Hindawi
Scientific Programming
Volume 2019, Article ID 1293194, 17 pages
https://doi.org/10.1155/2019/1293194

Research Article

A Nondisturbing Service to Automatically Customize Notification Sending Using Implicit-Feedback

Fernando López Hernández, Elena Verdú Pérez, J. Javier Rainer Granados, and Rubén González Crespo

Universidad International de la Rioja (UNIR), Av. de la Paz 136, 26006 Logroño, Spain

Correspondence should be addressed to Rubén González Crespo; ruben.gonzalez@unir.net

Received 19 December 2018; Accepted 4 February 2019; Published 3 March 2019

Guest Editor: Vijender K. Solanki

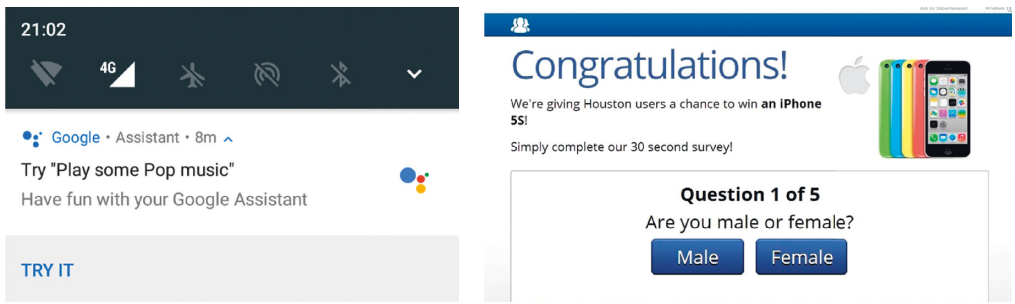
This paper addresses the problem of automatically customizing the sending of notifications in a nondisturbing way, that is, by using only implicit-feedback. Then, we build a hybrid filter that combines text mining content filtering and collaborative filtering to predict the notifications that are most interesting for each user. The content-based filter clusters notifications to find content with topics for which the user has shown interest. The collaborative filter increases diversity by discovering new topics of interest for the user, because these are of interest to other users with similar concerns. The paper reports the result of measuring the performance of this recommender and includes a validation of the topics-based approach used for content selection. Finally, we demonstrate how the recommender uses implicit-feedback to personalize the content to be delivered to each user.

1. Introduction

Companies want to keep their customers informed about the availability of new services and products. However, the continuous sending of notifications to the user's devices can produce the opposite effect [1] if these notifications end up bothering users, who in turn ignore, remove, or block them (Figure 1). To mitigate these adverse effects, it is sensible to be selective and only send the notifications that we are aware that really interest each user. Machine-learning techniques enable the analysis, summarization, and classification of text in a massive and automatic way. Therefore, this paper surveys the use of these tools to identify which clients are interested in which notifications.

The clues used to identify whether a notification subject matters to a user may be explicit or implicit [2]. The *explicit-feedback* constructs the user profile by asking the users for their personal characteristics and preferences for different topics or items. The *implicit-feedback* constructs the user profile by silently observing the behavior of the users (e.g., the time spent on a page, the amount of scrolling, or the number of mouse-clicks) and then inferring their rating [3]. Collecting explicit-feedback conflicts with modern marketing trends and policies. In particular, companies want to maximize the user experience by minimizing the cognitive effort and information-filling burden during any web interaction. An interaction that should be especially agile is registration, because this maximizes the number of new users who complete their registration. However, this policy results in collecting poor explicit-feedback.

This paper studies to which extent a recommender is able to extract clues that significantly improve the sending of notifications when only implicit-feedback is available (i.e., user actions which result in nondisturbing collection of information). For this purpose, on one hand, we have developed mechanisms that translate the implicit-feedback (i.e., user interactions) into topics of interest of the user. On the other hand, we have applied machine-learning techniques (i.e., text mining, summarization, and classification) to extract the topics of each notification. We hypothesize that these pieces of implicit-feedback, along with the automatically classified notifications, enable us to select the interesting notifications for each user. To analyze this hypothesis, we have conducted an experimental evaluation.



(a) (b)

Figure 1: Examples of disturbing notifications. (a) Android notification. (b) Web page campaign.

First, we have collected indicators that act as pieces of implicit-feedback. Second, we have measured the performance of the *recommender* (i.e., the automatic recommendation system) implemented in our service and related our findings with those of other state-of-the-art works.

This paper makes the following main contributions to the field:

(1) Determines to what extent implicit-feedback suffices to make automatic recommendation of notifications.

(2) Develops a mechanism to determine the user preference for the notifications by iteratively reducing dimensionality: terms, characteristic words, and topics.

(3) Demonstrates how text mining, summarization, and classification techniques are effectively combined to create a hybrid filter: a collaborative + a content-based filter that

  (a) Mitigates the cold-start problem (a well-known problem of collaborative filters) because once the user has positively rated one notification, we use the content-based filter to start recommending notifications of same topic.
  (b) Effectively integrates content similarity (the content-based filter finds notifications with the same topic) and diversity (the collaborative filter finds users with the same tastes).
  (c) Does not require explicit domain knowledge of the items (as is the case of knowledge-based techniques) and so does not require metadata-annotated items.
  (d) Completely automates customized notification sending, without any user intervention.

(4) Publishes the source code and the anonymized dataset to enable future research and repetitiveness of the reported experiments (https://github.com/ifernandolopez/hybrid-filter).

The rest of this paper is organized as follows: Section 2 reviews comparable state-of-the-art recommenders and background machine-learning techniques. Section 3 describes the proposed solution. Section 4 provides the design of the experiments. Section 5 provides and discusses the results of our approach and relates it to other approaches. Finally, Section 6 presents the conclusions and possible future work.

2. Literature Review

This section reviews related state-of-the-art recommenders in several application domains, as well as the basis of background machine-learning techniques used to conduct this research.

2.1. State-of-the-Art. Recommenders are popular and widely used. Well-known e-companies use them to improve the user experience [2, 4]. There is a wide variety of items recommend, including movies [5], books [2], music [6], research articles [7], clothes [8], travels [9], events [10], and many more.

Although these systems filter the relevant content and improve the user's satisfaction, some challenges remain; for instance, nondisturbing the user with the feedback collecting mechanism [11], monotonous recommendations [12], or the cold-start problem [2, 13]. Regarding the first challenge, currently many recommending solutions still rely on the explicit-feedback provided by the user, typically the user ratings [14–18]. However, not every user is willing to provide ratings, especially when ratings are optional. Moreover, ratings may be biased by user's contextual and emotional states.

Implicit-feedback is also available. The combination of both implicit-feedback and explicit-feedback may mitigate the problem of lack of explicit ratings. Therefore, different systems in the literature use both types of feedbacks. For example, the sports news recommender described in [19] bases its recommendations on the user's readings (i.e., implicit-feedback) and ratings (i.e., explicit-feedback). Bagherifard et al. [20] also use both implicit and explicit-feedback in a hybrid approach for movie recommendations, though their solution uses ontologies. A hybrid and ontology-based approach is surveyed in [17] using ratings for news recommendations. Also, Agarwal and Singhal [21] propose a solution based on a domain ontology, which uses explicit and implicit data of users; the registered user provides the explicit information, while the implicit information includes mouse behavior and user session data. The system proposed in [22] retrieves keywords from external user and

item sources to generate implicit-feedback to diminish the cold-start problem.

However, although using implicit-feedback and explicit-feedback can enhance recommendations, there are situations in which users are not willing to provide explicit-feedback. Fortunately, there are abundant implicit data that may serve to model the user preferences. In fact, there are authors that target the problem where only implicit-feedback is provided [23, 24]. Núñez-Valdez et al. [24] propose a system that converts implicit behavioral data into explicit-feedback to recommend books. Typical user actions are considered, such as highlighting content, adding notes, or suggesting content to other contacts. The mobile application advertising recommender in [25] also follows the approach of mapping the implicit-feedback (click, view, download, and installation information) into explicit ratings. Besides, they use slowly changing features of the mobile context for recommendations. The news recommender described in [26] uses the clicks on news items to create the user's profile. Also, Li and Li [27] use only user's reading actions for news recommendations, creating a hypergraph to model correlations among items and implicit relations among users. The system proposed by Zheng et al. [28] also keeps track of users' reading actions. Soft clustering is used to group users to enrich the user profile with general reading interest and relates users with similar behavior. For each user group, the system identifies the latent topics, and creates hierarchies. Lu et al. [29] consider implicit actions like browsing, commenting, and publishing. They target scalability and data sparsity by using Jaccard K-means based clustering technique. The users' similarities are calculated considering multiple dimensions, such as the user interactions with content and their eigenvectors of topics. Extracting information from the users' tweets is another strategy to build the user's models for recommendations [30]. Gu et al. [31] propose an approach that uses content in microblog or tweets of users and users' social network preferences (popularity) for news recommendations. Also, content in microblogs is taken into account for recommendations by Zheng and Wang [32], but from a sentimental point of view. Retweeting of news are the implicit activities considered in a content-based recommendation system that models user trends [33].

The second abovementioned challenge relates to monotony in recommendations, which is inherent in pure content-based approaches. Hybrid approaches target the lack of diversity in recommendations by combining content-based and collaborative filtering. For instance, Lenhart and Herzog [19] introduce a collaborative filter to target diversity in their sport news recommender. The keywords are automatically obtained from the pieces of news, as well as from the users' reading data, and are used to estimate the interest of users in topics and provide the content-based recommendation. All of this is complemented with a collaborative-based recommendation that uses users' ratings. Li et al. [26] use hierarchical clustering for grouping news articles and long- and short-term user profiles based on clicks on news, and they incorporate the absorbing random walk model to achieve a diversity of topics in recommendations. Lastly, some systems provide and merge multiple recommendation lists based on different criteria to enhance diversification, which is called *result diversification* [11].

