

▼ Name - Darshika Verma

Date - July 6, 2024

```
pip install pyLDAvis
```

 [Show hidden output](#)

```
pip install sumy
```

 [Show hidden output](#)

```
pip install rouge_score
```

 [Show hidden output](#)

```
pip install sentence_transformers
```

 [Show hidden output](#)

```
pip install faiss-cpu
```

 [Show hidden output](#)

```
# Import Libraries
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from gensim.models import LdaModel
from gensim.corpora import Dictionary
from pprint import pprint

import pyLDAvis
import pyLDAvis.gensim_models
import matplotlib.pyplot as plt
import pandas as pd

from sumy.parsers.html import HtmlParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.summarizers.text_rank import TextRankSummarizer
from urllib.request import urlopen
from bs4 import BeautifulSoup
import re
import requests

from sklearn.feature_extraction.text import TfidfVectorizer
from nltk import tokenize
import numpy as np

from sumy.parsers.plaintext import PlaintextParser
from sumy.nlp.tokenizers import Tokenizer
from sumy.nlp.stemmers import Stemmer
from sumy.utils import get_stop_words
from sumy.summarizers.lsa import LsaSummarizer
from rouge_score import rouge_scorer

import faiss
import time
import numpy as np
import torch
import os
import pandas as pd
from sentence_transformers import SentenceTransformer
```

 [Show hidden output](#)

▼ Step 1: Import Data

```
headers = {
    'User-Agent': 'Dverma1@my.harrisburgu.edu'
}

response = requests.get(
    'https://en.wikipedia.org/w/api.php',
    params={
        'action': 'parse',
        'format': 'json',
        'page': 'Diabetes',
        'prop': 'text',
        'redirects': ''
    }).json()

raw_html = response['parse']['text']['*']
soup = BeautifulSoup(raw_html, 'html.parser')
soup.find_all('p')
text = ''

paragraphs = []

for p in soup.find_all('p'):
    text = p.text
    text = re.sub('\\\\[[0-9]*\\\\]', '', text)
    text = re.sub('\\\\n', '', text)
    paragraphs.append(text)
```

```
⚡ /usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass and should_run_async(code)
```

```
summary = paragraphs[0:2]
main_text = paragraphs[2:]
summary = ' '.join(summary)
main_text = ' '.join(main_text)
```

```
⚡ /usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass and should_run_async(code)
```

▼ Step 2: Create A Search Engine

```
data = pd.DataFrame(paragraphs, columns = ['sentence'])
data = data.drop(0)
data.head()
```

```
⚡ /usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass and should_run_async(code)
```

	sentence
1	Diabetes mellitus, often known simply as diab...
2	The major types of diabetes are type 1 and typ...
3	As of 2021, an estimated 537 million people ha...
4	The classic symptoms of untreated diabetes are...
5	Diabetic ketoacidosis is a medical emergency t...

```
## Load a pre-trained model
model = SentenceTransformer('msmarco-MiniLM-L-12-v3')
article = data["sentence"].tolist()
article_nlp = model.encode(article)
```

```
# Create an index using FAISS
index = faiss.IndexFlatL2(article_nlp.shape[1])
index.add(article_nlp)
faiss.write_index(index, 'index_article')
index = faiss.read_index('index_article')
```

```
⚡ /usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass and should_run_async(code)
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/tokens)
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
modules.json: 100%                229/229 [00:00<00:00, 7.47kB/s]

config_sentence_transformers.json: 100%                122/122 [00:00<00:00, 5.55kB/s]

README.md: 100%                3.72k/3.72k [00:00<00:00, 131kB/s]

sentence_bert_config.json: 100%                53.0/53.0 [00:00<00:00, 1.58kB/s]
/usr/local/lib/python3.10/dist-packages/huggingface_hub/file_download.py:1132: FutureWarning: `resume_download` is deprecated and will be removed in version 1.0.0. To retry a failed download, please use the `resume` parameter instead.
  warnings.warn(
config.json: 100%                629/629 [00:00<00:00, 29.9kB/s]

model.safetensors: 100%                133M/133M [00:00<00:00, 182MB/s]

tokenizer_config.json: 100%                432/432 [00:00<00:00, 15.8kB/s]

vocab.txt: 100%                232k/232k [00:00<00:00, 3.39MB/s]

tokenizer.json: 100%                466k/466k [00:00<00:00, 9.86MB/s]

special_tokens_map.json: 100%                112/112 [00:00<00:00, 4.30kB/s]

1_Pooling/config.json: 100%                190/190 [00:00<00:00, 7.92kB/s]
```

```
# define a search
def search(query):

    t=time.time()
    query_vector = model.encode([query])
    k = 5
    top_k = index.search(query_vector, k)
    print('totaltime: {}'.format(time.time()-t))
    return [article[_id] for _id in top_k[1].tolist()[0]]
```

```
# use the search
results = search("symptoms")
```

```
⚡ totaltime: 0.044490814208984375
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass and should_run_async(code)
```

