Auto-Encoding User Ratings via Knowledge Graphs in Recommendation Scenarios

Vito Bellini
Polytechnic University of Bari
Via E. Orabona, 4
Bari, Italy 70126
vito.bellini@poliba.it

Vito Walter Anelli
Polytechnic University of Bari
Via E. Orabona, 4
Bari, Italy 70126
vitowalter.anelli@poliba.it

Tommaso Di Noia
Polytechnic University of Bari
Via E. Orabona, 4
Bari, Italy 70126
tommaso.dinoia@poliba.it

Eugenio Di Sciascio
Polytechnic University of Bari
Via E. Orabona, 4
Bari, Italy 70126
eugenio.disciascio@poliba.it

ABSTRACT

In the last decade, driven also by the availability of an unprecedented computational power and storage capabilities in cloud environments we assisted to the proliferation of new algorithms, methods, and approaches in two areas of artificial intelligence: knowledge representation and machine learning. On the one side, the generation of a high rate of structured data on the Web led to the creation and publication of the so-called knowledge graphs. On the other side, deep learning emerged as one of the most promising approaches in the generation and training of models that can be applied to a wide variety of application fields. More recently, autoencoders have proven their strength in various scenarios, playing a fundamental role in unsupervised learning. In this paper, we instigate how to exploit the semantic information encoded in a knowledge graph to build connections between units in a Neural Network, thus leading to a new method, SEM-AUTO, to extract and weigh semantic features that can eventually be used to build a recommender system. As adding content-based side information may mitigate the cold user problems, we tested how our approach behave in the presence of a few rating from a user on the Movielens 1M dataset and compare results with BPRSLIM.

KEYWORDS

Recommender Systems, Deep Learning, Autoencoders, Knowledge graphs, Linked Open Data, DBpedia

1 INTRODUCTION

Recommender systems (RS) are nowadays used in many of the services we daily access in order to provide a personalized experience in the browsing and selection of items in a catalogue. Their success strongly depends on how they can identify and exploit user tastes and preferences while suggesting potentially relevant items. RS mainly rely on the rates users provide to items, to predict the importance for unrates ones. Over the years, collaborative filtering (CF) approaches have shown to be very effective in suggesting accurate recommendation especially in the presence of many data in the user-ratings matrix while they suffer in situations of very sparse matrices. CF fail in providing good recommendation in situations where we have users who rated a few items (cold users) and items with a few ratings (cold items). This latter problem is mitigated in case the recommendation engine adopts a content-based (CB) approach where characteristics of the items are exploited to find those similar to the ones rated by the user in the past. Combining CF and CB usually leads to obtain better recommendation results [12][15][20]. As a matter of fact, adding side information to a collaborative filtering approach has proven to be more effective while computing recommendations to the end user [18]. Recently, among the ideal candidates to get side information to be injected in recommender systems we surely find knowledge graphs[1]. On the one hand, information encoded in such structures is an excellent mine of meaningful data that can be exploited to describe and categorize items in a catalogue. Among the various knowledge graph available online, we have those belonging to the Linked Open Data (LOD) cloud such as DBpedia [1] or Wikidata [27]. There, encyclopedic information is encoded in terms of RDF triples $\langle subject, predicate, object \rangle$ thus creating a huge interconnected graph of knowledge.

After being successfully adopted to cope with many task related to artificial intelligence such as image recognition or natural language processing, deep learning techniques are rapidly entering the world of recommender systems [3]. Built around the basic notion of a neural networks (NN), over the years many new techniques and approaches have been developed under the deep learning umbrella.

---

[1]https://googleblog.blogspot.it/2012/05/introducing-knowledge-graph-things-not.html

Among them, autoencoders are a particular configuration of a NN whose main aim is that of training a model able to reproduce the inputs of a system. They have been successfully adopted for feature selection [26] as well as a generative model of data [13].

In this paper we present a novel way to build a user profile by using features computed by means of an Autoencoder. The semantic information of classes and categories of a knowledge graph is exploited to draw the topology of the underlying neural network. Each class and category associated to an item is represented by a neuron in the hidden layer of the network which autoencodes the ratings of the users. After training the resulting neural network, the weights computed as inputs of the neurons in the hidden layer are then interpreted as the importance of the corresponding feature in the rating process from the users. Eventually, the vectors of feature weights are used to estimate the utility associated to items unknown to (unrated by) the user thus computing a top-N recommendation list.

