Applied Intelligence (2022) 52:1602–1614
https://doi.org/10.1007/s10489-021-02429-9

ORIGINAL SUBMISSION

Solving distribution problems in content-based recommendation system with gaussian mixture model

Nguyen Van Dat[1] · Pham Van Toan[1] · Ta Minh Thanh[2]

Accepted: 10 April 2021 / Published online: 25 May 2021

Abstract
Recommendation systems play an important role in boosting purchasing consumption for many manufacturers by helping consumers find the most appropriate items. Furthermore, there are quite range of recommendation algorithms so far that can be efficient; however, a content-based algorithm is always the most popular, powerful, and productive method taken at the begin time of any project. In the negative aspect, somehow the accuracy of content-based algorithm results is still a concern that correlates to probabilistic similarity. In addition, the similarity calculation method is another crucial that affect the accuracy of content-based recommendation in probabilistic problems. In order to solve these problems, we propose a new content-based recommendation based on the Gaussian Mixture Model to improve the accuracy with more sensitive results for probabilistic recommendation problems. Our proposed method is experimented on a liquor dataset including six main flavour tastes, liquor main taste tags, and some other criteria. The method clusters n liquor records relied on n vectors of six dimensions into k group ($k < n$) before applying a formula to sort the results. Compared our proposed algorithm with three other popular models on the above dataset, the accuracy of the experimental results not only outweighs the comparison to those of three other models but also attain a very speedy response time in real-life applications.

Keywords Recommendation system · Content-based · Gaussian Mixture Model (GMM) · Distribution recommendation

1 Introduction

1.1 Overview

Due to the proliferation of the internet, it has brought tremendous chance for people's lives. On the other hand, the myriad and abundance of information on the web has determined a rapidly increasing difficulty in finding what we actually need in a way that can fit the best our requirements [7, 29]. Recommendation systems can be effective way to solve such problems without requiring users provide explicit requirements [8]. Instead, the system can analysis the content data of item properties, which actively recommend information on users that can satisfy their needs and interests. So far, the algorithms applying for a recommendation system are diverse, but it would be grouped as three major basic approaches [22] as Content-Based (CB) [32], Collaborative Filtering (CF) [35] and Knowledge-Based [27] Recommendation System. The general content-based architecture is shown in Fig. 1. How to design an effective recommendation algorithm has become the focus of research.

Content-based filtering algorithm is widely used because of its simplicity and effectiveness, high efficiency at the beginning time of any recommendation systems. According to Pasquale Lops in Chapter-3 Content-based recommendation system: State of the Art and Trends [1], there are many benefits reaped from content-based recommendation systems compared to the other Collaborative Filtering (CF) one such as: user independence, transparency, new items in case of cold-start problems. Beside that, there is still some shortcomings existing as limited content for analyzing, over-specialization or lack of rating data of new users

✉ Ta Minh Thanh
thanhtm@mta.edu.vn

Nguyen Van Dat
nguyen.van.dat@sun-asterisk.com

Pham Van Toan
pham.van.toan@sun-asterisk.com

1 Research and Development Dept, Sun Asterisk, Ha Noi, Viet Nam

2 Le Quy Don Technical University, 236 Hoang Quoc Viet, Cau Giay,Ha Noi, Viet Nam

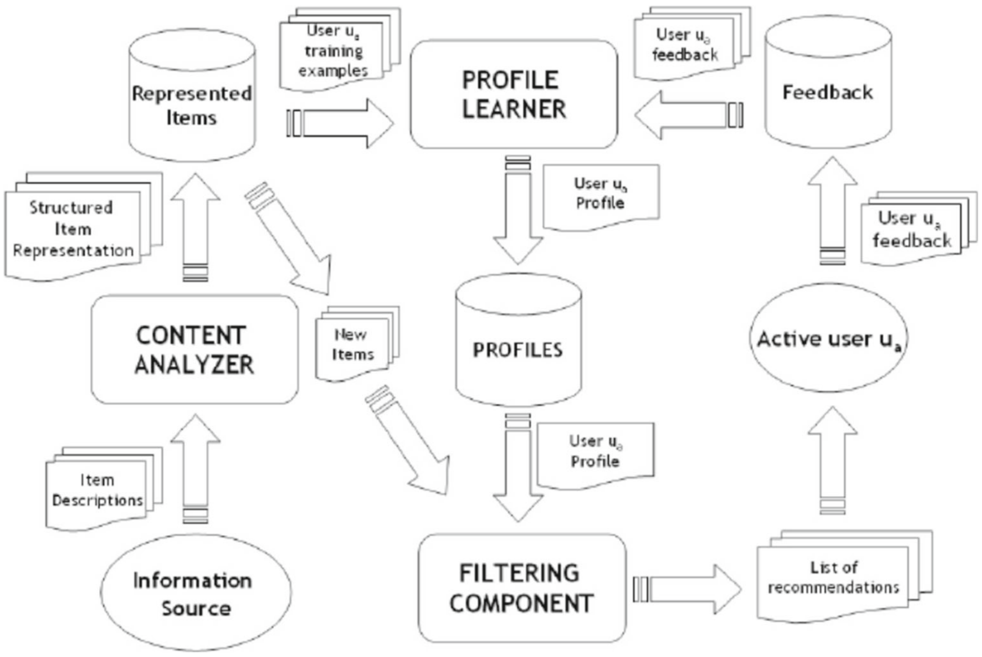


Fig. 1 High level architecture of a Content-based Recommendation

and adequate accuracy for some specific problems. According some available research, Hangyu et al. in [3] used Gaussian mixture model (GMM) for CF recommendation algorithm to solve the sparse user rating data. Chen et al. in [4] proposed a hybrid model, which combines GMM with item-based CF recommendation algorithm and predicted the ratings on items from users to improve the recommendation accuracy. Rui Chen et al. in [5] employed GMM with enhanced matrix factorization to reduce the negative effect of sparse and high dimension data. In the context of music recommender systems, Yoshii et al. [6] proposed a hybrid recommender system that combines collaborative filtering via user ratings and content-based features modeled via GMM over Mel Frequency Cepstrum Coefficients (MFCCs) by utilizing a Bayesian network. However, CF or hybrid systems require behaviour history of users that is the reason for the need of content-based recommendation (CB). Furthermore, CB based on distribution of item features have not been resolved yet. A telling of example is using content-based recommendation for automatically find similar items based on distribution and distance of its features. In Fig. 2, these two bar charts illustrate the deviation between six properties, called $\{f_1, f_2, f_3, f_4, f_5, f_6\}$ of our experimental dataset, belonging to two separated items (one original item (blue: item_1), one suggestion item (orange: item_2) are recommended by taking advantage of sorting formula, distance (the above chart) and distribution (the below chart)). The "Values" of chart shows the similarity of original data and that of suggestion one. These kind of probabilistic problems in recommendation systems is quite different which cannot be solved by usual common methods.

Furthermore, in some situation, the description of content data of items features is not reliable, inadequate, that detrimentally affect to the accuracy of content-based recommendation systems [7]. The other crucial aspect is the inference time that is unsatisfactory for real-life applications.

1.2 Challenging issues

Based on the above explanation, we define our challenge issues as follows:

(i) How to build the recommendation system that solve the distribution problems of item's properties in Content-Based Recommendation?

The conventional content-based recommendation systems are developed based on a matching property algorithm; or are driven by a rating prediction function such that when a new user rates a few items. In general, such CB that are user-independence which usually cannot obtain a high quality suggestion results on distribution of item's properties, and there has no any related-research to resolve the problem. Therefore, the proposal of recommendation system that can take advantage of current probabilistic models to utilize this type of data in the CB, is required.

