Comparison of SoP Set and KNN Methods for Recommendation Systems[*]

Piotr Solarczyk[1,*,†], Jakub Stachurski[1,†]

[1]Faculty of Applied Mathematics, Silesian University of Technology, Kaszubska 23, 44100 Gliwice, POLAND

Abstract

This paper presents a comparative study between Soft Set and K-Nearest Neighbors (KNN) methods for recommendation systems. We utilize the Netflix dataset to analyze and compare the performance of both methods. The Soft Set approach is highlighted as a potentially more effective method due to its flexibility in handling uncertainties and partial truths, as opposed to the more rigid KNN. The results indicate that the Soft Set method provides more accurate and relevant recommendations, showcasing its potential in improving recommendation systems.

Keywords

Recommendation Systems, Soft Set, K-Nearest Neighbors, Netflix, Data Analysis

1. Introduction

Recommendation systems have become an integral part of numerous online services, providing personalized content suggestions to users based on their preferences and behavior. These sys- tems enhance user experience by suggesting relevant items, thereby increasing user engagement and satisfaction. Various algorithms and techniques have been developed for recommendation systems, each with its strengths and weaknesses. K-Nearest Neighbors (KNN) is one of the most widely used algorithms due to its simplicity and effectiveness. It operates by finding the closest data points in the feature space and using them to make predictions or recommendations. However, KNN has its limitations, particularly in handling the uncertainties and partial truths that often exist in real-world data.

In contrast, Soft Set theory offers a more flexible approach to dealing with such uncertainties. Introduced by Molodtsov in 1999, Soft Set theory provides a framework for reasoning about data that is imprecise or incomplete. It allows for a more nuanced representation of relationships between items, which can be particularly useful in recommendation systems where user preferences are not always clear-cut. This paper explores the potential of Soft Set theory as an alternative to KNN for recommendation systems. We utilize the Netflix dataset to conduct a comparative analysis of the two methods, focusing on their accuracy and effectiveness in generating relevant recommendations. The study aims to demonstrate that the Soft Set approach can provide better recommendations by leveraging its inherent flexibility in handling uncertainties.

[*]IVUS2024: Information Society and University Studies 2024, May 17, Kaunas, Lithuania

[1,*]Corresponding author

[†] These author contributed equally.

ps309167@student.polsl.pl (P. Solarczyk); js308026@student.polsl.pl (J. Stachurski)

0009-0005-4831-9242 (P. Solarczyk); 0009-0005-1020-0414 (J.Stachurski)

2. Methodology

In this section, we detail the methodologies employed in our study: the K-Nearest Neighbors (KNN) algorithm and the Soft Set method. Both methodologies are implemented to generate recommendations based on the Netflix dataset, and their performance is compared.

2.1. Data Preprocessing

The initial step involves loading and preprocessing the Netflix dataset. The dataset contains information about various titles, including their type, rating, and genres. We start by loading the dataset and removing rows with missing values to ensure data quality. The relevant features for our recommendation system, such as title, type, rating, and genres (listed_in), are then extracted for further analysis.

```
import pandas as pd

# Load data
df = pd.read_csv('netflix_titles.csv')

# Data cleaning - remove rows with missing values
df_cleaned = df.dropna()

# Extract features needed for recommendations
df_features = df_cleaned[['title', 'type', 'rating'
    ,                          'listed_in']].copy()
```

2.2. K-Nearest Neighbors (KNN)

K-Nearest Neighbors (KNN) is a simple yet powerful algorithm used for classification and regression tasks. In our context, KNN is employed to generate movie recommendations. The algorithm works by calculating the similarity between the genres of a given movie and those of other movies in the dataset. The TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer is used to convert the genres into numerical features, and the cosine similarity metric is used to find the k-nearest neighbors. The KNN recommendation function is implemented as follows:

```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.neighbors import NearestNeighbors

# KNN recommendation function
def knn_recommendation(title, k=5):
    tfidf = TfidfVectorizer(stop_words='english')
    tfidf_matrix = tfidf.fit_transform(df_features['listed_
    in'
        ])

    knn = NearestNeighbors(n_neighbors=k+1, metric='cosi
    ne')knn.fit(tfidf_matrix)

    idx = df_features[df_features['title'].str.contains(t
        itle,case=False, regex=False)].index
    if len(idx) == 0:
        print(f"No title containing '{title}' found.")
        return []

    idx = idx[0]
    distances,indices = knn.kneighbors(tfidf_matrix[i
        dx],n_neighbors=k+1)

    recommended_titles = df_features.iloc[indices[0]].
        title.values[1:]
    return recommended_titles
```

