Sentiment Analysis, Recommender System and Association Rule Mining

Unit 4: Session 04

By Md. Meftaul Haque Mishu

Part 01

Sentiment Analysis

What is Sentiment Analysis?

- The process of computationally identifying and categorizing opinions expressed in a piece of text or document.
- Determine whether the writer's attitude towards a particular topic, product, etc. is positive, negative, or neutral.

Why Sentiment Analysis?

- **Business**: Understand customer opinions about products and services.
- **Politics**: Understand public opinion on policies, candidates, etc.
- **Public Actions**: Understand public opinion on social issues.
- **Brand Monitoring**: Understand how people feel about a brand.
- **Market Research**: Understand customer needs and preferences.

Types of Sentiments

1. **Positive**: Indicates a positive attitude or opinion.
2. **Negative**: Indicates a negative attitude or opinion.
3. **Neutral**: Indicates a neutral or unbiased opinion.

Sentiment Analysis Tools and Libraries

1. **Natural Language Toolkit (NLTK)**: Python library for text processing.
2. **VADER Sentiment Analysis**: Pre-built sentiment analysis tool.
3. **TensorFlow and PyTorch**: Deep learning frameworks for building custom models.

Steps of Sentiment Analysis

1. **Data Collection**: Collect data from various sources.
2. **Data Preprocessing**: Clean and preprocess the data.
    - Lowercasing
    - Tokenization
    - Remove stop words
    - Stemming or Lemmatization
    - Remove special characters and punctuations
    - Remove / Convert numbers

Steps of Sentiment Analysis

3. **Feature Extraction**: Convert text data into numerical data. Also known as Text vectorization.
    - Bag of Words
    - TF-IDF
    - Word Embeddings

Steps of Sentiment Analysis

4. **Model Selection**: Choose a model for sentiment analysis.
    - Naive Bayes
    - Support Vector Machine (SVM)
    - Recurrent Neural Network (RNN)
    - Long Short-Term Memory (LSTM)
    - Transformer
5. **Model Training**: Train the selected model using the training data.
6. **Model Evaluation**: Evaluate the model using the test data.
7. **Model Deployment**: Deploy the model for real-time sentiment analysis.

Naive Bayes Classifier

- A simple probabilistic classifier based on Bayes' theorem.

How Naive Bayes Classifier Works?

Training Data:

- Positive Tweet 1: "I love the new phone."
- Positive Tweet 2: "Great weather today!"
- Negative Tweet 1: "I hate waiting."

Step 1: Tokenization

Tokenization involves breaking down each tweet into individual words:

- Positive Tweet 1: ["I", "love", "the", "new", "phone"]
- Positive Tweet 2: ["Great", "weather", "today"]
- Negative Tweet 1: ["I", "hate", "waiting"]

Step 2: Calculate Prior Probabilities

Calculate the prior probabilities based on the training data:

- $P(Positive) = \frac{2}{3}$
- $P(Negative) = \frac{1}{3}$

Step 3: Calculate Likelihood

Calculate the likelihood of each word given the sentiment:

- Positive Tweet 1: ["I", "love", "the", "new", "phone"]
- Positive Tweet 2: ["Great", "weather", "today"]

- For Positive sentiment:
  - $P(I|Positive) = \frac{1}{5}$
  - $P(love|Positive) = \frac{1}{5}$
  - $P(the|Positive) = \frac{1}{5}$
  - $P(new|Positive) = \frac{1}{5}$
  - $P(phone|Positive) = \frac{1}{5}$
  - $P(Great|Positive) = \frac{1}{5}$
  - $P(weather|Positive) = \frac{1}{5}$
  - $P(today|Positive) = \frac{1}{5}$

- Negative Tweet 1: ["I", "hate", "waiting"]
- For Negative sentiment:
  - $P(I|Negative) = \frac{1}{3}$
  - $P(hate|Negative) = \frac{1}{3}$
  - $P(waiting|Negative) = \frac{1}{3}$

Step 4: Calculate Posterior Probabilities

Now, suppose we have a new tweet: "I love the great weather today."