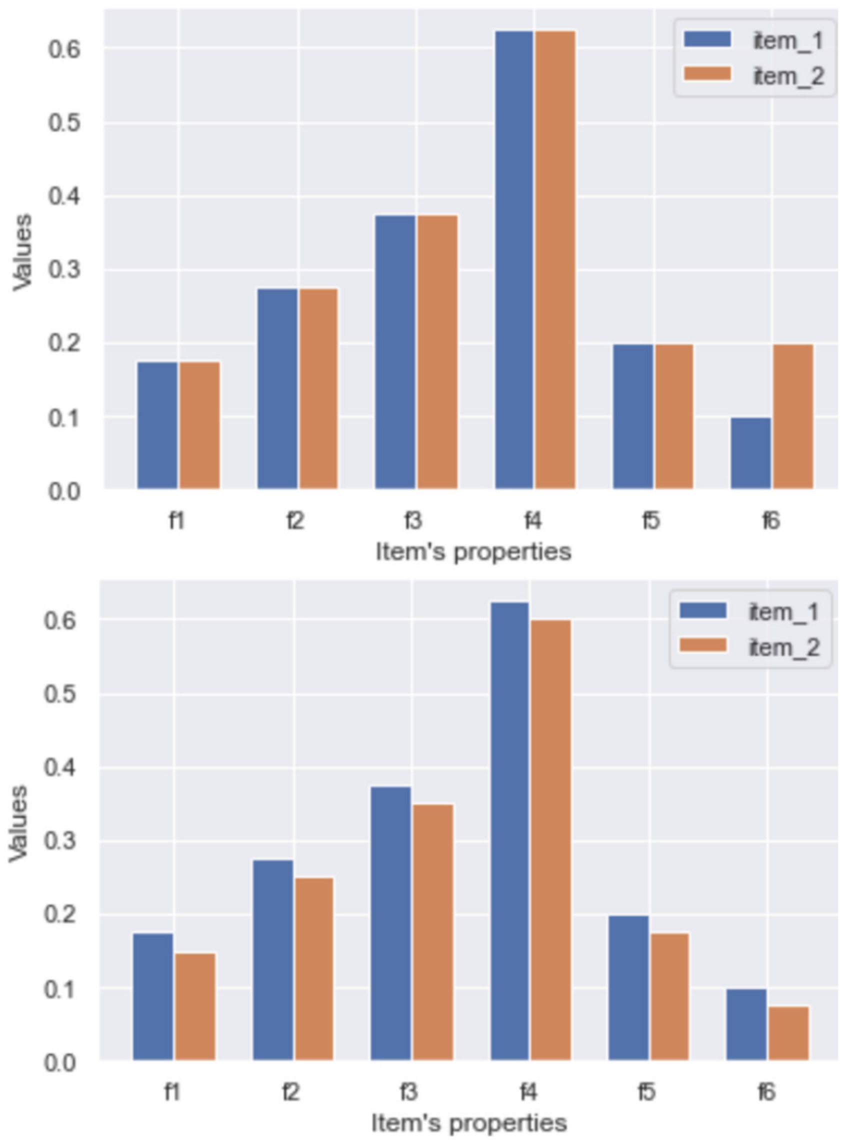


Fig. 2 An example between using distribution and distance calculation for distribution recommendation

It is the time for the probabilistic model like GMM, *etc* that come to light.

(ii) How to gain better performance of content-based recommendation systems both in accuracy and time response to apply for real applications?

Real-time response ability is of great importance to recommendation systems. After any user's interaction on the system, the recommended results are required to show as soon as possible. Our proposed method firstly clusters the prospective items into several groups by GMM, then uses Gaussian Filter Function (GFF) to arrange the recommendation list in order to suggest users real-time.

More specifically, our algorithm leverage the type of continuous data of item's properties, going parallel in reducing the reliance on item's text fields (flavour tags, description, *etc*.) which are sometimes inadequate, precise enough to detrimentally affect the recommendation results.

(iii) How to verify the efficiency of our proposed method comparing with other recommendation systems?

In order to verify the efficiency of our proposed method, we employ the dataset of a real recommendation system using for suggesting Japanese liquors, called Sakenowa. Our recommended results are compared to those of the Sakenowa system, so that it can demonstrate the improvement of our system which become better than the Sakenowa. Beside, we also compare our method to three previous recommendation techniques by accuracy, time reference stats to prove the practicality of ours.

1.3 Our contributions

Due to three problems mentioned above, in this paper we propose a new approach/model for solving these problems by using Gaussian mixture model [9] to cluster all items into different groups relied on distributions of those property before applying a Gaussian Filter function (GFF) as a calculation similarity method for sorting recommendation item results. Our technique have solved the issue (i) and (ii) with significant improvement on accuracy and time reference in the content-based distribution recommendation system by focusing on GMM and GFF.

In order to solve the issue (iii), for demonstrating our effective model, we experiment and compare to three other popular methods, Bag of Word [10] with GFF (BOW + GFF), GMM with euclidean distance (ED) [11] (GMM + ED), and Word2Vec [23] with GFF (W2V + GFF). Our proposed model not only outperform the accuracy of the three others, but it also get better in prediction time response.

1.4 Roadmap

The paper is organized as follows. Related works are introduced in Section 2. In Section 3, the dataset is described while Section 4, the architecture and details of proposed model is given. Experiments and evaluation in Section 5. The conclusion, improvement and future work will be explained in Section 6.

2 Related works

In this section, we introduce some preliminary knowledge that needs to be used in our method. The detailed explanation information can be referred as follows.

2.1 Preliminary knowledge

2.1.1 Popular similarities

In the `Content-based` algorithm, the similarity calculation method directly affects the accuracy of recommended results. Some similarity calculation methods have been widely used which are listed below:

`euclidean distance`: One of the most popular methods to measure the similarity between two vectors by calculating the sum of square distance of each element respectively in those vectors [11].

`Cosin`: The main idea here is to measure two vectors by calculating the cosine of angle between the two vectors [33].

`Pearson`: The pearson correlation coefficent reflects the degree of linear correlation between two vectors [36].

`Jaccard`: The Jaccard similarity is often used to compare similarity and different between two finite sample set [31].

2.1.2 Gaussian Mixture Model (GMM)

Gaussian mixture model is a function that is comprised of several Gaussians, GMM can fit any type of distribution, which is usually used to solve the case where the data in the same set contains multiple different distributions [14]. Each distribution is identified by $k \in \{1..K\}$, where K is the number of clusters of our dataset. Each Gaussian k in the mixture is comprised of the following parameters:

– A mean μ defined its center.
– A covariance Σ that defined its width. This would be equivalent to the dimension of an ellipsoid in a multivariate scenario.
– A mixing probability α that defines how big or small the Gaussian function will be.

