IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received January 22, 2020, accepted February 4, 2020, date of publication February 20, 2020, date of current version March 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.2975167

A Pre-Filtering Approach for Incorporating Contextual Information Into Deep Learning Based Recommender Systems

ISAM MASHHOUR AL JAWARNEH[1], PAOLO BELLAVISTA[1], (Senior Member, IEEE), ANTONIO CORRADI[1], (Senior Member, IEEE), LUCA FOSCHINI[1], (Senior Member, IEEE), REBECCA MONTANARI[1], JAVIER BERROCAL[2], AND JUAN MANUEL MURILLO[2]

[1]Dipartimento di Informatica–Scienza e Ingegneria, University of Bologna, 40136 Bologna, Italy
[2]Escuela Politécnica, Universidad de Extremadura, 10003 Caceres, Spain

Corresponding author: Isam Mashhour Al Jawarneh (isam.aljawarneh3@unibo.it)

This work was supported by the Smart Architecture for Cultural Heritage in Emilia Romagna (SACHER) Project funded by the POR-FESR 2014-20 through CIRI under Grant J32I16000120009.

ABSTRACT Depending on the Internet as the main source of information regarding all aspects of our life is becoming a trend. People seek relevant information, suggestions, and recommendations in an overloaded online world and through social ties regarding their daily activities, including places to visit and restaurants to try new food. The wide variety of choices that are available online causes information overloading, which thereby complicates the selection process. Traditional recommender systems are mostly dependent on a conventional model that is based on user-item-rating interaction without considering contextual information. We claim that new generations of recommendation systems able to exploit context in an innovative and efficient way is important and may statistically yield more significant rating predictions. However, only few research works have focused on how to effectively and efficiently exploit context metadata in Deep Learning (DL)-based recommendations. The main reason lies, perhaps most significantly, in the fact that most current DL algorithms are not intrinsically designed to incorporate contextual tags. In this paper, we provide a significant contribution for filling this gap by designing a hybrid algorithm that retrofits and repurposes a pre-filtering contextual incorporation method and feeds the new dimension to a DL-based neural collaborative filtering method, thus preserving and recovering the benefits of both without their limitations. The paper also reports quantitative results that show that our method outperforms the baselines by statistically significant margins.

INDEX TERMS Deep learning, recommender systems, collaborative filtering, context awareness, apache spark.

I. INTRODUCTION

The unprecedented adoption of IoTs, coupled with advancements in sensor-enabled devices, has caused an accumulation of massive amounts of datasets that are now, more than often, coming in contextually tagged forms. Those data are typically fed into various engines for analysis, aiming at gaining deep insights in many directions. A traditional long-lived application is Recommender Systems (RSs), which aim at resolving the problem of information overloading by presenting users with personalized recommendations on items that suit their preferences [1]. E-commerce depends heavily on RSs in order to improve the item-purchase hit ratio. A non-exhaustive list includes; news to read, music to listen, items to purchase and restaurants to visit. Conventional applications of RSs focus mainly on analyzing historical user-item-rating interactions (in explicit feedback, or simply user-item in implicit counterpart) with no awareness for contexts surrounding each feedback decision.

Comparatively speaking, despite the tremendous size of research activities in the literature focusing on RSs in various domains, yet the share of their application with

The associate editor coordinating the review of this manuscript and approving it for publication was Alba Amato.

context-awareness remains humble. In traditional recommenders, context information is fed into RSs as items and user profiles. Works of the relevant literature have focused on incorporating contextual information into conventional RSs by employing three models; prefiltering, postfiltering and modeling. However, despite the availability of sparse works that seek incorporating contextual information into DL models, the exploitation of context-based models and solutions remains limited. Also, most of them have focused on incorporating context using model incorporation methods. To the best of our knowledge, there are no works in the relevant literature that focus on incorporating contextual dimensions into DL-based RSs using pre-filtering methods such as item-splitting (discussed shortly) [2], [3]. DL is establishing itself as the new trend for data deep insightful exploration and looks set to remain that way at least for the foreseeable future. It then makes sense to investigate important steps toward bringing context-awareness into DL.

The lack of appropriate contributions that seek incorporating context-awareness into DL-based RSs has encouraged us to design a novel method for achieving such target. Incorporating user contexts is paramount because user ratings are highly dependent on contexts, where either user with different context have different preferences, or have different ratings for same items depending on a dynamic context. For example, users may rate movies differently multiple times depending on the location where they watch them (e.g., home, cinema) and who were accompanying them (e.g., friend, alone or family).

Our primary original contribution in this paper is the proposal of a hybridized algorithm that combines the benefits of a DL-based collaborative filtering algorithm, dubbed as Neural Collaborative Filtering [4] (NFC thereafter) and a retrofitted-version of item-splitting approach (the plain item-splitting is adopted from [2], [3]) for incorporating contextual information in Collaborative Filtering (CF), we dub our version as Context-Aware Neural Collaborative Filtering (CA-NCF). The novelty of our design stems from the fact that plain NCF does not have a method for incorporating contextual information (as it merely accepts only the traditional forms of either explicit-feedback represented as a user-item-rating interactions or an implicit feed-back on the user-item form), while on the other hand our method provides the appropriate incorporation. In addition, the paper provides the readers with the contribution of originally and quantitatively comparing our algorithm with plain NCF version and with state-of-art benchmarks for contextual incorporation. Taking advantage of the latest advents in big data processing, we depend on a DL framework known as BigDL [5] that is engineered atop Apache Spark [6], which makes it preferable over counterparts for the fact that it seamlessly integrates with the full-fledge stack of Spark core (because of the modular architecture of Spark), thus allowing seamlessly the fusion of other workloads as needed. We have benchmarked with three recently popularized contextually tagged datasets that are publicly available (Movielens 1M [7], DePaulMovie [8] and TripAdvisor [9]). Our results are statistically significant as we have obtained, on average, lower validation loss values compared against the baselines. Also, we have obtained higher top-one-accuracy (a.k.a. Top1Accuracy, we use those terms interchangeably hereafter) values (statistically desirable) by applying our method compared to the baselines. Those results experimentally validate the importance of explicitly incorporating contextual information into a DL-based RS.

The paper remainder is structured as follows. We first walk through a brief background, providing short primers of RSs, context-awareness and associated approaches. Most importantly, we recapitulate the baseline algorithm. Thereafter, we elaborate our CA-NCF method and associated algorithms. In the section that follows, we show and discuss our results. Related work, conclusions and recommendations for future research frontiers end the paper.

II. BACKGROUND

In this section, we outline the main concepts relevant to CARSs, and we show in a systematic way the transition from conventional solutions to latest DL-based trends.

A. RECOMMENDER SYSTEMS

RSs are loosely defined as those systems that provide recommendations to users on items of interest. Items can be '*products to purchase*', '*music to listen*' and '*restaurants to visit*', to mention just a few [1]. Human-oriented RSs are basically intended for users lacking proper experience in opting for items of interest over-the-wire, giving the overwhelming number of alternatives normally offered. A case in point is the giant movie company Netflix[1], which employs RSs to recommend top movies to customers wishing that they match their preferences and thus increasing the watch-and-purchase ratios accordingly by personalizing user's interaction experience. This means suggesting different items to various persons or groups. In a more utilitarian perspective, RSs work by calculating top ranking list of items recommended for users. Computations are based on a deep analysis of historical user-item interaction that could be modeled either explicitly as ratings (a.k.a. explicit feedback) or implicitly such as considering the time a user spends viewing a page of a specific item online. For example, a prolonged view could signify a big interest and thus considered a positive rating (for example, 1 on a binary rating scale or 4 on a scale from 1 to 5).

As-is the case for all information systems, RS has been emerged as a mimic for a traditional human behavior where people normally seek suggestions from their friends. This resemblance has been formalized in a method that signaled the birth of new breed of algorithms, the widely accepted algorithm known as Collaborative Filtering (CF) [10]. In its simplest form, it works by recommending items to users based on the ratings of their friends and similar users presuming they have same tastes, even for other items, meaning that they rate the same. RSs normally direct users to unseen items,

[1] https://www.netflix.com

collect their feedbacks and store them for more personalized future recommendations.