Calculate the posterior probabilities for both Positive and Negative sentiments:

- For Positive sentiment:
  - $P(D|Positive) = P(I|Positive) \times P(love|Positive) \times P(the|Positive) \times P(great|Positive) \times P(weather|Positive) \times P(today|Positive) \approx \frac{1}{5} \times \frac{1}{5} \times \frac{1}{5} \times \frac{1}{5} \times \frac{1}{5} \times \frac{1}{5} = \frac{1}{3125}$
  - $P(D) = P(D|Positive) \times P(Positive) + P(D|Negative) \times P(Negative) \approx \frac{1}{3125} \times \frac{2}{3} + 0 \times \frac{1}{3} = \frac{2}{4687}$
  - $P(Positive|D) = \frac{P(D|Positive) \times P(Positive)}{P(D)} \approx \frac{\frac{1}{3125} \times \frac{2}{3}}{\frac{2}{4687}} = \frac{1}{3125} \times \frac{2}{3} \times \frac{4687}{2} = \frac{1}{3125} \times 2343 \approx 0.74976$

- For Negative sentiment:
  - $P(D|Negative) = P(I|Negative) \times P(love|Negative) \times P(the|Negative) \times P(great|Negative) \times P(weather|Negative) \times P(''today''|Negative) \approx \frac{1}{3} \times 0 \times \frac{1}{3} \times 0 \times \frac{1}{3} \times 0 = 0$
  - $$P(Negative|D) = \frac{P(D|Negative) \times P(Negative)}{P(D)} = 0$$

In this case, the model correctly predicts that the tweet is positive because "love," "great," "weather," and "today" are words associated with positive sentiment in the training data.

Summary

1. Tokenization
   - Split the text into individual words (tokens) e.g. $w_1, w_2, w_3, \ldots, w_n$
2. Calculate prior probabilities based on training data
   - $P(Positive) = \frac{Number\ of\ Positive\ Tweets}{Total\ Number\ of\ Tweets}$
   - $P(Negative) = \frac{Number\ of\ Negative\ Tweets}{Total\ Number\ of\ Tweets}$

3. Calculate likelihood based on training data
    - For each sentiment, calculate the probability of each word appearing in a tweet of that sentiment
        - $P(D|Positive) = P(w_1|Positive) * P(w_2|Positive) * P(w_3|Positive) * \ldots * P(w_n|Positive)$
        - $P(D|Negative) = P(w_1|Negative) * P(w_2|Negative) * P(w_3|Negative) * \ldots * P(w_n|Negative)$

4. Calculate posterior probabilities
    - $$P(Positive|D) = \frac{P(D|Positive) * P(Positive)}{P(D)}$$
    - $$P(Negative|D) = \frac{P(D|Negative) * P(Negative)}{P(D)}$$

Code example

```python
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.pipeline import make_pipeline

# Training data
texts = ["I love this product!", "This is terrible.", "Neutral review."]
labels = ["positive", "negative", "neutral"]

# Create a Naive Bayes classifier pipeline
model = make_pipeline(CountVectorizer(), MultinomialNB())

# Train the model
model.fit(texts, labels)

# Test data
test_texts = ["This is a great experience!", "I don't like it.", "It's okay."]

# Predict sentiments
predictions = model.predict(test_texts)

# Display the results
for text, sentiment in zip(test_texts, predictions):
    print(f"Text: {text} | Predicted Sentiment: {sentiment}")
```

Sentiment analysis using NLTK

```python
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.sentiment import SentimentIntensityAnalyzer

nltk.download('punkt')
nltk.download('stopwords')
nltk.download('vader_lexicon')
```

```
SentimentIntensityAnalyzer().polarity_scores("I love this product!")
```

```
>>> {'neg': 0.0, 'neu': 0.308, 'pos': 0.692, 'compound': 0.6696}
```

```
SentimentIntensityAnalyzer().polarity_scores("My worst day ever.")
```

```
>>> {'neg': 0.577, 'neu': 0.423, 'pos': 0.0, 'compound': -0.6249}
```

The `SentimentIntensityAnalyzer` in NLTK is part of the Vader (Valence Aware Dictionary and sEntiment Reasoner) sentiment analysis tool. Vader is specifically designed for analyzing sentiments in text data and is well-suited for social media texts, reviews, and other short text snippets.

