RDF Graph Embeddings for Content-based Recommender Systems

Jessica Rosati¹,²
¹University of Camerino –
Piazza Cavour 19/f – 62032
Camerino, Italy
²Polytechnic University of Bari
– Via Orabona, 4 – 70125
Bari, Italy
jessica.rosati@unicam.it

Petar Ristoski
Data and Web Science Group,
University of Mannheim, B6,
26, 68159 Mannheim,
Germany
petar.ristoski@informatik.uni-mannheim.de

Tommaso Di Noia
Polytechnic University of Bari
– Via Orabona, 4 – 70125
Bari, Italy
tommaso.dinoia@poliba.it

Renato De Leone
University of Camerino –
Piazza Cavour 19/f – 62032
Camerino, Italy
renato.deleone@unicam.it

Heiko Paulheim
Data and Web Science Group,
University of Mannheim, B6,
26, 68159 Mannheim,
Germany
heiko@informatik.uni-mannheim.de

ABSTRACT

Linked Open Data has been recognized as a useful source of background knowledge for building content-based recommender systems. Vast amount of RDF data, covering multiple domains, has been published in freely accessible datasets. In this paper, we present an approach that uses language modeling approaches for unsupervised feature extraction from sequences of words, and adapts them to RDF graphs used for building content-based recommender systems. We generate sequences by leveraging local information from graph sub-structures and learn latent numerical representations of entities in RDF graphs. Our evaluation on two datasets in the domain of movies and books shows that feature vector representations of general knowledge graphs such as DBpedia and Wikidata can be effectively used in content-based recommender systems.

Categories and Subject Descriptors
H.3.3 [Information Systems]: Information Search and Retrieval

Keywords
Recommender System; Graph Embeddings; Linked Open Data

1. INTRODUCTION

One of the main limitations of traditional content-based recommendation approaches is that the information on which they rely is generally insufficient to elicit user’s interests and characterize all the aspects of her interaction with the system. This is the main drawback of the approaches built on textual and keyword-based representations, which cannot capture complex relations among objects since they lack the semantics associated to their attributes. A process of “knowledge infusion” [40] and semantic analysis has been proposed to face this issue, and numerous approaches that incorporate ontological knowledge have been proposed, giving rise to the newly defined class of semantics-aware content-based recommender systems [6]. More recently the Linked Open Data (LOD) initiative [3] has opened new interesting possibilities to realize better recommendation approaches. The LOD initiative in fact gave rise to a variety of open knowledge bases freely accessible on the Web and being part of a huge decentralized knowledge base, the LOD cloud, where each piece of little knowledge is enriched by links to related data. LOD is an open, interlinked collection of datasets in machine-interpretable form, built on World Wide Web Consortium (W3C) standards as RDF¹, and SPARQL². Currently the LOD cloud consists of about 1,000 interlinked datasets covering multiple domains from life science to government data [39]. It has been shown that LOD is a valuable source of background knowledge for content-based recommender systems in many domains [12]. Given that the items to be recommended are linked to a LOD dataset, information from LOD can be exploited to determine which items are considered to be similar to the ones that the user has consumed in the past, allowing to discover hidden information and implicit relations between objects [26]. While LOD is rich in high quality data, it is still challenging to find effective and efficient way of exploiting the knowledge for content-based recommendations. So far, most of the pro-

²http://www.w3.org/TR/rdf-sparql-query/, 2008
posed approaches in the literature are supervised or semi-supervised, which means cannot work without human interaction.

In this work, we adapt language modeling approaches for latent representation of entities in RDF graphs. To do so, we first convert the graph into a set of sequences of entities using graph walks. In the second step, we use those sequences to train a neural language model, which estimates the likelihood of a sequence of entities appearing in the graph. Once the training is finished, each entity in the graph is represented with a vector of latent numerical values. Projecting such latent representation of entities into a lower dimensional feature space shows that semantically similar entities appear closer to each other. Such entity vectors can be directly used in a content-based recommender system.

In this work, we utilize two of the most prominent RDF knowledge graphs [29], i.e. DBpedia [18] and Wikidata [42]. DBpedia is a knowledge graph which is extracted from structured data in Wikipedia. The main source for this extraction are the key-value pairs in the Wikipedia infoboxes. Wikidata is a collaboratively edited knowledge graph, operated by the Wikimedia foundation\(^3\) that also hosts various language editions of Wikipedia.

The rest of this paper is structured as follows. In Section 2, we give an overview of related work. In Section 3, we introduce our approach, followed by an evaluation in Section 4. We conclude with a summary and an outlook on future work.

