International Journal of Data Science and Analytics
https://doi.org/10.1007/s41060-023-00418-4

REGULAR PAPER

Improving recommendation diversity and serendipity with an ontology-based algorithm for cold start environments

Stanislav Kuznetsov[1] · **Pavel Kordík**[1]

Received: 25 November 2022 / Accepted: 15 June 2023

Abstract
Every real-life environments where users interact with items (products, films, research expert profiles) have several development phases. In the Cold-start phase, there are almost no interactions among users and items content-based recommendation systems (RS) can only recommend based on matching the attributes of the items. In the transition state, items start to collect user interactions but still a significant number of items have too small number of interactions, RS does not allow users to discover cold items. In a regular state, where most of the items in the system have enough interactions, the recommendations often suffer from low diversity of the items within a single recommendation. This article proposes a general recommendation algorithm based on Ontological-similarity, which is designed to address all the above problems. Our experiments show that recommendations generated by our approach are consistently better in all environment development phases and increase the success rate of recommendations by almost 50% measured using ontology-aware recall, which is also introduced in this article.

Keywords Recommender system · Ontology · Cold-start problem · Diversity · Serendipity

1 Introduction

Online stores offer thousands of products, such as movies, songs, and books, we call these products items, and the set of products call a catalogue. Additionally, companies want to provide as many different items as possible for two reasons: Firstly, to increase their revenues and secondly, to increase the probability of user purchases. [1]. The wide variety of items is also called diversity.

With many available items, finding one that the user would like to buy can be challenging. We should start to compute users' preferences (interactions) and recommend satisfying items. One of the valuable algorithms is CF (Collaborative-filtering) based on user interactions.

We always want to sell all products in stock, so we should recommend as many items as possible; we call it Catalogue-coverage. Next, in a retail business, we always have popular items (best-sellers) and many other items that companies want to sell. The set of these items is called a Long-tail catalogue. After some time, we would like to occasionally surprise the user by showing them a product that they want, is new to them and is unexpected, i.e., one that the user would not have found on their own. This concept is called serendipity.

Every business grows and goes through several phases. Initially, the catalogue has mostly items without user interaction; it is called the Cold-system problem or Cold-start problem [2]. Also, if there is a new item in our catalogue, we call it a cold-item. Then, there is a transition phase where the number of user interactions grows. However, the average number of interactions is still low, and the catalogue still has some items without interactions. We call the last phase a regular because, in this phase, the average number of user interactions is enough to use the CF algorithms.

This paper presents a Universal Ontological-Based Algorithm (OBACS) that uses diversification method, and we call it "diversity" to improve user experience in all three phases. Also, our algorithm could reduce Cold-start problems in the initial phase, where we cannot use CF algorithm and recommend cold-items.

✉ Stanislav Kuznetsov
kuznesta@fit.cvut.cz

Pavel Kordík
kordikp@fit.cvut.cz

[1] Department of Applied Mathematics, Faculty of Information Technology, Czech Technical University in Prague, Thákurova 9, 16000 Prague, Czech Republic

Published online: 28 July 2023

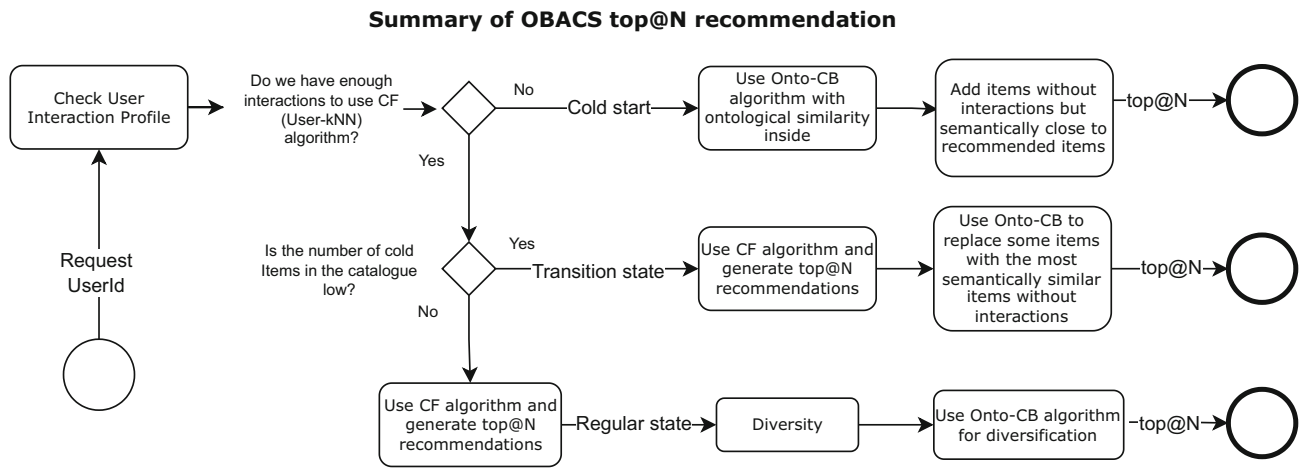

Fig. 1 The diagram shows the high-level summary of OBACS algorithms diagram (BPMN)

Our solution is to use algorithms that can calculate the similarities of the text attributes of new items with existing ones. Regarding CB Content-based algorithms based on attribute similarity, our approach uses the ontology approach and knowledge graph to calculate item similarities. Ontology has the advantage of adding a semantic layer to measure text similarities, even if the words in the text are different but have a similar meaning. The benefits of ontology over other approaches are explained in our previous work [3]. We introduce the Ontological-similarity 3.2.1 for computing semantic similarities and Onto-recall 3.2.2 metrics, which works with semantic similarity inside recommendations.

We present the benefits of our solution in three experiments that simulate individual phases of business development. For comparison with previous research, we chose the movie domain.

The paper is organized as follows. Section 2 provides the background methods that we use in our research; Sect. 3 describes our approach in detail. In Sect. 4, we provide improvements over state-of-the-art. Section 5 presents the result of our experiments. Finally, the discussion and conclusion are presented in Sect. 6 and Sect. 7 (Table 1).

2 Background

In this section, we describe three diversity concepts that we can use to improve the user experience and improve the coverage of the catalog. Next, we describe the difference between ontology and the knowledge graph and how we use both concepts in our approach. Also, we present our CF algorithm that uses the kNN method and works with interaction and CB algorithms that use ontological text attributes similarity. Finally, we demonstrate the Ontological-similarity and Onto-recall metrics that we use in our experiments to better measure the quality of recommendation.

2.1 Diversification methods

2.1.1 Diversity

It is a desirable property in an RS, as users often get bored with recommendation lists that contain items similar to each other [1]. It does not depend on information about the user, but refers to the diversity of items in the recommendation list. We can define it as the opposite of similarity. Diversity may be useful in scenarios where similar items may not be as relevant to the user. An example could be a vacation package in which hotels from the same resort may not be as interesting as offering a list of diverse activities. In addition, we can use diversity to allow the user to explore our catalog of items faster.

