A Linked Data Browser with Recommendations

Frederico Araújo Durão
Instituto de Matemática
Universidade Federal da Bahia
Salvador - Bahia, Brazil
freddurao@dcc.ufba.br

Derek Bridge
Insight Centre for Data Analytics
Department of Computer Science
University College Cork, Ireland
derek.bridge@insight-centre.org

***Abstract*—It is becoming more common to publish data in a way that accords with the Linked Data principles. In an effort to improve the human exploitation of this data, we propose a Linked Data browser that is enhanced with recommendation functionality. Based on a user profile, also represented as Linked Data, we propose a technique that we call LDRec that chooses in a personalized way which of the resources that lie within a certain neighbourhood in a Linked Data graph to recommend to the user. The recommendation technique, which is novel, is inspired by a collective classifier known as the Iterative Classification Algorithm. We evaluate LDRec using both an off-line experiment and a user trial. In the off-line experiment, we obtain higher hit rates than we obtain using a simpler classifier. In the user trial, comparing against the same simpler classifier, participants report significantly higher levels of overall satisfaction for LDRec.**

***Index Terms*—linked data, browsing, recommending, collective classification, iterative classification**

I. Introduction

The Linked Data principles and associated technologies[1] offer a way to format and publish ever-growing volumes of inter-linked, heterogeneous, distributed data. Published under an open license, the datasets of the Linked Open Data project[2] are an example, comprising billions of Rich Description Framework (RDF) triples that relate one entity to another. Technical-users and lay-users would benefit from tools, such as Linked Data browsers, that allow intuitive navigation through the data, thus supporting exploratory knowledge discovery [1]. However, the volume of data and its degree of inter-connectedness bring problems of information overload: there is too much data to display at a time and, even if it could be displayed intelligibly, there is too much data for a user to sift through to find the paths that she wants to explore next.

Recommender systems are one solution to the problem of information overload. From large volumes of items (products, services, news items, etc.), a recommender system surfaces just a few that it thinks are relevant to the user's interests.

We propose a Linked Data browser that incorporates recommender systems functionality. A user will have a user profile of items that she likes (also represented as Linked Data); as the user browses the Linked Data graph, the profile can grow through the addition of new items that the user likes; for any particular item that the user is focusing on, the browser displays just the part of the item's neighborhood that the recommender system predicts will be of interest to the user (based on the user profile). We refer to our enhanced browser as LDRec.

Linked Data browsers are not new. But we do not know of any that incorporate recommender systems functionality. There are, however, several recommender systems that use Linked Data. So we highlight here the ways in which a Linked Data browser like LDRec that uses a recommender system differs from a recommender system that uses linked data.

- Recommender systems typically recommend items for the user to consume (e.g. movies to watch, news items to read, restaurants to eat in, etc.). LDRec recommends Linked Data resources (entities). In some cases, these resources will correspond to items that the user can consume (movies, songs, etc.). But in other cases, they will correspond to real-world things that we would not normally 'consume' (an actor in a movie, the city in which the actor was born, and so on); they might even refer to abstract concepts (the period in which a work of art was produced, or the artistic movement to which it belongs). Consumption in a system like LDRec means learning about the resource by looking at its properties, which we assemble from the Linked Data graph into what we call an InfoBox. Of course, it may be that, after looking at a resource's InfoBox, the user does consume the corresponding real-world item, where this is possible.
- Recommender systems are most often single-domain in the sense that they recommend items that are all of the same type (e.g. songs). The research literature does have examples of cross-domain recommenders that operate across a small number of domains (e.g. where a person's movie preferences condition the recommendation of books); and there are notable examples of commercial multi-domain recommenders that operate across a wider product space (e.g. the recommender behind the Amazon web site). LDRec and systems like it are inevitably multi-domain: as the user browses, she receives recommendations for resources of many different type (songs, movies, books, actors, time periods, artistic movements, etc.).
- Because recommender systems typically confine recommendations to items of a single type (or one of a small number of types), recommender systems that use Linked Data can limit their attention to a subgraph of the Linked

[1]https://www.w3.org/standards/semanticweb/data
[2]https://lod-cloud.net/

Data graph. For example, they might index just a set of movie-related properties (e.g. stars in, directed by) for use by a recommendation algorithm. As the Linked Data graph changes, such a system will need to periodically rebuild this model. By contrast, LDRec cannot anticipate the resource types or property types that it will encounter. It needs to be open-ended [2], preferably capable of operating continuously as the Linked Data graph changes, including when new datasets are linked in.

LDRec treats recommendation as *one-class classification*. User profiles record resources that the user likes. During browsing, the recommender must classify candidate resources, i.e. it must predict whether the user will like each candidate.

In traditional classification, unseen objects are classified in isolation using just the features of that object (the *observed non-relational features*). The assumption is that class labels are independent of each other. However, if objects are inter-related, classification accuracy can often be improved by taking into account properties of the related objects. For example, the web pages of the faculty of a university might link to the web pages of postgraduate students, so if a page is classified as a faculty page then the ones it links to are more likely to be student pages. In this case, when classifying an unseen object, the classifier can also use the features and even labels (if known) of the related objects (the *observed relational features*) but might also use the *predicted labels* of the related objects (the *unobserved relational features*). For classifiers that use all three (observed non-relational features, observed relational features and unobserved relational features), the term *collective classification* is used [3].

LDRec recommendation is an instance of collective classification. As explained in detail later, it classifies candidate resources using the properties of the candidate resource, properties of resources that are in the neighbourhood of the candidate, and its predictions of the labels of the resources in the neighbourhood (i.e. whether it predicts the user will like the resources in the neighbourhood). All of this is seamless however because predicted labels are just additional RDF triples, temporarily inserted into the Linked Data graph.

