Passau Universität
Seminar Report

Online courses recommendation system

ILHEM MTIR

Contents

Introduction

Since its appearance, the web has been the most used global information medium over the Internet. When it has started, the websites presented the same static information and links to all visitors regardless of their interest neither their background. As it has significantly grown, a huge amount of data was provided which gave the user a wild range of information.
Year after year, the content selectivity became the user's major concern. The Web user found him/her self in front of a huge amount of data while only fewer content was required which has increased the complexity of the search operation.
To solve this problem, recommendation engines were designed and integrated to the web service which relatively shortens the search process for the user and guides him/her to the most useful content customized to his/her profile.
During this seminar work, we have studied the existing types of recommendation systems, the difference between them and finally suggested our recommendation system deployed in the context of Online courses recommendation.
In the next chapters, we will go through the work step by step. In the first chapter, we will present the state of the art by first clarifying what is exactly a recommendation system and giving a general overview of the most popular systems and scenarios where recommendation engine have made a great significant improvement for the general performance.
Next, the following chapter will go throw the different existent recommendation engine and explain each of them, their difference and for which use case which type is the most suitable to use.
The third chapter will present the design of our work, how did we proceed to implement our system and how did we deploy it.
The before-last chapter will contain concrete results and analyses of the performance of our system To finally conclude by a conclusion chapter and perspectives tracing.

1 State of the art

In this chapter, we are going to present what is a recommendation system, give examples of some recommendation systems and how they have improved the performance of the web platform they are integrated into.

1.1 What is a recommendation system?

Recommendation systems are active information filtering systems which personalize the information provided to users based on their interests and predict the preferences or ratings a user would give to a particular item.
There are three main types of recommendation systems we will go through in the following sections.

1.2 Some famous recommendation systems

Taking Netflix example, their dataset consists of more than 17K movies and 500K+ customers. To offer their customers a better user experience regarding this huge amount of movies, Netflix has offered a recommendation system that provides a valuable movies suggestion to their customers based on their interests. This recommendation system takes into account not only the user's information but also the movies they liked.

Another great example that illustrates the efficiency of recommendation systems is the sale forces of the famous book *"**Touching the Void**"*, this book was published in **1988** by *Joe Simpson* about mountaineering. By the beginning, this book has made a decent success, not so many people have bought the book until a few years later by **1997** when *John Krakauer* has published his book *"**Into thin air**"* which is also a book about mountaineering. This book has reached great sales in Amazon, where they noticed that some users who bought the second book have bought the first one as well. Amazon started recommending *"Touching the Void"* to users which made it a bestseller [2].

1.3 Types of recommendation systems

Depending on their different architecture and their customers, the different websites use different types of recommendation systems that give the best user experience to their customers and the best profit to the company.
There are three different types of recommendation systems: popularity based recommendation systems, Content-based recommendation systems, and collaborative based recommendation systems.

1.3.1 Popularity based recommendation systems

This type of systems is the most basic one. Based on the popularity of their items, the most popular one will be recommended to the users. In the context of trading systems, popularity can be measured by the numbers of purchases or the ratings given to each product. This kind of recommendation showed great success and increased the sales of the companies.

1.3.2 Content-based recommendation systems

This type of recommender works with data and metadata provided by a user. Based on the content that this user likes, whether gathered implicitly by tracking the links he/she has clicked on or has explicitly given by rating or reviewing. Information gathered about a user, his/her interests and preferences are used to define a user's profile. For example, the list of items he/she has purchased, his/her best-rated items, ect. all these information helps to define the USER PROFILE.

USER PROFILEs are continuously updated, as people always discover new things that interest them, then, preprocessed where useless data and stop words are filtered out and useful features are extracted and stored in vector representation. On the other hand, an ITEM PROFILE is defined for each item. This vector contains data, metadata, and features describing the item. This information are stored in vector representation.
By the end of this process, a prediction algorithm will be executed to determine the similarity between the USER PROFILE vector and each ITEM PROFILE vector. The tuple providing a higher similarity degree will be the first recommended content.

For calculating the similarity, a great choice of metrics can be used such Jaccard similarity, cosine similarity with TFIDF, or cosine similarity with LSA, etc.

1.3.3 Collaborative Filtering based recommendation system

For this type of recommendation system, there are two approaches: User-To-User and Item-To-Item collaborative filtering. The basic idea behind the first approach is to, first of all, find the neighborhood of a user, then recommend new content that interests the neighborhood.
The neighborhood of a user A is defined by the set of users B who have common interests with A and shown a similar rating to the same content as user A. the graph below illustrates the main idea of that algorithm.

