Recommender Systems
Content Based methods

Attention This week has been extracted from the given reference text on the recommendation systems. It is available on the BrightSpace.

Recommendation Systems

There is an extensive class of Web applications that involve predicting user responses to options. Such a facility is called a recommendation system. However, to bring the problem into focus, two good examples of recommendation systems are:

1 Offering news articles to on-line newspaper readers, based on a prediction of reader interests.

2 Offering customers of an on-line retailer suggestions about what they might like to buy, based on their past history of purchases and/or product searches.

Recommendation systems use a number of different technologies. We can classify these systems into two broad groups.

. **Content-based** systems examine properties of the items recommended. For instance, if a Netflix user has watched many cowboy movies, then recommend a movie classified in the database as having the ”cowboy” genre.

. **Collaborative filtering** systems recommend items based on similarity measures between users and/or items. The items recommended to a user are those preferred by similar users. This sort of recommendation system can use the groundwork laid in similarity search and clustering.

However, these technologies by themselves are not sufficient, and there are some new algorithms that have proven effective for recommendation systems.

- **The Utility Matrix**: In a recommendation-system application there are two classes of entities, which we shall refer to as *users* and *items*. Users have preferences for certain items, and these preferences must be teased out of the data. The data itself is represented as a utility matrix, giving for each user-item pair, a value that represents what is known about the degree of preference of that user for that item. Values come from an ordered set, e.g., integers 1-5 representing the number of stars that the user gave as a rating for that item. We assume that the matrix is sparse, meaning that most entries are "unknown". An unknown rating implies that we have no explicit information about the users preference for the item.

  Example 1 : In the below figure we see an example utility matrix, representing users' ratings of movies on a 1-5 scale, with 5 the highest rating. Blanks represent the situation where the user has not rated the movie. The movie names are HP1, HP2, and HP3 for Harry Potter I, II, and III, TW for Twilight, and SW1, SW2, and SW3 for Star Wars episodes 1, 2, and 3. The users are represented by capital letters A through D.

	HP1	HP2	HP3	TW	SW1	SW2	SW3
A	4			5	1		
B	5	5	4				
C		2	4	5			
D		3					

  Notice that most user-movie pairs have blanks, meaning the user has not rated the movie. In practice, the matrix would be even sparser, with the typical user rating only a tiny fraction of all available movies.

  The goal of a recommendation system is to predict the blanks in the utility matrix. For example, would user A like SW2? There is little evidence from the tiny matrix in the Figure We might design our recommendation system to take into account properties of movies, such as their producer, director, stars, or even the similarity of their names. If so, we might then note the similarity between SW1 and SW2, and then conclude that since A did not like SW1, they were unlikely to enjoy SW2 either. Alternatively, with much more data, we might observe that the people who rated both SW1 and SW2 tended to give them similar ratings. Thus, we could conclude that A would also give SW2 a low rating, similar to A's rating of SW1.

We should also be aware of a slightly different goal that makes sense in many applications. It is not necessary to predict every blank entry in a utility matrix. Rather, it is only necessary to discover some entries in each row that are likely to be high. In most applications, the recommendation system does not offer users a ranking of all items, but rather suggests a few that the user should value highly. It may not even be necessary to find all items with the highest expected ratings, but only to find a large subset of those with the highest ratings.

Summary 1 If a utility function defined as $u : X \times S \longrightarrow R$ where X is set of costumers and S is a set of items. The goal is to find the utility function such that (check example 1):

– R is a set of ratings with a rating type. This rating types can be either notes $1-5$ or a real number from $[0, 1]$

– R is a total ordered set

Summary 2 The main problems in using utility matrices for designing a recommender systems are:

– Gathering "known" ratings for matrix
  it means how to corroborate the data in the utility matrix.

– Extrapolate unknown ratings from known ratings
  This method is mainly interested in high unknown ratings i.e. this method is not interested in what the user doesn't like but in what the user likes.

– Evaluating extrapolation methods
  how to measure success/performance of recommendation methods.

Notice for generating a utility matrix, we need to gather the ratings of costumers on the items. Gathering ratings can be done in two ways which each one has its advantages and disadvantages:

- Explicit
  Ask people to rate items
  Doesn't work well in practice people cant be bothered
- Implicit
  Learn ratings from user actions e.g., purchase implies high rating
  What about low ratings? How can be extracted from the user actions?

- **Applications of Recommendation Systems**
  Here we present a list of applications for the recommendation systems:

  - *Product Recommendations*: Perhaps the most important use of recommendation systems is at on-line retailers. Amazon or similar on-line vendors strive to present each returning user with some suggestions of products that they might like to buy. These suggestions are not random, but are based on the purchasing decisions made by similar customers or on other techniques we shall discuss in this chapter.
  - *Movie Recommendations*: Netflix offers its customers recommendations of movies they might like. These recommendations are based on ratings provided by users, much like the ratings suggested in the example utility matrix of the above Figure. The importance of predicting ratings accurately is so high, that Netflix offered a prize of one million dollars for the first algorithm that could beat its own recommendation system by 10%. The prize was finally won in 2009, by a team of researchers called "Bellkor's Pragmatic Chaos," after over three years of competition.
  - *News Articles*: News services have attempted to identify articles of interest to readers, based on the articles that they have read in the past. The similarity might be based on the similarity of important words in the documents, or on the articles that are read by people with similar reading tastes. The same principles apply to recommending blogs from among the millions of blogs available, videos on YouTube, or other sites where content is provided regularly.

All in all The main problem in recommender system is to predict (learn) the blank part of the utility matrix. This matrix is sparse and this makes it more difficult.

Content-Based Recommendations

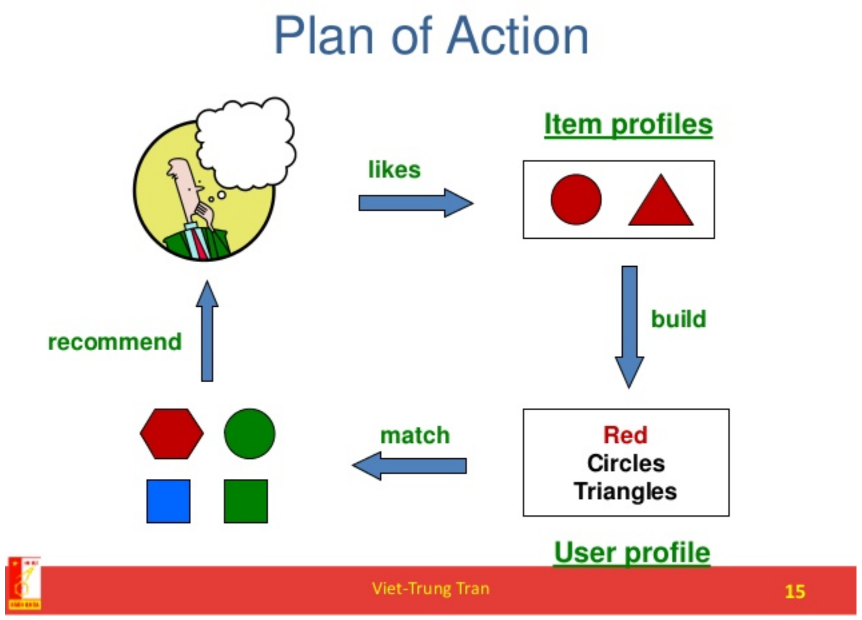

Figure 1:

Content-Based systems focus on properties of items. Similarity of items is determined by measuring the similarity in their properties.

Summary 3 In general, the main idea of the content based is to recommend items to the user x similar to previous items highly rated by the same user x (See figure 1).

- **Item Profile** In a content-based system, we must construct for each item a profile, which is a record or collection of records representing important characteristics of that item. In simple cases, the profile consists of some characteristics of the item that are easily discovered. For example, consider the features of a movie that might be relevant to a recommendation system.

1 The set of actors of the movie. Some viewers prefer movies with their favorite actors.

2 The director. Some viewers have a preference for the work of certain directors.

3 The year in which the movie was made. Some viewers prefer old movies; others watch only the latest releases.

4 The genre or general type of movie. Some viewers like only comedies, others dramas or romances.

There are many other features of movies that could be used as well. However, movie reviews generally assign a genre from a set of commonly used terms. For example the *Internet Movie Database (IMDB)* assigns a genre or genres to every movie. Each product can have different set of features for instance books have descriptions similar to those for movies, so we can obtain features such as author, year of publication, and genre. Music products such as CD's and MP3 downloads have available features such as artist, composer, and genre.

- **TF.IDF, Jaccard distance and cosine similarity**:
  Before continuing with the rest of this section several concepts definition are needed. We explain them her:

  - **tf-idf**[1]: short for term frequencyinverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus.

    In the case of the term frequency $tf(t,d)$ the simplest choice is to use the raw count of a term in a document, i.e., the number of times that term t occurs in document d. If we denote the raw count by $f_{t,d}$, then the tf scheme is $tf(t,d) = f_{t,d}$.

    Term Frequency: Suppose we have a set of English text documents and wish to rank which document is most relevant to the query, "the brown cow". A simple way to start out is by eliminating documents that do not contain all three words "the", "brown",

[1]This definition has taken from the wikipedia page `https://en.wikipedia.org/wiki/Tf-idf`

and "cow", but this still leaves many documents. To further distinguish them, we might count the number of times each term occurs in each document; the number of times a term occurs in a document is called its term frequency.
Inverse document frequency: Because the term "the" is so common, term frequency will tend to incorrectly emphasize documents which happen to use the word "the" more frequently, without giving enough weight to the more meaningful terms "brown" and "cow". The term "the" is not a good keyword to distinguish relevant and non-relevant documents and terms, unlike the less-common words "brown" and "cow". Hence an inverse document frequency factor is incorporated which diminishes the weight of terms that occur very frequently in the document set and increases the weight of terms that occur rarely. The weight of a term that occurs in a document is simply proportional to the term frequency

It is a measure of how much information the word provides, i.e., if it's common or rare across all documents. It is the logarithmically scaled inverse fraction of the documents that contain the word (obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient):

$$idf(t, D) = log\frac{N}{\{d \in D : td\}}$$

with:
N: total number of documents in the corpus $N = |D|$
$\{d \in D : t \in d\}$: number of documents where the term t appears (i.e. $tf(t, d) \neq 0$). If the term is not in the corpus, this will lead to a division-by-zero. It is therefore common to adjust the denominator to $1 + \{d \in D : td\}$

– **Jaccard Distance**[2]: The *Jaccard index*, also known as Intersection over Union and the Jaccard similarity coefficient, is a statistic used for comparing the similarity and diversity of sample sets. The Jaccard coefficient measures similarity between finite sample sets,

[2]This definition has taken from the wikipedia page `https://en.wikipedia.org/wiki/Jaccard_index`

and is defined as the size of the intersection divided by the size of the union of the sample sets:

$$J(A,B) = \frac{|A \cup B|}{|A \cap B|}$$

If A and B are both empty, we define $J(A,B) = 1$. Thus $0 \leq J(A,B) \leq 1$
The *Jaccard distance*, which measures dissimilarity between sample sets, is complementary to the Jaccard coefficient and is obtained by subtracting the Jaccard coefficient from 1, or, equivalently, by dividing the difference of the sizes of the union and the intersection of two sets by the size of the union:

$$d_J(A,B) = 1 - J(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

- **cosine similarity**[3]:is a measure of similarity between two non-zero vectors of an inner product space that measures the cosine of the angle between them. The cosine of 0 is 1, and it is less than 1 for any angle in the interval $(0, \pi]$ radians. It is thus a judgment of orientation and not magnitude: two vectors with the same orientation have a cosine similarity of 1, two vectors oriented at $\frac{\pi}{2}$ relative to each other have a similarity of 0, and two vectors diametrically opposed have a similarity of -1, independent of their magnitude.

  The cosine of two non-zero vectors can be derived by using the Euclidean dot product formula:

$$A \cdot B = cos(\theta) = \frac{A \cdot B}{||A|| \cdot ||B||} = \frac{\sum_{i=1} A_i B_i}{\sqrt{\sum_{i=1}^{N} A_i^2}\sqrt{\sum_{i=1}^{N} B_i^2}}$$

- **Discovering Features of Documents**
  There are other classes of items where it is not immediately apparent what the values of features should be. We shall consider two of

[3]This definition has taken from the wikipedia page `https://en.wikipedia.org/wiki/Cosine_similarity`

them: document collections and images. Documents present special problems, and we shall discuss the technology for extracting features from documents in this section.

There are many kinds of documents for which a recommendation system can be useful. These classes of documents do not tend to have readily available information giving features. A substitute that has been useful in practice is the **identification of words that characterize the topic of a document**. Before discovering any features, we should clean the data.

Data Cleaning
For the text based data, there are several cleaning data methods[4]

- Filter Out Punctuation: We can filter out all tokens that we are not interested in, such as all standalone punctuation, for instance : , ; " ? ! and etc
- Stop words are those words that do not contribute to the deeper meaning of the phrase. They are the most common words such as: "the",'a", and "is" and etc.
- Stem Words: Stemming refers to the process of reducing each word to its root or base. For example "fishing," "fished," "fisher" all reduce to the stem "fish".

Each programming language may have a package for cleaning the text data. The most used tool is NLTK in python.

After, eliminating stop words –the several hundred most common words, which tend to say little about the topic of a document. For the remaining words, compute the tf-idf score for each word in the document. The ones with the highest scores are the words that characterize the document.

We may then take as the features of a document the n words with the highest tf-idf scores. It is possible to pick n to be the same for all documents, or to let n be a fixed percentage of the words in the document. We could also choose to make all words whose tf-idf scores are above a given threshold to be a part of the feature set.

[4]For more detail reed this link: `https://machinelearningmastery.com/clean-text-machine-learning-python/`

summary Each item, product (for instance movie) requires an item profile.

- a profile contains a set of features (for instance for the movie):
  - movies: director, title, actors, country, ...
  - text: set of important words in document
- How to pick important words? using the tf-idf approach
  each document profile is a set of words with highest tf.idf scores, together with their scores.

There is no harm if some components of the vectors are boolean and others are real-valued or integer-valued. We can still compute the cosine distance between vectors, although if we do so, we should give some thought to the appropriate scaling of the nonboolean components, so that they neither dominate the calculation nor are they irrelevant.

- **User Profiles**
  We not only need to create vectors describing items; we need to create vectors with the same components that describe the user's preferences. We have the utility matrix representing the connection between users and items. Recall the nonblank matrix entries could be just 1's representing user purchases or a similar connection, or they could be arbitrary numbers representing a rating or degree of affection that the user has for the item.

  Summary For presenting user profiles as a vector we will introduce two possible approaches:

  - weighted average of rated item profiles
  - variation: weight by difference from average rating for items.

  weighted average:
  With this information, the best estimate we can make regarding which items the user likes is some aggregation of the profiles of those items. If the utility matrix has only 1's, then the natural aggregate is the average of the components. of the vectors representing the item profiles for the items in which the utility matrix has 1 for that user.

  Example 2 Suppose items are movies, represented by boolean profiles with components corresponding to actors. Also, the utility matrix has a 1 if the user has seen the movie and is blank otherwise. If 20% of the movies that user U likes have Julia Roberts as one of the actors, then the user profile for U will have 0.2 in the component for Julia Roberts.

variation:
If the utility matrix is not boolean, e.g., ratings 1–5, then we can weight the vectors representing the profiles of items by the utility value. It makes sense to normalize the utilities by subtracting the average value for a user. That way, we get negative weights for items with a below-average rating, and positive weights for items with above-average ratings.

Example 3 Such as the proposed movie dataset at the beginning of the exercise, now suppose the utility matrix has non blank entries that are ratings in the 1–5 range. Suppose user U gives an average rating of 3. There are three movies with Julia Roberts as an actor, and those movies got ratings of 3, 4, and 5. Then in the user profile of U , the component for Julia Roberts will have value that is the average of $3 - 3 = 0$, $4 - 3 = 1$, and $5 - 3 = 2$, with the average value of 1. On the other hand, user V gives an average rating of 4, and has also rated three movies with Julia Roberts (it doesnt matter whether or not they are the same three movies U rated). User V gives these three movies ratings of 2, 3,and 5. The user profile for V has, in the component for Julia Roberts, the average of $2 - 4 = -2$, $3 - 4 = -1$, and $5 - 4 = 1$, with the average value $-2/3$.

Average or variation method should be computed for each property of your item. For instance, for the movie example [title, year, country, director, genre, location] and a user who rated Toy Story and GoldenEye for the first element (title) as his/her movies; if again you assume that in all your database you have a list of the following movies:

– Toy story
– Jumanji
– Grumpy Old Men
– Waiting to Exhale
– Father of the Bride Part II

  - Heat
  - Sabrina
  - Tom and Huck
  - Sudden Death
  - GoldenEye

  you can generate a vector as [value, 0, 0, 0, 0, 0, 0, 0, 0, 0, another value] where values are calculated from average or variation methods.

  At the end, you should have a vector for each user with the same size as the item vectors.

- **Recommending Items to Users Based on Content**
  With profile vectors for both users and items, we can estimate the degree to which a user would prefer an item by computing the cosine distance between the user's and item's vectors.

  Definition A prediction heuristic for a given user profile c and item profile s can b defined as: $u(c, s) = cos(c, s) = \frac{c \cdot s}{|c||s|}$