How `SentimentIntensityAnalyzer` works ?

1. **Lexicon-based approach:**

   Vader uses a pre-built lexicon (a dictionary) that contains words and their associated sentiment scores. The lexicon is crafted to handle sentiments expressed in various contexts, including emoticons, capitalization, and intensifiers.

2. **Polarity Scores:**

   The `SentimentIntensityAnalyzer` assigns a polarity score to each word in the text. The scores include:

   - **Positive score:** The likelihood that the text expresses positive sentiment.
   - **Neutral score:** The likelihood that the text is neutral.
   - **Negative score:** The likelihood that the text expresses negative sentiment.
   - **Compound score:** A combination of the three scores above, normalized to fall between -1 (most negative) and +1 (most positive)

3. **Sentiment Classification:**
   Based on the compound score, the sentiment of the text is classified into three categories:
   - Positive
   - Negative
   - Neutral
4. **Handling Intensifiers and Negations:**
   Vader is designed to handle intensifiers (e.g., "very good") and negations (e.g., "not bad") effectively. It considers the impact of such words on the sentiment scores.

The `polarity_scores` method returns a dictionary containing positive, neutral, negative, and compound scores. You can then interpret these scores to determine the overall sentiment of the text.

Keep in mind that while **Vader is efficient and easy to use, it may not be suitable for all types of text or domains**. For more complex tasks or domain-specific sentiment analysis, you might consider using machine learning-based approaches with custom-trained models.

RoBERTa (Robustly Optimized BERT Pretraining Approach)

- State of the art model for sentiment analysis.

Part 02

Recommender Systems

The goal of recommender systems is to predict the preferences or interests of users and recommend items (such as products, movies, music, articles, or other content) that are likely to be of interest to them.

Why are recommender systems important?

- Content discovery
- Personalization
- Increase in sales

Practical applications of recommender systems include:

- Netflix movie recommendations
- Spotify music recommendations
- Facebook friend suggestions
- LinkedIn job suggestions
- YouTube video recommendations
- Google search suggestions
- Google News personalization

Types of Recommender Systems

1. **Collaborative Filtering**
    - Based on user-item interactions and preferences.
    - User preferences are inferred from similar users.
2. **Content-Based Filtering**
    - Recommends items similar to those the user has liked.
    - Focuses on item characteristics.
3. **Hybrid Recommender Systems**
    - Combines collaborative and content-based methods for improved accuracy.

Collaborative Filtering

- **User-Item Interaction Matrix**
  - Represents how users have interacted with items (ratings, clicks, purchases).
- **User-Based Collaborative Filtering**
  - Recommends items based on the preferences of users with similar tastes.
- **Item-Based Collaborative Filtering**
  - Recommends items similar to those a user has already interacted with.

Problems with Collaborative Filtering

- Popularity Bias
- Limited diversity

Content-Based Filtering

- **Item Profile**
  - Represents the characteristics or features of each item.
- **User Profile**
  - Represents the user's preferences based on the items they have liked or interacted with.
- **Recommendation Generation**
  - Recommends items that match the user's profile.

Problems with Content-Based Filtering

- Limited Serendipity
- Dependency on Feature Quality
- Difficulty in Capturing User Preferences Over Time

Hybrid Recommender Systems

- **Combining Strengths**
  - Leverages both collaborative and content-based methods.
  - Overcomes limitations of individual approaches.
- **Example**
  - Use collaborative filtering to identify user preferences and content-based filtering to refine recommendations based on item characteristics.

Challenges in Recommender Systems

1. **Cold Start Problem**
    - Difficulty in recommending to new users or items with limited data.
2. **Data Sparsity**
    - Incomplete user-item interaction data can lead to inaccurate recommendations.
3. **Scalability**
    - Challenges in handling large datasets and real-time recommendations.

Popular Recommender System Algorithms

1. **Matrix Factorization**
    - Factorizes the user-item interaction matrix to capture latent factors.
2. **Neural Collaborative Filtering**
    - Utilizes neural networks to model user-item interactions.
3. **Association Rule Mining**
    - Identifies patterns in user behavior to generate recommendations.

