CONTENTS

1. Analysis and evaluation of various *Text summarization* Model for summary extraction from financial documents

1.1 What is text summarization-

Text summarization is a process of expressing a bigger paragraph in shorter form with little loss of the actual or important information from the given passage.

1.2 History of text summarization-

The first piece of work to capture attention outside mainstream NLP was Winograd's SHRDLU thesis at MIT in 1971 (Yorick Wilks). It was during 1980s and 1990s that NLP started gaining interest as a topic for research and many developments in this field were kickstarted during this time. Along with the development of NLP, related areas like Statistical Language Processing, Information Extraction and Automatic summarization also gained interest.

Text summarization in early days were done exclusively using rule-based algorithms. It was called "importance evaluator", which worked based on ranking different parts of a text according to their importance. Two important knowledge bases were used by the evaluator: one being the "importance rule base" that made use of IF-THEN rules and other being the "encyclopaedia" which contained domain specific world knowledge represented using a network of frames. The importance rule-based method makes use of a concept called HPN (Hierarchical Propositional Network), where numerical representations are given to conceptual units of extended linear representations (ELR) of sentences to constitute the importance of it.

For example – The News app "Inshorts" gives a small 60 words summary of the latest news in the world.

1.3 Approaches of text summarization –

1. Extractive text summarization –

Extractive text summarization Is a process in which we purely use the sentence given in the passage and generates a summary by calculating the scores of the different sentence present In the input passage.

2. Abstractive text summarization –

In the Abstractive Summarization approach, we work on generating new sentences from the original text. The abstractive method is in contrast to the approach that was described above. The sentences generated through this approach might not even be present in the original text.

Steps for implementation of extractive text summarization-

1. Text cleaning

2. Sentence tokenization

3. Word frequency table

4. Clustering

5. Generating the summary by calcularting the score of each tokensised sentence.

Code of the model –

1.4 Text summarization using word2vec on chinese rocket falling news with metrics

NLTK is a leading platform for building Python programs to work with human language data. It provides easy-to-use interfaces to over 50 corpora and lexical resources such as WordNet, along with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning, wrappers for industrial-strength NLP libraries, and an active discussion forum.

July 29, 2022

[1]:
```
import re
import numpy as np
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords as stop
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]             /Users/pranavwadkar/nltk_data...
[nltk_data]         Package stopwords is already up-to-date!
```

[2]:
```
text ="""Experts have predicted that the latest piece of big Chinese space junk␣
→will fall back to Earth but are not sure where exactly it will land. The␣
→re-entry will be around the end of the month, probably on July 31 at 02:22 UTC␣
→± 17 hours. The mentioned rocket is roughly 21-tonne and was part of the␣
→Wentian space station module. It was launched on Sunday and docked␣
→successfully with the Tiangong space station. A Long March 5B (CZ-5B) rocket␣
→propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on␣
→China's tropical island of Hainan. It appeared to be concerning as experts␣
→have claimed that they are not sure where the debris will land. Holger Krag,␣
→the head of the Space Safety Program Office for the European Space Agency,␣
→told SpaceNews: "It is always difficult to assess the amount of surviving mass␣
→and number of fragments without knowing the design of the object, but a␣
→reasonable 'rule-of-thumb' is about 20-40 per cent of the original dry mass.
→"""
```

Program Office for the European Space Agency, told SpaceNews: “It is always difficult to assess the amount of surviving mass and number of fragments without knowing the design of the object, but a reasonable ‘rule-of-thumb’ is about 20-40 per cent of the original dry mass.”

[4]:
```
count=0
for i in text:
   count+=1
print(count)
```

972

[5]:
```
import re
text1=re.sub(r'[^\w\s\,\.]','',str(text))
text1=re.sub("\d+","",text1)
text1=re.sub(r"(\w)([A-Z])", r"\1 \2", str(text1))
print(text1)


tokenised_words = nltk.word_tokenize(text1)


for i in tokenised_words:
      if i in stopwords.words('english'):
            tokenised_words.remove(i)
print(tokenised_words)


import nltk
nltk.download('punkt')
```

Experts have predicted that the latest piece of big Chinese space junk will fall back to Earth but are not sure where exactly it will land. The reentry will be around the end of the month, probably on July at UTC hours. The mentioned rocket is roughly tonne and was part of the Wentian space station module. It was launched on Sunday and docked successfully with the Tiangong space station. A Long March B C ZB rocket propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on Chinas tropical island of Hainan. It appeared to be concerning as experts have claimed that they are not sure where the debris will land. Holger Krag, the head of the Space Safety Program Office for the European Space Agency, told Space News It is always difficult to assess the amount of surviving mass and number of fragments without knowing the design of the object, but a reasonable ruleofthumb is about per cent of the original dry mass.
['Experts', 'predicted', 'latest', 'piece', 'big', 'Chinese', 'space', 'junk',
'fall', 'back', 'Earth', 'sure', 'exactly', 'land', '.', 'The', 'reentry', 'be',
'around', 'end', 'month', ',', 'probably', 'July', 'U', 'TC', 'hours', '.',
'The', 'mentioned', 'rocket', 'roughly', 'tonne', 'part', 'Wentian', 'space',
'station', 'module', '.', 'It', 'was', 'launched', 'Sunday', 'docked',
'successfully', 'Tiangong', 'space', 'station', '.', 'A', 'Long', 'March', 'B',

```
'C', 'ZB', 'rocket', 'propelled', 'the', 'uncrewed', 'Wentian', 'spacecraft',
'the', 'Wenchang', 'launch', 'centre', 'Chinas', 'tropical', 'island', 'Hainan',
'.', 'It', 'appeared', 'be', 'concerning', 'experts', 'claimed', 'they', 'are',
'not', 'sure', 'the', 'debris', 'will', 'land', '.', 'Holger', 'Krag', ',',
'the', 'head', 'the', 'Space', 'Safety', 'Program', 'Office', 'the', 'European',
'Space', 'Agency', ',', 'told', 'Space', 'News', 'It', 'always', 'difficult',
'assess', 'the', 'amount', 'surviving', 'mass', 'number', 'fragments',
'without', 'knowing', 'the', 'design', 'the', 'object', ',', 'a', 'reasonable',
'ruleofthumb', 'about', 'per', 'cent', 'the', 'original', 'dry', 'mass', '.']

[nltk_data] Downloading package punkt to
[nltk_data]     /Users/pranavwadkar/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

[5]: True

[6]:
```
from nltk.tokenize.treebank import TreebankWordDetokenizer text =
TreebankWordDetokenizer().detokenize(tokenised_words)

print(text)

text=text.replace(',',',')
print(text)
```

Experts predicted latest piece big Chinese space junk fall back Earth sure exactly land . The reentry be around end month, probably July U TC hours . The mentioned rocket roughly tonne part Wentian space station module . It was launched Sunday docked successfully Tiangong space station . A Long March B C ZB rocket propelled the uncrewed Wentian spacecraft the Wenchang launch centre Chinas tropical island Hainan . It appeared be concerning experts claimed they are not sure the debris will land . Holger Krag, the head the Space Safety Program Office the European Space Agency, told Space News It always difficult assess the amount surviving mass number fragments without knowing the design the object, a reasonable ruleofthumb about per cent the original dry mass.

Experts predicted latest piece big Chinese space junk fall back Earth sure exactly land . The reentry be around end month. probably July U TC hours . The mentioned rocket roughly tonne part Wentian space station module . It was launched Sunday docked successfully Tiangong space station . A Long March B C ZB rocket propelled the uncrewed Wentian spacecraft the Wenchang launch centre Chinas tropical island Hainan . It appeared be concerning experts claimed they are not sure the debris will land . Holger Krag. the head the Space Safety Program Office the European Space Agency. told Space News It always difficult assess the amount surviving mass number fragments without knowing the design the object. a reasonable ruleofthumb about per cent the original dry mass.

[7]:
```
word_embeddings = {}
f=open("GoogleNews-vectors-negative450k(5).txt", encoding='utf-8')
for line in f:
        values = line.split()
```

```
        word = values[0]
        coefs = np.asarray(values[1:],dtype='float32')
        word_embeddings[word] = coefs
f.close()
```

[8]:
```
print(len(word_embeddings))
```

```
400000
```

[9]:
```
print("Vocab Size = ",len(word_embeddings))
#300 dimentions of the word
```

```
Vocab Size =  400000
```

[10]:
```
from nltk.tokenize import sent_tokenize
sentences = sent_tokenize(text)
print(sentences)
print(len(sentences))
```

```
['Experts predicted latest piece big Chinese space junk fall back Earth sure exactly land .',
'The reentry be around end month.', 'probably July U TC hours
.', 'The mentioned rocket roughly tonne part Wentian space station module .',
'It was launched Sunday docked successfully Tiangong space station .', 'A Long
March B C ZB rocket propelled the uncrewed Wentian spacecraft the Wenchang
launch centre Chinas tropical island Hainan .', 'It appeared be concerning
experts claimed they are not sure the debris will land .', 'Holger Krag.', 'the
head the Space Safety Program Office the European Space Agency.', 'told Space
News It always difficult assess the amount surviving mass number fragments
without knowing the design the object.', 'a reasonable ruleofthumb about per
cent the original dry mass.']
11
```

[11]:
```
dim=300
sentence_vectors = []
for i in sentences:
    if len(i) != 0:
        v = sum([word_embeddings.get(w, np.zeros((dim,))) for w in i.split()])/
→(len(i.split())+0.001)
    else:
        v = np.zeros((dim,))
    sentence_vectors.append(v)
    print(sentence_vectors)
```

```
[array([-4.77812831e-02, -5.02500526e-02, -3.01189273e-02, 8.30344584e-03, -
        4.43423108e-02, 3.98414053e-02, -2.13045952e-03, -4.01913214e-02,
        7.04687674e-02, -1.09281314e+00, 5.22934469e-02, -4.61638577e-02,
```

4.62929834e-02, 1.11561964e-01, 8.96844212e-02, -4.82347033e-03, --
4.22680492e-02, 1.94115059e-01, 4.03606429e-02, -1.27542297e-01, -
6.88279470e-02, 1.53310778e-01, -6.39725561e-02, 2.72174347e-02,
2.65734277e-02, -1.20560429e-01, -1.55297779e-01, 3.56421782e-02,
7.94850341e-02, -1.40701020e-01, -7.26670878e-02, -3.77314177e-02, -
8.77050213e-02, 4.89468065e-02, -8.62706363e-02, 1.00197318e-01, -
3.91912494e-02, -2.54240387e-02, -9.17132077e-03, 3.61385166e-02,
1.45638216e-01, 4.24436370e-02, -2.65324304e-02, -1.43232184e-01,
4.00085988e-02, -9.56302198e-02, 5.20874596e-02, -3.93706417e-02,
1.45944396e-02, 1.18750018e-01,
1.12289203e-02, 1.31151391e-01, -7.49546052e-02, -1.01503480e-01,
8.26095817e-03, 5.35750309e-02, -7.23486009e-02, -5.90203978e-02,
1.63409222e-03, 3.22508495e-02, 7.29285313e-02, 8.27656149e-02,
1.23772696e-01, 1.77711153e-01, -1.43916474e-01, 1.59911739e-01, -
6.37944006e-03, -5.45697304e-02, 9.89133835e-03, 8.08451452e-02, -
1.55264981e-01, -1.63326239e-01, -1.36657021e-01, -4.74659034e-02, -
3.34825016e-01, 4.70434644e-02, 4.98539936e-03, -1.91097125e-01, -
3.94038065e-02, 1.36322227e-01, -5.89523850e-02, -2.06457709e-02, -
4.60975859e-02, -2.51081276e-02, -3.73858872e-02, -7.

```
0695e-02,  3.10383258e-01, -1.47935932e-02, -6.38724138e-01, -
1.41264393e-01,  1.35959601e-01, -3.25352480e-02, -8.70030983e-02, -
1.26534256e-01, -1.02023798e-01,  4.63589656e-02, 9.07832200e-02, -
5.35976397e-02,  1.31332566e-01,  1.20947134e-01, -3.06566336e-02,
4.70151026e-02,  3.50064096e-01,  2.91000700e-01, 8.04453562e-02,
1.65162888e-01, -1.53388566e-01, -3.86829240e-02, -2.74407562e-01,
8.15358694e-03,  1.46429258e-01, -8.40864935e-02, 2.50103476e-01,
                                  1.62288869e-01, -1.54752325e-01, -1.03184742e-01,
 3.84241289e-04,    1.97985177e-02, -7.81791858e-02,      1.12590741e-01,
-1.38957404e+00, -1.42482855e-02,     2.16642635e-01, -1.69562024e-02,
-1.73807222e-01, -1.92242104e-01, -1.20750272e-01, -1.94992495e-02,
-3.97663268e-02,     2.50025098e-01,    3.01537902e-02, -2.49616841e-01,
-9.45063009e-02,     1.20180785e-01,    6.26658474e-03, -8.05051467e-02,
-2.10944926e-02,     1.40495351e-01,    2.92457553e-01,    1.17765619e-01,
-9.48995019e-02, -1.18952896e-01, -2.63302660e-02,        8.28449218e-02])]
```

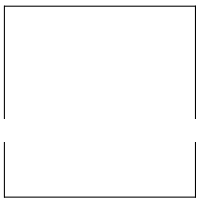
[ ]:

[12]:
```
from sklearn.metrics.pairwise import cosine_similarity import
networkx as nx
import matplotlib.pyplot as plt
sim_mat = np.zeros([len(sentences), len(sentences)])
for i in range(len(sentences)):
      for j in range(len(sentences)):
            if i != j:
                  sim_mat[i][j] = cosine_similarity(sentence_vectors[i].
  →reshape(1,dim),sentence_vectors[j].reshape(1,dim))[0,0]
sim_mat = np.round(sim_mat,3)
print(sim_mat)

# Creating the network graph
nx_graph = nx.from_numpy_array(sim_mat)
plt.figure(figsize=(10, 10))
pos = nx.spring_layout(nx_graph)
nx.draw(nx_graph, with_labels=True, font_weight='bold')
nx.draw_networkx_edge_labels(nx_graph,pos,font_color='red')
plt.show()
```