The remainder of this paper is structured as follows: in the next section we report about related work on the usage of autoencoders and deep learning techniques as well as Linked Open Data for recommendation tasks. Then we introduce the notion of autoencoder and that of semantic-autoencoder. In Section 4 we describe our recommendation model followed by the description of the experimental setting and of the metrics we used in the evaluation. Conclusions and Future Work close the paper.

2 RELATED WORK

Autoencoders and Deep Larning. Autoencoders have recently attracted attention in Recommender System community. In [30] the authors utilize the idea of Denoising Auto-Encoders for learning from corrupted inputs. The proposed approach assumes that observed user-item interactions are a corrupted version of user's preferences. The model then learns a latent representation of corrupted user-item preferences that can lead to a better reconstructed input. By training on corrupted data we can recover co-preference patterns. The authors show that this is an effective approach in collaborative filtering scenarios. [25] introduces a CF approach based on Stacked Denoising Autoencoders in order to learn a non-linear representation of the users-items in order to alleviate the cold start problem by integrating side information. So they came up with a Hybrid Recommender System. The main idea of this work is to use an hidden layer (Autoencoder's bottleneck) of size k « N (number of features) to let the network find the low-dimensional representation to feed the a Deep Neural Network. Experimental results show that side information bring only a small improvement if an item have many rates. A pure CF approach has been settled with autoencoders in [23]. They compare item-based autoencoding and user-based autoencoding, outperforming all the baselines in terms of RMSE. A Hybrid Recommender System is presented in [5]. Here the authors side information to address the problem of the sparse user-item matrix, then jointly learn users and items' latent factors from side information and collaborative filtering from the rating matrix. Other works like [28] are focused on learning users preferences in a high-dimensional semantic latent space, with the advantage to be able to recommend items using content that describes the items. As the authors say, describing items in a semantic space

provide the intersubstitutability of items or, in other words, items may be substituted by nearby items in such a space. Deep Learning is used in [29] to tackle CF's sparsity problem by integrating auxiliary information, such as item content information. Ratings and side information can be used together, where a collaborative topic regression method is capable to learn a latent representation. Then, a collaborative deep learning model jointly performs representation for content information and collaborative filtering for the ratings matrix. Mapping user and items to a latent space, as done in [6] seems to be a good approach to address the recommendation quality in content-based recommender systems. Using a rich feature set from different domains to represent users and items, allows the model to provide quality recommendation across all the domains, as well as having a semantically rich user latent feature vector. **Linked Open Data.** Many approaches have been proposed for exploiting information extracted from Linked Open Data in recommendation tasks. One of the very first proposal in this direction is [9] where the authors introduced for the first time the idea of using Linked Open Data in a recommender system. A system for recommending artists and music using DBpedia was presented in [21]. Several other approaches have been proposed afterwards such as a knowledge-based framework leveraging DBpedia for cross-domain recommendation task [7, 8], a content-based context-aware method able to adopt a semantic representation based on a combination of distributional semantics and entity linking techniques [17], a hybrid graph-based algorithm based on learning-to-rank method and path-based features extracted from heterogeneous information networks built upon DBpedia and collaborative information [19]. To the best of our knowledge, the only works dealing with automated feature selection from knowledge graphs are [16, 22]. While the former analyzes the performance of a recommender system after a feature selection based on classical statistical methods such as Information Gain, Chi Squared etc., the latter adopt a technique based on ontological schema summarization.

3 AUTOENCODERS

Artificial Neural Networks (ANNs) are computational models originally proposed to catch underlying relationships in a set of data by using positive and negative examples fed into the network (supervised learning). ANNs are composed by an input layer, one o more hidden layers and an output layer. Every layer is made of units and every layer is fully-connected, it means that every unit is connected with all the units in the following layer. Connections between units are usually initialized with a random weight. In a conventional neural network, the task is then predicting a target vector y from an input vector x.

