HW 3: Natural Language Processing

Each assignment needs to be completed independently. Never ever copy others' work or let someone copy your solution (even with minor modification, e.g. changing variable names). Anti-Plagiarism software will be used to check all submissions. No last minute extension of due date. Be sure to start working on it ASAP!

Q1: Extract data using regular expression

Suppose you have scraped the text shown below from an online source (https://www.google.com/finance/). Define a `extract` function which:

- takes a piece of text (in the format of shown below) as an input
- uses regular expression to transform the text into a DataFrame with columns: 'Ticker','Name','Article','Media','Time','Price',and 'Change'
- returns the DataFrame

In [799…

```
import pandas as pd
import nltk
from sklearn.metrics import pairwise_distances
import numpy as np
from matplotlib import pyplot as plt
from sklearn.preprocessing import normalize
import re
import spacy
from nltk.collocations import TrigramAssocMeasures, TrigramCollocationFinder,BigramAsso
from heapq import nlargest
from IPython.core.interactiveshell import InteractiveShell
InteractiveShell.ast_node_interactivity = "all"
```

In [800…

```
text = '''QQQ
Invesco QQQ Trust Series 1
Invesco Expands QQQ Innovation Suite to Include Small-Cap ETF
PR Newswire • 4 hours ago
$265.62
1.13%
add_circle_outline
AAPL
Apple Inc
Estimating The Fair Value Of Apple Inc. (NASDAQ:AAPL)
Yahoo Finance • 4 hours ago
$140.41
1.50%
add_circle_outline
TSLA
Tesla Inc
Could This Tesla Stock Unbalanced Iron Condor Return 23%?
Investor's Business Daily • 1 hour ago
$218.30
0.49%
```

```
add_circle_outline
AMZN
Amazon.com, Inc.
The Regulators of Facebook, Google and Amazon Also Invest in the Companies' Stocks
Wall Street Journal • 2 days ago
$110.91
1.76%
add_circle_outline'''
```

In [801…

```
def extract(text):

    result = None
    word = re.split(r"\nadd_circle_outline" ,text)
    data = []
    for i in range(len(word) -  1):
        if i == 0:
            a = re.split("\n| • ",word[i])
            data.append(a)
        else:
            a = re.split("\n| • ",word[i])
            data.append(a[1:])
    result = pd.DataFrame(data, columns= ['Ticker','Name','Article','Media','Time','Pri

    return result
```

In [802…

```
# test your function

extract(text)
```

Out[802…

	Ticker	Name	Article	Media	Time	Price	Change
0	QQQ	Invesco QQQ Trust Series 1	Invesco Expands QQQ Innovation Suite to Includ...	PR Newswire	4 hours ago	$265.62	1.13%
1	AAPL	Apple Inc	Estimating The Fair Value Of Apple Inc. (NASDA...	Yahoo Finance	4 hours ago	$140.41	1.50%
2	TSLA	Tesla Inc	Could This Tesla Stock Unbalanced Iron Condor ...	Investor's Business Daily	1 hour ago	$218.30	0.49%
3	AMZN	Amazon.com, Inc.	The Regulators of Facebook, Google and Amazon ...	Wall Street Journal	2 days ago	$110.91	1.76%

Q2: Analyze a document

When you have a long document, you would like to

- Quanitfy how `concrete` a sentence is
- Create a concise summary while preserving it's key information content and overall meaning. Let's implement an `extractive method` based on the concept of TF-IDF. The idea is to identify the key sentences from an article and use them as a summary.

Carefully follow the following steps to achieve these two targets.

Q2.1. Preprocess the input document

Define a function `proprocess(doc, lemmatized = True, remove_stopword = True, lower_case = True, remove_punctuation = True, pos_tag = False)`

- Four input parameters:
  - `doc` : an input string (e.g. a document)
  - `lemmatized` : an optional boolean parameter to indicate if tokens are lemmatized. The default value is True (i.e. tokens are lemmatized).
  - `remove_stopword` : an optional boolean parameter to remove stop words. The default value is True, i.e., remove stop words.
  - `remove_punctuation` : optional boolean parameter to remove punctuations. The default values is True, i.e., remove all punctuations.
  - `lower_case` : optional boolean parameter to convert all tokens to lower case. The default option is True, i.e., lowercase all tokens.
  - `pos_tag` : optional boolean parameter to add a POS tag for each token. The default option is False, i.e., no POS tagging.