As a field that is born as a multidisciplinary domain dense at the intersection of various sciences, including, among others, computer science, psychology and geography, it is considered as a sub-domain of Machine Learning (ML). Having said that, most naive ML algorithms flow seamlessly to RSs, including the two distinguished sub-parts; regression and ranking. In cases where RS involves rating prediction, it can be classified under regression problems, whereas it is considered a ranking task when enclosing item recommendation. The distinction is important as rating prediction means that data provides explicit feedback, whereas, on the contrary, feedback in item recommendation tasks is implicit. In this paper, we focus on explicit feedback category [10].

Despite the variety of data sources that feed RSs, they all agree in the general format of data fed for recommendation tasks. In its general form, data is provided as triplets constituting users, items and transactions encapsulating user-item interaction (formally referred as ratings). Recently, the attention has been given to more specialized breed of recommenders that are collectively known as Context-Aware Recommendation Systems (CARS) [11], which work with datasets that embed additional modes surrounding the decision-making process (generally known as contextual information or contexts for short). For example, in a restaurant recommendation system (RRS), contexts include extra information in user profiles such as age, alcohol drinking level and income. Also, for items (restaurants in this case), they include location and whether parking services are offered. Those are considered static contexts as they slowly or rarely change over time. However, another class of contexts include dynamic contexts. For example, in a movie watching survey, users may rate movies with various scores depending on the location where they watch the same movie (home or cinema), or the time (weekday or weekend). Reference studies have shown that CARS offer more personalized and effective recommendations [12].

B. CONTEXT AWARENESS

Context is loosely defined as any associated information that is useful for characterizing the situation of an object [13]. Objects include people, locations, and any information that is relevant for modelling the binary interaction between users and items, including user profiles and their associated item list. The additional contextual information is typically beneficial for personalized recommendations entwined robustly with contexts surrounding ranking decisions. Reference [14] classified context into six categories, where three belongs to human-related information (i.e., profiles, social ties and tasks), whereas the remainder are more about surrounding environment (e.g., locations) and associated conditions (e.g., weather). For example, in RRS, context plays a pivotal role in choosing a restaurant for next meal. User profiles, such as those available directly from the database, for example their drinking and smoking habits are considered also contexts that affects the rating decision. Also, weather conditions at the time of visit, the availability of parking slots and similar information.

C. INCORPORATING CONTEXT INFORMATION INTO RECOMMENDER SYSTEMS

Incorporating contextual features in a RS normally proceeds in one of three directions; pre-filtering, post-filtering, and contextual modeling [1]. Pre-filtering methods serve as dimensionality reduction approaches that embed contextual information into users or items components. Pre-filtering approaches simply work by selecting ratings data that corresponds to a contextual condition for generating relevant recommendations. For example, if a person wants to listen to a song in her car, only 'car' music rating data is used to recommend a song. Because exact contexts normally do not have enough rating data (e.g., because of the known data sparsity problem in recommender systems), some works went beyond selecting training subsets form data that corresponds to the exact contextual values. For example, some works are based on ontologies such as the work by [15] which applies a prefiltering step that first identifies and generalizes a user context (i.e., projecting it to a higher granularity level). Afterwards, only data that corresponds to that context instance are selected for training. Thereafter, a classical RS method such as Item-kNN is applied. In the same vein, [16] presented a distributional-semantics pre-filtering approach. They adopt a similarity measurement for contextual situations that is based on the distributional semantics of their constituent conditions. Stated another way, situations are considered similar in the case that they similarly influence user's preferences. Despite being a reduction-based approach, it performs segmentation in a way that slightly differs from similar counterparts of the relevant literature. The segmentation of ratings is based on aggregating ratings with contextual situations that are similar to the target. Contextual situations are an intermix descriptions composed of multiple contextual conditions. For example, ''today is a 'weekend' and the song is 'from Sarah with love'.''. by doing that, the method avoids the limited capacity of condition-to-condition context taxonomy.

On the contrary, post-filtering approaches launch with context kept aside, thereafter a filtering procedure passes through results and discard rating predictions rendered as non-relevant to context conditions. Contextual modeling explicitly model context data and inject it within the layers of the prediction model, specifically by parametrizing hidden unit transitions as a function of contextual information. For example, [17] presented two methods for incorporating contextual information with two kinds of collaborative filtering models (i.e., memory- and model-based). The contextual information is engraved into the computational models. The memory-based model is based on fusing contexts and user ratings in a multi-feature vector. For model-based CF, their method combines user preferences and contextual features into the learning model. In this paper, we focus on

pre-filtering paradigms. The reason behind this decision is that contextual pre-filtering approaches are, design-wise, compelling and conceptually appealing approaches, as they allow to seamlessly pave the way for applying various traditional recommendation approaches untouched (i.e., 'as-is' without modifications on the underlying algorithms). This also applies to post-filtering approaches that can also be applied in combination with traditional algorithms as-is. However, this comes at the price of a computational cost that could be induced by applying a post-filtering approach, which often exceeds the pre-filtering counterparts. Also, as it is recommended by several comparative studies between the two approaches, pre-filtering is preferred to post-filtering in cases where the former is able to yield recommendation predictions better than plain traditional methods (the methods that work with contextual information kept aside) [1].

For the contextual modelling approaches, we avoid the computational complexities associated with them, as by-design they work by stacking sequentially several hidden layers in a neural network, inducing an additional computational cost. Also, contextual modelling approaches have been widely researched, and many algorithms has been already developed in the last two decades or so [1]. On the contrary, only few works have touched the pre-filtering methods, especially in the deep learning arena.

D. RECOMMENDATION APPROACHES: FROM CONVENTIONAL TO CONTEXT-AWARE

In the relevant literature, [12] identified several approaches for classical recommendation tasks. For convenience, we here recapitulate only CF and content-based, then we refer the interested user to the original paper for further details about other approaches. In content-based approaches, similarity is measured among items and new items are rated based on that. Feature contents are the base for calculating similarity. For example, if a user likes a restaurant that falls under fast-food category, the system learns to rate other restaurants from the same category. CF remains the most applied classical RS approach for its simplicity. It is basically based on measuring the similarity of preferences between an active user and others with similar rating behaviors. Recent developments in CF include extensions such as latent factor models, including Matrix Factorization (MF) (the industry de facto standard for RSs), which projects items and users to a shared latent factor space, where factors are inferred from user explicit feedback.

Shortcomings of classical RSs in discovering deep relationships surrounding actual recommendation decisions led to the emergence of a constellation of systems better known as CARS. Context-aware ranking with factorization models is becoming a trend. Context is being regarded as an indispensable component in recommender systems that can deliver more personalized recommendations, matching user preferences. In addition to the traditional user-item-rating model, CARS consider additional contextual features such as temporal (for example, time of day) and weather, thus reformulating recommendation problem as a user-item-context-rating (for explicit feedback cases). For example, in a restaurant CARS, contextual information contains user profiles with features such as user's weight, height, smoking and drinking levels, which are key ingredients for self-managed RRS context-awareness recommendations.

Introducing additional features to RSs potentially enhance the prediction task, but however brings additional layers of computational complexity into the equation. This is hardly ever an issue giving a widespread adoption of distributed and parallel computing clouds, which has motivated a swift adoption of Deep Learning (DL) in recommendation tasks. Recent works have brought DL into the RSs scene, despite originally intended for complex analysis and prediction tasks in complex domains such as image and voice recognition. We specifically focus on a recently popularized DL-based recommendation algorithm dubbed as Neural Collaborative Filtering (NCF) [4]. We have selected this representative as it proved most effective in providing more accurate personalized recommendations by employing a hybridization that benefits from both the linearity of CF and the robustness of NN.

1) NEURAL COLLABORATIVE FILTERING

Neural collaborative Filtering (NCF for short) [4] is a newly introduced algorithm that basically hybridize the benefits of Multi-Layer Perceptron (MLP) and Collaborative Filtering (CF) to learn the user–item interaction function. Matrix Factorization (MF) can be recovered under its framework. In simpler forms, NCF can be leveraged with or without MF. As this is considered a DL-based conventional approach, it does not accept explicit context features. Hence, we aim at redesigning it so that we conveniently model context features without loss of generality. To this end, we have retrofitted a new version of a pre-filtering contextual incorporation method known as item-splitting [2], [3] as a frontstage that is responsible for encapsulating relevant contextual information into model's items.

III. OUR DL-BASED CONTEXT-AWARE RECOMMENDER SYSTEM (CARS)

Fig. 1 schematically depicts the primary ideas behind our DL-based CARS general framework. Raw data from sources is fed untouched into a "feature selection" stage, where relevant CA features are selected. Afterwards, selected features are passed through a specialized "context feature incorporation" stage, where we apply item-splitting thus preparing for CA-NCF, where we map raw data into a lower-dimensional space. Thereafter, the corresponding algorithm is applied where we train and use our models. The output is a list of context-aware recommendations delivered to the user.