GMM is defined as:

$$p(x) = \sum_{i=1}^{k} \alpha_i . N(x|\mu_i, \sigma_i), \tag{1}$$

where $N(x|\mu_i, \Sigma_i)$ is the i^{th} component of the hybrid model, which is a probability density function of the n dimensional random vector x obeying Gaussian distribution and can be defined as below:

$$N(x) = \frac{1}{(2\pi)^{\frac{n}{2}} \mid \Sigma \mid^{\frac{1}{2}}} \epsilon^{-\frac{1}{2}(x-\mu)^T \sum^{-1}(x-\mu)} \tag{2}$$

and

$$\sum_{i=1}^{k} \alpha_i = 1 \tag{3}$$

We assume that a sample set $D = \{x_1, x_2, x_3, ..., x_m\}$ is given that obey Gaussian distribution. We use the random variable $z_j \in \{1, 2, ..., k\}$ to represent the mixed component of the generated sample x_j, whose value is unknown. It can be seen that the prior probability $P(z_j = i)$ of z_j corresponds to $\alpha_i (i = 1, 2, 3, ..., k)$. According to Bayes' theorem [12], we can get the posterior probability of z_j which is defined as follows.

$$p(z_j = i|x_j) = \frac{P(z_j = i).p(x_j|z_j = i)}{p(x_j)} = \frac{\alpha_i . N(x_j|\mu_i, \Sigma_i)}{\sum_{l=1}^{k} \alpha_l . N(x_j|\mu_l, \Sigma_l)} \tag{4}$$

In the above formula, $p(z_j = i|x_j)$ represents the posterior probability of sample x_j generated by the i^{th} Gaussian mixture. Assuming $\gamma_{ij} = \{1, 2, 3, ..., k\}$ to represent $p(z_j = i|x_j)$. When the model parameters $\{(\alpha_i, \mu_i, \Sigma_i)|1 \leq i \leq k\}$

in the above equation are known, the GMM clusters divide the sample set D into k clusters $C = \{C_1, C_2, ..., C_k\}$ [14], and the cluster label λ_j of each sample x_j can be determined according to equation below:

$$\lambda_j = \arg max_{i\in 1,2,3,...,k} \gamma_{ji} \tag{5}$$

We get the cluster label λ_j to which x_j belongs and divides x_j into cluster C_{λ_j}. The model parameters $\{(\alpha_i, \mu_i, \Sigma_i)|1 \leq i \leq k\}$ is solved by applying EM algorithm [13] (Fig. 3).

(i) Related Content-based Recommendation Systems

`Content-Based Algorithm` is one of the most common method in building the recommendation systems. The algorithm is born from the idea of using the content descriptions of each item for recommending purposes. It can be divided into two main approaches: Analysing the description of item properties only; and building user profile for individuals based on content features of items and personal rating data [1]. In next sub-section, we will walk around these two concepts of CB and investigate some related CB systems.

(ii) Analysing the description of item properties only

In case of raw and pure data about item properties, and not personalized recommendation, we can build a system that can return similar items based on the identical properties in each item. For example, we have N records $X_n = \{x_1, x_2, ..., x_N\}$ with x_i containing h properties, $x_i = \{p_1, p_2, ..., p_h\}$. Therein, p_i can be any kind of factor of a product in real-life such as price, tags, content description, brand, and so on. The main conception here is to try to figure out items that have the same in segment of content as much as possible to conclude that is in a group items similar.

(iii) Building user profile for individuals based on the content of items

In this case, it supposes to be that we have C users $U_n = \{u_1, u_2, ..., u_c\}$, n items $X_n = \{x_1, x_2, ..., x_n\}$, and rating data on some items of each user. The main idea here is to take advantage of sparse rating data to predict some of the most possible items that fulfill the best interest for each user profile. The system analyze a set of documents and/or descriptions of items previously rated by users, and build a profile of user interest based on the features of the objects rated by that user. The profile is a structured representation of user interests, adopted to recommend new items. The system process basically consists in matching up the attributes of user profile against the attributes of a content object. The result is a relevance judgment that represents the user's level of interest in that object.

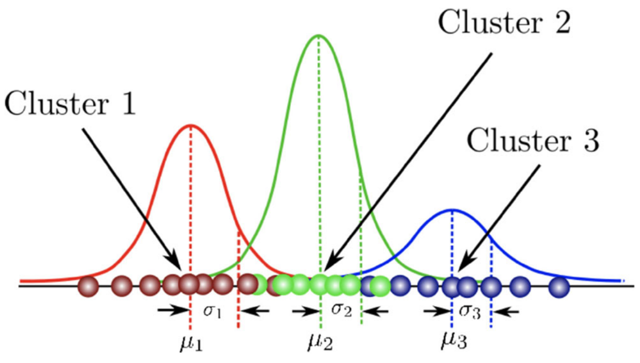


Fig. 3 Gaussian mixture model

(iv) Using Word Embedding in Content-based Recommendation Systems

Being inspired with another approach that is taking care more about the semantic meaning of words rather than just assessing the weight of words in a document by term frequency–inverse document frequency (TF-IDF) weighting. In natural language processing, due to the linguistic ambiguity, item representation by traditional keyword-based is unable to capture the semantic meaning of words because they are primarily driven by a string matching operation that causes exponentially to the accuracy of CB algorithms. As the result, the adoption of using word embedding in CB is to solve this problem. In paper [23], Cataldo Musto et al. developed a CB Recommendation system using textual features extracted from Wikipedia to learn users' profiles based on such word embedding. Their algorithm based on Word Embedding including Latent Semantic Indexing [26] (LSI), Random Indexing [34] (RI) and Word2Vec [30] (W2V) showed results comparable to those of well-performing algorithms based on CF and Matrix Factorization.

(v) Using Bag of Word for Movie Recommendation System

Recently, Bhattacharya et al. [10] have built a simple but potent web-app which take advantage of scikit-learn python library [21] to recommend movies on IMDB movie database [1]. The dataset has many columns; however, they merely take the plot, actors and directors for more accuracy. Firstly, the selected columns are cleaned by some preprocessing techniques then important keywords would be extracted to form bag of word (BoW) matrix. Their application relied on the concept of cosine similarity. The workflow of their web application is referred as Fig. 4. According to the flow of BoW method shown in Fig. 4, it allows word modeling based on dictionaries, where each bag contains a few words from the dictionary. Therefore, the movie recommendation system only used plot, actors

[1]https://datasets.imdbws.com/

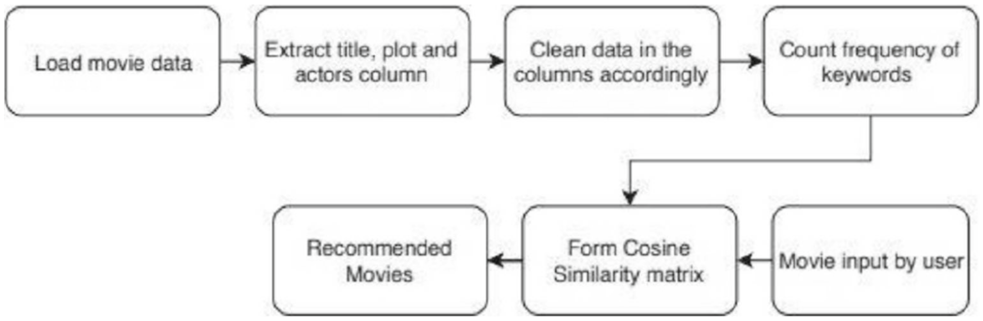

Load movie data
Extract title, plot and actors column
Clean data in the columns accordingly
Count frequency of keywords
Movie input by user
Form Cosine Similarity matrix
Recommended Movies

Fig. 4 Workflow of the web-app application

and directors from the dataset to generate recommendation model. It may not be flexible for big dataset included more important features.

3 Real dataset for our proposed method

Our proposed model is implemented on a dataset about liquors, more specifically, about sake which is one of the most prevalent kind of liquor in Japan. In addition, it was collected from Sakenowa[2] dataset being one of the most well-known and reputed website selling the sake. The dataset totally contains 1072 records characterized by 19 properties such as liquor name, liquor brand, year of manufacture, liquor images, liquor flavour tags, liquor 6-axis flavour tastes ($f_1, f_2, \cdots, f_6$) stands for fruity, mellow, rich, mild, dry and light, and so on.

