CRecSys: A Context-Based Recommender System Using Collaborative Filtering and LOD

VINEET K. SEJWAL1, MUHAMMAD ABULAISH2, (Senior Member, IEEE), AND JAHIRUDDIN1

1Department of Computer Science, Jamia Millia Islamia (A Central University), New Delhi 110025, India
2Department of Computer Science, South Asian University, New Delhi 110021, India
Corresponding author: Jahiruddin (jahir.jmi@gmail.com)

ABSTRACT Linked Open Data (LOD) is an emerging Web technology to store and publish structured data in the form of interlinked knowledgebases like DBpedia, Freebase, Wikidata, and Yago. It uses structured data from multiple domains, and it can be used to conceptualize a concept of interest. Recently, researchers have shown that incorporating contextual features in recommender systems improves rating prediction accuracy. However, identification of contextual features for building context-aware recommender systems is a major bottleneck. To this end, in this article, we present the development of a context-based recommender system, CRecSys, for item ratings prediction in movie domain. CRecSys extracts item-based contextual features from the underlying dataset and generates an RDF graph to model items and their contextual features for computing context-based items similarity using graph matching techniques and item-based collaborative filtering. It uses LOD and two well-known movie data sources – Rotten Tomatoes and IMDB for item profiling using a dataset of 1300 movies. CRecSys is experimentally evaluated over two movie datasets, one is generated by the authors and second is the MovieLens-1M benchmark dataset. CRecSys is also compared with ten baselines and two state-of-the-art recommendation methods, and performs significantly better. It is also empirically established that CRecSys is able to effectively deal with some of the open challenges like cold-start and limited content problems of the traditional recommender systems.

INDEX TERMS Recommender system, collaborative filtering, context-based recommendation, LOD, contextual similarity, RDF graph.

I. INTRODUCTION

The open nature of the Web 2.0 has resulted in an uncontrolled generation of a plethora of information leading to the information overload problem. E-commerce is one of the exponentially growing web-based services which is adding hundreds of thousands of products and services, and attracting a large number of buyers and sellers on daily basis. As a result, e-commerce platforms are also facing the information overload problem. To deal with the information overload problem in e-commerce, academia, and industry, researchers have proposed various recommendation techniques to filter out irrelevant items and recommend only those items to users that are relevant to their requirements, interests, and profiles [1]. Most of the world’s large corporations are successfully using one or another form of recommender system technologies to facilitate their customers. Among them notable are Netflix’s movie recommender system, Amazon’s product recommender system, and Last.fm’s songs recommender system. Although researchers have proposed several recommender systems for different domains, and organizations are successfully using their customized recommender systems, there are several challenges like cold-start, black box recommendation, limited content, and data sparsity problems that lower down the efficacy of the recommender systems. To this end, many researchers have considered the development of context-aware recommender system as a possible solution, which incorporates contextual features for recommendation. However, most of the existing approaches have used only user-decision contextual features, ignoring the item-based contextual features.
A. WHY CONTEXT IN RECOMMENDER SYSTEMS?

A recommender system predicts ratings and recommends items based on users’ interest, browsing history, and preferences. In the recommendation process, recommender systems select user-centric relevant items using filtering algorithms like collaborative filtering and content-based filtering. The filtering algorithms do not consider the services and conditional usage of items, where conditional usage represents different conditions from a user’s perceptive to consume an item, such as what products are used by which users and when. For example, a user may prefer to watch different movies at different place (home or theater) with different companion (family or spouse) on different time (weekend or weekday). To satisfy the constraint of the conditional usage, contextual information needs to be incorporated in traditional recommender systems for improving their recommendation and rating prediction accuracy. Abowd et al. [1] defined context as follows: “context is any information that can be used to characterize the situation of an entity such as person, place, or object which is relevant in the interaction between the entity and an application, including the user and application themselves”. On the other hand, Cantador and Castells [2] defined context as “the background topics under which activities of a user occur within a given unit of time”.

B. LINKED OPEN DATA

Linked Open Data (LOD) is a collection of multiple knowledgebases, such as DBpedia, Freebase, and Yago that are interlinked with each other and contain structured data related to different domains. It is developed using standard web technologies like HTTP, URI, and Resource Description Framework (RDF), where URI is a unique identifier to represent entities and RDF is a data model to represent data in a machine understandable triplet form (subject, predicate, object). The nucleus of LOD is DBpedia, which is connected to every other knowledgebase of the LOD. The linked datasets are used in many applications, including the development of item-based context-aware recommender systems. Catherine and Cohen [11] and Noia [12] discussed the procedure of using knowledge graphs in recommender systems, and emphasized that the integration of knowledge graphs in recommender systems can address various challenges and issues like data sparsity, limited-content, and cold-start problems. Data sparsity is a major issue in recommender systems which arises due to unavailability of sufficient ratings for the items. Lacking sufficient ratings on data items degrades the performance of the recommender systems because it is difficult to find most similar items in the system [47]. Similarly, limited content problem arises due to unavailability of sufficient contents for the items. This problem further leads to the over-specialization issue in which recommender systems are not able to recommend novel items [47]. Finally, cold-start problem can occur in recommender systems for both users and items, in the cases where either some users rated very few items or some items received very few ratings. One of the major drawbacks with cold-start problem is that it degrades the overall performance of the recommender systems [47].

C. OUR CONTRIBUTIONS

In this article, we present the development of a context-based recommender system, CRecSys, for movie domain which uses item-based collaborative filtering (IBCF) for rating prediction and recommendation. It is a major extension of one of our previously published conference papers [46], by considering larger datasets, additional evaluation metrics, comparison with many baselines and state-of-the-art methods from different perspectives, including cold-start users and limited content problem. In line to [17], CRecSys applies Natural Language Processing (NLP) and Information Extraction (IE) techniques, including LDA over a context-representing movie dataset to identify various contextual features.
(both representational and interactional) features, such as topic, subject, genre, certification, and cast-performance for movie profiling. It generates a labeled RDF graph to model items and their contextual features for computing context-based similarity between the items. In order to compute similarity between items using contextual features, CRecSys uses node-based graph matching techniques over the RDF graphs. The efficacy of CRecSys is established through experiments and validations using different evaluation metrics, such as error-based metrics (Mean Absolute Error and Root Mean Square Error) and decision support-based metrics (Precision, Recall, and F-score). CRecSys is also compared with ten baselines and two state-of-the-art methods, and performs significantly better. On empirical analysis, we found that CRecSys is able to effectively deal with the cold-start and limited content problems, in comparison to the baselines and state-of-the-art methods.

The rest of the paper is organized as follows. Section II presents a brief review of the existing literatures on context-aware recommendation, LOD-based recommender systems, and similarity computations using graph-based techniques. Section III presents a brief introduction of the preliminary concepts. Section IV presents a detailed description of the proposed CRecSys and context-based recommendation approach, including contextual feature extraction, labeled RDF graph generation, contextual feature-based semantic similarity computation, and rating prediction using item-based collaborative filtering. Section V presents the experimental setup and evaluation results. Finally, section VI concludes the paper with future directions of research.

II. RELATED WORKS

In this section, we present a brief review of the existing literatures that have used both item and user-decision based contextual features for rating prediction and recommendation. We also review the approaches that have utilized LOD and graph techniques to design recommender systems. Finally, we present various graph-based techniques that are used to compute similarity between the nodes of a graph.

A. CONTEXT-AWARE RECOMMENDER SYSTEM

The aim of recommender systems is to recommend most similar and relevant items based on the users’ profile, interest, preference, and interactions. In the existing literature, researchers have presented numerous content-based, collaborative filtering, and hybrid filtering recommendation techniques. A Recommender System (RS) is generally represented using user and item dimensions as $R : user \times item \rightarrow rating$. To handle the recommender system challenges and to improve the rating prediction and recommendation accuracy, Adomavicius et al. [3] introduced a third dimension, context, to incorporate contextual features and defined recommender system as $R : user \times item \times context \rightarrow rating$, termed as Context-Aware Recommender System (CARS). In such systems, contextual features are incorporated through contextual modeling, pre-filtering, and post-filtering algorithms.

