→ So both avg-w2v
tfidf-w2v
} weighting schemes
sentence(s) ⟶ Vec

21.11 Bag of words (Code sample):-

→ for Bag of words we use Scikit learn

→ we use function called CountVectorizer( )

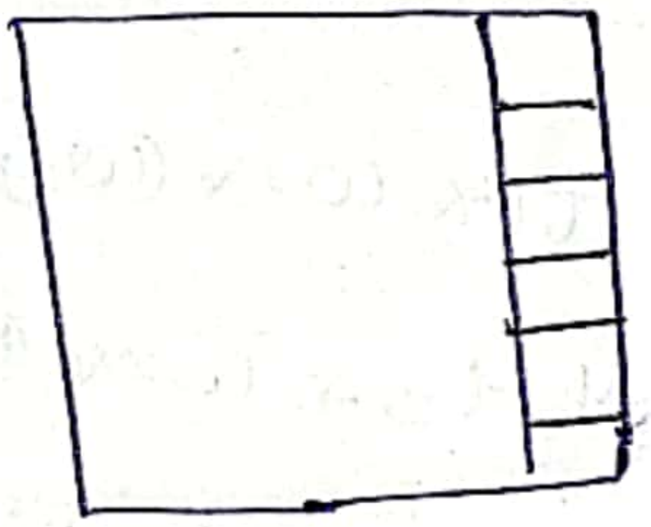
x = CountVectorizer( )

y = x.fit_transform (final['Text'].Value

final['Text'] basically means final data fram

and with a text

Text



→ getting Text column and convert them into Values

→ So final['Text'].values is converting final text ⟶ values (Vectors)

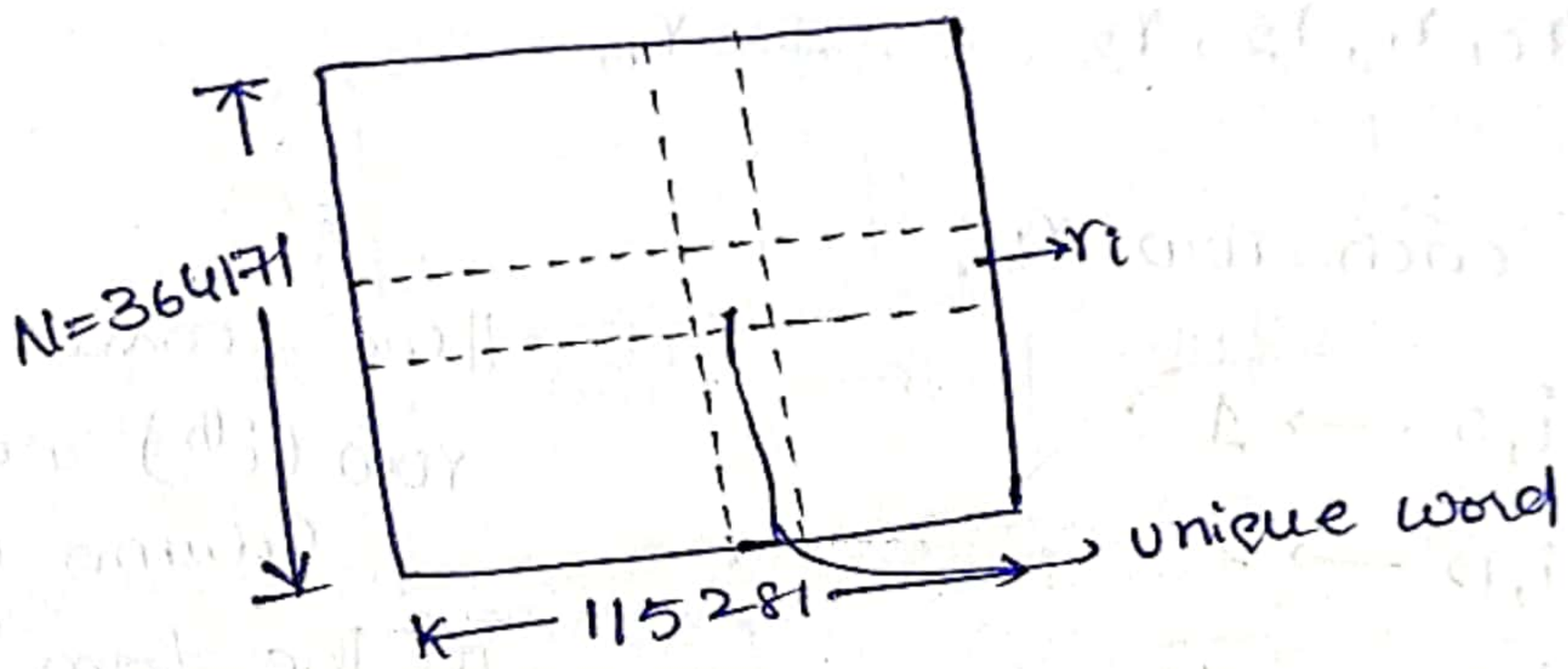
→ type(Y) gives scipy.sparse.csr.csr_matri

