelligence systems by means of which "socio-technical ensembles and their human and AI parts can co-evolve to improve over time" (Dellerman et. al, 2019). Therefore, it can be claimed that those interconnected 'aggregates' have the ability to accomplish detailed and complicated tasks by combining both human and artificial intelligence to obtain extraordinary effects and constantly make progress by learning from each other (Dellerman et al, 2019).

NLP: Multifaceted perspectives

NLP is a branch of artificial intelligence (AI) that deals with the abovementioned interaction (between computers and humans using the natural language); therefore, it is concerned with building the set of procedures and technology tools used to aid computers to read, understand, and derive meaning from the natural language.

Although the concept itself is fascinating, the real value behind this technology comes from the use cases. In general, in organizations, data scientists can determine customers' opinions about products and services by extracting and identifying information from social media by means of 'sentiment analysis', which displays their choices and decision drives. Some companies (e.g., Yahoo or Google) can filter and classify messages with NLP techniques of analyzing text in emails that are transferred through their servers and stopping spam before they even reach inboxes.

[2] Термін «параметрування даних» у значенні «процес трансформації доступу до інформації, призначеної для людини у доступ даних для машин», був уведений у 2017 році американським лексикографом Еріном Маккіном, який заснував проєкт "Reverb" для укладання онлайн-словника 'Wordnik'.

Interestingly, the NLP team at MIT developed a system to determine whether a news source is accurate or politically biased; therefore, if it can either be trusted or identified as fake news. There are also examples of intelligent voice-driven interfaces (e.g., Apple's Siri or Amazon's Alexa) that use NLP to respond to vocal prompts to suggest specific actions (for instance, to turn on the lights at home) or inform about particular matters (e.g., the weather forecast, the best restaurant nearby, the shortest route to the city center, etc.). The main hindrance, however, is connected with the fact that the process of understanding and manipulating language is extremely intricate from the computational perspective.

NLP is considered a difficult problem in computer science since it is not an easy task teaching machines to understand how people communicate. The rules that dictate the passing of information using natural languages are not easy for computers to understand, because a language is connected with "cognitive capacities that enable men to understand the world with ever more refined conceptual tools, and it is embedded in their experience of the world" (Saeed, 2009: 12). Some of these rules can be high-leveled and abstract. A comprehensive understanding of the human language requires knowledge of both the words and how the concepts are connected to deliver the intended message. While people can easily master the language, the ambiguity and imprecise characteristics of the natural languages are what makes NLP difficult for machines to implement (Eisenstein, 2019).

In linguistics, as the field that is concerned with the nature of language and communication (Akmajian et al, 1997), especially computational linguistics entails applying algorithms to identify and extract the natural language rules such that the unstructured language data is converted into a form that computers can process. When the text has been provided, the computer will utilize algorithms to extract the meaning associated with every sentence and collect the essential data from them. Sometimes, the computer (that is not trained in a satisfactory manner via machine learning techniques) may fail 'to decipher' the meaning of a sentence well and that may lead to obscure results.

In order to build complex computer programs, machine learning (as a specific subset of AI) trains and manipulates a machine how to learn by means of predictive analysis and deep learning techniques. Resurging interest in machine learning is due to the same factors that have made data mining and Bayesian analysis more popular in recent years. Constantly growing volumes and varieties of available data mean that it is possible to quickly and automatically produce models that can analyze bigger, more complex data and deliver faster, more accurate results — even on a very large scale. By building precise models, there are better chances of identifying profitable opportunities in organizations or even avoiding unknown risks.

Moreover, NLP plays quite a paramount role in emerging interdisciplinary spheres of knowledge (Eisenstein, 2019), such as digital humanities, computational social science, information extraction / retrieval (i.e., the latent semantic analysis), text mining (i.e., data mining techniques and scalable algorithms) or ethics (i.e., questions regarding access, bias, labor, privacy, and internet freedom).

Subject matters in linguistic approach to NLP.

A great number of scientific journals, as well as conferences and / or symposia that deal with NLP, cover syntactic and semantic methods of analysis (i.e., of how to complete various tasks).