There are many ways of building collective classifiers [4], [5]. LDRec is inspired by the Iterative Classification Algorithm (ICA) [6]. ICA uses what it calls a local classifier to classify all objects using just their observed features. Then, it repeatedly reclassifies the objects, this time using all three kinds of features, i.e. including the labels that it has just predicted, until predictions stabilize. This is well-suited to the open-ended nature of the task that we explained above.

In Section II, we describe other Linked Data browsers and recommenders that used Linked Data. Section III gives some definitions used in later parts of the paper. Section IV describes a non-iterative classifier that serves two purposes: (a) it is the local classifier used within LDRec's version of ICA, and (b) it is the system against which we compare LDRec, which is described in Section V. The comparison is reported in Section VI in the case of off-line experiments and Section VII in the case of a user trial.

II. Related Work

There have been many RDF browsers and visualizers, including e.g. Tabulator [7], LODlive [8], LODExplorer [9], Explorator [10], RelFinder [11], LODmilla [12] and LD Viewer [13]. These and others are surveyed in, e.g. [1], [14], [15]. Most of them, like LODlive, LODmilla and our own browser, visualize the graph by drawing circle-and-pointer diagrams; others, like Tabulator, convert to a more tabular format. There are no RDF browsers, to the best of our knowledge, that incorporate recommendation functionality to filter resources based on user tastes.

While we have no examples of Linked Data browsers that incorporate recommender functionality, recommender techniques have been incorporated into agents that support *web* browsing. One of the earliest was Letizia, which scans the neighbourhood of the page that the user is currently viewing and uses heuristics (e.g. word co-occurrence with the user's browsing history) to evaluate which neighbouring pages would be of interest to the user [16], [17]. Numerous systems have followed, e.g. [18]–[21]. For the most part, they either recommend pages that share document content with pages in the user's profile (e.g. [18], [21]) or they mine clickstreams to predict and recommend the pages that users tend to visit next (e.g. [19]); some do both (e.g. [20]). None of them takes the collective classification approach that we use in LDRec.

There are also many examples of content-based recommender systems where the items that are to be recommended are described using Linked Data. For example, non-personalized recommenders have been built, illustrating the use of the LDSD distance measure (see Section IV-B) and variants of that measure, that simply recommend resources that are similar to other resources [22], [23]. Members of the SISINF Lab at the Politecnico di Bari have developed a number of content-based recommender systems. In the first, which uses a vector space model, they represent RDF triples by a 3-dimensional tensor [24]–[26]. Cells in the tensor contain an analog of the TF-IDF values used in Information Retrieval. Similarity of two resources, such as one that the user likes (in her profile) and a candidate that could be recommended, is computed by cosine similarity for each RDF property (link type) and then aggregated. In their second system, similarity between two items involves a weighted count of entities that are common to the neighbourhoods of those two items; and this is used as a kernel in an SVM regression model learned from a user profile that comprises items and item ratings [27]. In their third system [28], they use an item-based nearest-neighbours recommender with similarity scores computed using either SimRank [29] or Personalized PageRank [30].

Collaborative recommenders that use Linked Data are rarer. In [31], it is proposed to extract from a Linked Data graph that includes user profile information a user-item matrix that is the conventional input to collaborative recommender systems. On the other hand, there are some collaborative/content-based hybrids. For example, in [32], a Factorization Machine is used where the features encode a traditional ratings matrix plus features that record the existence of properties in a Linked Data graph. The SISINF Lab has proposed a hybrid in which they model user profiles as a bipartite user-item graph and merge it with a graph that represents items and their properties (as we also do, Section III-C); they then find all paths that connect a user to a candidate item; and they score the item based on the quality of the paths. This scoring is done by a model that they have learned in supervised fashion from a dataset of such paths [33]. The SWAP Group at the University of Bari also model user profiles as a bipartite graph to construct a recommender that uses PageRank [34]. They explore the issue of feature selection: how to choose the most useful Linked Data properties to include in the PageRank calculation [35]. Separately, they also consider how Linked Data can be used for model-agnostic explanations of recommendations [36]. Finally, Zhang et al. and Khrouf & Troncy both use Linked Data to build collaborative/content-based hybrid recommender systems [37], [38].

There have been two 'challenges' in the area of Linked Open Data-enabled Recommender Systems, in which datasets are made available on which participants compare the performance of their recommender systems.[3] The participants in the 2014 challenge are described in [39]. One of the more successful participants used an ensemble of vector space, PageRank and other techniques [40].

A non-personalised recommender is described in [2]. It has separate mechanisms to recommend resources of the same type as a given input resource, to recommend people based on social network data, and to recommend resources based on textual descriptions. It is notable here because, like LDRec, it claims to be open-ended.

Finally, it is worth mentioning that Linked Open Data facilitates research in cross-domain recommendation, since it makes available descriptions of entities from more than one domain [41]–[43]. We see our own work as multi-domain recommendation, rather than cross-domain recommendation.

None of these recommender systems is targeted for use within a browser (although several can be adapted to be used in this way) and none of them uses collective classification. Notable too is that none of the papers that we cite except [2] includes an evaluation of the recommender with real users.

III. Basic Concepts

A. The Linked Data Graph

Following Passant [22], we can model a dataset that follows the Linked Data principles as a directed graph $G = \langle R, L, E\rangle$, where R is a set of resources; L is a set of link types; and E is a set of edges, i.e. instances of the link types between pairs of resources. A tuple $\langle r, l, r'\rangle \in E$ means that there is an instance of link type $l \in L$ from resource $r \in R$ to $r' \in R$. A tuple $\langle r, l, r'\rangle$ such as $\langle$`dbr:Oceans_Eleven`, `dbo:starring`, `dbr:Julia_Roberts`$\rangle$ corresponds to a Rich Description Framework (RDF) triple.