2.3. SoP Set

The Soft Set theory provides a mathematical framework for dealing with uncertainties and partial truths. Unlike traditional set theory, Soft Set theory allows for the representation of vague concepts, making it suitable for recommendation systems where user preferences are often imprecise. The Soft Set recommendation function calculates the similarity between the genres of the input title and those of other titles in the dataset using the Jaccard similarity coefficient. This method is more flexible and can handle the partial truths and uncertainties present in the data. The implementation is as follows:

```
# Soft set recommendation function
def soft_set_recommendation(title, threshold=0.6):
    input_row= df_features[df_features['title'].str.conta
        ins(title, case=False, regex=False)]
    if input_row.empty:
        print(f"No title containing '{title}' found.")
        return []

    input_genres = set([genre.strip().lower() for genre in
        input_row.iloc[0]['listed_in'].split(',')])

    genres_list = df_features['listed_in'].str.split(',').
    apply
        (lambda x: set([i.strip().lower() for i in x]))
    def calculate_similarity(genres):
        return len(input_genres.intersection(genres)) /
            len(input_genres.union(genres))

    df_features['similarity'] = genres_list.
        apply(calculate_similarity)
    recommendations = df_features.sort_values(by='similarit
        y',ascending=False).head(6).title.values[1:]

    return recommendations
```

3. Experiments

To evaluate the performance of the KNN and Soft Set methods, we conducted experiments using the Netflix dataset. We selected the popular TV show "Breaking Bad" as the input title and generated recommendations using both methods. The accuracy of the recommendations was then evaluated based on feature similarity, taking into account the type, rating, and genres of the titles.

3.1. Evaluation of Recommendations

The accuracy of the recommendations is assessed by calculating the average similarity of the recommended titles to the input title. The feature similarity is determined by comparing the type, rating, and genres of the titles. The following functions are used to calculate the feature similarity and evaluate the accuracy of the recommendations:

```
# Feature similarity calculation function
def feature_similarity(title1, title2):
    features1= df_features[df_features['title'].str.cont
        ains(title1, case=False, regex=False)]
    features2 = df_features[df_features['title'] == title2]

    if len(features1) == 0 or len(features2) == 0:
        return 0

    features1 = features1.iloc[
    0]features2 = features2.ilo
    c[0]

    similarity = 0
    if features1['type'] == features2['type
        ']:similarity += 1
    if features1['rating'] == features2['ratin
        g']:similarity += 1
    genres1 = set(features1['listed_in'].split('
    ,                                              '))
    genres2 = set(features2['listed_in'].split(',
                                                   ')
    )
    genre_similarity = len(genres1.intersection(genres2)) /
    len
      (genres1.union(genres2))
   similarity += genre_similarity

    return similarity / 3

# Evaluation of recommendations accuracy
def evaluate_recommendations_accuracy(input_title
   s,recommended_titles):
    total_accuracy = 0
    for input_title in input_title
         s:input_accuracy = 0
         recommended_titles_for_input=recommended_title
             s.get(input_title , [])
         if len(recommended_titles_for_input) > 0:
              for recommended_title in
                  recommended_titles_for_input:
                   input_accuracy+= feature_similar
                       ity(input_title , recommended
                       _title)
              input_accuracy/=len(recommended_titles_for_in
              put)total_accuracy += input_accuracy
    if total_accuracy == 0:
         return 0
    total_accuracy /= len(input_titles)
    return total_accuracy * 100
```

We generated recommendations for "Breaking Bad" using both KNN and Soft Set methods. The results were compared to assess the effectiveness of each method in providing relevant recommendations.