- Split the input `doc` into sentences. Hint, typically, `\n\n+` is used to separate paragraphs. Make sure a sentence does not cross over two paragraphs. You can replace `\n\n+` by a `.`

- Tokenize each sentence into unigram tokens and also process the tokens as follows:
  - If `lemmatized` is True, lemmatize all unigrams.
  - If `remove_stopword` is set to True, remove all stop words.
  - If `remove_punctuation` is set to True, remove all punctuations.
  - If `lower_case` is set to True, convert all tokens to lower case
  - If `pos_tag` is set to True, find the POS tag for each token and form a tuple for each token, e.g., ('recently', 'ADV'). Either Penn tags or Universal tags are fine. See mapping of these two tagging systems here: https://universaldependencies.org/tagset-conversion/en-penn-uposf.html

- Return the original sentence list (`sents`) and also the tokenized (or tagged) sentence list (`tokenized_sents`).

(Hint: you can use nltk and spacy package for this task.)

In [803…

```
nlp = spacy.load("en_core_web_sm")

def preprocess(doc, lemmatized=True, pos_tag = False, remove_stopword=True, lower_case

    sents, tokenized_sents = [], []

    text_1 = text.replace("\n\n", ". ")

    doc = nlp(text_1)


    for sent in doc.sents:
        sents.append(sent)
```

```
    if pos_tag:
        for i in range(len(sents)):
            temp = []
            for j in sents[i]:
                temp.append((j,j.pos_))

            tokenized_sents.append(temp)
    else:
        if lower_case:
            for idx in range(len(sents)):
                temp = []
                for i in sents[idx]:
                    if remove_stopword and remove_punctuation:
                        if i.is_punct == False and i.is_stop == False:
                            if lemmatized:
                                temp.append(str(i.lemma_.lower()))
                            else:
                                temp.append(str(i))
                    elif remove_stopword == False and remove_punctuation == False:
                        if lemmatized:
                            temp.append(str(i.lemma_.lower()))
                        else:
                            temp.append(str(i))

                    elif remove_stopword == True and remove_punctuation == False:
                        if i.is_stop == False:
                            if lemmatized:
                                temp.append(str(i.lemma_.lower()))
                            else:
                                temp.append(str(i))
                tokenized_sents.append(temp)


    return sents, tokenized_sents
```

In [804…

```
# load test document

text = open("power_of_nlp.txt", "r", encoding='utf-8').read()
```

In [805…

```
# test with all default options:

sents, tokenized_sents = preprocess(text)

# print first 3 sentences
for i in range(3):
    print(sents[i], "\n",tokenized_sents[i],"\n\n" )
```

```
The Power of Natural Language Processing.
 ['power', 'natural', 'language', 'processing']


Until recently, the conventional wisdom was that while AI was better than humans at data
-driven decision making tasks, it was still inferior to humans for cognitive and creativ
e ones.
 ['recently', 'conventional', 'wisdom', 'ai', 'well', 'human', 'data', 'drive', 'decisio
n', 'make', 'task', 'inferior', 'human', 'cognitive', 'creative', 'one']
```

```
But in the past two years language-based AI has advanced by leaps and bounds, changing c
ommon notions of what this technology can do.. .
 ['past', 'year', 'language', 'base', 'ai', 'advance', 'leap', 'bound', 'change', 'commo
n', 'notion', 'technology']
```