Noticeably, liquor 6-axis flavour tastes and liquor flavour taste tags would play much more important role than the others as the suggestion and observation from liquor experts. The more proportional distribution between each pair of elements in the 6-axis flavor taste, the better liquor recommendation will be. The range value of 6-axis flavour tastes ($f_1 - f_6$) axis is in [0, 1], meanwhile the dominant parts belong to [0.2, 0.6]. Here is an example 6-axis flavor taste elicited from the dataset $\{f_1$: 0.516103, f_2: 0.484763, f_3: 0.203029, f_4: 0.419481, f_5: 0.276085, f_6: 0.453028$\}$. Such 6-axis flavour tastes values mainly affect to the efficiency of recommendation system in real applications. In the Sakenowa service, we found that, the recommendation results are suggested by using the comparison between 6-axis flavour tastes values.

The text fields in the dataset all is written behind Japanese form. Our task is by somehow actively recommend liquors as much similar as possible when users take a glance to a random liquor. Figure 5 shows that 6-axis flavour tastes have same important role in algorithm of recommendation system.

However, this is a real challenging dataset due to lack of many fields that lead to sparse in data, especially in six main fields f_1, ..., f_6. So, our task of recommendation become more difficult and be negatively reduce the recommendation results. More specifically, a disappearance or null value of 6-axis fields is greater than 30%, a nearly 2% of null value flavour tags. Further more, many tag value is unreliable, untrust and incorrect that need to be cleaned and pre-processed. Table 1 is the table statistic about lack of field's value in the dataset, some other fields is not included in this table.

Based on the analytic from Table 1, the dataset is needed to be pre-processed and adjusted before applying the recommendation algorithms. Some terms in the dataset which is presented by Japanese, also is modified for standardizing data.

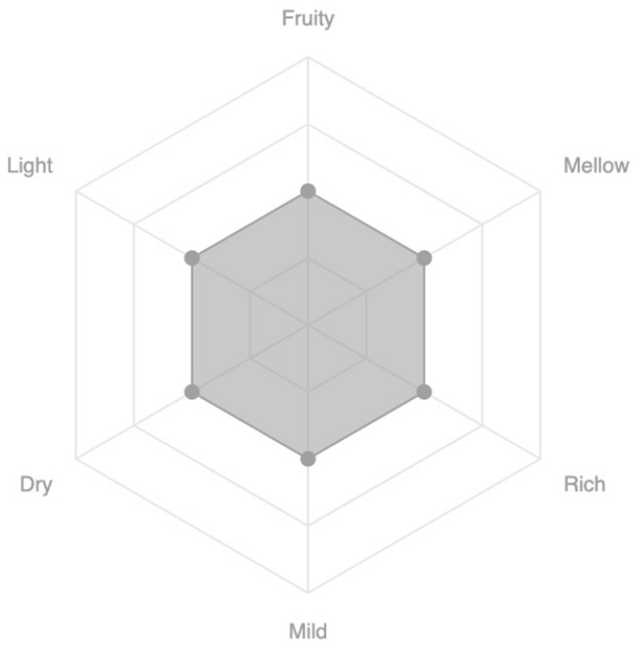


Fig. 5 A visualization about 6-axis flavour taste

[2]https://sakenowa.com

Table 1 Dataset blank fields statistic

$f_{1..6}$	Flavour tags	Product name(en)
Float	String	String
30.4%	1.77%	13.4%

4 Proposed content-based recommendation system

4.1 Proposed model

In this section, we introduce and explain our proposed model in detail. As it was mentioned in previous part, we have to return the most similar products based on 19 meta-data fields. In particular, 6-axis flavour tastes and flavour tags are the main factors mostly affecting to the results both in the sensibility and accuracy side. Therefore, we just select 6-axis flavour tastes and flavour tags for better results. The more similar in 6-axis flavour tastes, the better results will be.

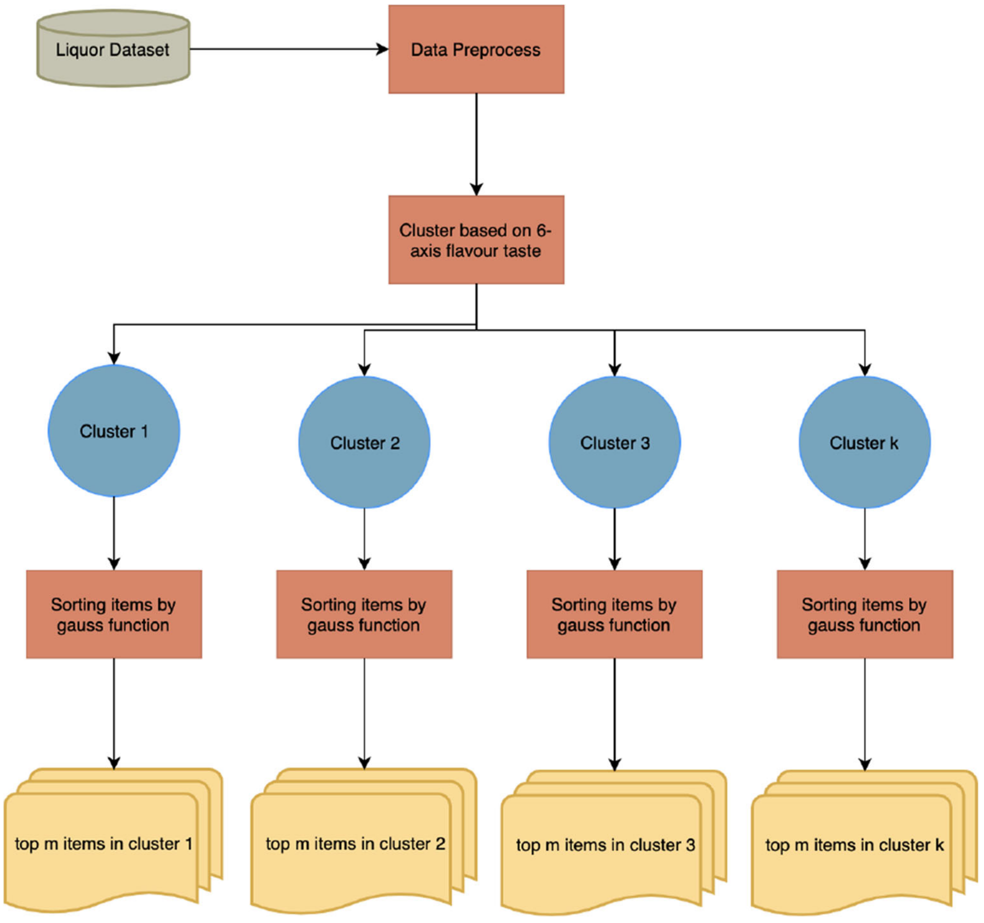
More detail, we initially use GMM to cluster all items into $K = \{1, 2, ..., k\}$ group, then sorting results in each group with each item. Whenever finding top similar items of a item, we just jump up to the group the item belongs to and sort the group's items to return top m similar items. To sort the results, it is also possible to use some popular similarity calculation such as `cosine` or `euclidean distance`. In this paper, for achieving better accuracy, we use a equation that calculates the distribution weight between two vectors obeying Gaussian distribution (normal distribution). The results illustrate that the more similar in 6-axis amongst items the bigger weights will be germinated. The flow of our proposed model is shown in Fig. 6.