While content-based approaches have the drawback of monotony, they effectively address the cold-start problem [13]. On one hand, apart from facilitating diversity, a clear advantage of collaborative-based approaches is that they do not need domain specific data. However, collaborative filtering alone suffers from the sparsity problem. On the other hand, content-based filtering is not very effective in recommending news items, because usually users only read a small part of these [17]. Therefore, our approach proposes combining collaborative-based filtering, which provides diversity, with content-based filtering, which mitigates the cold-start problem, and addresses the sparsity problem through aggregation and clustering techniques. In addition, our content-based filter does not have domain-dependency, as is the case with other approaches (e.g. [20]).

2.1.1. Datasets for Evaluation. This section reviews potential datasets to evaluate the performance of recommendation with notifications. Although we have not found a dataset of users rating notifications, we have found some datasets of users rating text documents. For instance, the authors of [34] studied topic diversification and have published their dataset with 278,858 users providing ratings about 271,379 books. The authors of [35] have studied dimensionality reduction for offline clustering, and they have published their dataset with 4.1 million ratings of 100 jokes from 73,421 users. Yahoo has published two datasets about news visits: "R6A-Yahoo! Front Page Today Module User Click Log Dataset, version 1.0" and "R6B-Yahoo! Front Page Today Module User Click Log Dataset, version 2.0" (https://webscope.sandbox.yahoo.com/). The first one includes 45,811,883 unique user visits to news articles displayed in the Featured Tab of the Today Module on Yahoo! Front Page during ten consecutive days. The second includes 15 consecutive days of data gathering with 28,041,015 user visits to the same module. Even though they are well populated with users' interactions, they lack other types of interactions (e.g., sharing, printing, skipping, and deletion). In addition, the short periods cause the events to repeat themselves, thus producing certain biases [13]. "Outbrain Click Prediction" (https://www.kaggle.com/c/outbrain-click-prediction/data) dataset also corresponds to a 2-week period, and its main limitation is that it does not provide the content, but only some semantic attributes of the documents.

In summary, after making this survey, we found that (1) most of the studies are based in unmentioned or private datasets [19, 21, 26–28, 36], and, what is worse, (2) we have not been able to find a public dataset of notifications, which motivates our decision to collect and publish our own dataset of user interactions with notifications (described in Section 4.1).

2.2. Background Review. This section reviews the background machine-learning techniques used to conduct this research.

2.2.1. Preprocessing Text. The curse of dimensionality is a well-known effect in text mining [37] that occurs when there are so many dimensions (different words in our case) and that the distance between two documents is always too far. Text preprocessing is a suite of methods that facilitate text analysis by eliminating uninformative text and reducing the dimensionality of the features [38]. Among these techniques, we find

(1) *Tokenization.* Parsing the text to generate terms. Although they are really different concepts [39], traditionally, in text mining *term* and *word* have been used interchangeably, and we also do this in this paper.

(2) *Removing stop words.* A stop word is a commonly used word of a language (e.g., "the", "for", "a"). They are the glue or link for more semantic words, and so stop words give little information about the content of the text. There exist lists of stop words for most languages.

(3) *Removing multiple spaces, numbers, and punctuation symbols.* Symbols provide little information when we analyze tokenized words, but punctuation symbols are useful if, for instance, we analyze phrases.

(4) *Removing repeated text.* Frequently, a document has repeated texts such as headers, footers, or copyright information that are removed to avoid outweighing the importance of these words.

(5) *Removing sparse terms.* The words that appear just once or a few times increase the dimensionality, and their scarceness indicates a low relevance in the document, so they can be removed. In particular, removing words appearing only once in the whole document is an effective approach to remove misspelled words.

(6) *Removing equivalences.* To further reduce the dimension of the terms, terms with the same meaning can be grouped, with techniques such as (a) *lowercasing*; (b) accent removal; (c) *synonyms grouping*, in which a thesaurus is used to homogenize synonymous words; (d) *stemming*, which groups similar words by removing the plurals, and after that it reduces them to their root [40]; or (e) *lemmatization*, which identifies lexically or semantically similar words [41].

2.2.2. Relationships between Words. Text is unstructured data, so we have to capture its structure. The most common ways to represent the relationships between words are the following [42]:

(1) *Bag of word.* This is a simplified representation of a document, in which only the number of occurrences of each word is taken into account, but disregards the order or the words.

(2) *Word vector.* This representation extends the idea of bag of words by assigning a rating to each word in the document. A popular rating is the number of times each word occurs (the approach followed by the bag of word). Below, we will describe a more effective rating technique named TF-IDF.

(3) *Term Document Matrix* (TDM). This representation describes the frequency of each term by using a matrix (Figure 2). For this purpose, we define a *vocabulary* as all the words appearing in any of the document. In this matrix, each row corresponds to a vocabulary and each column to a document, that is, each column is a word vector. Note that this representation requires a corpus, that is, a collection of documents, one for each column. Some authors transpose the TDM so that the rows correspond to the vocabulary and columns correspond to the words appearing in each document. This representation is referred as *Document Term Matrix* (DTM).

(4) *n-grams.* This is a contiguous sequence of n terms for considering the relationship between consecutive words. The clustering of 2 or more words is named a *collocation.* For instance, [43] has used n-grams collocations and Markov Chains to determine the probability of a word being followed by another word and to identify common grammar mistakes.

2.2.3. Characteristic Words of a Document. A problem that has been widely studied is how to find the characteristic words of a document (e.g., [44–47]). These characteristic words can be used, for instance, to implement keyword-based document search.

A first approximation is the *Term Frequency* (*TF*) [48]. This score counts the relative frequency of each term t in the analyzed document d and selects as characteristic terms of the document those with a higher frequency. That is, this score merely selects those terms that maximize the following $TF(t, d)$ function, where $|t \in d|$ is the number of times the term appears in document d, and $|d|$ is the number of terms in the document d:

$$TF(t, d) = \frac{|t \in d|}{|d|}. \quad (1)$$

Note that the ratio in (1) compensates for the differences in length of the analyzed documents, so that the *TF* score only depends on the relative frequency of the term and not on the size of the document.

The major problem with the *TF* is that words with higher frequency tend to be the same in all documents, even if we remove those words in a list of stop words. This effect is an empirical law known as Zipf's law. The conclusion is that the frequency of words is not the best way to find the characteristic terms of a document.

An effective and popular approach to finding the characteristic words of a document within a corpus is the *TF-IDF* score. The *TF-IDF* score has been frequently used along with a clustering algorithm to classify text [49] or decide on the topics of a corpus [50]. This score combines the *TF* with the Inverse Document Frequency (*IDF*) to choose the characteristic words of the document. The *IDF* gives a higher score to rare terms using the following

Term document matrix (TDM)

	d1	d2	d3	d4	d5
Boat	1	0	1	0	0
Sand	0	0	1	1	0
See	1	2	1	1	2
Ship	2	1	0	0	0
Sun	0	0	3	1	0

Word vector

Figure 2: Relationship between word vector, TDM.

formula, by dividing the number of documents in the corpus $|D|$, by the number of documents in the corpus where the term t appears $|t \in D|$. This ratio is never less than 1, so the logarithm is always positive:

$$IDF(t, D) = \ln\left(\frac{|D|}{|t \in D|}\right). \quad (2)$$

Note that in (2), D is uppercased to indicate that the score is calculated with respect to the corpus of documents and not the currently evaluated document d. The *TF-IDF* score takes advantage of the fact that Zipf's law does not only apply to a document, but also to a corpus of documents. Then, we can calculate the frequency of the words in a corpus D and compare it with the frequency of the words in a certain document d.

The *TF-IDF* score is the product of both indices (3), that is, it is the product of the times that the term appears in our document $TF(t, d)$ and the infrequency of the term in the general corpus $IDF(t, D)$:

$$\text{TF} - \text{IDF}(t, d, D) = TF(t, d) \cdot IDF(t, D). \quad (3)$$

The main reasons for the popularity of this score are the following:

(1) It captures how specific the term t is for a given document d only; therefore, it effectively selects a small set of rare words as characteristic words of the document.

(2) It is not necessary to remove stop words. This is because a very frequent term t receives an $IDF(t) = 0$ and so $TF\text{-}IDF = TF(t) \cdot 0 = 0$.

(3) The accuracy of the selection can be improved with stemming, as stemming groups words with the same root.

2.2.4. Topic Modeling. Topic modeling is the area of machine learning that applies unsupervised clustering techniques for discovering the topic of a collection of documents. Latent Dirichlet Allocation (LDA) [51, 52] is a popular topic modeling method to summarize the meaning of the words and documents (the observed variables) in hidden variables (latent low-dimensional clusters). Its popularity is based on its fuzzy clustering approach, in contrast with other hard clustering methods such as Explicit Semantic Analysis [53].

LDA assumes that each document is a mixture of a small number of topics, and each topic is a mixture of a number of words. Therefore, this one-to-many mapping implies an overlap in both the topics of a document and the terms of a topic.

In particular, the LDA algorithm performs Bayesian inference on the observable variables (words and documents) to update the posterior probabilities of the initial belief on the hidden latent variables (topics). The algorithm produces a set of topics, the topic proportion for each topic, and two posterior probabilities:

(1) *Beta probabilities* are the estimates of the probability of a word belonging to each topic. The more often a word occurs in a topic, the higher the value of beta. The words with higher beta probabilities in each topic can be used to summarize the topics.

(2) *Gamma probabilities.* While learning topics, LDA also learns topic proportions per document. The gamma probabilities are the estimates of the proportion of words of a document that are generated by each topic. The more the words of a document are assigned to a topic, the higher the gamma value is.