In particular, in the following of this work, we consider a prefiltering approach for contextual incorporation as described in the next subsections.

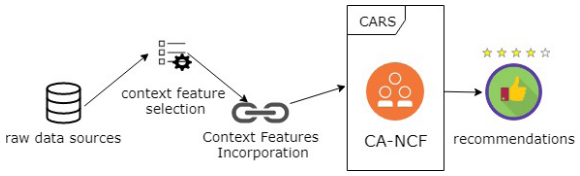

FIGURE 1. DL-based CARS framework.

A. RETROFITTING ITEM SPLITTING ALGORITHM

The plain item splitting algorithm as described in the seminal work [2], [3] has three costly nested loops and exhaustively needs to check all enumerations of context conditions for each context factor for each item in the rating matrix. The plain item splitting algorithm proceeds as follows. For each item, it generates two rating lists for each contextual condition pair combination, thereafter, it computes the two-sample t-statistics, then it finds the combination that leads to the maximum t-statistics. Afterwards, if that maximum value is greater than a threshold, it splits that item in the original list into two items based on the corresponding contextual condition that generated the maximum t-statistics.

For small data settings, the procedure described is manageable and takes a reasonable running time. However, for big datasets that contain millions of ratings and are explicitly tagged with several contextual factors, with each factor possibly having several contextual conditions, the plain procedure could turn prohibitive as it needs substantial amount of time and adds layers of complexity and overhead to the underlying baseline algorithms. For this reason, instead of exhaustively measuring the relevance of each pair of context conditions as described in the original paper of item-splitting algorithm [2], [3], we employ the Analysis of Variance (ANOVA) [18] test to further quantify the extent to which a pair of context conditions (values in contextual parlance) are superior to other permutations from the same conditions list of the same context-related feature. We follow ANOVA by a Tukey's HSD test, which is a multiple comparison statistical test that can be exploited to find contextual condition combinations with means in ratings that significantly differ from others, signifying that they positively contribute to the rating decision. In other terms, we first apply ANOVA as a quick-and-dirty sieve to extract contextual variables that significantly affect the rating decisions. Our retrofitted procedure is listed in Algorithm 1. The output of the algorithm is a list of relevant contextual features and the associated permutations of relevant contextual conditions that can be used for splitting.

For example, say we have a rating matrix with the following contextual factors: 'day of week', 'year', 'day of month' and 'month of year'. Each of those factors may have several conditions. For example, 'day of week' has the following conditions list; (weekday, weekend). Say the rating matrix has millions of ratings, if the plain item splitting procedure is applied as-is, it would take hours to perform the t-statistics that is needed as a core element of the procedure. This is because it needs to check every combination of contextual conditions for each item, which is an expensive exhaustive scan. On the other side, our retrofitted version checks only the combinations of contextual conditions that show statistical significance through the application of ANOVA. By applying our retrofitted version of item splitting, we significantly prune the search space in such a way that boosts the time-based quality performance of the overall system. Our approach resorts to a two-stages procedure in which, first, we filter the conditions space and, then, we refine the obtained results. The filtering stage (where we apply ANOVA, which is a greedy approach) results in a sub-list that is partially selected from the total contextual conditions space. In the second stage (i.e., refinement), we apply the impurity test [19] (i.e., t-test) as described in the seminal item-splitting work [2], [3] for the partial list that resulted from the filtering stage. The result of the impurity test is a list of contextual conditions that can be used for splitting items.

Stated another way, our retrofitted item splitting procedure operates as follows; First, for every relevant context feature we apply the Analysis of Variance (ANOVA), thereby extracting the F-value and p-value statistics. We depend on profiling the dataset for selecting the appropriate p-value. However, for most datasets p-value less than or equal 0.05 is a scientifically desirable margin. We select a feature with the

Algorithm 1 Retrofitted item-splitting with multi-samples paired test

```
Procedure retrofitted_itemSplitting (contextFeatures, threshold)
maxf ← MIN_VALUE
relevant_features_List ← {}
relevant_pairs_List ← {}
while cf ∈ contextFeatures!= NIL do
F-value, P-value = statistics (aov(cf)). get (F-value, P-value) //ANOVA
    If F-value >= maxf && P-value <= threshold// 0.05 is a scientifically recommended margin
    Maxf ← F-value
    relevant_features_List.add(cf)
    End if
End while
minp ← MIN_VALUE
Foreach relevant_feature in relevant_features_List
    Pairs = TukeyHSD (aov (relevant_feature)) //aov: analysis of variance
    For pair p in pairs
    If P-adj < threshold
     minp ← P-adj
     relevant-pair ← pair
     relevant_pairs_List.add(relevant-pair)
   End if
End for
Item-splitting(relevant_pairs_List)
End procedure
```

more-the-better F-value pattern (F statistic shows the joint effect of contextual features altogether), and to evaluate pair means, we further conduct a Tukey post hoc test, where p-values greater than a threshold (e.g., 0.05 is statistically desirable) indicate that there is no significant difference while values less than the threshold signify the opposite. The selected pairs are considered the base for item-splitting and thus we feed them to the plain item-splitting sub-procedure, which proceeds as exactly as described in the original paper [2], [3]. Running this procedure per se constitutes a fundamental part of our contributions in this paper. Referring to the previously mentioned contextual features example, our filtering stage may select only two contextual features from the contextual features space. For example, say that it selects 'day of week' and 'year'. Thereafter, the Tukey's HSD will show the most significant contextual conditions pairs from all the possible combinations of permutations in the 'day of week' and 'year' features. Suppose that the Tukey's HSD shows that the following pairs are significant: (weekend, weekday) for 'day of week' and ('2018', 'not 2018') for 'year'. Those show p-values less than 0.05. Those will then constitute the partial list of potential (yet probabilistic) contextual combinations that will be passed over to the refinement stage. In the refinement stage, we apply the t-test on each unique item from the ratings space. For each item, we split the ratings into two rating lists based on each of the combinations in the partial contextual pairs and apply the t-test. We then take the maximum t-test for each item and if it was greater than the threshold (e.g., greater than 4), we split the item based on the contextual pair combination that resulted in such a t-statistic (for example <'weekday', 'weekend'>).

B. CONTEXT-AWARE NCF (CA-NCF): PRE-FILTERING APPROACH FOR CONTEXTUAL INCORPORATION

Our novel algorithm is a hybridization between a retrofitted version of the item-splitting contextual incorporation paradigm and an adapted explicit feedback version of NCF, thus combining the benefits of both in a way that enables them to mutually reinforce each other without loss of generality. Dub our version as CA-NCF[2] (short for Context Aware Neural Collaborative Filtering). Item-splitting in this setting is considered as a frontstage that is intended for seamlessly capturing and incorporating context information into the item components of the NCF without changing the core of the plain NCF algorithm. The crux of this design is that it injects context-awareness in a manner that preserves (to a good extent) the running time of the underlying NFC engine, but at the same time benefiting from the planted context features in generating more accurate context-aware recommendations. Fig. 2 depicts the workflow of our algorithm.

We have added a stage before 'sparse item vectors', where we perform item-splitting, thus embedding explicit context-aware features to better capture the contextual interaction

[2]The source code of CA-NCF (including the retrofitted item-splitting algorithm) is available at: https://github.com/IsamAljawarneh/CA-NCF

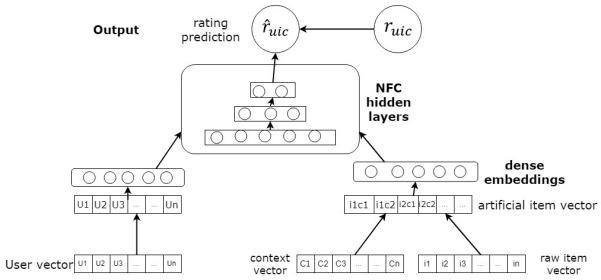

FIGURE 2. Context-aware neural collaborative filtering (CA-NCF) by retrofitting an explicit feedback NCF.

between users and items. We first restored NCF to work with explicit-feedback data representations. We then incorporate selected context features into NCF prediction model at no cost, since we are constructing a new artificial item, thus recovering the same pattern of the model. We thus redefine the prediction model as in (1).

$$\hat{r}_{uic} = f(P^T V_u^U, Q_{art}^T V_{ic}^{IC} | P, Q_{art}, \Theta_f) \tag{1}$$

We have simply replaced V_i^I in the plain NCF algorithm with a new item sparse vector V_{ic}^{IC} that incorporates context conditions (a step through a retrofitted item-splitting procedure as described in § III. A). The new setting thus constitutes a bottom-most layer encompassing two feature vectors V_u^U and V_{ic}^{IC} that features user u and an artificial item i (embedding item i and context condition c), respectively. We use content features for representing users, items and contexts since our adaptation of NCF works for explicit feedback cases.