Autoencoders are ANNs that apply backpropagation algorithm, setting the target values to be equal to the inputs in an unsupervised fashion. Roughly, in an autoencoder network one tries to "predict" x from x. The idea is to first compress (encode) the input vector to fit in a smaller representation, and then try to reconstruct (decode) it back. This means that the model learns in the hidden layers, a representation of the input and therefore a latent representation of the knowledge behind the input data. The task performed by autoencoders is quite similar to that of a Principal Components Analysis (PCA) operation. We suppose to have a two layers ANN,

with only one hidden layer using linear functions, the number of hidden units is less than the number of input units, the input and output layers composed by the same number of neurons, and we want the output to mimic as close as possible the input. After the training, the hidden layer will be a representation of the input in a space with a number of dimensions denoted by the number of neurons of the hidden layer. It is noteworthy that the overall training is proportional to the number of training cases and this operation is more inefficient with respect to PCA. Moreover, if we want to represent the input non-linearly (using curved dimensions) the training of the neural network will become more efficient compared to other methods because its complexity remains linear in the number of training cases. We can extend the neural network by injecting more hidden layers before and after the original one. The new subnetwork added before the original hidden layer will be trained to encode the input in the new latent space while the one we added after will decode the data in the original space. As known initializing weights in deep NN can be a non trivial task and this problem can be alleviated performing a pre-training operation or using the same methods by Echo-state networks. One of the first and more famous autoencoder [11] encoded 784 pixel images in a 30 dimensions space, initializing the weights using Restricted Boltzmann Machines (RBM).

Autoencoders can obviously be employed for entities different than images. One of the first examples was the Latent Semantic Analysis [14], in which the basic idea was to exploit the PCA to word count vectors to extract similarities between documents. This is a very common task in information retrieval that has been proved to be much better accomplished using autoencoders [10]. We can trivially convert documents into bags-of-words and generate the word count vectors. Extracting similarities in the high dimensional space of the words can be a really expensive operation and can be addressed using a much more compact representation using autoencoders for both documents and queries. The most important difference with the previous example is that, in this case, what we want to predict is a probability distribution representing the chance of encountering a specific word in a document, and this affects the model in terms of cost function and optimization objectives.

3.1 Semantics-aware autoencoders

Autoencoders, just like other methods for latent representation, are unable to provide an explanation for the latent factors they provide. To address this issue, we propose to give a meaning to connection with the hidden layer and to its neurons by exploiting semantic information explicitly available in knowledge graphs. The main idea of SEM-AUTO approach is to map connections between units from layer i to layer i+1, reflecting the nodes available in a knowledge-base graph as shown in figure 1. In particular, we mapped the autoencoder network topology with the categorical information related to items rated by users. Our assumption is that most of the valuable information encoded in a knowledge graph is represented by categorical information. We feed such a network with the item's rates provided by the user, normalized by [0,1].

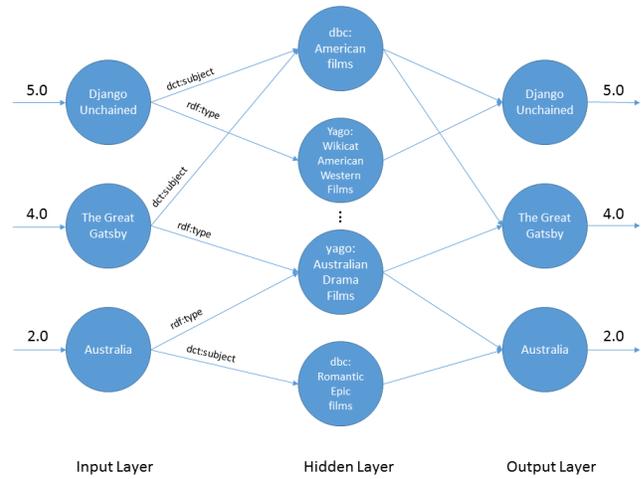It is worth noticing that the neural network we build is not fully connected.

Figure 1: Architecture of a semantic autoencoder.

In order to make the results of the network consistent during multiple trainings, weights are not initialized randomly, but to a very small value close to zero. We found that the smaller the weights, the better the network convergence with a less root mean squared error. As the nodes in the hidden layer correspond to categories in the knowledge graph, once the model has been trained, the sum of the weights of edges entering a node represents somehow its worthiness in the definition of a rating. If we consider the nodes associated (connected) to a specific item, their weight may be considered as a initial form of explanation for a given rating.

