Improving Personalization in Education:

An Approach to Automatically Tagging Resources with Tag Recommender Systems

Zakaria TAGDIMI [1], Souhaib AAMMOU [1], Ikram AMZIL [1] and Hicham ERRADI [1]

1 Abdelmalek Essaadi University, S2IPU, Morocco
zakaria.tagdimi@etu.uae.ac.ma

Abstract. This article discusses the use of Tag Recommender Systems (TRS) in Intelligent Educational Systems (IES) for providing personalized learning experiences to learners. The article explains how TRS works by analyzing tags associated with resources and suggesting similar content to users based on their tag preferences. The article also explains the Naive Bayes algorithm, which is used in building tag recommendation systems, and its application in LMSs. The goal is to recommend courses to learners on the LMS platform that align with their interests by recommending courses with tags that closely match their preferences.

Keywords: Intelligent Educational System (IES), Personalized learning, Tag Recommender Systems (TRS), Content-based filtering, Naive Bayes classifier

1 Introduction

Intelligent Educational Systems (IES) are designed to provide personalized learning experiences to students through the use of AI and ML technologies [1]. The system is aimed at understanding the student's abilities, interests, and learning style and then recommending appropriate content to enhance the learning experience.

The recommendation is considered a major issue in the field of education as it aims to provide personalized learning experiences for learners [2]. It can be a complex and time-consuming process as it involves understanding the learner's abilities, interests, and learning style, and then recommending appropriate resources. However, as a result of technological advancements, recommendation systems have been created to help teachers give learners personalized learning experiences.

In this article, we will present an approach based on Tag Recommender Systems (TRS) that can be useful for automatically tagging resources on an LMS platform [3], and thus, making it easier to find the resources that interest them.

Our goal is to recommend courses to learners on our LMS platform that align with their interests by recommending courses with tags that closely match their preferences.

M. Khaldi et al. (eds.), *Proceedings of the E-Learning and Smart Engineering Systems (ELSES 2023)*, Atlantis Highlights in Social Sciences, Education and Humanities 14,
https://doi.org/10.2991/978-94-6463-360-3_23

2 Theoretical framework

2.1 Recommender systems

Users receive individualized recommendations from recommender systems based on their preferences, actions, and interests [4]. Recommender systems are a sort of information filtering system. These systems evaluate user input and provide suggestions using data mining approaches including collaborative filtering, content-based filtering, and hybrid filtering.

Content-based filtering is a type of recommender system that suggests products or items to users based on their past interactions with similar items. This type of filtering uses the characteristics or features of items to recommend other items that have similar characteristics [5]. For example, if a user has shown interest in a specific type of book, a content-based recommender system will recommend other books with similar topics or genres.

Collaborative filtering is a technique used in recommender systems that suggests items to users based on the preferences of other users with similar interests. In collaborative filtering, the system analyzes the behavior and preferences of many users and finds patterns among them. These patterns are used to predict what other items a user might be interested in based on their past interactions with the system [6]. This process is often used in conjunction with content-based filtering to provide more accurate recommendations to users.

Hybrid filtering is a recommendation system approach that combines both collaborative filtering and content-based filtering techniques to provide personalized recommendations to users [7]. By using both methods, it can overcome some of the limitations of each individual technique, resulting in more accurate and diverse recommendations.

2.2 Tag Recommender Systems

Tag Recommender Systems (TRS) are a type of recommendation system that focuses on recommending tags or labels to resources such as documents, images, videos, and audio files. The goal of TRS is to improve the organization and retrieval of these resources by suggesting relevant and descriptive tags to users [8]. TRS use various algorithms and techniques such as content-based filtering and machine learning to recommend tags that are likely to be relevant to the content of a given resource. TRS can be useful for automatically tagging resources on a platform, making it easier to find resources that interest users.

Tag recommendation systems work by analyzing the tags associated with courses, articles, or other pieces of content, and then suggesting similar content to users based on their tag preferences. This is typically done using machine learning algorithms that analyze user behavior data to understand which tags are most relevant to each user's interests [9].

Once the system has identified a user's preferred tags, it can then suggest content that is tagged with those keywords, or with related keywords that are similar in meaning or context. This can help users discover new content that they may not have otherwise

found, and can also help to personalize their learning experience by providing content that is tailored to their individual interests and needs.

Tag recommendation systems and classifiers are closely related because classification algorithms are often used in building tag recommendation systems.