$\hat{r}_{uic}$ is the rating of a user for an item under a specific context condition. Θ are model parameters to be estimated. $P \in \mathbb{R}^{M \times K}$ and $Q_{art} \in \mathbb{R}^{N \times K}$ are latent factor matrices for users and artificial (i.e., fictious) items, respectively. Θ_f is a set of model parameters for the interaction function f, which then can be reformulated as in (2)

$$f\left(P^T V_u^U, Q_{art}^T V_{ic}^{IC}\right) = \phi_{output}(\phi_n(\ldots\ldots(\phi_1(P^T V_u^U, Q_{art}^T V_{ic}^{IC}))\ldots..)) \tag{2}$$

where ϕ_{output} is a mapping function employed on the output layer, and ϕ_n is the nth mapping function for a NCF layer.

Stated another way, our newly introduced 3-dimensional prediction function can be formulated with a 2-dimensional function such as in (3), adapted from [1].

$$\mathcal{R}^{RL}_{U \times I \times C}(u, i, t) = \mathcal{R}^{[c \epsilon cc](U,I,R)}_{U \times I}(u, i) \tag{3}$$

where RL is the complete rating list, containing records on the < user, item, context, rating > form. 'cc' is a context condition (e.g., 'weekday'), $[c \epsilon cc]$ is the contextual pre-filter. $[c \epsilon cc](U, I, R)$ is the subset selected from the RL list based on the $[c \epsilon cc]$ pre-filter. This subset is then projected into a two-dimensional space by only selecting the user (U) and item (I) dimensions. For example, a prediction of the form $\mathcal{R}^{RL}_{UxIxC}(Sarah, 'from\ sarah\ withlove', weekend)$ will be

transformed into $\mathcal{R}_{U\times I}^{[c\epsilon weekend](U,I,R)}(Sarah,'from\ sarah\ with\ love')$ in order to predict the rating for the *'from sarah with love'* song by user 'Sarah' in a 'weekend' day (contextual condition), including 'Saturday' and 'Sunday'.

Conceptualizing our context-aware incorporation this way allows us to easily recover NCF, thus ensuring that the robustness of NCF mutually reinforce our embedded retrofitted-version item-splitting in a way that provides a statistically significant context-aware prediction rating without loss of generality of NCF. Another advantage is that with this approach, the computational complexity of our algorithm is reduced to that of only performing NCF on a two-dimensional space (since the newly artificial number of items is equal to a maximum of $(n + m)$, where m is the number of split items), which is numerically preferred, but, at the same time gaining the privileges provided by seamlessly incorporating context-related features into the play. We show subsequently that we obtain better predictive performance than would be gained by independently employing each of those models to same settings.

We argue that our methodology has good potential for streaming DL jobs as it is computationally less expensive than counterparts that potentially stack more convolutional layers for the purpose of enhancing predictions with contextual tags.

IV. EXPERIMENTAL RESULTS AND DISCUSSION

In this section, we discuss experimental settings, datasets we rely on, and proposed methodologies for performance comparison. In addition, the section describes the baselines we compare our original solutions with, parameter configurations and coding. Afterwards, we report the experimental results measured and an extensive discussion about the motivations and lessons learned from these results.

A. EXPERIMENTAL SETTINGS

1) DATASETS

We depend on three public datasets of diverse sizes for experimentation; a big dataset, Movielens 1M [7], in addition to a small size dataset, DePaulMovie [8] and a medium-sized dataset, TripAdvisor [9]. By this diversity of combination, our intention is to prove the robustness of our algorithms under various real scenarios. The statistics of the datasets are as follows; Movielens dataset that we select contains around one million ratings by almost 6040 users for around 3706 movies (i.e., items). For TripAdvisor, it was crawled from online reviews on TripAdvisor website. there is only one context, trip type (values include Family, Couples, Business, Solo travel, Friends). The dataset is sparse in both ratings and contexts, comprising 14175 ratings, 2731 users and 2269 hotels. DePaulMovie was collected through surveys, where students are requested to rate movies in various time, location, and with different companions, which all are considered relevant contexts in this setting. Comprising 5035 ratings, 97 users, 79 items, and data density 17.48%. We specifically choose those datasets as they are well-established representatives for contextually tagged explicit-feedback recommendation data. Also, they have been used extensively by the creators for comparing state-of-art CARS algorithms, considering that contextually tagged datasets for explicit-feedback RSs is scarce and rare. MovieLens dataset does not contain contextual features explicitly. However, it contains a *timestamp* for each captured rating, which then is considered an implicit source of contextual information [1]. Simply put, we have flattened the timestamp into its constituent granular parts, which then constitute relevant contextual factors. Specifically, we have extracted the following context factors from each timestamp: 'day of month', 'day of week', 'month', 'year', 'day of year' and 'day of month'. For example, the original contextual conditions list for the 'day of week' contains all the seven days from 'Monday' to 'Sunday'. This potentially increases the data sparsity, which is a common problem in recommender systems research [1]. Sparsity can be loosely defined as the data distribution for which not enough ratings data is available for accurate rating predictions. To avoid this kind of sparsity, we have transformed such data into pairs. For example, the 'day of week' contextual conditions list is transformed into the ('weekend', 'weekday') pair, where 'weekend' implicitly contains ('Saturday', 'Sunday'), whereas 'weekday' contains implicitly all the other five days. Stated another way, the ratings data may not have enough ratings about the historical movie watching events for a user in a 'Sunday'. Taking the exact context conditions this way may expand the sparsity gap in the dataset, rendering the process inappropriate for context-aware recommendations. On the other hand, we reap many benefits by binarizing the contextual conditions so that each context factor's conditions are split in two groups.

In Movielens 1M dataset, to account for sparsity, we have dropped items that appear in less than 100 rating decisions. We also have dropped users with less than 20 ratings. The resulted dataset size equals around 940k. For DePaulMovie and TripAdvisor datasets we drop items with less than 5 ratings, resulting in around 5018 and 10414 ratings for DePaulMovies and TripAdvisor, respectively.

2) EVALUATION METHODS

To evaluate rating performances, thereby comparing our method with the baselines, we have conducted a train-testing evaluation (with 10 epochs training and 80% training dataset). We have used sparse-categorical-crossentropy as our loss function. The rationale for choosing sparse-categorical-crossentropy is to avoid the one-hot encoding imposed by using the plain categorical-crossentropy, as the latter requires one-hot encoding the target variables in multiclass classification problems (explicit feedback recommenders belongs to the multiclass classification family), thus converting them into categorical formats, whereas the former accepts the multiclass values as-is. Since we have multiclass values in the target field (i.e., ratings, which span from 1 to 5) then we have opted for sparse-categorical-crossentropy. Also, we have used Adam and SGD as the optimizers. Thereafter, we have captured the average Mean Absolute Error

(MAE) and validation loss as loss (error) metrics, which are widely accepted measurement in relevant literature. MAE and validation loss are decline-oriented scoring mechanisms for which values follow lower-better trend. We have redefined MAE by incorporating contextual features into the equation. The retrofitted MAE is calculated using (4)

$$MAE = {}^{1}/_{|\mathcal{R}_{valid}|} \sum_{r_{uic} \in \mathcal{R}_{valid}} |\hat{r}_{uic} - r_{uic}| \qquad (4)$$

where $\mathcal{R}_{valid}$ is the validation set, $\hat{r}_{uic}$ is the function that we want to learn for predicting the rating of a user u to an item i given the contextual condition c. The r_{uic} is the target rating.

MAE is the most adopted error metric for rating-oriented recommenders, and it is preferred to accuracy metrics (e.g., recall and precision) in cases where we have an access to explicit feedback systems such as those that we focus on in this paper [1], [20].