results

```
⚡ /usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass and should_run_async(code)
['The symptoms may relate to fluid loss and polyuria, but the course may also be insidious. Diabetic animals are more prone to infections. The long-term complications recognized in humans are much rarer in animals. The principles of treatment (weight loss, oral antidiabetics, subcutaneous insulin) and management of emergencies (e.g. ketoacidosis) are similar to those in humans.',
'Diabetes mellitus, often known simply as diabetes, is a group of common endocrine diseases characterized by sustained high blood sugar levels. Diabetes is due to either the pancreas not producing enough insulin, or the cells of the body becoming unresponsive to the hormone's effects. Classic symptoms include thirst, polyuria, weight loss, and blurred vision. If left untreated, the disease can lead to various health complications, including disorders of the cardiovascular system, eye, kidney, and nerves. Diabetes accounts for approximately 4.2 million deaths every year, with an estimated 1.5 million caused by either untreated or poorly treated diabetes.',
'The classic symptoms of untreated diabetes are polyuria, thirst, and weight loss. Several other non-specific signs and symptoms may also occur, including fatigue, blurred vision, and genital itchiness due to Candida infection. About half of affected individuals may also be asymptomatic. Type 1 presents abruptly following a pre-clinical phase, while type 2 has a more insidious onset; patients may remain asymptomatic for many years.',
'Hypoglycaemia is a recognised complication of insulin treatment used in diabetes. An acute presentation can include mild symptoms such as sweating, trembling, and palpitations, to more serious effects including impaired cognition, confusion, seizures, coma, and rarely death. Recurrent hypoglycaemic episodes may lower the glycaemic threshold at which symptoms occur, meaning mild symptoms may not appear before cognitive deterioration begins to occur.',
'Diabetic ketoacidosis is a medical emergency that occurs most commonly in type 1, but may also occur in type 2 if it has been longstanding or if the individual has significant β-cell dysfunction. Excessive production of ketone bodies leads to signs and symptoms including nausea, vomiting, abdominal pain, the smell of acetone in the
```

breath, deep breathing known as Kussmaul breathing, and in severe cases decreased level of consciousness. Hyperosmolar hyperglycemic state is another emergency characterised by dehydration secondary to severe hyperglycaemia, with resultant hypernatremia leading to an altered mental state and possibly coma.']

```
#Search for several items (at least three words or phrases).
import time

def search(queries):
    results = {}
    for query in queries:
        t = time.time()
        query_vector = model.encode([query])
        k = 5
        top_k = index.search(query_vector, k)
        print('Total time for "{}": {:.2f} seconds'.format(query, time.time() - t))
        results[query] = [article[_id] for _id in top_k[1].tolist()[0]]
    return results

# Use the search function for several items
queries = ["medical emergency", "Complications of diabetes", "World Health Organization"]
results = search(queries)

# Display the results
for query, res in results.items():
    print(f"\nResults for '{query}':")
    for item in res:
        print(item)
```

```
➦ Total time for "medical emergency": 0.05 seconds
Total time for "Complications of diabetes": 0.04 seconds
Total time for "World Health Organization": 0.04 seconds

Results for 'medical emergency':
In 2010, diabetes-related emergency room (ER) visit rates in the United States were higher among people from the lowest income communities (526 per 10,000 population) than from
Diabetic ketoacidosis is a medical emergency that occurs most commonly in type 1, but may also occur in type 2 if it has been longstanding or if the individual has significant β
The symptoms may relate to fluid loss and polyuria, but the course may also be insidious. Diabetic animals are more prone to infections. The long-term complications recognized i
Though it may be transient, untreated gestational diabetes can damage the health of the fetus or mother. Risks to the baby include macrosomia (high birth weight), congenital hea
Diabetes patients' comorbidities have a significant impact on medical expenses and related costs. It has been demonstrated that patients with diabetes are more likely to experie

Results for 'Complications of diabetes':
The major long-term complications of diabetes relate to damage to blood vessels at both macrovascular and microvascular levels. Diabetes doubles the risk of cardiovascular disea
Adverse childhood experiences, including abuse, neglect, and household difficulties, increase the likelihood of type 2 diabetes later in life by 32%, with neglect having the str
Diabetes patients' comorbidities have a significant impact on medical expenses and related costs. It has been demonstrated that patients with diabetes are more likely to experie
The following is a list of disorders that may increase the risk of diabetes:
Hearing loss is another long term complication associated with diabetes.

Results for 'World Health Organization':
In countries using a general practitioner system, such as the United Kingdom, care may take place mainly outside hospitals, with hospital-based specialist care used only in case
As of 2021, an estimated 537 million people had diabetes worldwide accounting for 10.5% of the adult population, with type 2 making up about 90% of all cases. It is estimated th
When there is too much glucose in the blood for a long time, the kidneys cannot absorb it all (reach a threshold of reabsorption) and the extra glucose gets passed out of the bc
The symptoms may relate to fluid loss and polyuria, but the course may also be insidious. Diabetic animals are more prone to infections. The long-term complications recognized i
Type 1 accounts for 5 to 10% of diabetes cases and is the most common type diagnosed in patients under 20 years; however, the older term "juvenile-onset diabetes" is no longer u
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass
and should_run_async(code)
```