4.2 Detailed algorithm

(i) Data pre-processing

It is a fact that text mining is very important in every text-related problems, and CB is not an exception. Previously mentioned, we only choose flavour tags and 6-axis flavour tastes as the features for computing the similar between items. The flavour tags are the set of text document written behind Japanese form which require to be cleaned. We convert 6-axis into float and need to do some pre-process techniques for such flavour tags text

Fig. 6 The model Activities Diagram


Liquor Dataset
Data Preprocess
Cluster based on 6-axis flavour taste
Cluster 1
Cluster 2
Cluster 3
Cluster k
Sorting items by gauss function
Sorting items by gauss function
Sorting items by gauss function
Sorting items by gauss function
top m items in cluster 1
top m items in cluster 2
top m items in cluster 3
top m items in cluster k

fields like tokenization, stemmings, stop word removal, find and replace synonyms, lemmatization, and so on [2, 28] before utilizing it. Moreover, the flavour tags field has been splitted into different semantic words, so we disregard the tokenization step and move forward with the other steps.

(ii) Clustering

As we recognize that the final recommendation items depend too much on 6-axis flavour tastes and flavour tags. In the common and traditional way, there is a way to build a vector representing for all properties of each item, then utilizing a similarity calculation method like cosine or euclidean to sort and return top m results. However, in some case, the flavour tags are not adequate and precise enough that adversely affect to the final recommendation. Moreover, there is always an unseen problem of using $cosine$ or $euclidean$ that a compensate between each element of 6-axis flavour tastes ($f_1 - f_6$) leads to unequal among those elements($f_1 - f_6$) of results. Therefore, we decide to group all items based on it's distribution 6-axis flavour tastes into different clusters to ensure items which have the same distribution will be in the same cluster that is the foundation for sorting afterwards (refer Fig. 7).

(iii) Gaussian Filter Function for sorting

As we have $K = \{1, 2, ..., k\}$ clusters, we assume a query item is the center of the cluster we want to find. Our destination is figure out top m items that have the same distribution as much as possible, so Gaussian filter function (GFF) is the better choice than cosine or euclidean. The Gaussian function equation is defined as follows:

$$G_{kl}(f_{il}, f_{jl}) = \exp - \frac{(f_{il} - f_{jl})^2}{2\sigma_{kl}^2}, \tag{6}$$

where $G_{kl}(f_{il}, f_{jl})$ is considered as a weight between each pair of element l^{th} in 6-axis flavour tastes of two different items (i, j) in cluster $k, l = \{1, 2, ..., 6\}$, and σ_{kl} is the standard deviation of the l^{th} element in 6-axis flavour tastes in group k. Equation for σ_{kl} is defined as follows:

$$\sigma_{kl} = \sqrt{\frac{\sum_{i=1}^{n_k} (f_{ilk} - \mu)^2}{n_k - 1}}, \tag{7}$$

where n_k is the number of items belong to cluster k, f_{ilk} is the value of l^{th} of f in 6-axis flavour tastes of i^{th} item and μ is the mean value of all f_l, in group k. We calculate $G(x, y)$ six times for 6 fields $f_1 - f_6$ for each pair items over all items of a group to sort in descending to find top best results.

(iv) Levenshtein distance for comparison

The flavour tags also play a quite significant role in the final results. We treat tags as vital as each element in 6-axis flavour tastes such as $f_1 - f_6$. To compare and measure the similarity between two tags of string type, we use a good of levenshtein distance to solve it [15]. In our experiments, we use levenshtein distance $lev_{a,b}(i, j)$ as a string metric for measuring difference between two sequences (tags). The equation of the levenshtein distance is defined as follows:

$$lev_{a,b}(i, j) = \begin{cases} max(i, j), \text{ if } \min(i, j){=}0 \\ min = \begin{cases} lev_{a,b}(i-1, j) + 1 \\ lev_{a,b}(i, j-1) + 1, \text{ otherwise} \\ lev_{a,b}(i-1, j-1) + 1_{(a_i \neq b_j)} \end{cases} \end{cases}$$

where a and b are two tags of string that need to compute the levenshtein distance. $lev_{a,b}(i, j)$ is the distance between the first i characters of tag a and the first j characters of tag b.

(v) Final sorting formula

Combine weight calculating function for 6-axis flavour tastes and tags comparison with levenshtein distance (LD), we establish a equation for sorting to get final results as below:

$$S(i, j) = \sum_{k=1}^{K} \sum_{l=1}^{6} G_{kl}(i, j) + lev_{tags}(i, j), \tag{8}$$

where G_{kl} is the Gaussian weight function (6) corresponding l^{th} in 6-axis flavour tastes between $item_i$ and $item_j$ in cluster k ($k = \{1, ..., K\}$ K groups), $lev_{tags}(i, j)$ is the levenshtein function to compare tags similarity of those two items. We determine that the bigger $S(i, j)$, the better similar between those two items, so we sort by descending order all items of a cluster and return top m items having bigger $S(i, j)$ value.

(vi) Proposed model pseudo code

For clearly, we give the proposed model its algorithm execution process to help readers more easily visualize and imagine our entire process. Let see pseudo code below:

Algorithm 1 Framework model proposal.

```
Input: number of clusters $k$
Output: Top $m$ other similar items of each item
Data: Dataset $L$
1. Data pre-processing for text fields
2. Build a matrix for 6-dimension vectors representing
for 6-axis flavour tastes ($f_1 - f_6$)
3. Taking the matrix as an input of GMM to train and
save corresponding cluster of each item into dataset
4. for item in dataset do
    - Get cluster number of item
    - Find all items that have the
    same clusters
    - Applying equation $S(i, j)$ in (8)
    for each pairs items
    - Return top $m$ similar items by
    sorting decreasingly
end
```

4.3 Applicable to real-life applications

Actually, our algorithm has been using to solve a liquor recommendation system which is deploying in Japan. According to our experiment compared to results of three other popular, potent methods, our algorithm not only outweighs results of those models, but it also gain better, faster inference time that is over eligible for real-life applications. In particular, the time required for training after a period of time of new adding items is also very marginal.

In this paper, we cluster all items based on 6-axis flavour tastes, but the algorithm can apply for lower or greater number of six properties.

5 Experiments

To prove the validity of our proposed model, we compare our proposed model to three other popular algorithms widely used in CB systems such as BOW + GFF, GMM + ED, W2V + GFF. We also illustrate the impact of GMM cluster into the accuracy and the efficiency of GFF equation in sorting results rather than those of cosine or euclidean distance.

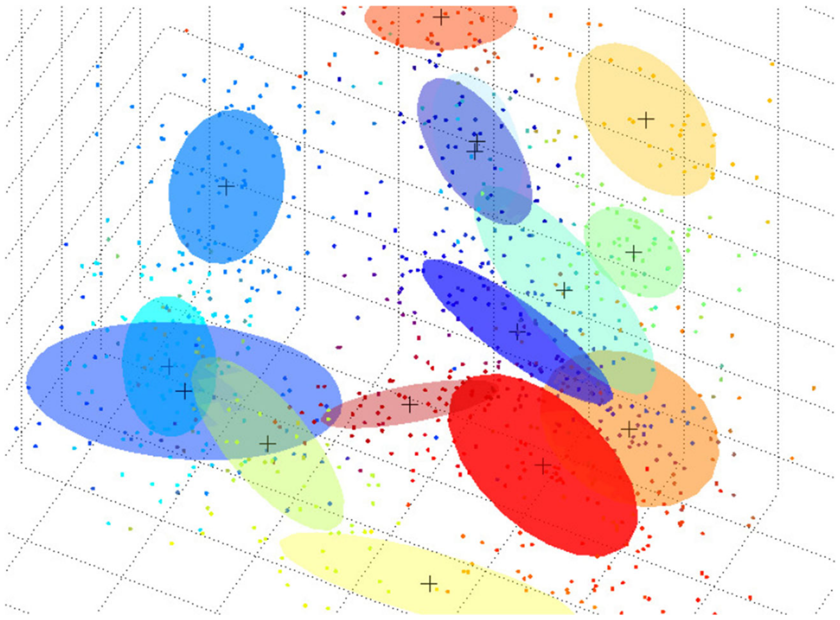
5.1 Experimental environment

All our experiments are implemented in python 3.6.8 environment. The experimental PC with Ubuntu version 16.04.7 is used to test the algorithms. The capacity of the liquor flavor taste (rawdata) crawled from Sakenowa website is 717Kb. Then, that of preprocessed liquor flavor taste data is 690Kb.

