EYA SAID

[Wael HCINE](mailto:wael.hcine@esprit.tn)

[Amine BOUZOUITA](mailto:amine.bouzouita@esprit.tn)

Wala Sassi

I. Business Understanding:

objectif commercial consiste à obtenir des informations et des connaissances à partir de l'article intitulé "Text Mining: Techniques, Applications and Issues" de Ramzan Talib et de ses collègues. Ces informations pourraient être précieuses pour diverses parties prenantes, telles que les chercheurs, les professionnels de l'extraction de texte ou les organisations intéressées par l'analyse de texte.

Objectif Métier : Comprendre comment les techniques de text mining peuvent être utilisées pour extraire des informations pertinentes à partir de grands volumes de données textuelles.

* Comment les entreprises et les organisations peuvent-elles tirer parti du text mining pour prendre des décisions plus éclairées et découvrir des connaissances cachées ?
* Quelles sont les tendances émergentes dans le domaine du text mining ?
* Comprendre comment les techniques de text mining peuvent améliorer la prise de décision.
* Identifier les domaines d'application potentiels du text mining.

II. Data understanding:

L'objectif de cette étape est d'acquérir une compréhension approfondie du contenu de l'article, de ses thèmes principaux et de ses caractéristiques textuelles. Cela servira de base pour les étapes suivantes de l'analyse, telles que l'identification des thèmes sous-jacents et la réduction de la dimensionnalité.

1. Collecte des données : Dans ce cas, la collecte de données consiste à obtenir le texte de l'article que vous analysez. Vous extrayez le contenu textuel de l'article, que ce soit à partir d'une source en ligne, d'un fichier PDF, ou d'un autre format de document.
2. Exploration des données : Cette étape inclut la lecture et la compréhension du texte de l'article. Vous devez identifier les sections clés de l'article, les thèmes abordés, les termes et concepts importants. Vous pouvez également effectuer une première analyse des statistiques de base, comme la longueur de l'article, le nombre de mots, etc.
3. Catégorisation des textes : Vous pouvez catégoriser les parties de l'article en fonction de leur contenu. Par exemple, vous pouvez identifier les sections qui traitent des techniques de text mining, des applications, des problèmes et des défis.
4. Prétraitement des données : Dans cette étape, vous pouvez nettoyer le texte en éliminant les caractères spéciaux, les chiffres, et en normalisant le texte (minuscules/majuscules). Cela vous permet de préparer le texte pour une analyse plus approfondie.
5. Création d'un Word Cloud : La création d'un Word Cloud est une technique courante pour visualiser les mots les plus fréquemment utilisés dans l'article. Cela vous donne une vue rapide des concepts clés.

**Extraction du texte du PDF**

Pour extraire le texte à partir du fichier PDF "Text\_Mining.pdf", on peut utiliser une bibliothèque Python appelée "PyPDF2":

import PyPDF2

pdf\_file\_path = 'C:\\Users\\SAID EYA\\Downloads\\Text\_Mining.pdf'

def extract\_text\_from\_pdf(pdf\_path):

text = ""

with open(pdf\_path, "rb") as pdf\_file:

pdf\_reader = PyPDF2.PdfReader(pdf\_file)

for page\_num in range(len(pdf\_reader.pages)):

page = pdf\_reader.pages[page\_num]

text += page.extract\_text()

return text

text\_from\_pdf = extract\_text\_from\_pdf(pdf\_file\_path)

**Catégorisation des textes**

Identifiez les sections de l'article en fonction de leur contenu.