▼ Step 3: Create Text Summary

```
import nltk
nltk.download('punkt')
main_text_sent = tokenize.sent_tokenize(main_text)
num_summary_sentence = 4

##Text Rank Summary
textrank_summary = []

parser = PlaintextParser.from_string(main_text, Tokenizer("english"))
summarizer = TextRankSummarizer()
for sentence in summarizer(parser.document, num_summary_sentence):
    textrank_summary.append(str(sentence))

##LSA Summary
lsa_summary = []

LANGUAGE = "english"
stemmer = Stemmer(LANGUAGE)

parser = PlaintextParser.from_string(main_text, Tokenizer(LANGUAGE))
summarizer = LsaSummarizer(stemmer)
summarizer.stop_words = get_stop_words(LANGUAGE)

for sentence in summarizer(parser.document, num_summary_sentence):
    lsa_summary.append(str(sentence))

➦ /usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass
and should_run_async(code)
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data] Unzipping tokenizers/punkt.zip.
```

```
#LDA Summary
import pandas as pd
nltk.download('stopwords')
#tokenize, remove stopwords, non-alphabetic words, lowercase
def preprocess(textstring):
    stops = set(stopwords.words('english'))
    tokens = word_tokenize(textstring)
    return [token.lower() for token in tokens if token.isalpha()
            and token not in stops]

processed_sentences = [preprocess(sent) for sent in main_text_sent]

dictionary = Dictionary(processed_sentences)
corpus = [dictionary.doc2bow(sent) for sent in processed_sentences]


LDAmode1 = LdaModel(corpus = corpus,
                    id2word = dictionary,
                    iterations = 400,
                    num_topics = 5,
                    random_state = 100,
                    update_every = 1,
                    chunksize = 100,
                    passes = 10,
                    alpha = 'auto',
                    per_word_topics = True)

probs = [LDAmode1.get_document_topics(sentence) for sentence in corpus]


save_probs = []
i = 0
for document in probs:
    for (topic, prob) in document:
        if topic == 1:
            save_probs.append((main_text_sent[i], prob))
    i = i + 1

DF = pd.DataFrame(save_probs, columns = ["sentence", "prob"])
lda_summary = "".join(DF.sort_values(by = ["prob"], ascending = False)[0:3].sentence)

lda_summary
```

 /usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `should_run_async(code)` and should_run_async(code)
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data] Package stopwords is already up-to-date!
'The following is a list of disorders that may increase the risk of diabetes: Insulin is the principal hormone that regulates the uptake of glucose from the blood into most cells of the body, especially liver, adipose tissue and muscle, except smooth muscle, in which insulin acts via the IGF-1.A 2016 systematic review found potential harm to treatment targets lower than 140 mmHg, and a subsequent systematic review in 2019 found no evidence of additional benefit from blood pressure lowering to between 130 – 140mmHg, although there was an increased risk of adverse events. The body obtains glucose from three main sources: the intestinal absorption of food; the breakdown of glycogen (glycogenolysis); the storage form of glucose found in the liver and gluconeogenesis, the generation of glucose from non-carbohydrate precursors.'

```
# Show top words for each topic
topics = LDAmode1.show_topics(num_topics=5, num_words=10, formatted=False)
for topic_num, topic in topics:
    print(f"Topic {topic_num}:")
    print(", ".join([word for word, prob in topic]))
```

 Topic 0:
meaning, symptoms, the, disease, also, may, word, treatment, insulin, comes
Topic 1:
diabetes, glucose, type, risk, the, blood, higher, people, body, lower
Topic 2:
diabetes, type, mellitus, also, likely, affected, time, breeds, in, due
Topic 3:
diabetes, disease, sugar, insulin, type, cardiovascular, blood, complications, people, evidence
Topic 4:
diabetes, the, first, cases, insulin, less, glucose, types, early, management
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass and should_run_async(code)

Topic 0: General information about diabetes, including definitions, symptoms, and treatments.