Please note, that such autoencoders do not need bias nodes because these latter are not representative of any semantic data in the graph. Hence, the structure of a generic hidden units looks like the one depicted in Figure 2.
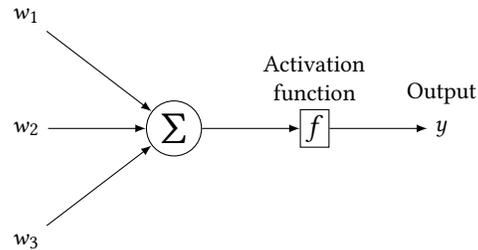

Figure 2: Structure of units

As activation function, in our implementation we used the well known sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$.

As a result, once the network converges we have a latent representation of features associated to a user profile together with their weight. However, very interestingly, this time the features also have an explicit meaning as they are in a one to one mapping with elements (nodes) in a knowledge graph. Our autoencoder is therefore capable to learn for each user the semantics behind her ratings and weigh them.

The rationale behind our idea is to catch the side information shared among items rated by a user, so that it gains higher value, tending
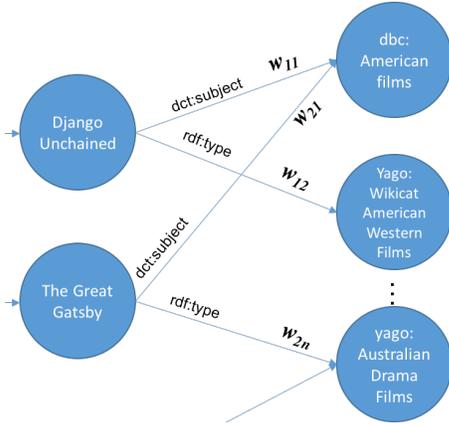
Figure 3: An excerpt of the network in Figure 1 after the training.

towards to 1, for positively rated items. On the other side, information shared among negatively rated item will tend towards 0. Therefore, our autoencoder finds the best value for those features that approximate the target user ratings by taking into account what are the best features the user is interested in. Features in a user profile are normalized within the interval $[0, 1]$ using a standard min-max normalization.

Given the trained autoencoder, a user profile is then built by considering the features associated to items she rated in the past.

It is worth noticing that in order to train our autoencoder, according to the semantic structure of the data contained in the hidden layer, at least 2 rates are required for each user, otherwise a constant value for all the features would be spread to approximate the targeted user ratings.

4 COMPUTING RECOMMENDATIONS

As we said before the weight associated to a feature f_n is the summation of the weights w_{jn} computed in the semantic autoencoder for each edge entering the node representing the feature itself. As we train an auto encoder for each user, we have weights changing depending on the original user profile $P(u) = \{\langle i, r \rangle\}$ with i being an item rated by the user and r its associated rating. More formally, we have

$$w(f_n, u) = \sum_{j=0}^{j=inndeg(f_n)} w_{jn}$$

where $inndeg(f_n)$ is the number of edges entering the node representing the feature f_n. As an example, if we consider the excerpt of the network in Figure 1 represented in Figure 3, for `American_films` we have

$$w(\texttt{American\_Films}, u) = w_{11} + w_{21}$$

By means of the weights associated to each feature, we can now move a user profile $P(u)$ from the user×item space to the user×feature one. Given $F(i)$ as the set of features composing an item defined as

$$F(i) = \{f_n \mid \text{there is an edge in the autoencoder between } i \text{ and } f_n\}$$

we represent the user profile as

$$\hat{P}(u) = \{\langle f_n, w(f_n, u) \rangle \mid f_n \in F(i) \text{ with } i \text{ being rated by } u\} \quad (1)$$

Due to the high sparseness of the feature-item matrix, the pure content-based information available in the user profile could not be enough to provide valuable recommendations. Hence, we exploited collaborative information available in the original dataset to further enhance user profiles.

