Compte rendu de TP 9

Introduction

Dans ce compte rendu, nous allons présenter un système simple de détection de sentiments à partir de documents. Pour ce faire, nous disposons d'une base de référence contenant un corpus de 200 phrases réparties en deux classes de sentiments : 100 phrases pour les sentiments négatifs et 100 pour les sentiments positifs. La base de test est composée de 100 phrases, 50 pour chaque classe. L'objectif est de soumettre ces phrases à notre système qui déterminera si elles appartiennent à la classe positive ou négative.

Méthodologie du Système

Notre démarche pour la détection de sentiments comprend les étapes suivantes :

1. **Prétraitement des Données** : Tout d'abord, nous commençons par la lecture des fichiers de la base de référence. Ensuite, nous appliquons un ensemble de techniques de prétraitement, notamment :

    - Mise en minuscules : Conversion de tous les textes en minuscules pour une cohérence.
    - Nettoyage de la ponctuation : Suppression de la ponctuation pour éviter tout bruit inutile.
    - Élimination des mots vides : Retrait des mots courants (stop words) qui n'apportent pas d'informations significatives.
    - Stemming : Réduction des mots à leur forme racine pour normaliser le texte.
2. **Vectorisation des Phrases** : Pour chaque phrase de la base de référence, nous effectuons la vectorisation en calculant le TF-IDF (Term Frequency-Inverse Document Frequency) de chaque mot. Cela nous permet de représenter chaque phrase sous forme de vecteur.

3. **Prétraitement et Vectorisation des Documents de Test** : Nous appliquons les mêmes étapes de prétraitement et de vectorisation aux documents de test, de manière à les préparer pour la phase de classification.

4. **Calcul des Distances** : Nous calculons les distances entre les vecteurs des documents de test et les vecteurs de référence des classes positives et négatives. Nous utilisons deux méthodes de calcul de distance :

    - Méthode de Cosinus : Nous mesurons la similarité cosinus entre les vecteurs, prenant en compte la proximité à 1. En fonction de cette similarité, nous prenons une décision quant à l'appartenance de la phrase à la classe positive ou négative.

- Méthode de Bray-Curtis : Dans cette méthode, nous calculons la distance de Bray-Curtis et évaluons la proximité à 0. En fonction de cette distance, nous déterminons si la phrase appartient à la classe positive ou négative.

5. **Évaluation du Système et Comparaison des Méthodes** : Après avoir effectué la classification, nous passons à l'évaluation de la performance de notre système. Nous utilisons des mesures d'évaluation telles que l'exactitude, la précision, le rappel et la F-mesure pour quantifier son efficacité dans la classification des sentiments. De plus, nous comparons les résultats des deux méthodes de calcul de distance pour déterminer laquelle est la plus performante.

Cette méthodologie nous permet de réaliser une classification de sentiments basique en se basant sur des techniques de traitement de texte et de vectorisation. Elle peut servir de point de départ pour des tâches de détection de sentiments plus avancées.

En suivant ces étapes, notre système est capable de classer les phrases de test en fonction de leur sentiment, en utilisant les vecteurs et les distances calculées.

Cette méthodologie nous permet de réaliser une classification de sentiments basique en se basant sur des techniques de traitement de texte et de vectorisation. Elle peut servir de point de départ pour des tâches de détection de sentiments plus avancées.

Mise en Place des Outils Essentiels

Dans cette section, nous allons implementer les outils et les bibliothèques nécessaires pour la mise en place de notre système de détection de sentiments. Avant de plonger dans les détails de l'implémentation, assurons-nous d'avoir tous les éléments nécessaires.