Topic 1: The relationship between diabetes, blood glucose levels, and associated risk factors.

Topic 2: Different types of diabetes and the demographics affected by it. Topic 3: Complications of diabetes, particularly cardiovascular issues.

Topic 4: The history of diabetes and its management.

```
import pyLDAvis
import pyLDAvis.gensim_models
import matplotlib.pyplot as plt


vis = pyLDAvis.gensim_models.prepare(LDAmode1, corpus, dictionary, n_jobs = 1)
pyLDAvis.save_html(vis, 'LDA_Visualization_diabetes.html') ##saves the file
```

 /usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass and should_run_async(code)

```
def print_rouge_score(rouge_score):
    for k,v in rouge_score.items():
        print(k, 'Precision:', "{:.2f}".format(v.precision), 'Recall:', "{:.2f}".format(v.recall), 'fmeasure:', "{:.2f}".format(v.fmeasure))

textrank_sum = "".join(str(textrank_summary))

scorer = rouge_scorer.RougeScorer(['rouge1'], use_stemmer=True)
scores = scorer.score(summary, textrank_sum)
print_rouge_score(scores)
```

 rouge1 Precision: 0.19 Recall: 0.37 fmeasure: 0.25
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass and should_run_async(code)

```
lsa_sum = "".join(str(lsa_summary))

scorer = rouge_scorer.RougeScorer(['rouge1'], use_stemmer=True)
scores = scorer.score(summary, lsa_sum)
print_rouge_score(scores)

rouge1 Precision: 0.20 Recall: 0.34 fmeasure: 0.26
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass
and should_run_async(code)

scorer = rouge_scorer.RougeScorer(['rouge1'], use_stemmer=True)
scores = scorer.score(summary, lda_summary)
print_rouge_score(scores)

rouge1 Precision: 0.20 Recall: 0.27 fmeasure: 0.23
/usr/local/lib/python3.10/dist-packages/ipykernel/ipkernel.py:283: DeprecationWarning: `should_run_async` will not call `transform_cell` automatically in the future. Please pass
and should_run_async(code)
```

Step 4: Intepretation

- **What did you learn about the chosen webpage from the above analysis? How well did the search engine work? What could make it better? How well does the text summary compare to your understanding of the text? (Write at least 5 sentences)**

Answer

1. **Webpage:** From the above analysis, I learned that the chosen webpage contains detailed information on diabetes, covering various aspects such as general definitions, symptoms, treatments, blood glucose levels, risk factors, types of diabetes, complications (particularly cardiovascular), and historical management.
2. **Search Engine:** The search engine effectively retrieved relevant information for the queried topics: "medical emergency," "Complications of diabetes," and "World Health Organization." Each query returned paragraphs closely related to the respective topics, discussing emergency situations related to diabetes, various complications, and global diabetes statistics and management. The search queries were processed quickly, with each query taking less than 0.05 seconds, indicating efficient retrieval from the indexed data.
3. **Improving Search Engine:** Implementing techniques to expand queries with synonyms or related terms could improve the search engine's ability to capture broader aspects of the topics. Also, enhancing the model's ability to understand the context of queries, especially in medical or technical domains like diabetes, could refine result relevance.
4. **Text Summary:** The summaries generated using LSA and LDA methods align well with the key themes identified in the text analysis, as reflected in the slightly higher ROUGE-1 precision scores. These methods successfully distilled the essential information, providing a coherent and relevant summary that matches the detailed content on the webpage.

ROUGE-1 Precision Scores:
TextRank Summary: Precision 0.19
LSA Summary: Precision 0.20
LDA Summary: Precision 0.20

Step 5: Application

Describe some real-world application that could use the techniques from this section. Define a problem or question, what data would be needed, how the data would be analyzed, and what insights/solutions could be drawn from the analysis.

Answer

One of such real-world application would be involving **Academic Research Literature Review**.

Problem: Researchers need to conduct comprehensive literature reviews across multiple academic journals and publications to identify gaps in knowledge, emerging trends, and relevant studies for their research.

Data Needed: Academic papers, journal articles, conference proceedings, and research abstracts.

Data Analysis Techniques: Text mining for keyword extraction and topic modeling to cluster research articles by themes; citation analysis to identify influential papers and authors; and summarization to distill key findings from extensive literature.

Insights/Solutions: Insights can help researchers stay updated on the latest advancements, identify collaborators, and formulate research hypotheses based on gaps in current knowledge.