MASTER ISI :

DATA MINING

TP N°2

Réalisé par :

ESSADEQ Ayoub

RAHHAOUI ABDESSAMAD

2023-2024

TF (Term Frequency - Fréquence du Terme) :

Le TF, ou fréquence du terme, est une mesure qui évalue à quelle fréquence un terme (ou un mot) particulier apparaît dans un document. Il s'agit d'un indicateur de l'importance relative d'un terme au sein d'un document donné. Plus un terme apparaît fréquemment dans un document, plus son TF est élevé pour ce document.

La formule du TF est généralement : TF(t, d) = (nombre d'occurrences du terme t dans le document d) / (nombre total de termes dans le document d).

Le TF capture l'importance du terme au sein d'un document spécifique. Il est utile pour l'analyse de texte et la classification de documents, car il peut aider à identifier les termes clés d'un document.

IDF (Inverse Document Frequency - Fréquence Inverse du Document) :

L'IDF, ou fréquence inverse du document, est une mesure qui évalue l'importance d'un terme dans un ensemble de documents. L'objectif de l'IDF est de donner un poids plus élevé aux termes qui sont rares et discriminants, et un poids plus faible aux termes courants.

La formule de l'IDF est généralement : IDF(t, D) = log(N / (1 + n(t, D))), où N est le nombre total de documents dans l'ensemble de documents D, et n(t, D) est le nombre de documents contenant le terme t.

L'IDF permet de différencier les termes qui sont répandus dans l'ensemble de documents de ceux qui sont spécifiques à un petit nombre de documents. Les termes courants auront un IDF proche de zéro, tandis que les termes rares auront un IDF plus élevé.

Utilité de TF et IDF :

Le TF et l'IDF sont couramment utilisés conjointement dans le schéma TF-IDF (Term Frequency-Inverse Document Frequency) en recherche d'information et en analyse de texte. Le schéma TF-IDF est utilisé pour évaluer l'importance d'un terme dans un document par rapport à un ensemble de documents.

Le TF-IDF est largement utilisé dans la recherche d'information, la récupération de documents, la classification de textes et la recommandation de contenu. Il permet de classer et de pondérer les termes d'un document en fonction de leur importance relative par rapport à l'ensemble de documents.

Les termes avec un TF élevé dans un document et un IDF élevé parmi l'ensemble de documents auront un poids TF-IDF élevé, indiquant ainsi leur importance dans le contexte de la recherche d'informations.

Importation des bibliothèques :

Nous avons importé les bibliothèques nécessaires, y compris csv pour travailler avec des fichiers CSV, afin de stocker nos résultats.

```
import csv
```

Définition des documents :

Nous avons pris deux documents comme exemple. Le premier document est un extrait de la première ligne de la base de données IMDB, et le deuxième est un extrait de la deuxième ligne.

```
doc1 = "One of the other reviewers has mentioned that after watching just
1 Oz episode you'll be hooked. They are right, as this is exactly what
happened with me. The first thing that struck me about Oz was its
brutality and unflinching scenes of violence, which set in right from the
word GO. Trust me, this is not a show for the faint-hearted or timid. This
show pulls no punches with regards to drugs, sex, or violence. It is
hardcore, in the classic use of the word. It is called OZ as that is the
nickname given to the Oswald Maximum Security State Penitentiary. It
focuses mainly on Emerald City, an experimental section of the prison
where all the cells have glass fronts and face inwards, so privacy is not
high on the agenda. Em City is home to many..Aryans, Muslims, gangstas,
Latinos, Christians, Italians, Irish and more....so scuffles, death
stares, dodgy dealings, and shady agreements are never far away. I would
say the main appeal of the show is due to the fact that it goes where
other shows wouldn't dare. Forget pretty pictures painted for mainstream
audiences, forget charm, forget romance...OZ doesn't mess around. The
first episode I ever saw struck me as so nasty it was surreal, I couldn't
say I was ready for it, but as I watched more, I developed a taste for Oz,
and got accustomed to the high levels of graphic violence. Not just
violence, but injustice (crooked guards who'll be sold out for a nickel,
inmates who'll kill on order and get away with it, well-mannered, middle-
class inmates being turned into prison bitches due to their lack of street
skills or prison experience) Watching Oz, you may become comfortable with
what is uncomfortable viewing....that's if you can get in touch with your
darker side."

doc2 = "Phil the Alien is one of those quirky films where the humor is
based around the oddness of everything rather than actual punchlines. At
first, it was very odd and pretty funny, but as the movie progressed, I
didn't find the jokes or oddness funny anymore. It's a low-budget film
(that's never a problem in itself), there were some pretty interesting
characters, but eventually I just lost interest. I imagine this film would
appeal to a stoner who is currently partaking. For something similar but
better try 'Brother from another planet'."
```

```
doc3 = "I thought this was a wonderful way to spend time on a too hot
summer weekend, sitting in the air conditioned theater and watching a
light-hearted comedy. The plot is simplistic, but the dialogue is witty
and the characters are likable (even the well bread suspected serial
killer). While some may be disappointed when they realize this is not
Match Point 2: Risk Addiction, I thought it was proof that Woody Allen is
still fully in control of the style many of us have grown to love.<br
/><br />This was the most I'd laughed at one of Woody's comedies in years
(dare I say a decade?). While I've never been impressed with Scarlet
Johanson, in this she managed to tone down her 'sexy' image and jumped
right into a average, but spirited young woman.<br /><br />This may not be
the crown jewel of his career, but it was wittier than 'Devil Wears Prada'
and more interesting than 'Superman' a great comedy to go see with
friends."

doc4 = "Basically there's a family where a little boy (Jake) thinks
there's a zombie in his closet & his parents are fighting all the time.<br
/><br />This movie is slower than a soap opera... and suddenly, Jake
decides to become Rambo and kill the zombie.<br /><br />OK, first of all
when you're going to make a film you must Decide if its a thriller or a
drama! As a drama the movie is watchable. Parents are divorcing & arguing
like in real life. And then we have Jake with his closet which totally
ruins all the film! I expected to see a BOOGEYMAN similar movie, and
instead i watched a drama with some meaningless thriller spots.<br /><br
/>3 out of 10 just for the well playing parents & descent dialogs. As for
the shots with Jake: just ignore them."

doc5 = "Petter Mattei's 'Love in the Time of Money is a visually stunning
film to watch. Mr. Mattei offers us a vivid portrait about human
relations. This is a movie that seems to be telling us what money, power
and success do to people in the different situations we encounter. <br
/><br />This being a variation on the Arthur Schnitzler's play about the
same theme, the director transfers the action to the present time New York
where all these different characters meet and connect. Each one is
connected in one way, or another to the next person, but no one seems to
know the previous point of contact. Stylishly, the film has a
sophisticated luxurious look. We are taken to see how these people live
and the world they live in their own habitat.<br /><br />The only thing
one gets out of all these souls in the picture is the different stages of
loneliness each one inhabits. A big city is not exactly the best place in
which human relations find sincere fulfillment, as one discerns is the
case with most of the people we encounter.<br /><br />The acting is good
under Mr. Mattei's direction. Steve Buscemi, Rosario Dawson, Carol Kane,
Michael Imperioli, Adrian Grenier, and the rest of the talented cast, make
```

```
these characters come alive.<br /><br />We wish Mr. Mattei good luck and
await anxiously for his next work."
```