```
[[0.    0.697 0.753 0.675 0.642 0.675 0.833 0.    0.658 0.788 0.72 ]
 [0.697 0.    0.706 0.624 0.618 0.658 0.741 0.    0.537 0.667 0.578]
 [0.753 0.706 0.    0.591 0.587 0.596 0.788 0.    0.597 0.767 0.697]
 [0.675 0.624 0.591 0.    0.826 0.769 0.593 0.    0.525 0.597 0.626]
 [0.642 0.618 0.587 0.826 0.    0.763 0.62  0.    0.514 0.577 0.536]
 [0.675 0.658 0.596 0.769 0.763 0.    0.671 0.    0.676 0.648 0.545]
 [0.833 0.741 0.788 0.593 0.62  0.671 0.    0.    0.716 0.871 0.687]
 [0.    0.    0.    0.    0.    0.    0.    0.    0.    0.    0.   ]
 [0.658 0.537 0.597 0.525 0.514 0.676 0.716 0.    0.    0.78  0.619]
 [0.788 0.667 0.767 0.597 0.577 0.648 0.871 0.    0.78  0.    0.715]]
```

[0.72 0.578 0.697 0.626 0.536 0.545 0.687 0. 0.619 0.715 0.]]



[13]:
```
scores = nx.pagerank(nx_graph)
print(scores)
```

{0: 0.1045595240446313, 1: 0.09585846880316598, 2: 0.09946932445299371, 3: 0.0959175500019556, 4: 0.09387926504265867, 5: 0.09838797079307565, 6: 0.10566673944065322, 7: 0.014778534868048577, 8: 0.09296329701120515, 9: 0.10411597135257858, 10: 0.09440335418903345}

[14]:
```
count=0
ranked_sentences = sorted(((scores[i],i) for i,s in enumerate(sentences)),␣
  →reverse=True)
arranged_sentences = sorted(ranked_sentences[0:int(len(sentences)*0.5)],␣
  →key=lambda x:x[1])
final_summary = " ".join([sentences[x[1]] for x in arranged_sentences])
```

[15]:
```
print(final_summary)
```

Experts predicted latest piece big Chinese space junk fall back Earth sure exactly land . probably July U TC hours . A Long March B C ZB rocket propelled the uncrewed Wentian spacecraft the Wenchang launch centre Chinas tropical island Hainan . It appeared be concerning experts claimed they are not sure the debris will land . told Space News It always difficult assess the amount surviving mass number fragments without knowing the design the object.

[16]:
```
##Appending a new matric to the given model for getting the results and comparing


from nltk.translate.bleu_score import sentence_bleu


reference = [['i','am','great','with','coding']] candidate=
['i','am','great']
score_1 =sentence_bleu(reference , candidate ,weights=(1,0,0,0))
```

```
print(score_1)


reference = [['i','am','great','with','coding']]
```

```
candidate= ['i','am','great']
score_2 =sentence_bleu(reference , candidate ,weights=(0,1,0,0))

print(score_2)


reference = [['i','am','great','with','coding']] candidate=
['i','am','great']
score_3 =sentence_bleu(reference , candidate ,weights=(0,0,1,0))

print(score_3)


reference_1 = [['i','am','great','with','coding']] candidate_1=
['i','am','great']
score_4 =sentence_bleu(reference_1, candidate_1 ,weights=(0,0,1,0))

print(score_4)

reference_2= [['i','am','great','with','coding']]
candidate_2= ['i','am','great']
score_5 =sentence_bleu(reference_2, candidate_2 ,weights=(0.5,0.5,0.5,0.5))

print(score_5)
```

```
0.513417119032592
0.513417119032592
0.513417119032592
0.513417119032592
7.658479621952598e-155
```

```
/opt/anaconda3/lib/python3.8/site-packages/nltk/translate/bleu_score.py:552:
UserWarning:
The hypothesis contains 0 counts of 4-gram overlaps. Therefore the
BLEU score evaluates to 0, independently of how many N-gram
overlaps of lower order it contains. Consider using lower n-gram
order or use SmoothingFunction()
  warnings.warn(_msg)
```

[17]:
```
reference_text ="""Experts have predicted that the latest piece of big Chinese␣
  →space junk will fall back to Earth but are not sure where exactly it will land.
  →The mentioned rocket is roughly 21-tonne and was part of the Wentian space␣
  →station module.the head of the Space Safety Program Office for the European␣
  →Space Agency, told SpaceNews: “It is always difficult to assess the amount of␣
  →surviving mass and number of fragments without knowing the design of the␣
  →object, but a reasonable ‘rule-of-thumb’ is about 20-40 per cent of the␣
  →original dry mass.”
"""
```

```
tokenised_words_1 = nltk.word_tokenize(reference_text)
tokenised_words_2 = nltk.word_tokenize(final_summary)
#tokenised_words_1=set(tokenised_words_1)
#tokenised_words_2=set(tokenised_words_2)
same_list = list(set(tokenised_words_1).intersection(tokenised_words_2))
overlaped_string = ""
for ele in same_list:
        overlaped_string+= " " + ele

## Now we got the system overlapped string



#to do--->
#Bigrams instead of the words comparing one at a time.
#Visualising the sentences using the embeddings.
#metrics on other models as well.


print(tokenised_words_1)

recall = (len(overlaped_string)/len(reference_text))
print("Recall value is =" ,recall)

precision = (len(overlaped_string)/len(final_summary))
print("precision value is =" ,precision)

## F-score of the given precision and recall

Fscore = 2*((precision*recall)/(precision+recall))

print("F_score value is =" ,Fscore)

from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

['Experts', 'have', 'predicted', 'that', 'the', 'latest', 'piece', 'of', 'big',
'Chinese', 'space', 'junk', 'will', 'fall', 'back', 'to', 'Earth', 'but', 'are',
'not', 'sure', 'where', 'exactly', 'it', 'will', 'land.The', 'mentioned',
'rocket', 'is', 'roughly', '21-tonne', 'and', 'was', 'part', 'of', 'the',
'Wentian', 'space', 'station', 'module.the', 'head', 'of', 'the', 'Space',
'Safety', 'Program', 'Office', 'for', 'the', 'European', 'Space', 'Agency', ',',
'told', 'SpaceNews', ':', '“', 'It', 'is', 'always', 'difficult', 'to',
'assess', 'the', 'amount', 'of', 'surviving', 'mass', 'and', 'number', 'of',
'fragments', 'without', 'knowing', 'the', 'design', 'of', 'the', 'object', ',',
'but', 'a', 'reasonable', '”', 'rule-of-thumb', '”', 'is', 'about', '20-40',
'per', 'cent', 'of', 'the', 'original', 'dry', 'mass', '.', '"']

```
Recall value is = 0.41947565543071164
precision value is = 0.49557522123893805
F_score value is = 0.4543610547667343
```

[17]:
```
{'rouge-1': {'r': 0.4714285714285714, 'p': 0.515625, 'f': 0.49253730844286037},
 'rouge-2': {'r': 0.1797752808988764,
  'p': 0.22535211267605634,
  'f': 0.19999999506328137},
 'rouge-l': {'r': 0.4571428571428713, 'p': 0.5, 'f': 0.477611935308532}}
```

[18]:
```
import nltk
from nltk.tokenize import word_tokenize


bigram_1 =list(nltk.bigrams(reference_text.split()))                    ###bigram of reference␣
  →text
bigram_2 =list(nltk.bigrams(final_summary.split())) ###bigram of final summary same_list =
list(set(bigram_1).intersection(bigram_2))


recall = ((2*len(same_list))/len(reference_text))
print("Recall value is =" ,recall)

precision = ((2*len(same_list))/len(final_summary))
print("Precision value is =" ,precision)

## F-score of the given precision and recall

Fscore = 2*((precision*recall)/(precision+recall))

print("F_score value is =" ,Fscore)

from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

```
Recall value is = 0.056179775280898875
Precision value is = 0.06637168141592921
F_score value is = 0.060851926977687626
```

[18]:
```
{'rouge-1': {'r': 0.4714285714285714, 'p': 0.515625, 'f': 0.49253730844286037},
 'rouge-2': {'r': 0.1797752808988764,
  'p': 0.22535211267605634,
  'f': 0.19999999506328137},
 'rouge-l': {'r': 0.4571428571428713, 'p': 0.5, 'f': 0.477611935308532
```

Code of the model-

Text summarization using Glove on the Chinese rocket falling new

July 29, 2022

[ ]:
```
!wget 'http://nlp.stanford.edu/data/glove.6B.zip'
```

```
--2022-07-29 09:38:08-- http://nlp.stanford.edu/data/glove.6B.zip Resolving
nlp.stanford.edu (nlp.stanford.edu)... 171.64.67.140 Connecting to
nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:80... connected.

HTTP request sent, awaiting response... 302 Found
Location: https://nlp.stanford.edu/data/glove.6B.zip [following] --2022-07-29
09:38:08-- https://nlp.stanford.edu/data/glove.6B.zip Connecting to
nlp.stanford.edu (nlp.stanford.edu)|171.64.67.140|:443... connected.

HTTP request sent, awaiting response... 301 Moved Permanently
Location: https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip [following]
--2022-07-29 09:38:08--  https://downloads.cs.stanford.edu/nlp/data/glove.6B.zip
Resolving downloads.cs.stanford.edu (downloads.cs.stanford.edu)... 171.64.64.22
Connecting to downloads.cs.stanford.edu
(downloads.cs.stanford.edu)|171.64.64.22|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 862182613 (822M) [application/zip]
Saving to: ‘glove.6B.zip’

glove.6B.zip          100%[==================>] 822.24M          5.01MB/s    in 2m 39s

2022-07-29 09:40:47 (5.17 MB/s) - ‘glove.6B.zip’ saved [862182613/862182613]
```

[ ]:
```
!unzip 'glove.6B.zip' !ls

!pwd
```

```
Archive:  glove.6B.zip
  inflating: glove.6B.50d.txt
  inflating: glove.6B.100d.txt
  inflating: glove.6B.200d.txt
  inflating: glove.6B.300d.txt
```

glove.6B.100d.txt glove.6B.300d.txt glove.6B.zip
glove.6B.200d.txt glove.6B.50d.txt sample_data /content

[ ]:
```
import re
import numpy as np
from nltk.corpus import stopwords
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords as stop
```

[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Unzipping corpora/stopwords.zip.

[ ]:
```
text ="""Experts have predicted that the latest piece of big Chinese space junk␣
→will fall back to Earth but are not sure where exactly it will land. The␣
→re-entry will be around the end of the month, probably on July 31 at 02:22 UTC␣→± 17 hours.
The mentioned rocket is roughly 21-tonne and was part of the␣→Wentian space station module.
It was launched on Sunday and docked␣→successfully with the Tiangong space station. A Long
March 5B (CZ-5B) rocket␣→propelled the uncrewed Wentian spacecraft from the Wenchang
launch centre on␣→China's tropical island of Hainan. It appeared to be concerning as experts␣
→have claimed that they are not sure where the debris will land. Holger Krag,␣→the head of the
Space Safety Program Office for the European Space Agency,␣→told SpaceNews: "It is always
difficult to assess the amount of surviving mass␣→and number of fragments without knowing the
design of the object, but a␣→reasonable 'rule-of-thumb' is about 20-40 per cent of the original dry
mass.→"""



count=0
for i in text:
    count+=1
print(count)
```

972

[ ]:
```
for i in text:
        if (i=="-" or i="," or i=")" or i="(" or i=" ' "):
                text=text.replace(i," ")
print(text)
import nltk
nltk.download('punkt')

tokenised_words = nltk.word_tokenize(text)
print(tokenised_words)
```

[nltk_data] Downloading package punkt to /root/nltk_data...

Experts have predicted that the latest piece of big Chinese space junk will fall back to Earth but are not sure where exactly it will land. The re entry will be

around the end of the month probably on July 31 at 02:22 UTC ± 17 hours. The mentioned rocket is roughly 21 tonne and was part of the Wentian space station module. It was launched on Sunday and docked successfully with the Tiangong space station. A Long March 5B CZ 5B rocket propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on China's tropical island of Hainan. It appeared to be concerning as experts have claimed that they are not sure where the debris will land. Holger Krag the head of the Space Safety Program Office for the European Space Agency told SpaceNews: "It is always difficult to assess the amount of surviving mass and number of fragments without knowing the design of the object but a reasonable 'rule of thumb' is about 20 40 per cent of the original dry mass."