2.1.2 Long-tail diversity

In real-life applications, we are often faced with the problem that, in our item catalogue, we could find a small amount of "best seller" and many other items which may sometimes be older or not so popular. This set of items we call "Long-tail items". We should also recommend these items to our users because it should increase diversification and improve the ability to find a new item that the user could be interested in but does not know yet. For example, it could be old classic movies.

Our primary CF algorithms are described in Sect. 2.3. To implement the "Long-tail" diversification mechanism, we add the β parameter, which, as mentioned [4], provides the penalization of the items with the best rating (best-sellers) and add items with a lower rating. This behavior swells our methods, pushes them to Catalogue-coverage maximization, and increases our diversification. The disadvantage is that it rapidly decreases recall.

Table 1 The table of notation where the reader could have at-a-glance explanation of notations used in this article to enhance understanding

Notation	Description
OBCAS	Universal **O**ntological-**B**ased **A**lgorithms for reducing the **C**old-**S**tart problem
RS	Recommendation system
CF	Collaborative-filtering algorithms
CB	Content-based algorithms
KG	Knowledge Graph
BPMN	Business Process Model and Notation
Long-tail	The unpopular set of items in our catalogue
β parameter	Parameter inside CF, that penalize the items that are similar to the user profile
γ parameter	Parameter inside OBACS, that control the ration between amount of items from CF and CB
z-score	A score we use for serendipity models inside OBACS algorithm. It's a combination of Ontological-similarity and the score from CF algorithm

Summary of our approach

1. Prepare the ontology and the knowledge graph.
   (a) We select the text attributes of all items and prepare a single text corpus.
   (b) We run a keyword extraction algorithm to enrich the existing keyword dictionary with new keywords. We use stop words, tokenization, lemmatization, and part-of-speech tagging, and we also use an n-gram combination of keywords.
   (c) For creating a KG, we use two types of relations, implicit and explicit
   (d) To generate implicit relations, we use the interaction between words, such as the number of co-occurrences of two keywords throughout the dataset or the average year of occurrence of a keyword. To generate explicit relations, we use external embedding representations. In our approach, we use ConceptNet, Word2vec, and GloVe frameworks. These relations add dependencies that are typical within the language, and these relations are dataset-independent.
   (e) The final KG contains nodes and edges. The nodes represent keywords from our dictionary. Edges are implicit and explicit relations. The final connection between two nodes is a combination of both connections with weights.
2. Our OBACS algorithm 3.1 uses three phases of recommendation.
   (a) In the first phase, it prepares a top@100 recommendation for the user based on the CF User-kNN algorithm 2.3.
   (b) The second phase uses ontological similarity for the re-ranking item based on the current setting.
   Second-phase settings:
   (i) For Onto-Diversity, we calculate it as an ontological dissimilarity between the items in the recommendation list.
   (ii) For Long-tail diversity, we increase the Long-tail biasing parameter (β) 2.3; it adds penalization to popular items and increases the number of Long-tail items.
   (iii) For serendipity, we find novelty items and unexpectedness, but ontologically close to the relevant items of the user.
   (c) In case we have a new cold item in the catalog, we use the third phase of the algorithm, where we replace several final items with the ontological nearest cold item based on ontological similarity. The ontological similarity between the final and cold items should be greater than the threshold.

2.1.3 Serendipity

It is a concept described in [1]. The main idea is to offer items that might surprise the user. The recommendation should be interesting (*relevant*), new (*novelty*) for the user, and *unexpected*. For instance, recommending a just-released album of a favorite musician may be novel to the user, but it is hardly surprising. Recommending a random item, on the other hand, might be very surprising but not relevant. Serendipity can therefore be defined as representing a delightful surprise for the user.

There are two key differences between diversity and serendipity. First, diversity is based on the comparison of recommended items, while serendipity is based on the comparison of recommended items and user preferences [5]. Second, diversity does not include relevance to a user, which is a necessary component of serendipity.

Our main work deals with diversification, but serendipity is the concept that goes with it. We introduce it in this paper and compare it with our primary diversification method; even the final results are unsatisfactory. We describe this in the discussion section 6

2.2 Ontology and knowledge graph

In this paper, ontology is the formal definition or the schema definition, and the example of KG (knowledge graph) is the framework that contains the movie data instance. Generate an ontology profile for a particular item.

We refer to the study [6], where the authors surveyed several definitions of KG and suggested their own: *"A knowledge graph acquires and integrates information into an ontology and applies a reasoner to derive new knowledge."*

This definition assumes that a knowledge graph contains an ontology (e.g., a knowledge base) and applies a reasoning engine to generate new knowledge while integrating one or

Table 2 Properties that we use in our ontology as implicit metrics

Metric	Description
wCommonOccur	Common occur of the keywords
wSourcePopularity	Popularity of source keyword
wTargetPopularity	Popularity of target keyword
wPopularityRation	Ratio between popularity of target and source keyword
wSourceAvgYear	Average year of items in which source keywords occurred
wTargetAvgYear	Average year of items in which target keywords occurred
wAvgYearRatio	Ratio between average year metrics
wWord2vec	Cosine similarity between two keywords word embedding in word2vec
wGlove	Cosine similarity between two keywords word embedding in GloVe
weight	Final weight

Table 3 Properties that we use as explicit metrics in our ontology

Relation URI	Description	Examples
/r/RelatedTo	The most general relation. There is some positive relationship between A and B	learn ↔ erudition
/r/HasProperty	A has B as a property; A can be described as B	ice → cold
/r/HasContext	A is a word used in the context of B, which could be a topic area, technical field, or regional dialect	retrieval → computer science
/r/IsA	A is a subtype or a specific instance of B; every A is a B	car → vehicle; Chicago → city
/r/Synonym	A and B have very similar meanings	network ↔ net

more information sources. The ontology is a formal specification of a dictionary of concepts. In Table 2, we present implicit relationships or properties that connect two classes, and in Table 3, we present explicit relationships. In addition, we use attributes such as genre or movie director to define movie classes. We integrate external information sources from ConceptNet, word2vec, GloVe, and we are ready to integrate other sources. We apply a reasoning engine that could generate new knowledge from the graph; this knowledge is related keywords, which could extend our item profile if we cannot find a similar item or need to increase exploration of our recommendation. Figure 2 shows the basic profile of the Iron Man movie.

Every time a new item is added to our database, we should recompute the implicit relation inside KG. If the text attributes contain the new keyword, we also update our ontology with explicit information from all extended sources that we present above. So, we always try to add the bulk of the movies and recompute our KG at once. To verify our ontology, we show the ontology profile of the particular item to our users, and they mark the wrong keyword. This information penalizes the weight relationship within KG.

In the next section, we present Ontological-similarity between movies ontology profiles.