as this is sparse matrix

→ y.get_shape( )

get_shape( ) fn gives the shape of the sparse matrix.

here o/p (364171, 115281)
- 364171 → N = no of reviews or documents
- 115281 → no of unique words.

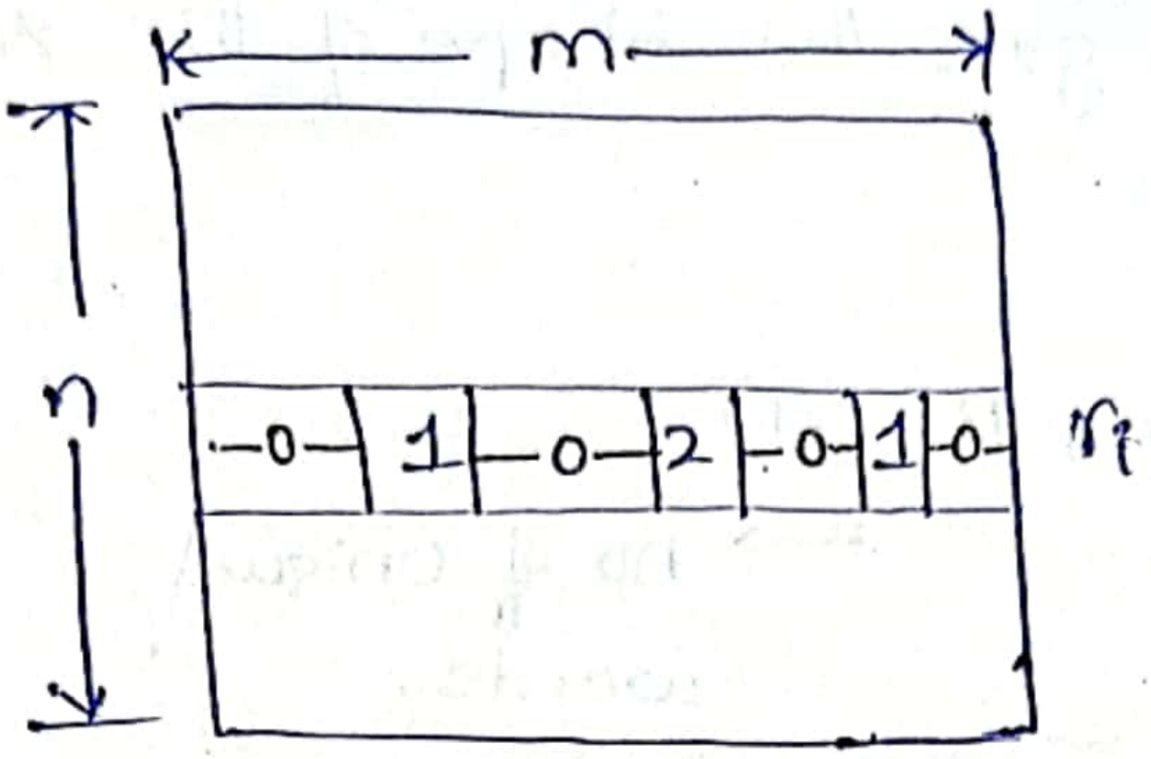


→ In General if we have a matrix of m rows and n columns the ~~time~~/space complexity would be

O(mxn)

→ But in BoW we get the matrix which is a Sparse matrix the space complexity of a sparse matrix can be reduced compared to a normal matrix, or even low.

