TEXT VECTORIZATION

Question 1) What is Vectorization and why vectorization?

Ans: ***The process of converting words into numbers is called Vectorization,It is also a method*** **to convert string data into numerical data. Since the beginning of the brief history of Natural Language Processing (NLP), there has been the need to transform text into something a machine can understand. That is, transforming text into a meaningful vector (or array) of numbers.**

VARIOUS STEPS TO BE FOLLOWED FOR TEXT VECTORIZATION→

. Text Preprocessing

· Bag of words

· TFIDF

· Word2Vec

STEP A → Text Preprocessing

Includes techniques like removing stopwords, lemmatizing words

Question → What are stop words?

Ans: Any of the words in a language falling under the categories given below is a stop word.

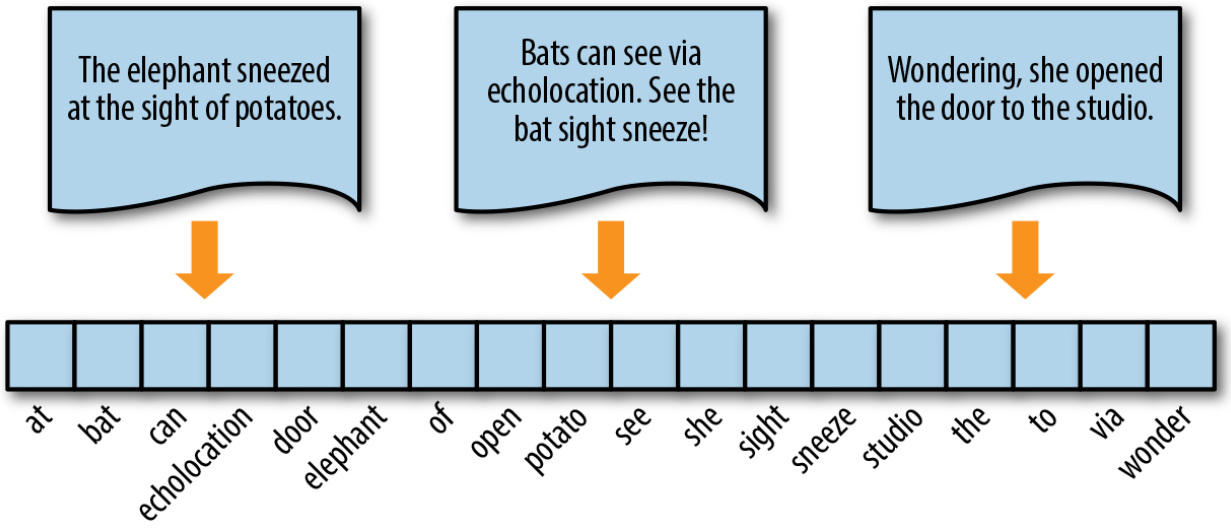
- **Determiners – Determiners tend to mark nouns where a determiner usually will be followed by a noun**
  examples: the, a, an, another
- **Coordinating conjunctions – Coordinating conjunctions connect words, phrases, and clauses**
  examples: for, an, nor, but, or, yet, so
- **Prepositions – Prepositions express temporal or spatial relations**
  examples: in, under, towards, before

Question → what do we mean by lemmatizing words?

Answer: The goal of lemmatizing is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form. For instance:

am, are, is ⇒ be
car, cars, car's, cars' ⇒ car

STEP B → Bag Of Words AND WORD TO VECTOR.



The idea behind this method is straightforward, though very powerful. First, we define a fixed length vector where each entry corresponds to a word in our *pre-defined dictionary* of words. The size of the vector equals the size of the dictionary. Then, for representing a text using this vector, we *count* how many times each word of our dictionary appears in the text and we put this number in the corresponding vector entry.

For example, if our dictionary contains the words {MonkeyLearn, is, the, not, great}, and we want to vectorize the text "MonkeyLearn is great", we would have the following vector: (1, 1, 0, 0, 1).