However, since we are also interested in measuring the quality of recommendations, we have also applied an accuracy measure. Specifically, a rank-aware metric for top-N rating predictions (a.k.a. precision-in-top-N). We have specifically applied the top-one-accuracy (a.k.a. P@1, read 'precision at 1', shorthand for precision at cutoff 1), which belongs to the class of Precision@k accuracy metrics [1]. The rationale for selecting this metric is to evaluate the Top-N recommendations and in this case quantifying the times that the top element in the Top-N list matches the target. Stated another way, P@1 is calculated by only considering the first rank in our recommendation list for each user. p@1 is an incline-oriented scoring system where the higher-better applies. Since we are incorporating contextual information into the recommendation model, the precision is calculated as the ratio of the correct top-1 ranks to the number of items suggested given a specific contextual condition. A further reason that rationales the selection of such an accuracy measure is that in real scenarios, most users are interested in checking the highest ranked items. Also, in our case scenarios, the counts of recommendations shown to user by the application are preordained, rendering precision-oriented measures (such as P@1) suitable [1].

To obtain good fit learning curves (a.k.a. convergence) that falls between an overfit and underfit model, in order to generalize the model, we have repeated the experiments 100 times, where we evaluate the same model on the same data many times and only vary the seed for the random number generator, then we calculate the mean of the estimated model skill (Loss, MAE and Top1Accuracy). We did this because DL methods are stochastic (i.e., they learn via a stochastic training algorithm), meaning that we get a different diagnostic plot each run. Hidden layers exploit ReLU as an activation function, whereas the output layer depends on softmax as an activation function.

To compare our retrofitted item-splitting procedure (refer to section *III. A.* for further information) with the plain item-splitting algorithm, we have measured the running time by varying the data size.

Baselines: To show how our method (CA-NCF) excels in achieving statistically plausible results, we choose a bunch of standard benchmark methods from the relevant state-of-art. Specifically, we choose three categories as baselines to compare with. Those categories are the following:

Category#1: Plain pre-filtering contextual incorporation and DL-based recommenders. Most importantly, we compare our algorithm with the baselines that in hybridization form the constituent parts of our algorithm. Since our algorithm (CA-NCF) is a hybridization between the plain prefiltering item-splitting contextual incorporation approach [2], [3] and the vanilla NCF [4], we first compare with the following:

- Plain baseline (NCF) [4], specifically an explicit feedback version, that constitutes the latest trending and benchmark DL-based algorithms for recommender systems.
- Item-splitting Item KNN [2], which constitutes the traditional application of item-splitting as it first appeared. An adapted version of the item-based CF, which injects contextual information into the item component, thus recovering classical CF-based recommendation method by turning the User-Item-Context Rating (UICR) representation into a User-Item-Rating (UIR) classical representation and thereafter feeding it to item KNN.

Category#2: A benchmark baseline that does not consider contextual information, which is the following:

- A biased matrix factorization method [21], which is a standard benchmark context-free (i.e., does not consider contextual information) method.

Category#3: A benchmark that uses contextual modelling for incorporating contextual data. That is the following:

- Context-Aware Matrix Factorization (CAMF) [22], which is a method for incorporating contextual-information into MF. It introduces new parameters based on contextual conditions (a.k.a., values of a contextual features), such that each item (or user)/context-condition pair has a distinct parameter (which significantly increases parameter space) and improves the prediction accuracy in cases where contextual conditions influence user-item interaction. It presumes that the contextual rating deviation is dependent on items.

We have selected CAMF, item-splitting on item KNN as they are the standard benchmarks for CARS research, utilizing both contextual incorporation methods; contextual-modeling and pre-filtering, respectively [23].

3) HYPER-PARAMETER CONFIGURATIONS

We have tuned our algorithms at various levels, CA-NCF models are learnt based on log loss optimized by NCF, since our retrofitted version of item-splitting is a pre-filtering contextual incorporation method it does not affect the working mechanism of the underlying algorithm. We use Stochastic Gradient Decent (SGD) and Adam as optimizers. We have tested the batch sizes of 64, 128 and 256 (for DePaulMovies and TripAdvisor datasets), whereas we use 8k and 16k for

MovieLens 1M dataset. We set learning rates ranging from 0.01 to 0.0001 and 0.05 to 0.0005. MLP operate on two hidden layers [20], [10], and embedding size is 20. NeuMF was excluded, thus recovering the raw NCF. Other hyper-parameters include epoch which is set to maximum 10. One epoch is when an entire dataset is passed once through a NN. We use impurity criteria [19] to decide whether to split an item. We have selected T_{mean} as a measure to estimate the statistical significance (i.e., using two-sample t-test for identifying the significance of differences of the means of ratings in the two ratings subsets divided based on a contextual condition, where bigger-better applies) between means of pairs of rating lists, where each list corresponds to a contextual condition. We have selected t-statistic threshold as the following: 4 for MovieLens and TripAdvisor datasets, 2.5 for DePaulMovies dataset. The threshold value 4 is almost equivalent to the 0.05 level of statistical significance (a.k.a. p-value), which is statistically plausible.

Notice that we perform the impurity test on a subset of contextual conditions. Those are the contextual conditions that we have selected through the filtering stage (recall our filter-and-refine approach from section *III. A.*).

4) IMPLEMENTATION INSIGHTS

We have conducted our feature engineering using *R* language for the broad range of tools it provides. We also use *R* language for conducting the prefiltering stage, where we exploit the libraries that offer ANOVA and Tukey's HSD tests. On the other side of the work, our algorithm has been implemented as patches using Python language on Spark based on the BigDL framework [5]. We have introduced our retrofitted-version of item-splitting to the BigDL framework by using functions from the Pandas library on Python. The rationale behind this selection of BigDL is that for Spark, principal functions come standard with the codebase distribution, but DL functions necessitate additional libraries. However, BigDL preserves the robustness and generality of Spark as its patches compile down to Sparks core abstraction (known as RDD [24]). Also, we find that by using Python for most of our application tasks, and porting some to R for feature selection (not Java or Scala) we take a performance hit, which is for DL jobs lesser when using Java or Scala. We aim at leveraging the best from both worlds, taking advantage from R's and Spark's BigDL strengths without risks.

B. CONTEXT-AWARE NEURAL COLLABORATIVE FILTERING (CA-NCF) TEST RESULTS

In this subsection, we delineate the results that we have obtained in a coherent and systematic way. We follow the same procedural steps as described in section III. Starting from the application of our retrofitted-version of item-splitting, where we compare its running time with the baseline plain item-splitting. Fig. 3 shows that the retrofitted- item-splitting significantly outperforms plain- item-splitting in the language of running time. As the figure shows, plain- item-splitting requires a substantial running time that significantly

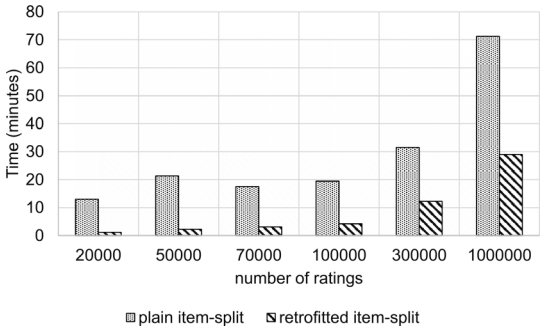

FIGURE 3. Running time of plain-item-split against retrofitted-item-split on MovieLens 1M dataset.

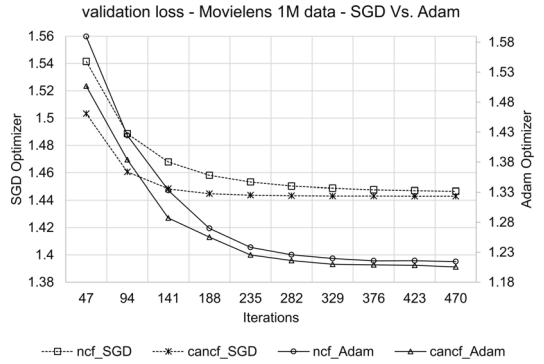

FIGURE 4. Validation loss of CA-NCF Vs. NCF (using SGD and Adam optimizers) w.r.t. the number of iterations on MovieLens 1M dataset.

increases as the data size increase. Those numbers have been captured with MovieLens 1M dataset [7], which contains moderate number of contextual features. Those number are slated to be doubled for larger number of contextual information, negatively affecting the overall performance of the system. On average, we obtain a running time decrease gain of roughly 70% by applying our retrofitted- item-splitting procedure instead of the plain- item-splitting.