First of all, syntax refers to the arrangement of words in a sentence and / or sequence of words in a way that they make grammatical sense. In NLP, syntactic analysis is used to assess how the natural language aligns with the grammatical rules. Computer algorithms are used to apply grammatical rules to a group of words and derive meaning from them. Some syntax techniques that can be used are (1) lemmatization: reducing the various inflected forms of a word into a single form for easy analysis, (2) morphological segmentation: dividing words into individual units called morphemes, (3) word segmentation: dividing a large piece of continuous text into distinct units, (4) part-of-speech tagging: identifying the part of speech for every word, (5) parsing: undertaking grammatical analysis for the provided sentence, (6) sentence breaking: placing sentence boundaries on a large piece of text (7) stemming: cutting the inflected words to their root form.

On the other hand, both the semantic theory that refers to the meaning that is conveyed by a text (Geeraerts, 2010) and the semantic analysis (as one of the difficult aspects of NLP) involve applying computer algorithms to deal with the meaning and interpretation of words. There are particular methods of semantic analysis: (1) named entity recognition (NER): determining the parts of a text that can be identified and categorized into preset groups (for example, names of people and names of places), (2) word sense disambiguation: giving meaning to a word based on the context, (3) natural language generation: using databases to derive semantic intentions and converting them into human language.

In addition, it must be underlined that NLP plays a critical role in supporting machine-human interactions based on linguistic phenomena; nonetheless, there are some contradictory facts regarding what actually are the core factors that linguists take into consideration in the language analysis versus NLP experts' expectations and goals (see *Table 1*):

Table 1

Linguistics vs. NLP

Linguistics wants:	NLP needs:
to know all there is to know about the complex structure mediating the pairings of meanings in natural language	to use the shortest and more reliable way to the meaning(s) in the text(s) being processed
to structure linguistic meaning and to relate it to context	to understand the text and make all the necessary inferences
to distinguish the various levels of linguistic structure, each with its own elements and relations	to use all the linguistic information, which is needed for processing the text(s) without any concern for its source
to draw a boundary between linguistic and encyclopedic information to delimit the extent of linguistic competence; therefore, the limits of the discipline	to use encyclopedic information on par with linguistic information, if necessary, for processing the text(s)
to present its findings formally, preferably as a set of rules in an axiomatic theory	to implement the available information in a practically accessible and convenient way

Each unit of linguistic analysis (word, phrase, sentence, text / discourse) can be viewed from three perspectives: relational, compositional, and distributional which contribute to the understanding of linguistic meaning. For instance, semantic compositionality is the crucial property of natural language according to which the meaning of a complex expression is a function of the meaning of its constituent parts and of the mode of their combination. Conversely, the distributional properties state that words occurring in similar (linguistic) contexts are semantically similar. Therefore, it is customary to use different methods to handle various challenges before binding everything together. Programming languages (e.g., Python or R) are strongly recommended to perform those techniques. Nonetheless, it is pivotal to explain essential concepts beneath them.

NLP 'recipes' for text processing

1. Bag-of-Words. It is a commonly used model that requires a vocabulary stock of known words present in the corpus, as well as a measure of the presence of known words (i.e., frequency of occurrence in the entire corpus). Text document is represented as a numeric vector with each dimension denoting a specific word. This method, however, does not take into account the relative importance of words in the text. Essentially, it creates an occurrence matrix for the sentence or document, disregarding grammar and word order. Word frequencies (or occurrences) are then used as features for training a classifier. Undoubtedly, this approach may reflect some downsides, for instance, the absence of semantic meaning and context, and the facts that stop words add noise to the analysis, and some words are not weighted accordingly. To solve this problem, one approach is to rescale the frequency of words by how often they appear in all texts (not just the one to be analyzed) so that the scores for frequent words, that are also frequent across other texts, get penalized. This approach to scoring is called 'Term Frequency-Inverse Document Frequency (TF-IDF)' and it improves the bag-of-words by weights:

$$TF = \frac{Frequency\ of\ the\ word\ in\ the\ sentence}{Total\ number\ of\ words\ in\ the\ sentence}$$

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF
Term x within document y

$tf_{x,y}$ = frequency of x in y
df_x = number of documents containing x
N = total number of documents

The idea behind the TF-IDF score is that if a word occurs frequently in a specific document, then it is important whereas a word which occurs frequently across all documents in the corpus should be down-weighted to be able to get the words (which are actually important). Nevertheless, this approach still has no context nor semantics.