2. RELATED WORK

It has been shown that LOD can improve recommender systems towards a better understanding and representation of user preferences, item features, and contextual signs they deal with. LOD has been used in content-based, collaborative, and hybrid techniques, in various recommendation tasks, i.e., rating prediction, top-N recommendations and improving of diversity in content-based recommendations. LOD datasets, e.g. DBpedia, have been used in content-based recommender systems in [11] and [12]. The former performs a semantic expansion of the item content based on ontological information extracted from DBpedia and LinkedMDB [16], the first open semantic web database for movies, and tries to derive implicit relations between items. The latter involves DBpedia and LinkedMDB too, but is an adaptation of the Vector Space Model to Linked Open Data: it represents the RDF graph as a 3-dimensional tensor where each slice is an ontological property (e.g. starring, director, ...) and represents its adjacency matrix. It has been proven that leveraging LOD datasets is also effective for hybrid recommender systems [4], that is in those approaches that boost the collaborative information with additional knowledge, such as the item content. In [10] the authors propose SPRank, a hybrid recommendation algorithm that extracts semantic path-based features from DBpedia and uses them to compute top-N recommendations in a learning to rank approach and in multiple domains, movies, books and musical artists. SPRank is compared with numerous collaborative approaches based on matrix factorization [17, 34] and with other hybrid RS, such as BPR-SSLM [25], and exhibits good performance especially in those contexts characterized by high sparsity, where the contribution of the content becomes essential. Another hybrid approach is proposed in [36], which builds on training individual base recommenders and using global popularity scores as generic recommenders. The results of the individual recommenders are combined using stacking regression and rank aggregation. Most of these approaches can be referred to as top-down approaches [6], since they rely on the integration of external knowledge and cannot work without human intervention. On the other side, bottom-up approaches ground on the distributional hypothesis [15] for language modeling, according to which the meaning of words depends on the context in which they occur, in some textual content. The resulting strategy is therefore unsupervised, requiring a corpora of textual documents for training as large as possible. Approaches based on the distributional hypothesis, referred to as discriminative models, behave as word embeddings techniques where each term (and document) becomes a point in the vector space. They substitute the term-document matrix typical of Vector Space Model with a term-context matrix on which they apply dimensionality reduction techniques such as Latent Semantic Indexing (LSI) [8] and the more scalable and incremental Random Indexing (RI) [38]. The latter has been involved in [22] and [23] to define the so called enhanced Vector Space Model (eVSM) for content-based RS, where user’s profile is incrementally built summing the features vectors representing documents liked by the user and a negation operator is introduced to take into account also negative preferences.

Word embedding techniques are not limited to LSI and RI. The word2vec strategy has been recently presented in [19] and [20], and to the best of our knowledge, has been applied to item recommendations in a few works [21, 28]. In particular, [21] is an empirical evaluation of LSI, RI and word2vec to make content-based movie recommendation exploiting textual information from Wikipedia, while [28] deals with check-in venue (location) recommendations and adds a non-textual feature, the past check-ins of the user. They both draw the conclusion that word2vec techniques are promising for the recommendation task. Finally there is a single example of product embedding [14], namely prod2vec, which operates on the artificial graph of purchases, treating a purchase sequence as a “sentence” and products within the sequence as words.

3. APPROACH

In our approach, we adapt neural language models for RDF graph embeddings. Such approaches take advantage of the word order in text documents, explicitly modeling the assumption that closer words in the word sequence are statistically more dependent. In the case of RDF graphs, we follow the approach sketched in [37], considering entities and relations between entities instead of word sequences. Thus, in order to apply such approaches on RDF graph data, we have to transform the graph data into sequences of entities, which can be considered as sentences. After the graph is converted into a set of sequences of entities, we can train the same neural language models to represent each entity in the RDF graph as a vector of numerical values in a latent feature space. Such entity vectors can be directly used in a content-based recommender system.

3.1 RDF Graph Sub-Structures Extraction

We propose random graph walks as an approach for con-
Definition 1. An RDF graph is a graph $G = (V, E)$, where $V$ is a set of vertices, and $E$ is a set of directed edges.

The objective of the conversion functions is for each vertex $v \in V$ to generate a set of sequences $S_v$, where the first token of each sequence $s \in S_v$ is the vertex $v$ followed by a sequence of tokens, which might be edges, vertices, or any substructure extracted from the RDF graph, in an order that reflects the relations between the vertex $v$ and the rest of the tokens, as well as among those tokens.