The evaluation method of recommendation systems commonly used is Root Mean Square Error (MSE) that is the average of the square errors [16]. The MSE value can be calculated as follows:

$$MSE = \frac{1}{N}\sum_{1}^{n}(r_i - \widehat{r_i})^2, \tag{9}$$

Fig. 7 GMM Visualization



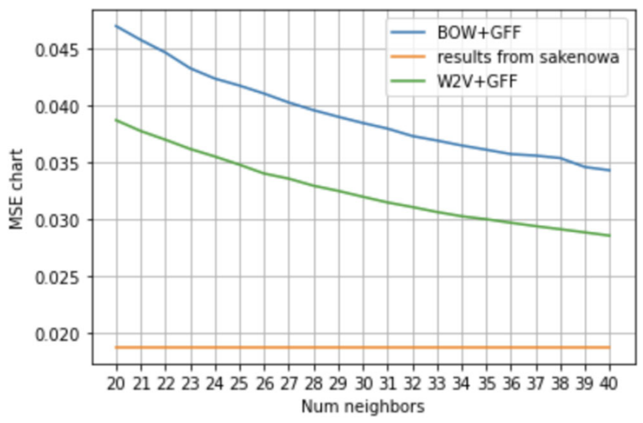


Fig. 8 MSE apply BOW+GFF and W2V+GFF

where r_i is the predicted representing vector item, and $\widehat{r_i}$ is the original representing vector item.

We also use the recommendation results in Sakenowa as the standard measure (baseline results) to compare with our three algorithms because the sake website has so much reputation, popularity, being well-known for commercial purpose in Japan for many years. Beside, the recommendation results of the Sakenowa is also very impressive.

5.2 Experimental analysis

Some experiments is conducted to verify the impact of GMM, GFF on probabilistic recommendation problem. Our main proposed model was implemented through such steps as data statistic, data cleaning, data missing value filling, clustering all items into different clusters and eventually using Gaussian filter and levenshtein distance to sort the results. To verify the effective impact of GMM and GFF on better prediction, we divided our experiments into four parts. Firstly, we use Bag-of-Word (BOW) [10] algorithm on some properties like flavour tags before applying GFF for sorting results. In the second way, we apply GMM + ED to clarify the influence of GMM, and W2V + GFF in the third experiment. Finally, we implemented our main proposed model to prove the impact of GMM + GFF then give some comparison. All experiments are unraveled in detail below:

(i) Experiment 1: BOW+GFF

The reason for this experiment is to verify the impact on result accuracy of GMM compared to BOW [10] algorithm. Therefore, in the experiment, we implement BOW algorithm comprised with GFF used for sorting on our liquor dataset. Firstly, we do some data preprocessing for text data like stemming, replacing synonyms, filling missing data, etc [2]. As it was mentioned above, all important text fields were written behind Japanese form, so we use some tools offered for Japanese preprocessing like Ginza [19], Janome [20], JapaneseStemmer [18] inspired by Porter Stemming Algorithm [17], etc. Before using GFF for sorting, we employ BOW on these preprocessed properties to find the vector matrix representing for the item. The next step, we feed the vector matrix into K-nearest neighbors (K-NN) algorithm using unsupervised K-NN Scikit-Learn [21] to find top similar items based on these vectors. In these top items, we calculate values S using the (8) to get the best similar items.

(ii) Experiment 2: GMM+ED

To demonstrate the impact of GMM, firstly, we still apply some preprocessing steps for text fields as the experiment above. After that, we build a matrix of 6 dimensions representing for 6-axis flavour tastes, then feed it into GMM for training, saving all cluster results corresponding to each item. Next step, we convert a collection of text flavour tags into a matrix of token counts using CountVectorizer of Scikit-Learn [21] and concatenate along same axis with the matrix of 6 dimensions for sorting. Finally, to return the best similar items of a given item, we just jump up to the cluster containing it and apply ED for sorting the results and get top best similar items of the given item.

(iii) Experiment 3: W2V+GFF

To reduce the linguistic ambiguity, and solve the limitation in semantic meaning of words of BOW algorithm and compare with our algorithm. In this part, we experiment W2V [23] combining with TF-IDF for flavour tags, then congregate with the matrix of 6-axis flavour tastes to get item representation before sorting results by GFF. As two previous experiments, we also conduct data preprocessing techniques as same steps as we did. Next step, as flavour tags are written behind Japanese form, so we use W2V pretrained model with 300 dimensions from [25] and

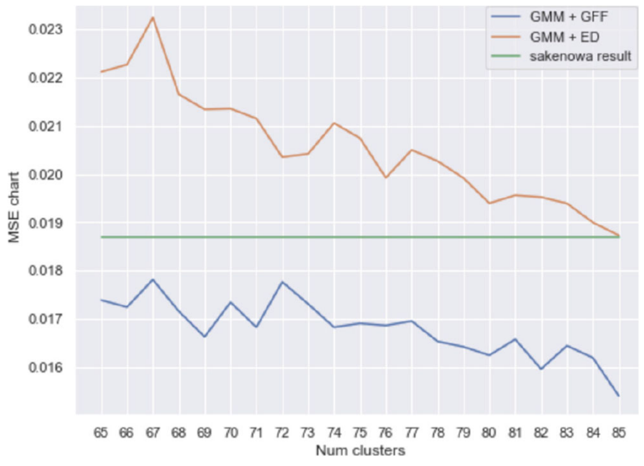


Fig. 9 MSE apply GMM+GFF and GMM+ED

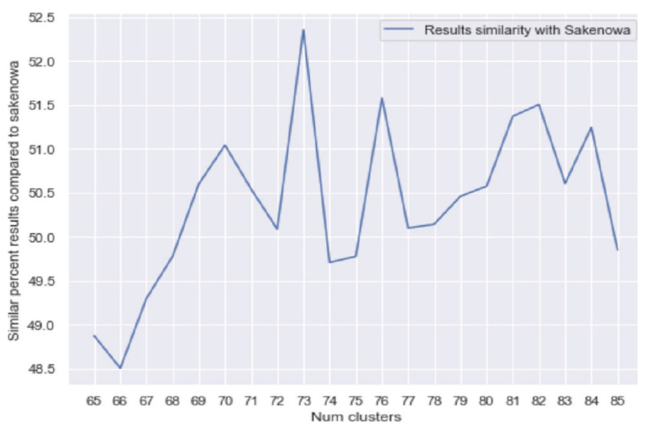


Fig. 10 Similar percent statistic compare to Sakenowa

Gensim [24] for loading word embeddings model for each Japanese tag. To represent each item by its flavour tag, we combine tag embeddings with TF-IDF by the below formula (10):

$$V_{ft} = \frac{\sum_{ti=1}^{q} TFIDF_{ti} * W2V_{ti}}{\sum_{ti=1}^{q} TFIDF_{ti}}, \tag{10}$$

where ti is corresponding to tag_i in flavour tags of a liquor item, q is the number of tags of the flavour tags which is equivalent to the length of the flavour tags of a liquor item.