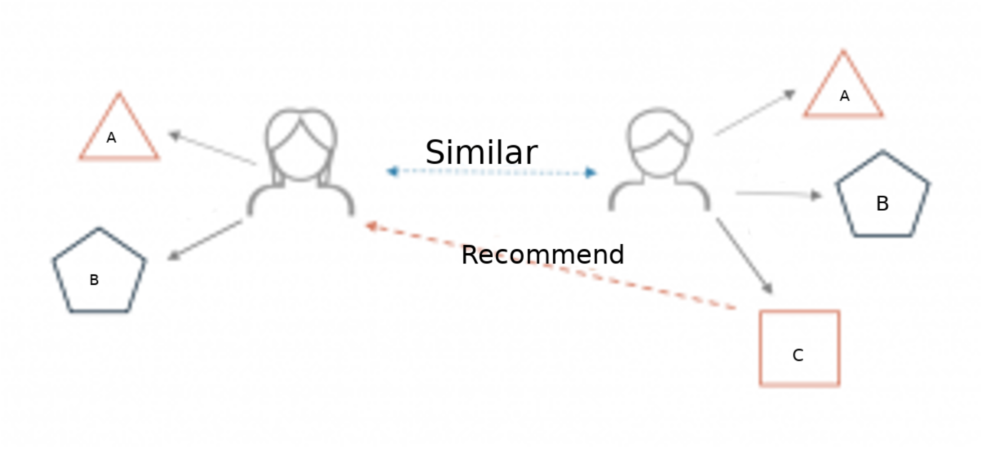


Figure 1 – User-To-User Collaborative filtering

To find the neighborhood, we need a similarity function. Jaccard similarity and cosine similarity are good candidates for this purpose.

The second approach is a similar one, the only difference is that we work with items instead of users. For each item, we look for similar items and estimate their rating based on the ratings for their similar items. For calculating the similarity, we can use the same similarity metrics used for in the first approach.

2 Online Courses recommendation systems

After studying the different types of recommendation systems, We came up with the following choice. During this work, we choosed to implement a Content-based recommendation system that predicts the courses that a certain user, a student, might be interested in based on his/her Profile.

In the following sections, we will go through the work done step by step, specify how did we define the USER PROFILE, THE COURSE PROFILE, then we will present which metric have we used to calculate the similarity. Finally, we will illustrate the results achieved.

2.1 Analyzing user's interests

In order to track the user's interest, we have created a basic HTML platform with a set of general IT topics where the user can pick the topics that he might be interested in.
To ensure the data privacy, the user will be asked to accept the term of uses of the website and guaranteed that his/her data will be only used internally and won't be shared or transferred to any other parties.

Once the user has made his/her choice, these data will be processed to generate the USER PROFILE.

2.2 USER PROFILE creation

Based on these data gathered from the user's topic selection, we will predict the similar topics that he might be interested in as well and suggest some links to courses he/she would like.

As a first step, we preprocess these data by filtering out irrelevant data, stop words, and then create the word vector. When defining the word vector, we assume that the user does not prefer some topics among others. i.e, we won't work with a weighted word vector.

2.3 COURSES PROFILE creation

At this level, we need to define a course profile. To do so, we need to crawl the web to find courses based on them we can make a recommendation.

2.3.1 Crawling the web

There are a bunch of online courses websites, each of them has its own hierarchy and structure. We have chosen the online course web site UDACITY since it has a big variety of topics/courses.
We have used BeatifulSoup Crawler with a depth of 2.
By the end of this phase, we have a collection of links to different online courses which needs preprocessing and filtering.

2.3.2 Data preprocessing

Stop words are a commonly used word (such as “the”, “a”, “an”, “in”) that need to be ignored, we do not want these words taking up valuable processing time or giing misleading results. For this purpose, we used the NLTK library to remove them.
The online courses are in English, so English stop words needs to be imported and filtered out:

```
import nltk
from nltk.corpus import stopwords
 set(stopwords.words('english'))
```

Figure 2 – Stop words removal

Now that the data are preprocessed, We proceed to the next step.

2.3.3 Information retrieval

At this level, we need to retrieve the useful information from our COURSE PROFILE and convert them from a row representation to a matrix representation.

Using TFIDF (Term Frequency-Inverse Term Frequency) heuristic, we generated the matrix that associates to each word in the data set a score/weight. This step is equivalent to a CountVectorizer followed by TfidfTransformer. Weights are defined using the following formula:

$$w_{ij} = tf_{ij} \times \log_2 \frac{N}{n} \text{ , where}$$

w_{ij} = weight of term T_j in document D_i
tf_{ij} = frequency of term T_j in document D_i
N = number of documents in collection
n = number of documents where T_j occurs at least once

Figure 3 – TF-IDF formula

The output of this step is a sparse matrix of type ’<class ’numpy.float64’>’ with stored elements in Compressed Sparse Row format.
Then, using Scikit learn library, we use truncated SVD (aka LSA) for Dimensionality Reduction. This transformer performs linear dimensionality reduction by means of truncated singular value decomposition (SVD).
And now our matrix is ready for the similarity calculation.