Coding Time …

Part 03

Association Rule Mining

- Rule based machine learning technique used to find patterns (relationships, structures) in data
- Used for market basket analysis, cross-marketing, catalog design, sale campaign analysis, web log (click stream) analysis, and risk management
- Used to find frequent patterns, associations, correlations, or causal structures among sets of items in transaction databases, relational databases, and other information repositories
- Predict customer behavior

Association Rule

- The association rule is a learning technique that helps identify dependency between two data items
- Simple If/Then statements
- If {condition} Then {conclusion} / antecedent -> consequent
- Example if it rains then he will take an umbrella

Important measures which is used for association rules:

- **Support**
- **Confidence**
- **Lift**

Example dataset:

TID	Items
1	**apple**, **orange**, banana, cherry
2	mango, grape, **orange**, pineapple
3	pear, **orange**, watermelon, **apple**
4	strawberry, mango, grape, banana

Problem:

- Are there any association rules between the items?
- If customer buys apple, does he buy orange too?

Support

- The support of an itemset X is defined as the proportion of transactions in the database which contain the itemset X
- Support is an indication of how frequently the itemset appears in the dataset
- Support is calculated by dividing the number of transactions containing the itemset by the total number of transactions
- Formula: $support = \frac{freq(A,B)}{N_{total\ number\ of\ transactions}}$

TID	Items
1	**apple**, **orange**, banana, cherry
2	mango, grape, **orange**, pineapple
3	pear, **orange**, watermelon, **apple**
4	strawberry, mango, grape, banana

- Example:
    - Support(apple) = 2/4 = 0.5
    - Support(orange) = 3/4 = 0.75
    - Support(apple U orange) = 2/4 = 0.5

In this example, support(apple) = 0.5, support(orange) = 0.75, support(apple U orange) = 0.5, so we can say that apple and orange are frequent item.

Confidence

- Confidence is an indication of how often the rule has been found to be true
- Confidence is calculated by dividing the number of transactions containing both the antecedent and consequent by the number of transactions containing the antecedent
- Formula: $confidence = \frac{freq(A,B)}{freq(A)}$

TID	Items
1	**apple**, **orange**, banana, cherry
2	mango, grape, **orange**, pineapple
3	pear, **orange**, watermelon, **apple**
4	strawberry, mango, grape, banana

- Example:
    - Confidence(apple -> orange) = Support(apple U orange) / Support(apple) = 0.5 / 0.5 = 1
    - Confidence(orange -> apple) = Support(apple U orange) / Support(orange) = 0.5 / 0.75 = 0.66

Confidence is not symmetric, so we need to calculate both confidence(apple -> orange) and confidence(orange -> apple)

In this example, confidence(apple -> orange) = 1 and confidence(orange -> apple) = 0.66, so we can say that if customer buys apple, then he will buy orange with 100% probability, but if customer buys orange, then he will buy apple with 66% probability

Lift

Formula: $lift(A->B) = \frac{confidence(A,B)}{support(B)}$

or $lift = \frac{support(A,B)}{support(A)*support(B)}$

- Lift is the ratio of the observed support to that expected if X and Y were independent
- Lift is calculated by dividing the confidence by the support of consequent

Example:

- Lift(apple -> orange) = Confidence(apple -> orange) / Support(orange) = 1 / 0.75 = 1.33
- Lift(orange -> apple) = Confidence(orange -> apple) / Support(apple) = 0.66 / 0.5 = 1.33

Lift is symmetric, so we can calculate only one of them

Conclusion:

- If lift = 1, then antecedent and consequent are independent
- If lift > 1, then antecedent and consequent are dependent
- If lift < 1, then antecedent and consequent are negatively dependent

In this example, lift(apple -> orange) = lift(orange -> apple) = 1.33, so we can say that apple and orange are dependent

Association Rule Mining Example

TID			
T1	A	B	C
T2	A	C	D
T3	B	C	D
T4	A	D	E
T5	B	C	E

Rules:

- A -> D
- C -> A
- A -> C
- B & C -> A

TID			
T1	A	B	C
T2	A	C	D
T3	B	C	D
T4	A	D	E
T5	B	C	E

A -> D

- support(A -> D) = freq(A,D) / N = 2/5 = 0.4
- confidence(A -> D) = freq(A,D) / freq(A) = 2/3 = 0.66
- lift(A -> D) = confidence(A -> D) / support(D) = 0.66 / 0.6 = 1.1
- **Conclusion**: A and D are dependent, meaning that if customer buys A, then he will buy D

TID			
T1	A	B	C
T2	A	C	D
T3	B	C	D
T4	A	D	E
T5	B	C	E

C -> A

- support(C -> A) = freq(C,A) / N = 1/5 = 0.2
- confidence(C -> A) = freq(C,A) / freq(C) = 1/4 = 0.25
- (C -> A) = confidence(C -> A) / support(A) = 0.25 / 0.6 = 0.41
- **Conclusion**: C and A are negatively dependent, meaning that if customer buys C, then he will not buy A

TID			
T1	A	B	C
T2	A	C	D
T3	B	C	D
T4	A	D	E
T5	B	C	E

A -> C

- support(A -> C) = freq(A,C) / N = 2/5 = 0.4
- confidence(A -> C) = freq(A,C) / freq(A) = 2/3 = 0.66
- lift(A -> C) = confidence(A -> C) / support(C) = 0.66 / 0.6 = 1.1

- **Conclusion**: A and C are dependent, meaning that if customer buys A, then he will buy C

TID			
T1	A	B	C
T2	A	C	D
T3	B	C	D
T4	A	D	E
T5	B	C	E

B & C -> A

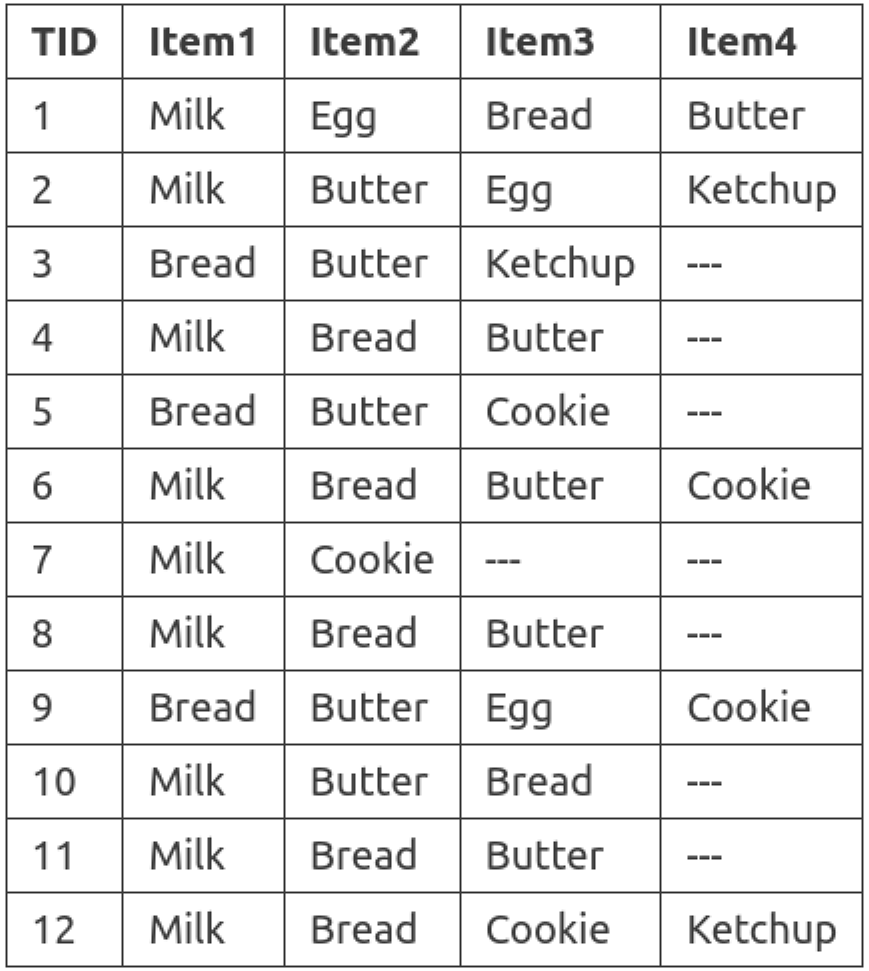
- support(B & C -> A) = freq(B,C,A) / N = 1/5 = 0.2
- confidence(B & C -> A) = freq(B,C,A) / freq(B,C) = 1/3 = 0.33
- lift(B & C -> A) = confidence(B & C -> A) / support(A) = 0.33 / 0.6 = 0.55
- Conclusion: B and C are negatively dependent, meaning that if customer buys B and C, then he will not buy A

