Marius
Casamian
(cm205773)

Introduction à l'Intelligence Artificielle Licence 2 — Portail Sciences et Technologies

Travaux dirigés No 5 : Règles d'association

Objectif du TP:

- ☐ Algorithme Apriori pour trouver règles d'associations
- ☐ Utilisation du langage Python
- ☐ Rédiger un compte rendu de TP

Réalisation du TP:
(Code source disponible dans le fichier zippé, avec le code python commenter)

Après avoir importé les librairies nécessaires:

```
import time
from apyori import apriori
import re
import sys
```

On lit le fichier contenant le jeu de données à traiter grâce à python:

f = open(r'corpus.txt', encoding="UTF-8")
On code ensuite la fonction normalize pour extraire les mots du texte proprement, en éliminant les espaces, etc.
Ensuite grâce au code suivant on obtient une liste de transactions contenant des mots qui apparaissent dans le titre d'une publication:

```
transactions = []
for line in f:
  if len(line) > 1 and not line.startswith("#"):
    words = normalize(line).split()
    items = set()
    for word in words:
      if word not in stopwords:
        items.add(word)
    if len(items) != 0:
      transactions.append(sorted(items))
print(transactions)
```

Pour la dernière partie on décide de créer une liste nommée **liste_params** qui contient les paramètres pour construire un modèle **apriori**, à savoir, **min_support** qui correspond au seuil minimum pour qu'une règle soit considérée comme valide, **min_confidence** qui est le seuil de confiance minimal, **min lift** qui est une mesure qui compare la confiance d'une règle à ce qu'elle serait si les deux ensembles d'articles étaient indépendant et enfin **min_length** correspond au nombre minimum d'articles dans une règle d'association.

Ensuite je vais itérer sur cette liste afin de créer un modèle apriori à partir de différents paramètres. Cela me permet de tester différents modèles et d'observer les différents temps d'exécution du modèle car par conséquent chaque exécution du programme utilisera un modèle apriori différent, basé sur la même liste de valeurs mais pas avec les mêmes données en paramètres.

Enfin, je redirige la sortie du programme dans un fichier nommé **résultat_apriori.txt** pour voir le résultat plus clairement. A noter que chaque exécution écrase le fichier, voici un exemple du contenu du fichier:

```
On utilise les paramètres suivants:
min_support=0.01, min_confidence=0.7, min_lift=1.2, min_length=3

Temps d'exécution est de: 6.198883056640625e-06s

Résultats d'exécution:

RelationRecord(items=frozenset({'Mining', 'Argument'}), support=0.01104417670682731,
ordered_statistics=[OrderedStatistic(items_base=frozenset({'Argument'}),
items_add=frozenset({'Mining'}), confidence=0.8461538461538463,
lift=32.414201183431956)])
…
…
(cf code python pour mieux comprendre le résultat généré)
```

Concernant le résultat de l'exécution:
RelationRecord indique qu'il y a une forte association entre "Consensus" et "Clustering" dans le jeu de données. Lorsque "Consensus" est présent, "Clustering" est toujours présent, ce qui est reflété par une confiance de 1.0. De plus, le lift élevé suggère une forte association entre ces deux articles.

On remarque que des valeurs plus élevées pour ces paramètres ont tendance à accélérer l'exécution de l'algorithme Apriori en réduisant le nombre de résultats générés. Cependant, cela peut également réduire la sensibilité de l'algorithme pour détecter des

associations moins évidentes. Le choix des valeurs dépend des objectifs et de la taille des données.