We projected them in a Vector Space Model where each feature is a dimension of the vector space and computed the cosine similarity between users and, for each user we computed the set $K(u)$ containing the k users most similar to u. Then, for each feature $f'_n \notin \hat{P}(u)$ and for each user $u' \in K(u)$ we estimated $w(f'_n, u)$ as

$$w(f'_n, u) = \frac{\sum\limits_{u' \in K(u)} w(f'_n, u')}{k} \quad (2)$$

and eventually we added $\langle f'_n, w(f'_n, u) \rangle$ to $\hat{P}(u)$.

After this post-processing step, ratings for unknown items $\tilde{i}$ to u can be computed by combining the weights in the user profiles associated to $F(\tilde{i})$. In our implementation we just sum their values.

$$r(\tilde{i}, u) = \sum_{f_n \in F(\tilde{i})} w(f_n, u) \quad (3)$$

5 EXPERIMENTS

In this section, we present the experimental evaluations. We describe the structure of the dataset used in the experiments and the evaluation protocol. In this experimental setup we focused on cold-users with a number of rating equal to 2, 5 or 10.

5.1 Dataset

We conducted the experiment on the Movielens 1M dataset, which is composed by 6040 users and 3952 items. Each user has at least 20 ratings and ratings are made on a 5-star scale.

In our experiments, we referred to the freely available knowledge graph of DBpedia[2]. In order to map items in Movielens to resources in DBpedia we adopted the mapping freely available at https://github.com/sisinflab/LODrecsys-datasets. For each item we extracted categorical information by considering the two RDF predicates:

```
http://purl.org/dc/terms/subject
http://www.w3.org/1999/02/22-rdf-syntax-ns#type
```

The former links to categories as the one available in Wikipedia while the latter is used to classify in a more engineered ontology all the resources available in DBpedia.

5.2 Evaluation protocol

Here, we show how we evaluate performances of our methods in recommending items to cold-start users. We split dataset using Hold-Out 80/20, ensuring that every user have 80% of their ratings in the training set and the remaining 20% in the test set. We look for users in the test set that have at least 10 rates, and we selected them as potential cold user candidates. The protocol presented in the following is inspired by [31]. We made the candidate users

cold by removing their ratings from the training set. We tested our approach with profiles reduced to 2, 5 and 10 ratings.

(1) Setup the cold-start user scenario
  - Randomly choose 25% of users from cold user candidates and put them into set $\mathbb{U}_c$
  - $\forall u \in \mathbb{U}_c$ move out their ratings from the training set to $\mathbb{F}_c$
(2) Evaluate the cold-start user scenario
  - Create an empty set $\mathbb{R}_c$
  - For $n \in \{2, 5, 10\}$ do
    - $\forall u \in \mathbb{U}_c$ do:
      * randomly pick up n of his ratings from $\mathbb{F}_c$ and move them to the training set
    - Train the model
    - $\forall u \in \mathbb{U}_c$ generate recommendation for all unrated items
    - Evaluate recommendations for cold-users only

5.3 Metrics

In this work we avoided to use Root Mean Squared Error (RMSE). It is known that it may estimate the same error for top-N items and bottom-N items, without taking into account that an error in top-N items should be more relevant compared to an error for lower ranked items. For this reason, we chose to use Precision and Recall and nDCG to evaluate the accuracy of our model in cold user scenarios.

Precision is defined as the proportion of retrieved items that are relevant to the user.

$$Precision@N = \frac{|L_u(N) \cap TS_u^+|}{N}$$

where $L_u(N)$ is the recommendation list up to the N-th element and TS_u^+ is the set of relevant test items for u. Precision measures the system's ability to reject any non-relevant documents in the retrieved set.

Recall is defined as the proportion of relevant items that are retrieved.

$$Recall@N = \frac{|L_u(N) \cap TS_u^+|}{TS_u^+}$$

Recall measures the system's ability to find all the relevant documents.