3.2. Results

The results of our experiments indicate that the Soft Set method outperforms the KNN method in terms of recommendation accuracy. The Soft Set method, with its ability to handle uncertainties and partial truths, provided recommendations that were more similar to the input title "Breaking Bad" in terms of type, rating, and genres. The accuracy of the KNN method was lower, highlighting its limitations in dealing with the imprecise nature of real-world data.

The accuracy of the recommendations is summarized as follows:

Movie	**Recommendations (KNN)**	**Recommendations (SoP Sets)**	**Accuracy (KNN) (%)**	**Accuracy (SoP Sets) (%)**
Breaking Bad	Krish Trish and Baltiboy: Face Your Fears, Fishtronaut: The Movie, Pinkfong & Baby Shark's Space Adventure	Come and Find Me, Victim of Beauty, The Girl with the Dragon Tattoo	33.33	66.67
Paranoia	Hera Pheri, Hunt for the Wilderpeople, Happy New Year	Dismissed, Deviant Love, Running Out Of Time	40.00	80.00
Labyrinth	Yes Man, Legally Blonde, The Kissing Booth 2	National Treasure, Red Dawn, Kung Fu Hustle	33.33	65.56
Murder Mystery	The Next Skin, Punjab 1984, Nimbe	Blue Mountain State: The Rise of Thadland, Dead in a Week (Or Your Money Back), Don Verdean	33.33	73.33
God of War	Jailbreak, Twins Mission, Manhunt	Department, Dragon Tiger Gate, Disciples Of The 36th Chamber	66.67	66.67

Table 1
Comparison of KNN and Soft Sets Recommendation Systems

4. Conclusion

In this study, we compared the performance of the K-Nearest Neighbors (KNN) and Soft Set methods for recommendation systems using the Netflix dataset. The dataset was preprocessed to remove missing values, and features such as title, type, rating, and genres were extracted for analysis.

The KNN method was implemented using the TF-IDF vectorizer to convert genre information into numerical features, and cosine similarity was used to find the nearest neighbors. The Soft Set method, on the other hand, calculated the similarity between the genres of the input title and other titles using the Jaccard similarity coefficient.

We evaluated the recommendations generated by both methods using the feature similarity calculation, which took into account the type, rating, and genres of the titles. The accuracy of the recommendations was measured by comparing the average similarity of the recommended titles to the input title.

Our experiments, which focused on the popular TV show "Breaking Bad," demonstrated that the Soft Set method provided more accurate and relevant recommendations compared to the

KNN method. Specifically, the Soft Set method achieved higher accuracy by effectively handling the uncertainties and partial truths inherent in user preferences and real-world data.

The results highlight the potential of the Soft Set method as a superior alternative to KNN for recommendation systems. The flexibility of Soft Set theory in dealing with imprecise and incomplete data makes it well-suited for applications where user preferences are not always clear-cut.

Future work could explore integrating Soft Set theory with other machine learning techniques to further enhance the accuracy and effectiveness of recommendation systems. Additionally, expanding the evaluation to include a larger variety of input titles and diverse datasets could provide a more comprehensive assessment of the methods' performance.

References

[1] Molodtsov, D. (1999). Soft set theory—First results. *Computers & Mathematics with Applications*, 37(4-5), 19-31.

[2] Netflix. (2016). Netflix Movies and TV Shows Dataset. Retrieved from https://www.kaggle.com/shivamb/netflix-shows

[3] Salton, G., & Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 24(5), 513-523.

[4] Cover, T., & Hart, P. (1967). Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1), 21-27.

[5] Adomavicius, G., & Tuzhilin, A. (2005). Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Transactions on Knowledge and Data Engineering*, 17(6), 734-749.

[6] Jaccard, P. (1901). Étude comparative de la distribution florale dans une portion des Alpes et du Jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37, 547-579.

[7] Huang, A. (2008). Similarity measures for text document clustering. In *Proceedings of the sixth new zealand computer science research student conference (NZCSRSC)* (pp. 49-56).

[8] Maji, P. K., Biswas, R., & Roy, A. R. (2002). An application of soft sets in a decision making problem. *Computers & Mathematics with Applications*, 44(8-9), 1077-1083.