In this approach, for a given graph $G = (V, E)$, for each vertex $v \in V$ we generate all graph walks $P_v$ of depth $d$ rooted in the vertex $v$. To generate the walks, we use the breadth-first algorithm. In the first iteration, the algorithm generates paths by exploring the direct outgoing edges of the root node $v_r$. The paths generated after the first iteration will have the following pattern $v_r \rightarrow c_1 \rightarrow v_1$, where $r \in E(v_r)$. In the second iteration, for each of the previously explored edges the algorithm visits the connected vertices. The paths generated after the second iteration will follow the following pattern $v_r \rightarrow c_1 \rightarrow \cdots \rightarrow v_1$. The algorithm continues until $d$ iterations are reached. The final set of sequences for the given graph $G$ is the union of the sequences of all the vertices $\bigcup_{v \in V} P_v$.

3.2 Neural Language Models – word2vec

Until recently, most of the Natural Language Processing systems and techniques treated words as atomic units, representing each word as a feature vector using a one-hot representation, where a word vector has the same length as the size of a vocabulary. In such approaches, there is no notion of semantic similarity between words. While such approaches are widely used in many tasks due to their simplicity and robustness, they suffer from several drawbacks, e.g., high dimensionality and severe data sparsity, which limit the performance of such techniques. To overcome such limitations, neural language models have been proposed, inducing low-dimensional, distributed embeddings of words by means of neural networks. The goal of such approaches is to estimate the likelihood of a specific sequence of words appearing in a corpus, explicitly modeling the assumption that closer words in the word sequence are statistically more dependent.

While some of the initially proposed approaches suffered from inefficient training of the neural network models, with the recent advancements in the field several efficient approaches has been proposed. One of the most popular and widely used is the word2vec neural language model [19, 20]. Word2vec is a particularly computationally-efficient two-layer neural net model for learning word embeddings from raw text. There are two different algorithms, the Continuous Bag-of-Words model (CBOW) and the Skip-Gram model.

3.2.1 Continuous Bag-of-Words Model

The CBOW model predicts target words from context words within a given window. The input layer is comprised from all the surrounding words for which the input vectors are retrieved from the input weight matrix, averaged, and projected in the projection layer. Then, using the weights from the output weight matrix, a score for each word in the vocabulary is computed, which is the probability of the word being a target word. Formally, given a sequence of training words $w_1, w_2, w_3, \ldots, w_T$, and a context window $c$, the objective of the CBOW model is to maximize the average log probability:

$$\frac{1}{T} \sum_{t=1}^{T} \log p(w_t | w_{t-c} \cdots w_{t+c}),$$

where the probability $p(w_t | w_{t-c} \cdots w_{t+c})$ is calculated using the softmax function:

$$p(w_t | w_{t-c} \cdots w_{t+c}) = \frac{\exp(\bar{v}_w^T \nu_{w_t})}{\sum_{w=1}^{V} \exp(\bar{v}_w^T \nu_{w})},$$

where $\bar{v}_w$ is the output vector of the word $w$, $V$ is the complete vocabulary of words, and $\bar{v}$ is the averaged input vector of all the context words:

$$\bar{v} = \frac{1}{2c} \sum_{-c \leq j \leq c, j \neq 0} v_{w_{t+j}}$$

3.2.2 Skip-Gram Model

The Skip-Gram model does the inverse of the CBOW model and tries to predict the context words from the target words. More formally, given a sequence of training words $w_1, w_2, w_3, \ldots, w_T$, and a context window $c$, the objective of the skip-gram model is to maximize the following average log probability:

$$\frac{1}{T} \sum_{t=1}^{T} \sum_{-c \leq j \leq c, j \neq 0} \log p(w_{t+j} | w_t),$$

where the probability $p(w_{t+j} | w_t)$ is calculated using the softmax function:

$$p(w_t | w_{t-c} \cdots w_{t+c}) = \frac{\exp(\nu_{w_t}^T \bar{v}_w)}{\sum_{w=1}^{V} \exp(\nu_{w_t}^T \bar{v}_w)},$$

where $\nu_w$ and $\nu_w'$ are the input and the output vector of the word $w$, and $V$ is the complete vocabulary of words.

In both cases, calculating the softmax function is computationally inefficient, as the cost for computing is proportional to the size of the vocabulary. Therefore, two optimization techniques have been proposed, i.e., hierarchical softmax and negative sampling [20]. The empirical studies show that in most cases negative sampling leads to better performances than hierarchical softmax, which depends on the selected negative samples, but it has higher runtime.