[3]https://2014.eswc-conferences.org/important-dates/call-RecSys.html, http://www.wikicfp.com/cfp/servlet/event.showcfp?eventid=44774

B. Neighbourhoods

In our work, we require a notion of the *neighbourhood* of a resource r: the resources that are within a certain number of edges of a given resource. In defining neighbourhood, similarly to [27], we ignore the directedness of the edges.

Hence, we define an *undirected path* in a directed graph $G = \langle R, L, E\rangle$ as a sequence of distinct resources $\langle r_0, r_1, \ldots, r_n\rangle$ where each $r_i \in R$ and there is a corresponding sequence of distinct edges $\langle e_1, e_2, \ldots, e_n\rangle$ where each $e_i \in E$ and $e_i = \langle r_{i-1}, l, r_i\rangle$ or $e_i = \langle r_i, l, r_{i-1}\rangle$. The length of the path is n, the number of edges.

We define the *neighbourhood* of a resource r in a graph G, $\mathrm{NbrHd}(n, r, G)$, to be the resources r' for which there is an undirected path between r and r' whose length is less than or equal to n:

$$\begin{aligned}&\mathrm{NbrHd}(n, r, G)\\ &\quad = \{r' \in R \mid \langle r_0, r_1, \ldots, r_m\rangle \textit{ is an undirected path in } G\\ &\qquad\qquad \wedge r_0 = r \wedge r_m = r' \wedge m \le n\}\end{aligned} \tag{1}$$

We emphasize that neighbourhood is based on paths in the graph. It is distinct from the notion of a nearest neighbour, which also appears in our work (Section IV-A), but which defines neighbours in terms of a semantic distance measure.

C. Representing user preferences

Any personalized recommender system must represent its users and their tastes in the form of user profiles. In LDRec, we assume that user profiles are represented as Linked Data. Specifically, we assume a set of users U who are themselves represented by resources in a Linked Data graph, $U \subset R$, each user being an instance of `User`. To represent user profiles, we assume the existence of a link type, `likes` $\in L$. An edge $\langle u, \texttt{likes}, r\rangle$ means that user $u \in U$ *likes* resource $r \in R$.

This means, in recommender system terminology, we only allow *explicit, unary* ratings. In other words, we link the user only to resources she has liked, and we know nothing about all other resources (whether they are disliked or simply unrated). Moreover, we do not assume that the user likes the resources that she *visits* during a browsing session — only the ones to which she gives an explicit 'thumbs up'. For example, if a user browses through all the actors who starred in *Ocean's Eleven* but gives a 'thumbs up' only to Andy Garcia, then we only add $\langle$u, `likes`, `dbr:Andy_Garcia`$\rangle$ to the graph.

Assume we have a resource $r \in R$ and an active user $u \in U$. Suppose r is a candidate for being recommended to u (e.g. it is not in the user's profile, $\langle u, \texttt{likes}, r\rangle \notin E$). We wish to predict whether u will like r. This is *classification* and, more specifically, since we have only positive examples (unary ratings), this is *one-class classification*.

D. Browsing with recommendation

In our work, we are interested in augmenting a Linked Data browser with recommendations. We assume the user has browsed to a particular resource in the graph. Candidates for

ALGORITHM 1: LDBrowser

```
Input: $G = \langle R, L, E \rangle$, a Linked Data graph
       $u \in R$, the active user
       $seed \in R$, a seed resource chosen by the user
       $n$, the max. path length in neighbourhoods
       $Recommender$, a recommender that predicts which
                                   candidates the user will like
$Taboo \leftarrow \{\}$;
$current \leftarrow seed$;
$Candidates \leftarrow$
       $\mathrm{NbrHd}(n, current, G) \setminus \{r | \langle u, \mathtt{likes}, r \rangle \in E\}$;
while $Candidates \neq \{\}$ do
    $Recs \leftarrow Recommender(G, u, Candidates)$;
    $current \leftarrow$ user choice from $Recs$;
    $Taboo \leftarrow Taboo \cup Recs$;
    $Candidates \leftarrow \mathrm{NbrHd}(n, current, G) \setminus$
              $(\{r | \langle u, \mathtt{likes}, r \rangle \in E\} \cup Taboo)$;
end
```

recommendation are other resources in the *neighbourhood* of that resource. We show a version of this as Algorithm 1. We only show the case where the user selects one of the recommended resources and the browser makes new recommendations based on that selection. To keep the pseudocode simple, we have deliberately omitted from Algorithm 1 the full range of user actions that our browser allows. For example, the user might signal that she likes one or more of the recommendations ('thumbs up'), resulting in the addition of new edges to the Linked Data graph; the user can consult history lists to resume browsing from a previously seen resource; she can enter a query term to search for resources; she can start a new browsing session by selecting a fresh seed; and, of course, she can abandon the browser session when she wishes.

In Sections IV and V, we present two recommendation algorithms that can be used within this browser.

IV. Recommending using a Non-Iterative Classifier

Among its parameters, the browser that we presented as Algorithm 1 takes in a recommendation algorithm, $Recommender$. Our main contribution, in Section V, is a collective classifier (LDRec) that uses an iterative algorithm that we believe is well-suited to this purpose. But, for two reasons, we present a simpler classifier first: (a) LDRec makes use of this simple classifier, and (b) we will compare LDRec against this simpler classifier in our experiments.

