We use summarizations in many places in our lives. While reading about a book, the blurb at the back provides us with a general idea about it. News apps often have small captions that talk about the article and movie reviews are another place where summaries prove to be extremely valuable.

With the advent of technology such as natural language processing (NLP) and machine learning, why not use them to generate summaries without human intervention?

Before you decide to summarize your textbook for the upcoming exam, let us learn some important concepts.

Summarization of a text can be of two types — extractive and abstractive.

Extractive Summarization

It can be defined as hand-picking any important sentence and adding it to the summary as it is.

1. Read the text and split it into sentences.
2. Parse through each sentence, identify which ones are more important than others based on certain parameters and assign them a score.
3. Pick out the sentences that rank above others.

4. Add these sentences to the summary and TA DA! You're done.

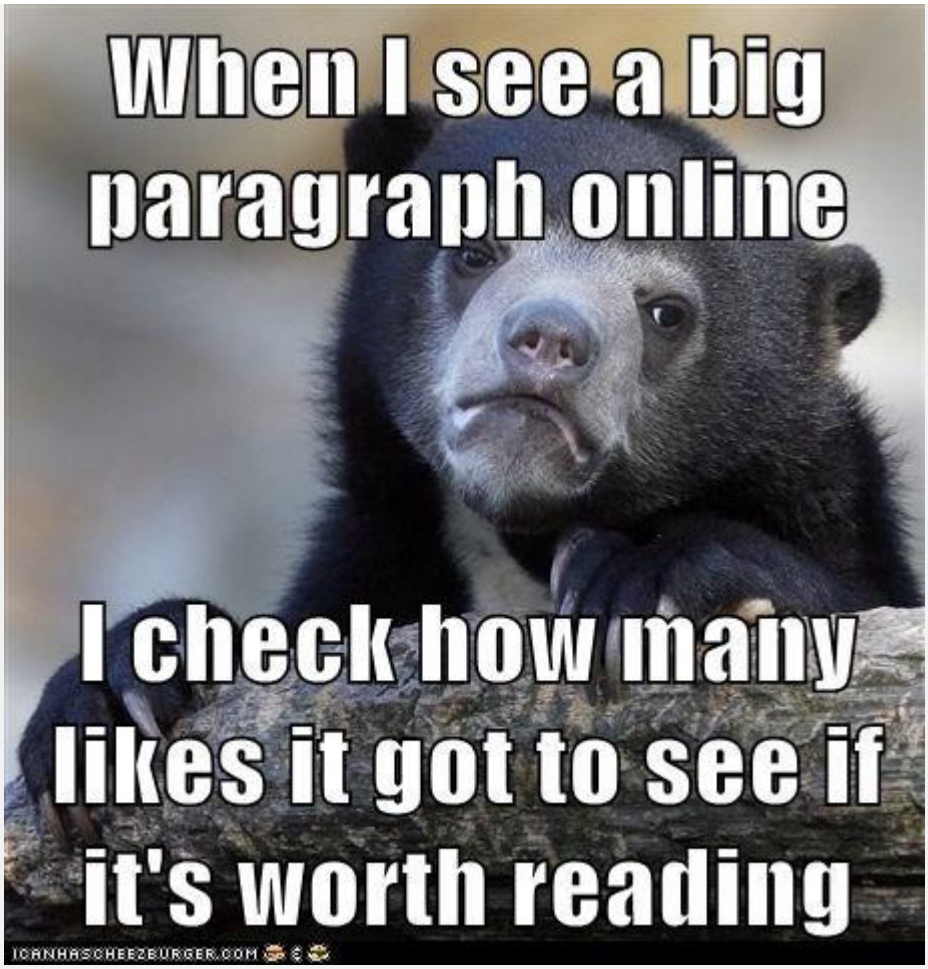
Abstractive Summarization

It is more like how humans summarize a piece of text, understanding and analyzing the article.

1. Read the text.
2. Analyze the underlying meaning of the text and the sentences.
3. Pick out the important topics and create new sentences (may or may not use vocabulary from the article).
4. Add these sentences to the summary and voila!

As you can see, extractive summarization is relatively easier when compared to abstractive summarization since it doesn't have to deal with semantics or vocabulary.