Once the training is finished, semantically similar words appear close to each other in the feature space. Furthermore, basic mathematical functions can be performed on the vectors, to extract different relations between the words.

4. EVALUATION

We evaluate different variants of our approach on two distinct datasets, and compare them to common approaches for creating content-based item representations from LOD and with state of the art collaborative approaches. Furthermore, we investigate the use of two different LOD datasets as background knowledge, i.e., DBpedia and Wikidata.

4.1 Datasets

In order to test the effectiveness of our proposal, we evaluate it in terms of ranking accuracy and aggregate diversity on two datasets belonging to different domains, i.e. MovieLens 1M for movies and LibraryThing5 for books. The

http://grouplens.org/datasets/movielens/

5https://www.librarything.com/
former contains 1 million 1-5 stars ratings from 6,040 users on 3,883 movies. The LibraryThing dataset contains more than 2 millions ratings from 7,564 users on 39,515 books. As there are many duplicated ratings in the dataset, when a user has rated more than once the same item, we select her last rating. This choice brings to have 626,000 ratings in the range from 1 to 10. The user-item interactions contained in the datasets are enriched with side information thanks to the item mapping and linking to DBpedia technique detailed in [27], whose dump is available at http://sisinflab.poliba.it/semanticweb/lod/recsys/datasets/. In the attempt to reduce the popularity bias from our final evaluation we decided to remove the top 1% most popular items from both datasets [5]. Moreover we keep out, from LibraryThing, users with less than five ratings and items rated less than five times, and to have a dataset characterized by lower sparsity we retain for Movielens only users with at least fifty ratings, as already done in [10]. Table 1 contains the final statistics for our datasets.

<table>
<thead>
<tr>
<th></th>
<th>Movielens</th>
<th>LibraryThing</th>
</tr>
</thead>
<tbody>
<tr>
<td>Number of users</td>
<td>4,186</td>
<td>7,149</td>
</tr>
<tr>
<td>Number of items</td>
<td>3,196</td>
<td>4,541</td>
</tr>
<tr>
<td>Number of ratings</td>
<td>822,597</td>
<td>352,123</td>
</tr>
<tr>
<td>Data sparsity</td>
<td>93.85%</td>
<td>98.90%</td>
</tr>
</tbody>
</table>

Table 1: Statistics about the two datasets

4.1.1 RDF Embeddings

As RDF datasets we use DBpedia and Wikidata.

We use the English version of the 2015-10 DBpedia dataset, which contains 4,641,890 instances and 1,369 mapping-based properties. In our evaluation we only consider object properties, and ignore the data properties and literals.

For the Wikidata dataset we use the simplified and derived RDF dumps from 2016-03-28\(^8\). The dataset contains 17,340,659 entities in total. As for the DBpedia dataset, we only consider object properties, and ignore the data properties and literals.

4.2 Evaluation Protocol

As evaluation protocol for our comparison, we adopted the all unrated items methodology presented in [41] and already used in [10]. Such methodology asks to predict a score for each item not rated by a user, irrespective of the existence of an actual rating, and to compare the recommendation list with the test set.

The metrics involved in the experimental comparison are precision, recall and nDCG as accuracy metrics, and catalog coverage and Gini coefficient for the aggregate diversity. precision@N represents the fraction of relevant items in the top-N recommendations. recall@N indicates the fraction of relevant items, in the user test set, occurring in the top-N list. As relevance threshold, we set 4 for Movielens and 8 for LibraryThing, as previously done in [10]. Although precision and recall are good indicators to evaluate the accuracy of a recommendation engine, they are not rank-sensitive. nDCG@N [2] instead takes into account also the position in the recommendation list, being defined as

\[
\text{nDCG@N} = \frac{1}{\text{idCG}} \sum_{i=1}^{N} \frac{2^{\text{rel}(u,i)} - 1}{\log(1 + i)}
\]

where \text{rel}(u,i) is a boolean function representing the relevance of item \text{i} for user \text{u} and iDCG is a normalization factor that sets nDCG@N value to 1 when an ideal ranking is returned [2]. As suggested in [41] and set up in [10], in the computation of nDCG@N we fixed a default “neutral” value for those items with no ratings, i.e. 3 for Movielens and 5 for LibraryThing.