```
[nltk_data]     Unzipping tokenizers/punkt.zip.
```

```
['Experts', 'have', 'predicted', 'that', 'the', 'latest', 'piece', 'of', 'big',
'Chinese', 'space', 'junk', 'will', 'fall', 'back', 'to', 'Earth', 'but', 'are',
'not', 'sure', 'where', 'exactly', 'it', 'will', 'land', '.', 'The', 're',
'entry', 'will', 'be', 'around', 'the', 'end', 'of', 'the', 'month', 'probably',
'on', 'July', '31', 'at', '02:22', 'UTC', '±', '17', 'hours', '.', 'The',
'mentioned', 'rocket', 'is', 'roughly', '21', 'tonne', 'and', 'was', 'part',
'of', 'the', 'Wentian', 'space', 'station', 'module', '.', 'It', 'was',
'launched', 'on', 'Sunday', 'and', 'docked', 'successfully', 'with', 'the',
'Tiangong', 'space', 'station', '.', 'A', 'Long', 'March', '5B', 'CZ', '5B',
'rocket', 'propelled', 'the', 'uncrewed', 'Wentian', 'spacecraft', 'from',
'the', 'Wenchang', 'launch', 'centre', 'on', 'China', "'", 's', 'tropical',
'island', 'of', 'Hainan', '.', 'It', 'appeared', 'to', 'be', 'concerning', 'as',
'experts', 'have', 'claimed', 'that', 'they', 'are', 'not', 'sure', 'where',
'the', 'debris', 'will', 'land', '.', 'Holger', 'Krag', 'the', 'head', 'of',
'the', 'Space', 'Safety', 'Program', 'Office', 'for', 'the', 'European',
'Space', 'Agency', 'told', 'SpaceNews', ':', '"', 'It', 'is', 'always',
'difficult', 'to', 'assess', 'the', 'amount', 'of', 'surviving', 'mass', 'and',
'number', 'of', 'fragments', 'without', 'knowing', 'the', 'design', 'of', 'the',
'object', 'but', 'a', 'reasonable', "'", 'rule', 'of', 'thumb', "'", 'is',
'about', '20', '40', 'per', 'cent', 'of', 'the', 'original', 'dry', 'mass', '.',
'"']
```

[ ]:
```
from typing import Text

# Text cleaning

# Removing stop words

count =0
for i in tokenised_words:
    if i in stopwords.words('english'):
        tokenised_words.remove(i)
        count+=1
```

```
print(tokenised_words)
print(count)
```

```
['Experts', 'predicted', 'latest', 'piece', 'big', 'Chinese', 'space', 'junk',
'fall', 'back', 'Earth', 'sure', 'exactly', 'land', '.', 'The', 'entry', 'be',
'around', 'end', 'month', 'probably', 'July', '31', '02:22', 'UTC', '±', '17',
'hours', '.', 'The', 'mentioned', 'rocket', 'roughly', '21', 'tonne', 'part',
'Wentian', 'space', 'station', 'module', '.', 'It', 'was', 'launched', 'Sunday',
'docked', 'successfully', 'Tiangong', 'space', 'station', '.', 'A', 'Long',
'March', '5B', 'CZ', '5B', 'rocket', 'propelled', 'the', 'uncrewed', 'Wentian',
'spacecraft', 'the', 'Wenchang', 'launch', 'centre', 'China', "'", 'tropical',
'island', 'Hainan', '.', 'It', 'appeared', 'be', 'concerning', 'experts',
'claimed', 'they', 'are', 'not', 'sure', 'the', 'debris', 'will', 'land', '.',
'Holger', 'Krag', 'the', 'head', 'the', 'Space', 'Safety', 'Program', 'Office',
'the', 'European', 'Space', 'Agency', 'told', 'SpaceNews', ':', '"', 'It',
'always', 'difficult', 'assess', 'the', 'amount', 'surviving', 'mass', 'number',
'fragments', 'without', 'knowing', 'the', 'design', 'the', 'object', 'a',
'reasonable', '‘', 'rule', 'thumb', '’', 'about', '20', '40', 'per', 'cent',
'the', 'original', 'dry', 'mass', '.', '”']
49
```

[ ]:
```
from nltk.tokenize.treebank import TreebankWordDetokenizer text =
TreebankWordDetokenizer().detokenize(tokenised_words)

print(text)

!pwd
!ls
```

```
Experts predicted latest piece big Chinese space junk fall back Earth sure exactly land .
The entry be around end month probably July 31 02:22 UTC ± 17 hours . The mentioned
rocket roughly 21 tonne part Wentian space station module
.  It was launched Sunday docked successfully Tiangong space station . A Long March 5B
CZ 5B rocket propelled the uncrewed Wentian spacecraft the Wenchang launch centre
China ’ tropical island Hainan . It appeared be concerning experts claimed they are not sure
the debris will land . Holger Krag the head the Space Safety Program Office the European
Space Agency told SpaceNews: “ It always difficult assess the amount surviving mass
number fragments without knowing the design the object a reasonable ‘ rule thumb ’ about
20 40 per cent the original dry mass . ”

/content
glove.6B.100d.txt  glove.6B.300d.txt  glove.6B.zip
glove.6B.200d.txt       glove.6B.50d.txt         sample_data
```

[ ]:
```
word_embeddings = {}
f=open("glove.6B.300d.txt", encoding='utf-8')
for line in f:
            values = line.split()
```

```
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        word_embeddings[word] = coefs
f.close()
```

[ ]:
```
print("Vocab Size = ",len(word_embeddings))
print(word_embeddings['strange'])
#300 dimentions of the word
```

```
Vocab Size =  400000
[-1.2994e-01 -1.2108e-01  2.3404e-01 -6.5949e-02 -1.6139e-01 -2.6273e-01
  6.4785e-02  3.4298e-01  6.3247e-01 -7.8099e-01  2.8563e-01 -1.4144e-01
 -5.1547e-01 -3.0758e-01 -2.8874e-01 -1.4539e-01  1.6981e-01 -1.1697e-01
  6.0420e-02  4.9866e-01  4.6151e-01  6.0865e-01 -3.7695e-01  5.9446e-01
 -1.9058e-01  5.0666e-02  3.1228e-01 -5.0988e-01  6.6872e-02 -4.3259e-02
 -4.2734e-01  4.8310e-02 -8.2898e-01 -3.7193e-01 -8.9425e-01  6.4222e-01
 -5.9454e-02 -1.1659e-01 -4.7404e-01  3.7247e-01  1.6142e-02  1.7383e-01
  2.2184e-01 -8.3836e-02 -8.9908e-02  2.2075e-01 -2.3885e-01 -4.2652e-01
 -4.1107e-02  5.7813e-01  2.9355e-01  1.9546e-01  1.8508e-01  1.6219e-01
 -1.0719e-01  4.2200e-02  2.6822e-01  8.9873e-02  5.8537e-01 -1.7343e-01
  1.4644e-01  2.0913e-02  5.6062e-01  3.8875e-01 -1.9264e-01 -3.4515e-01
  1.2737e-01  2.4755e-01 -1.5910e-01  9.0291e-02 -5.5990e-01 -3.2383e-01
 -2.0325e-01  2.0464e-01 -1.9474e-01  6.0434e-02 -4.3327e-02  1.9150e-01
  4.0405e-01 -1.2075e-01  5.7216e-02  2.7853e-02 -1.0826e-02  3.8338e-01
 -1.1041e-01 -1.3189e-01  7.7873e-01  1.4262e-01  2.4055e-01  1.0141e-01
 -1.5144e-01  1.1939e-01  2.3593e-02 -3.2611e-01 -2.0842e-01 -3.1127e-02
  1.7737e-01 -1.4222e-03  6.9722e-02 -5.5315e-01  1.4351e-01  3.6143e-01
  2.6344e-03  4.5468e-01 -3.7126e-01 -5.5825e-01 -3.4023e-02 -2.1619e-01
 -3.1448e-01  2.2076e-01  1.8764e-01  3.0983e-01 -6.8411e-02  5.7726e-02
  4.7350e-01  1.3628e-01  3.2537e-02  2.3947e-01  2.4394e-01  7.8253e-02
 -1.1361e-01 -4.2543e-01 -3.1025e-01  2.5215e-01  5.8917e-02 -2.6884e-01
 -2.5968e-01  4.7892e-01  1.2704e-01  1.0237e-02  3.3607e-01 -1.7306e-02
 -6.7366e-02 -1.3591e-01 -7.0106e-01 -2.6260e-01 -9.1817e-02 -1.8257e-01
-1.0155e+00 -6.7333e-03 -3.9645e-01  8.8922e-02 -3.4007e-01  5.9025e-01
 -1.1982e-01 -3.7869e-01 -1.5474e-01  4.0696e-01 -9.9119e-02  5.1123e-03
 -3.3980e-01  4.6331e-02 -1.8759e-01 -3.9189e-01  4.6638e-01 -4.7379e-01
 -1.0687e-01 -2.8392e-01 -1.1367e-02 -1.8672e-01  4.6954e-01 -5.6303e-01
  1.8728e-01  8.7934e-05  2.7944e-02  4.9934e-01  5.2732e-02  2.3143e-01
 -1.1890e-01 -2.7643e-01  1.2031e-01 -4.7779e-01  8.1166e-02  6.0715e-01
 -3.0223e-01 -3.8200e-02 -5.1435e-01 -2.6065e-01 -2.3474e-01 -3.3578e-01
 -1.5056e-02  4.8722e-02  4.5163e-01 -1.7380e-01  1.7211e-01 -3.6053e-01
  2.9985e-01 -5.1806e-01  1.2250e-01 -1.2750e-01 -4.5334e-01 -5.8010e-01
  4.0404e-01  1.9526e-01 -4.5161e-01  2.4835e-01 -3.2483e-01  2.2311e-02
 -6.3525e-01  1.8713e-01  1.0523e+00 -1.7336e-02 -1.7120e-01 -3.4297e-01
 -4.9749e-02 -3.8374e-01  8.0236e-02  1.2573e-01  9.4322e-02 -3.5407e-01
 -5.6937e-01  2.8301e-01  8.1505e-01 -2.5306e-01 -8.3156e-02 -8.0925e-01
  8.8053e-01  2.7086e-01 -6.7838e-02  5.0416e-01 -1.9030e-02 -2.2949e-01
 -9.8307e-02  3.2228e-01  2.5643e-02  1.0618e-01  2.5586e-01 -1.9905e-01
```

-2.8647e-02 -4.6936e-02 -2.9679e-01 1.2628e-02 -2.0649e-01 -6.2606e-01 -
7.1152e-02 4.0829e-01 -6.2981e-02 -1.1955e-01 -1.4722e-01 -3.0275e-01
5.8678e-01 -3.8148e-01 -3.5640e-01 6.8469e-01 -4.1410e-01 1.3262e-01 -
7.7308e-02 3.3868e-02 2.3878e-02 8.7179e-02 -7.5092e-02 -4.3837e-01 -
3.5403e-02 2.6383e-01 5.1795e-01 7.8851e-01 2.6547e-01 -3.4678e-01 -3.4751e-
01 -5.6284e-02 9.2502e-02 -2.6949e-01 3.9467e-03 -6.5255e-01 3.6893e-02 -
2.5051e-01 8.3669e-02 -2.4586e-01 -2.7512e-01 3.3930e-01 -9.1539e-02
1.5199e-01 4.1496e-01 -1.7310e-01 5.2575e-01 3.7118e-01 -7.5255e-01 1.9218e-
01 5.7107e-01 1.1798e-01 -3.9411e-01 3.2106e-01 4.1442e-01 1.3218e-01 -
2.4309e-02 -1.0378e-01 -3.2183e-01 1.2037e-01 -2.5500e-01 1.5080e-01
1.6968e-01 1.3122e-01 -3.8051e-01 7.9305e-02 2.4239e-01 -3.4545e-01 -
8.0297e-02 5.4951e-01 2.2896e-01 3.6889e-01]

[ ]:
```
from nltk.tokenize import sent_tokenize
sentences = sent_tokenize(text)
print(sentences)
```

['Experts predicted latest piece big Chinese space junk fall back Earth sure exactly land .',
'The entry be around end month probably July 31 02:22 UTC ± 17 hours .', 'The mentioned
rocket roughly 21 tonne part Wentian space station module .', 'It was launched Sunday
docked successfully Tiangong space station
.', 'A Long March 5B CZ 5B rocket propelled the uncrewed Wentian spacecraft the
Wenchang launch centre China ’ tropical island Hainan .', 'It appeared be concerning
experts claimed they are not sure the debris will land .', 'Holger Krag the head the Space
Safety Program Office the European Space Agency told SpaceNews: “ It always difficult
assess the amount surviving mass number fragments without knowing the design the
object a reasonable ‘ rule thumb ’ about 20 40 per cent the original dry mass .', '""]

[ ]:
```
dim=300
sentence_vectors = []
for i in sentences:
    if len(i) != 0:
        v = sum([word_embeddings.get(w, np.zeros((dim,))) for w in i.split()])/
→(len(i.split())+0.001)
    else:
        v = np.zeros((dim,))
    sentence_vectors.append(v)
    print(sentence_vectors)
```

[array([-4.77812831e-02, -5.02500526e-02, -3.01189273e-02, 8.30344584e-03, -
4.43423108e-02, 3.98414053e-02, -2.13045952e-03, -4.01913214e-02,
7.04687674e-02, -1.09281314e+00, 5.22934469e-02, -4.61638577e-02,
4.62929834e-02, 1.11561964e-01, 8.96844212e-02, -4.82347033e-03, -
1.35964268e-03, 1.15182721e-01, 9.59122729e-02, 9.78880071e-02, -
9.59668778e-04, 2.58554697e-01, 3.02931069e-01, 9.37210847e-02, -
3.80690624e-02, 8.03229118e-02, 3.61856224e-02, -7.72586376e-02,
3.74348404e-02, -8.56117598e-02, -8.49063317e-03, 9.91278587e-02, -
1.50680619e-01, -3.62776488e-02, -7.03295770e-01, 8.16948756e-02,

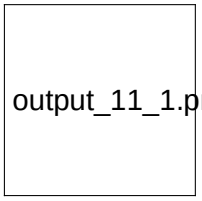
-1.08187440e-01, -5.25164346e-02, -4.22680492e-02, 1.94115059e-01,
4.03606429e-02, -1.27542297e-01, -6.88279470e-02, 1.53310778e-01, -
-01, -1.40573299e-02, 1.75646754e-01, -8.76369178e-02, 1.14753644e-01,
4.03032118e-02, 1.67255441e-01, -1.92911147e-02, 6.42385870e-02,
8.48528123e-02, 1.84682712e-01, 6.22691409e-02, -4.04659042e-02,
1.60971920e-02, -3.88481366e-02, -1.26071582e-02, 2.99610261e-02, -
6.17041525e-02, -9.73041927e-03, -2.44081733e-02, -4.52063724e-02,
1.74580632e-01, 3.17605531e-02, -2.44484033e-01, -4.63915273e-02, -
4.28697415e-02, 1.44222252e-01, -3.13924422e-02, -4.50054032e-02,
6.13423766e-02, -1.79074697e-02, -1.17624693e-01, -1.31004602e-01,
2.46798146e-01, 1.30741955e-02, 1.62182566e-03, 7.07516153e-02, -
4.84880544e-02, -2.50816574e-03,
8.65735625e-02, -6.54400360e-02,   7.20277978e-02, -7.74048472e-03,
2.03953899e-01, -3.18233211e-02, 1.50058731e-01, 5.58420771e-02, -\
1.04375616e-01, -3.74705270e-02, -1.27812177e-01, -9.96513441e-02, -
1.83766231e-01, -3.63666326e-01, -4.37542439e-01, 4.14595380e-02,
3.04045916e-01, 2.