A. One-class nearest-neighbour classifier

As explained earlier, we need a one-class classifier. The algorithm we choose is called One-Class Nearest-Neighbour Classifier (OC-NN), e.g. [44], [45]. It is simple and cheap to run, which is important when it gets used within LDRec. Its training set $TrainSet$ is a set of positive examples (in our case, these are drawn from the items that the user likes). Consider resource r; let's designate its nearest-neighbour in $TrainSet$ by r'; and let's designate the nearest-neighbour of r' in $TrainSet$ (excluding r) by r''. OC-NN predicts that u

ALGORITHM 2: OC-NN

```
Input: $G = \langle R, L, E \rangle$, a Linked Data graph
       $u \in U$, the active user
       $r \in R$, the candidate resource to be classified
       $TrainSet \subseteq \{r' | \langle u, \mathtt{likes}, r' \rangle \in E\}$, positive training
                                    examples for active user $u$
Output: Boolean: true iff we predict that $u$ will like $r$ using $u$'s
                                    training set, $TrainSet$
/* Predict false if the training set is too
   small                                           */
if $|TrainSet| < 2$ then
    return $false$;
end
/* Find the distance between r and its
   nearest-neighbour                               */
$nn \leftarrow$ null;
$dist\_to\_nn \leftarrow \infty$;
foreach $r' \in TrainSet$ do
    $d' \leftarrow \mathrm{dist}(r, r')$;
    if $d' < dist\_to\_nn$ then
        $nn \leftarrow r'$;
        $dist\_to\_nn \leftarrow d'$;
    end
end
/* Find the distance between the nearest-
   neighbour and its nearest-neighbour             */
$dist\_to\_nn\_of\_nn \leftarrow \infty$;
foreach $r' \in TrainSet$ do
    if $r' \neq nn$ then
        $d' \leftarrow \mathrm{dist}(nn, r')$;
        if $d' < dist\_to\_nn\_of\_nn$ then
            $dist\_to\_nn\_of\_nn \leftarrow d'$;
        end
    end
end
return $dist\_to\_nn \leq dist\_to\_nnofnn$;
```

ALGORITHM 3: LDRec-non-iterative

```
Input: $G = \langle R, L, E \rangle$, a Linked Data graph
       $u \in U$, the active user
       $Candidates \subset R$, candidate resources to be classified
       $LocalClassifier$, a one-class classifier such as OC-NN
Output: A subset of $Candidates$: those that it predicts the user
                                                will like
$Recs \leftarrow \{\,\}$
foreach $r \in Candidates$ do
    $TrainSet_r \leftarrow$
         $\{r' \in \mathrm{NbrHd}(n', r, G) | \langle u, \mathtt{likes}, r' \rangle \in E\}$;
    if $LocalClassifier(G, u, r, TrainSet_r)$ then
        /* local classifier predicts user will
           like r                                  */
        insert $r$ into $Recs$;
    end
end
return $Recs$;
```

will like r if the distance between r and r' is less than or equal to the distance between r' and r''; see Algorithm 2.

Algorithm 2 classifies a single candidate resource. Then, as Algorithm 3, we show how this classifier is used to recommend from a set of candidates. Quite simply, it calls the classifier on each candidate and returns the ones it predicts will be liked.

This may seem obvious but we emphasize it because our other recommender (LDRec-iterative, Section V) takes in a set of candidates and classifies them 'simultaneously'.

B. Semantic distance measures for Linked Data

The OC-NN algorithm requires a $[0,1]$-valued distance measure: $\text{dist} : R \times R \to [0,1]$. This is a semantic measure; it is not defined in terms of path lengths (unlike NbrHd above), although it is computed from the Linked Data graph.

There are many ways to define this measure. A well-known example is Linked Data Semantic Distance (LDSD). The definition can be found in [22]. In overview, the distance between r and r' is an aggregate of the number of times r and r' both link to the same third resource r'' and the number of times that a third resource r'' links to both r and r'.

In the offline experiment and user trial reported later in this paper, we use $\text{dist} = \text{LDSD}$ in the OC-NN algorithm. One of its strengths is that it works directly on the Linked Data graph. Unlike distance measures defined on a vector space, as in [24]–[26], LDSD does not require extraction of data for a given set of link types. LDSD does, however, have some weaknesses. One weakness is that it treats all link types equally. This can be 'fixed' by the use of weights for link types, and this has been experimented with by Piao & Breslin, where the weights were set by a Genetic Algorithm [23]. Another weakness is that it is quite 'local' in its operation. By contrast, SimRank by Jeh & Widom takes a more global view [29] but SimRank is correspondingly more costly to compute.

V. Recommending with an Iterative Classifier

As we said in Section I, the algorithm we use in LDRec is inspired by the Iterative Classification Algorithm (ICA). To classify candidate resources, it first classifies them with a local classifier (OC-NN). When it predicts a candidate will be liked, it *temporarily* inserts a new RDF triple into the Linked Data graph. It then reclassifies the candidates, but this allows the predicted class labels that were inserted into the graph to be used by the local classifier. Then it updates the graph again with these latest predictions: it may remove some of the triples that it previously inserted and it may insert new triples. In principle, this continues until classifications stabilize. In practice, we repeat for a fixed number of iterations.

Our algorithm is given as Algorithm 4. As the pseudocode shows, it 'simultaneously' classifies a set of candidate resources (designated $\mathit{Candidates}$). In fact, as with ICA, it also classifies resources in the union of the neighbourhoods of these candidates (designated $\mathit{ExtendedCandidates}$). It repeatedly classifies these extended candidates using the local classifier. We do not modify their training sets ($\mathit{TrainSet}_r$), which might make it appear that the local classifier will give the same result each time it is invoked on a given resource.[4] But this ignores the fact that the graph is updated with the predictions of the local classifier (the edges in pos and neg). This affects the

[4]We could recompute the training sets by moving the for-loop into the repeat-loop. In preliminary experiments, we found that this slowed the algorithm down considerably without much change to accuracy.