Providing accurate recommendations has been recognized as just one of the main task a recommender system must be able to perform. We therefore evaluate the contribution of our latent features in terms of aggregate diversity, and more specifically by means of catalog coverage and Gini coefficient [1]. The catalog coverage represents the percentage of available candidate items recommended at least once. It is an important quality dimension for both user and business perspective [13], since it exhibits the capacity to not settle just on a subset of items (e.g. the most popular). This metric however should be supported by a distribution metric which has to show the ability of a recommendation engine to equally spread out the recommendations across all users. Gini coefficient [1] is used for this purpose, since it measures the concentration degree of top-N recommendations across items and is defined as

\[
\text{Gini} = 2 \sum_{i=0}^{n} \left( \frac{n + 1 - i}{n + 1} \right) \cdot \frac{\text{rec}(i)}{\text{total}}
\]

In Equation (7), \text{n} is the number of candidate items available for recommendation, \text{total} represents the total number of top-N recommendations made across all users, and \text{rec}(i) is the number of users to whom item \text{i} has been recommended. Gini coefficient gives therefore an idea of the "equity" in the distribution of the items. It is worth to remark that we are following the notion given in [1], where the complement of the standard Gini coefficient is used, so that higher values correspond to more balanced recommendations.

4.3 Experimental Setup

The first step of our approach is to convert the RDF graphs into a set of sequences. Therefore, to extract the entities embeddings for the large RDF datasets, we use only random graph walks entity sequences. More precisely, we follow the approach presented in [32] to generate only a limited number of random walks for each entity. For DBpedia, we experiment with 500 walks per entity with depth of 4 and 8, while for Wikidata, we use only 200 walks per entity with depth of 4. Additionally, for each entity in DBpedia and Wikidata, we include all the walks of depth 2, i.e., direct outgoing relations. We use the corpora of sequences to build both CBOW and Skip-Gram models with the following parameters: window size = 5; number of iterations = 5; negative sampling for optimization; negative samples = 25; with average input vector for CBOW. We experiment with 200 and 500 dimensions for the entities’ vectors. All the models are publicly available\(^7\).

We compare our approach to several baselines. For generating the data mining features, we use three strategies that


\(^7\)http://data.dws.informatik.uni-mannheim.de/rdf2vec/
take into account the direct relations to other resources in the graph [30], and two strategies for features derived from graph sub-structures [7]:

- Features derived from specific relations. In the experiments we use the relations rdf:type (types), and dcterms:subject (categories) for datasets linked to DBpedia.
- Features derived from generic relations, i.e., we generate a feature for each incoming (rel in) or outgoing relation (rel out) of an entity, ignoring the value of the relation.
- Features derived from generic relations-values, i.e., we generate feature for each incoming (rel-vals in) or outgoing relation (rel-vals out) of an entity including the value of the relation.
- Kernels that count substructures in the RDF graph around the instance node. These substructures are explicitly generated and represented as sparse feature vectors.

- The Weisfeiler-Lehman (WL) graph kernel for RDF [7] counts full subtrees in the subgraph around the instance node. This kernel has two parameters, the subgraph depth \( d \) and the number of iterations \( h \) (which determines the depth of the subtrees). We use \( d = 1 \) and \( h = 2 \) and therefore we will indicate this strategy as WL12.
- The Intersection Tree Path kernel for RDF [7] counts the walks in the subtree that span from the instance node. Only the walks that go through the instance node are considered. We will therefore refer to it as the root Walk Count (WC) kernel. The root WC kernel has one parameter: the length of the paths \( l \), for which we test 2. This strategy will be denoted accordingly as WC2.

The strategies for creating propositional features from Linked Open Data are implemented in the RapidMiner LOD extension\(^8\) [31, 35].

### 4.4 Results

The target of the experimental section of this paper is twofold. On the one hand, we want to prove that the latent features we extracted are able to subsume the other kind of features in terms of accuracy and aggregate diversity. On the other hand, we aim at qualifying our strategies as valuable means for the recommendation task, through a first comparison with state of the art approaches. Both goals are pursued implementing an item-based K-nearest-neighbor method, hereafter denoted as ItemKNN, with cosine similarity among features vectors. Formally, this method determines similarities between items through cosine similarity between relative vectors and then selects a subset of them – the neighbors – for each item, that will be used to estimate the rating of user \( u \) for a new item \( i \) as follows:

\[
r^*(u, i) = \sum_{j \in \text{ratedItems}(u)} \text{cosineSim}(j, i) \cdot r_{u,j}
\]

where \( \text{ratedItems}(u) \) is the set of items already evaluated by user \( u \), \( r_{u,j} \) indicates the rating for item \( j \) by user \( u \) and \( \text{cosineSim}(j, i) \) is the cosine similarity score between items \( j \) and \( i \). In our experiments, the size of the considered neighbourhood is limited to 5. The computation of recommendations has been done with the publicly available library RankSys\(^9\). All the results have been computed @10, that is considering the top-10 lists recommended to the users: precision, recall and nDCG are computed for each user and then averaged across all users, while diversity metrics are global measures.