A DETAILED EXAMPLE TO UNDERSTAND BAG OF WORDS→

There used to be Stone Age"

"There used to be Bronze Age"

"There used to be Iron Age"

"There was Age of Revolution"

"Now it is Digital Age"

Here each sentence is separate document if we make list of the word such that one word should be occur only once than our list looks like as follow:

“There”,”was”,”to”,”be”,”used”,”Stone”,”Bronze,”Iron”,”Revolution”,” Digital”,”Age”,”of”,”Now”,”it”,”is”

So how a word can be converted to vector can be understood by simple word count example where we count occurrence of word in a document w.r.t list. For example- vector conversion of sentence *“There used to be Stone Age”* can be represented as :

“There” = 1

”was”= 0

”to”= 1

”be” =1

”used” = 1

”Stone”= 1

”Bronze” =0

“Iron” =0

”Revolution”= 0

”Digital”= 0

”Age”=1

”of”=0

”Now”=0

”it”=0

”is”=0

So here we basically convert word into vector . By following same approach other vector value are as follow:

“There used to be bronze age” = [1,0,1,1,1,0,1,0,0,0,1,0,0,0,0]

“There used to be iron age” = [1,0,1,1,1,0,0,1,0,0,1,0,0,0,0]

“There was age of revolution” = [1,1,0,0,0,0,0,0,1,0,1,1,0,0,0]

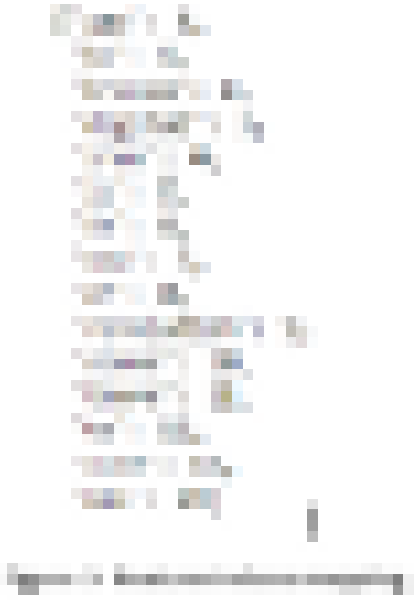
“Now its digital Age” = [0,0,0,0,0,0,0,0,0,0,1,1,0,1,1,1]

The approach which is discussed above is unigram because we are considering only one word at a time . Similarly we have bigram(using two words at a time- for example — There used, Used to, to be, be Stone, Stone age), trigram(using three words at a time- for example- there used to, used to be ,to be Stone,be Stone Age), ngram(using n words at a time)

Hence the process of converting text into vector is called vectorization.

By using CountVectorizer function we can convert text document to matrix of word count. Matrix which is produced here is sparse matrix. By using CountVectorizer on above document we get 5*15 sparse matrix of type numpy.int64.

After applying the CountVectorizer we can map each word to feature indices as shown in figure 3



STEP C→ NORMALIZATION USING TF-IDF FACTOR

TF-IDF stands for Term Frequency-Inverse Document Frequency which basically tells importance of the word in the corpus or dataset. TF-IDF

contain two concept Term Frequency(TF) and Inverse Document Frequency(IDF)

Term Frequency

Term Frequency is defined as how frequently the word appear in the document or corpus. As each sentence is not the same length so it may be possible a word appears in long sentence occur more time as compared to word appear in sorter sentence. Term frequency can be defined as:

$$TF = \frac{No\ of\ time\ word\ appear\ in\ the\ document}{Total\ no\ of\ word\ in\ the\ document}$$

Let's understand with this example

Suppose we have sentence "*The TFIDF Vectorization Process is Beautiful Concept*" and we have to find the find frequency count of these words in five different documents

	The	TFIDF	Vectorization	Process	Is	Beautiful	Concept
Document1	80	20	7	1	100	0	0
Document2	85	25	0	0	115	0	1
Document3	95	10	0	0	95	0	0
Document4	105	9	1	0	150	0	0
Document5	70	2	0	0	80	0	0