In this article, we'll discuss extractive summarization and then proceed to create our very own text summarizer from scratch. Excited?



credits

Term Frequency — Inverse Document Frequency (TF–IDF)

Given a sentence, we can easily identify what it's talking about and what it's trying to imply. However, a computer can't do it. So, what do we do? As discussed above, we rank the sentences and pick the highest-ranked ones. Who decides the rank of the sentences? TF-IDF.

Term frequency-inverse document frequency is what we'll be using as the basis for selecting sentences that'll make it to the final summary. The TF-IDF value is calculated as given below:

TF-IDF = Term Frequency x Inverse Document Frequency

Term Frequency (TF)

This is a measure of the number of times a word appears in our document. Taking only the term frequency will give a highly inaccurate result because a longer document will have a greater TF for a word 'w' when compared to a shorter document. However, this can be normalized if we divide the frequency by the total number of words. This makes TF a more valuable quantity for us.

TF(w, d) = occurrences of word 'w' in the document 'd / total number of words in the document 'd'

This is it right? Why do we need anything else? Well, if you think about it, all the words that are extremely common such as "is, am, was" etc. will have a very high TF value. The term TF on

its own will once again prove to be redundant, so behold IDF, our saviour!

Inverse Document Frequency (IDF)

Before we jump to IDF, let's discuss document frequency. DF is the number of documents that contain the word 'w' out of the total N documents. 'D' is the set of all documents.

$$DF(w, D) = \text{number of documents containing word 'w'}$$

However, DF is not of concern to us, it's the inverse which is relevant. It tells us how much information each term carries. As a result of inversion, common words will have a low IDF value while terms with a lesser occurrence gain higher precedence. We take the logarithm of this value because, for large documents, the value of this term would sky-rocket.

$$IDF(w, D) = \log(N / (DF(w, D) + 1))$$

We do division by (DF + 1) to avoid division by zero. It can prove to be a serious prick for some occasions.

$$TF\text{-}IDF(w, d) = TF(w, d) * \log(N / (DF + 1))$$

Once all this is calculated, we rank each document (here, sentences) based on their TF-IDF scores and a threshold value, we finally get a summary of our text. Pretty cool right? Now, let's get coding!

Code Segment

Importing Libraries

```
import re
import nltk
import math
from nltk.tokenize import word_tokenize
```

Cleaning the Text

```
def clean_text(file_name):
    file = open(file_name, "r")
    filedata = file.readlines()
    article = filedata[0].split(". ")
    sentences = []
    # removing special characters and extra whitespaces
    for sentence in article:
        sentence = re.sub('[^a-zA-Z]', ' ', str(sentence))
        sentence = re.sub('[\s+]', ' ', sentence)
        sentences.append(sentence)
    sentences.pop()
    display = " ".join(sentences)
    print('Initial Text: ')
    print(display)
    print('\n')
    return sentences
```

Converting the text file into individual sentences which will be our document

Number of Words in Each Sentence

```
def cnt_words(sent):
    cnt = 0
    words = word_tokenize(sent)
    for word in words:
        cnt = cnt + 1
    return cnt

# getting data about each sentence (frequency of words)
def cnt_in_sent(sentences):
    txt_data = []
    i = 0
    for sent in sentences:
        i = i + 1
        cnt = cnt_words(sent)
        temp = {'id' : i, 'word_cnt' : cnt}
        txt_data.append(temp)
    return txt_data
```

Number of words in each sentence

Creating List of Frequencies for each Word in all Documents

```
def freq_dict(sentences):
    i = 0
    freq_list = []
    for sent in sentences:
        i = i + 1
        freq_dict = {}
        words = word_tokenize(sent)
        for word in words:
            word = word.lower()
            if word in freq_dict:
                freq_dict[word] = freq_dict[word] + 1
            else:
                freq_dict[word] = 1
            temp = {'id' : i, 'freq_dict' : freq_dict}
        freq_list.append(temp)
    return freq_list
```

Here, we create a list which stores the frequency of each word of the document

Calculating TF and IDF Values

```
def calc_TF(text_data, freq_list):
    tf_scores = []
    for item in freq_list:
        ID = item['id']
        for k in item['freq_dict']:
            temp = {
                'id': item['id'],
                'tf_score': item['freq_dict'][k]/text_data[ID-1]['word_cnt'],
                'key': k
                }
            tf_scores.append(temp)
    return tf_scores

#calculating the inverse document frequency
def calc_IDF(text_data, freq_list):
    idf_scores =[]
    cnt = 0
    for item in freq_list:
        cnt = cnt + 1
        for k in item['freq_dict']:
            val = sum([k in it['freq_dict'] for it in freq_list])
            temp = {
                'id': cnt,
                'idf_score': math.log(len(text_data)/(val+1)),
                'key': k}
            idf_scores.append(temp)
    return idf_scores
```

The first function calculates the term frequency of words in each document (here, sentences). The second function calculates the inverse document frequency for each word in the sentences.