As discussed in [8], context can be divided into three categories – (i) user-based context, (ii) item-based context, and (iii) decision-based context. In the existing literatures, incorporation of item contextual features to design context-based recommendation techniques are rare. To this end, Dourish [5] explained that an item contextual features can be categorized into representational and interactional contextual features. Representational contexts are the attributes of the users and items defining their characteristics. For example, in movie domain, genre, sub-genre, cast, director, and certification can be considered as representational contextual features. Both user-related and item-related representational contextual features are explicitly encoded in datasets and do not change over time. Moreover, representational contexts are delineable, stable, and separate from activity [5]. On the other hand, interactional contextual features are extracted from the review documents written by the users. Interactional contexts are relational property between object and activity which are dynamically defined information and arises from the activities. We use both representational and interactional contextual features for item profiling. Yao et al. [8] introduced the construction of a Multi-Layer Contextual Graph (MLCG) which includes user, item, and user-decision based contextual features. The proposed approach used implicit feedback data as contextual features to design MLCG and then applied ranking algorithms for context-based recommendations. Allahyari and Kochut [17] proposed a probabilistic topic model which incorporates movie contexts with user interests, and the contextual information is represented as a subset of the items’ feature space. To extract contextual information of movies, an external knowledgebase, DBpedia, is used. In line to [17], the proposed CRecSys also uses LOD to extract contextual information of movies.

The approaches discussed so far used item contextual features to design context-based recommendations. However, there are various approaches which have also used user-decision context features in context-based recommendation techniques. These approaches can be used for the incorporation of context in traditional recommender systems and for the extraction of contextual features. The matrix factorization-based CARS, also known as Context-Aware Matrix Factorization (CAMF), was initially introduced in [37], where authors proposed three new models – CAMF-C (context not dependent on items), CAMF-CI (context represented in item-context pair), and CAMF-CC (single model for each context-item pair). The paper used non-probabilistic matrix factorization to split user-item rating matrix into two small matrices. On the other hand, probabilistic matrix factorization in CARS was used as point-of-interest, and ratings of unrated items were determined based on the review helpfulness votes [38], [39]. Ning and Karypis [40] introduced Sparse Linear Method (SLIM), a new traditional matrix factorization approach to predict top-N recommendations for sparse ratings on unrated items. The paper handled high sparsity challenge and reduced the learning time of the models. In this direction, Zheng et al. [41] extended SLIM and
proposed a new matrix factorization approach, Contextual Sparse Linear Method (CSLIM).

The contextual features used in the approaches discussed above are either explicit (directly obtained from entities) or implicit (obtained from a monitored system for user-item interactions). Baltrunas et al. [28] introduced a music-based recommender system, InCarMusic, which predicts songs based on user-decision. In addition to these, there are some CARS in which contextual information is obtained through inference method. Hariri et al. [31] proposed a context-based music recommender system where latent topics were extracted using topic modeling techniques and used as contextual features. Similarly, Lahlou et al. [32] introduced a text classification technique to infer contextual features from review documents. Table 1 presents a list of datasets used in CARS. Although, there are various approaches for CARS, most of them used synthetic contextual feature-based datasets and questionnaire approaches for integrating contextual features, by considering only user decisions and ignoring the user and item contexts. Moreover, the sparsity in the available datasets is high because it is difficult to find all contextual features for users through inference mechanism. To handle these issues, our proposed CRecSys generates a real-world item-based contextual feature dataset where both explicit and inference methods are used to identify contextual features. The generated dataset is able to handle various issues with CARS, such as data sparsity and limited content problems. The contextual features of the users are extracted by applying LDA over the review documents generated by them.

B. LOD IN RECOMMENDER SYSTEMS

Before the introduction of LOD, most recommender systems used semantic-aware and ontology-based approaches for rating prediction [18]. LOD represents and publishes textual data in structured format using graph-based data model and semantic web technologies, generating labeled RDF graphs. Incorporation of LOD in an existing recommender system requires content-based and collaborative filtering methods [19]. In the last few years, researchers have presented various LOD-based recommender systems. Passant [20] proposed a music recommendation system using DBpedia-based features. The authors evaluated the proposed semantic similarity measure using linked data properties. In another approach, Noia et al. [21] proposed a content-based recommender system for movie domain using three knowledgebases viz. DBpedia, LinkedMDB, and Freebase.

Oliveira et al. [22] integrated LOD with recommender system and used LinkedMDB and DBpedia knowledgebases. The proposed method predicts a rating for a given input item, recommends items based on the history of the user, and recommends items to users not based on his/her history rather recommends the trending items. Musto et al. [23] presented a graph-based recommender system using LOD. The authors first applied the PageRank algorithm on the LOD-based features and then fed them into the recommender system. In continuation to this work, Musto et al. [24] developed a hybrid recommender system using popularity, content, collaborative filtering, LOD, bipartite graph, and tripartite graph-based features. Different feature combinations were given as input to three classification models for rating prediction and recommendation. The accuracy values of these approaches confirm the efficacy of the LOD-enabled recommender systems that provide significantly better rating prediction in comparison to the content-based, collaborative filtering, matrix factorization, and PageRank-based recommendation algorithms. However, LOD-based features are hardly used in existing context-based recommender systems, except the one presented in [17]. Our proposed CRecSys uses contextual features extracted from LOD to develop efficient context-aware recommender system.

C. GRAPH-BASED SIMILARITY MEASURES

Graph-based similarity measures can be used to compute semantic association between the nodes of a graph that can be words or documents. There are various measures to compute similarity between two or more graphs [14], [16], and the most common approaches to compute inter-graph similarity are based on graph isomorphism, maximum/minimum common sub-graph (super graph), and iteration. To compute similarity between two graphs using an iterative method, first an initial similarity score is given to each nodes of both the graphs, and thereafter the similarity scores are repeatedly updated using a function, such as \( \text{sim}_{i,j}^{k+1} \leftarrow f(\text{sim}_{i,j}^{k}) \). The updation process is repeated until the values converge to a stationary distribution. In this direction, Kleinberg [34] proposed an iterative method to identify authoritative information in hyper-link environment that was further modified in [16]. The iteration-based similarity measure by Blondel et al. [16] is given in equation (1), where \( E_A \) and \( E_B \) are the sets of edges for graph \( G_A \) and \( G_B \), respectively. Zager and Verghese [35] improved equation (1) and presented it for both edge and node similarity calculation, as given

\[
\text{sim}_{i,j} = \frac{\text{length}(\text{path}_{i,j})}{\sqrt{\text{length}(\text{path}_{i,j})}}
\]
in equations (2) and (3), respectively. In these equations, \( S_{k}(u, v) \) represents the edge similarity score for edge \( u \in G_{k} \) and edge \( v \in G_{B} \), and \( S_{n}(u, v) \) represents the node similarity score for the nodes \( u, v \in G_{A} \).

\[
S_{k}^{k+1}(u, v) \leftarrow \sum_{(m, n) \in E_{A}} S_{k}(m, n) + \sum_{(u, v) \in E_{B}} S_{k}(m, n) \tag{1}
\]

\[
S_{k}^{k+1}(u, v) \leftarrow S_{k}(u(i)v(j)) + S_{n}(u(i)w(j)) \tag{2}
\]

\[
S_{k}^{k+1}(u, v) \leftarrow \sum_{s(t) = u, s(b) = v} S_{k}(a, b) + \sum_{s(a) = u, s(b) = v} S_{k}(a, b) \tag{3}
\]

Heymans et al. [36] proposed to consider both similar and dissimilar terms for compute node- and edge-based similarity. To identify similar terms, the original graph and its complement are used; whereas, for dissimilar terms, each graph and complements of all other graphs in the graph network are used. One major issue with these approaches is that they do not consider all natural and desirable properties of graphs, such as fixed values range of similarity score, assignment of zero similarity to nodes that have no in-degree or out-degree, and reflexivity of nodes for graph-based similarity computation [10]. Our proposed approach applies graph matching algorithm presented in [10] over the labeled RDF graphs, which represent LOD-based contextual features.

### III. PRELIMINARIES

This section presents a brief description of various concepts like item context, LDA, and notion of similarities in graphs that are used to design our proposed CRecSys.

