**N-Grams NLP**

**Notes 1**

N-Grams are phrases cut out of a sentence with N consecutive words. Thus a Unigram takes a sentence and gives us all the words in that we fence. A Bigram takes a sentence and gives us sets of two consecutive words in the sentence. A Trigram gives sets of three consecutive words in a sentence.

Let me explain with an example.

Unigram - [Let] [me] [explain] [with] [an] [example.]

Bigram [let me] [me explain] [explain with] [with an] [an example]

Trigram [let me explain] [me explain with] [explain with an] [with an example]

**Notes 2**

N-grams of texts are extensively used in text mining and natural language processing tasks. An n-gram is a contiguous sequence of n items from a given sample of text or speech. 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.

Formula to calculate number of N-grams in a sentence.

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

Ngramk = X - (N - 1)

Example:

Sentence: I want to learn Machine Learning

Unigram: now calculate number of unigrams in sentence using formula

here, X = 6 and N = 1 (for unigram)

Ngramk = X - (N - 1)

Ngramk = 6 - (1–1) = 6 (i.e. unigram is equal to number of words in a sentence)

[I][want][to][learn][Machine][Learning]

Biagram:

here, X = 6 and N = 2 (for biagram)

Ngramk = X - (N - 1)

Ngramk = 6 - (2–1) = 5

[I want][want to][to learn][learn Machine][Machine Learning]

Trigram:

here, X = 6 and N = 3 (for trigram)

Ngramk = X - (N - 1)

Ngramk = 6 - (3–1) = 4

[I want to][want to learn][to learn Machine][learn Machine Learning]

You can also generate for N=4,5,6 and so on.

**Sklearn tfidfVectorizer()**

**ngram\_rangetuple:**

ngram\_rangetuple (min\_n, max\_n), default=(1, 1)

The lower and upper boundary of the range of n-values for different n-grams to be extracted. All values of n such that min\_n <= n <= max\_n will be used. For example an ngram\_range of (1, 1) means only unigrams, (1, 2) means unigrams and bigrams, and (2, 2) means only bigrams. Only applies if analyzer is not callable.

**max\_dffloat:**

max\_dffloat or int, default=1.0

When building the vocabulary ignore terms that have a document frequency strictly higher than the given threshold (corpus-specific stop words). If float in range [0.0, 1.0], the parameter represents a proportion of documents, integer absolute counts. This parameter is ignored if vocabulary is not None.

**min\_dffloat:**

min\_dffloat or int, default=1

When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold. This value is also called cut-off in the literature. If float in range of [0.0, 1.0], the parameter represents a proportion of documents, integer absolute counts. This parameter is ignored if vocabulary is not None.

**Using stop words:**

Stop words are words like “and”, “the”, “him”, which are presumed to be uninformative in representing the content of a text, and which may be removed to avoid them being construed as signal for prediction. Sometimes, however, similar words are useful for prediction, such as in classifying writing style or personality.

There are several known issues in our provided ‘english’ stop word list. It does not aim to be a general, ‘one-size-fits-all’ solution as some tasks may require a more custom solution. See [NQY18] for more details.

Please take care in choosing a stop word list. Popular stop word lists may include words that are highly informative to some tasks, such as computer.

You should also make sure that the stop word list has had the same preprocessing and tokenization applied as the one used in the vectorizer. The word we’ve is split into we and ve by CountVectorizer’s default tokenizer, so if we’ve is in stop\_words, but ve is not, ve will be retained from we’ve in transformed text. Our vectorizers will try to identify and warn about some kinds of inconsistencies.

**Reference**

**How to Use Tfidftransformer & Tfidfvectorizer?**

<https://kavita-ganesan.com/tfidftransformer-tfidfvectorizer-usage-differences/>

**TF-IDF Vectorizer scikit-learn**

<https://medium.com/@cmukesh8688/tf-idf-vectorizer-scikit-learn-dbc0244a911a>

**AttributeError: 'TfidfVectorizer' object has no attribute 'get\_feature\_names\_out'**

<https://stackoverflow.com/questions/70215049/attributeerror-tfidfvectorizer-object-has-no-attribute-get-feature-names-out>

**Understanding Cosine Similarity in Python with Scikit-Learn**

<https://memgraph.com/blog/cosine-similarity-python-scikit-learn>

**Scikit-learn Count Vectorizers**

<https://medium.com/@cmukesh8688/scikit-learn-count-vectorizers-32b58dee0541>