```
       1.33646354e-01, 1.80649340e-01, -3.09680283e-01, -1.20419577e-01, -
    6.91298664e-01, -6.56793192e-02, 4.01828140e-01, 1.46773219e-01,
    1.08801194e-01, 1.80549443e-01, -9.22267660e-02, 2.00589404e-01],
    dtype=float32)]
```

[ ]:
```
from sklearn.metrics.pairwise import cosine_similarity import
networkx as nx
import matplotlib.pyplot as plt
sim_mat = np.zeros([len(sentences), len(sentences)]) for i
in range(len(sentences)):
      for j in range(len(sentences)): if i !=
            j:
                  sim_mat[i][j] = cosine_similarity(sentence_vectors[i].
  →reshape(1,dim),sentence_vectors[j].reshape(1,dim))[0,0]
sim_mat = np.round(sim_mat,3)
print(sim_mat)

# Creating the network graph
nx_graph = nx.from_numpy_array(sim_mat)
plt.figure(figsize=(10, 10))
pos = nx.spring_layout(nx_graph)
nx.draw(nx_graph, with_labels=True, font_weight='bold')
nx.draw_networkx_edge_labels(nx_graph,pos,font_color='red')
plt.show()
```

```
[[0.    0.765 0.684 0.642 0.68  0.833 0.82  0.409]
 [0.765 0.    0.702 0.645 0.656 0.805 0.811 0.315]
 [0.684 0.702 0.    0.832 0.776 0.613 0.687 0.345]
 [0.642 0.645 0.832 0.    0.757 0.62  0.613 0.326]
 [0.68  0.656 0.776 0.757 0.    0.683 0.723 0.419]
 [0.833 0.805 0.613 0.62  0.683 0.    0.864 0.4  ]
 [0.82  0.811 0.687 0.613 0.723 0.864 0.    0.545]
 [0.409 0.315 0.345 0.326 0.419 0.4   0.545 0.   ]]
```



[ ]:
```
scores = nx.pagerank(nx_graph)
print(scores)
```

```
{0: 0.13315905898599906, 1: 0.1297644118056371, 2: 0.12845444953840088, 3:
0.123521040880647, 4: 0.1298993101054548, 5: 0.13276820461321534, 6:
0.13893575288861815, 7: 0.08349777118202781}
```

[ ]:
```
count=0
ranked_sentences = sorted(((scores[i],i) for i,s in enumerate(sentences)),␣
  →reverse=True)
arranged_sentences = sorted(ranked_sentences[0:int(len(sentences)*0.5)],␣
  →key=lambda x:x[1])
final_summary = " ".join([sentences[x[1]] for x in arranged_sentences])
print(final_summary)
```

Experts predicted latest piece big Chinese space junk fall back Earth sure exactly land . A Long March 5B CZ 5B rocket propelled the uncrewed Wentian spacecraft the Wenchang launch centre China ’ tropical island Hainan . It appeared be concerning experts claimed they are not sure the debris will land . Holger Krag the head the Space Safety Program Office the European Space Agency told SpaceNews: “ It always difficult assess the amount surviving mass number fragments without knowing the design the object a reasonable ‘ rule thumb ’ about 20 40 per cent the original dry mass .

[ ]:
```
import nltk
from nltk.translate.bleu_score import sentence_bleu

list_1 = final_summary.split( )
candidate_list = list_1
reference_text =" Experts have predicted that the latest piece of big Chinese␣
  →space junk will fall back to Earth but are not sure where exactly it will land.
  →The mentioned rocket is roughly 21-tonne and was part of the Wentian space␣
  →station module.the head of the Space Safety Program Office for the European␣
  →Space Agency, told SpaceNews: “It is always difficult to assess the amount of␣
  →surviving mass and number of fragments without knowing the design of the␣
  →object, but a reasonable ‘rule-of-thumb’ is about 20-40 per cent of the␣
  →original dry mass."

print(len(reference_text))

reference_list = [reference_text.split( )]

score_1=sentence_bleu(reference_list,candidate_list,weights=(0,1,0,0))
print(score_1)
```

```
532
0.26262626262626265
```

[ ]:

```
reference_text =" Experts have predicted that the latest piece of big Chinese␣
  →space junk will fall back to Earth but are not sure where exactly it will land.
  →The mentioned rocket is roughly 21-tonne and was part of the Wentian space␣
  →station module.the head of the Space Safety Program Office for the European␣
  →Space Agency, told SpaceNews: “It is always difficult to assess the amount of␣
  →surviving mass and number of fragments without knowing the design of the␣
  →object, but a reasonable ‘rule-of-thumb’ is about 20-40 per cent of the␣
  →original dry mass."

tokenised_words_1 = nltk.word_tokenize(reference_text)
tokenised_words_2 = nltk.word_tokenize(final_summary)
#tokenised_words_1=set(tokenised_words_1)
#tokenised_words_2=set(tokenised_words_2)
same_list = list(set(tokenised_words_1).intersection(tokenised_words_2))
overlaped_string = ""
for ele in same_list:
        overlaped_string+= " " + ele


#to do
#Bigrams instead of the words comparing one at a time.
#Visualising the sentences using the embeddings.
#metrics on other models as well.

## Now we got the system overlapped string

recall = (len(overlaped_string)/len(reference_text))
print("Recall value is =" ,recall)

precision = (len(overlaped_string)/len(final_summary))
print("precision value is =" ,precision)

## F-score of the given precision and recall taking harmonic mean

Fscore = 2*((precision*recall)/(precision+recall))

print("F_score value is =" ,Fscore)

!pip install rouge
from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

```
Recall value is = 0.6127819548872181
precision value is = 0.5611015490533563
F_score value is = 0.5858041329739443
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
```

```
Collecting rouge
  Downloading rouge-1.0.1-py3-none-any.whl (13 kB)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from rouge)
(1.15.0)
Installing collected packages: rouge
Successfully installed rouge-1.0.1
```

[ ]:
```
{'rouge-1': {'f': 0.6174496594567812, 'p': 0.575, 'r': 0.6666666666666666}, 'rouge-2': {'f':
  0.3169398857176984,
   'p': 0.30526315789473685,
   'r': 0.32954545454545453},
  'rouge-l': {'f': 0.6174496594567812, 'p': 0.575, 'r': 0.6666666666666666}}
```

[ ]:
```
import nltk
from nltk.tokenize import word_tokenize


bigram_1 =list(nltk.bigrams(reference_text.split()))                    ###bigram of reference␣
  →text
bigram_2 =list(nltk.bigrams(final_summary.split())) ###bigram of final summary same_list =
list(set(bigram_1).intersection(bigram_2))


recall = ((2*len(same_list))/len(reference_text))
print("Recall value is =" ,recall)

precision = ((2*len(same_list))/len(final_summary))
print("Precision value is =" ,precision)

## F-score of the given precision and recall

Fscore = 2*((precision*recall)/(precision+recall))

print("F_score value is =" ,Fscore)

from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

```
Recall value is = 0.09774436090225563
Precision value is = 0.08950086058519793
F_score value is = 0.09344115004492363
```

[ ]:
```
{'rouge-1': {'f': 0.6174496594567812, 'p': 0.575, 'r': 0.6666666666666666}, 'rouge-2': {'f':
  0.3169398857176984,
   'p': 0.30526315789473685,
   'r': 0.32954545454545453},
  'rouge-l': {'f': 0.6174496594567812, 'p': 0.575, 'r': 0.6666666666666666}}
```

[ ]:
```
!pip install tsne
V=word_embeddings['strange']
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: tsne in /usr/local/lib/python3.7/dist-packages (0.3.1)

Requirement already satisfied: cython>=0.19.1 in /usr/local/lib/python3.7/dist-packages
(from tsne) (0.29.30)
Requirement already satisfied: numpy>=1.7.1 in /usr/local/lib/python3.7/dist-packages
(from tsne) (1.21.6)
Requirement already satisfied: scipy>=0.12.0 in /usr/local/lib/python3.7/dist-packages
(from tsne) (1.7.3)
```

[ ]:
```
for i in x:
        plt.scatter(x[i],y[i])
        plt.annotate(labels[i],
                     xy=(x[i], y[i]),
                     xytext=(5, 2),
                     textcoords='offset points',
                     ha='right',
                     va='bottom')
    plt.show()
```

[ ]:
```
!pip install scikit-learn !pip
install matplotlib import
matplotlib
from sklearn.manifold import TSNE
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
ERROR: Could not find a version that satisfies the requirement skikitlearn

(from versions: none)
ERROR: No matching distribution found for skikitlearn
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages
(1.0.2)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages
(from scikit-learn) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn) (1.21.6)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages
(from scikit-learn) (1.1.0)
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
```

```
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages
(3.2.2)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages
(from matplotlib) (0.11.0)
Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages
(from matplotlib) (1.21.6)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-
packages (from matplotlib) (1.4.4) Requirement already satisfied:
pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from
matplotlib) (3.0.9) Requirement already satisfied: python-dateutil>=2.1 in
/usr/local/lib/python3.7/dist-packages (from matplotlib) (2.8.2) Requirement already
satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from
kiwisolver>=1.0.1->matplotlib) (4.1.1)


Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages
(from python-dateutil>=2.1->matplotlib) (1.15.0)
```

[ ]:
```
X = np.
  →array([word_embeddings['strange'],word_embeddings['cold'],word_embeddings['water'],word_embed
Y = np.array([])
for i in tokenised_words:
   Y=np.array(['i'])
model = TSNE(n_components=2, random_state=0)
model.fit_transform(X)
```

```
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:783: FutureWarning: The
default initialization in TSNE will change from 'random' to 'pca' in 1.2.

   FutureWarning,
/usr/local/lib/python3.7/dist-packages/sklearn/manifold/_t_sne.py:793:
FutureWarning: The default learning rate in TSNE will change from 200.0 to
'auto' in 1.2.
   FutureWarning,
```

[ ]:
```
array([[ 269.5624    , -124.8034    ],
       [   1.7829943, -295.07968   ],
       [ 220.81143  , -343.83212   ],
       [  50.533993 ,  -76.050934 ]], dtype=float32)
```

[ ]:
```
print(text)
```

```
Experts predicted latest piece big Chinese space junk fall back Earth sure exactly land .
The entry be around end month probably July 31 02:22 UTC ± 17 hours . The mentioned
rocket roughly 21 tonne part Wentian space station module
.  It was launched Sunday docked successfully Tiangong space station . A Long March 5B
CZ 5B rocket propelled the uncrewed Wentian spacecraft the Wenchang launch centre China
' tropical island Hainan . It appeared be concerning experts claimed they are not sure the
debris will land . Holger Krag the head the Space
```

Safety Program Office the European Space Agency told SpaceNews: “ It always difficult assess the amount surviving mass number fragments without knowing the design the object a reasonable ‘ rule thumb ’ about 20 40 per cent the original dry mass . ”

[ ]:
```
print(tokenised_words)
```

```
['Experts', 'predicted', 'latest', 'piece', 'big', 'Chinese', 'space', 'junk',
'fall', 'back', 'Earth', 'sure', 'exactly', 'land', '.', 'The', 'entry', 'be',
'around', 'end', 'month', 'probably', 'July', '31', '02:22', 'UTC', '±', '17',
'hours', '.', 'The', 'mentioned', 'rocket', 'roughly', '21', 'tonne', 'part',
'Wentian', 'space', 'station', 'module', '.', 'It', 'was', 'launched', 'Sunday',
'docked', 'successfully', 'Tiangong', 'space', 'station', '.', 'A', 'Long',
'March', '5B', 'CZ', '5B', 'rocket', 'propelled', 'the', 'uncrewed', 'Wentian',
'spacecraft', 'the', 'Wenchang', 'launch', 'centre', 'China', '’', 'tropical',
'island', 'Hainan', '.', 'It', 'appeared', 'be', 'concerning', 'experts',
'claimed', 'they', 'are', 'not', 'sure', 'the', 'debris', 'will', 'land', '.',
'Holger', 'Krag', 'the', 'head', 'the', 'Space', 'Safety', 'Program', 'Office',
'the', 'European', 'Space', 'Agency', 'told', 'SpaceNews', ':', '“', 'It',
'always', 'difficult', 'assess', 'the', 'amount', 'surviving', 'mass', 'number',
'fragments', 'without', 'knowing', 'the', 'design', 'the', 'object', 'a',
'reasonable', '‘', 'rule', 'thumb', '’', 'about', '20', '40', 'per', 'cent',
'the', 'original', 'dry', 'mass', '.', '”']
```

[ ]

Text summarization using BERT on falling Chinese rocket

Bert- BERT (Bidirectional Encoder Representations from Transformers) is a recent paper published by researchers at Google AI Language. It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and others.

July 29, 2022

[1]:
```
text = """Experts have predicted that the latest piece of big Chinese space junk␣
→will fall back to Earth but are not sure where exactly it will land. The␣
→re-entry will be around the end of the month, probably on July 31 at 02:22 UTC␣
→± 17 hours. The mentioned rocket is roughly 21-tonne and was part of the␣
→Wentian space station module. It was launched on Sunday and docked␣
→successfully with the Tiangong space station. A Long March 5B (CZ-5B) rocket␣
→propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on␣
→China's tropical island of Hainan. It appeared to be concerning as experts␣
→have claimed that they are not sure where the debris will land. Holger Krag,␣
→the head of the Space Safety Program Office for the European Space Agency,␣
→told SpaceNews: "It is always difficult to assess the amount of surviving mass␣
→and number of fragments without knowing the design of the object, but a␣
→reasonable 'rule-of-thumb' is about 20-40 per cent of the original dry mass.
→"""
```

[2]:
```
from summarizer import Summarizer,TransformerSummarizer
```

[3]:
```
print(text)
```