#### A. ITEM CONTEXTUAL FEATURES

The contextual features of an item represent constraints and contexts for their consumption by the users. For example, in movie domain, certification, cast, sub-genre, based on, and director are the contextual features. The items along with their contextual features can be defined as \( I_{nc} \), where \( n = 1, 2, \ldots, m \) and \( \{c_{1}, c_{2}, \ldots, c_{k}\} \) are \( m \) items and \( k \) contexts, respectively. Items have multiple contexts, wherein each context can have set of values. For example, \( I_{1}(c_{1}, c_{2}) = \{\text{genre} : \text{horror}, \text{sub-genre} : \text{dystopia}\} \) represents two contextual dimensions, \( c_{1} \) and \( c_{2} \) of item \( I_{1} \), where \( c_{1} \) is genre and \( c_{2} \) is sub-genre. The contextual values of \( c_{1} \) and \( c_{2} \) are horror and dystopia, respectively. As discussed in section II, most of the existing CARS have used user-decision as context; however, only few approaches have used both user- and item-based contexts. Allahyari and Kochut [17] presented an item-driven contextual features-based method for rating prediction and recommendation. The authors used actors, genres, directors, and other item-driven contextual features, which are extracted using LOD. Figure 2 presents contextual features representation of the “Sicario” movie using DBpedia and Wikidata knowledgebases.

#### B. LATENT DIRICHLET ALLOCATION

Latent Dirichlet Allocation (LDA) is a generative probabilistic and statistical modeling method to extract topics from text corpus [6]. The basic idea behind LDA is that groups of contextually similar terms constitute different topics.

![Figure 2](image_url) **FIGURE 2.** LOD-based contextual features of “Sicario” movie using DBpedia and Wikidata.

![Figure 3](image_url) **FIGURE 3.** A graphical representation of LDA model.

Figure 3 presents a graphical representation of LDA containing an outer and inner block. The outer block, \( M \), represents each of the \( M \) documents, whereas inner block, \( N \), shows each of the \( N \) words and their assigned topics. The parameters \( \alpha \) and \( \beta \) represent per-document topics and per-topic word distributions, respectively. Moreover, \( \theta \) and \( \phi \) represent topic and word distributions, respectively in the text corpus, \( z \) represents the topic assigned to the \( N^{th} \) word in \( M^{th} \) document, i.e. w.

Table 2 presents a set of topics and their associated word distribution generated by LDA over the review documents of The Revenant, Sicario, and The Visit movies.

| Movie name | Topic terms |
|------------|-------------|
| The Revenant | revenge, survival, bear, beautiful, landscape, oscar, reality, visual, effects |
| Sicario | drug, mexco, agent, action, reality |
| The Visit | grandparents, camera, footage, kids, horror, twist, visit |

#### C. NOTIONS OF SIMILARITY IN GRAPHS

There are numerous graph matching algorithms to find similarity between two graphs [13]–[15] which has wide-range of applications in Web searching, social network analysis, biological network analysis, and so on. The most widely used notions of similarity in graphs are – isomorphism, edit...
distance, statistical methods, and iterative methods. A brief
description of these notions of similarity is presented in the
following paragraphs.

- **Isomorphism**: It is based on bijective function to
  identify structural similarity between the nodes of
two or more graphs. The adjacency matrices of two
isomorphic graphs are structurally same [13].

- **Edit distance**: It calculates the minimum cost for trans-
foming a graph into another form. The number of edit
operations like addition and deletion of nodes and edges
determines the cost of graph transformation. It is used to
compute the cost function for an optimal match between
two graphs [14].

- **Statistical methods**: This is a family of similarity mea-
sures defined using graph statistics like diameter, degree
distribution, and betweenness that are used to evaluate
the similarity of graph structures [15].

- **Iterative methods**: These methods compute similarity
  between nodes or edges of a graph based on their over-
lapping neighbors. Iterative methods are applied when
the nodes of a graph (or two graphs) have overlapping
neighbors [16].

The proposed CRecSys uses node-based iterative method
to compute similarity between the nodes of each pair of
directed graphs. At each iteration, the similarity score dis-
tribution of the nodes of each graph converges towards a
static distribution. Equation (4) presents an iterative pro-
cess to compute similarity between the nodes \( u \) and \( v \), where
\( x_{in}(u, v) \) and \( x_{out}(u, v) \) represent the common in-neighbors
and out-neighbors, respectively between the nodes \( u \) and \( v \).
The iterative process is repeated until the difference between
the similarity of two consecutive iterations \((k)\)th and \((k + 1)\)th
is less than or equal to a threshold \( \delta \), i.e. \(|x^{k+1}(u, v) - x^k(u, v)| \leq \delta\).

\[
x^{k+1}(u, v) \leftarrow \frac{x_{in}^k(u, v) + x_{out}^k(u, v)}{2}
\]

IV. PROPOSED METHODOLOGY

This section presents a detailed description of the proposed
CRecSys to predict ratings of unrated items using context-
based semantic similarity. Starting with the extraction of
contextual features from LOD and movie data sources, this
section further proceeds with the discussion of labeled RDF
graph generation, semantic similarity computation, and rat-
ing estimation. A detailed description of each module of
CRecSys is presented in the following sub-sections.

A. CONTEXTUAL FEATURES EXTRACTION

This section presents the extraction process of contextual
features for movie domain. The contextual features can be
categorized into two groups – representational contextual
features and interactional contextual features [5]. The rep-
resentational contextual features are priory known for the
items; e.g., genre, director, cast, and certificate in movie
domain. On the other hand, interactional contextual features
are inferred from interactions among users and items written
in the form of reviews. The representational contextual fea-
tures are extracted from both LOD and movie data sources,
whereas interactional features are extracted only from movie
data sources. Table 3 presents a list of notations and their
descriptions used in rest of the paper.

1) LOD-BASED CONTEXTUAL FEATURES

As discussed earlier, LOD is a cloud of multiple knowl-
edgebases that are interlinked to each other. We have used
LOD to extract representational contextual features. LOD
is a collection of RDF statements that are modeled as a
labeled directed graph containing 3—tuples, \((R, L, S)\), where
\(R = \{r_1, r_2, \ldots, r_k\}\) is a set of resources (nodes), \(L = \{l_1, l_2, \ldots, l_k\}\)
is a set of links (predicates), and \(S = \{s_1, s_2, \ldots, s_k\}\) is a set of states, wherein each statement
represents the association between the underlying pair of
resources through a link known as triple. For example, a triple
statement \((r_2, l_2, r_3) \in S\) represents that resources \(r_2\) and
\(r_3 \in R\) are linked through \(l_2 \in L\).

Contextual feature of a resource \(r \in R\) in LOD is pre-
sented in \(\{property, value\}\) pair like \(\{acted, Sandra Bullock\}\),
as shown in figure 4 for the resource node “Gravity”, where
\(act\) is a contextual feature and Sandra Bullock is its value.
In LOD, contextual features of a resource node \(r\) are the
incoming and outgoing predicates. Contextual features in the
proposed CRecSys are the descriptive features of items.

Figure 4 presents node resources and their corresponding
contextual features. In this figure, “The Martian” and “Grav-
ity” movies are the resource nodes, \(r = \{The Martian, Gravity\}\), predicates (links) contain the contextual features,
\(l = \{direction, topic, act, genre, subject\}\), \(\rightarrow = \{Matt Damon, Sandra Bullock\}\), and \(\rightarrow = \{Survival, Solitude, Space\}\)
represent the incoming and outgoing contextual values,
respectively for predicate \(l\). The contextual features along
with contextual values for a resource \(r\) is computed using
equation (5). In equation (5), \(\rightarrow\) and \(\rightarrow\) represent context-
ual features and values (in/out) for the resource node \(r\),
as given in equations (6) and (7), where \(v\) represents the

| Notation | Description |
|----------|-------------|
| \(r\)   | resource node |
| \(r_{cf}\) | contextual features of resource \(r\) |
| \(l\)   | predicate (link) to connect resources |
| \(\rightarrow\) | incoming predicates to resource \(r\) |
| \(\rightarrow\) | outgoing predicates to resource \(r\) |
| \(s\)   | a triple statement |
| \(R\)   | set of resources |
| \(L\)   | set of predicates |
| \(S\)   | set of triple statements |

TABLE 3. Basic notations and their descriptions.
contextual value.

\[
    r_{cf} = r_{cf}^{in} \cup r_{cf}^{out}
\]

\[
    r_{cf}^{in} = (v, l, in), \quad r \in R, l \in L
\]

\[
    r_{cf}^{out} = (v, l, out), \quad r \in R, l \in L
\]

For example, using equations (5), (6) and (7) in figure 4, the extracted contextual features and their values for “The Martian” movie are \{- (sci-fi, genre, out), (space, subject, out), (solitude, subject, in), (survival, topics, out), (Ridley Scott, directed, in)\}. Similarly, contextual features and their values for “Gravity” movie are \{- (sci-fi, genre, out), (space, subject, out), (solitude, subject, in), (survival, topics, out), (Alfonso Cuaron, directed, in)\}. The common contextual features and their values for both movie resources are \{- (sci-fi, genre, out), (space, subject, out), (survival, topics, out)\}.

| Property | Values |
|----------|--------|
| (dbr: director) | (db: Ridley_Scott) |
| (dbp: country) | (United States) |
| (dbr: starring) | (db: Jessica_Chastain) |
| (dct: subject) | (db: Films_about_astronauts) |
| (dct: subject) | (db: Space_adventure_films) |
| (rdf: type) | (db: Wikidata:Q11424) |
| ... | ... |
| (owl: sameAs) | (freebase: The Martian (film)) |

Table 4. Some exemplar contextual features of “The Martian” movie extracted from DBpedia.