Tables 2 and 3 contain the values of precision, recall and nDCG, respectively for Movielens and LibraryThing, for each kind of features we want to test. The best approach for both datasets is retrieved with a Skip-Gram model and with a size of 200 for vectors built upon DBpedia. For the sake of truth, on the Movielens dataset the highest value of precision is achieved using vector size of 500, but the size 200 is prevalent according to the F1 measure, i.e. the harmonic mean of precision and recall. A substantial difference however concerns the exploratory depth of the random walks, since for Movielens the results related to depth 4 outdo those computed with depth 8, while the tendency is reversed for LibraryThing. The advantage of the Skip-Gram model over the CBOW is a constant both on DBpedia and Wikidata. Moreover, the employment of the Wikidata RDF dataset turns out to be more effective for LibraryThing, where the Skip-Gram vectors with depth 4 exceeds the corresponding DBpedia vectors. Moving to the features extracted from direct relations, the contribution of the “categories” stands clearly out, together with relations-values “rel-vals”, especially when just incoming relations are considered. The extraction of features from graph structures, i.e. WC2 and WL12 approaches, seems not to provide significant advantages to the recommendation algorithm.

To point out that our latent features are able to capture the structure of the RDF graph, placing closely semantically similar items, we provide some examples of the neighbouring sets retrieved using our graph embeddings technique and used within the ItemKNN. Table 4 is related to movies and displays that neighboring items are highly relevant and close to the query item, i.e. the item for which neighbors are searched for.

To further analyse the semantics of the vector representations, we employ Principal Component Analysis (PCA) to project the “high”-dimensional entities’ vectors in a two dimensional feature space, or 2D scatter plot. For each of the query movies in Table 4 we visualize the vectors of the 5 nearest neighbors as shown in Figure 1. The figure illustrates the ability of the model to automatically cluster the movies.

The impact on the aggregate diversity. As a further validation of the interactivity of our latent features for recommendation task, we report the performances of the ItemKNN approach in terms of aggregate diversity. The relation between accuracy and aggregate diversity has gained the attention of researchers in the last few years and is generally characterized as a trade-off [1]. Quite surprisingly, however, the increase in accuracy, shown in Tables 2 and 3, seems not to rely on a concentration on a subset of items, e.g. the most

\(^8\)http://dws.informatik.uni-mannheim.de/en/research/rapidminer-lod-extension

\(^9\)http://ranksys.org/
Table 2: Results of the ItemKNN approach on Movielens dataset. P and R stand respectively for precision and recall, SG indicates the Skip-Gram model, and DB and WD represent DBpedia and Wikidata respectively.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>P@10</th>
<th>R@10</th>
<th>nDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2vec CBOW 200 4</td>
<td>0.03893</td>
<td>0.02167</td>
<td>0.30782</td>
</tr>
<tr>
<td>DB2vec CBOW 500 4</td>
<td>0.03663</td>
<td>0.02088</td>
<td>0.30557</td>
</tr>
<tr>
<td>DB2vec SG 200 4</td>
<td>0.05681</td>
<td>0.03119</td>
<td>0.31828</td>
</tr>
<tr>
<td>DB2vec SG 500 4</td>
<td>0.05786</td>
<td>0.0304</td>
<td>0.31726</td>
</tr>
<tr>
<td>DB2vec CBOW 200 8</td>
<td>0.01064</td>
<td>0.00548</td>
<td>0.29245</td>
</tr>
<tr>
<td>DB2vec CBOW 500 8</td>
<td>0.01137</td>
<td>0.00567</td>
<td>0.29289</td>
</tr>
<tr>
<td>DB2vec SG 200 8</td>
<td>0.04424</td>
<td>0.02693</td>
<td>0.30997</td>
</tr>
<tr>
<td>DB2vec SG 500 8</td>
<td>0.02191</td>
<td>0.01478</td>
<td>0.29863</td>
</tr>
<tr>
<td>WD2vec CBOW 200 4</td>
<td>0.01217</td>
<td>0.00596</td>
<td>0.29362</td>
</tr>
<tr>
<td>WD2vec CBOW 500 4</td>
<td>0.01027</td>
<td>0.00427</td>
<td>0.29211</td>
</tr>
<tr>
<td>WD2vec SG 200 4</td>
<td>0.02444</td>
<td>0.01246</td>
<td>0.29967</td>
</tr>
<tr>
<td>WD2vec SG 500 4</td>
<td>0.01033</td>
<td>0.00514</td>
<td>0.28864</td>
</tr>
<tr>
<td>categories</td>
<td>0.0305</td>
<td>0.02093</td>
<td>0.30444</td>
</tr>
<tr>
<td>rel in</td>
<td>0.01722</td>
<td>0.00858</td>
<td>0.29183</td>
</tr>
<tr>
<td>rel out</td>
<td>0.02844</td>
<td>0.01607</td>
<td>0.30274</td>
</tr>
<tr>
<td>rel vals in</td>
<td>0.03883</td>
<td>0.02293</td>
<td>0.29411</td>
</tr>
<tr>
<td>rel vals out</td>
<td>0.01279</td>
<td>0.00971</td>
<td>0.29333</td>
</tr>
<tr>
<td>WC2</td>
<td>0.0101</td>
<td>0.00288</td>
<td>0.28977</td>
</tr>
</tbody>
</table>