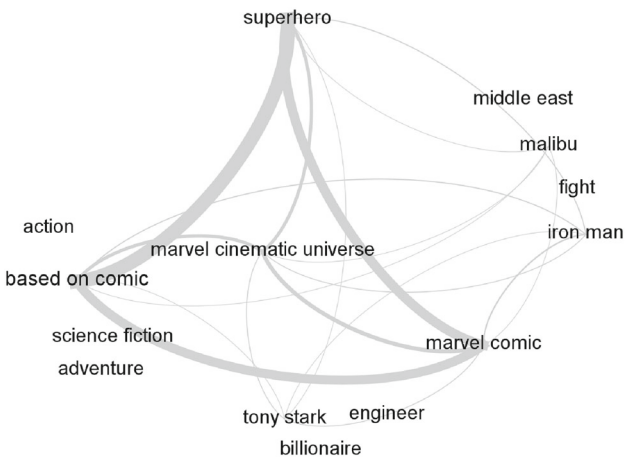

Fig. 2 The basic ontological profile of the Iron Man (2008) movie

2.3 User-kNN algorithm

User-kNN is a class of CF algorithms. We use the user-based k-Nearest Neighbors algorithm with cosine similarity of rows in the rating interaction matrix and voting. In [7], such an algorithm is referred to as *Popularity-Stratified, Non-normalized Cosine Neighborhood*, which we will refer to for

simplicity as *User-kNN* throughout the rest of the paper. To rank items in a user neighborhood, the following formula is used:

$$\text{rank}(u,i) = \frac{\sum_{\hat{u}\in N^k(u)} \text{sim}(u,\hat{u}) \cdot (r_{\hat{u},i} - \overline{r_{\hat{u}}})}{\left(\sum_{u\in\mathcal{U}} (r_{\hat{u},i} - \overline{r_{\hat{u}}})\right)^{\beta}}, \quad (1)$$

where $N^k(u)$ in the set of the k nearest neighbors, $\mathcal{U}$ is the set of all users in the database and β is the Long-tail biasing parameter as proposed in [4]. We introduced this algorithm in our previous research [3].

2.4 Ontological-similarity-based algorithm CB

This algorithm is an algorithm based on Ontological-similarity content. The principle is to use item text attributes as a text corpus, calculate the ontological profile, and use the Ontological-similarity metric.

The final recommendation is calculated on the basis of the cosine distance between the ontological profiles of the items. The schema of the algorithm is shown in Algorithm 2.

For the recommendation of top@N, we use the $itemToitem$ Function 2, where for each item in the user interaction profile, we recommend N most similar items. The final recommendation contains items that we sort by item score, where t (threshold) is a parameter that we can use to control the quality of the recommendation.

Algorithm 1 Preprocessing and generating Ontology and Knowledge graph

Input: I: list of the ∀ items
O: ontology
Output: $M_{i,\mathbf{j}}$: embeddings matrix
O: ontology
KG: Knowledge graph

for each item $i \in I$ **do**
  Get textual attributes
  Remove stop words
  Provide Tokenization
  Provide Part-of-speech tagiing and filter
  Provide Lemmantization
  n-gram combination of the Tags
  Prepare embedding $\mathbf{j_i}$ as bag-of-words
  Find the new ontology candidates $c_n \in O$
end for
Merge all embedding to matrix $M_{i,\mathbf{j}}$ where $i \in I, \mathbf{j} \in \mathbf{E_i}$

for each Candidate $c_n \in O$ **do**
  Compute implicit metrics
  Compute explicit metrics
  Add to KG as node and update edges weights
end for

return $M_{i,\mathbf{j}}$, O,KG

Algorithm 2 Ontological-similarity CB algorithm

Input: $M_{i,\mathbf{j}}$: embeddings matrix
KG: Knowledge graph
Ui: User interactions
t: threshold
Output: $top@N$ recommendation

Update $M_{i,\mathbf{j}}$ with the KG weights
Reduce dimensionality of $M_{i,\mathbf{j}}$
for each $(i_n, i_m) \in M_{i,\mathbf{j}}$ **do**
  Get $\mathbf{j_n}, \mathbf{j_m} \in M_{i,\mathbf{j}}$
  $Similarity(i_n, i_m) = \cos(\mathbf{j_n}, \mathbf{j_m}), i_n \neq i_m$
  Add $Similarity(i_n, i_m)$ to M_{sim}
end for

function RECOMMENDATION(Ui , n)
  for each $i \in Ui$ **do**
    $sim_{i,n} = itemToitem(i, N)$
    add $sim_{i,n}$ to $top@N$
  end for
  return $top@N[0{:}n]$
end function

function ITEMTOITEM($item, n, M_{sim}, t$)
  $sim_{i,n}$,$score_{i,n}$ = similar items $i_n \in M_{sim}$
  for each $i \in sim_{i,n}$ **do**
    if $score_i \leq t$ **then**
      remove i from $sim_{i,n}$
    end if
  end for
  return $sim_{i,n}$
end function

3 Our approach

At the beginning of this section, we will introduce our OBACS (Ontological-Based Algorithms for reducing the Cold-start problem) algorithm, which we use to improve the user experience and reduce the Cold-item problem, which happens pretty often in real-life cases. Next, we will describe in detail how our algorithm uses ontology and diversification methods to reduce the Cold-item problem.

3.1 Universal ontological-based algorithm (OBACS)

OBACS in the original means Universal **O**ntological-**B**ased **A**lgorithms for reducing the **C**old-**S**tart problem. This section describes why CF algorithms do not work in a Cold-start system phase. Next, we explain how we can use these algorithms in a transition period where the system goes from a Cold-start state to a regular state, the regular state being when there are enough users and interaction. Finally, we demonstrate our approach, using ontology to improve diversity in CF recommendation algorithms.

CF models in top@N recommendation tasks are important because, in normal systems, they provide significantly better results in terms of recall than content-based CF algorithms. While the CF algorithms work by attribute similarity

that enters a marketing employee and does not have any connection with user preference, the CF algorithms use the interactions, implicit or explicit, that a particular user has. They can dynamically adapt to current user preferences and are therefore better suited to the role of personalized recommendation.

The disadvantage of the CF approach in a Cold-start environment is the fact that if a given item does not have user interaction, the algorithm cannot recommend it. Also, if most of the users are Cold-start users, cosine similarity on the interaction matrix will always give the same users. However, when the interaction starts to appear in our system, we can then use CF. This period, where the system is still in the cold start, but also has some interaction; we call the "transition Cold-start period". In this period, we should use a hybrid combination of CF and CF. The main goal in this period is to get as many user interactions as possible with regard to cold-items.

One solution could be to generate random interaction for cold items and users. This is not a good idea because we lose relevance. CF compares users by their interactions, and the principle works on the idea that interactions are relevant. When the interactions are random, the CF algorithm offers a random recommendation, which discourages users from using the recommendation.

The second solution is to use CF algorithms throughout the transition period and wait until at least one relevant interaction has been recorded for each item. This is a proper solution, but, as we show in our previous work, CF has a significantly better recall, so users get a more relevant recommendation. Therefore, we should start to use CF as soon as possible.

The third method is to use a hybrid approach, that is, an ensemble of CF with good recall and CF that could recommend similar cold items. In this method, it is essential to define the ratio between the use of CF and CF. If we start using CF algorithms "too soon", the problem may be that the model will recommend the same items, but the later we use CF, the later we will have relevant personalized recommendations. Therefore, we should generate combined recommendations from both systems. In the beginning, we should use more items from CF in the final recommendation, but after a while the ratio should change to the CF side.