Implementation de la fonction du prétraitement

In [ ]:
```
import re, nltk
from nltk import word_tokenize
from nltk.corpus import stopwords
from nltk.stem import PorterStemmer as ps
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()

def lemmatization(tokens):
    return [lemmatizer.lemmatize(word) for word in tokens]

stemmer = ps()

def clean_words(text) :
    # remove html markup
    text = re.sub("(<.*?.>)", "", text)
    # remove non-ascii and digits
    text = re.sub(r'[^\w\s]', "", text)
    return text
```

```
# nltk.download('wordnet')

def tokenisation(cont) :
  stop_words = set(stopwords.words('english'))
  tokens = word_tokenize(cont)
  tokens = [word for word in tokens if not word in stop_words]
  return tokens


def stemming(tokens) :

  return [stemmer.stem(word = word) for word in tokens]

def pretraitement(doc) :
  doc = doc.lower()
  doc = clean_words(doc)
  tokens = tokenisation(doc)
  # stems = lemmatization(tokens)
  stems = stemming(tokens)

  return stems
```

Implementation de la fonction TF-IDF

In [ ]:
```
import math

def TFIDF(treated_doc, doc_collection):
    TFIDF = []
    # on faire le traitement pour chaque mot dans notre document
    for word in treated_doc :
    # Calcul de la fréquence du terme (TF) dans le document
      TF = treated_doc.count(word) / len(treated_doc)

      # Calcul de la fréquence inverse de document (IDF) :
      N = len(doc_collection)  # Nombre total de documents dans la collection
      DF = 0  # Initialise le décompte de documents contenant le terme

      # Parcours de la collection de documents pour compter le nombre de documen
      for doc in doc_collection:
          if word in doc:
              DF += 1

      # Calcul de l'IDF en utilisant la formule standard : log(N / DF)
      IDF = math.log(N / DF)  # Note : Vous pouvez envisager d'ajouter 1 au déno

      # Calcul du score final TF-IDF en multipliant TF par IDF
      TFIDF.append(TF * IDF)

    # en retoure un vecteur des TF-IDF de chaque mot de la document
    return TFIDF
```

Implementation de la fonction Cosinus

In [ ]:
```
import math
```

```python
def simCosine(v1, v2):
    """
    Calcule la similarité cosinus entre deux vecteurs.

    Cette fonction calcule la similarité cosinus entre deux vecteurs d'entrée en
    en calculant le produit scalaire (S1) et la magnitude de chaque vecteur (S2

    Args:
        v1: Le premier vecteur.
        v2: Le deuxième vecteur.

    Returns:
        La similarité cosinus entre les deux vecteurs d'entrée.
    """

    # Redimensionner les vecteurs pour qu'ils aient la même longueur
    mini = v2 if len(v1) > len(v2) else v1
    maxi = v1 if len(v1) > len(v2) else v2
    N = len(maxi)

    S1 = 0  # Initialisation du produit scalaire
    S2 = 0  # Initialisation de la magnitude de v1
    S3 = 0  # Initialisation de la magnitude de v2

    # Compléter le vecteur le plus court avec des zéros
    for i in range(N - len(mini)):
        mini.append(0)

    # Calcul du produit scalaire S1 et des magnitudes S2 et S3
    for i in range(N):
        S1 += maxi[i] * mini[i]
        S2 += maxi[i] ** 2
        S3 += mini[i] ** 2

    # Calcul de la similarité cosinus
    similarity = S1 / (math.sqrt(S2) * math.sqrt(S3))

    return similarity
```

Implementation de la fonciton Bray-Curtis

In [ ]:

```python
def simBray(v1, v2):
    """
    Calcule la distance de Bray-Curtis entre deux vecteurs.r

    Cette fonction calcule la distance de Bray-Curtis entre deux vecteurs d'entr
    puis en effectuant des calculs sur les composants des vecteurs et renvoie la

    Args:
        v1: Le premier vecteur.
        v2: Le deuxième vecteur.

    Returns:
        La distance de Bray-Curtis entre les deux vecteurs d'entrée.
    """

    # Redimensionner les vecteurs pour qu'ils aient la même longueur
    mini = v2 if len(v1) > len(v2) else v1
```

```
    maxi = v1 if len(v1) > len(v2) else v2
    N = len(maxi)

    S1 = 0  # Initialisation du numérateur
    S2 = 0  # Initialisation du dénominateur

    # Compléter le vecteur le plus court avec des zéros
    for i in range(N - len(mini)):
        mini.append(0)

    # Calcul du numérateur S1 et du dénominateur S2
    for i in range(N):
        S1 += min(maxi[i], mini[i])
        S2 += maxi[i] + mini[i]

    # Calcul de la distance de Bray-Curtis
    bray_curtis_distance = 2 * S1 / S2

    return bray_curtis_distance
```