ALGORITHM 4: LDRec-iterative

```
Input: $G = \langle R, L, E\rangle$, a Linked Data graph
       $u \in U$, the active user
       $\mathit{Candidates} \subset R$, candidate resources to be classified
       $n$, the max. path length in neighbourhoods (for $\mathit{ExtendedCandidates}$)
       $n'$, the max. path length in neighbourhoods (for $\mathit{TrainSet}$)
       $\mathit{LocalClassifier}$, a one-class classifier such as OC-NN
Output: A subset of $\mathit{Candidates}$: those that LDRec predicts the user will like

save $u$'s profile;
$\mathit{ExtendedCandidates} \leftarrow \mathit{Candidates} \cup$
    $\bigcup_{r \in \mathit{Candidates}} \{r' \in \text{NbrHd}(n, r, G) | \langle u, \texttt{likes}, r'\rangle \notin E\}$;
foreach $r \in \mathit{ExtendedCandidates}$ do
    $\mathit{TrainSet}_r \leftarrow$
        $\{r' \in \text{NbrHd}(n', r, G) | \langle u, \texttt{likes}, r'\rangle \in E\}$;
end
repeat
    $\mathit{pos} \leftarrow []$;
    $\mathit{neg} \leftarrow []$;
    foreach $r \in \mathit{ExtendedCandidates}$ do
        if $\mathit{LocalClassifier}(G, u, r, \mathit{TrainSet}_r)$ then
            /* local classifier predicts user
               will like $r$                        */
            append $r$ onto $\mathit{pos}$;
        end
        else
            append $r$ onto $\mathit{neg}$;
        end
    end
    foreach $r \in \mathit{pos}$ do
        if $\langle u, \texttt{likes}, r\rangle \notin E$ then
            insert $\langle u, \texttt{likes}, r\rangle$ into $E$;
        end
    end
    foreach $r \in \mathit{neg}$ do
        if $\langle u, \texttt{likes}, r\rangle \in E$ then
            remove $\langle u, \texttt{likes}, r\rangle$ from $E$;
        end
    end
until classifications stabilize;

restore $u$'s profile;
return $\{r \in \mathit{Candidates} \cap \mathit{pos}\}$;
```

LDSD, which in turn means that the local classifier can obtain different predictions each time.

We assume that predicted 'likes' should not be permanent, which is why we save and restore the user's profile at the start and end of the algorithm, respectively.

We refer to Algorithm 4, which uses iterative classification, as LDRec-iterative; and we refer to Algorithm 3, which is not a collective classifier, as LDRec-non-iterative. However, for brevity, most sections of this paper refer to LDRec-iterative as simply LDRec.

VI. Off-Line Experiments

We evaluate the recommenders using off-line experiments (to measure precision) and a user trial (to measure 'satisfaction'). In both cases, our main goal is to evaluate the

TABLE I
DATASET OVERVIEW

59083 users with a total of 1743719 'likes'		
	num. resources	mean (st.dev.) 'likes' per user
movies	5389	19.8 (5.7)
books	3225	8.3 (1.9)
music artists	6372	21.0 (6.2)

TABLE II
HIT RATES FOR 100 RANDOMLY-CHOSEN USERS

	top-N hit rate			
LDRec	$N = 1$	$N = 3$	$N = 5$	$N = 10$
non-iterative	11	17	25	34
iterative	14	31	43	57

usefulness of the LDRec-iterative algorithm, which repeatedly re-classifies the extended candidates, by comparing it with LDRec-non-iterative. In principle, LDRec-iterative runs until classifications stabilize, i.e. they stop changing; see Algorithm 4. In practice, the number of iterations needed before classifications stabilize is unpredictable, which is not suitable for a Linked Data browser because of the way it would make response times quite variable. Therefore, we limit the number of iterations to 3. We chose 3 because preliminary experiments showed that classifications often did stabilize within 3 iterations and also because it is a small enough number of iterations to keep run-times low. In computing neighbourhoods, we used $n = 1$ for $ExtendedCandidates$ and $n' = 2$ for $TrainSet$.

For these experiments, we used Linked Data that was originally made available for the 2015 *Linked Open Data-enabled Recommender Systems Challenge*, mentioned in Section II. The original data was Facebook 'likes' for movies, books and music artists; see Table I. The competition organizers anonymized the users and reconciled the Facebook movies, books and music artists with their corresponding Linked Data resources from DBPedia.[5]

This dataset does not provide any information about the resources (movies, books and music artists). Hence, when constructing neighbourhoods, we query live data at DBPedia to obtain their properties, the other resources that they are linked to, the properties of these other resources, and so on, sufficient for the needs of the recommender algorithm.

The methodology that we use in the off-line experiments is based on the *one plus random* method described by Koren [46]. For each of 100 randomly-selected users, we randomly select a *test resource* from their profile, i.e. a resource that we know that they like. We remove the corresponding edge from the Linked Data graph, in order to withhold the fact that this user likes this resource. We create a set of candidates ($Candidates$) that contains the test resource and 49 other candidates that are chosen at random from resources that are not in the user's profile. We pass the candidates to the recommender algorithms. For each candidate, the algorithms predict whether the user will like the candidate or not. We take the top-N of these (for different values of N) and we measure the hit rate: the proportion of users for whom the test resource is a member of the top-N.

We need to explain how we select the top-N given that our recommenders are classifiers: they predict whether a user will like a candidate or not; they do not predict numeric ratings. For the purposes of this off-line experiment and the user trial, we modify the algorithms to return scores, enabling us to select a top-N. For both LDRec-non-iterative and LDRec-iterative, for a resource r that the classifier predicts the user will like, the score is the distance between r and its nearest-neighbour r' minus the distance between r' and its nearest-neighbour r''.