In order to extract contextual features from LOD, first, URIs of the movies are identified and mapped to the respective movie names. The movie-mapped URIs use SPARQL end-points to extract contextual features from LOD cloud using RDF triples. Table 4 presents few exemplar contextual features of “The Martian” movie extracted from DBpedia in the form of \{(property, value)\} pair. It should be noted that the contextual features like dbo:pageTitleExternalLink, dbr:thumbnail, and dbp:image that do not provide semantic information are filtered out.

2) MOVIE DATA SOURCE-BASED CONTEXTUAL FEATURES

The movie data sources (IMDB and Rotten Tomatoes) are used to extract both representational and interactional contextual features. There are few representational features like certification and ratings which are not available in LOD. Therefore, movie data sources are used to extract such features. Interactional contextual features show the interaction between users and movies in terms of ratings and reviews provided by the users on movies. The reviews provide valuable information about various aspects and context of the items, such as entities, events, entity actions, keywords, special events, and comparison (with other entities), containing various contextual information like how, when, where, and with whom a user consumed an item. We have applied LDA over movie review documents to identify contextual features.

B. RDF GRAPH GENERATION

After extraction of contextual features from LOD and movie data sources, next task is to generate RDF graphs using these features. Figure 5 presents an RDF graph in which subjects and objects are represented as nodes using rectangles and ovals, respectively, and predicates connect the pair of resources (subject and object) and represented using labeled dashed edges. The generated RDF graphs represent movies in a triplet form like \{(subject, predicate, object)\}. For example, in \{(Interstellar, director, Christopher Nolan)\} triplet, Interstellar and Christopher Nolan are the subject and object represented using rectangle and oval, respectively, and director is the predicate connecting the underlying subject and object.

C. CONTEXT-BASED SEMANTIC SIMILARITY

In this section, we formulate the proposed CRecSys approach to compute contextual feature-based similarity between items. In line to [10], [16], CRecSys calculates similarity between two nodes based on their overlapping contextual features. The proposed semantic similarity metric holds the following properties.

- Two nodes \(i\) and \(j\), where \(i \in G_A\) and \(j \in G_B\) are said to be similar if they do not have any in-neighbors and out-neighbors, i.e., they are isolated nodes [36]. This is only applicable to directed graphs. We have modified this property in this study such that two nodes \(i\) and \(j\) are completely dissimilar (i.e., similarity score is 0) if the nodes do not have any in-neighbors and out-neighbors.
- The similarity score between a pair of nodes \((i, j)\) is in a particular range. This property defines that the computed semantic similarity between nodes is always within a range \((0, 1)\).
- Every node of a graph is related to itself. This is equivalent to reflexive property and defines that every node or edge in a graph is similar to itself.

The extracted contextual features of items (nodes) are assigned an initial weight. Existing state-of-the-art methods do not distinguish between rarely and frequently occurring
contextual features and assign a binary value of 1 and 0 to the matched and unmatched nodes, respectively. Unlike binary assignment in the existing approaches, our proposed approach assigns higher weights to rarely occurring features and lower weights to frequently occurring features, as defined in equation (8). In this equation, \( f_i \) and \( f_m \) are the features of node \( i \) and \( j \), respectively; and \( C(f_i) \) represents the completeness score of feature \( f_i \). The distinctive features are highly informative in comparison to frequently occurring features, which are less informative. The completeness of a feature \( f \) is calculated using equation (9), where \( n \) represents the number of resources (i.e., movies) containing the contextual features \( f_i \), and \( N \) represents the number of resources in the labeled RDF graph.

\[
\text{sim}(f_i, f_m) = \begin{cases} 
1 - C(f_i), & \text{if } f_i = f_m \\
0, & \text{otherwise}
\end{cases} \tag{8}
\]

\[
C(f_i) = \frac{n}{N} \tag{9}
\]

Thereafter, an initial similarity score is computed between each pair of resource nodes \( i \) and \( j \) using equations (10) to (12). The initial similarity value is based on the in-degree and out-degree of \( i \) and \( j \), as presented in equations (10) and (11), where \( \text{deg}_{in}(i) \), \( \text{deg}_{out}(i) \), \( \text{deg}_{in}(j) \) and \( \text{deg}_{out}(j) \) represent the cardinality of in-degree and out-degree of nodes \( i \) and \( j \), respectively.

\[
\text{inSim}^1(i, j) = \frac{\min\{\text{deg}_{in}(i), \text{deg}_{in}(j)\}}{\max\{\text{deg}_{in}(i), \text{deg}_{in}(j)\}} \tag{10}
\]

\[
\text{outSim}^1(i, j) = \frac{\min\{\text{deg}_{out}(i), \text{deg}_{out}(j)\}}{\max\{\text{deg}_{out}(i), \text{deg}_{out}(j)\}} \tag{11}
\]

\[
\text{CBSS}^1(i, j) = \frac{\text{inSim}^1(i, j) + \text{outSim}^1(i, j)}{2} \tag{12}
\]

Finally, context-based semantic similarity (CBSS) between each pair of resource nodes \( i \) and \( j \) is updated in an iterative manner, based on the completeness between their in-neighbors and out-neighbors, as given in equations (14) and (15), respectively. In these equations, \( \text{sim} \) is the similarity between \( i \) and \( j \) based on contextual features \( (f_i, f_j) \) using equation (8), and \( \text{inSim}(i, j) \) and \( \text{outSim}(i, j) \) represent in-degree and out-degree-based similarity between \( i \) and \( j \). This process is repeated until convergence, i.e., \(|\text{CBSS}^{k+1}(i, j) - \text{CBSS}^k(i, j)| \leq \delta\).

\[
\text{CBSS}^{k+1}(i, j) \leftarrow \text{inSim}^{k+1}(i, j) + \text{outSim}^{k+1}(i, j) \tag{13}
\]

\[
\text{inSim}^{k+1}(i, j) = \text{CBSS}^k(i, j) \frac{1}{\max\{\text{deg}_{in}(i), \text{deg}_{in}(j)\}} \times \sum_{l=1}^{\text{deg}_{in}(i)} \sum_{m=1}^{\text{deg}_{in}(j)} \text{sim}(\text{inNeighbor}^l_i, \text{inNeighbor}^m_j) \tag{14}
\]

\[
\text{outSim}^{k+1}(i, j) = \text{CBSS}^k(i, j) \frac{1}{\max\{\text{deg}_{out}(i), \text{deg}_{out}(j)\}} \times \sum_{l=1}^{\text{deg}_{out}(i)} \sum_{m=1}^{\text{deg}_{out}(j)} \text{sim}(\text{outNeighbor}^l_i, \text{outNeighbor}^m_j) \tag{15}
\]

D. RATING ESTIMATION

This section presents the process of rating estimation for the unrated items using item-based collaborative filtering (IBCF) and CBSS in line to the work reported in [7]. To estimate the rating value of a user (say \( u \)) on an unrated item (say \( i \)), first top-\( k \) items, \( I_u^k \), similar to \( i \) are identified using CBSS. Finally, estimated rating of \( u \) on \( i \), \( \hat{r}_{ui} \), is computed as the weighted average of rating of \( u \) on each \( j \in I_u^k \), i.e. \( r_{uj} \), such that each \( r_{uj} \) is adjusted using corresponding CBSS as weight, as given in equation (16).

\[
\hat{r}_{ui} = \frac{\sum_{j \in I_u^k} \text{CBSS}(i, j) r_{uj}}{\sum_{j \in I_u^k} \text{CBSS}(i, j)} \tag{16}
\]
However, there is a great chance that certain users may provide low or high rating because of their critical nature or biases, adversely affecting the estimated rating. To handle this issue, like [7], we have used first-order approximation in rating estimation, as given in equation (17), where \( b_{ui} = b_n + \mu + b_i \), \( \mu \) is the average rating of all movies, and \( b_n \) and \( b_i \) are the observed user rating deviation and item rating deviation, respectively.

\[
\hat{r}_{ui} = b_{ui} + \frac{\sum_{j \in I_i} CBSS(i,j)(r_{uj} - b_{ij})}{\sum_{j \in I_i} |CBSS(i,j)|}
\]

(17)

V. EXPERIMENTAL SETUP AND RESULTS

This section presents the experimental evaluation of our proposed CRecSys method. Starting with a brief description of the dataset curation process, evaluation metrics, and various baseline methods, it presents performance evaluation results of CRecSys in comparison to baseline methods and two state-of-the-art methods, MORE (MOvie REcommendation) [4] and PICSS (Partial Information Content Semantic Similarity) [33], which used semantic similarity-based method for rating prediction. It also presents a comparative analysis of CRecSys in comparison to the baseline methods and state-of-the-art methods to deal with the cold-start users. Finally, it presents an empirical analysis of CRecSys, showing the impact of LOD to deal with the limited content problem.