Implementation du Système

Lecture des documents de référence

Pour le document de référence positive :

In [ ]:
```
import pandas as pd
pd.set_option('display.max_rows', 10)
# On intialise le tableau pour stocker les documents ligne par ligne
pos_doc = []
for line in open("BR_Sentiment_Positif.txt", "r") :
  pos_doc.append(line)

pds = pd.DataFrame({"Phrase" : pos_doc})
pds
```

Out[ ]:

	Phrase
0	This quiet , introspective and entertaining in...
1	quiet , introspective and entertaining indepen...
2	entertaining\n
3	is worth seeking\n
4	A positively thrilling combination of ethnogra...
...	...
95	An unsettling , memorable cinematic experience...
96	, memorable cinematic experience\n
97	does its predecessors proud\n
98	proud\n
99	it 's a pretty good execution of a story that ...

100 rows × 1 columns

Pour le document de référence negative :

In [ ]:
```
neg_doc = []
for line in open("BR_Sentiment_Negatif.txt", "r") :
  neg_doc.append(line)

pds = pd.DataFrame({"Phrase" : neg_doc})
pds
```

Out[ ]:

	Phrase
0	would have a hard time sitting through this one\n
1	have a hard time sitting through this one\n
2	Aggressive self-glorification and a manipulati...
3	self-glorification and a manipulative whitewash\n
4	Trouble Every Day is a plodding mess .\n
...	...
95	ugly to look at and not a Hollywood product\n
96	bogged down in earnest dramaturgy\n
97	cheap\n
98	Violent , vulgar and forgettably\n
99	vulgar\n

100 rows × 1 columns

Prétraitement des documents de référence

Pour le document de référence positive :

In [ ]:
```
treated_pos_doc = [pretraitement(line) for line in pos_doc]
pds = pd.DataFrame({"Phrase": pos_doc, "Phrase apres traitement" : treated_pos_d
pds
```

Out[ ]:

	Phrase	Phrase apres traitement
0	This quiet , introspective and entertaining in...	[quiet, introspect, entertain, independ, worth...
1	quiet , introspective and entertaining indepen...	[quiet, introspect, entertain, independ]
2	entertaining\n	[entertain]
3	is worth seeking\n	[worth, seek]
4	A positively thrilling combination of ethnogra...	[posit, thrill, combin, ethnographi, intrigu, ...
...	...	...
95	An unsettling , memorable cinematic experience...	[unsettl, memor, cinemat, experi]
96	, memorable cinematic experience\n	[memor, cinemat, experi]
97	does its predecessors proud\n	[predecessor, proud]
98	proud\n	[proud]
99	it 's a pretty good execution of a story that ...	[pretti, good, execut, stori, lot, richer, one...

100 rows × 2 columns

Pour le document de référence negative :

In [ ]:
```
treated_neg_doc = [pretraitement(line) for line in neg_doc]
pds = pd.DataFrame({"Phrase": neg_doc, "Phrase apres traitement" : treated_neg_d
pds
```

Out[ ]:

	Phrase	Phrase apres traitement
0	would have a hard time sitting through this one\n	[would, hard, time, sit, one]
1	have a hard time sitting through this one\n	[hard, time, sit, one]
2	Aggressive self-glorification and a manipulati...	[aggress, selfglorif, manipul, whitewash]
3	self-glorification and a manipulative whitewash\n	[selfglorif, manipul, whitewash]
4	Trouble Every Day is a plodding mess .\n	[troubl, everi, day, plod, mess]
...	...	...
95	ugly to look at and not a Hollywood product\n	[ugli, look, hollywood, product]
96	bogged down in earnest dramaturgy\n	[bog, earnest, dramaturgi]
97	cheap\n	[cheap]
98	Violent , vulgar and forgettably\n	[violent, vulgar, forgett]
99	vulgar\n	[vulgar]

100 rows × 2 columns

Calcule de TF-IDF des document de référence

Pour le document de référence positive

In [ ]:
```
TFIDF_pos = []
i = 0
for line in treated_pos_doc:
  TFIDF_pos.append(TFIDF(line, treated_pos_doc + treated_neg_doc))

pds = pd.DataFrame({"Phrase": pos_doc, "Phrase apres traitement" : treated_pos_d
pds
```

