BAG OF WORDS

"Language is a wonderful medium of communication"

We, as human beings can easily understand the meaning of a sentence within a fraction of a second. But machines fail to process such texts. They need the sentences to be broken down with numerical formats for easy understanding.

In this article, we are going to learn about Bag of Words, a Natural Language Processing text modeling technique, its pros and cons, and will finally learn to implement it using the Python Programming Language.

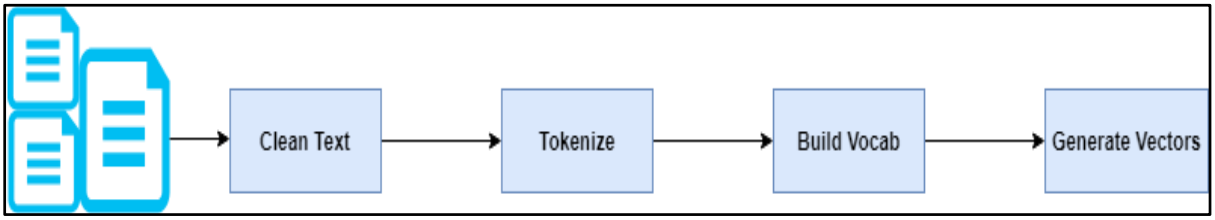
Bag of Words

Bag of Words model is the technique of pre-processing the text by converting it into a number/vector format, which keeps a count of the total occurrences of most frequently used words in the document. This model is mainly visualized using a table, which contains the count of words corresponding to the word itself. In other words, it can be explained as a method to extract features from text documents and use these features for training machine learning algorithms. It tends to create a vocabulary of all the unique words occurring in the training set of the documents.

Its Application:

Bag of words is applied in the field of natural language processing, information retrieval from documents, and also document classifications.

It follows the following steps:



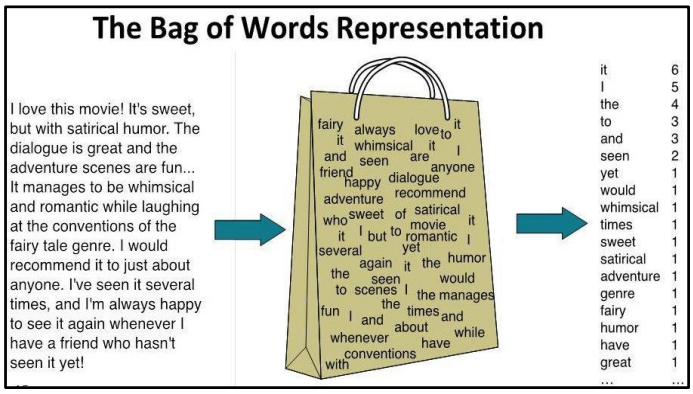
Real-Case Example:

We shall be taking a popular example to explain Bag-of-Words (BoW) and make this journey of understanding a better one. We all love doing online shopping, and yes, it is always important to look for reviews for a product before we commit to buying it. So, we will use this example here.

Here's a sample of reviews about a particular cosmetic product:

- Review 1: This product is useful and fancy
- Review 2: This product is useful but not trending
- Review 3: This product is awesome and fancy

We can actually observe a 100 such contrasting reviews about the product as well as its features, and there is a lot of interesting insights we can draw from it, and finally predict which product is best for us to buy.



Now, the basic requirement is to process the text and convert it into vectorized form. And this can be easily done using Bag of Words which is the simplest form of text representation in numbers.

As the term says, this embedding represents each sentence as a string of numbers. We will first build a vocabulary from all the three above reviews which consist of these 10 words: 'this', 'product', 'is', 'useful', 'and', 'fancy', 'but', 'not', 'trending', 'awesome'.

Before forming the table, we need to pre-process the reviews i.e., convert sentences into lower case, apply stemming and lemmatization, and remove stopwords.

Now, we can mark the word occurrence with 1s and 0s which is demonstrated below:

	F1 (product)	F2 (useful)	F3 (fancy)	F4 (trending)	F5 (awesome)	Vector (Review)
Review 1	1	1	1	0	0	11100
Review 2	1	1	0	1	0	11010
Review 3	1	0	1	0	1	10101

And that's the core idea behind a Bag of Words (BoW) model where 1 denotes the presence of a word in the sentence review and 0 denotes its absence.

From the above explanation, we can very easily conclude that the BOW model only works when a known word occurs in a document or not, and so it does not consider about meaning, context, and order of the sentences. On other side, it gives the insight that similar documents will have word counts similar to each other. Much precisely, more similar the words in two documents will tend to more similar in the documents.

Its Limitations:

But this word embedding technique has some pitfalls due to which developers prefer using TF-IDF or word2vec when dealing with a large amount of data.

Let's discuss them:

- First issue arises in cases when the new sentences contain new words. If such happens, then the vocabulary size would increase and thereby, the length of the vectors would increase too.
- Additionally, the vectors would also contain many 0s, thereby resulting in a sparse matrix (which is what we would like to avoid)
- Secondly, we are gaining no information about the grammatical section nor are we focussing on the order of words in text.

Practical Implementation in Python

Now, let's have an experience of understanding bag of words using the python programming language.