→ let us consider a sparse matrix. with n rows and m columns



let m=100

$r_0, r_1, r_2, r_3 - - - - - r_n$

for each row r_i:

i,5 → 1

i,12 → 2

i,36 → 1

#we consider the row (i^{th}) & elemen Column with va in the form key value pairs

row, column, → value

key

Like we try to store the non zero values in the python dictionaries with the help of row and column

→ So the space complexity will be reduced to

no of rows * total no of non zero cells

3

why 3 because for each data point we are storing

Row no, column no & data point

ri :- O(m) Space m=100.

as we are storing

row no, col no, Value ⟶ 3* #no of non zero cells

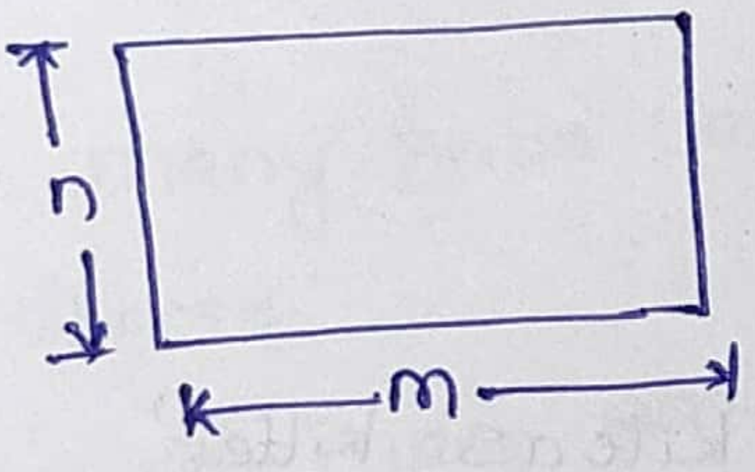
= 9 Values

Instead of 100 values we are storing only 9 values

→ Sparsity of a Matrix :-

for a matrix A with m rows and n columns

A =



#Total no of cells = n x m

k cells have non-zero values rest of them have '0's

where k = no of cells with zero values

n x m = rows x col.

$$\text{Sparsity of } A = \frac{k}{n \times m}$$

→ more Sparse a matrix is the more efficient is Sparse matrix Representation technique.

20.12 Text Preprocessing :-

Code :-

To get set of Stop words

```
# x= Set (Stopwords . words ('english'))

# To Intialize Snowball Stemmer

y= nltk. Stem. SnowballStemmer ('English')
```

regular expressions are the to Recognize patterns in the sentences or words

Ex :- Identifying 'that'

' You can Call that kite as killer'

Ex :- abaa identifying

' abaaaaaaabaaaabaaaa '

20.13 Bi-grams and n-grams

We begin analysis by getting the frequency distribution of the words

```
for x= nltk. FreqDist (all_positive_words)
   y= nltk. FreqDist (all_negative_words)

print ('Most common +ve words.', x. most_common(20

"  "  - negative "  y. most_common(20
```

Observation

→ In this if we see if we perform like this that the most common positive and negative words overlap for eg. 'like' could be used as 'not like' etc,

So it is good idea to consider pairs of consequent words (bi-grams) or q sequence of n.

Note :-

If there are many zeros values in your matrix it is not a Sparse.

→ And if there are many non-zero values in your Matrix it does not refer as dense matrix.

→ Sparse matrix & dense matrix is a way of storing your matrix

→ when to store a matrix as a Sparse matrix & when to store a matrix as a dense matrix.

→ Given a dense matrix

$$\begin{bmatrix} 7 & 0 & 0 \\ 4 & 5 & 0 \\ 0 & 0 & 6 \end{bmatrix}_{3\times3}$$

Sparse representation
(row, column, value)

(0, 0, 7)
(1, 0, 4)
(1, 1, 5)
(2, 2, 6)

} 4 values

where as in dense matrix only 9 values got stored.

- If your matrix is having many non zero values th
there will not be much differences if you store it
in either a sparse matrix form or in dense form
→ use Sparse matrix format mainly when your
matrix has several zero values.