```
Experts have predicted that the latest piece of big Chinese space junk will fall back to Earth
but are not sure where exactly it will land. The re-entry will be around the end of the month,
probably on July 31 at 02:22 UTC ± 17 hours. The mentioned rocket is roughly 21-tonne and
was part of the Wentian space station module. It was launched on Sunday and docked
successfully with the Tiangong space station. A Long March 5B (CZ-5B) rocket propelled the
uncrewed Wentian spacecraft from the Wenchang launch centre on China's tropical island of
Hainan. It appeared to be concerning as experts have claimed that they are not sure where
the debris will land. Holger Krag, the head of the Space Safety Program Office for the
European Space Agency, told SpaceNews: "It is always difficult to assess the amount of
surviving mass and number of fragments without knowing the design of the object, but a
reasonable 'rule-of-thumb' is about 20-40 per cent of the original dry mass."
```

[4]:
```
# word count comaprison
count=0
    for i in text:
        count+=1
 print(count)
```

972

[7]:
```
bert_model =Summarizer()
bert_summary = ''.join(bert_model(text,min_length=30))
print(bert_summary)
final_summary = bert_summary
```

Some weights of the model checkpoint at bert-large-uncased were not used when initializing BertModel: ['cls.predictions.decoder.weight',
'cls.predictions.transform.LayerNorm.weight',
'cls.predictions.transform.dense.bias',
'cls.predictions.transform.LayerNorm.bias',
'cls.predictions.transform.dense.weight', 'cls.seq_relationship.bias',
'cls.seq_relationship.weight', 'cls.predictions.bias']
- This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be exactly identical (initializing a BertForSequenceClassification model from a BertForSequenceClassification model).

Experts have predicted that the latest piece of big Chinese space junk will fall back to Earth but are not sure where exactly it will land. The mentioned rocket is roughly 21-tonne and was part of the Wentian space station module.

[8]:
```
count=0
for i in bert_summary:
    count+=1
print(count)
```

230

[10]:
```
#analysis part

import nltk

reference_text ="""Experts have predicted that the latest piece of big Chinese␣
→space junk will fall back to Earth but are not sure where exactly it will land.
→The mentioned rocket is roughly 21-tonne and was part of the Wentian space␣
→station module.the head of the Space Safety Program Office for the European␣
→Space Agency, told SpaceNews: “It is always difficult to assess the amount of␣
→surviving mass and number of fragments without knowing the design of the␣
→object, but a reasonable ‘rule-of-thumb’ is about 20-40 per cent of the␣
→original dry mass.”
"""
tokenised_words_1 = nltk.word_tokenize(reference_text)
tokenised_words_2 = nltk.word_tokenize(final_summary)
#tokenised_words_1=set(tokenised_words_1)
#tokenised_words_2=set(tokenised_words_2)
```

```
same_list = list(set(tokenised_words_1).intersection(tokenised_words_2))
overlaped_string = ""
for ele in same_list:
        overlaped_string+= " " + ele

## Now we got the system overlapped string



#to do--->
#Bigrams instead of the words comparing one at a time.
#Visualising the sentences using the embeddings.
#metrics on other models as well.


print(tokenised_words_1)


recall = (len(overlaped_string)/len(reference_text))
print("Recall value is =" ,recall)

precision = (len(overlaped_string)/len(final_summary))
print("precision value is =" ,precision)

## F-score of the given precision and recall

Fscore = 2*((precision*recall)/(precision+recall))

print("F_score value is =" ,Fscore)

from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

```
['Experts', 'have', 'predicted', 'that', 'the', 'latest', 'piece', 'of', 'big',
'Chinese', 'space', 'junk', 'will', 'fall', 'back', 'to', 'Earth', 'but', 'are',
'not', 'sure', 'where', 'exactly', 'it', 'will', 'land.The', 'mentioned',
'rocket', 'is', 'roughly', '21-tonne', 'and', 'was', 'part', 'of', 'the',
'Wentian', 'space', 'station', 'module.the', 'head', 'of', 'the', 'Space',
'Safety', 'Program', 'Office', 'for', 'the', 'European', 'Space', 'Agency', ',',
'told', 'SpaceNews', ':', '“', 'It', 'is', 'always', 'difficult', 'to',
'assess', 'the', 'amount', 'of', 'surviving', 'mass', 'and', 'number', 'of',
'fragments', 'without', 'knowing', 'the', 'design', 'of', 'the', 'object', ',',
'but', 'a', 'reasonable', '”', 'rule-of-thumb', '”', 'is', 'about', '20-40',
'per', 'cent', 'of', 'the', 'original', 'dry', 'mass', '.', '”']
Recall value is = 0.36891385767790263
precision value is = 0.8565217391304348
F_score value is = 0.5157068062827225
```

[10]: {'rouge-1': {'r': 0.5285714285714286, 'p': 1.0, 'f': 0.6915887805223164},
'rouge-2': {'r': 0.449438202247191, 'p': 1.0, 'f': 0.620155034481101},
'rouge-l': {'r': 0.5285714285714286, 'p': 1.0, 'f': 0.6915887805223164}}

[11]:
```
import nltk
from nltk.tokenize import word_tokenize

bigram_1 =list(nltk.bigrams(reference_text.split()))                    ###bigram of reference␣
  →text
bigram_2 =list(nltk.bigrams(final_summary.split())) ###bigram of final summary same_list =
list(set(bigram_1).intersection(bigram_2))

recall = ((2*len(same_list))/len(reference_text))
print("Recall value is =" ,recall)

precision = ((2*len(same_list))/len(final_summary))
print("Precision value is =" ,precision)

## F-score of the given precision and recall Fscore =
2*((precision*recall)/(precision+recall)) print("F_score
value is =" ,Fscore)
from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

Recall value is = 0.1348314606741573
Precision value is = 0.3130434782608696
F_score value is = 0.18848167539267016

[11]: {'rouge-1': {'r': 0.5285714285714286, 'p': 1.0, 'f': 0.6915887805223164},
'rouge-2': {'r': 0.449438202247191, 'p': 1.0, 'f': 0.620155034481101},
'rouge-l': {'r': 0.5285714285714286, 'p': 1.0, 'f': 0.6915887805223164}}

[ ]:

Text summarization using GPT on the falling Chinese rocket news and metric evaluation

GPT is **a neural network machine learning model trained using internet data to generate any type of text**. Developed by OpenAI, it requires a small amount of input text to generate large volumes of relevant and sophisticated machine-generated text.

July 29, 2022

[5]:
```
from summarizer import Summarizer,TransformerSummarizer
```

[3]:
```
text = """Experts have predicted that the latest piece of big Chinese space junk␣
→will fall back to Earth but are not sure where exactly it will land. The␣
→re-entry will be around the end of the month, probably on July 31 at 02:22 UTC␣
→± 17 hours. The mentioned rocket is roughly 21-tonne and was part of the␣
→Wentian space station module. It was launched on Sunday and docked␣
→successfully with the Tiangong space station. A Long March 5B (CZ-5B) rocket␣
→propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on␣
→China's tropical island of Hainan. It appeared to be concerning as experts␣
→have claimed that they are not sure where the debris will land. Holger Krag,␣
→the head of the Space Safety Program Office for the European Space Agency,␣
→told SpaceNews: "It is always difficult to assess the amount of surviving mass␣
→and number of fragments without knowing the design of the object, but a␣
→reasonable 'rule-of-thumb' is about 20-40 per cent of the original dry mass.
→"""
```

[6]:
```
print(text)
```

```
Experts have predicted that the latest piece of big Chinese space junk will fall back to Earth but are not sure where exactly it will land. The re-entry will be around the end of the month, probably on July 31 at 02:22 UTC ± 17 hours. The mentioned rocket is roughly 21-tonne and was part of the Wentian space station module. It was launched on Sunday and docked successfully with the Tiangong space station. A Long March 5B (CZ-5B) rocket propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on China's tropical island of Hainan. It appeared to be concerning as experts have claimed that they are not sure where the debris will land. Holger Krag, the head of the Space Safety Program Office for the European Space Agency, told SpaceNews: "It is always difficult to assess the amount of surviving mass and number of fragments without knowing the design of the object, but a reasonable 'rule-of-thumb' is about 20-40 per cent of the original dry mass."
```

[7]:
```
# word count comaprison
count=0
for i in text:
```

```
        count+=1
print(count)
```

```
972
```

[12]:
```
GPT2_model =␣
  →TransformerSummarizer(transformer_type="GPT2",transformer_model_key="gpt2-medium")
full = ''.join(GPT2_model(text, min_length=20))
print(full)
final_summary = full
```

```
Experts have predicted that the latest piece of big Chinese space junk will fall back to Earth
but are not sure where exactly it will land. It was launched on Sunday and docked
successfully with the Tiangong space station.
```

[13]:
```
count=0
    for i in full:
        count+=1
 print(count)
```

```
222
```

[15]:
```
import nltk
reference_text ="""Experts have predicted that the latest piece of big Chinese␣
  →space junk will fall back to Earth but are not sure where exactly it will land.
  →The mentioned rocket is roughly 21-tonne and was part of the Wentian space␣
  →station module.the head of the Space Safety Program Office for the European␣
  →Space Agency, told SpaceNews: “It is always difficult to assess the amount of␣
  →surviving mass and number of fragments without knowing the design of the␣
  →object, but a reasonable ‘rule-of-thumb’ is about 20-40 per cent of the␣
  →original dry mass.”
"""
tokenised_words_1 = nltk.word_tokenize(reference_text)
tokenised_words_2 = nltk.word_tokenize(final_summary)
#tokenised_words_1=set(tokenised_words_1)
#tokenised_words_2=set(tokenised_words_2)
same_list = list(set(tokenised_words_1).intersection(tokenised_words_2))
overlaped_string = ""
for ele in same_list:
        overlaped_string+= " " + ele

## Now we got the system overlapped string

#to do--->
#Bigrams instead of the words comparing one at a time.
#Visualising the sentences using the embeddings.
#metrics on other models as well.
```

```
print(tokenised_words_1)
recall = (len(overlaped_string)/len(reference_text))
print("Recall value is =" ,recall)
precision = (len(overlaped_string)/len(final_summary))
print("precision value is =" ,precision)

## F-score of the given precision and recall

Fscore = 2*((precision*recall)/(precision+recall))

print("F_score value is =" ,Fscore)

from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

```
['Experts', 'have', 'predicted', 'that', 'the', 'latest', 'piece', 'of', 'big',
'Chinese', 'space', 'junk', 'will', 'fall', 'back', 'to', 'Earth', 'but', 'are',
'not', 'sure', 'where', 'exactly', 'it', 'will', 'land.The', 'mentioned',
'rocket', 'is', 'roughly', '21-tonne', 'and', 'was', 'part', 'of', 'the',
'Wentian', 'space', 'station', 'module.the', 'head', 'of', 'the', 'Space',
'Safety', 'Program', 'Office', 'for', 'the', 'European', 'Space', 'Agency', ',',
'told', 'SpaceNews', ':', '“', 'It', 'is', 'always', 'difficult', 'to',
'assess', 'the', 'amount', 'of', 'surviving', 'mass', 'and', 'number', 'of',
'fragments', 'without', 'knowing', 'the', 'design', 'of', 'the', 'object', ',',
'but', 'a', 'reasonable', '“', 'rule-of-thumb', '”', 'is', 'about', '20-40',
'per', 'cent', 'of', 'the', 'original', 'dry', 'mass', '.', '”']
Recall value is = 0.2808988764044944
precision value is = 0.6756756756756757
F_score value is = 0.39682539682539686
```

[15]:
```
{'rouge-1': {'r': 0.4, 'p': 0.7777777777777778, 'f': 0.5283018823068708},
 'rouge-2': {'r': 0.29213483146067415,
  'p': 0.6842105263157895,
  'f': 0.4094488147039494 5},
 'rouge-l': {'r': 0.4, 'p': 0.7777777777777778, 'f': 0.5283018823068708}}
```

[16]:
```
import nltk
from nltk.tokenize import word_tokenize


bigram_1 =list(nltk.bigrams(reference_text.split()))                    ###bigram of reference␣
  →text
bigram_2 =list(nltk.bigrams(final_summary.split())) ###bigram of final summary same_list =
list(set(bigram_1).intersection(bigram_2))



recall = ((2*len(same_list))/len(reference_text))
```

```
print("Recall value is =" ,recall)

precision = ((2*len(same_list))/len(final_summary))
print("Precision value is =" ,precision)

## F-score of the given precision and recall

Fscore = 2*((precision*recall)/(precision+recall))

print("F_score value is =" ,Fscore)

from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

```
Recall value is = 0.0898876404494382
Precision value is = 0.21621621621621623
F_score value is = 0.12698412698412698
```

[16]:
```
{'rouge-1': {'r': 0.4, 'p': 0.7777777777777778, 'f': 0.5283018823068708},
 'rouge-2': {'r': 0.29213483146067415,
  'p': 0.6842105263157895,
  'f': 0.40944881470394945},
 'rouge-l': {'r': 0.4, 'p': 0.7777777777777778, 'f': 0.5283018823068708}}
```

[ ]:

Text summarization using pegasus on the falling chinese rocket

Pegasus' pretraining task is intentionally similar to summarization: **important sentences are removed/masked from an input document and are generated together as one output sequence from the remaining sentences, similar to an extractive summary**.

July 29, 2022

[ ]:
```
!pip3 install torch==1.8.2+cu102 torchvision==0.9.2+cu102 torchaudio==0.8.2 -f␣
↪https://download.pytorch.org/whl/lts/1.8/torch_lts.html
!pip3 install transformers
!pip3 install sentencepiece
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Looking in links: https://download.pytorch.org/whl/lts/1.8/torch_lts.html Collecting
torch==1.8.2+cu102
  Downloading https://download.pytorch.org/whl/lts/1.8/cu102/torch-1.8.2%2Bcu102 -cp37-
cp37m-linux_x86_64.whl (804.1 MB)
     ||  804.1 MB 7.3 kB/s Collecting
torchvision==0.9.2+cu102
  Downloading https://download.pytorch.org/whl/lts/1.8/cu102/torchvision-0.9.2%2 Bcu102-
cp37-cp37m-linux_x86_64.whl (17.3 MB)
     ||  17.3 MB 249 kB/s
Collecting torchaudio==0.8.2
  Downloading https://download.pytorch.org/whl/lts/1.8/torchaudio-0.8.2-cp37-cp3
7m-linux_x86_64.whl (1.9 MB)
     || 1.9 MB 3.2 MB/s
```