Table 3: Results of the ItemKNN approach on LibraryThing dataset.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>P@10</th>
<th>R@10</th>
<th>nDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2vec CBOW 200 4</td>
<td>0.05127</td>
<td>0.11777</td>
<td>0.21244</td>
</tr>
<tr>
<td>DB2vec CBOW 500 4</td>
<td>0.05065</td>
<td>0.11557</td>
<td>0.21039</td>
</tr>
<tr>
<td>DB2vec SG 200 4</td>
<td>0.05719</td>
<td>0.12763</td>
<td>0.22050</td>
</tr>
<tr>
<td>DB2vec SG 500 4</td>
<td>0.05811</td>
<td>0.12864</td>
<td>0.22116</td>
</tr>
<tr>
<td>DB2vec CBOW 200 8</td>
<td>0.00836</td>
<td>0.02334</td>
<td>0.14147</td>
</tr>
<tr>
<td>DB2vec CBOW 500 8</td>
<td>0.00813</td>
<td>0.02335</td>
<td>0.14257</td>
</tr>
<tr>
<td>DB2vec SG 200 8</td>
<td>0.07681</td>
<td>0.17769</td>
<td>0.25234</td>
</tr>
<tr>
<td>DB2vec SG 500 8</td>
<td>0.07446</td>
<td>0.17430</td>
<td>0.24809</td>
</tr>
<tr>
<td>categories</td>
<td>0.01854</td>
<td>0.04535</td>
<td>0.16064</td>
</tr>
<tr>
<td>rel in</td>
<td>0.04577</td>
<td>0.10219</td>
<td>0.20198</td>
</tr>
<tr>
<td>rel out</td>
<td>0.04118</td>
<td>0.09655</td>
<td>0.19449</td>
</tr>
<tr>
<td>rel vals in</td>
<td>0.06416</td>
<td>0.14101</td>
<td>0.22574</td>
</tr>
<tr>
<td>rel vals out</td>
<td>0.06058</td>
<td>0.13662</td>
<td>0.22615</td>
</tr>
<tr>
<td>WC2</td>
<td>0.00155</td>
<td>0.00306</td>
<td>0.12858</td>
</tr>
<tr>
<td>WL12</td>
<td>0.00155</td>
<td>0.00306</td>
<td>0.12857</td>
</tr>
</tbody>
</table>

Table 4: Examples of K-nearest-neighbor sets on Movielens, for the Skip-Gram model with depth of 4 and size vectors 200, on DBpedia.

<table>
<thead>
<tr>
<th>Query Movie</th>
<th>K Nearest Neighbours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Batman</td>
<td>Batman Forever, Batman Returns, Batman &amp; Robin, Superman IV: The Quest for Peace, Dick Tracy</td>
</tr>
<tr>
<td>Bambi</td>
<td>Cinderella, Dumbo, 101 Dalmatians, Pinocchio, Lady and the Tramp</td>
</tr>
</tbody>
</table>

Figure 1: Two-dimensional PCA projection of the 200-dimensional Skip-gram vectors of movies in Table 4.