2. Tokenization. It is the process of segmenting a running text into sentences and words. In essence, it is the task of cutting a text into pieces called tokens, and at the same time throwing away certain characters, such as punctuation. as an example, the NLTK library for Python (i.e., the Natural Language Toolkit as a leading platform for building Python programs in order to synergize with human language data) can be used by means of the following code:

TEXT: "It is a method of extracting essential features from row text so that we can use it for machine learning models. We call it "Bag" of words because we discard the order of occurrences of words. A bag of words model converts the raw text into words, and it also counts the frequency for the words in the text. In summary, a bag of words is a collection of words that represent a sentence along with the word count where the order of occurrences is not relevant".

```
for sentence in sentences:
    words = nltk.word_tokenize(sentence)
    print(words)
    print()
```

OUTPUT (the result of the tokenization):

['It', 'is', 'a', 'method', 'of', 'extracting', 'essential', 'features', 'from', 'row', 'text', 'so', 'that', 'we', 'can', 'use', 'it', 'for', 'machine', 'learning', 'models', '.']

['We', 'call', 'it', "", 'Bag', "", 'of', 'words', 'because', 'we', 'discard', 'the', 'order', 'of', 'occurrences', 'of', 'words', '.']

['A', 'bag', 'of', 'words', 'model', 'converts', 'the', 'raw', 'text', 'into', 'words', ',', 'and', 'it', 'also', 'counts', 'the', 'frequency', 'for', 'the', 'words', 'in', 'the', 'text', '.']

['In', 'summary', ',', 'a', 'bag', 'of', 'words', 'is', 'a', 'collection', 'of', 'words', 'that', 'represent', 'a', 'sentence', 'along', 'with', 'the', 'word', 'count', 'where', 'the', 'order', 'of', 'occurrences', 'is', 'not', 'relevant', '.']

It has to be underlined that tokenization can remove punctuation by easing the path to a proper word segmentation; nevertheless, also triggering possible complications. This method can be particularly problematic when dealing with texts, which contain hyphens, parentheses, and other punctuation marks.

Another 'recipe' could be 'stop words removal' that helps to discard common language articles, pronouns, and prepositions, such as 'and', 'the' or 'to' in English. In this process, some very common words that appear to provide little or no value to the NLP objective are filtered and excluded from the text to be processed, hence removing widespread and frequent terms that are not informative about the corresponding text. Moreover, stop words can be safely ignored by carrying out a lookup in a predefined list of keywords, freeing up database space, and improving processing time.

Since there is no universal list of stop words, these can be either preselected or built from scratch. Therefore, a potential tactical maneuver is to begin by adopting pre-defined stop words and then add words to the list. Additionally, it must be stated that stop words removal can wipe out relevant information and modify the context in a given sentence. For example, while performing sentiment analysis, one might throw an algorithm off track if a stop word, such as 'not' is removed. Under these conditions, one might select a minimal stop word list and add additional terms depending on the specific objective.

3. Probabilistic classification: language models. The theory of probability, loosely speaking, "aims at defining a mathematical structure to describe random outcomes of experiments" (Deisenroth et al., 2020), which generalizes logical reasoning. Computing the probability of a word sequence is used as a constituent element for other concepts, such as machine translation or summarization.

Models that assign probabilities to sequences of words are called language models (LMs). A well-developed model might provide essential information on whether the probability of translation is minor in comparison to more grammatical alternatives.

A. *N-gram language models: basic information*

The first step might be the task of computing the probability of a sequence of tokens is to use a relative frequency estimate (a number of times an event has occurred subdivided by a number of trials).

First of all, describing it in formal terms is required. To put it in simple words, given a text corpus with vocabulary V and a sequence of words: x(1), x(2),…, x(t), the so-called language model essentially computes the probability distribution of the next word: x(t+1):

$$P(\boldsymbol{x}^{(t+1)} | \, \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)})$$

where $\boldsymbol{x}^{(t+1)}$ can be any word in the vocabulary $V = \{\boldsymbol{w}_1, ..., \boldsymbol{w}_{|V|}\}$

A language model, thus, assigns a probability to a piece of text. The probability can be expressed using the chain rule as the product of the following probabilities:

- probability of the first word being x(1);
- probability of the second word being x(2), given that the first word is x(1);
- probability of the third word being x(3), given that the first two words are x(1) and x(2);
- the conditional probability that x(i) is word i, given that the first (i-1) words are x(1), x(2),…, x(i-1).