In [806…

```
# process text without remove stopwords, punctuation, lowercase, but with pos tagging

sents, tokenized_sents = preprocess(text, lemmatized = False, pos_tag = True,
                                    remove_stopword=False, remove_punctuation = False,
                                    lower_case = False)

for i in range(3):
    print(sents[i], "\n",tokenized_sents[i],"\n\n" )
```

```
The Power of Natural Language Processing.
 [(The, 'DET'), (Power, 'PROPN'), (of, 'ADP'), (Natural, 'PROPN'), (Language, 'PROPN'),
(Processing, 'PROPN'), (., 'PUNCT')]


Until recently, the conventional wisdom was that while AI was better than humans at data
-driven decision making tasks, it was still inferior to humans for cognitive and creativ
e ones.
 [(Until, 'ADP'), (recently, 'ADV'), (,, 'PUNCT'), (the, 'DET'), (conventional, 'ADJ'),
(wisdom, 'NOUN'), (was, 'AUX'), (that, 'SCONJ'), (while, 'SCONJ'), (AI, 'PROPN'), (was,
'AUX'), (better, 'ADJ'), (than, 'ADP'), (humans, 'NOUN'), (at, 'ADP'), (data, 'NOUN'),
(-, 'PUNCT'), (driven, 'VERB'), (decision, 'NOUN'), (making, 'VERB'), (tasks, 'NOUN'),
(,, 'PUNCT'), (it, 'PRON'), (was, 'AUX'), (still, 'ADV'), (inferior, 'ADJ'), (to, 'AD
P'), (humans, 'NOUN'), (for, 'ADP'), (cognitive, 'ADJ'), (and, 'CCONJ'), (creative, 'AD
J'), (ones, 'NOUN'), (., 'PUNCT')]


But in the past two years language-based AI has advanced by leaps and bounds, changing c
ommon notions of what this technology can do.. .
 [(But, 'CCONJ'), (in, 'ADP'), (the, 'DET'), (past, 'ADJ'), (two, 'NUM'), (years, 'NOU
N'), (language, 'NOUN'), (-, 'PUNCT'), (based, 'VERB'), (AI, 'PROPN'), (has, 'AUX'), (ad
vanced, 'VERB'), (by, 'ADP'), (leaps, 'NOUN'), (and, 'CCONJ'), (bounds, 'NOUN'), (,, 'PU
NCT'), (changing, 'VERB'), (common, 'ADJ'), (notions, 'NOUN'), (of, 'ADP'), (what, 'PRO
N'), (this, 'DET'), (technology, 'NOUN'), (can, 'AUX'), (do, 'AUX'), (.., 'PUNCT'), (.,
'PUNCT')]
```

Q2.2. Quantify sentence concreteness

`Concreteness` can increase a message's persuasion. The concreteness can be measured by the use of :

- `article` (e.g., a, an, and the),
- `adpositions` (e.g., in, at, of, on, etc), and
- `quantifiers` , i.e., adjectives before nouns.

Define a function `compute_concreteness(tagged_sent)` as follows:

- Input argument is `tagged_sent` , a list with (token, pos_tag) tuples as shown above.
- Find the three types of tokens: `articles` , `adposition` , and `quantifiers` .
- Compute `concereness` score as: `(the sum of the counts of the three types of tokens)/(total non-punctuation tokens)` .
- return the concreteness score, articles, adposition, and quantifiers lists.

Find the most concrete and the least concrete sentences from the article.

Reference: Peer to Peer Lending: The Relationship Between Language Features, Trustworthiness, and Persuasion Success, https://socialmedialab.sites.stanford.edu/sites/g/files/sbiybj22976/files/media/file/larrimore-jacr-peer-to-peer.pdf

In [807…

```
def compute_concreteness(tagged_sent):


    concreteness,articles, adpositions,quantifier = None, [], [], []

    puct_len = 0

    for i in range(len(tagged_sent)):
        if tagged_sent[i][1] == 'DET':
            articles.append((tagged_sent[i][0], tagged_sent[i][1]))

        if tagged_sent[i][1] == 'ADP' and str(tagged_sent[i][0]) != 'than' : # I met pr
            adpositions.append((tagged_sent[i][0], tagged_sent[i][1]))

        if tagged_sent[i][1] == 'ADJ' and tagged_sent[i+1][1] == 'NOUN' :
            quantifier.append((tagged_sent[i][0], tagged_sent[i][1]))
        if tagged_sent[i][1] == 'PUNCT':
            puct_len = puct_len +1

    concreteness = len(articles + adpositions + quantifier) / (len(tagged_sent) - puct_

    return concreteness, articles, adpositions,quantifier
```