2.2.5. Recommendation. A *recommender* is an automatic information-retrieval filter that builds a model from the user's profile and behavior to predict the rating or preference that a user would have for an item. One type of recommender is a *content-based filter* [54], which uses the description of the item and the user profile. That is, the content-based filter basically decides the best-matching between these two sets:

(1) *Content*, annotated with keywords that describe the content itself.

(2) *User profile*, in which keywords describe the user's preferences.

For instance, if the content is documents, the keywords that describe this content may be inferred from the *TF-IDF* score of their words. The keywords in a user's profile will correspond to the query to the search engine. Sometimes *tagging* is used to make the content and the user profile comparable (e.g., [55]). As the user visits content, the user's profile is annotated with tags form the content.

Another popular type of recommender is a *collaborative filter*. A collaborative filter [56] collects the preferences that a large group of people have assigned to a group of items and predicts the rating of a user for an item that has not been rated yet. Usually, the collaborative filter operates in two phases:

(1) *Training phase.* During this phase, we collect the user's preferences for particular items and create an Item User Matrix (IUM), like the one shown in Table 1. The rows of the IUM correspond to the items $\mathbf{i} = \{i_1, i_2, \ldots, i_k\}$, the columns correspond to the individual users $\mathbf{u} = \{u_1, u_2, \ldots, u_p\}$, and the entries correspond to the relevance that each user has assigned to each item. Note that usually the IUM is a

Table 1: Example of IUM with user ratings of some items created during the training state.

	u_1	u_2	...	...	u_p
i_1	**		*****		**
i_2		*	***		
⋮	*****	***		**	****
⋮		*	***		
i_k	****		***		**

sparse matrix, that is, most items are not voted by users. The basis of any collaborative filter is to find people with similar tastes. If user u_i and user u_j have assigned similar ratings to a set of items, we assume that they have similar tastes. If we now detect that u_i has assigned a rating to an item that u_j has not rated, we assume that the rating that u_i assigns to this item will be similar to that of u_j.

(2) *Exploitation phase.* During this phase, we receive a new item that a user has not rated, and the collaborative filter predicts the rating that this user will assign to that item.

An issue that collaborative recommenders have is the so-called *cold-start* problem. In particular, the recommender will not properly predict the rating of users when it has not yet collected enough information. This issue occurs in two cases: (a) when the user is new, so we do not have enough information about the topics of interest of the user, and (b) when the item is new, and we do not have enough ratings for this item.

A *hybrid recommender* is a combination of content-based and collaborative filtering. The hybrid recommender takes advantage of both the representation of the content as well as the similarities among users. One advantage of combining information is that this process can produce a more informed prediction. Another advantage is that it can reduce the cold-start problem by overweighing the content analysis when an item has not received enough ratings, and vice versa.

2.2.6. Evaluating Recommendations. Evaluating recommendation basically consists in measuring the ability of a recommender to generalize current user's rates for some items to other unrated items.

Regarding the recommendation tasks to evaluate, Shani and Gunawardana [57] describe different approaches for the evaluation: top-N recommendation, some good items, all good items, rating prediction, utility optimization, etc. In Sections 4.2 and 5.1, we have evaluated the classification performance of our proposal using the top-N recommendation approach. Top-N recommendation assumes that there are a large number of items, but the user does not have time to review all of them. Therefore, top-N recommendation aims to identify the N items that the user will most probably accept. To this end, top-N recommendation orders the items by predicted ratings and chooses the first N.

Regarding the performance metrics, in a different paper [58], Shani and Gunawardana have proposed a set of metrics to compare collaborative filters, such as accuracy, prediction, recall, sensibility, specificity, utility, diversity, coverage, and novelty. Section 4.2 justifies the selection of the metrics used in this research, which are introduced below. Table 2 summarizes the formulas for calculating these metrics:

(1) *Accuracy.* This is a measure of the closeness of agreement between a prediction and its actual value (whether the user has actually accepted it or not). It is computed as the number of items correctly classified (TP + TN) divided by the total number of items analyzed (TP + TP + FP + FN).

(2) *Precision.* This is a measure of how reliable a recommendation is. It measures the proportion of positive items correctly predicted as such (TP), among all items that have been classified as positive (TP + FP).

(3) *Recall.* This is a measure of how complete a result is. It measures the proportion of positive items predicted as such (TP) among all items that are truly positive (TP + FN), that is, among all recommendations that the user would have accepted.

Usually, a recommender aims to find items of the *class of interest*, that is, the items rated as positive, among all the available items. The problem is that usually the items of the class of interest are far less than the total number of items. This problem is known as the *class imbalance problem*. The accuracy alone is not enough to measure the performance of the recommender suffering from the class imbalance problem. This is because the recommender can reach a high accuracy by simply predicting every item as non-recommendable. When the class imbalance problem occurs, precision is a more reliable metric since it only ponders positive predictions. We can also detect that the recommender is recommending too little if the recall falls near to 0.0.

3. Proposed Solution

In this section, we describe the architecture of the service, as well as our proposal of implicit indicators to be used. Then, we describe how we have implemented the recommender, how to compute each of the intermediate matrices and their interpretation, and how to train and use the recommender.

3.1. Service Architecture. The architecture of a recommender has a great impact in the way pieces of feedback are gathered and how they are used. Therefore, this section summarizes the architecture of the recommender that we have implemented. In particular, our recommender is a software-as-a-service (SaaS) application that helps organizations be polite with their users by preventing the sending of notifications that are not likely to interest them. Figure 3 shows the roles involved in our service and the relationships between them.

In particular, there are three roles involved:

(i) *Client company.* This is the organization interested in sending notifications to its users without

Table 2: Formulas for the selected performance measures.

Accuracy = TP + TN/TP + TN + FP + FN	Precision = TP/TP + FP	Recall = TP/TP + FN

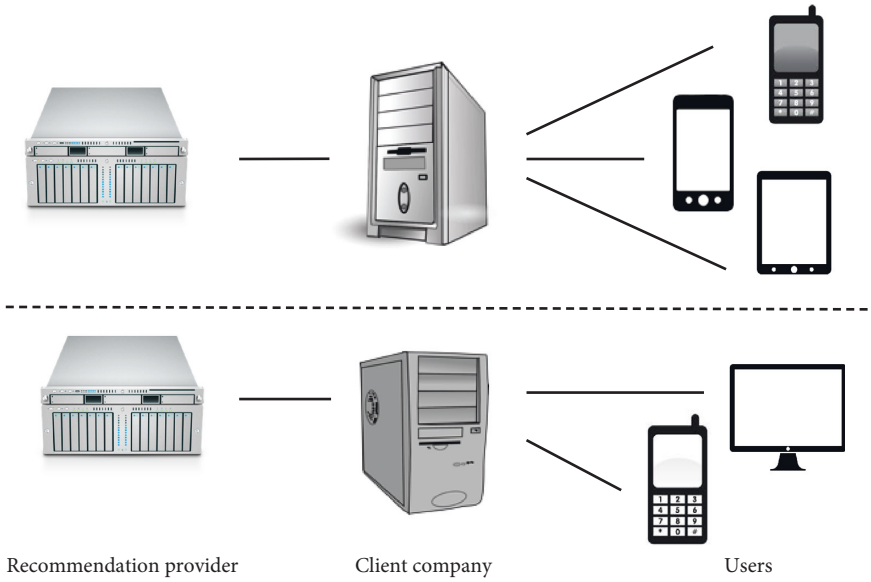


Figure 3: Service architecture (clipart source: pixabay.com).

disturbing them. Therefore, they do not want to send notifications if they are not of interest to their users. For example, a mobile application development company has a list of users. However, this is a polite company, and so does not want to bother its users by asking them to fill out surveys, or sending notifications that are not of their interest.

(ii) *Client users.* These are the users of the above client company. Obtaining implicit-feedback on these users can help send them only notifications of their interest. For example, the mobile application can collect information about the behavior of these users without disturbing them with explicit questions.

(iii) *Recommendation provider.* This is our classification service that this paper describes in more detail. The client company provides implicit-feedback to our recommendation provider. This feedback contains the interactions of its users during past notifications and corresponds to the training phase of our recommender. Given a new notification, our service returns both: (1) the top N notifications for each user and (2) a numerical prediction rating (from 0 to 10) estimating the interest in the N notifications for each user. The client company now has to decide the N number of notifications to select to each user and next from which threshold of numerical interest rating to send the notifications to its users.

Note that this architecture facilitates the service being implemented independently of the business model of the client company. For example, notifications for a cinema may be movies, and notifications for a repair shop may be promotional discounts. In this model, the client company subscribing to our SaaS forms a natural grouping of users, and information is not shared between client companies. The dashed line in Figure 3 indicates this lack of data sharing between companies.

3.2. Selecting Implicit Interest Indicators. As motivated in Section 1, we do not want to bother users asking for explicit-feedback (rating) about the content that they are interested in. Conversely, we want to represent the value of the notifications for the user by mapping the user's interactions to numerical ratings indicators. We use the 1 to 5 Likert scale to capture the level of interest-disinterest on a symmetric scale. We have identified 5 sources of useful interaction (summarized in Table 3):

(1) *Examination.* The first time the user opens a notification, we increase the estimation of the interest of the user in the content by increasing the rating of that user in this notification in +1.

(2) *Reopening.* If later on, the user reopens the notification, this indicates that the user found its content useful, and so we increase the rating by +1 for each new reopening.

(3) *Frequency.* We overweigh the interest from users with little examinations, with respect to a user who examine notifications frequency.

(4) *Fast reading.* We underweigh the interest if we detect a short reading time as an indicator of lower interest. To detect it, we study the time until the next access.

(5) *Printing.* A user who prints a document is showing interest in using it or reading it in more detail.

Table 3: Implicit interest and disinterest indicators.