Comparison with state of the art collaborative approaches. It is a quite common belief in the RS field that using pure content-based approaches would not be enough to provide accurate suggestions and that the recommendation engines must ground on collaborative information too. This motivated us to explicitly compare the best approaches built on graph embeddings technique with the well-known state of the art collaborative recommendation algorithms listed be-
Table 5: Methods comparison in terms of aggregate diversity on the Movielens dataset. Coverage stands for catalog coverage and Gini for Gini coefficient.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Coverage</th>
<th>Gini</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2vec SG 200 4</td>
<td>0.35198</td>
<td>0.07133</td>
</tr>
<tr>
<td>WD2vec CBOW 200 4</td>
<td>0.27749</td>
<td>0.04052</td>
</tr>
<tr>
<td>categories</td>
<td>0.29798</td>
<td>0.04714</td>
</tr>
</tbody>
</table>

Table 6: Methods comparison in terms of aggregate diversity on the LibraryThing dataset.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>Coverage</th>
<th>Gini</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2vec SG 200 8</td>
<td>0.76386</td>
<td>0.29534</td>
</tr>
<tr>
<td>WD2vec SG 200 4</td>
<td>0.73037</td>
<td>0.28525</td>
</tr>
<tr>
<td>categories</td>
<td>0.7246</td>
<td>0.26409</td>
</tr>
</tbody>
</table>

Table 7: Comparison with state of the art collaborative approaches on Movielens.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>P@10</th>
<th>R@10</th>
<th>nDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2vec SG 200 4</td>
<td>0.0568</td>
<td>0.0312</td>
<td>0.3183</td>
</tr>
<tr>
<td>MF</td>
<td>0.2522</td>
<td>0.1307</td>
<td>0.4427</td>
</tr>
<tr>
<td>PopRank</td>
<td>0.1673</td>
<td>0.0787</td>
<td>0.3910</td>
</tr>
<tr>
<td>BPRMF</td>
<td>0.2522</td>
<td>0.1307</td>
<td>0.4427</td>
</tr>
<tr>
<td>SLIM</td>
<td>0.2632</td>
<td>0.1474</td>
<td>0.4599</td>
</tr>
<tr>
<td>RankMF</td>
<td>0.1417</td>
<td>0.0704</td>
<td>0.3736</td>
</tr>
</tbody>
</table>

Table 8: Comparison with state of the art collaborative approaches on LibraryThing.

<table>
<thead>
<tr>
<th>Strategy</th>
<th>P@10</th>
<th>R@10</th>
<th>nDCG@10</th>
</tr>
</thead>
<tbody>
<tr>
<td>DB2vec SG 200 8</td>
<td>0.0768</td>
<td>0.1777</td>
<td>0.2523</td>
</tr>
<tr>
<td>MF</td>
<td>0.0173</td>
<td>0.0209</td>
<td>0.1423</td>
</tr>
<tr>
<td>PopRank</td>
<td>0.0397</td>
<td>0.0472</td>
<td>0.1598</td>
</tr>
<tr>
<td>BPRMF</td>
<td>0.0449</td>
<td>0.0764</td>
<td>0.1858</td>
</tr>
<tr>
<td>SLIM</td>
<td>0.0543</td>
<td>0.0988</td>
<td>0.2317</td>
</tr>
<tr>
<td>RankMF</td>
<td>0.0369</td>
<td>0.0459</td>
<td>0.1714</td>
</tr>
</tbody>
</table>

5. CONCLUSION

In this paper, we have presented an approach for learning low-dimensional real-valued representations of entities in RDF graphs, in a completely domain independent way. We have first converted the RDF graphs into a set of sequences using graph walks, which are then used to train neural language models. In the experimental section we have shown that a content-based RS relying on the similarity between items computed according to our latent features vectors, outdo the same kind of system but grounding on explicit features (e.g. types, categories,...) or features generated with the use of kernels, from both perspectives of accuracy and aggregate diversity. Our purely content-based system has been further compared to state of the arts collaborative approaches for rating prediction and item ranking, giving outstanding results on a dataset with a realistic sparsity degree.

As future work, we intend to introduce the features vectors deriving from the graph embeddings technique within a hybrid recommender system in order to get a fair comparison against state of the art hybrids approaches such as SPRank [10] and BRP-SSLIM [25]. In this perspective we could take advantage of the Factorization Machines [33], general predictor working with any features vector, that combine Support Vector Machines and factorization models. We aim to extend the evaluation to additional metrics, such as the individual diversity [44, 9], and to provide a deeper insight into cold-start users, i.e. users with a small interaction with the system for whom the information inference is difficult to draw and that generally benefit most of content “infusion”.

6. REFERENCES


...


