

**CSE523 Machine Learning**

**Weekly Report 5**

**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.**

This week we resolved the dimension issue in the SVM model and analyzed the Mid Semester presentation remarks.

1. **Reviewing the Remarks**

How is the reference summary generated?

There are two kinds of reference summaries in our project. One is the abstractive reference summary which is there is the dataset itself. And other is the summary generated in form of a binary array, by Sentence Level Annotation. As discussed in the previous report the binary array is generated by assigning “1” to the sentences which have an entity, else assigning “0”.

How to quantify the importance of the sentence?

The importance of a sentence is quantified using Sentence Level Annotation. Firstly, we have extracted the entities from the extractive summary given in the dataset. As aforementioned, we then assign importance to each sentence based on whether it contains any entity or not. The entities are ['ORGANIZATION', 'PERSON', 'LOCATION', 'DATE', 'TIME']. These entities usually contain important information and help to identify the importance of a sentence.

Why SVM?

One of the reasons to use SVM is that it is more affection with high dimensional spaces where the number of features is greater than the number of entities. This is mostly the case with text data where we have a large number of features (unique words). In addition, it uses kernel functions that map the data to high dimensional space. In high dimensional space, the data points are easily separable. Also, another problem with text data is that every time we will have different kinds of data, which can be linearly separable or not. This information can not be known prior, thus, SVM acts as the best option to get appropriate results because it uses kernel functions that also work with non-linearly separable data. We can use different kernel functions and compare the results to find the one that gives the most accurate summary.

Formulation for the Rouge score

Rouge (Recall-Oriented Understudy for Gisting Evaluation) is used to measure similarities or overlap between reference and generated summary. Itcalculates the ROUGE score for each generated summary using the **rouge.get\_scores()** function from the Rouge library. The r**ouge.get\_scores()** function returns a dictionary with three ROUGE scores: ROUGE-1, ROUGE-2, and ROUGE-L

**Precision** is calculated by dividing the **number of overlapping unigrams by the total** number of unigrams in the **generated summary.**

**Recall** is calculated by dividing the **number of overlapping unigrams by the total number** of unigrams in the **reference summary**.

**F1-score** is calculated as the **harmonic mean of precision and recall:**

F1-score = 2 \* (precision \* recall) / (precision + recall)

1. **Solving the SVM error**

Previously we were not able to train the SVM model using the scikit-learn library of python due to the dimensions of the input matrix and output matrix.

Following output image displays the dimensions of the x\_train(input) and y\_train(actual output) which is used to train the SVM model.

The error we were facing was that the dimensions of the input and output matrices were not compatible with the SVM model functions.

We fixed that error by reshaping the matrices following the code for reshaping the matrices.

| cv\_ar = cv\_arr1.reshape(-1,1) binAr = binArray1.reshape(-1,1) |
| --- |

Now the dimensions of the matrices x\_train(cv\_ar) and y\_train(binAr) are

We have also predicted the output for sample article

Following is the code for finding the input features (Count vectorizer) for that sample text as well as the actual binary array for that sample article which could be used to compare the results of the predicted output.

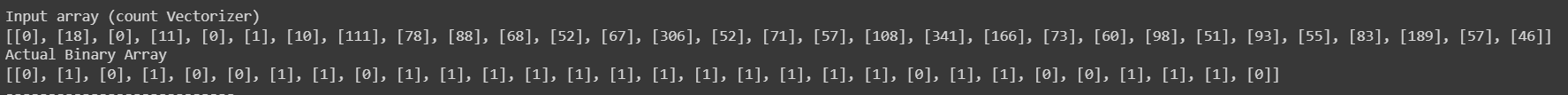
| # Creating a y\_actual value for a test Article Test\_art = df.loc[601].at["article"] actual\_summ = df.loc[601].at["highlights"]  Sents = nltk.sent\_tokenize(Test\_art) Summ\_sent = nltk.sent\_tokenize(actual\_summ)  # Count vectorization for Test Article x\_test = [] for sent in Sents:  sum1 = 0  count\_dict = {}  for word in sent:  if(word not in stopwords and word not in punctuation):  if(word in count\_dict):  count\_dict[word]+=1  else:  count\_dict[word] = 0   sum1 = sum(count\_dict.values())  x\_test.append([sum1])   # Creating an actual Binary Array which will be used to measure the accuracy of the model. # Finding the actual binArray values for comparing the results  entity\_types = ['ORGANIZATION', 'PERSON', 'LOCATION', 'DATE', 'TIME'] words = nltk.word\_tokenize(actual\_summ) tagged = nltk.pos\_tag(words) entities = nltk.chunk.ne\_chunk(tagged) summ\_entity = [] for subtree in entities.subtrees():  if subtree.label() in entity\_types:  summ\_entity.append(subtree.label())  # NER for Test Articel binTestDict = {} for sent in Sents:  binTestDict[sent] = 0  words = [nltk.word\_tokenize(sent) for sent in Sents] pos\_tags = [nltk.pos\_tag(sent) for sent in words]  # Perform named entity recognition (NER) on the POS tagged words ne\_tags = [nltk.ne\_chunk(tagged) for tagged in pos\_tags] entity\_types = summ\_entity   for k,ne in enumerate(ne\_tags):  for subtree in ne.subtrees():  if subtree.label() in entity\_types:  binTestDict[Sents[k]] = 1  y\_actual\_test = [] for h in binTestDict.values():  y\_actual\_test.append([h])  print("Input array (count Vectorizer)") print(x\_test) print("Actual Binary Array") print(y\_actual\_test) print("---------------------------") x\_test = np.array(x\_test) y\_actual\_test = np.array(y\_actual\_test) |
| --- |

Following is the code for predicting the binary array using input array (x\_test)

| y\_pred = clf.predict(x\_test) print(y\_pred) |
| --- |

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

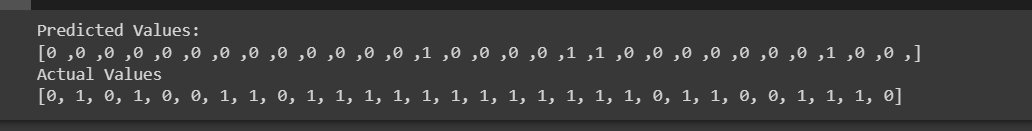
Following is the output of code used for feature extraction and creating the actual output for that using the Named entity recognition concept.



Following is the predicted array output



Following is output consisting both actual output and predicted to compare them



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

In the upcoming week we will try making use of Named Entity Recognition in Count vectorisation. Currently, in count vectorizer we are counting all the words in a sentence and representing their frequencies. However, by making use of NER, we will only count the words that represent an entity and represent in the form of frequency. In addition, we will also implement different kernels of SVM to check which kernel provides best accuracy.