N	Indicator	Value (rating)
1	Examination	Add +1 for first time it is open
2	Reopening	Add +1 for subsequent reopening
3	Frequency	Add +0.5 or −0.5 depending on whether the number of exploration is above or below the mean
4	Fast reading	Add −0.5 if we detect a reading time below 5 seconds
5	Printing	Add +1 as printing indicate interest in detailed reading
6	Sharing	Add +1 for each time the user shares it
7	Skipping	Rate 2 (dislike)
8	Deletion	Rate 1 (strongly dislike)

(6) *Sharing*. The notifications are annotated with a "share it" button. If the user shares a notification, this is an indicator that the notification has value for the person with whom the user is sharing it. Therefore, we increase the notification rating in +1 each time the user presses the "share it" button.

(7) *Skipping*. If the users go through the summary of a notification without opening it, this is an indicator that this content does not particularly interest them. Therefore, we assign a rating of 2 (dislike) to the notifications that the user has not opened.

(8) *Deletion*. Notifications are annotated with a "delete" button. If the user makes the effort of explicitly deleting a notification, this indicates that its content not only is not of their interest, but somehow it is disturbing or even offending. Therefore, we assign a rating of 1 (strongly dislike) to the notifications that the user deletes.

The maximum rating that a notification can receive is +5, as this value is an enough indicator of high interest of the user in this notification.

Note that these indicators have been chosen to ease their nondisturbing collection in different web applications. Of course, other more accurate indicators can and have been used (e.g., reading time, mouse movements, and highlighting content [2, 3]). However, the additional effort of their collection in a web page (i.e., using JavaScript client scripting) causes the websites to refuse the integration of these gathering protocols, because this may slow down the page rendering. In fact, this is what happened when we asked website administrators of our University, UNIR, to integrate these scripts to collect more advanced implicit indicators for our experiments. Therefore, we opted to develop a solution that can be integrated into websites easily and effortlessly. Nonetheless, the reader can add to the model more elaborated indicators if they find their collection feasible in their website.

Note that different website owners will be willing or able to collect different indicators. Therefore, you can optimize the initial values of the indicators proposed in Table 3 by adjusting the model parameters during the training phase. Note also that Table 3 contains variables, and this adjustment may not be necessary if the classifier automatically scales their values during the training phase.

3.3. Recommendation Approach. This section describes how our recommender predicts the most interesting notifications for each user. The process described herein performs a dimensionality reduction, so that we start with the terms of the documents and end with the topics of interest for each user.

This section describes our recommender and so uses the term "notifications," although traditionally the literature has used the term "documents." Therefore, in the rest of this document, the terms document and notification are used as interchangeable synonyms.

Our recommender is a hybrid recommender combining the two filtering approaches described in Section 2.2.5. In particular, our hybrid recommender operates in two major phases:

(1) *Content-based filtering*. The recommender uses text mining techniques to reduce the dimensionality of the words in the documents to topics in the following two steps. First, we use the *Term Document Matrix* (TDM) to identify the characteristic words of the document, that is, those words with the higher *TF-IDF* score. Second, the recommender applies a LDA algorithm (Section 2.2.4) to compute the *Document Topic Matrix* (DTM), which represents the topic proportion for each document.

(2) *Collaborative filtering*. The recommender applies collaborative filtering on the dataset to further reduce dimensionality and predict the interest of the user in an unseen notification. In particular, the recommender first creates a *Document User Matrix* (DUM) gathering the interest of the user in each document. Second, the recommender utilizes the DUM and the DTM calculated above to produce the (*Topic User Matrix*) TUM, which contains the interest of each user in each topic.

When a new unseen notification arrives, we apply the first step to identify the topics of the notification. Then, we use the second step to predict the interest of each user in the topics of the notification.

Figure 4 summarizes the flowchart and concepts that our hybrid recommender uses. In particular:

(1) *Flowchart*. The dashed arrows indicate the matrices computation steps. First, we use the indicators dataset to generate the DUM (Section 3.3.3). Second, we retrieve the documents and compute the TDM and DTM. Third, we combine the DTM and DUM in the TUM (Section 3.3.4).

(2) *Relationships*. Double arrows indicate the matrices relating these concepts. The following sections describe in more details the relationships in Figure 4 and how we compute each matrix.

3.3.1. Computing the TDM. The first task that the recommender has to do is to analyze the text of the notification and obtain a concise representation by means of the TDM, which eases the selection of the characteristic words of the notifications. This TDM performs a dimensionality

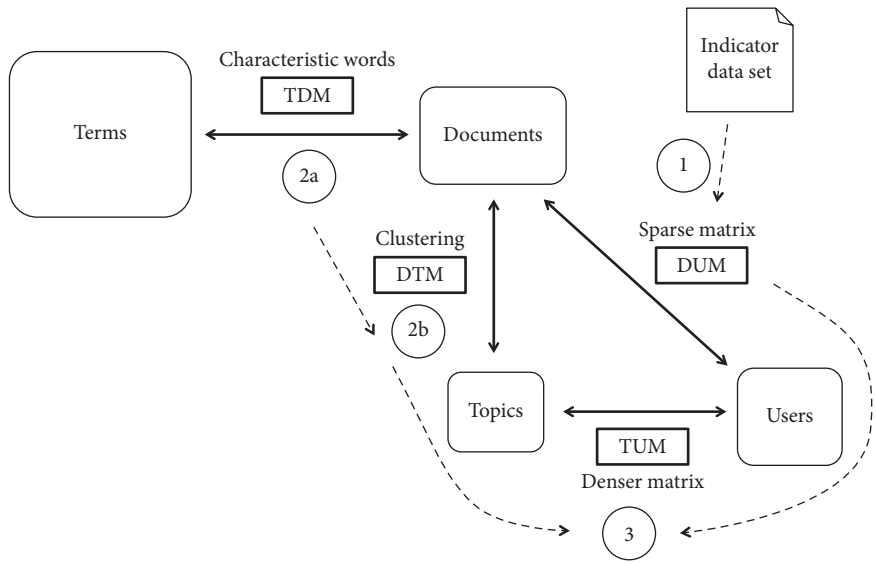


Figure 4: Flowchart and matrices relating the concepts in our hybrid recommender. The TDM (Term Document Matrix) counts the terms in each document. The DTM (Document Term Matrix) contains the estimated topic to document imputation. The DUM (Document User Matrix) indicates the user-document interest. The TUM (Topic User Matrix) contains the estimated interest of the users in each topic.

reduction by discarding uninformative words; for the remaining words, the TDM contains the *TF-IDF* scores. We compute the TDM in the following steps:

(1) *Preprocessing.* In this step, the notifications are tokenized, and then we remove stop words, white-spaces, numbers, and punctuation symbols (Section 2.2.1). We also group equivalent terms using lowercasing, accents removal, synonym grouping, and stemming. Finally, we remove words appearing only once in the whole corpus because by inspection, we have found that it is a useful way to eliminate misspelled words. Note that it is not necessary to remove repeated terms (such as headers, footers, or copyright information) because according to (2), those terms will have an $IDF(t, D) = 0$, and therefore, they will never be selected as characteristic words in the next step.

(2) *Feature selection.* In this step, we identify and ponder the characteristic words of a document. We have observed that merely counting the occurrence of terms in the notifications implies that the words more frequently used in the language are over-pondered during classification. To remove this bias, we have used the *TF-IDF* score to decide whether a word is a characteristic word of a document. In particular, as described in Section 2.2.3, this score considers the number of times that a term t appears in a document $|t \in d|$ as well as the document length $|d|$, and compares it with the number of times that this term appears in the corpus $|t \in D|$, to provide more weight to uncommon words that are appearing relatively often in the document d. These words will be the characteristic words of the document.

(3) *TDM matrix generation.* As described in Section 2.2.2, the TDM is a sparse matrix representation of the words in each document. In particular, in the TDM, a *word vector* corresponds to a column in the TDM, a row corresponds to the vocabulary, and the *TF-IDF* score indicates the level to which a word is important (i.e., a characteristic word) for a document. Therefore, we can reduce the dimensionality of the TDM by selecting the words with a higher *TF-IDF* score, that is, the less frequent words correspond to the characteristic words.

3.3.2. Computing the DTM. When two notifications have a relatively similar *TF-IDF* score for their word vectors, this is an indicator that they are dealing with the same topics. To find related notifications, we have to project them into groups with related topics by applying a standard clustering algorithm. In this way, once we determine that a notification is related to a specific topic, we assume that the notifications of its cluster deal with relatively similar topics.

As we do not want to have our customer manually labeling the topics of their notifications, we have opted to use an unsupervised clustering algorithm. In particular, we use the collapsed Gibbs sampling method as described in [59]. To determine an appropriate number of topics, we used the *binary logarithm rule* (4), where the number of nodes of the tree corresponds to the number of documents in the corpus $|D|$, and the depth of the tree corresponds to the number of topics k, that is,

$$k = \lceil \log_2 |D| \rceil. \tag{4}$$

After executing the clustering algorithm with k topics, each row in the DTM corresponds to a document, each column corresponds to a topic, and each entry is an integer

indicating the number of times words in each document were assigned to each topic.

Finally, we normalize the DTM so that all rows sum up to 1.0. In this way, an entry in the DTM will contain the levels of belonging of each document to each topic.

3.3.3. Computing the DUM. The DUM corresponds to the IUM traditionally used in collaborative filters (as described in Section 2.2.5), but where items correspond to documents, and the user's ratings are implicit. We compute the DUM by converting the collected implicit-feedback into the user's ratings according to Table 3. In particular, the rows in the DUM represent documents, and the columns represent users.