Out[ ]:

	Phrase	Phrase apres traitement	TF-IDF
0	This quiet , introspective and entertaining in...	[quiet, introspect, entertain, independ, worth...	[0.7675283643313486, 0.7675283643313486, 0.652...
1	quiet , introspective and entertaining indepen...	[quiet, introspect, entertain, independ]	[1.151292546497023, 1.151292546497023, 0.97800...
2	entertaining\n	[entertain]	[3.912023005428146]
3	is worth seeking\n	[worth, seek]	[1.753278948659991, 2.302585092994046]
4	A positively thrilling combination of ethnogra...	[posit, thrill, combin, ethnographi, intrigu, ...	[0.35424386046062245, 0.3230542367599944, 0.35...
...	...	...	...
95	An unsettling , memorable cinematic experience...	[unsettl, memor, cinemat, experi]	[1.151292546497023, 0.8381018043731808, 1.0499...
96	, memorable cinematic experience\n	[memor, cinemat, experi]	[1.1174690724975744, 1.3999016926266423, 1.399...
97	does its predecessors proud\n	[predecessor, proud]	[2.302585092994046, 2.0998525389399636]
98	proud\n	[proud]	[4.199705077879927]
99	it 's a pretty good execution of a story that ...	[pretti, good, execut, stori, lot, richer, one...	[0.44152644721233636, 0.3499754231566606, 0.44...

100 rows × 3 columns

Pour le document de référence negative :

In [ ]:
```
TFIDF_neg = []

for line in treated_neg_doc:
  TFIDF_neg.append(TFIDF(line, treated_pos_doc + treated_neg_doc))



pds = pd.DataFrame({"Phrase": neg_doc, "Phrase apres traitement" : treated_neg_d
pds
```

Out[ ]:

	Phrase	Phrase apres traitement	TF-IDF
0	would have a hard time sitting through this one\n	[would, hard, time, sit, one]	[0.9210340371976184, 0.8399410155759854, 0.921...
1	have a hard time sitting through this one\n	[hard, time, sit, one]	[1.0499262694699818, 1.151292546497023, 1.1512...
2	Aggressive self-glorification and a manipulati...	[aggress, selfglorif, manipul, whitewash]	[1.324579341637009, 1.151292546497023, 1.15129...
3	self-glorification and a manipulative whitewash\n	[selfglorif, manipul, whitewash]	[1.5350567286626973, 1.5350567286626973, 1.535...
4	Trouble Every Day is a plodding mess .\n	[troubl, everi, day, plod, mess]	[1.0596634733096073, 1.0596634733096073, 1.059...
...	...	...	...
95	ugly to look at and not a Hollywood product\n	[ugli, look, hollywood, product]	[0.683342002271625, 0.683342002271625, 0.66481...
96	bogged down in earnest dramaturgy\n	[bog, earnest, dramaturgi]	[1.7661057888493454, 1.7661057888493454, 1.766...
97	cheap\n	[cheap]	[5.298317366548036]
98	Violent , vulgar and forgettably\n	[violent, vulgar, forgett]	[1.7661057888493454, 1.5350567286626973, 1.766...
99	vulgar\n	[vulgar]	[4.605170185988092]

100 rows × 3 columns

Lecture des documemts de test

Document de test positive :

In [ ]:
```
pos_test = []
for line in open("Test_Sentiment_Positif.txt", "r") :
  pos_test.append(line)

pds = pd.DataFrame({"Phrases" : pos_test})
pd.set_option('display.max_rows', 20)
pds
```

Out[ ]:

	Phrases
0	A smart , provocative drama that does the near...
1	A smart , provocative drama that does the near...
2	A smart , provocative drama that does the near...
3	A smart , provocative drama that does the near...
4	smart , provocative drama\n
...	...
45	a particularly good film\n
46	particularly good film\n
47	particularly good\n
48	surprisingly ` solid ' achievement\n
49	One of recent memory 's most thoughtful films ...