We use the third method. As a CF algorithm, we use the basic similarity-based algorithms with Ontological-similarity inside. As a CF algorithm, we use the User-kNN algorithms 2.3. The experiment in this area is presented in the next chapter 5

Furthermore, we use the hybrid approach in those cases where we have a new item in our system. Since we assume that the new item has the same attributes as other items in our system, we can then automatically generate an ontological profile and include this item with similar relevant items.

3.2 Our metrics

3.2.1 Ontological-similarity metric

Our idea is to use the item's ontology profile 2 to compute the similarity between the items. The metric should contain information about how the two items are similar or how these two items are semantically close. For this purpose, we define an Ontological-similarity metric. This metric presents the cosine similarity between two ontology profiles of particular items. Ontology profiles present embeddings of size equal to the number of keywords in our ontology, where each keyword has its position. For each keyword in item profiles from our ontology, we place one in particular positions; other positions are filled with zeros. We use Ontological-similarity metrics inside our CB algorithm and compare them with other metrics presented in this paper.

The formula presents an Ontological-similarity metric:

$$\text{OntoSim}(\overrightarrow{I_n})^{\text{COS}} = \frac{1}{n} \cdot \sum_{i,j=1,i\neq j}^{n} \frac{\overrightarrow{I_i} \cdot \overrightarrow{I_j}}{\| \overrightarrow{I_i} \| \| \overrightarrow{I_j} \|} \tag{2}$$

where $\overrightarrow{I_n}$ is the ontological profile of items within a single recommendation in the top@N recommendation task.

3.2.2 Onto-recall metric

The recall is the most common metric that we use in our offline experiment. The problem is that it does not show complete information about the actual diversification within the recommendation because it only shows the hits of our algorithms. Onto-recall metrics show how our algorithm can hit ontologically diversified items. We modified the recall metric using a filter that tracks ontological similarity within a single recommendation and counts as a hit only when the ontological similarity within the recommendation is higher than the average ontological similarity of our catalogue.

Equations 3 and 4 describe the Onto-recall metrics.

$$\text{OntoTP} = \begin{cases} 1, & \text{ifOntoSim}(\overrightarrow{I_n}) \geq \text{OntoSim}(\overrightarrow{I_{\text{all}}}) \\ 0, & \text{otherwise.} \end{cases} \tag{3}$$

Where $\overrightarrow{I_n}$ is the ontological profile of the items inside a single recommendation in the top@N recommendation task and $\text{OntoSim}(\overrightarrow{I_{\text{all}}})$ is the average ontological similarity between all ontological profiles of our items inside our catalog.

$$\text{OntoRecall} = \frac{\text{OntoTP}}{\text{OntoTP} + \text{FP}} \tag{4}$$

where OntoTP is the total number of hits, defined by eq. 3 and FP are all other recommendations. In our experimental data

sets 5, the average ontological diversity is 5%. So, as the hit in the calculation of Onto-recall, we count only recommendations that contain the control item, and their ontological diversity is greater than the average in the dataset. We use Onto-recall and Catalogue-coverage in all the experiments in this paper.

3.3 Diversification methods inside OBACS

In Sect. 2.1.1, we described the concept of diversity and serendipity. Both concepts are useful in top@N recommendations because with them we can cover as many relevant items as possible, and the user can still explore new items, both diverse and unexpected. [8].

3.3.1 Diversity inside OBACS

In Sect. 2.1.1, we describe two types of diversification methods: diversity 2.1.1 and Long-tail diversity 2.1.2.

Our approach is to use an ensemble of a CF algorithm, in our case User-kNN 2.3 as a primary model and an ontological similarity CB algorithm 2.4 as a secondary model. The primary model will provide the first, relevant recommendation, while a secondary algorithm will re-rank this recommendation. The re-ranking method works on the principle that, for each next item in the recommendation, it finds the most dissimilar item according to ontological similarity. Our diversity provides a relevant recommendation because of the primary algorithm and novelty and because we recommend unseen items to the user. In each stage of the Cold-start system, the diversity requirements could change, so we present the parameter γ for manipulation of the composition of the final recommendation.

3.3.2 Serendipity inside OBACS

The concept of serendipity, as described in Sect. 2.1.1, should provide three main properties. *Relevance* the item should be relevant to a user. *Unexpectedness* the item should be a surprise to the user. *Novelty* the item should be new. The behavior of this concept, compared to diversity, provides a recommendation that slightly decreases recall, but increases Catalogue-coverage. In the experiment below, we will discuss this behavior. We found that this type of diversity increases Catalog-coverage and keeps recall at high levels.

Our approach to serendipity uses the same method as for diversity, as described above, but with a slight difference. Serendipity should, instead of diversity, provide unexpectedness. Unexpectedness, in our case, means the outlier item or the item that is most ontologically distant from the user profile. To compute this, we should first calculate the ontological profile of our user. We can do this by a combination of the ontological profile of the interacted item and by the merging of all nodes that the historical items have into a single profile. After we calculate the profile and obtain the recommendation from the primary algorithm, we can compute the Ontological-similarity between the user profile and the item in the recommendation. After we get the ontological score, we can calculate the *z-score*. The final recommendation is sorted by *z-score*.

4 Improvements over state-of-the-art

The paper [9] proposed the Tagging Product Review (TPR) system that uses the unsupervised method of generating tags and ranking them according to relevance. They use user reviews of the product and create a cloud of tags. They define a cold product as a product with an average overall rating of one or two and with fewer reviews. The top 50 candidate tags contain aspect information from each popular product belonging to a similar domain and use it to form a tag cloud. They leverage popular product reviews with a similar domain to find proper tags from cold product reviews.

Our approach uses metadata text attributes, and it does not depend on user reviews at all. Our goal is to build a semantic layer based on ontology and KG for a more in-depth understanding of the dependencies inside our datasets. So, we try to understand the essence of a "product", not the user's opinion. That is why we use implicit and explicit information about each keyword inside our KG. Also, the premise that the same product in the same domain could have similar tags does not work in the case of Movies or Research Experts. For example, there are hundreds of thousands of comedian movies with absolutely different scenarios, or there is a vast difference between researchers, even though they belong to similar research fields.

This article [10] authors presents the idea that if two items were tagged with similar tags, the two items are similar. By taking the semantic distance between tags and items into consideration, they propose a collaborative filtering approach to CFBTSS (Collaborative Filtering Based on Tag Semantic Similarity) to improve the tagging system. They assume that incorporating the semantic information of the tags associated with the shared resources into collaborative filtering can significantly improve the predictions of a recommender system. Our approach is furthermore similar because our similarity also contains information on the semantic similarity of the keywords. The items can be similar even if they have "textually" different keywords but the same semantically. Also, in our work, we improve the CF approach in a Cold-start environment, which is very often in real-life applications, while the authors skip this problem.

The authors [11] describe the fuzzy logic-based product recommendation system for customers in online shopping. They use product rating scores based on review comments

and demographic age categories. Fuzzy rules are applied to predict highly relevant recommendation products by performing sentiment analysis and ontology alignment in the decision-making process. The authors mention the importance of using diversification methods because user preferences change over time. Our solution is different because we can recommend items without the need for user interaction. The ontological profiles, which we get from text attributes, are essential. These profiles determine the semantic similarity of two items, which will always be the same, and users' opinions have no influence on it.