Tokenization des documents :

Nous avons divisé chaque document en mots individuels, en supprimant la ponctuation et en convertissant les mots en minuscules. Cela nous a donné une liste de mots pour chaque document.

```
def tokenize(text):
    words = text.split()
    words = [word.strip(".,!?'\"()[]").lower() for word in words]
    return words

# Tokenizez les 5 documents
documents = [doc1, doc2, doc3, doc4, doc5]
tokenized_docs = [tokenize(doc) for doc in documents]
```

Calcul du TF (Term Frequency) :

Nous avons calculé la fréquence de chaque terme (mot) dans chaque document, c'est-à-dire le nombre de fois où chaque terme apparaît dans le document.

```
def calculate_tf(tokens):
    tf_dict = {}
    total_words = len(tokens)
    for word in tokens:
        if word in tf_dict:
            tf_dict[word] += 1
        else:
            tf_dict[word] = 1
    for word, count in tf_dict.items():
        tf_dict[word] = count / total_words
    return tf_dict

tf_docs = [calculate_tf(tokens) for tokens in tokenized_docs]
```

Calcul de l'IDF (Inverse Document Frequency) :

Nous avons calculé l'inverse de la fréquence du terme dans l'ensemble du corpus (les deux documents). L'IDF est une mesure de l'importance du terme dans l'ensemble du corpus.

```
def calculate_idf(documents, term):
    num_documents_containing_term = sum(1 for doc in documents if term in
doc)
    if num_documents_containing_term == 0:
        return 0
    return len(documents) / num_documents_containing_term

idf_dict = {}
unique_terms = set(term for tokens in tokenized_docs for term in tokens)
for term in unique_terms:
    idf_dict[term] = calculate_idf(documents, term)
```

Calcul du poids TF-IDF (Term Frequency-Inverse Document Frequency) :

Nous avons multiplié le TF de chaque terme par son IDF correspondant pour obtenir le poids TF-IDF. Cela permet de donner un poids plus élevé aux termes rares et discriminants.

```
# Calculez le poids TF-IDF pour chaque document
def calculate_tfidf(tf, idf):
    tfidf_dict = {}
    for term in tf:
        tfidf_dict[term] = tf[term] * idf[term]
    return tfidf_dict

tfidf_docs = [calculate_tfidf(tf, idf_dict) for tf in tf_docs]
```

Création du fichier CSV :