Therefore, the probability of the text according to the language model is:

$$P(\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(T)}) = P(\boldsymbol{x}^{(1)}) \times P(\boldsymbol{x}^{(2)} | \, \boldsymbol{x}^{(1)}) \times \cdots \times P(\boldsymbol{x}^{(T)} | \, \boldsymbol{x}^{(T-1)}, \ldots, \boldsymbol{x}^{(1)})$$
$$= \prod_{t=1}^{T} P(\boldsymbol{x}^{(t)} | \, \boldsymbol{x}^{(t-1)}, \ldots, \boldsymbol{x}^{(1)})$$

In an n-gram language model, an assumption can be made that the word x(t+1) depends only on the previous (n-1) words. The idea is to collect how frequently the n-grams occur in a corpus and use it to predict the next word:

$$P(\boldsymbol{x}^{(t+1)} | \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(1)}) = P(\boldsymbol{x}^{(t+1)} | \overbrace{\boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(t-n+2)}}^{n\text{-1 words}})$$

This equation, on applying the definition of conditional probability yields:

$$\text{prob of a n-gram} \rightarrow P(\boldsymbol{x}^{(t+1)}, \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(t-n+2)})$$
$$=$$
$$\text{prob of a (n-1)-gram} \rightarrow P(\boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(t-n+2)})$$

To compute the probabilities of these n-grams and n-1 grams, they need to be counted in a large text corpus. Generally, the probability of n-gram can be given by the following formula:

$$\frac{\text{count}(\boldsymbol{x}^{(t+1)}, \boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(t-n+2)})}{\text{count}(\boldsymbol{x}^{(t)}, \ldots, \boldsymbol{x}^{(t-n+2)})}$$

B. *Sketch Engine's n-grams: exemplification*

In order to present what word is likely to follow a given sequence, a probability can be assigned to each possible next word, for instance in a text: "*However, **our compositional representation with a** simple cosive classifier still achieves the best performance [...]".*

Bearing in mind the fact that that the simplest language model is the n-gram (i.e., a sequence of *N* words), the task of computing the probability of a word

w given some history *h: P(w|h)* was performed. (see *Figure 1*); therefore, *P(a|our compositional representation with).*

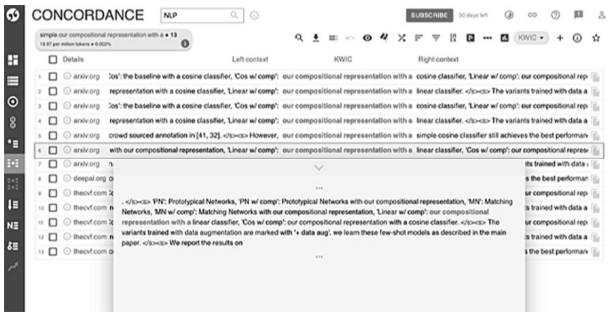

Figure 1. Sketch Engine concordance description of "our compositional representation with a"

The way to estimate this probability is from relative frequency counts. Therefore, the number of times '***our compositional representation with***' was checked in the corpora, as well as the number of times this followed by '***a***', based on the following formula:

$$P(a|our\ compositional\ representation\ with) = \frac{C(our\ compositional\ representation\ with\ a)}{C(our\ compositional\ representation\ with\ )}$$

Each corpus content compiled via Sketch Engine is based on texts found on the web. Texts are divided into tokens by a tokenizer (specific for each language). Two corpora were created: NLP_English (total frequency: 24,547) and NLP_Polish (total frequency: 871,056), providing the following counts of tokens and their total frequencies subdivided according to specific n-grams (from 2 to 6).

	NLP_ENGLISH: 24,547		***NLP_POLISH: 871,056***	
0: n-grams	**items**	**total frequency**	**items**	**total frequency**
2: bi-grams	529	5,208	6,657	141,281
3: tri-grams	119	848	5,111	47,561
4: quad-grams	34	221	2,386	19,388
5: penta-grams	16	93	1,471	11,445
6: hexa-grams	8	46	1,033	7,678