After that, to get all item representation, we concatenate the word embeddings matrix along axis with 6-axis flavour tastes to get the final matrix of 306 dimensions. By the concept of Cosine Similarity, we leverage it for finding top prospective items before calculating the value S from the (8) in these items to get the best similar items.

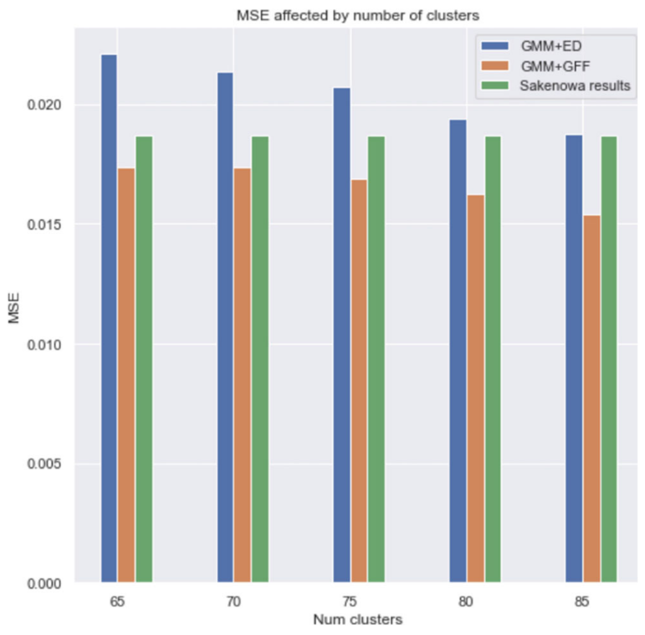


Fig. 11 MSE affected by number of clusters

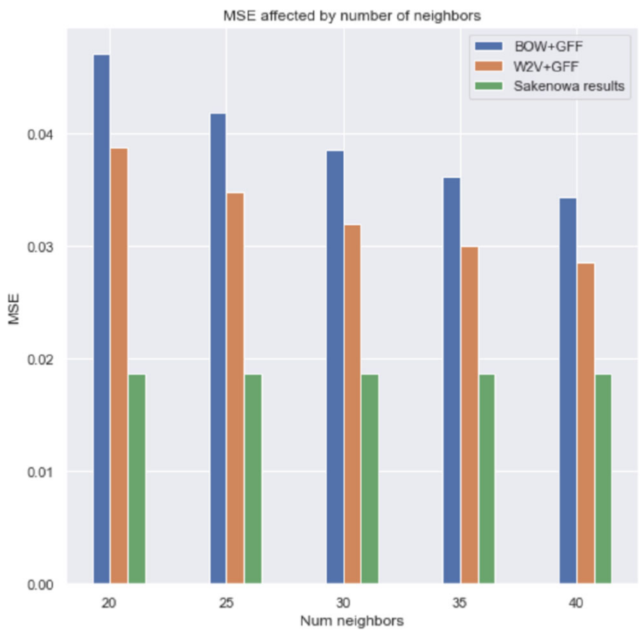


Fig. 12 MSE affected by number of neighbors

(iv) Experiment 4: GMM+GFF

Our three above experiment to prove the important role of GMM and GFF in our proposed model. In this experiment, in the first place, we also do preprocessing techniques for text fields as same steps in three previous experiments. After that, we build a matrix of 6 dimensions representing for 6-axis flavour tastes and feed the matrix into GMM for training purpose, then save cluster results for each items. To find top best similar items of a given item, we jump up into the cluster the query item lied in, consider the query item as center then apply (8) equation pair in pair with all items in the cluster, then sorting discerningly to return the top best similar items.

5.3 Experimental results and comparison

In this section, we compare our proposed algorithm with the recommendation results from the Sakenowa website and three other popular CB algorithms. The recommendation results from Sakenowa for each item are returned from an web API (Application Programming Interface) [3]; therein, $f_{1...6}$ in the API are the value for each flavour taste, respectively.

According to the experimental results, we conclude that our result accuracy outweighs the Sakenowa and these three algorithm counter parts. Our comparison results are shown in Figs. 8, 9, and 10.

[3] https://sakenowa.com/api/v1/brands/flavor?f=0&fv=f_1,f_2,f_3,f_4,f_5,f_6

Table 2 Time response per query

BOW+GFF	GMM+ED	W2V+GFF	GMM+GFF
0.1856s	0.0174s	0.0251s	0.0156s

All four experiments return top ten(10) best similar items for each item in the dataset. In Fig. 8, list values of MSE are shown and affected through an array of number of prospective neighbors of a item ranging from [20-40] in K-NN algorithm and cosine similarity matrix. Despite the tendency of decrease, but it is insignificant and the inference time is extremely slow due to bigger number of neighbors, particularly in the K-NN algorithm. It is very clearly to see that in this figure, caring more about the semantic meaning of words from W2V totally beat the performance of the BOW algorithm; despite, using the same GFF similarity calculation behind.

In Fig. 9, the gap of MSE between GMM + ED and GMM + GFF is shown. It is very clearly seen that GMM + GFF generated better results than the other that verify the effect of GMM in comparison to ED in sorting results. Both these two experiments showed the effect of the number of clusters of GMM ranging from [65-85]. In Fig. 10, we compare our prediction results of all items in dataset to the recommendation results from the Sakenowa and construct a list of similarity proportion affected by the number of clusters.

In Figs. 11 and 12, we constructed a chart of statistic of MSE generated from GMM + ED, BOW + GFF, W2V + GFF, GMM + GFF and recommendation results from Sakenowa. It is matter of fact that our GMM + GFF algorithm outperforms all the others method that demonstrate the effective of our algorithm.

Further more, our time response in Table 2 also beat these three others, GMM + ED, W2V + GFF and BOW + GFF.

Referring to Figs. 9, 8, 11 and 12, we have a conclusion that taking 6-axis flavour tastes superior priority rather than flavour tags give better results in our recommendation system, and the very effective, robust and productive GMM in solving the distribution recommendation.

There is much evidence that recommendation-based GMM would works well on datasets in which the values of those main properties obey continuous datatype or the like that is the main reason other algorithms like BOW, W2V taking text fields higher priority give lower quality recommendation results compared to our algorithm.

6 Conclusion and future work

In this paper, we have proposed an very effective algorithm for recommendation system using content-based features with GMM, and apply for solving a liquor recommendation system which is deploying in Japan. Furthermore, we have also proposed a new sorting similarity method for list of potential items instead of using popular methods like Cosine or Euclidean.

More specially, our probabilistic-based recommendation systems not only acquire a remarkable prediction accuracy, but it has also very speedy prediction time response for real-time application. Our algorithm are flexible that can use for lower or greater 6 properties in other datasets which show similar distribution with our liquor dataset. The reason for this because we have tried several experiments with 3, 4, 5, 6 of liquor flavor taste combining with flavor tags, and all of them outperform the other approaches in Experiment section. We only choose 6 flavor taste (f_1 - f_6) instead of 3, 4, 5 as representative for our paper due to f_1-f_6 play important role equally. Although, the algorithm has many advantages, its limitation is need to re-train after a period of time of adding new items into the system. In additional, our dataset currently is collected from Sakenowa (a very popular website selling sake in Japan) possessing characteristics and type of distribution quite distinctively which does not show the resemblance of context to the current benchmark datasets. Therefore, our proposed algorithm would not be suitable with these available datasets.