```
    Found existing installation: torchaudio 0.12.0+cu113
    Uninstalling torchaudio-0.12.0+cu113:
      Successfully uninstalled torchaudio-0.12.0+cu113
ERROR: pip's dependency resolver does not currently take into account all
the packages that are installed. This behaviour is the source of the following
dependency conflicts.
torchtext 0.13.0 requires torch==1.12.0, but you have torch 1.8.2+cu102 which is
incompatible.
Successfully installed torch-1.8.2+cu102 torchaudio-0.8.2
torchvision-0.9.2+cu102
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting transformers
  Downloading transformers-4.21.0-py3-none-any.whl (4.7 MB)
     || 4.7 MB 5.1 MB/s
Collecting huggingface-hub<1.0,>=0.1.0
  Downloading huggingface_hub-0.8.1-py3-none-any.whl (101 kB)
     || 101 kB 10.9 MB/s
Collecting tokenizers!=0.11.3,<0.13,>=0.11.1
  Downloading
tokenizers-0.12.1-cp37-cp37m-manylinux_2_12_x86_64.manylinux2010_x86_64.whl (6.6
MB)
     || 6.6 MB 46.9 MB/s
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.7/dist-
packages (from transformers) (4.64.0)
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/dist-packages
(from transformers) (21.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.7/dist-
packages (from transformers) (3.7.1)
Collecting pyyaml>=5.1
  Downloading PyYAML-6.0-cp37-cp37m-manylinux_2_5_x86_64.manylinux1_x86_64.manyl
inux_2_12_x86_64.manylinux2010_x86_64.whl (596 kB)
     || 596 kB 56.1 MB/s
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.7/dist-
packages (from transformers) (1.21.6) Requirement already satisfied: importlib-
metadata in /usr/local/lib/python3.7/dist-packages (from transformers) (4.12.0)
Requirement already satisfied: requests in /usr/local/lib/python3.7/dist-packages
(from transformers) (2.23.0)

Requirement already satisfied: regex!=2019.12.17 in
/usr/local/lib/python3.7/dist-packages (from transformers) (2022.6.2)
Requirement already satisfied: typing-extensions>=3.7.4.3 in
/usr/local/lib/python3.7/dist-packages (from huggingface-hub<1.0,>=0.1.0-
>transformers) (4.1.1)
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in
```

```
/usr/local/lib/python3.7/dist-packages (from packaging>=20.0->transformers)
(3.0.9)
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-packages
(from importlib-metadata->transformers) (3.8.1)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dist-packages
(from requests->transformers) (2.10)
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (1.24.3) Requirement
already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.7/dist-packages (from requests-
>transformers) (2022.6.15) Requirement already satisfied: chardet<4,>=3.0.2 in
/usr/local/lib/python3.7/dist-packages (from requests->transformers) (3.0.4) Installing
collected packages: pyyaml, tokenizers, huggingface-hub, transformers

  Attempting uninstall: pyyaml
    Found existing installation: PyYAML 3.13
    Uninstalling PyYAML-3.13:
      Successfully uninstalled PyYAML-3.13
Successfully installed huggingface-hub-0.8.1 pyyaml-6.0 tokenizers-0.12.1
transformers-4.21.0
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting sentencepiece
  Downloading
sentencepiece-0.1.96-cp37-cp37m-manylinux_2_17_x86_64.manylinux2014_x86_64.whl
(1.2 MB)
     || 1.2 MB 5.1 MB/s
Installing collected packages: sentencepiece
Successfully installed sentencepiece-0.1.96
```

[ ]:
```
from transformers import PegasusForConditionalGeneration , PegasusTokenizer
```

[ ]:
```
model_tokeniser = PegasusTokenizer.from_pretrained("google/pegasus-xsum")
```

```
Downloading spiece.model:       0%|              | 0.00/1.82M [00:00<?, ?B/s]
Downloading special_tokens_map.json:       0%|              | 0.00/65.0 [00:00<?, ?B/s]
Downloading tokenizer_config.json:       0%|              | 0.00/87.0 [00:00<?, ?B/s]
Downloading config.json:       0%|              | 0.00/1.36k [00:00<?, ?B/s]
```

[ ]:
```
#we have to import model of the pegasus

pegasus_model= PegasusForConditionalGeneration.from_pretrained("google/
→pegasus-xsum")
```

```
Downloading pytorch_model.bin:       0%|              | 0.00/2.12G [00:00<?, ?B/s]
```

[ ]:

```
text = "Experts have predicted that the latest piece of big Chinese space junk␣
  →will fall back to Earth but are not sure where exactly it will land. The␣
  →re-entry will be around the end of the month, probably on July 31 at 02:22 UTC␣
  →± 17 hours. The mentioned rocket is roughly 21-tonne and was part of the␣
  →Wentian space station module. It was launched on Sunday and docked␣
  →successfully with the Tiangong space station. A Long March 5B (CZ-5B) rocket␣
  →propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on␣
  →China's tropical island of Hainan. It appeared to be concerning as experts␣
  →have claimed that they are not sure where the debris will land. Holger Krag,␣
  →the head of the Space Safety Program Office for the European Space Agency,␣
  →told SpaceNews: "It is always difficult to assess the amount of surviving mass␣
  →and number of fragments without knowing the design of the object, but a␣
  →reasonable 'rule-of-thumb' is about 20-40 per cent of the original dry mass.""
```

[ ]:
```
#Role of the tokeniser comes into the play

tokenised_words = model_tokeniser(text, truncation=True, padding="longest",␣
  →return_tensors="pt")
```

[ ]:
```
tokenised_words
```

```
[ ]: {'input_ids': tensor([[11989,   133,  9198,   120,   109,   807,  1038,   113,
461,  1950,
           501,  8033,   138,  1251,   247,   112,  2774,   155,   127,   146,
           334,   241,  1270,   126,   138,  1241,   107,   139,   920,   121,
         20695,   138,   129,   279,   109,   370,   113,   109,   625,   108,
           864,   124,  1307,  3184,   134, 89507,   522, 20226,   110,   105,
          1689,   539,   107,   139,  2137, 10763,   117,  5864, 23653, 75374,
           111,   140,   297,   113,   109, 35587,  3262,   501,  2088,  4352,
           107,   168,   140,  2365,   124,  1342,   111, 43978,  2640,   122,
           109, 30301, 48624,   501,  2088,   107,   202,  2859,  1051,   371,
           788,   143,   529,  2680,  7119,   788,   158, 10763, 30832,   109,
          1596, 41791,   316, 35587,  3262, 18328,   135,   109, 28115, 43722,
          2170,  2060,   124,  1224,   123,   116,  6253,  2273,   113, 85155,
           107,   168,  2893,   112,   129,  3990,   130,  1497,   133,  4620,
           120,   157,   127,   146,   334,   241,   109,  7481,   138,  1241,
           107, 19622,  6867, 17414,   838,   108,   109,   693,   113,   109,
          4108,  4214,  2173,  1584,   118,   109,  1489,  4108,  4260,   108,
           898,  4108, 12737,   151,   185,   362,   117,   329,  1011,   112,
          4676,   109,   713,   113, 12558,  2977,   111,   344,   113, 19320,
           347,  2753,   109,   354,   113,   109,  2951,   108,   155,   114,
          3048,   402, 27841,   121,  1313,   121, 52074,   123,   117,   160,
           599, 13837,   446,  2978,   113,   109,   856,  1514,  2977,  1181,
                                                1]]), 'attention_mask': tensor([[1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
             1,  1, 1,  1, 1,  1,  1, 1,    1, 1, 1,  1, 1,  1, 1,    1, 1, 1,  1,  1, 1,  1, 1, 1,
             1,  1, 1,  1, 1,  1,  1, 1,    1, 1, 1,  1, 1,  1, 1,    1, 1, 1,  1,  1, 1,  1, 1, 1,
```

```
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1]])}
```

[ ]:
```
summary = pegasus_model.generate(**tokenised_words)
```

```
/usr/local/lib/python3.7/dist-packages/transformers/generation_utils.py:1207:
UserWarning: Neither `max_length` nor `max_new_tokens` have been set,
`max_length` will default to 64 (`self.config.max_length`). Controlling
`max_length` via the config is deprecated and `max_length` will be removed from
the config in v5 of Transformers -- we recommend using `max_new_tokens` to
control the maximum length of the generation.
  UserWarning,
```

[ ]:
```
print(summary)
```

```
tensor([[     0,   202,   1950, 10763,   4069,    114,    501,   2088,  4352,   138,
           920,   121, 22511,    109,   2774,    123,    116,   2918,   678,   136,
           625,   107,      1]])
```

[ ]:
```
final_summary = model_tokeniser.decode(summary[0])
```

[ ]:
```
print(final_summary)
```

```
A Chinese rocket carrying a space station module will re-enter the Earth’s
atmosphere later this month.
```

[ ]:
```
import nltk
from nltk.translate.bleu_score import sentence_bleu

list_1 = final_summary.split( )
candidate_list = list_1

reference_text= """Holders of stocks, on the other hand, have the possibility␣
 →that a loss of the dollar’s purchasing power may be offset by advances in␣
 →their dividends and the prices of their shares. We have heard of charitable␣
 →institutions being advised that their portfolios should consist 100% of stocks␣
 →and zero percent of bonds.* This is quite a reversal from the earlier days␣
 →when trust investments were restricted by law to high-grade bonds"""

reference_list = [reference_text.split( )]

score_1=sentence_bleu(reference_list,candidate_list,weights=(0,1,0,0))
print(score_1)
```

```
6.71914084483213e-310
```

```
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:552:
UserWarning:
The hypothesis contains 0 counts of 2-gram overlaps. Therefore the
BLEU score evaluates to 0, independently of how many N-gram
overlaps of lower order it contains. Consider using lower n-gram
order or use SmoothingFunction()
  warnings.warn(_msg)
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:552:
UserWarning:
The hypothesis contains 0 counts of 3-gram overlaps. Therefore the
BLEU score evaluates to 0, independently of how many N-gram
overlaps of lower order it contains. Consider using lower n-gram
order or use SmoothingFunction()
  warnings.warn(_msg)
/usr/local/lib/python3.7/dist-packages/nltk/translate/bleu_score.py:552:
UserWarning:
The hypothesis contains 0 counts of 4-gram overlaps. Therefore the
BLEU score evaluates to 0, independently of how many N-gram
overlaps of lower order it contains. Consider using lower n-gram
order or use SmoothingFunction()
  warnings.warn(_msg)
```

[ ]:
```
!pip install rouge from
rouge import Rouge
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting rouge
  Downloading rouge-1.0.1-py3-none-any.whl (13 kB)
Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from rouge)
(1.15.0)
Installing collected packages: rouge
Successfully installed rouge-1.0.1
```

[ ]:
```
reference_text = "Experts have predicted that the latest piece of big Chinese␣
→space junk will fall back to Earth but are not sure where exactly it will land.
→The mentioned rocket is roughly 21-tonne and was part of the Wentian space␣
→station module.the head of the Space Safety Program Office for the European␣
→Space Agency, told SpaceNews: “It is always difficult to assess the amount of␣
→surviving mass and number of fragments without knowing the design of the␣
→object, but a reasonable ‘rule-of-thumb’ is about 20-40 per cent of the␣
→original dry mass.” "

import nltk
nltk.download('punkt')
tokenised_words_1 = nltk.word_tokenize(reference_text)
tokenised_words_2 = nltk.word_tokenize(final_summary)
#tokenised_words_1=set(tokenised_words_1)
```

```
#tokenised_words_2=set(tokenised_words_2)
same_list = list(set(tokenised_words_1).intersection(tokenised_words_2))
overlaped_string = ""
for ele in same_list:
        overlaped_string+= " " + ele

## Now we got the system overlapped string

recall = (len(overlaped_string)/len(reference_text))
print("Recall value is =" ,recall)

precision = (len(overlaped_string)/len(final_summary))
print("precision value is =" ,precision)

## F-score of the given precision and recall taking harmonic mean

Fscore = 2*((precision*recall)/(precision+recall))

print("F_score value is =" ,Fscore)

!pip install rouge
from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]       Package punkt is already up-to-date!

Recall value is = 0.09380863039399624
precision value is = 0.4854368932038835
F_score value is = 0.15723270440251572
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: rouge in /usr/local/lib/python3.7/dist-packages (1.0.1)

Requirement already satisfied: six in /usr/local/lib/python3.7/dist-packages (from rouge)
(1.15.0)
```

[ ]:
```
{'rouge-1': {'f': 0.18604650859924285, 'p': 0.5, 'r': 0.11428571428571428}, 'rouge-2': {'f':
  0.03846153599297353,
   'p': 0.13333333333333333,
   'r': 0.02247191011235955},
 'rouge-l': {'f': 0.18604650859924285, 'p': 0.5, 'r': 0.11428571428571428}}
```

[ ]:
```
import nltk
from nltk.tokenize import word_tokenize
```

```
bigram_1 =list(nltk.bigrams(reference_text.split()))                    ###bigram of reference_
  →text
bigram_2 =list(nltk.bigrams(final_summary.split())) ###bigram of final summary same_list =
list(set(bigram_1).intersection(bigram_2))


recall = ((2*len(same_list))/len(reference_text))
print("Recall value is =" ,recall)

precision = ((2*len(same_list))/len(final_summary))
print("Precision value is =" ,precision)

## F-score of the given precision and recall

Fscore = 2*((precision*recall)/(precision+recall))

print("F_score value is =" ,Fscore)

from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

```
Recall value is = 0.00375234521575985
Precision value is = 0.019417475728155338
F_score value is = 0.006289308176100628
```

[ ]:
```
{'rouge-1': {'f': 0.18604650859924285, 'p': 0.5, 'r': 0.11428571428571428}, 'rouge-2': {'f':
  0.03846153599297353,
   'p': 0.13333333333333333,
   'r': 0.02247191011235955},
 'rouge-l': {'f': 0.18604650859924285, 'p': 0.5, 'r': 0.11428571428571428}}
```