The paper [12] presents an approach that integrates items genre data in item-based CF and reduces the Cold-start item problem. The main contribution is to generate item similarities. First is the asymmetric interconnectivity based on the item's genre. The second is based on relative interconnectivity. Our approach is to use Ontological-Similarity with a combination of CF algorithms. The benefit of our approach is that we can dynamically change the number of the keywords included in the item ontology profile by exploring similar semantic keywords. Another benefit is a self-explanation of the item ontology profile.

The authors [13] proposed an approach that is a hybrid recommender system that combines collaborative filtering, content-based approaches, and ontologies. A combination of both the CB and CF approach limits the shortcomings of Cold-start problem data sparsity. Furthermore, the use of an ontology reinforces the accuracy in computing the semantic similarity measures on vehicle descriptions, user profiles, and contextual information. Our approach is to use the ontology inside the CB algorithm with a combination of the CF algorithm to utilize the Cold-start problem. In addition, we define different phases of the system and combine these approaches with respect to the actual phase in different ways. We use the Onto-Recall metric to measure ontological (semantic) similarity inside each recommendation and use it to prepare diversification recommendations semantically.

This article [14] describes an ontology model that can be used to address the pure Cold-start problem in content recommenders in a PLE (Personalized Learning Environment). The developed ontology model conceptualizes both learner and learning object characteristics that are found to be relevant for the content recommendation in an adaptive learning environment. A multivariate version of the k-means clustering algorithm has been used for data evaluation purposes. We build our ontology and knowledge graph from textual attributes using implicit and explicit information [3]. We use KG to extend the ontological item profile to increase the probability of finding similar items. In the Cold-start phase, we use the CB algorithm primarily with ontological similarity inside.

5 Offline experiments

We prepare several offline experiments to verify our solution and show the benefits of our OBACS algorithms in all three stages of development. There are several ways to simulate user behavior in a system [15]. The most common method is to use historical data, with some user interactions hidden, and then, use them to predict their value. Ideally, a timestamp interaction is used, based on which we can simulate the system behavior over time. The most commonly used procedure for large datasets is one in which we randomly select test users, randomly choose a time for these users, hide all the interactions of these users from this time, and try to recommend items to those users. The problem with this approach is that it needs to perform the entire process before each recommendation, which is not very efficient in terms of performance.

5.1 Experimental setup

Since our RS always recommends top@N items (N = 5) for each user, a leave-one-out cross-validation methodology [16] seems like a good compromise to test the RS. Within this method, cross-validation is carried out across all users in the system.

For model comparisons, we use Onto-recall 3.2.2 and Catalogue-coverage[1] metrics. We chose these metrics because users prefer more diversified recommendations with greater coverage. Therefore, we optimize both metrics. The exact description of the methodology and a discussion of the selection of these metrics can be found here [16].

5.2 Experimental dataset

For our experiments, we selected the Full MovieLens 20 M [17] [2] dataset and its extension *The Movie Dataset*.[3] MovieLens is a very well-known dataset that is used as a benchmark for most academic articles concerning recommendation systems, and is primarily designed for offline testing. This dataset comes from the GroupLens Research Laboratory of Computer Science and Engineering at the University of Minnesota.

5.2.1 The movie dataset

We extends the MovieLens dataset with additional metadata extracted from the TMDB[4] database. For each movie, Movie-

[1] This metric shows whether a system can recommend diversity, i.e., if it can offer items other than best sellers.

[2] https://grouplens.org/datasets/movielens/

[3] https://www.kaggle.com/rounakbanik/the-movies-dataset.

[4] https://www.themoviedb.org/

Table 4 List of datasets, files and metadata used in our experiments

Dataset	File	Metadata
MovieLens 20 M	tags.csv	film tags
MovieLens 20 M	movies.csv	film genres
The Movie Dataset	movies_metadata.csv	original title
		overview
The movie Dataset	keywords.csv	keywords

Table 5 Cold-item datasets with statistics of total users interactions, total users count, average interaction per user and the total number of movies

Dataset	Total inter.	User count	Aver. user inter.	Movies count
2–3	12975	5252	2.5	2735
3–5	46054	10763	4	4809
5–10	194911	26784	7	7637
10–20	1063130	70170	15	10686
20–50	1879318	58864	32	12173

Lens collects additional text data such as title, overview, keywords, production company, year, and director. This extended information is essential as it gives us a more significant amount of text data to build an ontology. For a complete description of the attributes and the description of the file, see Table 4. The average Ontological-similarity 3.2.1 for all datasets is around 5–6%.

In our experiment, we would like to show the behavior of the OBACS algorithm in all stages of the system development to confirm its universality. For this purpose, we prepared five datasets with different amounts of interactions per user to simulate all phases and where all users have the same amount of interactions. MovieLens contains more than 20 M interactions and is not homogenous; before splitting the data, we removed items with less than 50 interactions as noise. We present our new datasets in Table 5.

5.3 Leave-one-out cross-validation methodology

We constructed an interaction matrix for all users of the RS and all items, where each row is a user and each column an item. A single cell of the matrix contains the number of interactions between an item and a user. If no interaction has taken place, the cell is filled out with zero. During each cross-validation step, we assign 90% of the users to a training set and the remaining 10% to a test set. For each user, we always choose Top@5 items (inputs). The following procedure is divided into steps:

1. We train a model on training data for each cross-validation step.
2. We hide user interaction from the system and let the Top 5 recommendations be generated for each user from the test set and each item (matrix cell).
3. If the current item is between the recommended items while finding that the value in a cell is greater than zero[5] and the average ontological similarity between items is greater than 5% (the average value in my datasets), the Onto-recall value of a single test is one. Otherwise, it is zero.
4. After going through all matrix cells, we calculate the total recall of the cross-validation step by dividing the number of successful tests by the number of all tests (all non-empty matrix cells). Total coverage is the number of unique items that our model recommends during the experiment.
5. Then, we move on to the next step of cross-validation, i.e., to point 1.
6. After completing cross-validation, we can count the average recall, the average on-to-recall, and the Catalogue-coverage across all tested users.

5.4 OBACS experiments

In our experiments, we use five cold-user stages datasets presented in Table 4, where each dataset represents different phases of system expansion, from a nearly Cold-start system to a regular stage. As a primary CF algorithm, we select User-kNN 2.3. Then, we choose the Ontology-similarity and re-ranking method suggested in [1].

5.4.1 Results

Figure 3 shows the results of the experiment on the Cold-item dataset. The graph shows the results of all diversification methods presented in this paper compared to the random algorithm and the pure OBACS algorithms without diversity. The X-axis shows Onto-recall, and the Y-axis shows Catalogue-coverage.

Figure 4 shows the results of the experiment on three datasets, with an increasing number of interactions. These datasets represent a transition state, where we reduce the Cold-item problem, but we do not yet have a significant number of interactions to get CF algorithms working correctly.

Figure 5 shows the last experiment on a dataset with many interactions. This dataset represents a system without a Cold-item problem and has enough interactions to properly utilize the CF algorithms.

[5] If we have a system with a large number of interactions, we can set a certain threshold, e.g., greater than 5.