In [808…

```
# tokenize with pos tag, without change the text much

sents, tokenized_sents = preprocess(text, lemmatized = False, pos_tag = True,
                                    remove_stopword=False, remove_punctuation = False,
                                    lower_case = False)
```

In [809…

```
# find concreteness score, articles, adpositions, and quantifiers in a sentence

idx = 1    # sentence id
x = tokenized_sents[idx]
concreteness, articles, adpositions,quantifier = compute_concreteness(x)

# show sentence
sents[idx]
# show result
concreteness, articles, adpositions,quantifier
```

Out[809…
```
Until recently, the conventional wisdom was that while AI was better than humans at data
-driven decision making tasks, it was still inferior to humans for cognitive and creativ
e ones.
```

Out[809…
```
(0.2333333333333334,
 [(the, 'DET')],
 [(Until, 'ADP'), (at, 'ADP'), (to, 'ADP'), (for, 'ADP')],
 [(conventional, 'ADJ'), (creative, 'ADJ')])
```

In [810…
```
# Find the most concrete and the least concrete sentences from the article
concrete = []
for i in range(71):
    concreteness = compute_concreteness(tokenized_sents[i])[0]
    concrete.append(concreteness)
max_id = np.argsort(concrete)[-2] #Fot this question, I get 0.45 concrete on the bigges
min_id = np.argsort(concrete)[0]

print (f"The most concerete sentence:  {sents[max_id]}, {concrete[max_id]:.3f}\n")
print (f"The least concerete sentence:  {sents[min_id]}, {concrete[min_id]:.3f}")
```

```
The most concerete sentence:  Through a combination of your data assets and open dataset
s, train a model for the needs of specific sectors or divisions., 0.429

The least concerete sentence:  What NLP Can Do., 0.000
```

Q2.3. Generate TF-IDF representations for sentences

Define a function `compute_tf_idf(sents, use_idf)` as follows:

- Take the following two inputs:
  - `sents` : tokenized sentences (without pos tagging) returned from Q2.1. These sentences form a corpus for you to calculate `TF-IDF` vectors.
  - `use_idf` : if this option is true, return smoothed normalized `TF_IDF` vectors for all sentences; otherwise, just return normalized `TF` vector for each sentence.
- Calculate `TF-IDF` vectors as shown in the lecture notes (Hint: you can slightly modify code segment 7.5 in NLP Lecture Notes (II) for this task)
- Return the `TF-IDF` vectors if `use_idf` is True. Return the `TF` vectors if `use_idf` is False.

In [811…
```
def compute_tf_idf(sents, use_idf = True, min_df = 1):

    tf_idf = None

    docs_tokens = {i:{token:sents[i].count(token) for token in set(sents[i])} for i in

    dtm=pd.DataFrame.from_dict(docs_tokens, orient="index" )
    dtm=dtm.fillna(0)
    dtm = dtm.sort_index(axis = 0)

    tf=dtm.values
    doc_len=tf.sum(axis=1, keepdims=True)
    tf=np.divide(tf, doc_len)

    df=np.where(tf>0,1,0)
```

```
    smoothed_idf = np.log(np.divide(len(sents)+1, np.sum(df, axis=0)+1))+1

    if use_idf:

        tf_idf= tf*smoothed_idf

    else:
        tf_idf = tf

    return tf_idf
```

In [812…

```
# test compute_tf_idf function

sents, tokenized_sents = preprocess(text)
tf_idf = compute_tf_idf(tokenized_sents, use_idf = True)

# show shape of TF-IDF
tf_idf.shape
```

Out[812… (80, 487)

Q2.4. Identify key sentences as summary

The basic idea is that, in a coherence article, all sentences should center around some key ideas. If we can identify a subset of sentences, denoted as S_{key}, which precisely capture the key ideas, then S_{key} can be used as a summary. Moreover, S_{key} should have high similarity to all the other sentences on average, because all sentences are centered around the key ideas contained in S_{key}. Therefore, we can identify whether a sentence belongs to S_{key} by its similarity to all the other sentences.

