

**CSE523 Machine Learning**

**Weekly Report 7**

**Group Name: Precision Précis**

| **NAME** | **ENROLLMENT NUMBER** |
| --- | --- |
| Twinkle Popat | AU2040216 |
| Kathan Bhavsar | AU2040170 |
| Rushali Moteria | AU2040210 |
| Neel Buddhdev | AU2040176 |

1. **Tasks performed in the week.**

* **Creating functions for different features**

We took the important features related to the text summarization. These include sentence length, TFIDF score, Count Vectoriser score, and Entity count score. To find the most relevant features we have come up with 5 models by pairing up these features.

**Features:**

1. x1 = Entity Count
2. x2 = TFIDF
3. x3 = Countvectorizer
4. x4 = Sentence length

**Total Models:**

* Model-1: Model with x1,x2
* Model-2: Model with x1,x3
* Model-3: Model with x1,x4
* Model-4: Model with x2,x4
* Model-5: Model with x3,x4

We have created functions for each feature. TFIDF, Countvectoriser, and Sentence length functions are the same as provided in the previous reports.

Below is the code to calculate Entity count for each sentence.

| def count\_entity(article,entities):  *# doc = nlp(article)*  *# count\_s = 0;*  CE = []  sentences = nltk.sent\_tokenize(article)  for sent in sentences:  word\_c = 0  doc = nlp(sent)  for ent in doc.ents:  if(ent.label\_ in entities):  word\_c+=1  CE.append(word\_c)  return CE |
| --- |

Firstly, we are defining a function called count\_entity that takes an article and a list of strings representing named entity labels ‘entities’.

The function uses ‘nltk’ library to tokenize the article into sentences. Then we identify entities and count the number of times an entity with a label that matches one of the labels in the list ‘entities’. The count of each entity is then appended to the list ‘CE’. Finally, the function returns the list CE.

* **Creating functions to form the binary array of true output.**

We are creating the binary array in two ways.

Below is the code for the first method to calculate the binary array of true output (which will be given as y input while training different models).

def count\_binArray(article,entites):

text = article

final\_BE1 = []

f = 0

sentences = nltk.sent\_tokenize(text)

for sent in sentences:

f = 0

doc = nlp(sent)

for ent in doc.ents:

if(ent.label\_ in entites):

final\_BE1.append(1)

f = 1

break

if(f==0):

final\_BE1.append(0)

return final\_BE1

Here, we define a function called ‘count\_binArray’ that takes two parameters, article, and entities. We are creating a list ‘final\_BE1’ to store the values generated in the function. Firstly, we are tokenizing the article into sentences using the ‘sent\_tokenize’ function from the nltk library, then we are applying natural language processing using the ‘nlp’ function and store the processed text in the ‘doc’ variable. Next, we are looping through each named entity using ‘for ent in docs.ents’, check if the label of the named entity is present in the entities list, if it is present, append 1 to the ‘final\_BE1’ and if no named entity is found append 0 to ‘final\_BE1’ and return the ‘final\_BE1’ list

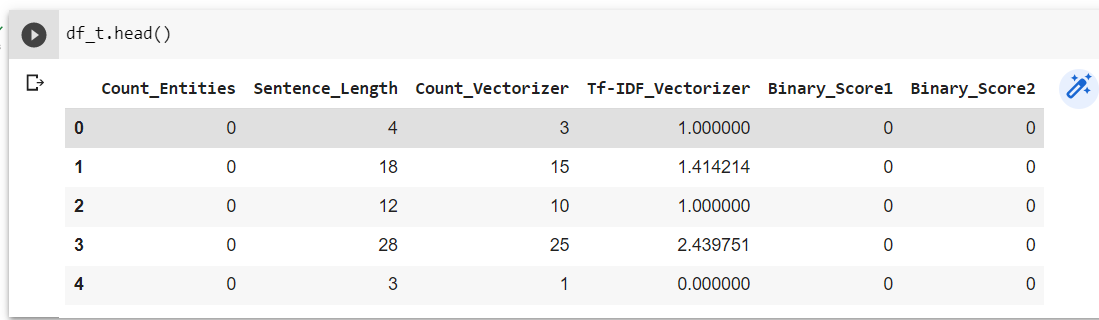
Below is the code for the second method to calculate the binary array of true output (which will be given as y input while training different models).

| def count\_binArray2(article,entities):  text = article  # binDict = {}  binArray = []   sentences = nltk.sent\_tokenize(article)  article\_ec = 0  doc\_art = nlp(article)  for ent in doc\_art.ents:  if(ent.label\_ in entities):  article\_ec+=1    word\_count\_art = len(article.split(" "))   threshold = article\_ec/word\_count\_art   for sent in sentences:  sent\_ec = 0  doc = nlp(sent)  for ent in doc.ents:  if(ent.label\_ in entities):  sent\_ec += 1    score = sent\_ec/len(sent.split(" "))  if(score >= threshold):  binArray.append(1)  else:  binArray.append(0)   return binArray |
| --- |

First, sentence tokenization is done, then nlp() is applied to the entire article which extracts various linguistic features such as named entities. **“doc\_art”** contains all the named entities identified in the article. Then we iterate through the entire article and find the total number of entity words which is stored **“article\_ec”.** Then we set a threshold that is equal to the total number of entity words divided by the total number of words in the article. Then we iterate through each sentence in the article and find the number of entity words in the sentence which is stored in **“sent\_ec”**. Then we give a score to each sentence which is equal to the entity count in the sentence/ total number of words in the sentence. If this score is greater than the threshold then the score assigned to the sentence is “1” in the binary array, otherwise “0”.

1. **Outcomes of the tasks performed.**

We have created five functions for calculating the sentence score for each feature and formed a data frame out of it. Below is the output of the head of the data frame.



1. **Tasks to be performed in the upcoming week.**

In the upcoming week, we plan to train the models mentioned above using different kernel functions, which are Gaussian rbf, linear, polynomial, and sigmoid functions.