Note that we will use the DUM in two ways:

(1) *Normalized.* For content filtering, we do the normalization because, in this case, the DUM is merely an intermediate step to calculate the TUM (computed in the next step), which represents the prediction of topics of interest for each user. In particular, we normalize the DUM so that the column of each user sum to 1.0. In this way, the votes of all users weigh the same. That is, if the user has accessed several documents, the user has shown more interest in more documents, but the opinions of all users are equally important to determine the topics in the next step.

(2) *Unnormalized.* For collaborative filtering, we are searching for documents that are of interest to other similar users. Therefore, we use the DUM as is to retain information on all documents evaluated; that is, without lessening the relevance of entries of users who have interacted with more documents.

3.3.4. Computing the TUM. The above DUM has two main difficulties:

(1) There is no direct mechanism to use the DUM to predict the level of interest that a user will have when new unseen notifications arrive.

(2) The number of notifications grows rapidly, and the users do not provide any implicit-feedback for most of the notifications. As a consequence, the DUM will be a sparse matrix.

The TUM addresses both problems:

(1) The TUM relates the users to their topics of interest. Therefore, on the arrival of a new notification, we use the DTM to determine the topics of the notification, and therefore, we map these topics to the level of interest of each user.

(2) The number of topics tends to grow more slowly than the number of notifications; therefore, the TUM is denser than the DUM.

The TUM is computed as follows:

(1) The recommender normalizes the DUM so that the documents of interest for each user sum 1.0. This normalization aims to represent the relative importance of topics for each user, irrespective of how active the users are individually.

(2) The recommender multiplies the transposed DUM by the DTM to obtain the TUM. In the TUM, rows represent users, columns represent topics, and entries represent the levels of interest of each user in each topic.

Note that we use the DUM colwise normalized and the DTM rowwise normalized. Therefore, the TUM will be rowwise normalized, which means that the interest of each user for the topics will sum 1.0.

3.3.5. Recommendation Phases. As usually in automatic recommendation, our recommender also operates in two major phases described here: the *training* and the *exploitation phases*.

(1) Training phase. During this phase, we analyze the text of the training notifications and use the implicit *indicator dataset* collected in Section 4.1, to generate the TDM, DTM, DUM, and TUM. In particular, during this phase, the following tasks are executed:

(1) The recommender uses the indicator dataset to retrieve the text of the notifications.

(2) The recommender analyzes the text of the notifications and computes the TDM (Section 3.3.1).

(3) The recommender applies the topic-clustering algorithm to the TDM in order to obtain the DTM, which indicates the proportion of topics of interest in each document (Section 3.3.2).

(4) The recommender uses this indicator dataset and indicators described in Section 3.2 to create the DUM, which contains the user's interest in each document (Section 3.3.3).

(5) The recommender computes the TUM, which indicates the interest of the user in each topic (Section 3.3.4). In particular, the TUM is computed with formula (5), where DUM'_n represents the normalized and transposed DUM:

$$\mathrm{TUM} = \mathrm{DUM}'_{\mathrm{norm}} \cdot \mathrm{DTM}. \tag{5}$$

(2) Exploitation phase. During this phase, we receive a new *user*, and we have to predict a top-N recommendation list for that user with notifications that this user has not rated. This implies the following task:

(1) The recommender uses formula (6) to obtain the $\mathrm{DUM}_{\mathrm{predict}}$, which predicts the level of interest of the user for each unseen document.

$$\mathrm{DUM}_{\mathrm{precict}} = \mathrm{TUM} \cdot \mathrm{DTM}'. \tag{6}$$

Then, we normalize the $\mathrm{DUM}_{\mathrm{precict}}$ so that all rows sum up to 1.0. Lastly, we select the higher predicted

scores up to $1-1/k$ for the user. Note that formula (4) defines k clustering topics for 2^k documents. Thus, $1-1/k$ approaches 1.0 as the number of topics (and also documents) increases. This means that the more the documents there are, the higher the threshold will be for selecting a document.

(2) We create a top-N recommendation list in the following way:

(a) *Content filtering phase.* The recommender selects the documents for which the predicted user interest is above $1-1/k$ threshold, where k is the number of topics (4). This phase uses the content similarity criterion to recommend content.
If the above phase is not enough to obtain N unseen documents, this means that there is no more content on the topics that are of interest to the user. Then, we activate the collaborative filter.

(b) *Collaborative filtering phase.* We compute the user-user similarity to find the documents that have been of interest to similar users. In particular, we compute the *User User Matrix* (UUM) using the cosine similarity formula (8). Then, we use the higher rated documents by the most similar users to complete the top-N recommendation list. This phase uses the diversity criterion to recommend content.

4. Methods

This section summarizes how we have created the dataset for evaluating our proposal, evaluation criteria, and protocol for evaluating the classification performance. We also describe how we have validated dimensionality reduction; that is, the selection of the characteristic words of each document and the unsupervised model to cluster by topics.

4.1. Experiments Setup. Before initiating this collection, we have reviewed different datasets (Section 2.1.1). However, we found these datasets inappropriate for our research, because we need implicit indicators, such as the ones defined in Table 3 for notifications of a company. Therefore, we have accomplished the collection of our implicit indicators in the *indicator dataset*.

4.1.1. Collecting the Notifications. As described in Section 4, our ultimate goal is to create a recommendation provider that helps client companies customize the sending of notifications to their users. As we have not been able to find a standard dataset containing these notifications from a company, we initiated the construction of our own dataset with the resources we have in UNIR. In particular,

(i) The notifications we have used in our experiments are blog posts from a list of RSS URLs in Spanish at UNIR Revista (http://www.unir.net/vive-unir/). UNIR shows the students these blog posts in the front page of a number of virtual courses (4 graduate courses and 20 postgraduate courses, all of them on different topics related to technology and engineering). Therefore, these blog posts resemble the notifications that we want to simulate. RSS and Atom are standard protocols for publishing blog posts. As they use a well-defined XML format, this content can be easily collected.

(ii) With this dataset, we are assuming that the blog posts are equivalent to the notifications that we intend to evaluate. However, the performance of blog posts vs. notifications recommenders is not always directly comparable. For instance, notifications are often extremely short texts, compared to blog posts.

(iii) The indicators dataset we currently have is a dataset in progress. Though currently it is a small dataset, we have published it.

4.1.2. Collecting the User Interactions. To collect the implicit indicators for our experiments, we have published the abovementioned blog posts in the front page of the virtual courses of various subjects that have taken place in the academic year 2018-2019 at our university (UNIR).

The protocol to show the blog post to the student has been as follows:

(1) When student enters the classroom, the latter 5 blog posts are shown.

(2) Merely clicking on its title redirects the user to the blog post, and we record the date, user ID, and visited URL.

(3) We have been registering this activity for 1 month.

Although the collection of this dataset is a work in progress, at the time of writing this article, we have obtained the interactions of more than 100 students. With the logs of this activity, we are able to collect indicators 1–4 of Table 3.

4.2. Evaluation Criteria and Protocol. Since our approach aims to be nondisturbing, we are interested in selecting the best notifications from a large number of notifications. Therefore, we measured the classification performance using the top-N recommendation task (Section 2.2.6).

To measure the classification performance of our implicit-feedback recommender, we follow a leave-one-out approach. In particular:

(1) The recommender iterates the users in the DUM in which each nonempty entry indicates the actual interest of the user in this notification. We remove a nonempty entry from the DUM for each user with 2 or more entries. That is, we need at least one remaining rating in the DUM to know something about the user. The *removed document* will be the document to be tested, the corresponding user is the *test user*, and the new matrix will be the *test DUM*.

(2) The recommender uses test DUM to regenerate the TUM as described in Section 3.3.4.

(3) The recommender obtains the top-N recommendation list, executing the prediction phase described in Section 3.3.5. In this top-N recommendation list, N is the number of documents in the *user list*, i.e., the number of elements for which the user has shown interest according to the original DUM.

(4) We contrast the user list with the top-N recommendation list measuring the accuracy, precision, and recall. In particular, for each recommendation in the top-N recommendation list, the confusion matrix is generated according to the imputation rules described in Table 4.

4.3. Dimensionality Reduction Model Validation. In addition to evaluating the classification performance, we validate the consistency of the dimensionality reduction implemented in the content-based filter. In particular, the content-based filter implements two reductions of dimensionality:

(1) *Characteristic words of a document.* We aim to validate the semantic coherence between the characteristic words chosen to represent the documents and the latent topics. The results are reported in Section 5.2.

(2) *Unsupervised topic clustering.* As LDA topic clustering is unsupervised, we wonder whether the obtained clusters are semantically adequate. For this purpose, we have studied the coherence and convergence, as described below. The results are reported in Sections 5.3 and 5.4.

4.3.1. Coherence. To determine to what extent documents classified by topic are coherent with the documents classified by characteristic words, we have calculated the *Document Distance Matrix* (DDM) using two features:

(i) The distance among documents according to the *TF-IDF* score: $\mathrm{DDM}_{TF\text{-}IDF}$

(ii) The distance among documents according to the proportion of topic imputation: $\mathrm{DDM}_{\mathrm{topic}}$

If these features are coherent, the difference between both will be low:

$$\text{heatmap} = \left|\mathrm{DDM}_{TF-IDF} - \mathrm{DDM}_{\mathrm{topics}}\right|. \tag{7}$$

To calculate the DDM, we first measure the similarity between documents as the degree to which the features (either *TF-IDF* or topics) overlap. For this purpose, we use the cosine similarity between the features vectors of each pair of documents. The cosine similarity takes the sum of the n features product normalized by the product of their Euclidean lengths. In particular, for the documents with word vectors $\mathbf{u}$, $\mathbf{v}$, the cosine similarity is defined as

$$\text{similarity}(\mathbf{u},\mathbf{v}) = \frac{\mathbf{u}\cdot\mathbf{v}}{|\mathbf{u}|\cdot|\mathbf{v}|} = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2}\sqrt{\sum_{i=1}^{n} v_i^2}}. \tag{8}$$

Table 4: Confusion matrix classification rules.