50 rows × 1 columns

Document de test negative :

In [ ]:

```
neg_test = []
for line in open("Test_Sentiment_Negatif.txt", "r") :
  neg_test.append(line)

pds = pd.DataFrame({"Phrases" : neg_test})
# pd.set_option('display.max_rows', 20)
pds
```

Out[ ]:

	Phrases
0	is trying to dupe the viewer into taking it al...
1	trying to dupe the viewer into taking it all a...
2	to dupe the viewer into taking it all as Very ...
3	dupe the viewer into taking it all as Very Imp...
4	simply because the movie is ugly to look at an...
...	...
45	an utterly incompetent conclusion\n
46	utterly incompetent conclusion\n
47	utterly incompetent\n
48	Horrendously amateurish filmmaking that is pla...
49	filmmaking that is plainly dull and visually u...

50 rows × 1 columns

Prétraitement des documents de test

Pour le document de test positive :

In [ ]:
```
treated_pos_test = [pretraitement(doc) for doc in pos_test]
pds = pd.DataFrame({"Phrase" : pos_test, "Phrase apres traitement" : treated_pos
pds
```

Out[ ]:

	Phrase	Phrase apres traitement
0	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs, get, sk...
1	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs, get, sk...
2	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs]
3	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs]
4	smart , provocative drama\n	[smart, provoc, drama]
...	...	...
45	a particularly good film\n	[particularli, good, film]
46	particularly good film\n	[particularli, good, film]
47	particularly good\n	[particularli, good]
48	surprisingly ` solid ' achievement\n	[surprisingli, solid, achiev]
49	One of recent memory 's most thoughtful films ...	[one, recent, memori, thought, film, art, ethi...

50 rows × 2 columns

Pour le document de test negative :

In [ ]:
```
treated_neg_test = [pretraitement(doc) for doc in neg_test]
pds = pd.DataFrame({"Phrase" : neg_test, "Phrase apres traitement" : treated_neg
pds
```

Out[ ]:

	Phrase	Phrase apres traitement
0	is trying to dupe the viewer into taking it al...	[tri, dupe, viewer, take, import, simpli, movi...
1	trying to dupe the viewer into taking it all a...	[tri, dupe, viewer, take, import, simpli, movi...
2	to dupe the viewer into taking it all as Very ...	[dupe, viewer, take, import, simpli, movi, ugl...
3	dupe the viewer into taking it all as Very Imp...	[dupe, viewer, take, import, simpli, movi, ugl...
4	simply because the movie is ugly to look at an...	[simpli, movi, ugli, look, hollywood, product]
...	...	...
45	an utterly incompetent conclusion\n	[utterli, incompet, conclus]
46	utterly incompetent conclusion\n	[utterli, incompet, conclus]
47	utterly incompetent\n	[utterli, incompet]
48	Horrendously amateurish filmmaking that is pla...	[horrend, amateurish, filmmak, plainli, dull, ...
49	filmmaking that is plainly dull and visually u...	[filmmak, plainli, dull, visual, ugli, nt, inc...

50 rows × 2 columns

Calcule de TF-IDF des document de test

Pour le document de test positive :

In [ ]:
```
TFIDf_test_pos = []
corpus = treated_neg_test + treated_pos_test

for line in treated_pos_test :
  TFIDf_test_pos.append(TFIDF(line, corpus))

pds = pd.DataFrame({"Phrase" : pos_test, "Phrase apres traitement" : treated_pos
pds
```

Out[ ]:

	Phrase	Phrase apres traitement	TF-IDF
0	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs, get, sk...	[0.24964435612949923, 0.23445089306333636, 0.2...
1	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs, get, sk...	[0.24964435612949923, 0.23445089306333636, 0.2...
2	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs]	[0.5991464547107982, 0.5626821433520073, 0.562...
3	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs]	[0.5991464547107982, 0.5626821433520073, 0.562...
4	smart , provocative drama\n	[smart, provoc, drama]	[0.9985774245179969, 0.9378035722533454, 0.937...
...	...	...	...
45	a particularly good film\n	[particularli, good, film]	[1.168852632439994, 0.9985774245179969, 1.1688...
46	particularly good film\n	[particularli, good, film]	[1.168852632439994, 0.9985774245179969, 1.1688...
47	particularly good\n	[particularli, good]	[1.753278948659991, 1.4978661367769954]
48	surprisingly ` solid ' achievement\n	[surprisingli, solid, achiev]	[1.5350567286626973, 1.5350567286626973, 1.535...
49	One of recent memory 's most thoughtful films ...	[one, recent, memori, thought, film, art, ethi...	[0.2995732273553991, 0.4605170185988092, 0.460...