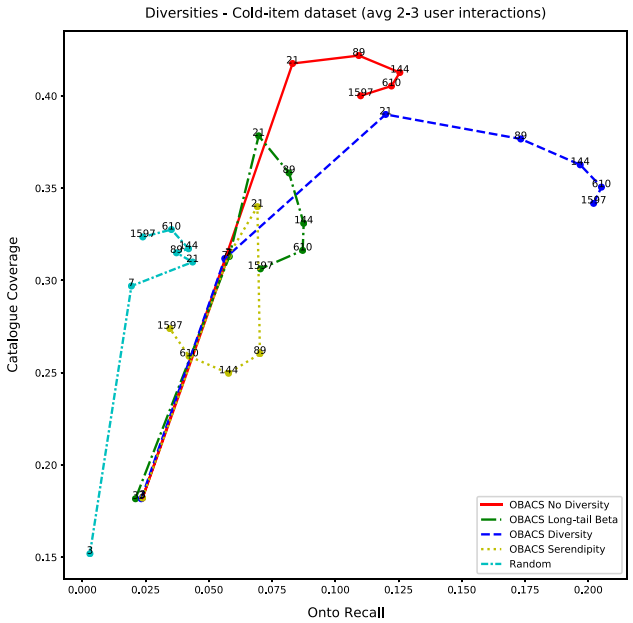

Fig. 3 The Figure shows the results of the Cold-item dataset, where the average number of interactions is 2–3 for each item. We can see that the best model, the Pareto dominant, in the case of Catalogue-coverage is the OBACS without diversity. The best model, the Pareto dominant, in the case of Onto-recall, is OBACS with Diversity. The OBACS with Diversity is the better choice because it has significantly better Onto-recall and still has a good Catalogue-coverage. The basis of this model is the hybrid model, which uses the primary Ontology-based CB model and then, its use diversity re-ranking method that we describe in section 3.3.1. Other diversification methods have a worse result

In the first experiment, where users have a small number of interactions, we can observe that OBCAS with diversity is Pareto dominant.

In the second experiment, we can observe that the Catalogue-coverage of regular algorithms decreases and, for the same hyperparameter k (User-kNN), the diversification approach starts to work. All three datasets show the same characteristic; the regular dataset shifts to the recall part, while Catalogue-coverage decreases, but the diversification approach also increases Catalogue-coverage.

Now, we get to the last experiment, where the average number of interactions is high and shows the behavior of each diversity measure. In all datasets, the best model is OBACS with diversity; it is Pareto dominant with the highest Onto-recall and sufficient Catalogue-coverage. All results are presented in Table 6.

5.4.2 Onto-diversity

Increases the Catalogue-coverage compared to serendipity but also decreases Onto-recall, while the number of items (parameter k) increases. Compared to the OBACS without diversity, the Onto-recall is higher. This is because, in the diversity re-ranking method, we compare items to each other, and we do not care about the similarity with the user profile, while the CF algorithm beside OBACS uses items from users' interactions history, which are more ontological similar.

5.4.3 Serendipity

As we mentioned in Sect. 3.3, it decreases Catalogue-coverage and has, compared to other diversification methods, the worst Onto-recall. It happens because we move the "outlier" or the "unexpectedness" item to a higher position in the recommendation. We also do this based on the user profile, so we still try to be relevant to the user. This means that we decrease Onto-recall twice. The first time, because we decrease original recall, and the second time, we try to add some relevant items to the user profile. Unfortunately, only properly using CF algorithms is better for users. That is why this diversity in the case of Onto-recall has a worse performance. Improvement of this diversity is our future work.

5.4.4 Long-tail diversity

The parameter β uses for the penalization of the items that are similar to the user profile and k nearest neighbor and adds items that are popular in the whole catalogue. This behavior moves the method to Catalogue-coverage maximization. The disadvantage is that it rapidly decreases Onto-recall. As we see, the total performance of the model is low, so this diversity has poor performance in the case of Onto-recall.

6 Discussion

This experiment proves our hypothesis that ontology-based models and Ontological-similarity can improve the characteristics of CF in the cases of diversity and Onto-recall. This experiment aims to show the behavior of particular diversity methods with Ontological-similarity in a different phase of system expansion.

Each method could assist in a different case. Because all users have different preferences, we can use different approaches. For example, if a user is looking for relevant movies, we can keep the algorithm as is. If the user is looking for relevant, but maybe older, items or movies in our case, or does not find an appropriate movie for a long time, we can use serendipity, which could recommend a film high in unexpectedness. If the user wants to explore the catalog, we can use diversity, or we can turn on our Long-tail diversity.

We are also able to switch algorithms dynamically as needed. The optimal switching strategy will be determined in future work as many experiments in an online environment are needed to prove our ideas.