Imputation	Description
TP	The recommended document is in the user list
TN	A nonrecommended document is not in the user list
FP	A recommended document is not in the user list
FN	A nonrecommended document is in the user list

In general, the cosine similarity ranges from −1.0, meaning exactly opposite vectors, to 1.0, meaning exactly the same vectors. However, as the vector values are all positives, (8) will range from 0.0 (completely disjointed documents) to 1.0 (the same document).

To use the similarity measure in (8) as a distance metric, we use the following formula:

$$\text{distance}(\mathbf{u},\mathbf{v}) = 1.0 - \text{similarity}(\mathbf{u},\mathbf{v}). \tag{9}$$

The resulting DDM is a squared symmetric matrix where the entry on row i and column j represents the distance between documents d_i and d_j.

4.3.2. Convergence. The collapsed Gibbs sampling method [59] repetitively iterates all words in all documents updating prior and posterior probabilities of the hidden variables (topics). After a number of iterations, the model tends to converge to a stable topic assignment state. The *perplexity index* [60] has been proposed to determine when the model is fitted, and we can stop the iterations. Basically, this index computes the likelihood of the parameters given the observations. The perplexity is defined as the natural log of two likelihood values:

(i) *Full likelihood.* The log-likelihood including the prior.

(ii) *Assignments likelihood.* The log-likelihood of the observations conditioned to the assignments.

A lower likelihood score indicates better generalization performance. Section 5.4 studies this convergence.

5. Result and Discussion

This section provides the results of the classification performance as well as a validation for the coherence and convergence of our dimensionally reduction approach.

5.1. Classification Performance. Figure 5 shows the classification performance of executing the above evaluation protocol. You can obtain the numerical values of this figure in the file *evaluation.R*. The horizontal axis shows the time evolution, and the vertical axis shows the classification performance using the indicators accumulated up to the day of the evaluation.

Inspecting the collected indicator dataset, we found that notification recommendation suffers from the class imbalance problem (Section 2.2.6), that is, the user does not show interest in most of the notifications that were presented.

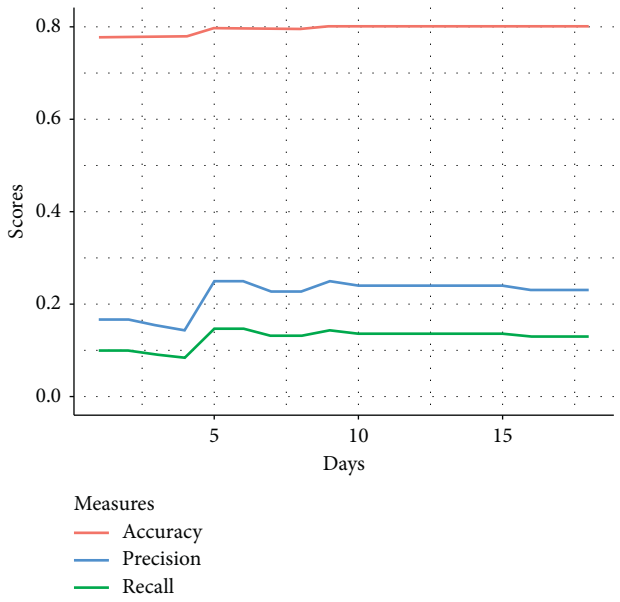


Figure 5: Performance of the top-N recommendation using users with 1 or more interactions.

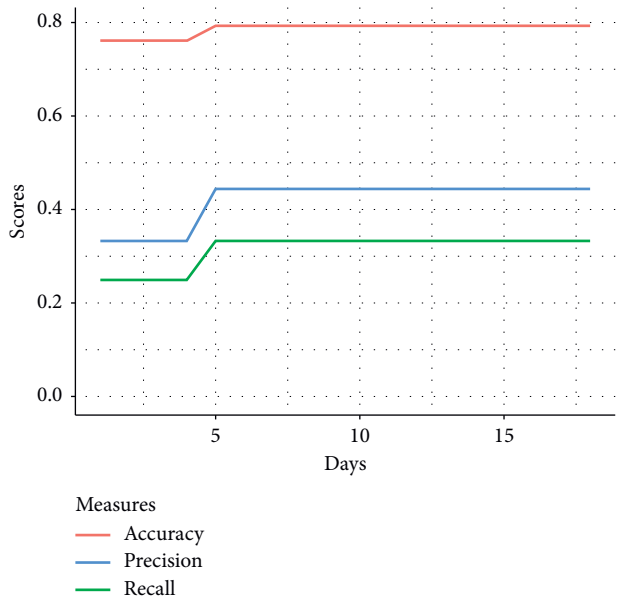


Figure 6: Performance of the top-N recommendation using users with 2 or more interactions.

Similarly, the top-N recommendation list also suffers this effect, as most of the notifications are not in this list. Therefore, accuracy overestimates the classification performance of the recommender (Figure 5). Nonetheless, we have included accuracy in our analysis to ease the comparison, as it is a standard metric in most of the state-of-the-art recommenders.

Limiting false positives is essential to avoid the lack of trust that occurs whenever the recommender returns a noninteresting notification. Precision indicates the ability of the recommender to create a top-N recommendation list that resembles the user list, that is, without FP. This is an ambitious goal, because the precision formula (Table 2) compares how many times we succeed against how many times we make a wrong recommendation, but disregards all the documents that were correctly filtered (i.e., TN). This fact justifies the high difference between accuracy and precision in Figure 5. That is, approximately, only in 13% of the cases, the recommender is able to correctly guess which is the leave-one-out element.

To best estimate the classification performance, we have also added recall to Figure 5, which indicates the completeness of the actual top-N notifications for the user.

The most obvious way to increase precision and recall is to increase the number of user interactions. Figure 6 shows how precision and recall improve if we repeat the above evaluation using only those users who have interacted with 2 or more documents.

Other authors obtain the same effect increasing the number of user interactions. For instance, [26] follows our top-N performance evaluation approach; when they use 10 elements in the list, they obtain a precision of 0.22 and a recall of 0.25. If they increase these elements to 30, they achieve a precision of 0.32 and a recall of 0.42. Similar figures are obtained by the system presented in [28], in which the recall increases from 0.25 to 0.41 when the number of elements increases from 10 to 30. Also, the system presented in [27] gets higher f-measure values as the number of elements in the list increases. Unfortunately, we have not been able to reproduce their experiments with our indicators dataset because our dataset is sparser and we barely have users with more than 3 interactions. Notice that although the results are similar, datasets used in the experiments are not the same, and so the results cannot be directly compared.

The authors of [19] also combine content-based and collaborative filtering, by targeting sports news, which is a further controlled domain than ours. In addition, they combine explicit rating for collaborative filtering, with implicit-feedback for content-based filtering by counting the number of times the user accesses the news. To evaluate the system, they analyze the user clicks (of around 5000 users during 10 days) and they find that 27% of the recommended articles are viewed, being 50% of recommended articles removed. Although the evaluation method is user based (i.e., it is not offline), we find these results to be in concordance with the precision obtained in the other studies mentioned above.

Finally, it is worth to mention that the design of all these experiment assumes that the users always choose the documents that are of maximum interest for them. However, it is known that the behavior of users on the Internet is impulsive and explorative [61]. Therefore, we hypothesize that part of these relatively low precision and recall scores are due to the fact that users access the documents without analyzing in details which are the most interesting for them. To further analyze this hypothesis, we would have to ask the user, which would involve comprehensive fieldwork for future work. An argument in favor of this hypothesis is that the user's choices are only based on the title of the post, while the recommender analyzes the entire text thoroughly.

Table 5: The top words and representative document for each cluster.

Doc ID	Top characteristic words		Title of the most representative document
2	Eficiencia	Mejora	Luis lizasoain: "Cualquier centro de cualquier tipo puede ser de alta eficacia"
3	Colombia	Empresa	Los nuevos graduados se suman a la creciente familia colombiana de UNIR
4	Derecho	Seguridad	Nace la escuela sagardoy de derecho del trabajo

Note that top characteristic words have a high semantic relationship with the document that best represents each topic.

Although implicit-feedback is easier to obtain, it is a challenge to convert raw data into user ratings because implicit-feedback is inherently noisy. Given that the ratings are somehow artificially created from implicit data, a confidence level may be considered to gauge the confidence in the estimated ratings. Particularly, there are some studies finding that reducing confidence in the preferences of those users with more intense activity improves the performance [25].

5.2. Representative of Each Topic. To determine the representativeness of the characteristic words and topics, we have used our dataset to generate the 2 top words in each topic, as well as the document that best represents each topic. The rows in Table 5 correspond to the topics. For each topic, we show the two most representative words along with the title of the most representative post for this topic.

The DTM represents the assignments of the document to the topics. Figure 7 shows the distribution of topics across all the documents. You can obtain the numerical values of this figure in the file *validation.R*. Note that the representative documents in Table 5 match the documents with the highest proportion of the corresponding topic in Figure 7.

5.3. Coherence. This section studies the coherence of the dimensionality reduction approaches by measuring the difference among the documents classification according to the $\mathrm{DDM}_{TF\text{-}IDF}$ and the $\mathrm{DDM}_{\mathrm{topic}}$ (Section 4.3.1 for further details).

Remember that formula (9) calculates this distance between these matrices, where 0.0 means the same document and 1.0 means completely disjointed documents. Figure 8 shows the heatmap of this difference (7). You can obtain the numerical values of this figure in the file *validation.R*. Light colors indicate high coherence; that is, low difference between both approaches to measuring distances. Note that the heatmap is symmetric, and both metrics reach maximum coherence when both documents are equal (the main diagonal). In general, both metrics give similar distances, and so the heatmap is light. The darkest squares correspond to a lower coherence; that is, the documents are not receiving the same distance with both approaches.