Classification algorithms are used to categorize data into different groups or classes based on specific criteria. In the case of tag recommendation systems, these algorithms are used to analyze the tags associated with different pieces of content and categorize them into different topics or themes.

One common classification algorithm used in tag recommendation systems is the Naive Bayes algorithm [10]. The Naive Bayes algorithm can be an effective tool for building tag recommendation systems in LMSs, and can help to personalize the learning experience for individual users by providing relevant course recommendations based on their interests and preferences.

3 Conceptual framework

3.1 Naïve bayes algorithm

The Naive Bayes algorithm is a probabilistic machine learning algorithm that is commonly used for classification tasks. The algorithm is based on Bayes' theorem, which is a fundamental concept in probability theory.

$$P\left(\frac{A}{B}\right) = \frac{P\left(\frac{B}{A}\right).P(A)}{P(B)} \quad (1)$$

The algorithm works by first learning the probabilities of each feature (e.g. word or tag) appearing in each class (e.g. topic or category). This is done by analyzing a labeled training set of data, where each data point is labeled with its corresponding class or category.

Once the probabilities have been learned, the algorithm can be used to classify new data points based on their features. For example, in text classification, the features may be the words that appear in a piece of text, and the classes may be different topics or categories (e.g. politics, sports, entertainment).

To classify a new piece of text, the algorithm calculates the probability of the text belonging to each of the classes based on the probabilities of each feature appearing in each class. It then selects the class with the highest probability as the predicted class for the new data point.

The Naive Bayes algorithm is called "naive" because it assumes that the features are independent of each other, which is often not the case in real-world data. Despite this simplifying assumption, Naive Bayes is often very effective in practice, particularly for text classification tasks where there are many features (i.e. words) and relatively few data points.

4 Implementation

4.1 Implementation process

This section will involve the implementation of the Naive Bayes algorithm on our Learning Management System (LMS), which consists of ten different courses:

1. **Data collection**: Collect data on the 10 courses offered on the LMS and the tags associated with each course. The courses and their associated tags are as follows:
   - Course 1: programming, coding, Python
   - Course 2: business, marketing, advertising
   - Course 3: history, politics, social studies
   - Course 4: literature, poetry, writing
   - Course 5: mathematics, statistics, calculus
   - Course 6: art, design, creativity
   - Course 7: science, biology, chemistry
   - Course 8: music, composition, theory
   - Course 9: sports, fitness, health
   - Course 10: cooking, food, nutrition
2. **Data preprocessing**: Clean and preprocess the data by removing stop words, stemming words, and converting text to lowercase. For example, we might convert "Python" to "python" and remove words like "the" and "and".
3. **Data splitting**: Split the data into a training set and a testing set. We split the data so that 70% of the data is used for training and 30% is used for testing.
4. **Feature extraction**: Extract the features from the training set. In this case, the features are the tags associated with each course.
5. **Training the Naive Bayes classifier**: Train the Naive Bayes classifier using the training set. The algorithm will learn the probabilities of each tag being associated with each course by counting the frequency of each tag in each course. For example, the algorithm might learn that the tag "programming" is associated with course 1, but not with any of the other courses.
6. **Testing and evaluation**: Test the performance of the Naive Bayes classifier using the testing set. This involves predicting the course of each data point (i.e. tag) in the testing set and comparing the predicted courses to the actual courses to calculate the accuracy of the classifier.
7. **Recommendation generation**: Once the Naive Bayes classifier has been trained and tested, it can be used to generate tag recommendations for users based on their preferences. When a user indicates their preferences, the classifier will analyze the tags associated with the courses on the LMS and recommend the courses that are most closely aligned with the user's preferences.

For example, if a user indicates that they are interested in programming and Python, the Naive Bayes classifier might recommend course 1 to the user. Alternatively, if a user indicates that they are interested in literature and writing, the classifier might recommend course 4 to the user.