In the future, our tendency research is finding a solution of GMM improvement for clustering items to get even better recommendation results.

Acknowledgements We would like to thanks to Sun* Inc in general, and particularly, R&D Department that helping us a lot. Without the support this paper cannot be accomplished.

References

1. Lops PdeG, Giovanni MS (2011) Content-based recommender systems: State of the art and Trends. https://doi.org/10.1007/978-0-387-85820-3-3
2. Rahutomo RL, Muljo F, Bens HP (2019) Preprocessing methods and tools in modelling japanese for text classification. https://doi.org/10.1109/ICIMTech.2019.8843796
3. Yan H, Yan T (2019) collaborative filtering based on gaussian mixture model and improved jaccard similarity. IEEE Access. PP.1–1 https://doi.org/10.1109/ACCESS.2019.2936630
4. Fan-sheng K (2010) Hybrid Gaussian pLSA model and item based collaborative filtering recommendation. Computer Engineering and Applications
5. Chen RH, Gao Q, Ying QX (2018) A hybrid recommender system for gaussian mixture model and enhanced social matrix factorization technology based on multiple interests. Math Problems Eng 2018:1–22. https://doi.org/10.1155/2018/9109647
6. Yoshii KG, Komatani M, Ogata K, Hiroshi OT (2006) Hybrid collaborative and content-based music recommendation using probabilistic model with latent user preferences. In: ISMIR 2006 - 7th international conference on music information retrieval, pp 296–301

7. Khusro SA, Irfan ZU (2016) Recommender systems: Issues, challenges, and research opportunities. https://doi.org/10.1007/978-981-10-0557-2-112
8. Zhu BB, Fernando JO (2018) Reliability quality measures for recommender systems. Information Sciences
9. Douglas R (2008) Gaussian mixture models. Encyclopedia of Biometrics. https://doi.org/10.1007/978-0-387-73003-5-196
10. Bhattacharya S, Ankit L (2019) Movie recommendation system using bag of words and scikit-learn. Int J Eng Appl Sci Technol 04:526–528. https://doi.org/10.33564/IJEAST.2019.v04i05.076
11. Liberti LL, Maculan C, Antonio MN (2012) Euclidean distance geometry and applications. SIAM Rev 56 https://doi.org/10.1137/120875909
12. Lee D, Hull J, Erol B (2003) A Bayesian framework for Gaussian mixture background modeling. In: Proceedings 2003 international conference on image processing (Cat. No.03CH37429), vol 3, pp III–973
13. Lu Y, Bai X, Wang F (2015) Music recommendation system design based on gaussian mixture model. ICM 2015
14. Görür D, Carl R (2010) Dirichlet process gaussian mixture models: Choice of the base distribution. J. Comput. Sci. Technol. 25:653–664. https://doi.org/10.1007/s11390-010-9355-8
15. Haldar R, Debajyoti M (2011) Levenshtein distance technique in dictionary lookup methods: An improved approach. Computing Research Repository - CORR
16. Shani G, Asela G (2011) Evaluating recommendation systems. https://doi.org/10.1007/978-0-387-85820-3-8
17. Robertson S (1997) Readings in information retrieval
18. MrBrickPanda (2019) Japanese Stemmer. Github. https://github.com/MrBrickPanda/Japanese-stemmer
19. Hiroshi M, Masayuki (2019) Ginza NLP Library. Github. http://www.anlp.jp/proceedings/annual-meeting/2019/pdf-dir/F2-3.pdf
20. Janomepy (2019) Janome. Github. https://github.com/mocobeta/janome
21. Pedregosa F, Alexandre V, Michel (2011) scikit-learn: Machine learning in Python. J Machine Learn Res 12:2825–2830
22. Jeran FA, Ninaus M, Reinfrank G, Reiterer F, Martin SS (2014) Basic approaches in recommendation systems. https://doi.org/10.1007/978-3-642-45135-5–2
23. Musto CS, de Gemmis G, Pasquale ML (2016) Learning Word Embeddings from Wikipedia for Content-Based Recommender Systems. 9626:729–734 https://doi.org/10.1007/978-3-319-30671-1-60
24. Řehůřek R, Sojka P (2010) Software framework for topic modelling with large corpora. 45–50. https://doi.org/10.13140/2.1.2393.1847
25. Yamada IS, Takeda H, Yoshiyasu HT (2016) Joint learning of the embedding of words and entities for named entity disambiguation. 250–259. https://doi.org/10.18653/v1/K16-1025
26. Quispe P, Ocsa OE, Ricardo AC (2017) Latent semantic indexing and convolutional neural network for multi-label and multi-class text classification. 1–6. https://doi.org/10.1109/LA-CCI.2017.8285711
27. Rosa RS, Ruggiero G, Rodriguez WV, Zegarra D (2018) A Knowledge-Based recommendation system that includes sentiment analysis and deep learning. IEEE Trans Indust Inform PP:1–1. https://doi.org/10.1109/TII.2018.2867174
28. Kadhim A (2018) An evaluation of preprocessing techniques for text classification. Int J Comput Sci Inform Secur 16:22–32
29. Mansur FP, Mihir VP (2017) A review on recommender systems. 1–6. https://doi.org/10.1109/ICIIECS.2017.8276182
30. Czarnowska P, Emerson GE, Copestake AA, 2019 Words are Vectors, Dependencies are Matrices: Learning Word Embeddings from Dependency Graphs. IWCS
31. Cheng J, Li Z (2019) Jaccard coefficient-based bi-clustering and fusion recommender system for solving data sparsity https://doi.org/10.1007/978-3-030-16145-3-29
32. Rutkowski TR, Woldan J, Staszewski P, Nielek P, Leszek RR (2018) A Content-Based Recommendation System Using Neuro-Fuzzy Approach. 1–8. https://doi.org/10.1109/FUZZ-IEEE.2018.8491543
33. Roy PC, Rocky BS (2020) A Machine Learning approach for automation of Resume Recommendation system. Procedia Comput Sci 167:2318–2327. https://doi.org/10.1016/j.procs.2020.03.284
34. Chatterjee N, Nidhika Y (2019) Hybrid latent semantic analysis and random indexing model for text summarization: proceedings of third international conference on ICTCS 2017 https://doi.org/10.1007/978-981-13-0586-3-15
35. Li Y, Wang S, Pan Q, Peng H, Yang T, Cambria E (2019) Learning binary codes with neural collaborative filtering for efficient recommendation systems. Knowl Based Syst 172:64–75
36. Feng WZ, Zhuang Q, Yu JS (2019) An expert recommendation algorithm based on Pearson correlation coefficient and FP-growth. Cluster Comput 22 https://doi.org/10.1007/s10586-017-1576-y

Nguyen Van Dat is currently pursuing the master's degree of Computer Science at the University of Engineering and Technology. He has also graduated at the Military Technical Academy as the Bachelor's degree of Software Engineering in 2017. His major interested research areas including Computer Vision, Recommendation Systems, and Cognitive Science.

Ta Minh Thanh is Lecturer of Faculty of Information Technology, Le Qui Don Technical University, Ha Noi, Viet Nam. He is also Postdoctoral Fellow of Department of Mathematical and Computing Sciences at Tokyo Institute of Technology. He received his B.S. and M.S of Computer Science from National Defense Academy, Japan, in 2005 and 2008, and his Ph.D. from Tokyo Institute of Technol- ogy, Japan, in 2015, respectively. He is the member of IPSJ Japan and IEEE. His research interests lie in the area of watermarking, network security, and computer vision.