5.4. Convergence. Our LDA clustering algorithm uses four parameters:

(i) *Number of topics.* We use the binary logarithm rule (4). Sections 5.2 and 5.3 discuss the suitability of this parameter.

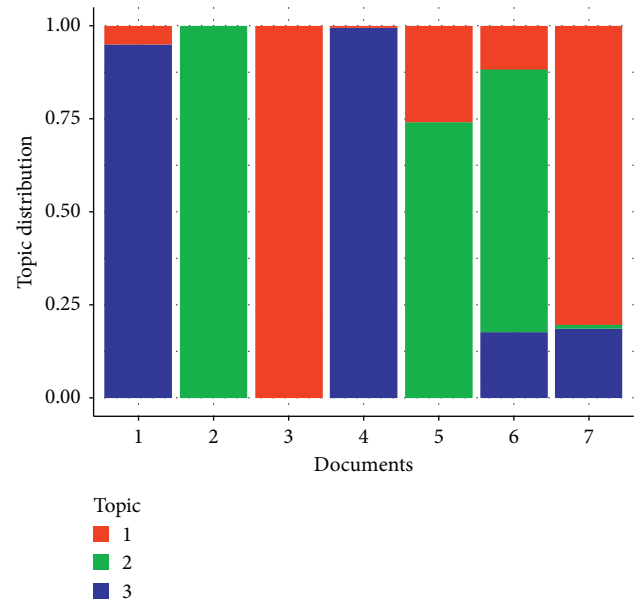


Figure 7: Distribution of topics in the documents.

(ii) *Iterations and learning rates.* We execute $i = 100$ iterations. The $\alpha = 0.1$, $\beta = 0.1$ probabilities can be interpreted as learning rates. Section 5.4 discusses the suitability of these parameters.

Figure 9 shows the convergence of our dataset with the learning rates $\alpha = 0.1$, $\beta = 0.1$, and $i = 100$ iterations. You can obtain the numerical values of this figure in the file *validation.R*. A greater log-likelihood is considered more adequate for the parameters. We can observe that after 15 iterations, the model has stabilized.

6. Conclusions

This paper shows how the gathering of pieces of implicit-feedback is enough to personalize content delivery, that is, without the need to disturb the user by asking them to fill in additional personal information. The recommender operates autonomously and automatically with standard data mining techniques, so its use does not imply an additional cost of adding to the notifications metadata (as is usually the case with other content-based and knowledge-based recommenders). The recommender is able to select content for users with a similar profile using standard collaborative-filtering techniques. The adding of content-based filtering allows us to effectively address the cold-start problem (a limitation of pure collaborative filters). In particular, it is

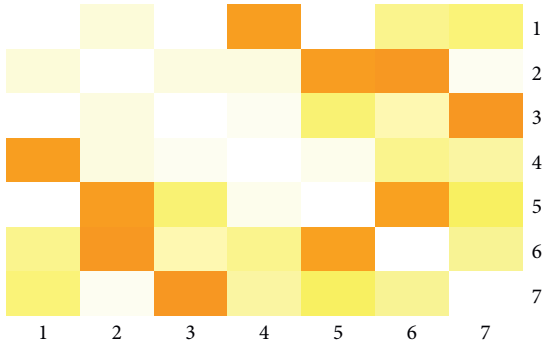


Figure 8: Heatmap of the difference between $DDM_{TF\text{-}IDF}$ and DDM_{topics}.

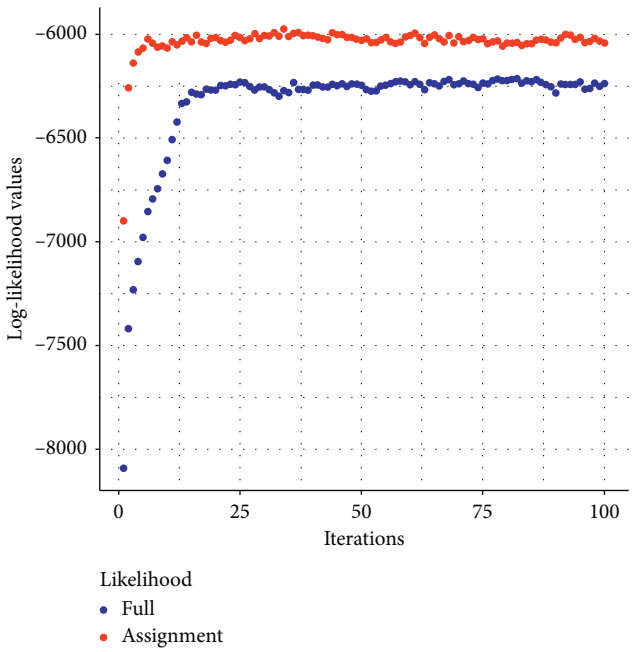


Figure 9: Convergence.

enough that a user has chosen a single document to determine the topics of interest and initiate recommendations on similar topics.

6.1. Future Work. We have identified three areas of future work:

(1) Our recommender has been evaluated offline, without the explicit participation of the user in the evaluation. However, according to some studies, great offline performance does not necessarily mean online success [62]. Therefore, it is important to also consider the perceived utility of recommendations by the user in future work. This work would also allow us to analyze the hypothesis laid out in Section 5.1: to what extent the user exhaustively analyzes or impulsively chooses the documents [61].

(2) The user interests change over time [26]. Therefore, as future work, we may introduce some temporal mechanism that model the gradual decay of the relevance of past readings [26, 29], or the user trends [33].

(3) Finally, the evaluation of the classification performance has been implemented with a relatively small dataset, which also does not include all the indicators defined in Table 3. For this reason, we want to increase the volume and type of implicit indicators and update our published dataset.

Data Availability

The anonymized dataset and source code needed to reproduce these results are publicly available at https://github.com/ifernandolopez/hybrid-filter.

Disclosure

The authors have not published the raw dataset, but an anonymized version of it.

Conflicts of Interest

The authors declare that there are no conflicts of interests.

Acknowledgments

The authors thank the technical department of UNIR for creating the tool that collects the user interaction of several virtual courses.

References

[1] J. S. Keem and S. Lee, "A study on consumers' experiences and avoidances of mobile shopping application advertisements," *Journal of Global Fashion Marketing*, vol. 9, no. 2, pp. 148–160, 2018.

[2] E. R. Núñez-Valdez, J. M. C. Lovelle, G. I. Hernández, A. J. Fuente, and J. E. Labra-Gayo, "Creating recommendations on electronic books: a collaborative learning implicit approach," *Computers in Human Behavior*, vol. 51, pp. 1320–1330, 2015.

[3] M. Claypool, P. Le, M. Wased, and D. Brown, "Implicit interest indicators," in *Proceedings of the 6th International Conference on Intelligent User Interfaces-IUI '01*, pp. 33–40, Santa Fe, NM, USA, January 2001.

[4] G. Linden, B. Smith, and J. York, "Amazon.com recommendations: item-to-item collaborative filtering," *IEEE Internet Computing*, vol. 7, no. 1, pp. 76–80, 2003.

[5] R. Katarya, "Movie recommender system with metaheuristic artificial bee," *Neural Computing and Applications*, vol. 30, no. 6, pp. 1983–1990, 2018.

[6] R. Katarya and O. P. Verma, "Efficient music recommender system using context graph and particle swarm," *Multimedia Tools and Applications*, vol. 77, no. 2, pp. 2673–2687, 2018.

[7] J. Beel, B. Gipp, S. Langer, and C. Breitinger, "Research-paper recommender systems: a literature survey," *International Journal on Digital Libraries*, vol. 17, no. 4, pp. 305–338, 2016.

[8] X. Hu, W. Zhu, and Q. Li, "HCRS: a hybrid clothes recommender system based on user ratings and product features," 2014, http://arxiv.org/abs/1411.6754.

[9] J. P. Lucas, N. Luz, M. N. Moreno, R. Anacleto, A. Almeida Figueiredo, and C. Martins, "A hybrid recommendation approach for a tourism system," *Expert Systems with Applications*, vol. 40, no. 9, pp. 3532–3550, 2013.

[10] D. Horowitz, D. Contreras, and M. Salamó, "EventAware: a mobile recommender system for events," *Pattern Recognition Letters*, vol. 105, pp. 121–134, 2018.

[11] M. R. Ghorab, D. Zhou, A. O'Connor, and V. Wade, "Personalised information retrieval: survey and classification," *User Modeling and User-Adapted Interaction*, vol. 23, no. 4, pp. 381–443, 2013.

[12] M. Kunaver and T. Požrl, "Diversity in recommender systems-a survey," *Knowledge-Based Systems*, vol. 123, pp. 154–162, 2017.

[13] M. Karimi, D. Jannach, and M. Jugovac, "News recommender systems-survey and roads ahead," *Information Processing & Management*, vol. 54, no. 6, pp. 1203–1227, 2018.

[14] M. H. Aghdam, M. Analoui, and P. Kabiri, "Analiysis of self-similarity in recommender systems," in *Proceedings of 2014 Iranian Conference on Intelligent Systems (ICIS)*, pp. 1–4, Bam, Iran, February 2014.

[15] A. Montes-García, J. M. Álvarez-Rodríguez, J. E. Labra-Gayo, and M. Martínez-Merino, "Towards a journalist-based news recommendation system: the Wesomender approach," *Expert Systems with Applications*, vol. 40, no. 17, pp. 6735–6741, 2013.

[16] O. Sanjuán, E. Torres, H. Castán, R. Gonzalez, C. Pelayo, and L. Rodriguez, *Viabilidad de la Aplicación de Sistemas de Recomendación a Entornos de e-Learning*, University of Oviedo, Oviedo, Spain, 2009.