50 rows × 3 columns

Pour le document de test negative :

In [ ]:
```
TFIDf_test_neg = []
for line in treated_neg_test :
  TFIDf_test_neg.append(TFIDF(line, corpus))

pds = pd.DataFrame({"Phrase" : pos_test, "Phrase apres traitement" : treated_pos
pds
```

Out[ ]:

	Phrase	Phrase apres traitement	TF-IDF
0	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs, get, sk...	[0.24964435612949923, 0.23445089306333636, 0.2...
1	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs, get, sk...	[0.24964435612949923, 0.23445089306333636, 0.2...
2	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs]	[0.5991464547107982, 0.5626821433520073, 0.562...
3	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs]	[0.5991464547107982, 0.5626821433520073, 0.562...
4	smart , provocative drama\n	[smart, provoc, drama]	[0.9985774245179969, 0.9378035722533454, 0.937...
...	...	...	...
45	a particularly good film\n	[particularli, good, film]	[1.168852632439994, 0.9985774245179969, 1.1688...
46	particularly good film\n	[particularli, good, film]	[1.168852632439994, 0.9985774245179969, 1.1688...
47	particularly good\n	[particularli, good]	[1.753278948659991, 1.4978661367769954]
48	surprisingly ` solid ' achievement\n	[surprisingli, solid, achiev]	[1.5350567286626973, 1.5350567286626973, 1.535...
49	One of recent memory 's most thoughtful films ...	[one, recent, memori, thought, film, art, ethi...	[0.2995732273553991, 0.4605170185988092, 0.460...

50 rows × 3 columns

Calcule des distances

Par la methode de Cosinus :

Pour le document de test positive :

In [ ]:
```
similarite_pos_cos = []  # Liste pour stocker les résultats de classification (p
Spp = []  # Liste pour stocker les similarités maximales (cosinus) pour la class
Snn = []  # Liste pour stocker les similarités maximales (cosinus) pour la class

for doc in TFIDf_test_pos:  # Parcourir les vecteurs TF-IDF des documents de tes
    Sp = []  # Liste pour stocker les similarités cosinus pour la classe positiv
    Sn = []  # Liste pour stocker les similarités cosinus pour la classe négativ
    for refp in TFIDF_pos:  # Parcourir les vecteurs TF-IDF des documents de réf
        Sp.append(simCosine(doc, refp))  # Calculer la similarité cosinus et l'a
    for refn in TFIDF_neg:  # Parcourir les vecteurs TF-IDF des documents de réf
        Sn.append(simCosine(doc, refn))  # Calculer la similarité cosinus et l'a
    Spp.append(max(Sp))  # Stocker la similarité cosinus maximale pour la classe
    Snn.append(max(Sn))  # Stocker la similarité cosinus maximale pour la classe

    # Classer le document comme "positive" si la similarité maximale pour la cla
```

```
    # ou égale à la similarité maximale pour la classe négative ; sinon, le clas
    similarite_pos_cos.append("positive" if max(Sp) >= max(Sn) else "negative")

# Créer un DataFrame pour afficher les résultats, y compris les phrases d'origin
# les distances maximales pour la classe positive et négative, et la décision fi
pds = pd.DataFrame({
    "Phrase": pos_test,
    "Phrase traite": treated_pos_test,
    "Distance de pos": Spp,
    "Distance de neg": Snn,
    "Décision": similarite_pos_cos
})

pds
```

Out[ ]:

	Phrase	Phrase traite	Distance de pos	Distance de neg	Décision
0	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs, get, sk...	0.999108	0.991990	positive
1	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs, get, sk...	0.999108	0.991990	positive
2	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs]	0.997127	0.999533	negative
3	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs]	0.997127	0.999533	negative
4	smart , provocative drama\n	[smart, provoc, drama]	0.999553	0.999553	positive
...	...	...	...	...	...
45	a particularly good film\n	[particularli, good, film]	0.997405	0.999971	negative
46	particularly good film\n	[particularli, good, film]	0.997405	0.999971	negative
47	particularly good\n	[particularli, good]	0.999530	0.999717	negative
48	surprisingly ` solid ' achievement\n	[surprisingli, solid, achiev]	1.000000	1.000000	positive
49	One of recent memory 's most thoughtful films ...	[one, recent, memori, thought, film, art, ethi...	0.985267	0.984947	positive