Define a function `get_summary(tf_idf, sents, topN = 5)` as follows:

- This function takes three inputs:
  - `tf_idf` : the TF-IDF vectors of all the sentences in a document
  - `sents` : the original sentences corresponding to the TF-IDF vectors
  - `topN` : the top N sentences in the generated summary
- Steps:
  1. Calculate the cosine similarity for every pair of TF-IDF vectors
  2. For each sentence, calculate its average similarity to all the others
  3. Select the sentences with the `topN` largest average similarity
  4. Print the `topN` sentences index
  5. Return these sentences as the summary

In [813…

```
def get_summary(tf_idf, sents, topN = 5):

    summary = None

    from numpy import mean
```

```
    similarity= 1-pairwise_distances(tf_idf, metric = 'cosine')

    mean_similarity= [mean(similarity[i]) for i in range(len(similarity))]

    idx = np.argsort(mean_similarity)[::-1][:topN]

    summary = [sents[i] for i in (idx)]


    return summary
```

In [814…

```
# put everything together and test with different options

sents, tokenized_sents = preprocess(text)
tf_idf = compute_tf_idf(tokenized_sents, use_idf = True)
summary = get_summary(tf_idf, sents, topN = 6)

for sent in summary:
    print(sent,"\n")
```

```
Begin incorporating new language-based AI tools for a variety of tasks to better underst
and their capabilities..

Models like GPT-3 are considered to be foundation models — an emerging AI research area
— which also work for other types of data such as images and video.

Powerful generalizable language-based AI tools like Elicit are here, and they are just t
he tip of the iceberg; multimodal foundation model-based tools are poised to transform b
usiness in ways that are still difficult to predict.

A Language-Based AI Research Assistant.

There is so much text data, and you don’t need advanced models like GPT-3 to extract its
value.

Language models are already reshaping traditional text analytics, but GPT-3 was an espec
ially pivotal language model because, at 10x larger than any previous model upon releas
e, it was the first large language model, which enabled it to perform even more advanced
tasks like programming and solving high school-level math problems.
```

The reason that my answer is different than Test answer :

First, the number of sents are dismatched. I got 80 sentences in this doc, but the test answer is 67 sentence. Becuase the test case use nltk.sent_tokenize(text), but I used bulit-in function nlp(text).sents from Spacy.

Second, I think using nltk.sent_tokenize is not accurate for this document because some of sentences are merged.

For example, the entence below is defined as one sentence, but it is obviously two sentences. (the 15th sentence by using nltk.sent_tokenize(text))

'This transformative capability was already expected to change the nature of how programmers do their jobs, but models continue to improve — the latest from Google’s DeepMind AI lab, for

example, demonstrates the critical thinking and logic skills necessary to outperform most humans in programming competitions.. Models like GPT-3 are considered to be foundation models — an emerging AI research area — which also work for other types of data such as images and video.'