[ ]:
```
!pip3 install textstat

import textstat
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Collecting textstat
  Downloading textstat-0.7.3-py3-none-any.whl (105 kB)
     [    105 kB 5.0 MB/s
Collecting pyphen
  Downloading pyphen-0.12.0-py3-none-any.whl (2.0 MB)
     [    2.0 MB 32.6 MB/s
Installing collected packages: pyphen, textstat Successfully
installed pyphen-0.12.0 textstat-0.7.3
```

```
[ ]: print(textstat.flesch_reading_ease(text) )
     print(textstat.flesch_reading_ease(final_summary) )

     print("----------------------*------------------")

     print(textstat.automated_readability_index(text) )
     print(textstat.automated_readability_index(final_summary) )
     print("----------------------*------------------")
     print(textstat.linsear_write_formula(text))
     print(textstat.linsear_write_formula(final_summary))
```

```
52.73
57.61
----------------------*------------------
15.7
17.8
----------------------*------------------
14.5
15.0
```

```
[ ]: !pip install skikitlearn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
ERROR: Could not find a version that satisfies the requirement skikitlearn

(from versions: none)
ERROR: No matching distribution found for skikitlearn
```

```
[ ]: !pip install scikit-learn !pip
     install matplotlib import
     matplotlib
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages
(1.0.2)
Requirement already satisfied: numpy>=1.14.6 in /usr/local/lib/python3.7/dist-packages
(from scikit-learn) (1.21.6)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages
(from scikit-learn) (1.7.3)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-
packages (from scikit-learn) (3.1.0) Requirement already satisfied: joblib>=0.11 in
/usr/local/lib/python3.7/dist-packages (from scikit-learn) (1.1.0)

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-
wheels/public/simple/
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages
(3.2.2)
```

```
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (0.11.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (1.4.4) Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (3.0.9) Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (2.8.2) Requirement already satisfied: numpy>=1.11 in /usr/local/lib/python3.7/dist-packages (from matplotlib) (1.21.6)

Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib) (4.1.1)

Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib) (1.15.0)
```

[ ]:
```
from sklearn.manifold import TSNE
```

[ ]:
```

```

Text Summarization using spacy on falling rocket by Chinese and its metric evaluation

July 30, 2022

[1]:
```
## Before moving forward with the text summarization we have to pre process the␣
  →steps

#Text cleaning
##       Sentence tokenisation
##       Word frequency
table #Clustering
#Summarization
```

[2]:
```
text ="""Experts have predicted that the latest piece of big Chinese space junk␣
  →will fall back to Earth but are not sure where exactly it will land. The␣
  →re-entry will be around the end of the month, probably on July 31 at 02:22 UTC␣
  →± 17 hours. The mentioned rocket is roughly 21-tonne and was part of the␣
  →Wentian space station module. It was launched on Sunday and docked␣
  →successfully with the Tiangong space station. A Long March 5B (CZ-5B) rocket␣
  →propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on␣
  →China's tropical island of Hainan. It appeared to be concerning as experts␣
  →have claimed that they are not sure where the debris will land. Holger Krag,␣
  →the head of the Space Safety Program Office for the European Space Agency,␣
  →told SpaceNews: "It is always difficult to assess the amount of surviving mass␣
  →and number of fragments without knowing the design of the object, but a␣
  →reasonable 'rule-of-thumb' is about 20-40 per cent of the original dry mass.
  →"""
```

[35]:
```
print(text)
print(len(text))
```

Experts have predicted that the latest piece of big Chinese space junk will fall back to Earth but are not sure where exactly it will land The re entry will be around the end of the month probably on July 31 at 02:22 UTC ± 17 hours The mentioned rocket is roughly 21 tonne and was part of the Wentian space station module It was launched on Sunday and docked successfully with the Tiangong space station A Long March 5B (CZ 5B) rocket propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on China's tropical island of Hainan It appeared to be concerning as experts have claimed that they are not sure where the debris will land Holger Krag the head of the Space Safety Program Office for the European Space Agency told SpaceNews: "It is always difficult to

assess the amount of surviving mass and number of fragments without knowing the design of the object but a reasonable ‘rule of thumb’ is about 20 40 per cent of the original dry mass ”
972

[4]:
```
## Text cleaning

for i in text:
    if (i=="-" or i ==":" or i==";" or i=="'" or i=="-"):
        text=text.replace(i," ")
print(text)
```

Experts have predicted that the latest piece of big Chinese space junk will fall back to Earth but are not sure where exactly it will land The re entry will be around the end of the month probably on July 31 at 02:22 UTC ± 17 hours The mentioned rocket is roughly 21 tonne and was part of the Wentian space station module It was launched on Sunday and docked successfully with the Tiangong space station A Long March 5B (CZ 5B) rocket propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on China’s tropical island of Hainan It appeared to be concerning as experts have claimed that they are not sure where the debris will land Holger Krag the head of the Space Safety Program Office for the European Space Agency told SpaceNews: “It is always difficult to assess the amount of surviving mass and number of fragments without knowing the design of the object but a reasonable ‘rule of thumb’ is about 20 40 per cent of the original dry mass ”

[5]:
```
## Removing stop word

import nltk
from nltk.corpus import stopwords as STOP_WORDS

tokenised_words = nltk.word_tokenize(text)

count =0
for i in tokenised_words:
    if i in STOP_WORDS.words('english'):
        tokenised_words.remove(i)
        count+=1

print(tokenised_words)
```

['Experts', 'predicted', 'latest', 'piece', 'big', 'Chinese', 'space', 'junk',
'fall', 'back', 'Earth', 'sure', 'exactly', 'land', 'The', 'entry', 'be',
'around', 'end', 'month', 'probably', 'July', '31', '02:22', 'UTC', '±', '17',
'hours', 'The', 'mentioned', 'rocket', 'roughly', '21', 'tonne', 'part',
'Wentian', 'space', 'station', 'module', 'It', 'was', 'launched', 'Sunday',
'docked', 'successfully', 'Tiangong', 'space', 'station', 'A', 'Long', 'March',
'5B', '(', 'CZ', '5B', ')', 'rocket', 'propelled', 'the', 'uncrewed', 'Wentian',
'spacecraft', 'the', 'Wenchang', 'launch', 'centre', 'China', "'", 'tropical',

'island', 'Hainan', 'It', 'appeared', 'be', 'concerning', 'experts', 'claimed',
'they', 'are', 'not', 'sure', 'the', 'debris', 'will', 'land', 'Holger', 'Krag',
'the', 'head', 'the', 'Space', 'Safety', 'Program', 'Office', 'the', 'European',
'Space', 'Agency', 'told', 'SpaceNews', ':', '“', 'It', 'always', 'difficult',
'assess', 'the', 'amount', 'surviving', 'mass', 'number', 'fragments',
'without', 'knowing', 'the', 'design', 'the', 'object', 'a', 'reasonable', '”',
'rule', 'thumb', '”', 'about', '20', '40', 'per', 'cent', 'the', 'original',
'dry', 'mass', '”']

[6]:
```
## Word frequency table

count =0
dict1={}

##First we have to assign each of the tokens as the key pair in the dictionary


for i in tokenised_words:
    count+=1
    dict1[i]=0


## Now for creating the value of the word
word_count=0

for i in tokenised_words:
    word_count=0
    for j in tokenised_words:
        if(i == j):
            word_count+=1
            dict1[i]=word_count

print(dict1)
```

{'Experts': 1, 'predicted': 1, 'latest': 1, 'piece': 1, 'big': 1, 'Chinese': 1,
'space': 3, 'junk': 1, 'fall': 1, 'back': 1, 'Earth': 1, 'sure': 2, 'exactly':
1, 'land': 2, 'The': 2, 'entry': 1, 'be': 2, 'around': 1, 'end': 1, 'month': 1,
'probably': 1, 'July': 1, '31': 1, '02:22': 1, 'UTC': 1, '±': 1, '17': 1,
'hours': 1, 'mentioned': 1, 'rocket': 2, 'roughly': 1, '21': 1, 'tonne': 1,
'part': 1, 'Wentian': 2, 'station': 2, 'module': 1, 'It': 3, 'was': 1,
'launched': 1, 'Sunday': 1, 'docked': 1, 'successfully': 1, 'Tiangong': 1, 'A':
1, 'Long': 1, 'March': 1, '5B': 2, '(': 1, 'CZ': 1, ')': 1, 'propelled': 1,
'the': 10, 'uncrewed': 1, 'spacecraft': 1, 'Wenchang': 1, 'launch': 1, 'centre':
1, 'China': 1, '”': 2, 'tropical': 1, 'island': 1, 'Hainan': 1, 'appeared': 1,
'concerning': 1, 'experts': 1, 'claimed': 1, 'they': 1, 'are': 1, 'not': 1,
'debris': 1, 'will': 1, 'Holger': 1, 'Krag': 1, 'head': 1, 'Space': 2, 'Safety':
1, 'Program': 1, 'Office': 1, 'European': 1, 'Agency': 1, 'told': 1,
'SpaceNews': 1, ':': 1, '“': 1, 'always': 1, 'difficult': 1, 'assess': 1,

```
'amount': 1, 'surviving': 1, 'mass': 2, 'number': 1, 'fragments': 1, 'without':
1, 'knowing': 1, 'design': 1, 'object': 1, 'a': 1, 'reasonable': 1, '‘': 1,
'rule': 1, 'thumb': 1, 'about': 1, '20': 1, '40': 1, 'per': 1, 'cent': 1,
'original': 1, 'dry': 1, '”': 1}
```

[7]:
```
### Performing the clustering
import spacy
from string import punctuation
```

[8]:
```
nlp = spacy.load('en_core_web_sm')
```

[9]:
```
doc = nlp(text)
```

[10]:
```
print(type(doc))
```

```
<class 'spacy.tokens.doc.Doc'>
```

[11]:
```
## making the list of the tokens

token= [token.text for token in doc]
print(token)
```

```
['Experts', 'have', 'predicted', 'that', 'the', 'latest', 'piece', 'of', 'big',
'Chinese', 'space', 'junk', 'will', 'fall', 'back', 'to', 'Earth', 'but', 'are',
'not', 'sure', 'where', 'exactly', 'it', 'will', 'land', ' ', 'The', 're',
'entry', 'will', 'be', 'around', 'the', 'end', 'of', 'the', 'month', ' ',
'probably', 'on', 'July', '31', 'at', '\xa0', '02:22', 'UTC', '±', '17',
'hours', '\xa0', 'The', 'mentioned', 'rocket', 'is', 'roughly', '21', 'tonne',
'and', 'was', 'part', 'of', 'the', 'Wentian', 'space', 'station', 'module', ' ',
'It', 'was', 'launched', 'on', 'Sunday', 'and', 'docked', 'successfully',
'with', 'the', 'Tiangong', 'space', 'station', '\xa0', 'A', 'Long', 'March',
'5B', '(', 'CZ', '5B', ')', 'rocket', 'propelled', 'the', 'uncrewed', 'Wentian',
'spacecraft', 'from', 'the', 'Wenchang', 'launch', 'centre', 'on', 'China',
'’s', 'tropical', 'island', 'of', 'Hainan', '\xa0', 'It', 'appeared', 'to',
'be', 'concerning', 'as', 'experts', 'have', 'claimed', 'that', 'they', 'are',
'not', 'sure', 'where', 'the', 'debris', 'will', 'land', ' ', 'Holger', 'Krag',
'  ', 'the', 'head', 'of', 'the', 'Space', 'Safety', 'Program', 'Office', 'for', 'the', 'European', 'Space',
'Agency', ' ', 'told', 'SpaceNews', ':', '‘', 'It', 'is', 'always', 'difficult', 'to', 'assess', 'the', 'amount',
'of', 'surviving', 'mass', 'and', 'number', 'of', 'fragments', 'without', 'knowing', 'the', 'design', 'of',
'the', 'object', ' ', 'but', 'a', 'reasonable', '’', 'rule', 'of', 'thumb', '’', 'is', 'about', '20', '40', 'per',
'cent', 'of', 'the', 'original', 'dry', 'mass', '”']
```

[12]:
```
max_frequency = max(dict1.values())
print(max_frequency)
```

```
10
```

[13]:
```
for i in dict1.keys():
    dict1[i]=dict1[i]/max_frequency

    print(dict1)
```

```
{'Experts': 0.1, 'predicted': 1, 'latest': 1, 'piece': 1, 'big': 1, 'Chinese':
1, 'space': 3, 'junk': 1, 'fall': 1, 'back': 1, 'Earth': 1, 'sure': 2,
'exactly': 1, 'land': 2, 'The': 2, 'entry': 1, 'be': 2, 'around': 1, 'end': 1,
'month': 1, 'probably': 1, 'July': 1, '31': 1, '02:22': 1, 'UTC': 1, '±': 1,
'17': 1, 'hours': 1, 'mentioned': 1, 'rocket': 2, 'roughly': 1, '21': 1,
'tonne': 1, 'part': 1, 'Wentian': 2, 'station': 2, 'module': 1, 'It': 3, 'was':
1, 'launched': 1, 'Sunday': 1, 'docked': 1, 'successfully': 1, 'Tiangong': 1,
'A': 1, 'Long': 1, 'March': 1, '5B': 2, '(': 1, 'CZ': 1, ')': 1, 'propelled': 1,
'the': 10, 'uncrewed': 1, 'spacecraft': 1, 'Wenchang': 1, 'launch': 1, 'centre':
1, 'China': 1, '’': 2, 'tropical': 1, 'island': 1, 'Hainan': 1, 'appeared': 1,
'concerning': 1, 'experts': 1, 'claimed': 1, 'they': 1, 'are': 1, 'not': 1,
'debris': 1, 'will': 1, 'Holger': 1, 'Krag': 1, 'head': 1, 'Space': 2, 'Safety':
1, 'Program': 1, 'Office': 1, 'European': 1, 'Agency': 1, 'told': 1,
'SpaceNews': 1, ':': 1, '“': 1, 'always': 1, 'difficult': 1, 'assess': 1,
'amount': 1, 'surviving': 1, 'mass': 2, 'number': 1, 'fragments': 1, 'without':
1, 'knowing': 1, 'design': 1, 'object': 1, 'a': 1, 'reasonable': 1, '”': 1,
'rule': 1, 'thumb': 1, 'about': 1, '20': 1, '40': 1, 'per': 1, 'cent': 1,
'original': 1, 'dry': 1, '”': 1}
{'Experts': 0.1, 'predicted': 0.1, 'latest': 1, 'piece': 1, 'big': 1, 'Chinese':
1, 'space': 3, 'junk': 1, 'fall': 1, 'back': 1, 'Earth': 1, 'sure': 2,
'exactly': 1, 'land': 2, 'The': 2, 'entry': 1, 'be': 2, 'around': 1, 'end': 1,
'month': 1, 'probably': 1, 'July': 1, '31': 1, '02:22': 1, 'UTC': 1, '±': 1,
'17': 1, 'hours': 1, 'mentioned': 1, 'rocket': 2, 'roughly': 1, '21': 1,
'tonne': 1, 'part': 1, 'Wentian': 2, 'station': 2, 'module': 1, 'It': 3, 'was':
1, 'launched': 1, 'Sunday': 1, 'docked': 1, 'successfully': 1, 'Tiangong': 1,
'A': 1, 'Long': 1, 'March': 1, '5B': 2, '(': 1, 'CZ': 1, ')': 1, 'propelled': 1,
'the': 10, 'uncrewed': 1, 'spacecraft': 1, 'Wenchang': 1, 'launch': 1, 'centre':
1, 'China': 1, '’': 2, 'tropical': 1, 'island': 1, 'Hainan': 1, 'appeared': 1,
'concerning': 1, 'experts': 1, 'claimed': 1, 'they': 1, 'are': 1, 'not': 1,
'debris': 1, 'will': 1, 'Holger': 1, 'Krag': 1, 'head': 1, 'Space': 2, 'Safety':
{'Experts': 0.1, 'predicted': 0.1, 'latest': 0.1, 'piece': 0.1, 'big': 0.1,
'Chinese': 0.1, 'space': 0.3, 'junk': 0.1, 'fall': 0.1, 'back': 0.1, 'Earth':
0.1, 'sure': 0.2, 'exactly': 0.1, 'land': 0.2, 'The': 0.2, 'entry': 0.1, 'be':
0.2, 'around': 0.1, 'end': 0.1, 'month': 0.1, 'probably': 0.1, 'July': 0.1,
```