50 rows × 5 columns

Pour le document de test negative :

In [ ]:

```
similarite_neg_cos = []
```

```
Spp = []
Snn = []

for doc in TFIDf_test_neg :
  Sp = []
  Sn = []
  for refp in TFIDF_pos :
    Sp.append(simCosine(doc, refp))
  for refn in TFIDF_neg :
    Sn.append(simCosine(doc, refn))
  Spp.append(max(Sp))
  Snn.append(max(Sn))
  similarite_neg_cos.append("positive" if max(Sp) > max(Sn)  else "negative")

pds = pd.DataFrame({"Phrase" : neg_test,"Phrase traite" : treated_neg_test, "Dis

pds
```

Out[ ]:

	Phrase	Phrase traite	Distance de pos	Distance de neg	Decision
0	is trying to dupe the viewer into taking it al...	[tri, dupe, viewer, take, import, simpli, movi...	0.993949	0.992385	positive
1	trying to dupe the viewer into taking it all a...	[tri, dupe, viewer, take, import, simpli, movi...	0.993949	0.992385	positive
2	to dupe the viewer into taking it all as Very ...	[dupe, viewer, take, import, simpli, movi, ugl...	0.992903	0.995290	negative
3	dupe the viewer into taking it all as Very Imp...	[dupe, viewer, take, import, simpli, movi, ugl...	0.992903	0.995290	negative
4	simply because the movie is ugly to look at an...	[simpli, movi, ugli, look, hollywood, product]	0.993870	0.994482	negative
...	...	...	...	...	...
45	an utterly incompetent conclusion\n	[utterli, incompet, conclus]	0.999700	0.999700	negative
46	utterly incompetent conclusion\n	[utterli, incompet, conclus]	0.999700	0.999700	negative
47	utterly incompetent\n	[utterli, incompet]	1.000000	1.000000	negative
48	Horrendously amateurish filmmaking that is pla...	[horrend, amateurish, filmmak, plainli, dull, ...	0.939806	0.987022	negative
49	filmmaking that is plainly dull and visually u...	[filmmak, plainli, dull, visual, ugli, nt, inc...	0.995345	0.991152	positive

50 rows × 5 columns

Par la methode de Bray-Cortis :

Pour le document de test positive :

In [ ]:
```
similarite_pos_bray = []
Spp = []
Snn = []

for doc in TFIDf_test_pos :
  Sp = []
  Sn = []
  for refp in TFIDF_pos :
    Sp.append(simBray(doc, refp))
  for refn in TFIDF_neg :
    Sn.append(simBray(doc, refn))
  Spp.append(min(Sp))
  Snn.append(min(Sn))
  similarite_pos_bray.append("positive" if min(Sp) <= min(Sn)  else "negative")

pds = pd.DataFrame({"Phrase" : pos_test,"Phrase traite" : treated_pos_test, "Dis

pds
```

Out[ ]:

	Phrase	Phrase traite	Distance de pos	Distance de neg	Decision
0	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs, get, sk...	0.056749	0.056749	positive
1	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs, get, sk...	0.056749	0.056749	positive
2	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs]	0.145758	0.145758	positive
3	A smart , provocative drama that does the near...	[smart, provoc, drama, nearli, imposs]	0.145758	0.145758	positive
4	smart , provocative drama\n	[smart, provoc, drama]	0.244375	0.136868	negative
...	...	...	...	...	...
45	a particularly good film\n	[particularli, good, film]	0.230043	0.129383	negative
46	particularly good film\n	[particularli, good, film]	0.230043	0.129383	negative
47	particularly good\n	[particularli, good]	0.143247	0.082943	negative
48	surprisingly ` solid ' achievement\n	[surprisingli, solid, achiev]	0.196422	0.112489	negative
49	One of recent memory 's most thoughtful films ...	[one, recent, memori, thought, film, art, ethi...	0.062199	0.062199	positive