In [815…

```
# Please test summary generated under different configurations

# remove_stopword=False, remove_punctuation = False
sents, tokenized_sents = preprocess(text, lemmatized = True, pos_tag = False,
                                    remove_stopword=False, remove_punctuation = False,
                                    lower_case = True)
tf_idf = compute_tf_idf(tokenized_sents, use_idf = True)
summary = get_summary(tf_idf, sents, topN = 5)
for sent in summary:
    print(sent,"\n")
#lemmatized = False remove_stopword=False, remove_punctuation = False
sents, tokenized_sents = preprocess(text, lemmatized = False, pos_tag = False,
                                    remove_stopword=False, remove_punctuation = False,
                                    lower_case = True)
tf_idf = compute_tf_idf(tokenized_sents, use_idf = True)
summary = get_summary(tf_idf, sents, topN = 5)
for sent in summary:
    print(sent,"\n")
# use_idf = False
sents, tokenized_sents = preprocess(text)
tf_idf = compute_tf_idf(tokenized_sents, use_idf = False)
summary = get_summary(tf_idf, sents, topN = 5)
for sent in summary:
    print(sent,"\n")
```

```
It is difficult to anticipate just how these tools might be used at different levels of
your organization, but the best way to get an understanding of this tech may be for you
and other leaders in your firm to adopt it yourselves.

Powerful generalizable language-based AI tools like Elicit are here, and they are just t
he tip of the iceberg; multimodal foundation model-based tools are poised to transform b
usiness in ways that are still difficult to predict.

This transformative capability was already expected to change the nature of how programm
ers do their jobs, but models continue to improve — the latest from Google’s DeepMind AI
lab, for example, demonstrates the critical thinking and logic skills necessary to outpe
rform most humans in programming competitions..

Because Elicit is an AI research assistant, this is sort of its bread-and-butter, and wh
en I need to start digging into a new research topic, it has become my go-to resource..

And data is critical, but now it is unlabeled data, and the more the better.

It is difficult to anticipate just how these tools might be used at different levels of
your organization, but the best way to get an understanding of this tech may be for you
and other leaders in your firm to adopt it yourselves.

Powerful generalizable language-based AI tools like Elicit are here, and they are just t
he tip of the iceberg; multimodal foundation model-based tools are poised to transform b
usiness in ways that are still difficult to predict.

Because Elicit is an AI research assistant, this is sort of its bread-and-butter, and wh
en I need to start digging into a new research topic, it has become my go-to resource..
```

```
This transformative capability was already expected to change the nature of how programm
ers do their jobs, but models continue to improve — the latest from Google's DeepMind AI
lab, for example, demonstrates the critical thinking and logic skills necessary to outpe
rform most humans in programming competitions..

Elicit is designed for a growing number of specific tasks relevant to research, like sum
marization, data labeling, rephrasing, brainstorming, and literature reviews..

Begin incorporating new language-based AI tools for a variety of tasks to better underst
and their capabilities..

Powerful generalizable language-based AI tools like Elicit are here, and they are just t
he tip of the iceberg; multimodal foundation model-based tools are poised to transform b
usiness in ways that are still difficult to predict.

Models like GPT-3 are considered to be foundation models — an emerging AI research area
— which also work for other types of data such as images and video.

A Language-Based AI Research Assistant.

NLP practitioners call tools like this "language models," and they can be used for simpl
e analytics tasks, such as classifying documents and analyzing the sentiment in blocks o
f text, as well as more advanced tasks, such as answering questions and summarizing repo
rts.
```

Q2.5. Analysis

- Do you think the way to quantify concreteness makes sense? Any other thoughts to measure concreteness or abstractness? Share your ideas in pdf.
- Do you think this method is able to generate a good summary? Any pros or cons have you observed?
- Do these options `lemmatized, remove_stopword, remove_punctuation, use_idf` matter?
- Why do you think these options matter or do not matter?
- If these options matter, what are the best values for these options?

Write your analysis as a pdf file. Be sure to provide some evidence from the output of each step to support your arguments.

Analysis

Q1: I think the way the quantify concreteness makes sense. Concreteness is to ensure the recipient of a message has a clear sense of the sender's intent. If a sentence satisfy in "When" ,"What", "Who", "Why" and "Where", it will be concrete. we have already add the articles, adpositions and quantifier,"When" ,"What", "Who" and "Where" are fixed. So, We might can add word like "Because" and "So" etc to figure out "Why" part.

Q2: I do think this method is not able to generate a good summary. Pros: using tf_idf and cosine similarity to find some good key words and ranking the similarity to find the center idea. Cons: bigrams and trigrams.etc are not considered in this case, the weight of unigram will be unbalanced.

For example, 'language-based AI tools' appeared mutiple times in serval sentences but it counts 4 unigram in one sentences. Since we used TF-IDF and cosine similarity based summary, those kind of mutiple-grams will take the most part of our summary if they have many although this word combnation only have one meaning.

Q3 & Q4: I think remove_stopword, remove_punctuation are matter since most of them do not have any meaning. But use_idf is not matter, it just a normlization. For lemmatized, I think sometimes it will change the meaning of word. For example, 'data-driven' is a adjective, but it became to data and drive after lemmatization, which totally change the meaning.

Q5: I think the best value is remove_stopword because it is a data-cleaning process and it will help user to extract key information to an extend.