'31': 0.1, '02:22': 0.1, 'UTC': 0.1, '±': 0.1, '17': 0.1, 'hours': 0.1,
'mentioned': 0.1, 'rocket': 0.2, 'roughly': 0.1, '21': 0.1, 'tonne': 0.1,
'part': 0.1, 'Wentian': 0.2, 'station': 0.2, 'module': 0.1, 'It': 0.3, 'was':
0.1, 'launched': 0.1, 'Sunday': 0.1, 'docked': 0.1, 'successfully': 0.1,
'Tiangong': 0.1, 'A': 0.1, 'Long': 0.1, 'March': 0.1, '5B': 0.2, '(': 0.1, 'CZ':
0.1, ')': 0.1, 'propelled': 0.1, 'the': 1.0, 'uncrewed': 0.1, 'spacecraft': 0.1,
'Wenchang': 0.1, 'launch': 0.1, 'centre': 0.1, 'China': 0.1, '’': 0.2,
'tropical': 0.1, 'island': 0.1, 'Hainan': 0.1, 'appeared': 0.1, 'concerning':
0.1, 'experts': 0.1, 'claimed': 0.1, 'they': 0.1, 'are': 0.1, 'not': 0.1,
'debris': 0.1, 'will': 0.1, 'Holger': 0.1, 'Krag': 0.1, 'head': 0.1, 'Space':
0.2, 'Safety': 0.1, 'Program': 0.1, 'Office': 0.1, 'European': 0.1, 'Agency':
0.1, 'told': 0.1, 'SpaceNews': 0.1, ':': 0.1, '“': 0.1, 'always': 0.1,
'difficult': 0.1, 'assess': 0.1, 'amount': 0.1, 'surviving': 0.1, 'mass': 0.2,
'number': 0.1, 'fragments': 0.1, 'without': 0.1, 'knowing': 0.1, 'design': 0.1,
'object': 0.1, 'a': 0.1, 'reasonable': 0.1, '‘': 0.1, 'rule': 0.1, 'thumb': 0.1,
'about': 0.1, '20': 0.1, '40': 0.1, 'per': 0.1, 'cent': 0.1, 'original': 0.1,
'dry': 0.1, '”': 0.1}

[14]:
```
## Sentence tokensisation

sentence_tokens = [sent for sent in doc.sents] print
(sentence_tokens)
```

[Experts have predicted that the latest piece of big Chinese space junk will fall back to Earth but are not sure where exactly it will land The re entry will be around the end of the month probably on July 31 at 02:22 UTC ± 17 hours The mentioned rocket is roughly 21 tonne and was part of the Wentian space station module It was launched on Sunday and docked successfully with the Tiangong space station A Long March 5B (CZ 5B) rocket propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on China’s tropical island of Hainan It appeared to be concerning as experts have claimed that they are not sure where the debris will land Holger Krag the head of the Space Safety Program Office for the European Space Agency told SpaceNews: “It is always difficult to assess the amount of surviving mass and number of fragments without knowing the design of the object but a reasonable ‘rule of thumb’ is about 20 40 per cent of the original dry mass ”]

[15]:
```
sentence_scores = {}
for sent in sentence_tokens:
    for word in sent:
        if word.text.lower() in dict1.keys():
            if sent not in sentence_scores.keys():
                sentence_scores[sent] = dict1[word.text.lower()]
            else:
                sentence_scores[sent] += dict1[word.text.lower()]
```

[16]:
```
sentence_scores
```

[16]: {Experts have predicted that the latest piece of big Chinese space junk will fall back to Earth but are not sure where exactly it will land The re entry will be around the end of the month probably on July 31 at 02:22 UTC ± 17 hours The mentioned rocket is roughly 21 tonne and was part of the Wentian space station module It was launched on Sunday and docked successfully with the Tiangong space station A Long March 5B (CZ 5B) rocket propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on China's tropical island of Hainan It appeared to be concerning as experts have claimed that they are not sure where the debris will land Holger Krag the head of the Space Safety Program Office for the European Space Agency told SpaceNews: "It is always difficult to assess the amount of surviving mass and number of fragments without knowing the design of the object but a reasonable 'rule of thumb' is about 20 40 per cent of the original dry mass ": 29.300000000000036}

[17]:
```
from heapq import nlargest
```

[18]:
```
select_length = int(len(sentence_tokens)*3)
print(select_length)
```

3

[34]:
```
summary = nlargest(select_length,sentence_scores,key = sentence_scores.get)
final_summary = str(summary[0])
print(final_summary)
print(len(final_summary))
```

Experts have predicted that the latest piece of big Chinese space junk will fall back to Earth but are not sure where exactly it will land The re entry will be around the end of the month probably on July 31 at 02:22 UTC ± 17 hours The mentioned rocket is roughly 21 tonne and was part of the Wentian space station module It was launched on Sunday and docked successfully with the Tiangong space station A Long March 5B (CZ 5B) rocket propelled the uncrewed Wentian spacecraft from the Wenchang launch centre on China's tropical island of Hainan It appeared to be concerning as experts have claimed that they are not sure where the debris will land Holger Krag the head of the Space Safety Program Office for the European Space Agency told SpaceNews: "It is always difficult to assess the amount of surviving mass and number of fragments without knowing the design of the object but a reasonable 'rule of thumb' is about 20 40 per cent of the original dry mass "

972

[32]:

```
reference_text ="""Experts have predicted that the latest piece of big Chinese␣
  →space junk will fall back to Earth but are not sure where exactly it will land.
  →The mentioned rocket is roughly 21-tonne and was part of the Wentian space␣
  →station module.the head of the Space Safety Program Office for the European␣
  →Space Agency, told SpaceNews: “It is always difficult to assess the amount of␣
  →surviving mass and number of fragments without knowing the design of the␣
  →object, but a reasonable ‘rule-of-thumb’ is about 20-40 per cent of the␣
  →original dry mass.”
"""
tokenised_words_1 = nltk.word_tokenize(reference_text)
tokenised_words_2 = nltk.word_tokenize(final_summary)
#tokenised_words_1=set(tokenised_words_1)
#tokenised_words_2=set(tokenised_words_2)
same_list = list(set(tokenised_words_1).intersection(tokenised_words_2))
overlaped_string = ""
for ele in same_list:
        overlaped_string+= " " + ele

## Now we got the system overlapped string



#to do--->
#Bigrams instead of the words comparing one at a time.
#Visualising the sentences using the embeddings.
#metrics on other models as well.


print(tokenised_words_1)



recall = (len(overlaped_string)/len(reference_text))
print("Recall value is =" ,recall)

precision = (len(overlaped_string)/len(final_summary))
print("precision value is =" ,precision)

## F-score of the given precision and recall

Fscore = 2*((precision*recall)/(precision+recall))

print("F_score value is =" ,Fscore)

from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

```
['Experts', 'have', 'predicted', 'that', 'the', 'latest', 'piece', 'of', 'big',
'Chinese', 'space', 'junk', 'will', 'fall', 'back', 'to', 'Earth', 'but', 'are',
'not', 'sure', 'where', 'exactly', 'it', 'will', 'land.The', 'mentioned',
'rocket', 'is', 'roughly', '21-tonne', 'and', 'was', 'part', 'of', 'the',
'Wentian', 'space', 'station', 'module.the', 'head', 'of', 'the', 'Space',
'Safety', 'Program', 'Office', 'for', 'the', 'European', 'Space', 'Agency', ',',
'told', 'SpaceNews', ':', '“', 'It', 'is', 'always', 'difficult', 'to',
'assess', 'the', 'amount', 'of', 'surviving', 'mass', 'and', 'number', 'of',
'fragments', 'without', 'knowing', 'the', 'design', 'of', 'the', 'object', ',',
'but', 'a', 'reasonable', '‘', 'rule-of-thumb', '’', 'is', 'about', '20-40',
'per', 'cent', 'of', 'the', 'original', 'dry', 'mass', '.', '”']
Recall value is = 0.7509363295880149
precision value is = 0.4125514403292181
F_score value is = 0.5325365205843293
```

[32]:
```
{'rouge-1': {'r': 0.9285714285714286,
  'p': 0.5327868852459017,
  'f': 0.6770833287000869},
 'rouge-2': {'r': 0.8764044943820225,
  'p': 0.4727272727272727,
  'f': 0.6141732237940976},
 'rouge-l': {'r': 0.9285714285714286,
  'p': 0.5327868852459017,
  'f': 0.6770833287000869}}
```

[33]:
```
import nltk
from nltk.tokenize import word_tokenize


bigram_1 =list(nltk.bigrams(reference_text.split()))                    ###bigram of reference␣
 →text
bigram_2 =list(nltk.bigrams(final_summary.split())) ###bigram of final summary same_list =
list(set(bigram_1).intersection(bigram_2))


recall = ((2*len(same_list))/len(reference_text))
print("Recall value is =" ,recall)

precision = ((2*len(same_list))/len(final_summary))
print("Precision value is =" ,precision)

## F-score of the given precision and recall

Fscore = 2*((precision*recall)/(precision+recall))

print("F_score value is =" ,Fscore)
```

```
from rouge import Rouge
rouge=Rouge()
rouge.get_scores(final_summary,reference_text,avg=True)
```

```
Recall value is = 0.26591760299625467
Precision value is = 0.14609053497942387
F_score value is = 0.18857901726427623
```

[33]:

```
{'rouge-1': {'r': 0.9285714285714286,
  'p': 0.5327868852459017,
  'f': 0.6770833287000869},
 'rouge-2': {'r': 0.8764044943820225,
  'p': 0.4727272727272727,
  'f': 0.6141732237940976},
 'rouge-l': {'r': 0.9285714285714286,
  'p': 0.5327868852459017,
  'f': 0.6770833287000869}}
```


[

Analysis of the Text summarization models –

For the analysis of the model I have mainly used the basic metrics precision ,recall and f1-score and another metric which is very popular for nlp text summarization analysis which is rouge.

What is rouge –

ROUGE is actually a set of metrics, rather than just one. We will cover the main ones that are most likely to be used, starting with ROUGE-N.

Precision

To avoid this we use the **precision** metric — which is calculated in almost the exact same way, but rather than dividing by the **reference** n-gram count, we divide by the **model** n-gram count.

Recall

The **recall** counts the number of overlapping n-grams found in both the model output and reference — then divides this number by the total number of n-grams in the reference.

ROUGE-N

ROUGE-N measures the number of matching ‘n-grams’ between our model-generated text and a ‘reference’.

An n-gram is simply a grouping of tokens/words. A unigram (1-gram) would consist of a single word.

Scores of the text summarization models –

	Rouge Precision	Rouge Recall	Rouge F1-score	Formula based Precision	Formula based Recall	Formula based F1-score
Glove Model (Unigram)	0.575	0.666	0.671	0.5611	0.611	0.585
Glove Model (Bigram)	0.305	0.329	0.316	0.089	0.097	0.093
Word2vec Model (Unigram)	0.515	0.471	0.49	0.495	0.419	0.454
Word2vec Model (Bigram)	0.225	0.179	0.199	0.066	0.05	0.060
Google's BERT (Unigram)	1.0	0.528	0.691	0.856	0.368	0.515
Google's BERT (Bigram)	1.0	0.528	0.691	0.313	0.134	0.188

	Rouge Precision	Rouge Recall	Rouge F1-score	Formula based Precision	Formula based Recall	Formula based F1-score
GPT -transformer (Unigram)	0.4	0.4	0.528	0.675	0.289	0.393
GPT -transformer (Bigram)	0.684	0.292	0.409	0.216	0.089	0.126
Pegasus model (Unigram)	0.5	0.114	0.186	0.485	0.093	0.157
Pegasus model (Bigram)	0.133	0.022	0.038	0.019	0.0037	0.006
Spacy model (Unigram)	0.532	0.928	0.677	0.412	0.750	0.532
Spacy model (Bigram)	0.472	0.876	0.614	0.146	0.265	0.188

Conclusion –

So concluding with the performance of the models Glove ie. Global vectors is the best model In getting the intended and complete results with which information was expected by the humans.

There is also much potential in the google's BERT model as the above table shows it.