### TABLE 5. Statistics of the Movie\textsubscript{LOD} and MovieLens-1M datasets.

| Category     | Movie\textsubscript{LOD} | MovieLens-1M |
|--------------|---------------------------|--------------|
| #Users       | 49080                     | 6040         |
| #Items       | 1300                      | 3883         |
| #Reviews     | 250,882                   | 232,852      |
| #Ratings     | 191,050                   | 1,000,209    |
| Data Sparsity| 99.70%                    | 95.73%       |

A. DATASET CURATION

As discussed earlier, existing datasets of movie domain, such as LDOS-CoMoDa [25] and DePaulMovie [27] generally do not contain item-based contextual features. Therefore, we crawled and constructed a new movie dataset from IMDB, Rotten Tomatoes, and DBpedia. We named the curated movie dataset as Movie\textsubscript{LOD}. For this, we developed a crawler\textsuperscript{1} in Python using urllib\textsuperscript{2} and beautifulsoup\textsuperscript{3} libraries to extract data from the aforementioned movie data sources. Since LOD provides SPARQL endpoints with each domain to consume linked data on the Web, we have used SPARQLWrapper\textsuperscript{4}, a python library, to access the endpoints of linked data. Table 5 presents a brief statistics of our curated dataset.

\textsuperscript{1}https://github.com/vineet-sejwal/CBSS-Context-based-Semantic-Similarity
\textsuperscript{2}https://docs.python.org/3/library/urllib.html
\textsuperscript{3}https://pypi.org/project/beautifulsoup4/
\textsuperscript{4}https://pypi.org/project/SPARQLWrapper/

The crawled movie dataset contains contextual features like based on, about, and cast with different categories of data, such as users’ reviews and ratings, as shown in table 5. The movies at IMDB are rated on a 10-point scale by the users, where 1 and 10 represent the lowest and highest ratings, respectively. We retrieved a total number of 191050 ratings from 49080 different users of 1300 movies. Thereafter, we applied LDA over movie review documents to identify latent topics that represent contextual features. In addition, data retrieved using the subject property (dct:subject) of LOD are filtered and segmented into sub-genres using the phrases like about and based on. Finally, contextual features are modeled as a labeled RDF graph.

As described in table 5, we have also used a benchmark dataset, MovieLens-1M,\textsuperscript{5} which is frequently used for empirical evaluation of the movie-based recommender systems. In order to generate contextual features for all movies of the MovieLens-1M dataset, each of them is mapped to a DBpedia entry, and a mapping table\textsuperscript{6} is used to identify its contextual features, in line to [17].

B. EVALUATION METRICS

This section presents a detailed description of the metrics that are used to evaluate our proposed CRecSys rating prediction model, and to perform comparative analysis with the baselines and state-of-the-art methods. The evaluation metrics used in this study are briefly described in the following paragraphs.

- Error-based metrics: These metrics are used to evaluate prediction error of the filtering algorithms by computing difference between actual and predicted ratings. We have used MAE and RMSE error-based metrics for evaluating the rating prediction methods. MAE is defined as the average of the absolute differences between the actual and predicted ratings over a set of items [9], as given in equation (18). On the other hand, RMSE is computed as the square root of the average of the square of the differences between the actual and predicted ratings, as given in equation (19). RMSE measures the intensity of data in context to best fit to a line and penalizes large error values. In equations (18) and (19), \( r_{ui} \) and \( \hat{r}_{ui} \) are the actual and predicted ratings, respectively of user \( u \) on item \( i \), and \( T \) represents the test dataset.

\[
MAE = \frac{\sum_{(u,i) \in T} |r_{ui} - \hat{r}_{ui}|}{|T|}
\]

(18)

\[
RMSE = \sqrt{\frac{\sum_{(u,i) \in T} (r_{ui} - \hat{r}_{ui})^2}{|T|}}
\]

(19)

- Decision support-based metrics: This category of metrics evaluates the accuracy of a recommender system based on the recommended list of items to a user. It presents the evaluation results in terms of Precision,
Recall, and F-score. In context of recommender system, Precision and Recall are computed using the sets of relevant and recommended items to users. Relevant items are those items that are liked by the users and contain actual ratings, whereas recommended items are the set of predicted items containing predicted ratings. Precision is the fraction of relevant recommended items to the total number of recommended items, as defined in equation (20). On the other hand, Recall is the fraction of relevant recommended items to the total number of relevant items in the dataset, as given in equation (21). Finally, F-score is the harmonic mean of Precision and Recall, as given in equation (22).

\[
\text{Precision}(P) = \frac{\text{#recommended relevant items}}{\text{#of recommended items}} \quad (20)
\]

\[
\text{Recall}(R) = \frac{\text{#of relevant items}}{\text{#recommended relevant items}} \quad (21)
\]

\[
F - \text{score}(F) = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (22)
\]

C. BASELINE METHODS

In order to establish the efficacy of the proposed CRecSys method, we have considered 10 baseline methods viz. non-negative matrix factorization (NMF), singular-value decomposition (SVD), SVD++, three variants of k-nearest neighbors (KNN), normal predictor, baseline, slope one, and co-clustering for experimental evaluation. A brief description of these baseline methods is presented in the following paragraphs.

- NMF, SVD, and SVD++ methods are based on matrix factorization (MF), where user-item interaction matrix is factorized into two new matrices – user interests and item feature. MF helps to identify latent features and preferences of users and items.

- Co-clustering method is based on pair-wise interactions of two simultaneous entities. Co-clustering-based rating prediction for items is presented in equation (23), where \( C_j \) and \( C_u \) are the average ratings of user cluster and items cluster, respectively, and \( C_{ui} \) represents the average rating of co-cluster \( (C_{ui}) \) [42].

\[
\hat{r}_{ui} = \bar{C}_{ui} + (\mu_u - \bar{C_u}) + (\mu_i - \bar{C_i}) \quad (23)
\]

- **Slope One** based rating prediction method is based on user and item average ratings for rating prediction, as presented in equation (24). In this equation, \( \text{dev}_{j,i} \) represents the rating deviation of item \( j \) on item \( i \), \( R_j \) is the set of relevant items, and \( \bar{n} \) is the user’s average rating [43].

\[
\hat{r}_{uj} = \bar{n} + \frac{1}{\text{card}(R_j)} \sum_{i \in R_j} \text{dev}_{j,i} \quad (24)
\]

- **K-nearest neighbors** (KNN) is a memory-based collaborative filtering approach which uses user-item rating matrix to predict ratings. The neighborhood formation in KNN is formulated using users or items-based similarity approaches. The KNN-based rating prediction is presented in equation (25), where \( \text{sim}(i,j) \) represents similarity between users \( i \) and \( j \), \( r_{uj} \) represents user \( u \) rating on item \( j \), and \( k \) represents the number of similar users to \( u \). To compute centered-KNN, equation (25) is modified as \( \hat{r}_{ui} + \mu_u \), where \( \mu_u \) is the mean rating of the users. Similarly, for KNN-baseline, equation (25) is modified as \( \hat{r}_{ui} + b_u + b_i \), where \( b_u = \mu_u + \sigma_u \), \( \mu \) is the mean of the user ratings, and \( b_i \) and \( b_u \) are the users’ and items’ observed rating deviations, respectively.

\[
\hat{r}_{ui} = \frac{\sum_{j \in N^k_i} \text{sim}(i,j) r_{uj}}{\sum_{j \in N^k_i} \text{sim}(i,j)} \quad (25)
\]