C. *N-grams: other 'recipes' based on NLTK library for Python'*

Supposedly, one would like to apply a different technique in generating a hexagram of a sentence: "this is a foo bar sentences and i want to ngramize it". The n-gram tool producing a frequency list of tokens/ items sequences (i.e., wordforms as the smallest units that each corpus divides to). The following code and its solution are presented below:

```
from nltk import ngrams

sentence = 'this is a foo bar sentences and i want to ngramize it'

n = 6
sixgrams = ngrams(sentence.split(), n)

for grams in sixgrams:
print(grams)
```

OUTPUT:
('this', 'is', 'a', 'foo', 'bar', 'sentences')
('is', 'a', 'foo', 'bar', 'sentences', 'and')
('a', 'foo', 'bar', 'sentences', 'and', 'i')
('foo', 'bar', 'sentences', 'and', 'i', 'want')
('bar', 'sentences', 'and', 'i', 'want', 'to')
('sentences', 'and', 'i', 'want', 'to', 'ngramize')
('and', 'i', 'want', 'to', 'ngramize', 'it')

From a computational perspective, two additional 'recipes' are executed by two different Python functions: (1) tokenize (generating single tokens of the text) and (2) message.split (showing a list of n-grams up to a maximum of 6 tokens in a sequence).

```
(1)
from nltk.util import ngrams
text = "I am aware that nltk only offers bigrams and trigrams, but is there a way to split my text in four-grams, five-grams, or even hundred-grams"
tokenize = nltk.word_tokenize(text)
tokenize
```

OUTPUT:
['I',
'am',
'aware',
'that',
'nltk',
'only',
'offers',
'bigrams',
'and',
'trigrams',
',',
'',
'but',
'is',
'there',
'a',
'way',
'to',
'split',
'my',
'text',
'in',
'four-grams',
',',
'five-grams',

```
'or',
'even',
'hundred-grams']

(2)
from nltk.util import everygrams
message = "I am aware that nltk only offers bigrams
and trigrams"
msg_split = message.split()
list(everygrams(msg_split, max_len=6))

OUTPUT:
[('I',),
('am',),
('aware',),
('that',),
('nltk',),
('only',),
('offers',),
('bigrams',),
('and',),
('trigrams',),
('I', 'am'),
('am', 'aware'),
('aware', 'that'),
('that', 'nltk'),
('nltk', 'only'),
('only', 'offers'),
('offers', 'bigrams'),
('bigrams', 'and'),
('and', 'trigrams'),
('I', 'am', 'aware'),
('am', 'aware', 'that'),
('aware', 'that', 'nltk'),
('that', 'nltk', 'only'),
('nltk', 'only', 'offers'),
('only', 'offers', 'bigrams'),
('offers', 'bigrams', 'and'),
('bigrams', 'and', 'trigrams'),
('I', 'am', 'aware', 'that'),
('am', 'aware', 'that', 'nltk'),
('aware', 'that', 'nltk', 'only'),
('that', 'nltk', 'only', 'offers'),
('nltk', 'only', 'offers', 'bigrams'),
('only', 'offers', 'bigrams', 'and'),
('offers', 'bigrams', 'and', 'trigrams'),
('I', 'am', 'aware', 'that', 'nltk'),
('am', 'aware', 'that', 'nltk', 'only'),
('aware', 'that', 'nltk', 'only', 'offers'),
('that', 'nltk', 'only', 'offers', 'bigrams'),
('nltk', 'only', 'offers', 'bigrams', 'and'),
('only', 'offers', 'bigrams', 'and', 'trigrams'),
('I', 'am', 'aware', 'that', 'nltk', 'only'),
('am', 'aware', 'that', 'nltk', 'only', 'offers'),
('aware', 'that', 'nltk', 'only', 'offers', 'bigrams'),
('that', 'nltk', 'only', 'offers', 'bigrams', 'and'),
('nltk', 'only', 'offers', 'bigrams', 'and', 'trigrams')]
```

N-grams are the fusion of multiple letters and / or words. They are formed in such a way that even the previous and next items are captured. Nonetheless, the most optimal strategy of achieving the preferred solution can be performed by means of the following code:

```
Import re
def generate_ngrams(text, n):
  # split sentences into tokens
  tokens = re.split("\\s+", text)
  ngrams = []
  # collect the n-grams
  for i in range(len(tokens)-n+1):
    temp=[tokens[j] for j in range(i,i+n]
    ngrams.append(" ".join(temp))
  return ngrams
```

Concluding remarks

The present paper presented some tactical procedures of dealing with text processing by means of NLP. Results indicate that n-gram language models are quite interesting for computing the probability of a sequence of items and more broadly to text corpora analysis.