Step 1: Importing Libraries

Foremostly, we have to import the library NLTK which is the leading platform and helps to build python programs for working efficiently with human language data. Then, we need to put our text as the syntax shown below.

```
import nltk

paragraph = """……"""
```

```
In [1]:
import nltk

In [2]:
paragraph =  """Natural language processing (NLP) is a subfield of linguistics, computer science, information engineering, a
    Challenges in natural language processing frequently involve speech recognition, natural language understand
    Neuro-linguistic programming (NLP) is a pseudoscientific approach to communication, personal development, an
    What are the techniques of NLP?
    To understand this better, we list 5 techniques among the many NLP techniques you can use to elevate your co
    1.Anchoring. This NLP technique is useful to regenerate a resourceful emotion. ...
    2.Belief Change. ...
    3.Mirroring and Rapport. ...
    4.Reframing Thoughts. ...
    5.Creative Visualization (Meditation, Hypnosis)
    In simple terms, NLP represents the automatic handling of natural human language like speech or text, and al
    NLP can help you with lots of tasks and the fields of application just seem to increase on a daily basis. Le
    NLP enables the recognition and prediction of diseases based on electronic health records and patient's own
    Organizations can determine what customers are saying about a service or product by identifying and extracti
    An inventor at IBM developed a cognitive assistant that works like a personalized search engine by learning
    Companies like Yahoo and Google filter and classify your emails with NLP by analyzing text in emails that fl
    To help identifying fake news, the NLP Group at MIT developed a new system to determine if a source is accur
    Amazon's Alexa and Apple's Siri are examples of intelligent voice driven interfaces that use NLP to respond
    Having an insight into what is happening and what people are talking about can be very valuable to financial
    NLP is also being used in both the search and selection phases of talent recruitment, identifying the skills
    Powered by IBM Watson NLP technology, LegalMation developed a platform to automate routine litigation tasks
```

Step 2: Preprocessing the text

Those words which do not hold any significance to be used in our model implementation are known as Stopwords, and so it's necessary to remove them. Removing punctuations, alphanumeric characters will also help in better result.

```
## Cleaning the texts

print(corpus)
```

```
In [3]:
# Cleaning the texts
import re
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer
from nltk.stem import WordNetLemmatizer

ps = PorterStemmer()
wordnet=WordNetLemmatizer()
sentences = nltk.sent_tokenize(paragraph)
corpus = []
for i in range(len(sentences)):
    review = re.sub('[^a-zA-Z]', ' ', sentences[i])
    review = review.lower()
    review = review.split()
    review = [ps.stem(word) for word in review if not word in set(stopwords.words('english'))]
    review = ' '.join(review)
    corpus.append(review)

In [7]:
print(corpus)

['natur languag process nlp subfield linguist comput scienc inform engin artifici intellig concern interact comput human natur
languag particular program comput process analyz larg amount natur languag data', 'challeng natur languag process frequent invo
lv speech recognit natur languag understand natur languag gener', 'nlp use method person develop promot skill self reflect conf
id commun', 'practition appli nlp commerci achiev work orient goal improv product job progress', 'neuro linguist program nlp ps
eudoscientif approach commun person develop psychotherapi creat richard bandler john grinder california unit state', 'techniqu
nlp', 'understand better list techniqu among mani nlp techniqu use elev coach practic', 'anchor', 'nlp techniqu use regener res
ourc emot', 'belief chang', 'mirror rapport', 'refram thought', 'creativ visual medit hypnosi simpl term nlp repres automat han
dl natur human languag like speech text although concept fascin real valu behind technolog come use case', 'nlp help lot task f
ield applic seem increas daili basi', 'let mention exampl nlp enabl recognit predict diseas base electron health record patient
speech', 'capabl explor health condit go cardiovascular diseas depress even schizophrenia', 'exampl amazon comprehend medic ser
vic use nlp extract diseas condit medic treatment outcom patient note clinic trial report electron health record', 'organ deter
min custom say servic product identifi extract inform sourc like social media', 'sentiment analysi provid lot inform custom cho
ic decis driver', 'inventor ibm develop cognit assist work like person search engin learn remind name song anyth rememb moment
need', 'compani like yahoo googl filter classifi email nlp analyz text email flow server stop spam even enter inbox', 'help ide
ntifi fake news nlp group mit develop new system determin sourc accur polit bias detect news sourc trust', 'amazon alexa appl s
```

Step 3: BoW Model

It is not needed to code BOW whenever we need it. It is already available with many frameworks such as scikit-learn, CountVectorizer.

```
# Creating the Bag of Words model
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()
print(cv)
print(X)
```

```
In [4]:
# Creating the Bag of Words model
from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features = 1500)
X = cv.fit_transform(corpus).toarray()

In [5]:
print(cv)

CountVectorizer(analyzer='word', binary=False, decode_error='strict',
                dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
                lowercase=True, max_df=1.0, max_features=1500, min_df=1,
                ngram_range=(1, 1), preprocessor=None, stop_words=None,
                strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
                tokenizer=None, vocabulary=None)

In [6]:
print(X)

[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 1 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
```

Featured Image