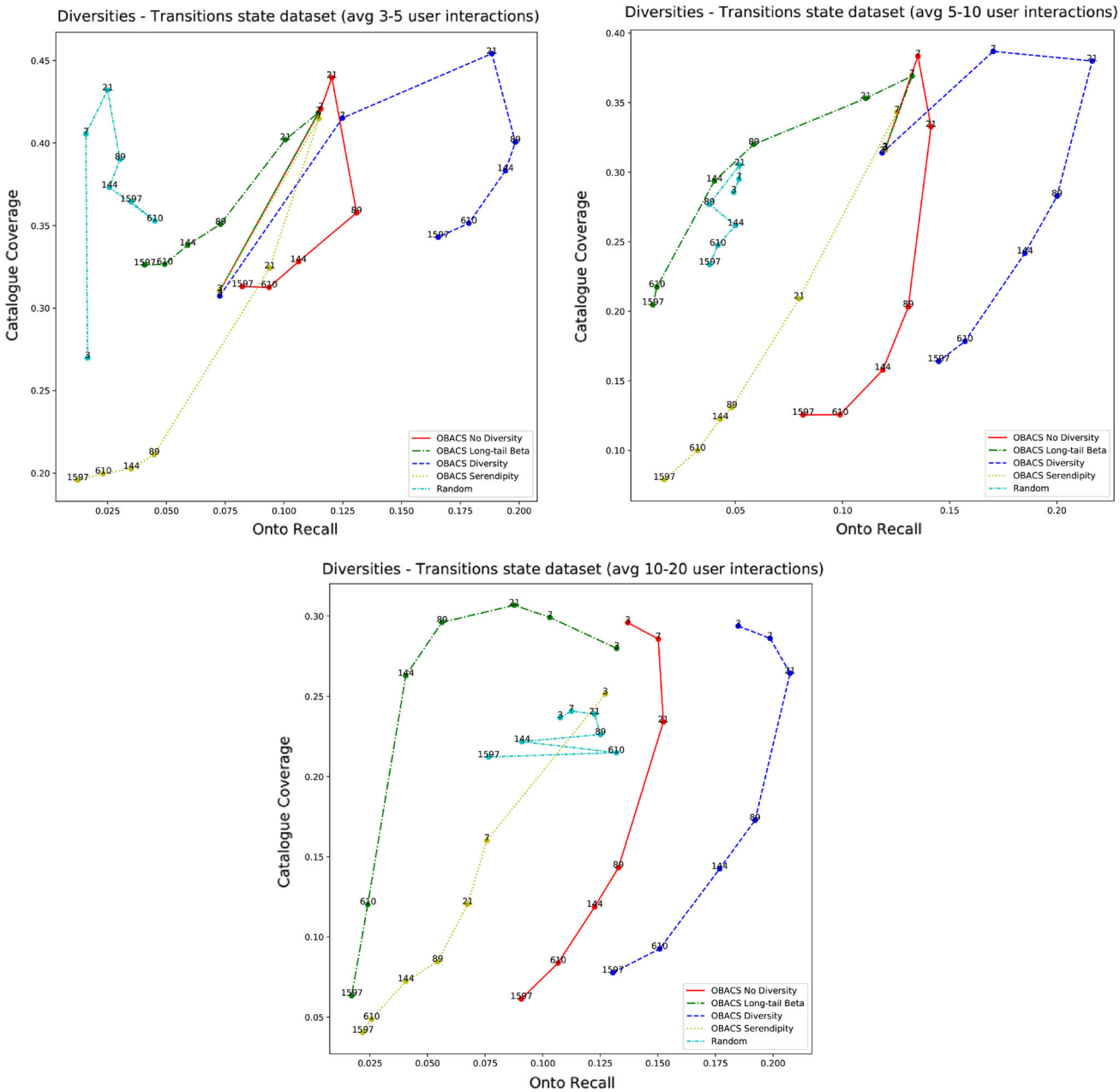

Fig. 4 The Figure shows the model's results in the transitive state, where the average number of interactions for users increases. For the basic OBACS model, we see an increase in Onto-recall and a significant decrease, in Catalogue-coverage. This phenomenon clearly shows how the models are starting to recommend more accurately but with fewer items. If we have ten user interactions on average in our dataset, we should start using diversification methods; otherwise, our dataset will be unbalanced. In the case of diversity, the best model is the OBACS with Diversity; other models have worse performance

7 Conclusion

In this paper, we present our OBACS algorithm, which uses the Ontology-based approach in combination with the CF algorithm, thus creating a universal hybrid system that can be used in all stages of the system. From a system with a highly sparse interaction matrix with the Cold-item problem. Transit state, where there are still many cold items and the CF algorithm performs poorly and does not allow users to discover items that do not have enough interactions. The regular state, where there are many interactions and the CF algorithms have good performance, but recommendations suffer from low diversity of the items inside a single recommendation.

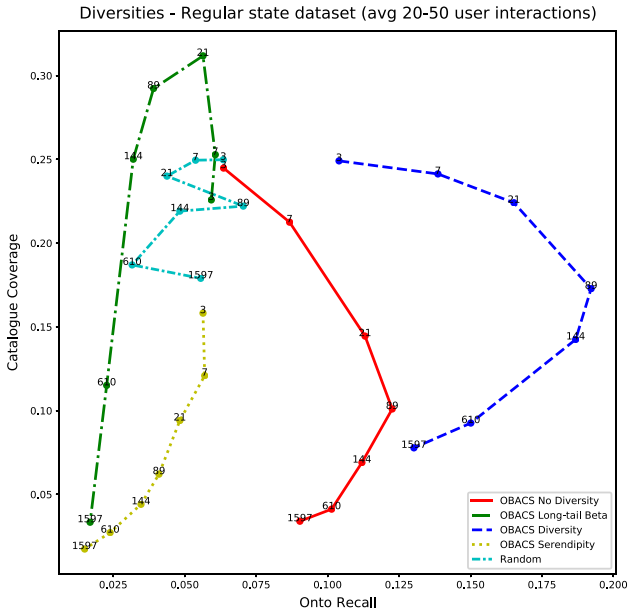

Fig. 5 The last Figure shows the use of all diversification methods on the regular state dataset, i.e., dataset where all users have a relatively large number of interactions to make the CF algorithm work well. If we want to recommend real ontological diversification items, we should use OBACS diversity. It has the best Onto-recall with no small Catalogue-coverage. Also, we see that OBACS without diversity that primarily uses CF User-kNN algorithm recommend ontologically close items. If we want to surprise the user, we turn on serendipity; if we want to re-balance our dataset and get more interaction, we use the Long-tail β parameter. The final solution always depends on a specific business request

Another advantage of our approach is creating an automatic ontology profile for new items with textual attributes. Our approach will work even if the text attributes of items are grammatically different from the item in our catalog but semantically have the same meaning.

Our future work is to improve the ontological model. We want to start integrating the current, trended information about user search and behavior, and thus, dynamically changing the relationships between entities inside our knowledge graph. It could bring a better adaptation of the systems and, as a result, better performance of the CB part of our approach.

We are also about to perform the online evaluation of our algorithms because the offline evaluation is biased towards algorithms similar to those used to obtain training data.

The OBACS algorithm is actively used in the real-life application experts.ai[6] to recommend academic experts to collaborate with businesses.

[6] https://find.experts.ai.

Table 6 There are the results of our offline experiments

Dataset	Model	Onto recall	Catalogue coverage	k
2–3	No Diversity	0.13	**0.41**	144
2–3	Long-tail β	0.09	0.33	144
2–3	Diversity	**0.21**	0.35	610
2–3	Serendipity	0.07	0.26	89
2–3	Random	0.04	0.31	21
3–5	No Diversity	0.13	0.36	89
3–5	Long-tail β	0.11	**0.42**	7
3–5	Diversity	**0.20**	0.40	89
3–5	Serendipity	0.11	0.41	7
3–5	Random	0.05	0.35	610
5–10	No Diversity	0.14	**0.38**	21
5–10	Long-tail β	0.13	0.37	7
5–10	Diversity	**0.22**	**0.38**	21
5–10	Serendipity	0.13	0.34	7
5–10	Random	0.05	0.31	21
10–20	No Diversity	0.15	0.23	21
10–20	Long-tail β	0.13	**0.28**	3
10–20	Diversity	**0.21**	0.26	21
10–20	Serendipity	0.13	0.25	3
10–20	Random	0.13	0.21	610
20–50	No Diversity	0.12	0.10	89
20–50	Long-tail β	0.06	**0.25**	7
20–50	Diversity	**0.19**	0.17	89
20–50	Serendipity	0.06	0.12	7
20–50	Random	0.07	0.22	89

All results, we round to 2 digital numbers. Onto-recall is the metrics described 3.2.2, Catalogue-coverage is the percentage of all items inside our catalogue that we have covered within our recommendations, and k is the number of neighbors in our version of CF algorithms. 2.3

Acknowledgements This research has been supported by the Grant Agency of the Czech Technical University in Prague (SGS20/213/OHK3/3T/18). The Computational resources were partially supplied by the project "e-Infrastruktura CZ" (e-INFRA LM2018140) provided within the Projects of Large Research, Development and Innovations Infrastructures programme. I would like to express thanks to my colleagues from ML-CIG group and DataLab, FIT, CTU, for their valuable comments and proofreading.