In the future, I shall continue to experiment with new text processing methods (possibly in new settings) that could be integrated into the existing NLP tools. The analytical work is performed within the framework of the LADDER project (Linguistic Analysis of Data in the Digital Era of Research) that postulates an ambitious plan to create an intelligent system of interdisciplinary information (based on AI & data science).

REFERENCES

1. Abend, O & Rappoport, A. 'The State of the Art in Semantic Representation'. *Proceedings of the Association for Computational Linguistics (ACL)*. [Available online]: https://www.aclweb.org/anthology/P17-1008.pdf
2. Ahmed, B., Cha, S. H. & Tappert, C. (2004). 'Language Identification from Text Using N-gram Based Cumulative Frequency Addition'. *Proceedings of Student/Faculty Research Day*, CSIS, Pace University.
3. Akmajian, A., Demers, R. A., Farmer, A. K. & Harnish, R. M. (1997). *Linguistics: An Introduction to Language and Communication*. 4th ed., MIT Press, Cambridge, MA.
4. Briscoe, T. (2013). 'Introduction to Linguistics for Natural Language Processing'. [Available online]: https://www.cl.cam.ac.uk/teaching/1314/L100/introling.pdf
5. Brown, R. D. (2012). 'Finding and Identifying Text in 900+ Languages'. *Digital Investigation*, 9, pp. 34–43. [Available online]: https://www.cl.cam.ac.uk/teaching/1314/L100/introling.pdf

6. Cavnar, W. B., Trenkle, J. M. (1994). 'N-Gram-Based Text Categorization'. *Proceedings of SDAIR 1994*, 3rd Annual Symposium on Document Analysis and Information Retrieval, UNLV Publications/ Reprographics, pp. 161–175.
7. Deisenroth, M. P., Faisal, A. A. & Ong, C. S. (2020). *Mathematics for Machine Learning.* [Available online]: https://mml-book.github.io/book/mml-book.pdf
8. Dellerman, D., Calma, A., Lipusch, N., et al (2019). 'The Future of Human-AI Collaboration: A Taxonomy of Design Knowledge for Hybrid Intelligence Systems'. *Hawaii International Conference on System Sciences (HICSS)*, Hawaii, USA.
9. Geeraerts, D. (2010). *Theories of lexical semantics*. Oxford University Press, Oxford.
10. Cantos Gómez, P. (2013). *Statistical Methods in Language and Linguistic Research*. Equinox, UK/USA.
11. Eisenstein, J. (2019). *Natural Language Processing*. Massachusetts Institute of Technology Press, Cambridge, Massachusetts/London, England.
12. Hammond, M. (2020). *Python for Linguists*. Cambridge University Press, Cambridge.
13. Jurafsky, D. & Martin, J. H. (2019). *Speech and Language Processing.* [Available online]: https://web.stanford.edu/~jurafsky/slp3/3.pdf
14. Kulkarni, A. & Shivananda, A. (2019). *Natural Processing Recipes: Unlocking Text Data with Machine Learning and Deep Learning using Python*. Apress, Bangalore, Karnataka, India.
15. Lutz, M. *Learning Python* (4th ed.). [*Python. Wprowadzenie*]. O'Reilly Media, Helion, Gliwice.
16. Martin, R. C. (2014). *Clean Code: A Handbook of Agile Software Craftmanship.* [Czysty Kod. Podręcznik Dobrego Programisty]. Helion, Gliwice.
17. Poibeau, T. (2017). *Machine Translation*. The MIT Press, Cambridge (Massachussetts)/ London (England).
18. Raskin, V. (1985). 'Linguistics and Natural Language Processing'. *Proceedings of the Conference on Theoretical and Methodological Issues in Machine Translation of Natural Languages*, Colgate University, Hamilton, New York, August 14–16, 268–282. [Available online]: http://mt-archive.info/TMI-1985-Raskin.pdf.
19. Reinchenbach, H. (1947). *Elements of symbolic logic*. The Macmillan Company, New York, 1947.
20. Saeed, J. I. (2009). *Semantics*. Wiley-Blackwell.

Дата надходження статті до редакції: 07.04.2021 р.

Прийнято до друку: 20.04.2021 р.