Calculating TF-IDF Values

```
def calc_TFIDF(tf_scores, idf_scores):
    tfidf_scores = []
    for j in idf_scores:
        for i in tf_scores:
            if j['key'] == i['key'] and j['id'] == i['id']:
                temp = {
                    'id': j['id'],
                    'tfidf_score': j['idf_score'] * i['tf_score'],
                    'key': j['key']
                    }
                tfidf_scores.append(temp)
    return tfidf_scores
```

Computing the TF-IDF scores for each term

Ranking all the Documents

```
def sent_scores(tfidf_scores, sentences, text_data):
    sent_data = []
    for txt in text_data:
        score = 0
        for i in range(0, len(tfidf_scores)):
            t_dict = tfidf_scores[i]
            if txt['id'] == t_dict['id']:
                score = score + t_dict['tfidf_score']
        temp = {
            'id': txt['id'],
            'score': score,
            'sentence': sentences[txt['id']-1]}
        sent_data.append(temp)
    return sent_data
```

We calculate the score for each sentence based on the TF-IDF value for each word in the sentence

Generating the Summary

```
def summary(sent_data):
    cnt = 0
    summary = []
    for t_dict in sent_data:
        cnt  = cnt + t_dict['score']
    avg = cnt / len(sent_data)
    for sent in sent_data:
        if sent['score'] >= (avg * 0.9):
            summary.append(sent['sentence'])
    summary = ". ".join(summary)
    return summary
```

The threshold is calculated as a linear function of the average of the TF-IDF scores

(Note: the threshold for average can be modified by multiplying a scalar to reduce or increase the size of the summary)

And…Time to call all the functions!

```
sentences =  clean_text('text.txt')
text_data = cnt_in_sent(sentences)

freq_list = freq_dict(sentences)
tf_scores = calc_TF(text_data, freq_list)
idf_scores = calc_IDF(text_data, freq_list)

tfidf_scores = calc_TFIDF(tf_scores, idf_scores)

sent_data = sent_scores(tfidf_scores, sentences, text_data)
result = summary(sent_data)

print('Final Summary: ')
print(result)
```

Function calls

(Note: Ensure that your textual matter is in a file (*.txt) and is in the same directory as your Python script)

Testing…testing…1, 2, 3!

```
In [13]: runfile('/Users/anjaneyatripathi/Desktop/nlp_1/prac.py', wdir='/Users/anjaneyatripathi/Desktop/nlp_1')
Initial Text:
America has changed dramatically during recent years Not only has the number of graduates in traditional engineering disciplines such as
mechanical  civil  electrical  chemical  and aeronautical engineering declined  but in most of the premier American universities
engineering curricula now concentrate on and encourage largely the study of engineering science  As a result  there are declining
offerings in engineering subjects dealing with infrastructure  the environment  and related issues  and greater concentration on high
technology subjects  largely supporting increasingly complex scientific developments While the latter is important  it should not be at
the expense of more traditional engineering Rapidly developing economies such as China and India  as well as other industrial countries
in Europe and Asia  continue to encourage and advance the teaching of engineering Both China and India  respectively  graduate six and
eight times as many traditional engineers as does the United States


Final Summary:
America has changed dramatically during recent years.  As a result  there are declining offerings in engineering subjects dealing with
infrastructure  the environment  and related issues  and greater concentration on high technology subjects  largely supporting
increasingly complex scientific developments. While the latter is important  it should not be at the expense of more traditional
engineering. Both China and India  respectively  graduate six and eight times as many traditional engineers as does the United States
```

Oooh, La La!

Conclusion — The End of a Wonderful Journey

Well, I hope you had fun coding along with me and learning about text summarization while we were at it. However, this isn't the only way to summarize a text, there are many more techniques that can be implemented to accomplish the task. Try checking them out too!

Hope you enjoyed it and see you next time!