2.4 Similarity Calculation

We used the *Cosine similarity* to calculate the similarity between the USER PROFILE and the COURSE PROFILE. Cosine similarity works in the following way: having the vector representation of the USER PROFILE, for each vector representation of a course, we calculate the angle between the two vectors, the closer the two vectors are, the smaller the angle between them. And, if an angle decreases, the cosine of that angle increases.

The Cosine of the angle between the two vectors models the similarity between that two vectors. i.e., The closer the two vectors are, the bigger the similarity is. Figure 4 explains the cosine similarity and Figure 5 gives the formula used to calculate it:

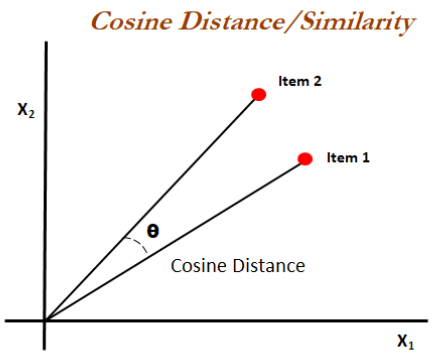


Figure 4 – Cosine Similarity modeling

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum\limits_{i=1}^{n} A_i B_i}{\sqrt{\sum\limits_{i=1}^{n} A_i^2} \sqrt{\sum\limits_{i=1}^{n} B_i^2}},$$

Figure 5 – Cosine Similarity Formula

3 Tests and Results

In this section, we will present some scenarios tested on our online courses recommendation system.

First of all, from UDACITY web site we have crawled all the links to the existent courses there and extracted all the feature and set the TFIDF matrix.

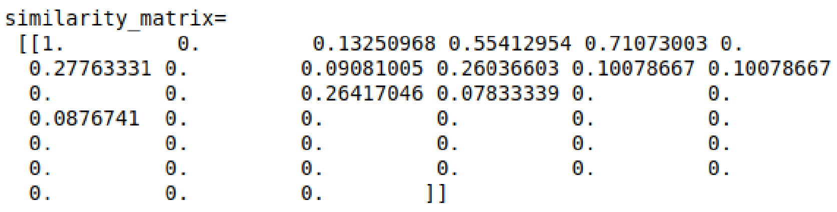
Next, we have asked some users to use the web platform and select some courses that interest them. By clicking on the topic that the user find interesting, the feature will be extracted and set into a vector.

3.1 First scenario

The First user has chosen the following topics:
'Python programming, Deep learning, Artificial neural network, Machine learning, data science'

These data were processed to the recommendation engine then, stop words were removed and the user profile was defined.

Using the cosine similarity, we calculated the similarity between the features extracted from the online courses into the TFIDF matrix and the feature extracted from the user's choice. Finally, we got the following similarity matrix:

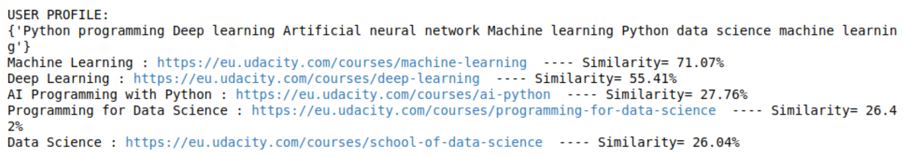


```
similarity_matrix=
 [[1.         0.         0.13250968 0.55412954 0.71073003 0.
  0.27763331 0.         0.09081005 0.26036603 0.10078667 0.10078667
  0.         0.         0.26417046 0.07833339 0.         0.
  0.0876741  0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.         0.         0.         0.
  0.         0.         0.        ]]
```

Figure 6 – Similarity matrix of User 1

As we can see, it is a sparse matrix with some non-Null values which corresponds to similarity values. The similarity is a value in [0,1] where 0 means no match and 1 is a perfect match.
Here, the first value is 1, which is so predictable since we are comparing the match between the USER PROFILE and itself.
We need to sort this matrix in descendent order to get the biggest similarity on top. Once the matrix was sorted, we return the courses corresponding to that similarity value and the expected recommendation.