Apriori Algorithm

- The Apriori algorithm uses frequent itemsets to generate association rules.
- The Apriori algorithm is based on the concept that a subset of a frequent itemset must also be a frequent itemset.
- The Apriori algorithm uses a "bottom up" approach, where frequent subsets are extended one item at a time (a step known as candidate generation), and groups of candidates are tested against the data. The algorithm terminates when no further successful extensions are found.
- Frequent itemset is an itemset whose support value is greater than or equal to a threshold value.

Apriori Algorithm Explanation with example

- Minimum support = 33%



TID	Item1	Item2	Item3	Item4
1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	---
4	Milk	Bread	Butter	---
5	Bread	Butter	Cookie	---
6	Milk	Bread	Butter	Cookie
7	Milk	Cookie	---	---
8	Milk	Bread	Butter	---
9	Bread	Butter	Egg	Cookie
10	Milk	Butter	Bread	---
11	Milk	Bread	Butter	---
12	Milk	Bread	Cookie	Ketchup

First Iteration

TID	Item1	Item2	Item3	Item4
1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	---
4	Milk	Bread	Butter	---
5	Bread	Butter	Cookie	---
6	Milk	Bread	Butter	Cookie
7	Milk	Cookie	---	---
8	Milk	Bread	Butter	---
9	Bread	Butter	Egg	Cookie
10	Milk	Butter	Bread	---
11	Milk	Bread	Butter	---
12	Milk	Bread	Cookie	Ketchup

1-item sets	Frequency	Support
Milk	9	75%
Bread	10	83%
Butter	10	83%
Egg	3	25%
Ketchup	3	25%
Cookie	5	41%

First Iteration

TID	Item1	Item2	Item3	Item4
1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	---
4	Milk	Bread	Butter	---
5	Bread	Butter	Cookie	---
6	Milk	Bread	Butter	Cookie
7	Milk	Cookie	---	---
8	Milk	Bread	Butter	---
9	Bread	Butter	Egg	Cookie
10	Milk	Butter	Bread	---
11	Milk	Bread	Butter	---
12	Milk	Bread	Cookie	Ketchup

1-item sets	Frequency	Support
Milk	9	75%
Bread	10	83%
Butter	10	83%
Egg	3	25%
Ketchup	3	25%
Cookie	5	41%

Frequent 1-item sets	Frequency	Support
Milk	9	75%
Bread	10	83%
Butter	10	83%
Cookie	5	41%

Second Iteration

TID	Item1	Item2	Item3	Item4
1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	---
4	Milk	Bread	Butter	---
5	Bread	Butter	Cookie	---
6	Milk	Bread	Butter	Cookie
7	Milk	Cookie	---	---
8	Milk	Bread	Butter	---
9	Bread	Butter	Egg	Cookie
10	Milk	Butter	Bread	---
11	Milk	Bread	Butter	---
12	Milk	Bread	Cookie	Ketchup

2-item sets	Frequency	Support
Milk, Bread	7	58%
Milk, Butter	7	58%
Milk, Cookie	3	25%
Bread, Butter	9	75%
Bread, Cookie	4	33%
Butter, Cookie	3	25%

Second Iteration

TID	Item1	Item2	Item3	Item4
1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	---
4	Milk	Bread	Butter	---
5	Bread	Butter	Cookie	---
6	Milk	Bread	Butter	Cookie
7	Milk	Cookie	---	---
8	Milk	Bread	Butter	---
9	Bread	Butter	Egg	Cookie
10	Milk	Butter	Bread	---
11	Milk	Bread	Butter	---
12	Milk	Bread	Cookie	Ketchup