- Baseline-based rating prediction method uses both user and item biases to predict ratings for unrated items, as presented in equation (26). In this equation, \( \mu \) represents the average ratings (users or items), and \( b_i \) and \( b_u \) are the observed items’ and users’ rating deviations, respectively.

\[
\hat{r}_{ui} = \mu + b_i + b_u \quad (26)
\]

- Normal predictor predicts random ratings for users through normal distribution method, \( N(\mu, \sigma) \) on rating-based training set, where \( \mu \) and \( \sigma \) represent mean and variance, respectively, and computed using maximum likelihood estimation. The predicted rating \( \hat{r}_{ui} \) uses \( \mu \) and \( \sigma \) to compute ratings on unrated items, as presented in equation (27) and (28), respectively. In these equations, \( R_{\text{train}} \) is the rating-based training set and \( r_{ui} \) is the rating given by user \( u \) on item \( i \).

\[
\mu = \frac{1}{|R_{\text{train}}|} \sum_{r_{ui} \in R_{\text{train}}} r_{ui} \quad (27)
\]

\[
\sigma = \frac{\sum_{r_{ui} \in R_{\text{train}}} (r_{ui} - \mu)}{|R_{\text{train}}|} \quad (28)
\]

D. STATE-OF-THE-ART METHODS

The proposed CRecSys method is compared with two state-of-the-art methods viz. MORE [4] and PICSS [33] that are briefly described in the following paragraphs.

- **MORE** [4] computes semantic similarity between movies using a variant of the Vector Space Model (VSM) by exploiting LOD. Given \( p \) property values, each movie \( m_j \) is represented as a \( d \)-dimensional vector, as shown in equation 29. In this equation, \( w_{d,j,p} \) is TF-IDF computed using equation 30, where \( \text{freq}_{n,p} \) is TF of element \( n \), \( N \) is the total number of movies in the dataset, and \( d_{n,p} \) is the number of movies having property \( p \). The \( d \)-dimensional vectors of the movies are used to compute Cosine similarity between the movies for rating prediction.

\[
m_{jp} = \left( w_{1,j,p}, w_{2,j,p}, \ldots, w_{d,j,p} \right) \quad (29)
\]

\[
w_{n,j,p} = \text{freq}_{n,p} * \log \left( \frac{N}{d_{n,p}} \right) \quad (30)
\]
• PICSS [33] used LOD to compute semantic similarity between items for rating prediction. PICSS computes information content of each feature in LOD and uses modified Tversky ratio model to compute semantic similarity between two items, as given in equation 31. In this equation, PICSS(\(i,j\)) computes the semantic similarity between items \(i\) and \(j\) and PIC(\(F_i\)) represents the set of partial information content for item \(i\) in the dataset. The partial information content of an item shows its appropriateness in the dataset. The more the information is shared in the dataset, the less it is distinctive in the dataset. This signifies that resources (items) with more distinctive features are more informative.

\[
\text{PICSS}(i,j) = \frac{\text{PIC}(F_i \cap F_j)}{\text{PIC}(F_i \cap F_j) + \text{PIC}(F_i - F_j) + \text{PIC}(F_j - F_i)}
\]  

(31)

E. COMPARATIVE EVALUATION RESULTS OF CRecSys WITH BASELINE METHODS

In this section, we present a comparative evaluation of CRecSys with all 10 baseline methods using MAE, RMSE, Precision, Recall, and F-score values. To this end, we have used SurPRISE,7 a Python library, to implement the baseline methods. Figures 6 and 7 present the evaluation results in terms of MAE and RMSE values, whereas figures 8 and 9 present the evaluation results in terms of Precision, Recall, and F-Score values for both MovieLOD and MovieLens-1M datasets, respectively. It can be observed from figures 6 and 7 that CRecSys outperforms all baseline methods in terms of MAE and RMSE values. It can also be observed that SVD++ has lowest MAE and RMSE values, whereas Normal Predictor has highest MAE and RMSE values. SVD++ performed better in comparison to other baseline approaches because it includes implicit feedback information (implicit ratings). On the other hand, Normal Predictor contains user biases; hence, there is great chance that prediction value is low for high rated movies and vice versa. CRecSys performs 9.94\% better in terms of MAE and 8.96\% better in term of RMSE in comparison to SVD++ over MovieLOD dataset, as shown in figure 6. Similarly, CRecSys performs 8.51\% better in terms of MAE and 9.17\% better in term of RMSE in comparison to SVD++ over MovieLens-1M dataset, as shown in figure 7.

---

7http://surpriselib.com/ (last accessed: 30th July 2020)
It can be observed from figure 8 that CRecSys performs 4.62% better in terms of Precision, 5.77% better in terms of Recall, and 4.72% better in terms of F-score, in comparison to SVD++ over MovieLOD dataset. Similarly, it can be observed from figure 9 that CRecSys performs 6.65% better in terms of Precision, 7.83% better in terms of Recall, and 7.28% better in terms of F-score, in comparison to SVD++, over MovieLens-1M dataset.

F. COMPARATIVE EVALUATION RESULTS OF CRecSys WITH STATE-OF-THE-ART METHODS

In this evaluation, CRecSys is compared with two state-of-the-art methods, MORE [4] and PICSS [33], which have also used LOD to compute semantic similarity between items for rating prediction. Though both MORE and PICSS are similar to our proposed work, they only consider features extracted from LOD to compute semantic similarity. On the other hand, CRecSys uses multiple movie data sources, LOD, and review documents to extract contextual features, which seem very important to determine most similar movies. For comparative evaluation, the test dataset includes only those users in the dataset who have rated more than 5 movies. Table 6 presents the comparative evaluation results of CRecSys, MORE, and PICSS in terms of MAE and RMSE values, whereas table 7 presents the results in terms of Precision, Recall, and F-Score for different values of $k$, where $k$ represents top-$k$ nearest neighbors. It can be observed from table 6 that the lowest MAE and RMSE values for both CRecSys and state-of-the-art methods are at $k = 30$. Although, both MORE and PICSS use the concept of semantic similarity, PICSS performs comparatively better because it first identifies the overlapping features between two movies and then computes their semantic similarity. On analysis, we found that CRecSys outperforms PICSS with an improvement of 4.24% and 4.56% over MovieLOD dataset and 6.50% and 4.47% over MovieLens-1M datasets in terms of MAE and RMSE values, respectively. Similarly, it can be observed from table 7 that CRecSys outperforms PICSS in terms of Precision, Recall, and F-score by 3.96%, 3.18%, and 3.42% over MovieLOD dataset and 4.79%, 6.46%, and 4.10% over MovieLens-1M dataset, respectively.

G. DEALING WITH COLD-START USERS

In this section, we present an empirical evaluation of CRecSys in comparison to the baselines and state-of-the-art methods to deal with the problem of rating prediction for cold-start users. The cold-start problem occurs when some users rate very few items, and rating prediction for such users...
TABLE 6. Comparative performance evaluation results of CRecSys vs. MORE vs. PICSS in terms of MAE and RMSE values over Movie LOD and MovieLens-1M datasets.

| Dataset      | CRecSys | MORE [4] | PICSS [33] | CRecSys | MORE [4] | PICSS [33] | CRecSys | MORE [4] | PICSS [33] | CRecSys | MORE [4] | PICSS [33] |
|--------------|---------|----------|------------|---------|----------|------------|---------|----------|------------|---------|----------|------------|
|               | MAE     | RMSE     | MAE        | RMSE    | MAE       | RMSE       | MAE     | RMSE     | RMSE       | MAE     | RMSE     | RMSE       |
| Movie LOD     | 0.7198  | 0.9215   | 0.6738     | 0.8591  | 0.6525    | 0.8321     | 0.6799  | 0.8608   | 0.8612     | 0.7368  | 0.9082   | 0.8872     |
| MovieLens-1M  | 0.7016  | 0.8911   | 0.6743     | 0.8652  | 0.6350    | 0.8267     | 0.6528  | 0.8410   | 0.8457     | 0.7259  | 0.9146   | 0.8693     |

TABLE 7. Comparative performance evaluation results of CRecSys vs. MORE vs. PICSS in terms of Precision (P), Recall (R), and F-score (F) for different values of $k$ over Movie LOD and MovieLens-1M datasets.