50 rows × 5 columns

Pour le document de test negative :

In [ ]:

```
similarite_neg_bray = []

Spp = []
Snn = []

for doc in TFIDf_test_neg :
  Sp = []
  Sn = []
  for refp in TFIDF_pos :
    Sp.append(simBray(doc, refp))
  for refn in TFIDF_neg :
    Sn.append(simBray(doc, refn))
  Spp.append(min(Sp))
  Snn.append(min(Sn))
  similarite_neg_bray.append("positive" if min(Sp) < min(Sn)  else "negative")

pds = pd.DataFrame({"Phrase" : neg_test,"Phrase traite" : treated_neg_test, "Dis
```

```
pds
```

Out[ ]:

	Phrase	Phrase traite	Distance de pos	Distance de neg	Decision
0	is trying to dupe the viewer into taking it al...	[tri, dupe, viewer, take, import, simpli, movi...	0.086981	0.086981	negative
1	trying to dupe the viewer into taking it all a...	[tri, dupe, viewer, take, import, simpli, movi...	0.086981	0.086981	negative
2	to dupe the viewer into taking it all as Very ...	[dupe, viewer, take, import, simpli, movi, ugl...	0.079733	0.079733	negative
3	dupe the viewer into taking it all as Very Imp...	[dupe, viewer, take, import, simpli, movi, ugl...	0.079733	0.079733	negative
4	simply because the movie is ugly to look at an...	[simpli, movi, ugli, look, hollywood, product]	0.128372	0.128372	negative
...	...	...	...	...	...
45	an utterly incompetent conclusion\n	[utterli, incompet, conclus]	0.213993	0.142283	negative
46	utterly incompetent conclusion\n	[utterli, incompet, conclus]	0.213993	0.142283	negative
47	utterly incompetent\n	[utterli, incompet]	0.160271	0.090820	negative
48	Horrendously amateurish filmmaking that is pla...	[horrend, amateurish, filmmak, plainli, dull, ...	0.112370	0.112370	negative
49	filmmaking that is plainly dull and visually u...	[filmmak, plainli, dull, visual, ugli, nt, inc...	0.125875	0.125875	negative

50 rows × 5 columns

Evaluation de système

Maintenant on va calculer les mesures d'evaluation et comparer les deux methodes

In [ ]:

```
# Calcule des mesures pour la methode de Cosinuns

TP_cos = similarite_pos_cos.count("positive")
FP_cos = similarite_neg_cos.count("positive")
TN_cos = similarite_neg_cos.count("negative")
FN_cos = similarite_pos_cos.count("negative")



ac_cos = (TP_cos + TN_cos ) / (TP_cos + TN_cos + FN_cos + FP_cos)
prec_cos = TP_cos / (TP_cos + FP_cos)
rec_cos = TP_cos / (TP_cos + FN_cos)
fmesure_cos = 2 * (prec_cos * rec_cos) / (prec_cos + rec_cos)
```

```
# Calcule des mesures pour la methode de Bray-Cortis


TP_bray = similarite_pos_bray.count("positive")
FP_bray = similarite_neg_bray.count("positive")
TN_bray = similarite_neg_bray.count("negative")
FN_bray = similarite_pos_bray.count("negative")



ac_bray = (TP_bray + TN_bray ) / (TP_bray + TN_bray + FN_bray + FP_bray)
prec_bray = TP_bray / (TP_bray + FP_bray)
rec_bray = TP_bray / (TP_bray + FN_bray)
fmesure_bray = 2 * (prec_bray * rec_bray) / (prec_bray + rec_bray)

data = {
    "Accuracy": [ac_cos * 100, ac_bray * 100 ],
    "Precision": [prec_cos * 100, prec_bray * 100],
    "Recall": [rec_cos * 100, rec_bray * 100],
    "F-mesure": [fmesure_cos * 100, fmesure_bray * 100],
}

# Create the DataFrame
pds = pd.DataFrame(data, index=["Cosinuns", "Bray-Cortis"])

pds
```

Out[ ]:

	Accuracy	Precision	Recall	F-mesure
Cosinuns	69.0	77.142857	54.0	63.529412
Bray-Cortis	73.0	100.000000	46.0	63.013699