4.2 Python code

```python
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import MultinomialNB
from sklearn.model_selection import train_test_split

# Step 1: Data collection
courses = pd.DataFrame({
    'course_name': ['Course 1', 'Course 2', 'Course 3', 'Course 4', 'Course 5',
                    'Course 6', 'Course 7', 'Course 8', 'Course 9', 'Course 10'
    'tags': ['programming coding python', 'business marketing advertising',
             'history politics social studies', 'literature poetry writing',
             'mathematics statistics calculus', 'art design creativity',
             'science biology chemistry', 'music composition theory',
             'sports fitness health', 'cooking food nutrition']
})

# Step 2: Data preprocessing (optional)
courses['tags'] = courses['tags'].str.lower()

# Step 3: Data splitting
X_train, X_test, y_train, y_test = train_test_split(
    courses['tags'], courses['course_name'], test_size=0.3, random_state=42)

# Step 4: Feature extraction
vectorizer = CountVectorizer()
X_train_counts = vectorizer.fit_transform(X_train)
X_test_counts = vectorizer.transform(X_test)

# Step 5: Training the Naive Bayes classifier
clf = MultinomialNB()
clf.fit(X_train_counts, y_train)

# Step 6: Testing and evaluation
accuracy = clf.score(X_test_counts, y_test)
print("Accuracy of the classifier:", accuracy)

# Step 7: Recommendation generation
user_preferences = "python programming"
user_tags = vectorizer.transform([user_preferences])
recommended_course = clf.predict(user_tags)[0]
print("Recommended course:", recommended_course)
```

Fig. 1. Implementation of Naive Bayes algorithm in Python for a tag recommendation system

We first collect data on the 10 courses offered on the LMS and the tags associated with each course. We then preprocess the data by converting the tags to lowercase.

We split the data into a training set and a testing set using the train_test_split function from the sklearn library. We use the CountVectorizer function to extract the features (i.e. tags) from the training set.

We train the Naive Bayes classifier using the MultinomialNB function from sklearn. We then test the accuracy of the classifier using the testing set.

Finally, we generate recommendations for users based on their preferences. We create a new set of tags based on the user's preferences and use the trained Naive Bayes classifier to predict the most likely course based on those tags.

5 Conclusion

In conclusion, Naive Bayes is a useful algorithm for implementing tag recommendation systems in LMSs. By analyzing the tags associated with each course, Naive Bayes can learn the probabilities of each tag being associated with each course and use this information to recommend courses to users based on their preferences. The implementation of Naive Bayes in a tag recommendation system involves stages such as data collection, data preprocessing, data splitting, feature extraction, training the Naive Bayes classifier, testing and evaluation, and recommendation generation.

References

1. Moallem, M. (2021). Smart Educational System. In *Smart and Intelligent Systems* (pp. 1-32). CRC Press.
2. MacLeod, A., Burm, S., & Mann, K. (2022). Constructivism: learning theories and approaches to research. *Researching Medical Education*, 25-40.
3. Pan, Y., Huo, Y., Tang, J., Zeng, Y., & Chen, B. (2021). Exploiting relational tag expansion for dynamic user profile in a tag-aware ranking recommender system. *Information Sciences*, *545*, 448-464.
4. Cheung, K. L., Durusu, D., Sui, X., & de Vries, H. (2019). How recommender systems could support and enhance computer-tailored digital health programs: a scoping review. *Digital health*, *5*, 2055207618824727.
5. Liao, M., Sundar, S. S., & B. Walther, J. (2022, April). User trust in recommendation systems: A comparison of content-based, collaborative and demographic filtering. In *Proceedings of the 2022 CHI Conference on Human Factors in Computing Systems* (pp. 1-14).
6. Murty, C. S., Saradhi Varma, G. P., & Satyanarayana, C. (2022). Content-based collaborative filtering with hierarchical agglomerative clustering using user/item based ratings. *Journal of Interconnection Networks*, *22*(Supp01), 2141026.
7. Walek, B., & Fojtik, V. (2020). A hybrid recommender system for recommending relevant movies using an expert system. *Expert Systems with Applications*, *158*, 113452.
8. Montanés, E., Quevedo, J. R., Díaz, I., & Ranilla, J. (2009). Collaborative tag recommendation system based on logistic regression. *ECML PKDD Discovery Challenge*, 173-188.
9. Kowald, D., Kopeinik, S., & Lex, E. (2017, July). The tagrec framework as a toolkit for the development of tag-based recommender systems. In *Adjunct publication of the 25th conference on user modeling, adaptation and personalization* (pp. 23-28).
10. Tahir, S., Hafeez, Y., Abbas, M. A., Nawaz, A., & Hamid, B. (2022). Smart Learning Objects Retrieval for E-Learning with Contextual Recommendation based on Collaborative Filtering. *Education and Information Technologies*, *27*(6), 8631-8668.