The results for $N = 1, 3, 5$ and 10 are shown in Table II.

We see that LDRec-iterative always has a higher hit rate than LDRec-non-iterative, which suggests that collective classification gives a more accurate recommender.

VII. USER TRIAL

We conducted a user trial to compare LDRec-non-iterative and LDRec-iterative. We recruited 100 volunteers to participate in the trial. Each was a student or researcher from Brazil. For half of them the browser used the LDRec-non-iterative recommender; for the other half, it used the LDRec-iterative recommender. We used the same dataset of movies, books and music artists that we used in the offline experiment.

The first step was for each participant to create her user profile, comprising a minimum of 20 items that she liked. The next step was for the participant to select one resource from her on-screen profile to act as a seed r. The candidates for recommendation are $\mathrm{NbrHd}(1, r, G)$, i.e. all the resources r' that link to, or are linked to, the seed r. The recommender then classifies these candidates and selects the top-3 to show to the user.

The user was then required to click on *each* recommendation in turn to bring up an InfoBox, describing the recommended resource. For example, if the resource were a movie, the box would include data from the Linked Data graph about the director, the actors and so on. She was then required to answer four questions about the recommendation:

- *Accuracy*: How accurate is this recommendation based on your profile?
- *Understanding*: How well does the recommendation reflect your preferences?
- *Novelty*: How familiar are you with the object that is being recommended?
- *Satisfaction*: How much do you like this recommendation?

Each of these questions could be answered on a rating scale: "Not at all", "A little", "Neutral, "Much", or "Very much". A final question asked: Do you like this recommendation enough to add it to your profile? If the user chose "Yes", the recommended item was inserted into the on-screen profile and a new edge was added into the Linked Data graph.

The user had to then repeat this step two more times, i.e. she had to choose a seed from her profile or from the

[5]http://dbpedia.org

TABLE III
RATINGS FROM THE USER TRIAL

	LDRec	
	non-iterative	iterative
	mean (std.dev.)	mean (std.dev.)
Accuracy	2.5 (0.9)	3.4 (0.8)
Understanding	2.7 (0.9)	3.4 (1.0)
Novelty	2.5 (0.8)	3.2 (1.0)
Satisfaction	2.6 (1.0)	3.5 (0.9)

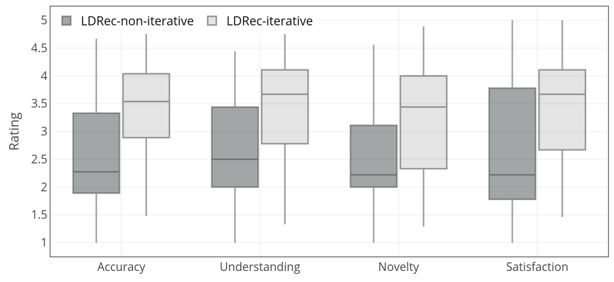

Fig. 1. Box-plots of user ratings

three recommendations already on the screen. She would then receive three new recommendations and evaluate them in the same way. The trial was only over when we had evaluations for all nine recommendations.

In order to quantify the results, we converted the answers to the questions to a 1 to 5 scale, where 1 = "Not at all' and 5 = "Very much". Table III shows mean ratings and standard deviations. Figure 1 gives an alternative visualization of the ratings. As can be seen from both the Table and the Figure, for all four questions LDRec-iterative was rated more highly than LDRec-non-iterative, which again suggests that collective classification gives a better recommender (although it may not be desirable that LDRec-iterative makes recommendations that are less novel, and this may require further investigation). The differences in the results are statistically significant at the 95% level using a two-tailed Student's t-test.

Finally, Table IV shows us two other aspects of user behaviour. As mentioned, users are asked whether they want to add a recommended resource to their profile. Given that for each system there were 50 participants each receiving nine recommendations, there were 450 times this could happen. The table shows that it happened in 30% of LDRec-non-iterative recommendations and 52% of LDRec-iterative recommendations: an increase relative to LDRec-non-iterative of 76%. We also mentioned that the user must choose the next seed and this might be one of the resources that had been recommended. Given that for each system there were 50 participants and this applies to the second and third seeds that the user selects, there were 100 times the user could choose a seed from the recommendations. The table shows that it happened in 52% of the time with LDRec-non-iterative and 70% of the time with

TABLE IV
HOW PARTICIPANTS IN THE TRIAL USED THE RECOMMENDATIONS

	LDRec	
	non-iterative	iterative
total num. recs. added to profiles	133	234
total num. recs. chosen as next seeds	52	70

LDRec-iterative: an increase relative to LDRec-non-iterative of 35%. These results confirm the results that we obtained from the survey questions.

VIII. CONCLUSIONS

In this paper, we have presented a Linked Data browser that incorporates recommender system functionality. It treats the resources that it can recommend as open-ended, allowing new datasets to be linked in dynamically. To achieve this, we take a novel approach to recommendation, treating it as collective classification and using an algorithm that is inspired by the Iterative Classification Algorithm (ICA) [6].

We report the results of an offline experiment and a user trial. In the offline experiment, hit rates are consistently higher if we use the iterative algorithm than if we use a non-iterative variant. Similarly, in the user trial, users gave higher mean ratings to recommendations from the iterative algorithm and were much more likely to add them to their profiles and to use them as seeds for subsequent rounds of recommendations.

There are many possible avenues for future work. We could vary the semantic distance measure used in the OC-NN classifier; we could even replace the OC-NN classifier by some other one-class classifier. Similarly, we could replace the ICA algorithm with one that is inspired by other ways of doing collective classification. These include methods that define and optimize a set of weights in a global objective function [4] and so-called 'cautious' variants of ICA [3], [47].