Now we focus on the validation learning curves to capture the way both models (our CA-NCF and plain NCF) are generalizing. We show those curves using different settings for our CA-NCF and the plain NCF by executing them both on top of the same software stack (i.e., BigDL) [5].

We have tested our CA-NCF relying on three publicly available open source datasets (refer to section *VI. A.* for information). Fig. 4 shows the validation loss using SGD and Adam optimizers, averaged from 100 runs. We obtain, on average, 1.85 % loss gain by applying CA-NCF comparing to the seminal baseline NCF, with Adam optimization method applied. The same trend almost occurs by applying SGD (despite obtaining less optimization compared to Adam, being on average 0.91 %, however still outperforming plain NCF).

The advantage of SGD however is that the convergence happens earlier. This particularly supports findings from relevant literature that adaptive optimization techniques (such as Adam) generalize poorly compared to SGD [25].

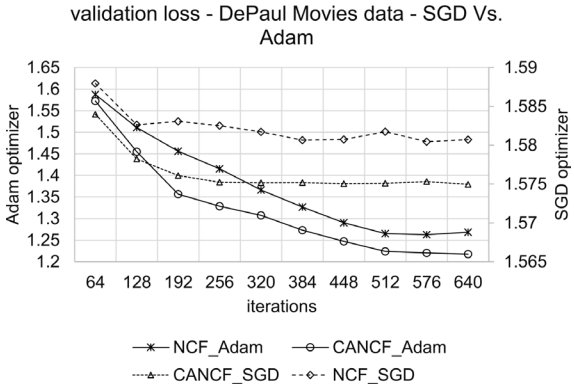

FIGURE 5. Validation loss of CA-NCF Vs. NCF (using SGD and Adam optimizers) w.r.t. the number of iterations on DePaulMovies dataset.

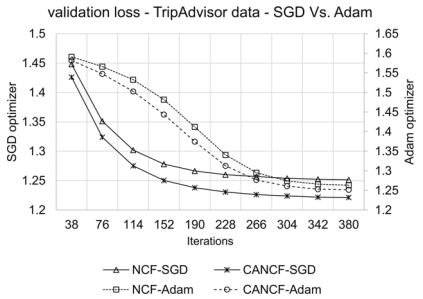

FIGURE 6. Validation loss of CA-NCF Vs. NCF (using SGD and Adam optimizers) w.r.t. the number of iterations on TripAdvisor dataset.

The biggest shift occurs after 140 iterations for SGD, and after 240 for Adam, that is when CA-NCF and NCF stabilize. We also note that Adam needs more iterations to stabilize, showing, however, ultimately similar trend. Those results are obtained by choosing a batch size that is equal to 16k.

Similar trends occur for DepaulMovies and TripAdvisor datasets as shown in figures 5 and 6, respectively. We have obtained 0.37% loss gain for SGD and 3.98% for Adam on average for the DepaulMovies datasets. Also, we have obtained 1.6% and 2.16% on Adam and SGD, respectively, for the TripAdvisor dataset.

It worth mentioning that all results show similar trends comparing SGD to Adam, as they both converge earlier using SGD. We also find that for both settings, SGD requires a learning rate that is higher than Adam counterpart (by a scale of 100) in order to obtain a good fit (not over- or under-fitting), ranging from 0.0005 to 0.05 and 0.0001 to 0.01, respectively. All in all, our version CA-NCF significantly outperforms the plain NCF for the validation loss skill (remind that we use the sparse-categorical-crossentropy as a validation loss function). We note that extra iterations (smaller batch size) has a property that allows obtaining more loss gain (statistically plausible).

It is well established that a good fit model is the one performing good on the train and validation sets. This can be diagnosed from a plot where the train and validation loss decrease and stabilize around the same point. An example running session from our case is shown

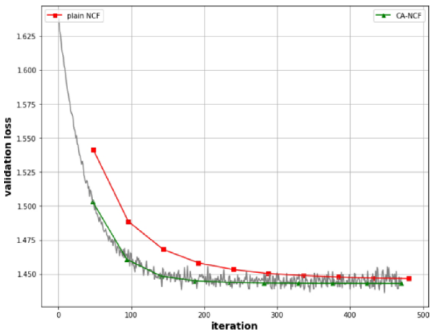

FIGURE 7. Good fit of CA-NCF (validation and train losses converge at almost the same point) on MovieLens datasets.

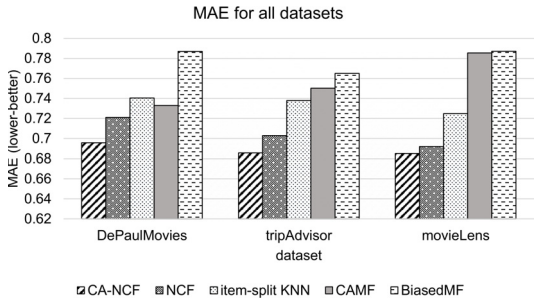

FIGURE 8. MAE of CA-NCF against counterparts for all benchmark datasets.

in fig. 7, where the shading area constitutes the train loss. We note that the convergence applies to both NCF and CA-NCF, with CA-NCF outperforming.

To conclude this test, our CA-NCF model is able to generalize better than the vanilla NCF as it is shown from figures 4,5,6 and 7.

In another accuracy test, we aim at comparing the accuracy of our CA-NCF model with other benchmark baselines (refer to section IV. A. for more information). For this, we have selected MAE. Fig. 8 shows that our model CA-NCF outperforms baselines with statistically significant margins. Our model CA-NCF registers a gain that is roughly equals to 2.35 %, on average. Those figures are doubled when comparing CA-NCF with the plain item-split KNN, where we obtain roughly a gain that is equal to 6.2%, on average, whereas they are quadrupled when comparing CA-NCF to CAMF, where we obtain roughly 8.9% gain, on average. In addition, we obtain a significant 11.66 % gain over the context-free biasedMF.

It is worth mentioning though that the plain NCF performs better than the other classical benchmarks (such as item-split KNN and CAMF). From all the methods that have been compared, context-free biasedMF performs the worst. This is expected as it does not consider contextual features. Those results support that explicitly considering contextual information yields more accurate recommendations.

Another important learning curve that shows how well a model generalizes is the performance learning curve, Top-one-accuracy, which shows an opposite trend (statistically attractive), where more accuracy is better. As shown

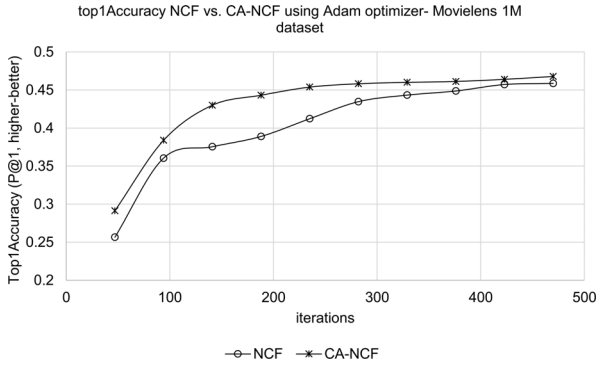

FIGURE 9. Top1Accuracy performance learning curve for CA-NCF against plain NCF on MovieLens 1M dataset.

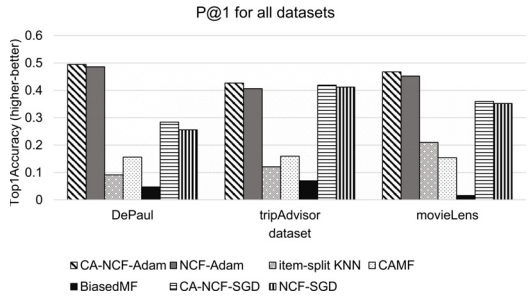

FIGURE 10. Top1Accuracy performance learning curve for CA-NCF against plain NCF and the state-of-art benchmarks on all datasets.

in fig. 9, by applying the Adam to MovieLens dataset, we obtain, on average, (for 100 running sessions) roughly 6.4 % top-one-accuracy gain. Our model generalizes better than the plain model as the figure shows. This empirically confirms the effectiveness and productiveness of explicitly incorporating contextual information into explicit-feedback NCF layers. As a general trend, validation loss of CA-NCF models decline while predictivity performance enhance. However, adding more iterations does not seem to improve the accuracy above a specific margin and both models stabilize at almost the same iterations.