References

1. Kotkov, D., Wang, S., Veijalainen, J.: A survey of serendipity in recommender systems. Knowl.-Based Syst. **111**, 180–192 (2016). https://doi.org/10.1016/j.knosys.2016.08.014
2. Pourgholamali, Fatemeh.: Mining information for the cold-item problem. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, page 451-454,(2016). ISBN 9781450340359. https://doi.org/10.1145/2959100.2959102
3. Kuznetsov, S., Kordík, P., vRehořek, T., Dvořák, J., Kroha, P.: Reducing cold start problems in educational recommender systems. In *2016 International Joint Conference on Neural Networks (IJCNN)*, pages 3143–3149, (2016) https://doi.org/10.1109/IJCNN.2016.7727600
4. Steck, H.: Item popularity and recommendation accuracy. In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, page 125–132, New York, NY, USA. Association for Computing Machinery. ISBN 9781450306836. (2011) https://doi.org/10.1145/2043932.2043957
5. Iaquinta, Leo., de Gemmis, Marco., Lops, Pasquale., Semeraro, Giovanni., Molino, Piero.: Can a recommender system induce serendipitous encounters? In Kyeong Kang, editor, *E-commerce*, chapter 13. IntechOpen, Rijeka, (2010). https://doi.org/10.5772/8905
6. Ehrlinger, Li., Wöß, W.: Towards a definition of knowledge graphs. *SEMANTiCS (Posters, Demos, SuCCESS)*, (2016). URL http://ceur-ws.org/Vol-1695/paper4.pdf
7. Cremonesi, P., Koren, Y., Turrin, R.: Performance of recommender algorithms on top-n recommendation tasks. In *Proceedings of the Fourth ACM Conference on Recommender Systems*, RecSys '10, pages 39–46, New York, NY, USA. ACM. ISBN 978-1-60558-906-0. (2010) https://doi.org/10.1145/1864708.1864721
8. Nguyen, T.T., Hui, P.-M., Harper, F.M., Terveen, L., Konstan, J.A.: Exploring the filter bubble: The effect of using recommender systems on content diversity. In *Proceedings of the 23rd International Conference on World Wide Web*, (2014). ISBN 9781450327442. https://doi.org/10.1145/2566486.2568012
9. Konjengbam, A., Kumar, N., Singh, M.: Unsupervised tag recommendation for popular and cold products. J. Intell. Inf. Syst. (2020). https://doi.org/10.1007/s10844-019-00574-9
10. Hang, Chen., Meifang, Zhang. Improve tagging recommender system based on tags semantic similarity. In *2011 IEEE 3rd International Conference on Communication Software and Networks*, pages 94–98, (2011). https://doi.org/10.1109/ICCSN.2011.6013670
11. Karthik, R.V., Ganapathy, S.: A fuzzy recommendation system for predicting the customers interests using sentiment analysis and ontology in e-commerce. Appl. Soft Comput. **108**, 107396 (2021). https://doi.org/10.1016/j.asoc.2021.107396
12. Hasan, M., Roy, F.: An item-item collaborative filtering recommender system using trust and genre to address the cold-start problem. Big Data Cogn. Comput. (2019). https://doi.org/10.3390/bdcc3030039
13. Le, Ngoc Luyen., Abel, Marie-Hélène., Gouspillou, Philippe.: Towards an ontology-based recommender system for the vehicle sales area. In *Progresses in Artificial Intelligence & Robotics: Algorithms & Applications*, pages 126–136, (2022). ISBN 978-3-030-98531-8. https://doi.org/10.1007/978-3-030-98531-8_13
14. Joy, J., Raj, N.S., VG, R.: Ontology-based e-learning content recommender system for addressing the pure cold-start problem. J. Data Inf. Qual. (2021). https://doi.org/10.1145/3429251
15. Shani, G., Gunawardana, A.: Evaluating Recommendation Systems, 257–297. (2011). ISBN 978-0-387-85820-3. https://doi.org/10.1007/978-0-387-85820-3_8
16. Campochiaro, E., Casatta, R., Cremonesi, P., Turrin, R.: Do metrics make recommender algorithms? In: *Advanced Information Networking and Applications Workshops, 2009. WAINA '09. International Conference on*, pages 648–653, May (2009). https://doi.org/10.1109/WAINA.2009.127
17. Harper, F.M., Konstan, J.A.: The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.* (2015). https://doi.org/10.1145/2827872

Terms and Conditions

Springer Nature journal content, brought to you courtesy of Springer Nature Customer Service Center GmbH (“Springer Nature”).
Springer Nature supports a reasonable amount of sharing of research papers by authors, subscribers and authorised users (“Users”), for small-scale personal, non-commercial use provided that all copyright, trade and service marks and other proprietary notices are maintained. By accessing, sharing, receiving or otherwise using the Springer Nature journal content you agree to these terms of use (“Terms”). For these purposes, Springer Nature considers academic use (by researchers and students) to be non-commercial.
These Terms are supplementary and will apply in addition to any applicable website terms and conditions, a relevant site licence or a personal subscription. These Terms will prevail over any conflict or ambiguity with regards to the relevant terms, a site licence or a personal subscription (to the extent of the conflict or ambiguity only). For Creative Commons-licensed articles, the terms of the Creative Commons license used will apply.
We collect and use personal data to provide access to the Springer Nature journal content. We may also use these personal data internally within ResearchGate and Springer Nature and as agreed share it, in an anonymised way, for purposes of tracking, analysis and reporting. We will not otherwise disclose your personal data outside the ResearchGate or the Springer Nature group of companies unless we have your permission as detailed in the Privacy Policy.
While Users may use the Springer Nature journal content for small scale, personal non-commercial use, it is important to note that Users may not:

1. use such content for the purpose of providing other users with access on a regular or large scale basis or as a means to circumvent access control;
2. use such content where to do so would be considered a criminal or statutory offence in any jurisdiction, or gives rise to civil liability, or is otherwise unlawful;
3. falsely or misleadingly imply or suggest endorsement, approval , sponsorship, or association unless explicitly agreed to by Springer Nature in writing;
4. use bots or other automated methods to access the content or redirect messages
5. override any security feature or exclusionary protocol; or
6. share the content in order to create substitute for Springer Nature products or services or a systematic database of Springer Nature journal content.

In line with the restriction against commercial use, Springer Nature does not permit the creation of a product or service that creates revenue, royalties, rent or income from our content or its inclusion as part of a paid for service or for other commercial gain. Springer Nature journal content cannot be used for inter-library loans and librarians may not upload Springer Nature journal content on a large scale into their, or any other, institutional repository.
These terms of use are reviewed regularly and may be amended at any time. Springer Nature is not obligated to publish any information or content on this website and may remove it or features or functionality at our sole discretion, at any time with or without notice. Springer Nature may revoke this licence to you at any time and remove access to any copies of the Springer Nature journal content which have been saved.
To the fullest extent permitted by law, Springer Nature makes no warranties, representations or guarantees to Users, either express or implied with respect to the Springer nature journal content and all parties disclaim and waive any implied warranties or warranties imposed by law, including merchantability or fitness for any particular purpose.
Please note that these rights do not automatically extend to content, data or other material published by Springer Nature that may be licensed from third parties.
If you would like to use or distribute our Springer Nature journal content to a wider audience or on a regular basis or in any other manner not expressly permitted by these Terms, please contact Springer Nature at

onlineservice@springernature.com