| Dataset      | k=10  | P      | R      | F      | k=20  | P      | R      | F      | k=30  | P      | R      | F      | k=40  | P      | R      | F      |
|--------------|-------|--------|--------|--------|-------|--------|--------|--------|-------|--------|--------|--------|-------|--------|--------|--------|
| Movie LOD    | CRecSys | 0.8263 | 0.7571 | 0.7901 | 0.8101 | 0.7615 | 0.7988 | 0.8867 | 0.8067 | 0.8436 | 0.8591 | 0.7859 | 0.8208 |
|              | MORE [4] | 0.7811 | 0.7164 | 0.7473 | 0.7883 | 0.7242 | 0.7548 | 0.8017 | 0.7619 | 0.7962 | 0.7971 | 0.7761 |        |
|              | PICSS [33] | 0.8042 | 0.7514 | 0.7768 | 0.8283 | 0.7984 | 0.7923 | 0.8515 | 0.7810 | 0.8147 | 0.8514 | 0.7754 | 0.8115 |
| MovieLens-1M | CRecSys | 0.8673 | 0.7710 | 0.8163 | 0.9183 | 0.8210 | 0.8669 | 0.9425 | 0.8461 | 0.8917 | 0.9316 | 0.8374 | 0.8819 |
|              | MORE [4] | 0.8102 | 0.7251 | 0.7652 | 0.8425 | 0.7663 | 0.8025 | 0.8693 | 0.7850 | 0.8250 | 0.8591 | 0.7682 | 0.8111 |
|              | PICSS [33] | 0.8397 | 0.7529 | 0.7939 | 0.8751 | 0.7910 | 0.8309 | 0.8973 | 0.7914 | 0.8551 | 0.8914 | 0.7862 | 0.8355 |

is still an open challenge in the field of recommender system. Cold-start problem may occur for items as well, when some of the items receive very few user ratings. However, in this study, we have considered rating prediction for cold-start users only.

1) **CRecSys VS. BASELINE METHODS**

To perform empirical evaluation of CRecSys and baseline methods, we considered all those users who rated at most 5 items as cold start users, as used in [44], [45] as well. Hence, we repeated the same experiment, discussed in previous section, on the dataset containing only cold-start users. Figures 10 and 12 present the comparison results of CRecSys with baseline methods in terms of MAE and RMSE values over both datasets. Similarly, figures 11 and 13 present the performance comparison results in terms of Precision, Recall, and F-score values over both datasets. It can be observed from these figures that both MAE and RMSE values are high, whereas Precision, Recall, and F-score values are low in comparison to previous experiments for all methods.

One possible reason behind such results is data sparsity, which is very high for cold-start users. In baseline methods, SVD++ performed best because it considers implicit ratings. But, CRecSys performed significantly better in comparison to SVD++ because it uses contextual features that help to identify similar items for those who have rated very few items. In comparison to SVD++, CRecSys showed an improvement of 19.79% and 16.35% over Movie LOD and 5.67% and 4.92% over MovieLens-1M datasets in terms of MAE and RMSE values, respectively. Similarly, CRecSys also performed better in terms of Precision, Recall, and F-score values over both datasets.

2) **CRecSys VS. STATE-OF-THE-ART METHODS**

In this section, we present a comparative evaluation of CRecSys vs. state-of-the-art methods viz. MORE [4] and PICSS [33] to deal with the problem of cold-start users. Like previous experiment, we considered the dataset containing only cold-start users, i.e., users who rated at most 5 items. Figure 14 presents the comparison results of CRecSys with...
state-of-the-art methods in terms of MAE and RMSE values. Similarly, figure 15 presents the comparison results in terms of Precision, Recall, and F-score values. The lowest MAE and RMSE values for both CRecSys and state-of-the-art methods are at $k = 30$. It can be observed from figure 14 that CRecSys outperforms both MORE and
V. K. Sejwal et al.: CRecSys: A Context-Based RS Using Collaborative Filtering and LOD

**H. IMPACT OF LOD TO DEAL WITH THE LIMITED CONTENT PROBLEM**

In this section, we present an empirical evaluation of CRecSys to deal with the limited content problem. As discussed earlier, limited content problem arises due to unavailability of sufficient contents for items. To handle this issue, we have integrated LDA and LOD to CRecSys to find most similar items (movies). Table 8 presents top-5 movies similar to the “Avengers: Age of Ultron” movie using both LDA and LOD, and LDA alone. It can be observed from this table that similar movies identified using both LDA and LOD-based features are highly similar in terms of subject, genre, and themes, in comparison to the similar movies identified using only LDA-based features. Table 9 presents the impact of LOD over the MAE and RMSE values when it is integrated with CRecSys.

**TABLE 8. Impact of LDA and LOD to identify top-5 movies similar to the “Avengers: Age of Ultron” movie.**

| Movie         | Sim. Value | Movie         | Sim. Value |
|---------------|------------|---------------|------------|
| Ex Machina    | 0.2648     | After Earth   | 0.2991     |
| Thor 2        | 0.2450     | Fantastic Four| 0.3262     |
| Chappie       | 0.2442     | The Hunger Games| 0.3352 |
| The Winter Soldier| 0.2088 | Pacific Rim   | 0.1932     |
| Pacific Rim   | 0.2041     | Oblivion      | 0.1842     |
VI. CONCLUSION AND FUTURE WORK

In this article, we have proposed a context-based recommender system, CRecSys, which computes semantic similarity between movies using contextual features and predicts ratings of the unrated movie items. We have also curated a movie dataset containing contextual features from two movie data sources and LOD. The main advantage of integrating movie data sources and LOD is to define movie context in a broader perspective. We have also extended item representation approach by incorporating latent topics extracted using LDA from movie review documents. The novelty of CRecSys lies in predicting ratings using items’ contextual features and item-based collaborative filtering. The efficacy of CRecSys is evaluated using well-known metrics like MAE, RMSE, Precision, Recall, and F-score. Moreover, it is compared with ten baseline recommendation methods and two state-of-the-art methods – MORE [4] and PICSS [33], and performs significantly better. One of the distinguishing advantages of CRecSys is to handle the cold-start problem effectively in comparison to the baselines and state-of-the-arts methods. The overall rating prediction results of CRecSys for cold-start users are significantly better than the baselines and state-of-the-art methods. It is also found that incorporation of LOD improves the performance of CRecSys, mainly to deal with the limited content problem. Application of deep learning techniques, mainly word representation models, over textual data to identify contextual features seems one of the promising directions of research for the development of context-aware recommender systems.

REFERENCES

[1] G. D. Abowd, A. K. Dey, P. J. Brown, N. Davies, M. Smith, and P. Steggles, “Towards a better understanding of context and context-awareness,” in Proc. Int. Symp. Handheld Ubiquitous Comput., Karlsruhe, Germany, Sep. 1999, pp. 304–307.

[2] I. Cantador and P. Castells, “Semantic contextualisation in a news recommender system,” in Proc. CARS-RecSys, New York, NY, USA, Oct. 2009, pp. 1–5.

[3] G. Adomavicius, B. Mobasher, F. Ricci, and A. Tuzhilin, “Context-aware recommender systems,” Assoc. Advancement Artif. Intell., vol. 32, no. 3, pp. 217–253, 2011.

[4] R. Mirizzi, T. D. Noia, A. Ragone, V. C. Ostuni, and E. D. Sciascio, “Movie recommendation with DBpedia,” in Proc. IIR, Bari, Italy, 2012, pp. 101–112.

[5] P. Dourish, “What we talk about when we talk about context,” Pers. Ubiquitous Comput., vol. 8, no. 1, pp. 19–30, Feb. 2004.

[6] D. M. Blei, A. Y. Ng, and M. I. Jordan, “Latent Dirichlet allocation,” J. Mach. Learn. Res., vol. 3, pp. 993–1022, Mar. 2003.

[7] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, “Collaborative filtering recommender systems,” The Adaptive Web, vol. 4321, Berlin, Germany: Springer, 2007, pp. 291–324.

[8] W. Yao, J. He, G. Huang, J. Cao, and Y. Zhang, “A graph-based model for context-aware recommendation using implicit feedback data,” World Wide Web, vol. 18, no. 5, pp. 1351–1371, Sep. 2015.

[9] J. L. Herlocker, J. A. Konstan, L. G. Terveen, and J. T. Riedl, “Evaluating collaborative filtering recommender systems,” ACM Trans. Inf. Syst., vol. 22, no. 1, pp. 5–53, Jan. 2004.

[10] M. Nikolić, “Measuring similarity of graph nodes by neighbor matching,” Intell. Data Anal., vol. 16, no. 6, pp. 865–878, Nov. 2012.