Our work focuses on the recommendation algorithm. We would like to improve the browser's user interface so as to incorporate more of the functionality found in other Linked Data browsers and to engineer a more deployable system.

Finally, we have shown how to use LDRec within a Linked Data browser. But the algorithm could be used as a more conventional recommender. Then it would be interesting to compare it with other recommenders that use Linked Data (e.g. [24]–[26]).

ACKNOWLEDGMENTS

This paper emanates from research supported by a grant from Science Foundation Ireland (SFI) under Grant Number SFI/12/RC/2289 which is co-funded under the European Regional Development Fund.

REFERENCES

[1] A.-S. Dadzie and M. Rowe, "Approaches to Visualising Linked Data: A Survey," *Semant. web*, vol. 2, no. 2, pp. 89–124, 2011.
[2] J. Oliveira, C. Delgado, and A. C. Assaife, "A recommendation approach for consuming linked open data," *Expert Systems with Applications*, vol. 72, pp. 407 – 420, 2017.

[3] L. McDowell, K. M. Gupta, and D. W. Aha, “Cautious Collective Classification,” *Journal of Machine Learning Research*, vol. 10, pp. 2777–2836, 2009.

[4] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, “Collective Classification in Network Data,” *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.

[5] L. McDowell, K. M. Gupta, and D. W. Aha, “Case-Based Collective Classification,” in *Procs. of the 20th Intl. Florida Artificial Intelligence Research Society Conf.*, 2007, pp. 399–404.

[6] Q. Lu and L. Getoor, “Link-based Classification,” in *Procs. of the 20th Intl. Conf. on Machine Learning*, 2003, pp. 496–503.

[7] T. Berners-Lee, Y. Chen, L. Chilton, D. Connolly, R. Dhanaraj, J. Hollenbach, A. Lerer, and D. Sheets, “Tabulator: Exploring and Analyzing linked data on the Semantic Web,” in *Procs. of the 3rd Intl. Semantic Web User Interaction Workshop*, 2006.

[8] D. V. Camarda, S. Mazzini, and A. Antonuccio, “LodLive, Exploring the Web of Data,” in *Procs. of the 8th Intl. Conf. on Semantic Systems*, 2012, pp. 197–200.

[9] K. Jacksi, S. R. Zeebaree, and N. Dimililer, “LOD Explorer: Presenting the Web of Data,” *Intl. Journal of Advanced Computer Science and Applications*, vol. 9, no. 1, pp. 45–51, 2018.

[10] S. F. C. D. Araújo, D. Schwabe, and S. D. J. Barbosa, “Experimenting with explorator: a direct manipulation generic rdf browser and querying tool,” in *CEUR Workshop Proceedings*, vol. 443, 2009.

[11] S. Lohmann, P. Heim, T. Stegemann, and J. Ziegler, “The RelFinder User Interface: Interactive Exploration of Relationships Between Objects of Interest,” in *Procs. of the 15th Intl. Conf. on Intelligent User Interfaces*, 2010, pp. 421–422.

[12] A. M. A, Z. Tóth, and S. Turbucz, “LODmilla: Shared Visualization of Linked Open Data,” in *Theory and Practice of Digital Libraries – TPDL 2013 Selected Workshops*, 2014.

[13] D. Lukovnikov, C. Stadler, and J. Lehmann, “LD Viewer - Linked Data Presentation Framework,” in *Procs. of the 10th Intl. Conf. on Semantic Systems*, 2014, pp. 124–131.

[14] K. Jacksi, N. Dimililer, and S. R. Zeebaree, “State of the Art Exploration Systems for Linked Data: A Review,” *Int. J. Adv. Comput. Sci. Appl.*, vol. 7, pp. 155–164, 2016.

[15] N. Marie and F. Gandon, “Survey of Linked Data Based Exploration Systems,” in *Procs. of the 3rd International Conf. on Intelligent Exploration of Semantic Data*, 2014, pp. 66–77.

[16] H. Lieberman, “Letizia: An agent that assists web browsing,” in *Procs. of the 14th Intl. Joint Conf. on Artificial Intelligence*, 1995, pp. 924–929.

[17] H. Lieberman, C. Fry, and L. Weitzman, “Exploring the Web with Reconnaissance Agents,” *Comm. ACM*, vol. 44, no. 8, pp. 69–75, 2001.

[18] M. Balabanovic and Y. Shoham, “Learning Information Retrieval Agents: Experiments with Automated Web Browsing,” in *Procs. of the AAAI Spring Symposium on Information Gathering*, 1995.

[19] E. Gams, T. Berka, and S. Reich, “The TrailTRECer Framework - A Platform for Trail-Enabled Recommender Applications,” in *Procs. of the 13th Intl. Conf. on Database and Expert Systems Applications*, 2002, pp. 638–647.

[20] S. Stober and A. Nürnberger, “Context-based Navigational Support in Hypermedia,” in *Procs. of the 4th Intl. Conf. on Adaptive Hypermedia and Adaptive Web-Based Systems*, 2006, pp. 328–332.

[21] T. Zhu, R. Greiner, G. Häubl, K. Jewell, and B. Price, “Using learned browsing behavior models to recommend relevant web pages,” in *Procs. of the 19th Intl. Joint Conf. on Artificial Intelligence*, 2005, pp. 1589–1591.

[22] A. Passant, “Measuring Semantic Distance on Linking Data and Using it for Resources Recommendations,” in *AAAI Spring Symposium Series*, 2010.

[23] G. Piao and J. G. Breslin, “Measuring Semantic Distance for Linked Open Data-enabled Recommender Systems,” in *Procs. of the 31st Annual ACM Symposium on Applied Computing*, 2016, pp. 315–320.