Precision and recall can be combined with each other in the F1 measure computed as the harmonic mean between precision and recall.

$$F1@N = 2 \cdot \frac{Precision@N \cdot Recall@N}{Precision@N + Recall@N}$$

In information retrieval, Discounted cumulative gain (DCG) is a metric of ranking quality that measures the usefulness of a document based on its position in the result list. Recommended results may vary in length depending on the user, therefore is not possibile to compare performance among different users, so the cumulative gain at each position should be normalized across users. Hence, normalized discounted cumulative gain, or nDCG, is computed as:

$$nDCG_u@N = \frac{1}{IDCG@N} \sum_{p=1}^{N} \frac{2^{r_{up}} - 1}{\log_2(1 + p)}$$

where p is the position of an item in the recommendation list and $IDCG@N$ indicates the score obtained by an ideal ranking of $L_u(N)$. Accuracy metrics are a valuable way to evaluate the performance of a recommender system. Nonetheless, it has been argued [24] that also diversity should be taken into account when evaluating users' satisfaction. In order to evaluate the diversification power of our approach we also measure ERR-IA[2].

$$ERR - IA = \sum_{r=1}^{n} \frac{1}{r} \sum_{t} P(t|q) \prod_{i=1}^{r-1} (1 - R_i^t) R_r^t$$

where r is the position of an item i, t is the topic (in our case topics are movie genres as stated in Movielens Datatset ancillary files), $P(t|q)$ is the conditional probability of the topic given the query (user profiles in this case), R_i is the probability of the relevance of the item and R_r is the probability of the relevance of the list of items from 1 to r. With this metric, the contribution of each items in the recommendation list is based on the relevance of documents ranked above it. The discount function then depends also on the relevance of previously ranked documents.

5.4 Results Discussion

In experiments we conducted, we compared with BPRSLIM as baseline. BPRSLIM is a CF state-of-the-art sparse linear method that leverages the objective function as Bayesian personalized ranking. In Table 1 we report only those configurations for which our semantic-autoencoder gets the best results compared to BPRSLIM. We can see that for a number of ratings equal to 2 and 5, we outperform BPRSLIM in terms of precision and nDCG. From Table 1 we see that our approach gets much better results also in terms of recall and ERR-IA for very cold users, i.e., with only 2 ratings in the profile. As the number of ratings grows, the collaborative component becomes more relevant and BPRSLIM beats our SEM-AUTO approach. This result is more evident if we compare the plots in Figure 4 and 5 reporting the value of F1 for users with 2 and 5 ratings in their profile respectively. In the former, with only 2 ratings, SEM-AUTO shows a much better behavior than BPRSLIM while in the latter case BPRSLIM is never beaten by SEM-AUTO. It is interesting to note that, depending on the number of ratings in the user profile,the performance in therm of accuracy decreases as the number of neighbors increases. Another interesting result here is that increasing the number of ratings we need a higher number of neighbors in $K(u)$ to compute $w(f_n', u)$ in Equation (2) to reach results comparable with BPRSLIM. As for diversity, in very cold user situations, SEM-AUTO shows to diversify recommendation results better than BPRSLIM.

6 CONCLUSION AND FUTURE WORK

In this paper, we have presented a novel method to design semantics-aware autoencoders (SEM-AUTO) driven by information encoded in knowledge graphs. As for classical applications of autoencoders to feature selection, we compute a latent representation of items but, in our case, we can attach an explicit semantics to selected

	#ratings	k	f1@10	precision@10	recall@10	nDCG@10	ERR-IA@10
BPRSLIM	2	−	0.021741632	0.032649007	0.016297099	0.023353576	0.018825394
SEM-AUTO		10	**0.023096283**	**0.033046358**	**0.017751427**	**0.028283378**	**0.02600731**
BPRSLIM	5	−	0.039078954	0.050066225	0.032046252	0.045158629	0.042121369
SEM-AUTO		100	0.038598535	**0.054039735**	0.030020531	**0.048623943**	**0.047124717**

Table 1: Experimental Results. #ratings represents the number of ratings in cold users. k is the number of similar users belonging to $K(u)$ used in Equation (2)
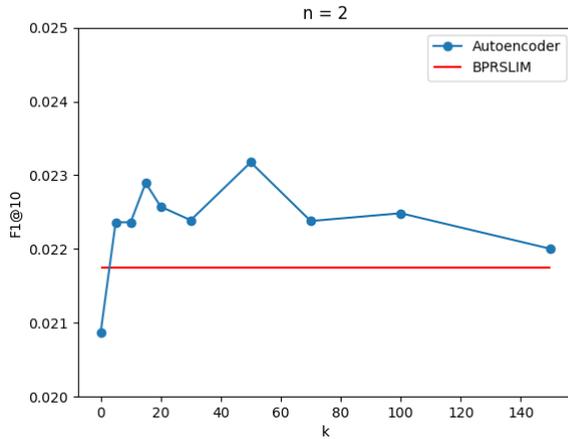