Nous avons créé un fichier CSV pour stocker nos résultats. Les colonnes du fichier CSV comprennent les termes, le TF pour les deux documents, l'IDF, les poids TF-IDF pour les deux documents, le nombre total de mots dans chaque document, et le nombre d'apparitions de chaque terme dans chaque document.

```
with open('resultats.csv', 'w', newline='') as csvfile:
    fieldnames = ['Terme', 'TF - Document 1', 'TF - Document 2', 'TF -
Document 3', 'TF - Document 4', 'TF - Document 5',
                  'IDF', 'Poids TF-IDF - Document 1', 'Poids TF-IDF -
Document 2', 'Poids TF-IDF - Document 3',
                  'Poids TF-IDF - Document 4', 'Poids TF-IDF - Document
5', 'Nombre de Mots - Document 1',
                  'Nombre de Mots - Document 2', 'Nombre de Mots -
Document 3', 'Nombre de Mots - Document 4',
```

```
                    'Nombre de Mots - Document 5', 'Apparitions - Document
1', 'Apparitions - Document 2',
                    'Apparitions - Document 3', 'Apparitions - Document 4',
'Apparitions - Document 5']

    writer = csv.DictWriter(csvfile, fieldnames=fieldnames)

    # Écrivez l'en-tête du fichier CSV
    writer.writeheader()

    # Triez les colonnes de poids TF-IDF de chaque document
    for term in sorted(unique_terms, key=lambda term:
tfidf_docs[0].get(term, 0)):
        row = {'Terme': term}
        for i, tfidf_dict in enumerate(tfidf_docs):
            row[f'TF - Document {i + 1}'] = tf_docs[i].get(term, 0)
            row[f'Poids TF-IDF - Document {i + 1}'] = tfidf_dict.get(term,
0)
            row[f'Nombre de Mots - Document {i + 1}'] =
len(tokenized_docs[i])
            row[f'Apparitions - Document {i + 1}'] =
tokenized_docs[i].count(term)
        writer.writerow(row)
```

Le fichier CSV résultant contiendra les termes, les valeurs de TF, d'IDF, de poids TF-IDF, ainsi que des informations sur les documents. Vous pouvez adapter ce processus à l'ensemble de données IMDB complet ou à tout autre ensemble de documents que vous souhaitez analyser.

resultats.csv :

Terme	TF - Document 1	TF - Document 2	TF - Document 3	TF - Document 4	TF - Document 5	IDF	Poids TF-IDF -	Poids TF-IDF - Document 2	Poids TF-IDF -	Poids TF-IDF	Poids TF-IDF - Document 5	Nombre de Mots	Nombre de M	Nombre de Nc
meet	0	0	0	0	0.004347826		0	0	0	0	0.02173913	301	91	166
been	0	0	0.006024096	0	0		0	0	0.03012048	0	0	301	91	166
risk	0	0	0.006024096	0	0		0	0	0	0	0	301	91	166
wish	0	0	0	0	0.004347826		0	0	0	0	0.02173913	301	91	166
characters	0	0.010989011	0.006024096	0	0.008695652		0	0.018315018	0.01004016	0	0.014492754	301	91	166
different	0	0	0	0	0.013043478		0	0	0	0	0.065217391	301	91	166
arguing	0	0	0	0.007246377	0		0	0	0	0.0362319	0	301	91	166
state	0.003322259	0	0	0	0		0	0	0	0	0	301	91	166
rather	0	0.010989011	0	0	0		0	0.054945055	0	0	0	301	91	166
encounter.	0	0	0	0	0.004347826		0	0	0	0	0.02173913	301	91	166
weekend	0	0	0.006024096	0	0		0	0	0.03012048	0	0	301	91	166
rambo	0	0	0	0.007246377	0		0	0	0	0	0	301	91	166
woody	0	0	0.006024096	0	0		0	0	0	0	0	301	91	166
i've	0	0	0.006024096	0	0		0	0	0	0	0	301	91	166
new	0	0	0	0	0.004347826		0	0	0	0	0	301	91	166
offers	0	0	0	0	0.004347826		0	0	0	0	0.02173913	301	91	166
decides	0	0	0	0.007246377	0		0	0	0	0.0362319	0	301	91	166
trust	0.003322259	0	0	0	0		0	0	0	0	0	301	91	166
proof	0	0	0.006024096	0	0		0	0	0.03012048	0	0	301	91	166
thinks	0	0	0	0.007246377	0		0	0	0	0.0362319	0	301	91	166
bread	0	0	0.006024096	0	0		0	0	0.03012048	0	0	301	91	166
talented	0	0	0	0	0.004347826		0	0	0	0	0.02173913	301	91	166
2:00	0	0	0.006024096	0	0		0	0	0.03012048	0	0	301	91	166
security	0.003322259	0	0	0	0		0	0	0	0	0	301	91	166
like	0	0	0	0.007246377	0		0	0	0	0.0362319	0	301	91	166
jake	0	0	0	0.02173913	0		0	0	0	0	0	301	91	166
phil	0	0.010989011	0	0	0		0	0	0	0	0	301	91	166
imagine	0	0.010989011	0	0	0		0	0.054945055	0	0	0	301	91	166

En résumé, TF mesure l'importance d'un terme dans un document spécifique, tandis que IDF mesure l'importance d'un terme dans un ensemble de documents. Ensemble, ils forment le schéma TF-IDF qui est largement utilisé pour l'analyse de texte et la recherche d'information, aidant à identifier les termes clés et à classer les documents en fonction de leur pertinence.