To further quantify the significance of our model, we have compared CA-NCF and NCF with several state-of-art benchmarks (as described in Section VI.A. for information). Fig. 10 shows that our model performs the best in terms of top-one-accuracy, where we obtain roughly 3.25% and 4% accuracy gain by applying CA-NCF instead of the plain NCF, on average, by applying Adam and SGD, respectively. Even though NCF underperforms our model, it outperforms the state-of-art benchmark systems by statistically significant magnitudes. We obtain roughly 60%, 55%, and 87% accuracy gain, on average, by applying our model CA-NCF against item-split KNN, CAMF and biasedMF, respectively. It worth noticing that even though we have obtained less gain by applying Adam instead of SGD and comparing CA-NCF to NCF, we obtain higher accuracy gain when applying Adam instead of SGD.

All the results shown in this section provide a compelling argument on the level of improvement of CA-NCF over NCF. CA-NCF noticeably improves the recommendations quality for datasets that are explicitly (or implicitly) tagged with relevant contextual features. Traditionally, those features have been treated as additional information that are embedded into hidden layers of sequentially stacked neural layers. By explicitly incorporating context into a DL-based recommender system, we reap many benefits that significantly enhance the overall prediction quality of the recommender system.

Also, since we are applying a prefiltering stage that selectively prunes the contextual features (and thereby conditions) search space, the results show similar trends for a variety of datasets despite the variance in data sizes and the number of contextual features. This provides a glimpse that the important thing to consider is the presence of relevant contextual features in the datasets either explicitly or implicitly (e.g. a 'timestamp'), not accounting significantly for the number of those features. This in part is due to the fact that we apply an impurity check as an integral part of the prefiltering stage, where we only select statistically significant features and their associated conditions. However, we argue that completeness of contextual information may not influence the quality of recommendation in the same way the underlying procedure does. In addition, this may need further future investigation with varying sets of datasets that have dissimilar statistical distributions (i.e., being normally distributed, skewed or highly skewed). This however falls outside the scope of this paper.

V. RELATED WORK

Very few works in the relevant literature explicitly incorporate contextual information into DL-based recommender systems. Most importantly, we mention few recent works that are based on Recurrent Neural Networks (RNN). Reference [26] propose to extract static user-side contexts and model a high-order interaction with a previous item using a product-based neural network (PNN), a Neural Network (NN) variant of factorization machine (FM), in order to augment an existing RNN model, a method that is dubbed as Augmented RNN (ARNN), thereafter, they compare that to a baseline untouched RNN. ARNN achieves favorable results over plain RNN, the drawback, however, is that it is only applicable for static user contexts such as age, country and job. Hence, it does not consider the dynamicity of real-life context application scenarios, where user explicit feedback may change depending on a dynamic surrounding context.

In the same vein, aiming at mitigating the deficiency of RNN in modeling contextual information, [27] propose a model termed as Context-Aware Recurrent Neural Networks (CA-RNN) that operates adaptive context-specific input matrices to model contextual information into and end-to-end network, by incorporating contexts into the reference RNN model through the items matrix. Time intervals (between adjacent behaviors) have been treated as transition contexts and incorporated into the model. The design is however driven by the fusion of sequential and contextual analysis in one framework.

Aiming at improving next item prediction, [28] introduce a Contextual RNN (CRNN) algorithm by leveraging contextual information for the item sequence modeling. They basically achieve this by either incorporating item representation with context via several non-linear transformations (which is considered a prefiltering) or context-dependent dynamics modeling, where context is used to parametrize the dynamics of the hidden state transition. This algorithm, most importantly, has an input module that creates a dense input embedding from sparse original input data, thus bringing context information into item representation, mimicking the prefiltering incorporation approach. Then the final combined representation is passed to the recurrent update module. Concatenation assumes no effect of context on item representation.

It worth mentioning that contextual pre-filtering is a contextual incorporation approach that normally precedes the recommender model in order to get more accurate values. Hence, there are only few contextual pre-filtering incorporation approaches in the relevant literature. Some approaches are based on dimensionality reduction. For example, [29] incorporates contextual information into RSs using a multidimensional approach. However, the proposed method considers the exact values of contextual conditions, which then increases the data sparsity, negatively affecting the quality of the RS. Accounting for sparsity, the same authors suggested a generalized prefiltering approach, where they use a generalized filter for context conditions instead of the exact values. For example, if we want to recommend a song to a person based on the 'day of week', we would use not only ratings that correspond to 'Saturday' for predicting the song she may be interested in listening to in a 'Saturday', but also we use the ratings from 'Sunday' as both fall into the 'weekend' pairwise context condition split. A variant for the item-splitting approach is the user-splitting (a.k.a. micro-profiling), which acts as exactly as the item-splitting but instead of splitting items based on statistical significance of differences in rating means, it splits user profiles based on the same statistics into the so-called micro-profiles so that each user has multiple profiles with each profile corresponds to a specific contextual condition [30].

Despite significant, approaches of the relevant literature act by either capturing complex transitions of item preferences through time intervals (modeling them as a transition context) or modelling contextual information into the layers of the NN. We contend, however, that simpler methods for incorporating contexts are more time-efficient while yielding statistically significant results. Also, their incorporation into end-to-end DL-based RSs is simpler, require less hyperparameter tuning and does not require the adaptation while jumping through domains.

VI. CONCLUDING REMARKS, CHALLENGES, AND OPEN ISSUES

Information overloading is currently challenging state-of-the-art pervasive solutions in their ability to provide customized recommendations based on user preferences. Traditionally, explicit feedback systems consider user-item-rating interaction patterns, by neglecting important contextual information. It has been extensively proven recently that contextually enriched models yield more statistically significant results and personalized recommendations. We argue that discarding contextual information while performing recommendations has associated dangers and subtleties. The introduction of contextual modeling techniques mostly dominated by MF-based methods has encouraged practitioners in all domains to accept contextual metadata as an integral part of any successful RS. We have found that, to the best of our knowledge, no DL-based RS is incorporating context using pre-filtering methods such as item splitting.

In this paper, we have designed a DL-based RS method as hybridization between the recently popularized NCF and an important contextual incorporation method known as item-splitting. The hybridization benefits from both worlds without their limitations. State-of-art DL-based incorporations involve stacking many convolutional layers, which is computationally expensive, complicates the model and makes it susceptible to over and under-fitting problems. These issues are exacerbated by the fact that those settings require heavy hyperparameter tuning efforts. We instead opted for encoding context cues explicitly through a prefiltering incorporation method that simply acts as a frontstage which feeds its output to the network, thus simplifying the model and making it easier. By incorporating contexts, CA-NCF makes more personalized recommendations than the plain NCF, which does not intrinsically consider contexts. Limitations of beefed-up centralized server-based computational models have motivated us to employ a distributed robust framework atop Apache Spark (specifically relying on libraries provided by BigDL), aiming at combining our algorithms with other jobs that potentially contain burst streaming workloads.

As a future perspective, we encourage engaging multi-criteria ratings during prediction, which is loosely defined as the correlation between the overall rating and other fine-grained ratings. Moreover, the only computational cost associated with our framework lies behind feature engineering and selection aspects, which works just fine for data-at-rest. However, we plan to optimize in a future work so that we prepare our algorithms for data-in-flight (online) dynamic settings. We are starting by analyzing how our models in their current state perform in the wild (aggressive online dynamic settings with burst streaming workloads).

REFERENCES

[1] F. Ricci, L. Rokach, and B. Shapira, "Introduction to recommender systems handbook," in *Recommender Systems Handbook*. Berlin, Germany: Springer, 2011, pp. 1–35.
[2] L. Baltrunas and F. Ricci, "Context-based splitting of item ratings in collaborative filtering," in *Proc. 3rd ACM Conf. Recommender Syst. (RecSys)*, 2009, pp. 245–248.
[3] L. Baltrunas and F. Ricci, "Experimental evaluation of context-dependent collaborative filtering using item splitting," *User Model. User-Adapted Interact.*, vol. 24, nos. 1–2, pp. 7–34, Feb. 2013.
[4] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T. Chua, "Neural collaborative filtering," in *Proc. 26th Int. Conf. World Wide Web*, 2017, pp. 173–182.