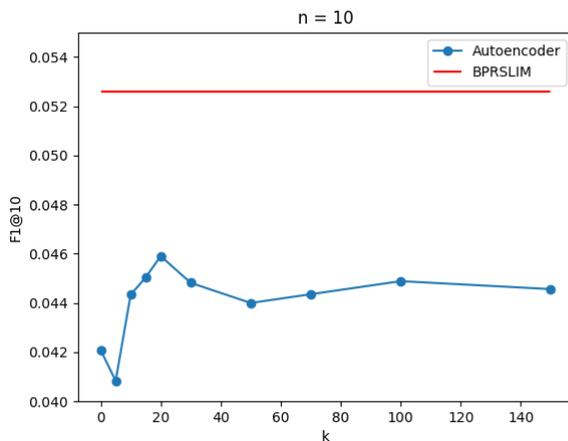**Figure 4: Plot for 2 ratings.**

Figure 5: Plot for 10 ratings.

features. This allows our system to exploit both the power of deep learning techniques and, at the same time to have a meaningful and understandable representation of the trained model. We used our approach to autoencode user ratings in a recommendation scenario via the DBpedia knowledge graph and proposed a simple algorithm to compute recommendations based on the semantic features we extract with our autoencoder. Experimental results show that even

with a simple approach that just sums the weights associated to features we are able to beat state of the art recommendation algorithms for cold user scenarios. The preliminary results presented in this paper pave the way to various further investigations. We are currently testing SEM-AUTO on other knowledge graphs such as Wikidata to see how much our results depend on the underlying semantic data. Moreover, we want to exploit the features extracted from SEM-AUTO as side information in hybrid state of the art approaches to test their representational effectiveness. Finally, having an explicit representation of latent features opens the door to a better user modeling by means of preference based languages such as CP-theories [4] that can be further exploited to provide meaningful explanations to the user for recommendation results.

REFERENCES

[1] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. DBpedia: A Nucleus for a Web of Open Data. In *Proceedings of the 6th International The Semantic Web and 2Nd Asian Conference on Asian Semantic Web Conference (ISWC'07/ASWC'07)*. Springer-Verlag, 722–735.
[2] Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM conference on Information and knowledge management*. ACM, 621–630.
[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems*. ACM, 191–198.
[4] Tommaso Di Noia, Thomas Lukasiewicz, Maria Vanina Martínez, Gerardo I. Simari, and Oana Tifrea-Marciuska. 2015. Combining Existential Rules with the Power of CP-Theories. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*. 2918–2925.
[5] Xin Dong, Lei Yu, Zhonghuo Wu, Yuxia Sun, Lingfeng Yuan, and Fangxi Zhang. 2017. A Hybrid Collaborative Filtering Model with Deep Structure for Recommender Systems. (2017). https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14676
[6] Ali Mamdouh Elkahky, Yang Song, and Xiaodong He. 2015. A Multi-View Deep Learning Approach for Cross Domain User Modeling in Recommendation Systems. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 278–288. https://doi.org/10.1145/2736277.2741667
[7] Ignacio Fernández-Tobías, Iván Cantador, Marius Kaminskas, and Francesco Ricci. 2011. A generic semantic-based framework for cross-domain recommendation. In *Proceedings of the 2nd International Workshop on Information Heterogeneity and Fusion in Recommender Systems (HetRec '11)*.
[8] Ignacio Fernández-Tobías, Paolo Tomeo, Iván Cantador, Tommaso Di Noia, and Eugenio Di Sciascio. 2016. Accuracy and Diversity in Cross-domain Recommendations for Cold-start Users with Positive-only Feedback. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)*. ACM.
[9] Benjamin Heitmann and Conor Hayes. 2010. C.: Using linked data to build open, collaborative recommender systems. In *In: AAAI Spring Symposium: Linked Data Meets Artificial Intelligenceâ ÂŽ. (2010*.
[10] Geoffrey E. Hinton and Ruslan Salakhutdinov. 2006. Reducing the Dimensionality of Data with Neural Networks. *Science* 313 (July 2006), 504–507. https://doi.org/10.1126/science.1127647
[11] Geoffrey E Hinton and Ruslan R Salakhutdinov. 2006. Reducing the dimensionality of data with neural networks. *science* 313, 5786 (2006), 504–507.
[12] Houda Khrouf and Raphaël Troncy. 2013. Hybrid Event Recommendation Using Linked Data and User Diversity. In *Proceedings of the 7th ACM Conference on*