[17] E. Mannens, S. Coppens, T. De Pessemier et al., "Automatic news recommendations via aggregated profiling," *Multimedia Tools and Applications*, vol. 63, no. 2, pp. 407–425, 2013.

[18] N. Jonnalagedda, S. Gauch, K. Labille, and S. Alfarhood, "Incorporating popularity in a personalized news recommender system," *PeerJ Computer Science*, vol. 2, p. e63, 2016.

[19] P. Lenhart and D. Herzog, "Combining content-based and collaborative filtering for personalized sports news recommendations," in *Proceedings of ACM Conference on Recommender Systems CBRecSys@ RecSys*, pp. 3–10, Boston, MA, USA, 2016.

[20] K. Bagherifard, M. Rahmani, M. Nilashi, and V. Rafe, "Performance improvement for recommender systems using ontology," *Telematics and Informatics*, vol. 34, no. 8, pp. 1772–1792, 2017.

[21] S. Agarwal and A. Singhal, "Handling skewed results in news recommendations by focused analysis of semantic user profiles," in *Proceedings of 2014 International Conference on Reliability Optimization and Information Technology (ICROIT)*, pp. 74–79, USA, February 2014.

[22] M. Y. Hsieh, T. H. Weng, and K. C. Li, "A keyword-aware recommender system using implicit feedback on Hadoop," *Journal of Parallel and Distributed Computing*, vol. 116, pp. 63–73, 2018.

[23] N. Jiang, "Implicit feedback recommender system based on matrix factorization," in *Proceedings of the 12th EAI International Conference on Testbeds and Research Infrastructures for the Development of Networks & Communities*, pp. 77–85, Wuhan, China, September 2018.

[24] E. R. Núñez-Valdez, D. Quintana, R. González Crespo, P. Isasi, and E. Herrera-Viedma, "A recommender system based on implicit feedback for selective dissemination of ebooks," *Information Sciences*, vol. 467, pp. 87–98, 2018.

[25] J. Hu, J. Liang, Y. Kuang, and V. Honavar, "A user similarity-based Top-N recommendation approach for mobile in-application advertising," *Expert Systems with Applications*, vol. 111, pp. 51–60, 2018.

[26] L. Li, L. Zheng, F. Yang, and T. Li, "Modeling and broadening temporal user interest in personalized news recommendation," *Expert Systems with Applications*, vol. 41, no. 7, pp. 3168–3177, 2014.

[27] L. Li and T. Li, "News recommendation via hypergraph learning: encapsulation of user behavior and news content," in *Proceedings of the sixth ACM international conference on Web search and data mining*, pp. 305–314, Rome, Italy, February 2013.

[28] L. Zheng, L. Li, W. Hong, and T. Li, "PENETRATE: personalized news recommendation using ensemble hierarchical clustering," *Expert Systems with Applications*, vol. 40, no. 6, pp. 2127–2136, 2013.

[29] M. Lu, Z. Qin, Y. Cao, Z. Liu, and M. Wang, "Scalable news recommendation using multi-dimensional similarity and Jaccard-Kmeans clustering," *Journal of Systems and Software*, vol. 95, pp. 242–251, 2014.

[30] W. J. Lee, K. J. Oh, C. G. Lim, and H. J. Choi, "User profile extraction from Twitter for personalized news recommendation," in *Proceedings of 16th International Conference on Advanced Communication Technology*, pp. 779–783, Pyeongchang, South Korea, February 2014.

[31] W. Gu, S. Dong, Z. Zeng, and J. He, "An effective news recommendation method for microblog user," *Scientific World Journal*, vol. 2014, Article ID 907515, 14 pages, 2014.

[32] J. Zheng and Y. Wang, "Personalized recommendations based on sentimental interest community detection," *Scientific Programming*, vol. 2018, Article ID 8503452, 14 pages, 2018.

[33] R. C. Bagher, H. Hassanpour, and H. Mashayekhi, "User trends modeling for a content-based recommender system," *Expert Systems with Applications*, vol. 87, pp. 209–219, 2017.

[34] C. N. Ziegler, S. M. McNee, J. A. Konstan, and G. Lausen, "Improving recommendation lists through topic diversification," in *Proceedings of the 14th international conference on World Wide Web-WWW '05*, p. 22, Chiba, Japan, May 2005.

[35] K. Goldberg, T. Roeder, D. Gupta, and C. Perkins, "Eigentaste: a constant time collaborative filtering algorithm," *Information Retrieval*, vol. 4, no. 2, pp. 133–151, 2001.

[36] H. J. Lee and S. J. Park, "MONERS: a news recommender for the mobile web," *Expert Systems with Applications*, vol. 32, no. 1, pp. 143–150, 2007.

[37] E. Keogh and A. Mueen, "Curse of dimensionality," in *Encyclopedia of Machine Learning and Data Mining*, pp. 314-315, Springer, Berlin, Germany, 2017.

[38] G. C. Banks, H. M. Woznyj, R. S. Wesslen, and R. L. Ross, "A review of best practice recommendations for text analysis in R (and a user-friendly app)," *Journal of Business and Psychology*, vol. 33, no. 4, pp. 445–459, 2018.

[39] L. Wetzel, *Types and Tokens*, Stanford Encyclopedia of Philosophy, USA, 2018.

[40] J. Singh and V. Gupta, "A systematic review of text stemming techniques," *Artificial Intelligence Review*, vol. 48, no. 2, pp. 157–217, 2017.

[41] B. Banerjee, T. Sarkar, P. Chakraborty, and A. R. Pal, "A comparison between extrinsic and intrinsic technique for multi-document text summarization," in *2017 2nd IEEE International Conference on Recent Trends in Electronics, Information & Communication Technology (RTEICT)*, pp. 768–772, Bengaluru, India, May 2017.

[42] C. Y. J. Wen, "Text categorization based on a similarity approach," in *Proceedings of International Conference on*

Intelligent System and Knowledge Engineering, Chengdu, China, October 2007.

[43] D. Forsyth, "Markov chains and hidden Markov models," in *Proceedings of Probability and Statistics for Computer Science*, pp. 331–351, Springer International Publishing, Cham, Switzerland, January 2018.

[44] J. Sun, X. Zhang, D. Liao, and V. Chang, "Efficient method for feature selection in text classification," in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, Antalya, Turkey, August 2017.

[45] A. Moreno and T. Redondo, "Text analytics: the convergence of big data and artificial intelligence," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 3, no. 6, p. 57, 2016.

[46] H. Cordobés, A. F. Anta, L. F. Chiroque, F. Pérez, T. Redondo, and A. Santos, "Graph-based techniques for topic classification of tweets in Spanish," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 2, no. 5, p. 31, 2014.

[47] Y. Boulid, A. Souhar, and M. Ouagague, "Spatial and textural aspects for Arabic handwritten characters recognition," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 5, no. 1, p. 86, 2018.

[48] Y. Chen and X. Wang, "Text feature extraction based on joint conditional entropy," in *Proceedings of 2012 2nd International Conference on Computer Science and Network Technology*, pp. 2055–2058, Changchun, China, December 2012.

[49] Z. Yun-tao, G. Ling, and W. Yong-cheng, "An improved TF-IDF approach for text classification," *Journal of Zhejiang University SCIENCE*, vol. 6, no. 1, pp. 49–55, 2005.

[50] F. Horn, L. Arras, G. Montavon, K.-R. Müller, and W. Samek, "Discovering topics in text datasets by visualizing relevant words," 2017, http://arxiv.org/abs/1707.06100.

[51] H. Jelodar, Y. Wang, C. Yuan et al., "Latent dirichlet allocation (LDA) and topic modeling: models, applications, a survey," *Multimedia Tools and Applications*, pp. 1–43, 2018, In press.

[52] Á. M. Navarro and P. Moreno-Ger, "Comparison of clustering algorithms for learning analytics with educational datasets," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 5, no. 2, p. 9, 2018.

[53] M. Anderka and B. Stein, "The ESA retrieval model revisited," in *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval-SIGIR '09*, pp. 670-671, Boston, MA, USA, July 2009.

[54] M. J. Pazzani and D. Billsus, "Content-based recommendation systems," in *The Adaptive Web*, pp. 325–341, Springer-Verlag, Berlin, Germany, 2007.

[55] R. Jacoby and B. O'Kane, "System and method for tagging content and delivering the tag to buddies of a given user," US9848246B2 Patent, 2017.

[56] M. Jalili, "A survey of collaborative filtering recommender algorithms and their evaluation metrics," *International Journal of System Modeling and Simulation*, vol. 2, no. 2, pp. 14–17, 2017.

[57] G. Shani and A. Gunawardana, "Evaluating recommendation systems," in *Recommender Systems Handbook*, pp. 257–297, Springer, Berlin, Germany, 2011.

[58] G. Shani and A. Gunawardana, "Tutorial on application-oriented evaluation of recommendation systems," *AI Communications*, vol. 26, no. 2, pp. 225–236, 2013.

[59] Y. Zhao and Y. Cen, *Data Mining Applications with R*, Elsevier, Amsterdam, Netherlands, 2013.

[60] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.

[61] Y. Chen, Y. Lu, B. Wang, and Z. Pan, "How do product recommendations affect impulse buying? An empirical study on WeChat social commerce," *Information & Management*, vol. 56, no. 2, pp. 236–248, 2018.

[62] F. Garcin, B. Faltings, O. Donatsch, A. Alazzawi, C. Bruttin, and A. Huber, "Offline and online evaluation of news recommender systems at swissinfo.ch," in *Proceedings of the 8th ACM Conference on Recommender systems-RecSys '14*, pp. 169–176, Silicon Valley, CA, USA, October 2014.