

**CSE523 Machine Learning**

**Weekly Report 3**

**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 learned and applied TF-IDF, Term Frequency-Inverse Document Frequency. We utilized TF-IDF to rank sentences based on their relevance by identifying the sentences that contain more of the important words or terms and have higher TF-IDF scores for those terms. The following steps are involved in the method.

* We have started by breaking the text documents into individual words or tokens by removing the stop words.
* We are now calculating TF(Term-frequency). The process of calculating TF (Term Frequency) involves counting the frequency of each word in the given text, and the resulting value is used to calculate the TF score using a specific formula.

Term Frequency =

* The next step is to calculate IDF(Inverse document frequency). This measures how important the term is in the collection of sentences. The purpose of IDF is to measure how rare or common a term is in a sentence of the text. Terms that are common across all sentences in the text will have a low IDF score, while terms that appear in only a few sentences will have a high IDF score.

IDF = )

* To calculate TF-IDF, Multiply the term frequency by the inverse document frequency for each term in the text. This gives you the TF-IDF score for each term in the text.

Note: Calculation of TF-IDF for each word is done by **TfidfVectorizer()**

**Code Explanation:**

* We will declare all the necessary arrays needed as follows.

| article = [] refehighlights = [] originalLen = [] |
| --- |

* We will append the texts in the article array for future use.
* We also use the originalLen array to compare the final summary length with the original article.
* We will be using a “refehighlights” array to store the summary generated by a human from the data set so that it can be employed to compare the original summary (human generated) with the final summary of our model.

| # Sample array to hold all the article article.append(text1) article.append(text2) article.append(text3) |
| --- |

| # Sample array to hold the original length of the article originalLen.append(len(text1)) originalLen.append(len(text2)) originalLen.append(len(text3)) |
| --- |

* We will declare all the necessary arrays needed as follows.

| article = [] refehighlights = [] originalLen = [] |
| --- |

* We will append the texts in the article array for future use.
* We also use the originalLen array for comparing the length of the final summary with the original article.
* We will be using a refehighlights array to store the summary generated by a human from the data set so that it can be employed to compare the original summary (human generated) with the final summary of our model.

| # Sample array to hold all the article article.append(text1) article.append(text2) article.append(text3) |
| --- |

| # Sample array to hold the original length of the article originalLen.append(len(text1)) originalLen.append(len(text2)) originalLen.append(len(text3)) |
| --- |

| # Sample reference array to hold all the highlights  refehighlights.append(refSumm1) refehighlights.append(refSumm1) refehighlights.append(refSumm1) |
| --- |

* We will initialize an empty array “generatedArrayTFIDF” to store the generated summaries

| # Clearing the array everytime it runs generatedArrayTFIDF = [] |
| --- |

* Loop through each article in the each and split the article into sentences using split()

| i = 0 while(i<len(article)):  sentences = article[i].split(".") |
| --- |

* Create a TFidVectorizer and fit and transform the sentences list into TF-IDF representation using fit\_transform() method. This step calculates the TF-IDF scores for each word in each sentences.

| # Create a TfidfVectorizer  vectorizer = TfidfVectorizer()   # Transform the sentences into a TF-IDF representation  tfidf = vectorizer.fit\_transform(sentences) |
| --- |

* Set n to the number of most important sentences to select. Sort the n highest sentences using argsort().

| n = 2  indices = np.argsort(tfidf.toarray().sum(axis=1))[-n:]  most\_important\_sentences = [sentences[i] for i in indices] |
| --- |

* Join the most important sentences together and append it to the generatedArrayTFIDF array. Print the summary and a separator line. The resulting generatedArrayTFIDF array contains each article's generated summaries, and the newLenTfiDF array contains their corresponding lengths.

| # Generate the text summary  summary = " ".join(most\_important\_sentences) + "."  generatedArrayTFIDF.append(summary)  newLenTfiDF.append(len(summary))  print(summary)  print("------------------------------------------------------------------------------")   i+=1 |
| --- |

1. **Outcomes of the tasks performed**

Following is the output of summaries generated from selecting n most important sentences from the original article.

| [' Bishop John Folda (pictured) of the Fargo Catholic Diocese in North Dakota has exposed potentially hundreds of church members in Fargo, Grand Forks and Jamestown to the hepatitis A The bishop of the Fargo Catholic Diocese in North Dakota has exposed potentially hundreds of church members in Fargo, Grand Forks and Jamestown to the hepatitis A virus in late September and early October.',  " 'We weren't able to take breath or blood tests from him immediately and although blood taken several hours after the collision showed he was marginally under the limit, we maintain he would have been over the limit at the time of the collision and in summing up today, the judge agreed â€ Mr Eccleston-Todd took the decision to pick up his mobile phone whilst driving and, either reading or replying to this text message, was so distracted that he failed to negotiate a left-hand bend, crossing the central white line into the path of Miss Titleyâ€TMs oncoming car.",  " The Cod Army are playing in the third tier for the first time in their history after six promotions in nine years and their remarkable ascent shows no sign of slowing with Jamie Proctor and Gareth Evans scoring the goals at Glanford Park Bristol City, who beat Nigel Cloughâ€TMs men on the opening day, were held to a goalless draw by last season's play-off finalists Leyton Orient while Chesterfield, the League Two champions, were beaten 1-0 by MK Dons, who play Manchester United in the Capital One Cup in seven daysâ€TM time."] |
| --- |

Comparison of the length of original article and generated summary to evaluate the performance of the summary (in future we will be using “Rogue” library of python to evaluate the score of generated summary in comparison to human summary).

| Original Text: 1213 , Summary: 389 Original Text: 4785 , Summary: 566 Original Text: 3461 , Summary: 529 |
| --- |

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

We are planning to use the “Rouge” library of python to get the final evaluation of the correctness score of the Summary generated by our model

What Rough library does is it evaluate the score of how similar the computer generated summary is to Human generated summary (which is the reference summary from our dataset)