[5] J. Dai, Y. Wang, X. Qiu, D. Ding, Y. Zhang, Y. Wang, X. Jia, C. Zhang, Y. Wan, Z. Li, J. Wang, S. Huang, Z. Wu, Y. Wang, Y. Yang, B. She, D. Shi, Q. Lu, K. Huang, and G. Song, "BigDL: A distributed deep learning framework for big data," 2018, *arXiv:1804.05839*. [Online]. Available: http://arxiv.org/abs/1804.05839

[6] M. Zaharia, M. Chowdhury, M. J. Franklin, S. Shenker, and I. Stoica, "Spark: Cluster computing with working sets," *HotCloud*, vol. 10, nos. 10–10, p. 95, Jun. 2010.

[7] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interact. Intell. Syst.*, vol. 5, no. 4, pp. 1–19, Dec. 2015.

[8] Y. Zheng, B. Mobasher, and R. Burke, "CARSKit: A java-based context-aware recommendation engine," in *Proc. IEEE Int. Conf. Data Mining Workshop (ICDMW)*, Nov. 2015, pp. 1668–1671.

[9] Y. Zheng, B. Mobasher, and R. Burke, "Context recommendation using multi-label classification," in *Proc. IEEE/WIC/ACM Int. Joint Conferences Web Intell. (WI) Intell. Agent Technol. (IAT)*, vol. 2, Aug. 2014, pp. 288–295.

[10] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, "Collaborative filtering recommender systems," in *The Adaptive Web*. Berlin, Germany: Springer, 2007, pp. 291–324.

[11] Y. Zheng, "Interpreting contextual effects by contextual modeling in recommender systems," 2017, *arXiv:1710.08516*. [Online]. Available: http://arxiv.org/abs/1710.08516

[12] Y. Zheng, B. Mobasher, and R. D. Burke, "Incorporating context correlation into context-aware matrix factorization," in *Proc. CPCR+ ITWP IJCAI*, 2015, pp. 1–7.

[13] R. Burke, "Hybrid Web recommender systems," in *The Adaptive Web*. Berlin, Germany: Springer, 2007, pp. 377–408.

[14] A. Schmidt, M. Beigl, and H.-W. Gellersen, "There is more to context than location," *Comput. Graph.*, vol. 23, no. 6, pp. 893–901, Dec. 1999.

[15] A. Karpus, I. Vagliano, K. Goczyła, and M. Morisio, "An ontology-based contextual pre-filtering technique for recommender systems," in *Proc. Federated Conf. Comput. Sci. Inf. Syst.*, Oct. 2016, pp. 411–420.

[16] V. Codina, F. Ricci, and L. Ceccaroni, "Distributional semantic pre-filtering in context-aware recommender systems," *User Model. User-Adapted Interact.*, vol. 26, no. 1, pp. 1–32, Mar. 2015.

[17] W.-P. Lee and G.-Y. Tseng, "Incorporating contextual information and collaborative filtering methods for multimedia recommendation in a mobile environment," *Multimedia Tools Appl.*, vol. 75, no. 24, pp. 16719–16739, Sep. 2015.

[18] G. James, D. Witten, T. Hastie, and R. Tibshirani, *An Introduction to Statistical Learning*. Berlin, Germany: Springer, 2013.

[19] L. Breiman, *Classification and Regression Trees*. Evanston, IL, USA: Routledge, 2017.

[20] G. Adomavicius and A. Tuzhilin, "Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions," *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005.

[21] Y. Koren, R. Bell, and C. Volinsky, "Matrix factorization techniques for recommender systems," *Computer*, vol. 42, no. 8, pp. 30–37, Aug. 2009.

[22] L. Baltrunas, B. Ludwig, and F. Ricci, "Matrix factorization techniques for context aware recommendation," in *Proc. 5th ACM Conf. Recommender Syst. (RecSys)*, 2011, pp. 301–304.

[23] Y. Zheng and A. A. Jose, "Context-aware recommendations via sequential predictions," in *Proc. 34th ACM/SIGAPP Symp. Appl. Comput. (SAC)*, 2019, pp. 2525–2528.

[24] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. J. Franklin, S. Shenker, and I. Stoica, "Resilient distributed datasets: A fault-tolerant abstraction for in-memory cluster computing," in *Proc. 9th USENIX Conf. Netw. Syst. Des. Implement.*, 2012, pp. 15–28.

[25] N. Shirish Keskar and R. Socher, "Improving generalization performance by switching from Adam to SGD," 2017, *arXiv:1712.07628*. [Online]. Available: http://arxiv.org/abs/1712.07628

[26] Y. Song and J.-G. Lee, "Augmenting recurrent neural networks with high-order user-contextual preference for session-based recommendation," 2018, *arXiv:1805.02983*. [Online]. Available: http://arxiv.org/abs/1805.02983

[27] Q. Liu, S. Wu, D. Wang, Z. Li, and L. Wang, "Context-aware sequential recommendation," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Dec. 2016, pp. 1053–1058.

[28] E. Smirnova and F. Vasile, "Contextual sequence modeling for recommendation with recurrent neural networks," in *Proc. 2nd Workshop Deep Learn. Recommender Syst. (DLRS)*, 2017, pp. 2–9.

[29] G. Adomavicius, R. Sankaranarayanan, S. Sen, and A. Tuzhilin, "Incorporating contextual information in recommender systems using a multidimensional approach," *ACM Trans. Inf. Syst.*, vol. 23, no. 1, pp. 103–145, Jan. 2005.

[30] L. Baltrunas and X. Amatriain, "Towards time-dependant recommendation based on implicit feedback," in *Proc. Workshop Context-Aware Recommender Syst. (CARS)*, 2009, pp. 25–30.

ISAM MASHHOUR AL JAWARNEH is currently and pursuing the Ph.D. degree with the Computer Science and Engineering Department (DISI), University of Bologna, Italy. He is also a research Assistant with DISI, University of Bologna. He has authored or coauthored many international journal articles and papers for flagship conferences (such as the IEEE GLOBECOM and ICC). He has a research and teaching experience at higher-education level for more than 12 years. His research interests include many aspects of big data stream processing and active data warehousing for highly dynamic application scenarios.

PAOLO BELLAVISTA (Senior Member, IEEE) received the M.Sc. and Ph.D. degrees in computer science engineering from the University of Bologna, Italy. He is currently a Full Professor of distributed and mobile systems with the University of Bologna. He is also the Scientific Coordinator of a large H2020 big data innovation action called the IoTwins about distributed digital twins for the manufacturing industry. His research activities span from pervasive wireless computing to location/context-aware services, from edge cloud computing to middleware for Industry 4.0 applications. He has serves on the Editorial Boards for the IEEE Communications Surveys and Tutorials, the IEEE TRANSACTIONS ON NETWORK AND SERVICE MANAGEMENT, *Pervasive and Mobile Computing* (Elsevier), the *Journal of Network and Computer Applications* (Elsevier), and the *Journal of Network and Systems Management* (Springer).

ANTONIO CORRADI (Senior Member, IEEE) graduated from the University of Bologna, Italy. He received the M.S. degree in electrical engineering from Cornell University, USA. He is currently a Full Professor of computer engineering with the University of Bologna. His research interests include distributed systems, middleware for pervasive and heterogeneous computing, infrastructure for services, and network management.

LUCA FOSCHINI (Senior Member, IEEE) received the Ph.D. degree in computer science engineering from the University of Bologna, Italy, in 2007. He is currently an Associate Professor of computer engineering with the University of Bologna. His interests span from integrated management of distributed systems and services to wireless pervasive computing, and scalable context data distribution infrastructures and context-aware services. He is also working on mobile crowdsensing and crowdsourcing, and management of Cloud systems for Smart City environments.

REBECCA MONTANARI received the Ph.D. degree in computer science engineering from the University of Bologna, in 2001. She is currently an Associate Professor of computer engineering with the University of Bologna. Her research primarily focuses on semantic-based middleware supports for service provisioning, context-aware services, security solutions for pervasive environments, policy-based service management, and adaptive and scalable middleware solutions for systems and service management.

JAVIER BERROCAL received the Ph.D. degree in computer science from the University of Extremadura, Spain, in 2014. In 2016, he joint as an Associate position at the University of Extremadura. His main research interests are mobile computing, context awareness, pervasive systems, crowd sensing, the Internet of Things, and fog computing. He is currently a Co-Founder of the company Gloin, which is a software-consulting company.

JUAN MANUEL MURILLO received the Ph.D. degree in computer science from the University of Extremadura. He is currently a Co-Founder of Gloin and a Full Professor with the University of Extremadura. His research interests include software architectures, mobile computing, and cloud computing.