Q2.5. (Bonus 3 points).

- Can you think a way to improve this extractive summary method? Explain the method you propose for improvement, implement it, use it to generate a new summary, and demonstrate what is improved in the new summary.
- Or, you can research on some other extractive summary methods and implement one here. Compare it with the one you implemented in Q2.1-Q2.3 and show pros and cons of each method.

In [816…

```
vocab = {}
score = {}
text_without_puncts = []
words = []

for i in nlp(text_1):
    if i.is_punct == False and i.is_stop == False:
        temp = str(i.lower_)
        if len(temp) != 1:
            text_without_puncts.append(temp)
            for i in text_without_puncts:
                vocab[i] = text_without_puncts.count(i)

max_freq = max(vocab.values())

for word in vocab.keys():
    vocab[word] = vocab[word]/max_freq


sents, tokenized_sents = preprocess(text, lemmatized = False, pos_tag = False,
                                    remove_stopword=True, remove_punctuation = True,
                                    lower_case = True)
for i in tokenized_sents:
    for j in i:
        words.append(j)


Trigram_measures = TrigramAssocMeasures()

finder = TrigramCollocationFinder.from_words(words)
```

```
finder.nbest(Trigram_measures.raw_freq, 10)


bigram_measures = BigramAssocMeasures()

finder = BigramCollocationFinder.from_words(words)

finder.nbest(bigram_measures.raw_freq, 10)
#most frequent 4grams word: 'ai language based tools'
#1.0
#0.7241379310344828
#0.4482758620689655
#0.4482758620689655
penalty = 0.724 + 0.448
penalty_1 = 0.724 + 0.448 + 0.448

score = {}
for sent in sents:
    for word in sent:
      if word.text.lower() in vocab.keys():
        if sent  not in score.keys():
            score[sent] = vocab[word.text.lower()]
        else:
            score[sent] += vocab[word.text.lower()]
    score[sent] = score[sent] / len(sent) # normalize by length of sentence
    if 'language-based AI' in sent.text:
        score[sent] = ((score[sent] * len(sent)) - penalty) / (len(sent) - 3)
    elif 'language-based AI tools' in sent.text:
        score[sent] = ((score[sent] * len(sent)) - penalty_1) / (len(sent) - 4)

original = nlargest(8, score, key = score.get)
# sentence less than 10 words are title, we filiter out title.
for i in original:
    if len(i.sent) > 10:
        print(i.sent)
```

Out[816…
```
[('language', 'based', 'AI'),
 ('based', 'AI', 'tools'),
 ('supply', 'chain', 'crisis'),
 ('text', 'data', 'assets'),
 ('AI', 'poised', 'replace'),
 ('Language', 'Based', 'AI'),
 ('artificial', 'general', 'intelligence'),
 ('finance', 'want', 'model'),
 ('general', 'artificial', 'intelligence'),
 ('models', 'like', 'GPT-3')]
```

Out[816…
```
[('based', 'AI'),
 ('data', 'assets'),
 ('language', 'based'),
 ('foundation', 'models'),
 ('tasks', 'like'),
 ('text', 'data'),
 ('tools', 'like'),
 ('AI', 'tools'),
 ('chain', 'crisis'),
 ('language', 'model')]
Begin incorporating new language-based AI tools for a variety of tasks to better underst
and their capabilities..
Startups like Verneek are creating Elicit-like tools to enable everyone to make data-inf
```

```
ormed decisions.
Models like GPT-3 are considered to be foundation models — an emerging AI research area
— which also work for other types of data such as images and video.
Language-based AI won't replace jobs, but it will automate many tasks, even for decision
makers.
Understand how you might leverage AI-based language technologies to make better decision
s or reorganize your skilled labor..
```

Analysis:

For this problem, I use a score to calculate the word frequency, then I divide it by sentence length for normalization. I also got the most 10 frequent bigrams and trigrams in order to find out what collocations are appeared very often. Then I added a penalty for 4-garms. Which are ai language based tools since it appear in many sentence and it counts as 4 term frequency. We want to eliminate the light of frequcy of repeting colloations. At the last, I set that sentence having 10 words more can be shown because i found sentence less than 10 words mostly are title and collocations which is meaningless compare with other sentences.