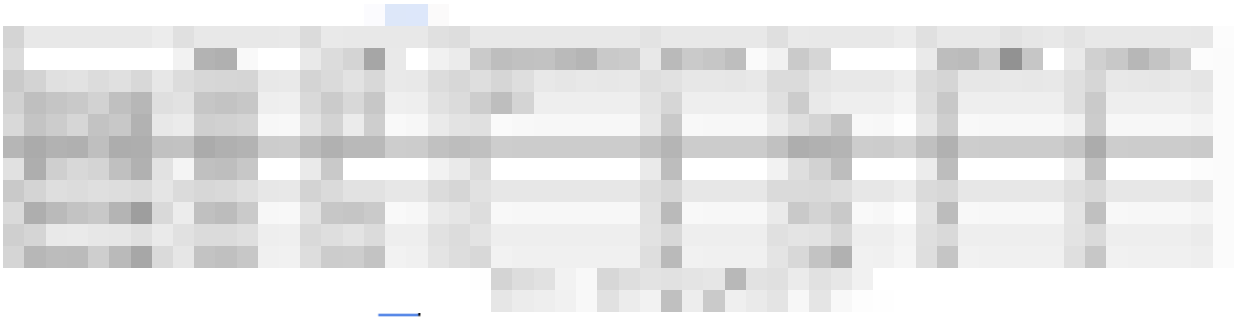
Table 1 -Term Frequency

As shown in Table 1 frequency of '*The*' is maximum in every Document. Suppose frequency of '*The*' in Document6 is 2 million while frequency of '*The*' in Document7 in 3 million. Frequency of '*The*' is very large in Document6 and Document7 so we can add log term to reduce the value of frequency count (log(2 million) =21). Adding log not only dampen the performance of idf but also reduce the frequency count of TF. Hence formula of TF can be defined as:

$$w_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{tf}_{t,d} & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0 & \text{otherwise} \end{cases}$$

When tf = 1 log term will become zero and value will become 1 . Adding 1 is just to differentiate between tf=0 and tf =1

Hence Table 1 can be modified to :



	The	TFIDF	Vectorization	Process	Is	Beautiful	Concept
Document1	1.90	1.30	0.84	1	2	0	0
Document2	1.92	1.39	0	0	2.06	0	1
Document3	1.97	1	0	0	1.97	0	0
Document4	2.02	0.95	1	0	2.17	0	0
Document5	1.84	0.30	0	0	1.90	0	0

Table 2- Adding log factor to TF

Inverse Document Frequency

Inverse Document frequency is another concept which is used for finding out importance of the word. It is based on the fact that less frequent words are more informative and important. IDF is represented by formula:

$$IDF = log_{10} \frac{Number\ of\ Document}{Number\ of\ document\ in\ which\ word\ appear}$$

Let us consider the above example again

	The	TFIDF	Vectorization	Process	Is	Beautiful	Concept
Number of document(N)	5	5	5	5	5	5	5
Word appear(D)	5	1	2	4	5	2	1
N/D	1	5	2.5	1.25	1	2.5	5
Log(N/D)	0	0.698	0.397	0.096	0	0.397	0.698

Table 3- Inverse Document Frequency

In Table 3 most frequent word is '*The*' and '*is* ' but it is least important according to IDF and the word which appear very less such as '*TFIDF*', '*Concept*' are important words. Hence, we can say that IDF of rare term is high and IDF of frequent term is low

TF-IDF

TF-IDF is basically a multiplication between Table 2 (TF table) and Table 3(IDF table) . It basically reduces values of common word that are used in different document. As we can see that in Table 4 most important word after multiplication of TF and IDF is '*TFIDF*' while most frequent word such as '*The*' and '*is*' are not that important

	The	TFIDF	Vectorization	Process	Is	Beautiful	Concept
Documen1	0	0.9704	0.33	0.096	0	0	0
Document2	0	0.97	0	0	0	0	0.698
Document3	0	0.698	0	0	0	0	0
Document4	0	0.66	0.397	0	0	0	0
Document5	0	0.21	0	0	0	0	0

Table 4 – TF-IDF

```
{'age': 0,
 'be': 1,
 'bronze': 2,
 'digital': 3,
 'iron': 4,
 'is': 5,
 'it': 6,
 'now': 7,
 'of': 8,
 'revolution': 9,
 'stone': 10,
 'there': 11,
 'to': 12,
 'used': 13,
 'was': 14}
```

Figure 3- Feature indices mapping

This can be transformed into sparse matrix by using as shown in figure 4

```
array([[1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0],
       [1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0],
       [1, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 0],
       [1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1],
       [1, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0]], dtype=int64)
```

Figure 4- Sparse Matrix

Note — Countvectorizer produces sparse matrix which sometime not suited for some machine learning model hence first convert this sparse matrix to dense matrix then apply machine learning model.

RESEARCH PAPERS:

1) https://pdfs.semanticscholar.org/caf5/b10072c03fc78b4d4a5c007c8e9e1feaa0d4.pdf
2) https://openreview.net/forum?id=HylJtiRqYQ
3) https://ieeexplore.ieee.org/abstract/document/7259377
4) https://ieeexplore.ieee.org/abstract/document/8455685
5) https://ieeexplore.ieee.org/abstract/document/9022842