The matrix was sorted and returned the following courses recommendation in Figure 7 as Top 5 similar courses:



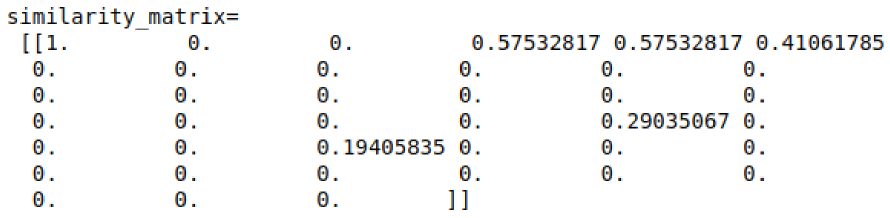
```
USER PROFILE:
{'Python programming Deep learning Artificial neural network Machine learning Python data science machine learnin
g'}
Machine Learning : https://eu.udacity.com/courses/machine-learning  ---- Similarity= 71.07%
Deep Learning : https://eu.udacity.com/courses/deep-learning  ---- Similarity= 55.41%
AI Programming with Python : https://eu.udacity.com/courses/ai-python  ---- Similarity= 27.76%
Programming for Data Science : https://eu.udacity.com/courses/programming-for-data-science  ---- Similarity= 26.4
2%
Data Science : https://eu.udacity.com/courses/school-of-data-science  ---- Similarity= 26.04%
```

Figure 7 – Recommended courses for User 1

3.2 Second scenario

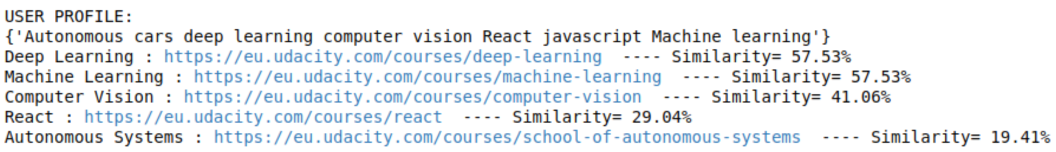
The second user has provided the following choice:

"autonomous cars, deep learning, computer vision, React JavaScript, Machine learning"
And he got the similarity matrix in Figure 8 and got, as a result, the courses recommended in Figure 9.



```
similarity_matrix=
 [[1.          0.          0.          0.57532817 0.57532817 0.41061785
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.          0.          0.29035067 0.
  0.          0.          0.19405835 0.          0.          0.
  0.          0.          0.          0.          0.          0.
  0.          0.          0.         ]]
```

Figure 8 – Similarity matrix for User 2



```
USER PROFILE:
{'Autonomous cars deep learning computer vision React javascript Machine learning'}
Deep Learning : https://eu.udacity.com/courses/deep-learning  ---- Similarity= 57.53%
Machine Learning : https://eu.udacity.com/courses/machine-learning  ---- Similarity= 57.53%
Computer Vision : https://eu.udacity.com/courses/computer-vision  ---- Similarity= 41.06%
React : https://eu.udacity.com/courses/react  ---- Similarity= 29.04%
Autonomous Systems : https://eu.udacity.com/courses/school-of-autonomous-systems  ---- Similarity= 19.41%
```

Figure 9 – Recommended courses for User 2

Conclusion and Perspectives

During this Seminar work, we learned what is a recommendation system, what types of recommendation engine exist and the difference between them. Then, we made the choice of implementing a content-based recommendation system and deploy it in the context of online courses recommendation that helps IT-students choosing content that might interest them.
We proceeded by defining a USER PROFILE, and crawled the web to get a list of online courses based on which we will make our future recommendation.
Preprocessing data was an important step. Then, using some data science libraries, we proceeded by filtering out useless data such as stop words, keeping the useful feature and saving them into a matrix format using the TFIDF vectorizer.
By using the cosine similarity metric, We have achieved an acceptable recommendation results where the system returned 5 relevant online courses with a similarity value that goes from 30% to 71% for the tested scenarios.

The similarity that we got is not too high for some reasons: First, because of the used data set. Since we have used only one online courses website for our Web crawler, we cannot expect a big variety of courses that we can recommend based on them. Second, the USER PROFILE was defined based on the choices that the user made on the provided simple HTML platform.
For a seminar project, these results are acceptable but these could be good perspectives for this work:

- To improve the USER PROFILE creation by tracking the user's interest from different sides (not only the HTML platform)
- To crawl much more websites and enlarge the used online courses data set.

Abbreviations

TFIDF	Term Frequency Inverse Document Frequency
SVD	Singular Value Decomposition
NLTK	Natural Language ToolKit
LSA	Latent Semantic Analysis

List of Figures

References

[1] https://eu.udacity.com/courses/all
[2] www.wired.com/2004/10/tail/
[3] https://www.bluepiit.com/blog/classifying-recommender-systems/
[4] Recommendation as Classification: Using Social and Content-Based Information in Recommendation, Chumki Basu, Haym Hirsh, William Cohen
[5] Content-based Recommender Systems: State of the Art and Trends, Pasquale Lops, Marco de Gemmis, Giovanni Semeraro, Springer Link.
[6] content-based, collaborative recommendation, Marko Balabanović, Yoav Shoham,Stanford University, Stanford, Calif.