[24] R. Mirizzi, T. Di Noia, E. Di Sciascio, and A. Ragone, “Web 3.0 in Action: Vector Space Model for Semantic (Movie) Recommendations,” in *Procs. of the 27th Annual ACM Symposium on Applied Computing*, 2012, pp. 403–405.

[25] T. Di Noia, R. Mirizzi, V. C. Ostuni, D. Romito, and M. Zanker, “Linked Open Data to Support Content-based Recommender Systems,” in *Procs. of the 8th Intl. Conf. on Semantic Systems*, 2012, pp. 1–8.

[26] T. Di Noia, R. Mirizzi, V. C. Ostuni, and D. Romito, “Exploiting the Web of Data in Model-based Recommender Systems,” in *Procs. of the 6th ACM Conf. on Recommender Systems*, 2012, pp. 253–256.

[27] V. C. Ostuni, T. D. Noia, R. Mirizzi, and E. D. Sciascio, “A Linked Data Recommender System Using a Neighborhood-Based Graph Kernel,” in *Procs. of E-Commerce and Web Technologies*, 2014, pp. 89–100.

[28] P. Nguyen, P. Tomeo, T. Di Noia, and E. Di Sciascio, “An Evaluation of SimRank and Personalized PageRank to Build a Recommender System for the Web of Data,” in *Procs. of the 24th Intl. Conf. on World Wide Web*, 2015, pp. 1477–1482.

[29] G. Jeh and J. Widom, “SimRank: A Measure of Structural-context Similarity,” in *Procs. of the 8th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2002, pp. 538–543.

[30] E. Agirre and A. Soroa, “Personalizing PageRank for Word Sense Disambiguation,” in *Procs. of the 12th Conf. of the European Chapter of the Association for Computational Linguistics*, 2009, pp. 33–41.

[31] B. Heitmann and C. Hayes, “Using Linked Data to Build Open, Collaborative Recommender Systems,” in *AAAI Spring Symposium Series*, 2010.

[32] G. Piao and J. G. Breslin, “Factorization Machines Leveraging Lightweight Linked Open Data-enabled Features for Top-N Recommendations,” *CoRR*, vol. abs/1707.05651, 2017.

[33] V. C. Ostuni, T. Di Noia, E. Di Sciascio, and R. Mirizzi, “Top-N Recommendations from Implicit Feedback Leveraging Linked Open Data,” in *Procs. of the 7th ACM Conf. on Recommender Systems*, 2013, pp. 85–92.

[34] C. Musto, P. Lops, P. Basile, M. de Gemmis, and G. Semeraro, “Semantics-aware Graph-based Recommender Systems Exploiting Linked Open Data,” in *Procs. of the 2016 Conf. on User Modeling Adaptation and Personalization*, 2016, pp. 229–237.

[35] 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, 2017.

[36] C. Musto, F. Narducci, P. Lops, M. De Gemmis, and G. Semeraro, “ExpLOD: A Framework for Explaining Recommendations Based on the Linked Open Data Cloud,” in *Procs. of the 10th ACM Conf. on Recommender Systems*, 2016, pp. 151–154.

[37] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, “Collaborative Knowledge Base Embedding for Recommender Systems,” in *Procs. of the 22nd ACM SIGKDD International Conf. on Knowledge Discovery and Data Mining*, 2016, pp. 353–362.

[38] H. Khrouf and R. Troncy, “Hybrid Event Recommendation Using Linked Data and User Diversity,” in *Procs. of the 7th ACM Conf. on Recommender Systems*, 2013, pp. 185–192.

[39] T. D. Noia, I. Cantador, and V. C. Ostuni, “Linked Open Data-Enabled Recommender Systems: ESWC 2014 Challenge on Book Recommendation,” in *Semantic Web Evaluation Challenge*, 2014, pp. 129–143.

[40] P. Basile, C. Musto, M. de Gemmis, P. Lops, F. Narducci, and G. Semeraro, “Aggregation strategies for linked open data-enabled recommender systems,” in *Procs. of the European Semantic Web Conf.*, 2014.

[41] I. Fernández-Tobías, P. Tomeo, I. Cantador, T. Di Noia, and E. Di Sciascio, “Accuracy and Diversity in Cross-domain Recommendations for Cold-start Users with Positive-only Feedback,” in *Procs. of the 10th ACM Conf. on Recommender Systems*, 2016, pp. 119–122.

[42] B. Heitmann, “An Open Framework for Multi-source, Cross-domain Personalisation with Semantic Interest Graphs,” in *Procs. of the 6th ACM Conf. on Recommender Systems*, 2012, pp. 313–316.

[43] B. Heitmann, M. Dabrowski, A. Passant, C. Hayes, and K. Griffin, “Personalisation of Social Web Services in the Enterprise Using Spreading Activation for Multi-Source, Cross-Domain Recommendations,” in *AAAI Spring Symposium Series*, 2012.

[44] D. de Ridder, D. Tax, and R. Duin, “An experimental comparison of one-class classification methods,” in *Procs. of the 4th Annual Conf. of the Advanced School for Computing and Imaging*, 1998.

[45] M. Yousef, N. Najami, and W. Khalifav, “A comparison study between one-class and two-class machine learning for MicroRNA target detection,” *Journal of Biomedical Science and Engineering*, vol. 3, pp. 247–252, 2010.

[46] Y. Koren, “Factorization meets the neighborhood: A multifaceted collaborative filtering model,” in *Procs. of the 14th ACM SIGKDD Intl. Conf. on Knowledge Discovery and Data Mining*, 2008, pp. 426–434.

[47] L. McDowell, K. M. Gupta, and D. W. Aha, “Cautious inference in collective classification,” in *Procs. of the 22nd AAAI Conf. on Artificial Intelligence*, 2007, pp. 596–601.