To compare with test case, 2 of 5 sentences are same because I used term-frequncy based text summarization in both methods. For test case, the rest 3 of 5 sentences are more close with key words "AI" and "language". For my method, the rest of sentences gave more infomation since I eliminate short sentence and reduce the collocation frequncy.

Reference: https://medium.com/analytics-vidhya/text-summarization-using-nlp-3e85ad0c6349

Main block to test all functions

In [505…

```
if __name__ == "__main__":


    text=text = '''QQQ
Invesco QQQ Trust Series 1
Invesco Expands QQQ Innovation Suite to Include Small-Cap ETF
PR Newswire • 4 hours ago
$265.62
1.13%
add_circle_outline
AAPL
Apple Inc
Estimating The Fair Value Of Apple Inc. (NASDAQ:AAPL)
Yahoo Finance • 4 hours ago
$140.41
1.50%
add_circle_outline
TSLA
Tesla Inc
Could This Tesla Stock Unbalanced Iron Condor Return 23%?
Investor's Business Daily • 1 hour ago
$218.30
0.49%
add_circle_outline
AMZN
Amazon.com, Inc.
The Regulators of Facebook, Google and Amazon Also Invest in the Companies' Stocks
```

```
Wall Street Journal • 2 days ago
$110.91
1.76%
add_circle_outline'''


    print("\n==================\n")
    print("Test Q1")
    print(extract(text))

    print("\n==================\n")
    print("Test Q2.1")

    text = open("power_of_nlp.txt", "r", encoding='utf-8').read()

    sents, tokenized_sents = preprocess(text, lemmatized = False, pos_tag = True,
                                        remove_stopword=False, remove_punctuation = False,
                                        lower_case = False)

    idx = 1    # sentence id
    x = tokenized_sents[idx]
    concreteness, articles, adpositions,quantifier = compute_concreteness(x)

    # show sentence
    sents[idx]
    # show result
    concreteness, articles, adpositions,quantifier

    print("\n==================\n")
    print("Test Q2.2-2.4")
    sents, tokenized_sents = preprocess(text)
    tf_idf = compute_tf_idf(tokenized_sents, use_idf = True)
    summary = get_summary(tf_idf, sents, topN = 5)
    print(summary)
```

```
==================

Test Q1
  Ticker                         Name  \
0    QQQ  Invesco QQQ Trust Series 1
1   AAPL                   Apple Inc
2   TSLA                   Tesla Inc
3   AMZN            Amazon.com, Inc.

                                            Article  \
0  Invesco Expands QQQ Innovation Suite to Includ...
1  Estimating The Fair Value Of Apple Inc. (NASDA...
2  Could This Tesla Stock Unbalanced Iron Condor ...
3  The Regulators of Facebook, Google and Amazon ...

                      Media         Time    Price Change
0               PR Newswire  4 hours ago  $265.62  1.13%
1             Yahoo Finance  4 hours ago  $140.41  1.50%
2  Investor's Business Daily   1 hour ago  $218.30  0.49%
3       Wall Street Journal   2 days ago  $110.91  1.76%

==================

Test Q2.1
```

Out[505… Until recently, the conventional wisdom was that while AI was better than humans at data-driven decision making tasks, it was still inferior to humans for cognitive and creative ones.

Out[505… (0.2333333333333334,
 [(the, 'DET')],
 [(Until, 'ADP'), (at, 'ADP'), (to, 'ADP'), (for, 'ADP')],
 [(conventional, 'ADJ'), (creative, 'ADJ')])

==========

Test Q2.2-2.4

[Begin incorporating new language-based AI tools for a variety of tasks to better understand their capabilities.., Models like GPT-3 are considered to be foundation models — an emerging AI research area — which also work for other types of data such as images and video., Powerful generalizable language-based AI tools like Elicit are here, and they are just the tip of the iceberg; multimodal foundation model-based tools are poised to transform business in ways that are still difficult to predict., A Language-Based AI Research Assistant., There is so much text data, and you don’t need advanced models like GPT-3 to extract its value.]