2-item sets	Frequency	Support
Milk, Bread	7	58%
Milk, Butter	7	58%
Milk, Cookie	3	25%
Bread, Butter	9	75%
Bread, Cookie	4	33%
Butter, Cookie	3	25%

Frequent 2-item sets	Frequency	Support
Milk, Bread	7	58%
Milk, Butter	7	58%
Bread, Butter	9	75%
Bread, Cookie	4	33%

Third Iteration

TID	Item1	Item2	Item3	Item4
1	Milk	Egg	Bread	Butter
2	Milk	Butter	Egg	Ketchup
3	Bread	Butter	Ketchup	---
4	Milk	Bread	Butter	---
5	Bread	Butter	Cookie	---
6	Milk	Bread	Butter	Cookie
7	Milk	Cookie	---	---
8	Milk	Bread	Butter	---
9	Bread	Butter	Egg	Cookie
10	Milk	Butter	Bread	---
11	Milk	Bread	Butter	---
12	Milk	Bread	Cookie	Ketchup

3-item sets	Frequency	Support
Milk, Bread, Butter	6	50%
Milk, Bread, Cookie	1	8%
Milk, Butter, Cookie	2	16%
Bread, Butter, Cookie	3	25%

Frequent 3-item sets	Frequency	Support
Milk, Bread, Butter	6	50%

Association Rule Mining: Subset creation

- Frequent 3-item sets are Milk, Bread, Butter (Let's call it I)
- Non-empty subsets of Milk, Bread, Butter are: (Let's call it S)
  - Milk, Bread
  - Milk, Butter
  - Bread, Butter
  - Milk
  - Bread
  - Butter

How to create association rules?

- For every non empty subset of Milk, Bread, Butter, check if the confidence of the rule is greater than or equal to the minimum confidence threshold
- If the confidence of the rule is greater than or equal to the minimum confidence threshold, then the rule is a valid rule

- $S- > (I - S)$
- if $support(I)/support(S) >= minimum\ confidence\ threshold$

Example

Minimum support = 30%

Minimum confidence = 60%

- **Rule 1**: Milk -> Bread, Butter
  - support = 6/12 = 50%
  - confidence = support(Milk, Bread, Butter) / support(Milk) = (6/12) / (9/12) = 0.66 = 66%
  - **Valid rule**

- **Rule 2**: Bread -> Milk, Butter
  - support = 6/12 = 50%
  - confidence = support(Milk, Bread, Butter) / support(Bread) = (6/12) / (10/12) = 0.6 = 60%
  - **Valid rule**

- **Rule 3**: Butter -> Milk, Bread
  - support = 6/12 = 50%
  - confidence = support(Milk, Bread, Butter) / support(Butter) = (6/12) / (10/12) = 0.6 = 60%
  - **Valid rule**

- **Rule 4**: Milk, Bread -> Butter
  - support = 6/12 = 50%
  - confidence = support(Milk, Bread, Butter) / support(Milk, Bread) = (6/12) / (7/12) = 0.85 = 85%
  - **Valid rule**

- **Rule 5**: Milk, Butter -> Bread
    - support = 6/12 = 50%
    - confidence = support(Milk, Bread, Butter) / support(Milk, Butter) = (6/12) / (7/12) = 0.85 = 85%
    - **Valid rule**

- **Rule 6**: Bread, Butter -> Milk
  - support = 6/12 = 50%
  - confidence = support(Milk, Bread, Butter) / support(Bread, Butter) = (6/12) / (9/12) = 0.66 = 66%
  - **Valid rule**

- Further reading
- Code Example

Summary

- **Sentiment Analysis**: The process of computationally identifying and categorizing opinions expressed in a piece of text or document.
  - Naive Bayes Classifier
  - Sentiment Analysis using NLTK
  - RoBERTa

Summary

- **Recommender Systems**: Predict the preferences or interests of users and recommend items that are likely to be of interest to them.
    - Collaborative Filtering
    - Content-Based Filtering
    - Hybrid Recommender Systems

Summary

- **Association Rule Mining**: A rule based machine learning technique used to find patterns in data.
  - Support
  - Confidence
  - Lift
  - Apriori Algorithm

Thank You

Happy Learning 🚀 !!

meftaulhaque@gmail.com