**Création d'un Word Cloud**

pour créer un Word Cloud à partir du texte extrait on peut utiliser la bibliothèque "wordcloud" et "matplotlib" :

from wordcloud import WordCloud

import matplotlib.pyplot as plt

def create\_wordcloud(text):

# Specify the path to the downloaded Google Font file

font\_path = "C:\\Users\\SAID EYA\\Downloads\\Roboto\\Roboto-Bold.ttf"

wordcloud = WordCloud(width=800, height=400, font\_path=font\_path).generate(text)

plt.figure(figsize=(10, 5))

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis("off")

plt.show()

create\_wordcloud(text\_from\_pdf)

III. Data Preparation:

**Prétraitement des données**

Nettoyez le texte en éliminant les caractères spéciaux, les chiffres et en normalisant le texte (minuscules/majuscules). Vous pouvez utiliser des expressions régulières pour cela.

import re

def preprocess\_text(text):

text = re.sub(r"[^a-zA-Z]", " ", text)

text = text.lower()

return text

cleaned\_text = preprocess\_text(text\_from\_pdf)

print(cleaned\_text)

IV. Modeling:

**Analyse des thèmes**

Pour identifier les thèmes sous-jacents, on peut utiliser des techniques comme LDA (Latent Dirichlet Allocation) en utilisant des bibliothèques telles que "scikit-learn" :

from sklearn.decomposition import LatentDirichletAllocation

from sklearn.feature\_extraction.text import CountVectorizer

# Créer une matrice de termes

vectorizer = CountVectorizer()

X = vectorizer.fit\_transform(corpus)

# Appliquer LDA

lda = LatentDirichletAllocation(n\_components=3, random\_state=42)

lda.fit(X)

# Identifier les thèmes

print\_top\_words(lda, vectorizer.get\_feature\_names(), n\_top\_words=5)

Pour effectuer une analyse sémantique latente (LSI, Latent Semantic Indexing) en utilisant Python, on peut utiliser le module Gensim. Gensim est une bibliothèque populaire pour le traitement du langage naturel qui prend en charge diverses techniques de modélisation de sujets, y compris LSI:

import gensim

from gensim import corpora, models

# Tokenize your text (assuming cleaned\_text is a list of strings)

tokenized\_text = [re.findall(r'\w+', text) for text in cleaned\_text]

# Create a dictionary

dictionnaire = corpora.Dictionary(tokenized\_text)

# Create a corpus using doc2bow

corpus = [dictionnaire.doc2bow(text) for text in tokenized\_text]

# Apply LSI

lsi = models.LsiModel(corpus, id2word=dictionnaire, num\_topics=3) # Change the number of topics as needed

# Display topics

topics = lsi.print\_topics()

for topic\_id, topic in topics:

print(f"Topic {topic\_id + 1}: {topic}")

# Generate and display word clouds for each LSI topic

for topic\_id, topic in topics:

words = [word for word, \_ in re.findall(r'(\w+)', topic)]

print(f"Topic {topic\_id + 1}: {' '.join(words)}")

# Create and display a word cloud

wordcloud = WordCloud(width=800, height=400).generate(' '.join(words))

plt.figure(figsize=(8, 4))

plt.imshow(wordcloud, interpolation="bilinear")

plt.axis("off")

plt.title(f"Topic {topic\_id + 1} Word Cloud")

plt.show()

V. Test & Evaluation:

For LSI

lsi\_topics = lsi.print\_topics()

lsi\_results = lsi[corpus] # Assuming 'corpus' is the preprocessed text corpus

# Print the LSI results for the documents in cleaned\_text

for i, doc in enumerate(lsi\_results):

print(f"Document {i + 1}:")

for topic\_id, score in doc:

print(f"LSI Topic {topic\_id + 1}: Score = {score:.4f}, Words = {lsi\_topics[topic\_id]}")

print()

For Lda

test\_corpus = vectorizer.transform([cleaned\_text]) # Vectorize the test data

perplexity = lda.perplexity(test\_corpus)

print(f"Perplexity: {perplexity}")

La perplexité est une mesure couramment utilisée pour évaluer la qualité d'un modèle de traitement du langage naturel, comme le modèle LDA (Latent Dirichlet Allocation) que nous avons utilisé. Plus la perplexité est basse, meilleure est la capacité du modèle à prédire un échantillon de données textuelles non vues. Une perplexité plus élevée indique une moins bonne performance du modèle.Dans notre cas, une perplexité de 507.54 signifie que le modèle LDA a une certaine difficulté à prédire correctement les données textuelles non vues.

VI. Deployment: