



# A Survey Of Collaborative Filtering Based On Nearest-Neighbors

**TANG ZHI-HANG**

School of Computer and Communication  
Hunan Institute of Engineering  
Xiangtan 411104, China

**SHU YUAN-JIE**

School of Computer and Communication  
Hunan Institute of Engineering  
Xiangtan 411104, China

**OUYANG WENMIN**

School of Computer and Communication  
Hunan Institute of Engineering  
Xiangtan 411104, China

**Abstract**—This paper explains the k-NN classification algorithm and its operator in RapidMiner. The Use Case of this chapter applies the k-NN operator on the Teacher Evaluation dataset. The operators explained in this chapter are: Read URL, Rename, Numerical to Binominal, Numerical to Polynomial, Set Role, Split Validation, Apply Model, and Performance. The k-Nearest Neighbor algorithm is based on learning by analogy, that is, by comparing a given test example with the training examples that are similar to it. The training examples are described by n attributes. Each example represents a point in an n-dimensional space. In this way, all of the training examples are stored in an n-dimensional pattern space. When given an unknown example, the k-nearest neighbor algorithm searches the pattern space for the k training examples that are closest to the unknown example. These k training examples are the k “nearest neighbors” of the unknown example. The “Closeness” is defined in terms of a distance metric, such as the Euclidean distance.

**Keywords**- Recommender systems, Collaborative-based Systems, nearest neighbour.

## I. INTRODUCTION

The growth of the Internet has made it much more difficult to effectively extract useful information from all the available online information. The overwhelming amount of data necessitates mechanisms for efficient information filtering. One of the techniques used for dealing with this problem is called collaborative filtering. The motivation for collaborative filtering comes from the idea that people often get the best recommendations from someone with similar tastes to themselves. Collaborative filtering explores techniques for matching people with similar interests and making recommendations on this basis. Collaborative filtering algorithms often require (1) users' active participation, (2) an easy way to represent users' interests to the system, and (3) algorithms that are able to match people with similar interests. Typically, the workflow of a collaborative filtering system is: 1. A user expresses his or her preferences by rating items (e.g. books, movies or CDs) of the system. These ratings can be viewed as an approximate representation of the user's interest in the corresponding domain. 2. The system matches this user's ratings against other users' and finds the people with most “similar” tastes. 3. With similar users, the system recommends items that the similar users have rated highly but not yet being rated by this user (presumably the absence of rating is often considered as the unfamiliarity of an item). A key problem of collaborative filtering is how to combine and weight the preferences of user neighbours. Sometimes, users can immediately rate the

recommended items. As a result, the system gains an increasingly accurate representation of user preferences over time.

Recommender Systems (RecSys) are software tools and techniques providing suggestions for items to be of use to a user [1]. The suggestions relate to various decision-making processes, such as what items to buy, what music to listen to, or what online news to read.

As e-commerce Web sites began to develop, a pressing need emerged for providing recommendations derived from filtering the whole range of available alternatives. Users were finding it very difficult to arrive at the most appropriate choices from the immense variety of items (products and services) that these Web sites were offering.

The explosive growth and variety of information available on the Web and the rapid introduction of new e-business services (buying products, product comparison, auction, etc.) frequently overwhelmed users, leading them to make poor decisions. The availability of choices, instead of producing a benefit, started to decrease users' well-being. It was understood that while choice is good, more choice is not always better. Indeed, choice, with its implications of freedom, autonomy, and self-determination can become excessive, creating a sense that freedom may come to be regarded as a kind of misery-inducing tyranny [2].

The study of recommender systems is relatively new compared to research into other classical information system tools and techniques. Recommender systems

emerged as an independent research area in the mid-1990s [3]. In recent years, the interest in recommender systems has dramatically increased.

Collaborative filtering (CF) is a technique used by some recommender systems[4]. Collaborative filtering has two senses, a narrow one and a more general one[5]. In general, collaborative filtering is the process of filtering for information or patterns using techniques involving collaboration among multiple agents, viewpoints, data sources, etc. Applications of collaborative filtering typically involve very large data sets. Collaborative filtering methods have been applied to many different kinds of data including: sensing and monitoring data, such as in mineral exploration, environmental sensing over large areas or multiple sensors; financial data, such as financial service institutions that integrate many financial sources; or in electronic commerce and web applications where the focus is on user data, etc. The remainder of this discussion focuses on collaborative filtering for user data, although some of the methods and approaches may apply to the other major applications as well.

In the newer, narrower sense, collaborative filtering is a method of making automatic predictions (filtering) about the interests of a user by collecting preferences or taste information from many users (collaborating). The underlying assumption of the collaborative filtering approach is that if a person A has the same opinion as a person B on an issue, A is more likely to have B's opinion on a different issue x than to have the opinion on x of a person chosen randomly. For example, a collaborative filtering recommendation system for television tastes could make predictions about which television show a user should like given a partial list of that user's tastes (likes or dislikes). Note that these predictions are specific to the user, but use information gleaned from many users. This differs from the simpler approach of giving an average (non-specific) score for each item of interest, for example based on its number of votes.

## II. BASIC CONCEPTION

### 2.1 K-Nearest Neighbors

The algorithm caches all training samples and predicts the response for a new sample by analyzing a certain number (K) of the nearest neighbors of the sample using voting, calculating weighted sum, and so on. The method is sometimes referred to as "learning by example" because for prediction it looks for the feature vector with a known response that is closest to the given vector.

### 2.2 Algorithm

The neighbors are taken from a set of examples for which the correct classification or, in the case of regression, the value of the label is known. This can be thought of as the training set for the algorithm,

though no explicit training step is required. The basic k-Nearest Neighbor algorithm is composed of two steps:

- Find the k training examples that are closest to the unseen example.
- Take the most commonly occurring classification for these k examples or, in the case of regression, take the average of these k label values and assign this value as the label of this unseen example.

These two steps are performed for all the unseen examples i.e., all examples of the training dataset.

### 2.3 The k-NN Operator in RapidMiner

The k-Nearest Neighbor algorithm is implemented by the k-NN operator in RapidMiner. This operator is located at "Modeling/ Classification and Regression/ Lazy Modeling" in the Operators Window. This operator expects an ExampleSet as input and it generates a k-Nearest Neighbor model from the given ExampleSet. This model can be a classification or regression model depending on the given ExampleSet. If the type of the label of the ExampleSet is polynominal or binominal, then this operator generates a classification model. If the type of the label is numerical, then this operator generates a regression model. Some important parameters of this operator are:

- k: This parameter specifies the number of nearest neighbors of the unseen example to look for. This parameter is equivalent to the k variable in the k-Nearest Neighbor algorithm. If the parameter k is set to 1, the example is simply assigned the class of its nearest neighbor.
- weighted vote: This parameter specifies if the votes should be weighted by similarity. If this parameter is set to true, the weight of examples is also taken into account. It can be useful to weight the contributions of the neighbors, so that the nearer neighbors contribute more than the more distant ones.
- measure types: This parameter is used for selecting the type of measure to be used for finding the nearest neighbors. In other words, this parameter specifies the type of measure to use for measuring closeness of examples. The following options are available: mixed measures, nominal measures, numerical measures, and Bregman divergences.

### 2.4 Attributes

This dataset has six attributes (including the label attribute). The data set comes with some basic information but the type and role of attributes is set by the user of the dataset. Even the attribute names are specified by the user. Here is an explanation of the attributes of this dataset:

1. **English Speaker:** This attribute specifies whether or not the TA is a native English speaker. If the value of this attribute is 1 it implies that the TA is a native English speaker. If the value of this attribute is 2 it implies that the TA is not a native English speaker. As this attribute has only two possible values, its type should be set to binominal in RapidMiner. It should be noted that values 1 and 2 are not integer values here; they are used to represent native and non-native English speakers, respectively.
2. **Instructor:** This attribute specifies the course instructor. To hide the identity of the instructors, this attribute is represented by numbers from 1 to 25 (instead of instructor name or id etc.). As there were 25 instructors, this attribute can have 25 possible values i.e., {1, 2, 3, . . . 25}. The type of this attribute should be set to polynomial in RapidMiner. It should be noted that values 1 to 25 are not integer values here; they are used to represent 25 different instructors.
3. **Course:** This attribute specifies the course. To hide the identity of the courses, this attribute is represented by numbers from 1 to 26 (instead of course name or id etc.). As there were 26 courses, this attribute can have 26 possible values, i.e., {1, 2, 3, . . . 26}. The type of this attribute should be set to polynomial in RapidMiner. It should be noted that values 1 to 26 are not integer values here; they are used to represent 26 different courses.
4. **Summer:** This attribute specifies if the course was offered in summer or regular semester. If the value of this attribute is 1 it implies that the course was offered in the summer semester. If the value of this attribute is 2 it implies that the course was offered in regular semester. As this attribute has only two possible values, its type should be set to binominal in RapidMiner. It should be noted that values 1 and 2 are not integer values here; they are used to represent summer and regular semesters, respectively.
5. **Class Size:** This attribute specifies the number of students in the class. The type of this attribute should be set to integer in RapidMiner.
6. **Score Category (label attribute):** This attribute specifies the score category of performance evaluation. The values 1, 2, and 3 indicate that the score was low, medium or high, respectively. As this attribute has three possible values its type should be set to polynomial in RapidMiner. It should be noted that values 1, 2, and 3 are not integer values here; they are used to represent different score categories. The role of this attribute should be set to label because this is the target attribute or the attribute whose value will be predicted by the classification algorithms. The role of all other attributes should be set to regular.

## 2.5 Operators in This Use Case

### 2.5.1 Read URL Operator

The Read URL operator can be very useful if the required dataset is available on the Internet at a particular url. The url of the dataset should be given in the url parameter of the Read URL operator. Connectivity to the Internet is required for this operator to fetch data from the specified url. The column separators parameter is also important. It specifies the column separators in form of a regular expression. In most cases, the default value of the column separators parameter works well. It is important to note that the Read URL operator does not ask the user for meta data. It automatically guesses the types of attributes. The type conversion operators can be used after data import to change the type of the attributes (if required). The Set Role operator can be used to set the role of attributes after the dataset has been imported. Some role and type conversion operators are discussed later in this chapter.

### 2.5.2 Rename Operator

Pre-processing starts in RapidMiner as soon as the data has been loaded into Rapid-Miner. Some data import operators (e.g., Read CSV and Read Excel) import data in such a way that the attributes are assigned the correct name during the data import procedure. In this case, there may be no need for renaming the attributes. Names of attributes do not have any effect on the outcome of the classification model, but it is a good practice to assign meaningful names to attributes so that the attribute names reflect the sort of information stored in them. The most commonly used operators for renaming attributes in RapidMiner are the Rename and Rename by Replacing operators. The Rename operator can be used for renaming one or more attributes of the given dataset. The names of attributes in a dataset should be unique. The Rename operator has no impact on the type or role of an attribute.

### 2.5.3 Numerical to Binominal Operator

The Numerical to Binominal operator changes the type of the selected numeric attributes to binominal type. This operator not only changes the type of selected attributes but it may also change the data of those attribute in the dataset. Binominal attributes can have only two possible values e.g., “true” or “false”. If the value of the selected attribute in an example is between the specified minimal and maximal value, this operator changes it to “false”, otherwise it changes it to “true”. Minimal and maximal values can be specified by the min and max parameters, respectively.

### 2.5.4 Numerical to Polynomial Operator

The Numerical to Polynomial operator changes the type of the selected numeric attributes to

polynomial type. It simply changes the type of selected attributes i.e., every new numerical value is considered to be another possible value for the polynomial attribute. As numerical attributes can have a huge number of different values even in a small range, converting such a numerical attribute to polynomial type will generate a huge number of possible values for the new polynomial attribute. Such a polynomial attribute may not be a very useful one. This operator cannot be used for grouping numerical values into groups represented by a polynomial attribute. For grouping the numerical values discretization operators are used.

### 2.5.5 Set Role Operator

It is extremely important to assign the right role to the attributes in the dataset. Most classification operators will not work if there is no attribute with the label role in the dataset (or even if there are multiple attributes with the label role). It should be made sure that only one attribute (and the right one!) has the label role. In a very basic classification setup all other attributes will have regular roles. If an attribute uniquely identifies examples it can be assigned an id role. RapidMiner provides the Set Role operator for changing the role of the attributes. This operator is used for changing the role of one or more attributes. It is very simple to use. The user only needs to provide the name of the attribute and select the desired role for it. The name is provided in the name parameter and the desired role is selected by the target role parameter. Roles of multiple attributes can be set using the set additional roles parameter.

### 2.5.6 Split Validation Operator

The set of examples that the model is trained on is called the training data set. The trained model is applied on new examples to test the performance of the model; this is known as testing a model. The dataset on which the trained model is tested is called the testing dataset. Testing the model gives an idea of how the model will perform in practice. Testing the model validates the performance of the model.

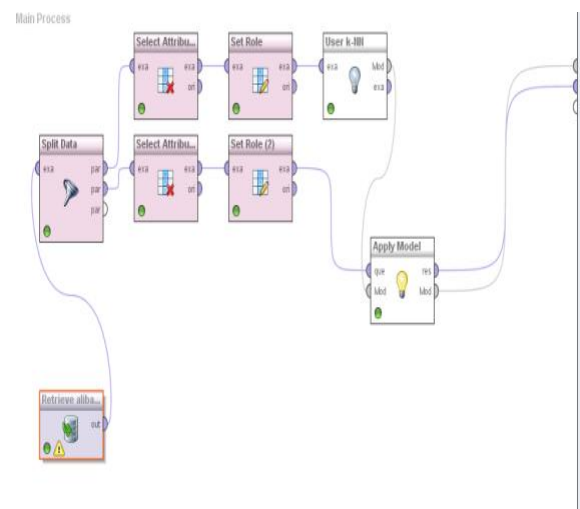
Training and testing is very important part of a classification process. A classification model cannot be applied in practice without knowing the performance (e.g., prediction accuracy) of the model. Testing and training can be implemented in different ways. A very common method is to split the dataset into two portions. One portion is used for training the model and the other portion is used for testing the trained model. Usually the larger portion of the dataset is reserved for training the model.

The Split Validation operator is a nested operator. It has two sub processes: a training sub-process and a testing sub-process. The training sub-process is used for learning or training a model. The classification operators are placed in this sub-process to train a

classification model. The trained model is then applied in the testing sub-process. Mostly, the Apply Model operator is used for applying the trained model. The performance of the model is also measured during the testing phase. Performance measuring operators like the Performance, Performance (Classification), and Performance (Binominal Classification) operators are used for measuring the performance of the model. The given dataset is partitioned into two subsets. One subset is used as the training set and the other one is used as the testing set. The size of two subsets can be adjusted by parameters like split, split ratio, training set size, and test set size parameters. The sampling type parameter is used for selecting the type of sampling. The model is learned on the training set and is then applied on the testing set. This is done in a single iteration, as compared to the X-Validation operator that iterates a number of times using different subsets for testing and training purposes.

## III. PERSONALIZING RECOMMENDER SYSTEMS

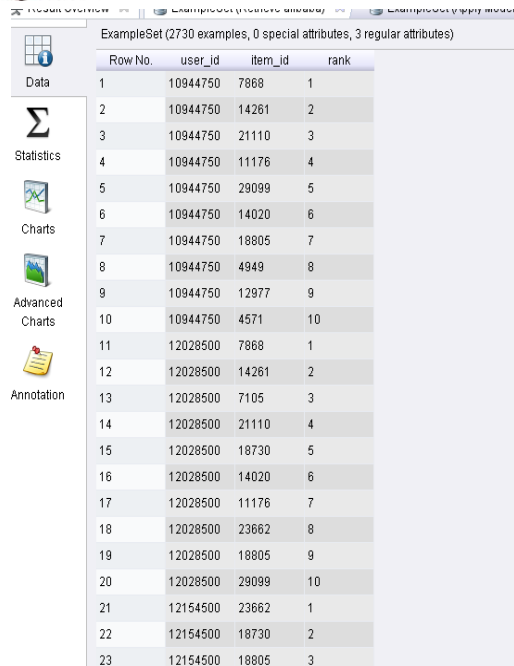
Collaborative recommender operators use the user-item matrix to build a recommendation model. This user-item matrix is presented as an example set of user-item pairs describing user consumption history. The recommendation model built with this matrix is used to recommend items to users from a query set. The query set is an example set containing identification numbers of users for which we want to make recommendations. For each user in the query set we recommend only the items not consumed by this user. Figure 1 depicts a basic collaborative recommender operator workflow.



**Figure 1: An example of an item recommendation workflow**

The Recommended results shown in Figure 2.

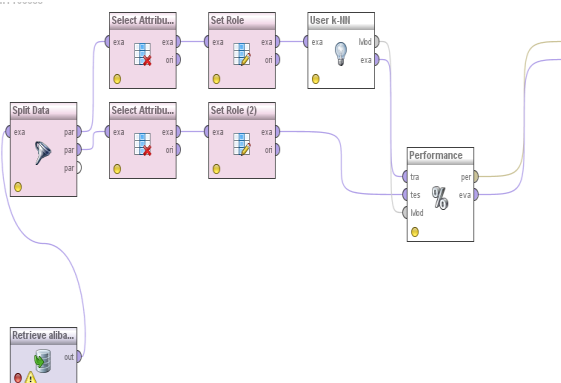




| Row No. | user_id  | item_id | rank |
|---------|----------|---------|------|
| 1       | 10944750 | 7868    | 1    |
| 2       | 10944750 | 14261   | 2    |
| 3       | 10944750 | 21110   | 3    |
| 4       | 10944750 | 11176   | 4    |
| 5       | 10944750 | 29099   | 5    |
| 6       | 10944750 | 14020   | 6    |
| 7       | 10944750 | 18805   | 7    |
| 8       | 10944750 | 4949    | 8    |
| 9       | 10944750 | 12977   | 9    |
| 10      | 10944750 | 4571    | 10   |
| 11      | 12028500 | 7868    | 1    |
| 12      | 12028500 | 14261   | 2    |
| 13      | 12028500 | 7105    | 3    |
| 14      | 12028500 | 21110   | 4    |
| 15      | 12028500 | 18730   | 5    |
| 16      | 12028500 | 14020   | 6    |
| 17      | 12028500 | 11176   | 7    |
| 18      | 12028500 | 23662   | 8    |
| 19      | 12028500 | 18805   | 9    |
| 20      | 12028500 | 29099   | 10   |
| 21      | 12154500 | 23662   | 1    |
| 22      | 12154500 | 18730   | 2    |
| 23      | 12154500 | 18805   | 3    |

**Figure 2 The Recommended results**

In the item recommendation workflow, the first two operators read the train and the query example sets using the Read AML operators (1,4). Following, the appropriate roles are set to attributes using the Set Role operator (2). The user identification role was set to user id attribute and item identification role to item id attribute. Data attributes can have arbitrary names but roles for those attributes must be set. Next, we use the train data with the appropriately set roles to train an Item k-NN model (3). At this point we can use our trained model to recommend new items to users in the query set using the Apply Model operator (6). Prior to model application, the user identification role was set for the query set (5). The Apply Model operator (6) returns an example set containing the first n ranked recommendations for every user in a query set. In Figure 1 we have seen how to make recommendations for particular users. In the following figure 3, we show how to measure performance of a recommendation model.



**Figure 3 Measuring performance of a recommendation model.**

The data management part of the workflow for measuring recommender model performance in Figure 2 is the same as in Figure 3. We use the Read AML operators (1,4) to load the data input, and the Set Role operators (2,5) to set the appropriate roles. In this workflow we use the test data (4) containing two attributes, the user id and the item id attribute and we set user identification and item identification roles to those attributes, respectively. The difference from the previous workflow is the need to calculate the performance of our built recommendation model (3). We use the Performance operator (6) to measure standard recommendation error measures we previously defined: AUC, Prec@k, NDCG, and MAP. The Performance operator (6) returns a performance vector and an example set containing performance measures. This enables a user to choose which format suits his or her needs. We can get Figure 4.



| Row No. | AUC   | prec@5 | prec@10 | prec@15 | NDCG  | MAP   |
|---------|-------|--------|---------|---------|-------|-------|
| 1       | 0.245 | 0.026  | 0.029   | 0.026   | 0.112 | 0.025 |

**Figure 4 The performance of Recommender Systems**

#### IV. CONCLUSIONS

In this paper we explain the k-NN classification algorithm and its operator in RapidMiner. The Use Case of this chapter applies the k-NN operator on the Teacher Evaluation dataset. The operators explained in this chapter are: Read URL, Rename, Numerical to Binominal, Numerical to Polynomial, Set Role, Split Validation, Apply Model, and Performance.

The k-Nearest Neighbor algorithm is based on learning by analogy, that is, by comparing a given test example with the training examples that are similar to it. The training examples are described by n attributes. Each example represents a point in an n-dimensional space. In this way, all of the training examples are stored in an n-dimensional pattern space. When given an unknown example, the k-nearest neighbor algorithm searches the pattern space for the k training examples that are closest to the unknown example. These k training examples are the k “nearest neighbors” of the unknown example. The “Closeness” is defined in terms of a distance metric, such as the Euclidean distance.

## V. ACKNOWLEDGEMENTS

This work was supported by the 2014 science and technology plan of Hunan province (No.2014GK3157).

## VI. REFERENCES

- [1]. Mahmood, T., Ricci, F.: Improving recommender systems with adaptive conversational strategies. In: C. Cattuto, G. Ruffo, F. Menczer (eds.) Hypertext, 2009, pp. 73–82.
- [2]. Schwartz, B.: The Paradox of Choice. ECCO, 2004, New York
- [3]. Anand, S.S., Mobasher, B.: Intelligent techniques for web personalization. In: Intelligent Techniques for Web Personalization, 2005, pp. 1–36. Springer
- [4]. Francesco Ricci and Lior Rokach and Bracha Shapira. Introduction to Recommender Systems Handbook, Recommender Systems Handbook, Springer, 2011, pp. 1-35
- [5]. Terveen, Loren; Hill, Will . "Beyond Recommender Systems: Helping People Help Each Other" . Addison-Wesley. 2012, p. 6.