[11] R. Catherine and W. Cohen, “Personalized recommendations using knowledge graphs: A probabilistic logic programming approach,” in Proc. 10th ACM Conf. Recommender Syst., Boston, MA, USA, Sep. 2016, pp. 325–332.

[12] T. Di Noia, “Recommender systems meet linked open data,” in Proc. ICWE, Lugano, Switzerland, 2016, pp. 620–623.

[13] M. Pelillo, “Replicator equations, maximal cliques, and graph isomorphism,” in Proc. NIPS, Denver, CO, USA, 1999, pp. 550–556.

[14] H. Bunke, “Error correcting graph matching: On the influence of the underlying cost function,” IEEE Trans. Pattern Anal. Mach. Intell., vol. 21, no. 9, pp. 917–922, Sep. 1999.

[15] S. Dill, R. Kumar, K. S. McCurley, S. Rajagopalan, S. Sivakumar, and A. Tomkins, “Self-similarity in the Web,” ACM Trans. Internet Technol., vol. 2, no. 3, pp. 205–223, 2002.

[16] V. D. Blondel, A. Gajardo, M. Heymans, P. Senellart, and P. Van Dooren, “A measure of similarity between graph vertices: Applications to synonym extraction and Web searching,” SIAM Rev., vol. 46, no. 4, pp. 647–666, Jan. 2004.

[17] M. Allahyari and K. Kochut, “Semantic context-aware recommendation via topic models leveraging linked open data,” in Proc. WISE, Shanghai, China, Nov. 2016, pp. 263–277.

[18] I. Cantador, A. Bellogín, and P. Castells, “A multilayer ontology-based hybrid recommendation model,” AI Commun., vol. 21, nos. 2–3, pp. 203–210, 2008.

[19] T. D. Noia and V. C. Ostuni, “Recommender systems and linked open data,” in Proc. Reasoning Web Int. Summer School, Berlin, Germany, Jul. 2015, pp. 88–113.

[20] A. Passant, “dibrec—Music recommendations using DBpedia,” in Proc. ISWC, Bonn, Germany, Nov. 2010, pp. 209–224.

[21] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker, “Linked open data to support content-based recommender systems,” in Proc. ICSS, Graz, Austria, 2012, pp. 1–8.

[22] J. Oliveira, C. Delgado, and A. C. Assaife, “A recommendation approach for consuming linked open data,” Expert Syst. Appl., vol. 72, pp. 407–420, Apr. 2017.

[23] C. Musto, P. Basile, P. Lops, M. de Gemmis, and G. Semeraro, “Introducing linked open data in graph-based recommender systems,” Inf. Process. Manage., vol. 53, no. 2, pp. 405–435, Mar. 2017.

[24] C. Musto, P. Lops, M. de Gemmis, and G. Semeraro, “Semantics-aware recommender systems exploiting linked open data and graph-based features,” Knowl.-Based Syst., vol. 136, pp. 1–14, Nov. 2017.

[25] A. Košir, A. Odic, M. Kunaver, M. Tkalcic, and J. F. Tasic, “Database for contextual personalization,” Elektrotehniški Vestnik, vol. 78, no. 5, pp. 270–274, 2011.

[26] L. Baltrunas, K. Church, A. Karatzoglou, and N. Oliver, “Frappe: Understanding the usage and perception of mobile app recommendations in-the-wild,” 2015, arXiv:1505.03014. [Online]. Available: http://arxiv.org/abs/1505.03014

| k=10 | k=20 | k=30 | k=40 |
|------|------|------|------|
| CRecSys+LDA+LOD | MAE | RMSE | MAE | RMSE | MAE | RMSE | MAE | RMSE |
| 0.7198 | 0.9215 | 0.6738 | 0.8591 | 0.6525 | 0.8321 | 0.6799 | 0.8608 |
| CRecSys+LDA | MAE | RMSE | MAE | RMSE | MAE | RMSE | MAE | RMSE |
| 0.7515 | 0.9759 | 0.7085 | 0.9093 | 0.6849 | 0.8815 | 0.7017 | 0.9041 |
Y. Zheng, B. Mobasher, and R. Burke, “CARSKit: A java-based context-aware recommendation engine,” in Proc. IEEE Int. Conf. Data Mining Workshop (ICDMW), Atlantic City, NJ, USA, Nov. 2015, pp. 1668–1671.

L. Baltrunas, M. Kaminskas, B. Ludwig, O. Moling, F. Ricci, A. Aydin, K. Lüke, and R. Schweiger, “Incarmusic: Context-aware music recommendations in a car,” in Proc. Int. Conf. Electron. Commerce Web Technol., Toulouse, France, Aug./Sep. 2010, pp. 89–100.

M. Brauhofer, M. Elahi, and F. Ricci, “STS: A context-aware mobile recommender system for places of interest,” in Proc. UMAP, Aalborg, Denmark, Jul. 2014, pp. 1–12.

Y. Zheng, R. Burke, and B. Mobasher, “Differential context relaxation for context-aware travel recommendation,” in Proc. EC-Web, Vienna, Austria, Sep. 2012, pp. 88–99.

N. Hariri, B. Mobasher, and R. Burke, “Context-aware music recommendation based on latenttopic sequential patterns,” in Proc. RecSys, Dublin, Ireland, Sep. 2012, pp. 131–138.

F. Z. Lahliou, H. Benbrahimand, A. Mountassir, and I. Kassou, “Context extraction from reviews for context aware recommendation using text classification techniques,” in Proc. ACS Int. Conf. Comput. Syst. Appl. (AICCSA), Fes, Morocco, May 2013, pp. 1–4.

R. Meymandpour and J. G. Davis, “A semantic similarity measure for linked data: An information content-based approach,” Knowl.-Based Syst., vol. 109, pp. 276–293, Oct. 2016.

J. M. Kleinberg, “Authoritative sources in a hyperlinked environment,” J. ACM, vol. 46, no. 5, pp. 604–632, Sep. 1999.

L. A. Zager and G. C. Verghese, “Graph similarity scoring and matching,” Appl. Math. Lett., vol. 21, no. 1, pp. 86–94, Jan. 2008.

M. Heymans and A. K. Singh, “Deriving phylogenetic trees from the similarity analysis of metabolic pathways,” J. Bioinf., vol. 119, no. 2, pp. 138–146, 2003.

L. Baltrunas, B. Ludwig, and F. Ricci, “Matrix factorization techniques for context aware recommendation,” in Proc. 5th ACM Conf. Recommender Syst. (RecSys), Chicago, IL, USA, 2011, pp. 301–304.

X. Ren, M. Song, and J. Song, “Context-aware probabilistic matrix factorization modeling for point-of-interest recommendation,” J. Neurocomput., vol. 104, no. 6, pp. 165–178, 2017.

J. Tang, H. Gao, X. Hu, and H. Liu, “Context-aware review helpfulness rating prediction,” in Proc. RecSys, Hong Kong, 2013, pp. 1–8.

X. Ning and G. Karypis, “SLIM: Sparse linear methods for Top-N recommender systems,” in Proc. ICDM, Vancouver, BC, Canada, Dec. 2011, pp. 497–506.

Y. Zheng, B. Mobasher, and R. Burke, “CSLIM: Contextual SLIM recommendation algorithms,” in Proc. RecSys, Los Angeles, LA, USA, 2014, pp. 301–304.

T. George and S. Merugu, “A scalable collaborative filtering framework based on co-clustering,” in Proc. ICDM, Houston, TX, USA, 2005, pp. 625–628.

D. Lemire and A. Maclachlan, “Slope one predictors for online rating-based collaborative filtering,” in Proc. SDM, Los Angeles, CA, USA, Apr. 2005, pp. 1–5.

P. Massa and P. Avesani, “Trust metrics on controversial users: Balancing between tyranny of the majority and echo chambers,” Int. J. Semantic Web Inf. Syst., vol. 3, no. 1, pp. 39–64, Jan. 2007.

M. Jamali and M. Ester, “TrustWalker: A random walk model for combining trust-based and item-based recommendation,” in Proc. KDD, Paris, France, Jun./Jul. 2009, pp. 397–406.

V. K. Sejwal and M. Abulaish, “Context-based rating prediction using collaborative filtering and linked open data,” in Proc. WIMS, Seoul, South Korea, 2019, pp. 1–9.

S. Khosro, Z. Ali, and I. Ullah, “Recommender systems: Issues, challenges, and research opportunities,” in Proc. Inf. Sci. Appl., 2016, vol. 376, no. 12, pp. 1179–1189.