Recommender Systems (RecSys '13). ACM, New York, NY, USA, 185–192. https://doi.org/10.1145/2507157.2507171

[13] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[14] Todd A. Letsche and Michael W. Berry. 1997. Large-scale Information Retrieval with Latent Semantic Indexing. *Inf. Sci.* 100, 1-4 (1997), 105–137. https://doi.org/10.1016/S0020-0255(97)00044-3

[15] Rouzbeh Meymandpour and Joseph G Davis. 2015. Enhancing Recommender Systems Using Linked Open Data-Based Semantic Analysis of Items. In *Proceedings of the 3rd Australasian Web Conference (AWC 2015)*, Vol. 27. 11–17.

[16] Cataldo Musto, Pasquale Lops, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. 2016. Semantics-aware Graph-based Recommender Systems Exploiting Linked Open Data. http://doi.acm.org/10.1145/2930238.2930249. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization, UMAP 2016, Halifax, NS, Canada, July 13 - 17, 2016*, Julita Vassileva, James Blustein, Lora Aroyo, and Sidney K. D. Mello (Eds.). ACM, 229–237. https://doi.org/10.1145/2930238.2930249

[17] Cataldo Musto, Giovanni Semeraro, Pasquale Lops, and Marco de Gemmis. 2014. Combining Distributional Semantics and Entity Linking for Context-Aware Content-Based Recommendation. In *User Modeling, Adaptation, and Personalization - 22nd International Conference, UMAP 2014*.

[18] Xia Ning and George Karypis. 2012. Sparse linear methods with side information for top-n recommendations. In *Proceedings of the sixth ACM conference on Recommender systems (RecSys '12)*. ACM, New York, NY, USA, 155–162.

[19] Tommaso Di Noia, Vito Claudio Ostuni, Paolo Tomeo, and Eugenio Di Sciascio. 2016. SPrank: Semantic Path-Based Ranking for Top-N Recommendations Using Linked Open Data. *ACM Trans. Intell. Syst. Technol.* 8, 1 (Sept. 2016).

[20] Vito Claudio Ostuni, Tommaso Di Noia, Eugenio Di Sciascio, and Roberto Mirizzi. 2013. Top-N Recommendations from Implicit Feedback Leveraging Linked Open Data. In *Proceedings of the 7th ACM Conference on Recommender Systems (RecSys '13)*. ACM, New York, NY, USA, 85–92. https://doi.org/10.1145/2507157.2507172

[21] Alexandre Passant. 2010. *dbrec — Music Recommendations Using DBpedia*.

[22] Azzurra Ragone, Paolo Tomeo, Corrado Magarelli, Tommaso Di Noia, Matteo Palmonari, Andrea Maurino, and Eugenio Di Sciascio. 2017. Schema-summarization in Linked-data-based Feature Selection for Recommender Systems. In *Proceedings of the Symposium on Applied Computing (SAC '17)*. ACM, 330–335. https://doi.org/10.1145/3019612.3019837

[23] Suvash Sedhain, Aditya Krishna Menon, Scott Sanner, and Lexing Xie. 2015. AutoRec: Autoencoders Meet Collaborative Filtering. In *Proceedings of the 24th International Conference on World Wide Web (WWW '15 Companion)*. ACM, New York, NY, USA, 111–112. https://doi.org/10.1145/2740908.2742726

[24] Barry Smyth and Paul McClave. 2001. Similarity vs. Diversity. In *Proceedings of the 4th International Conference on Case-Based Reasoning: Case-Based Reasoning Research and Development (ICCBR '01)*. Springer-Verlag, London, UK, UK, 347–361. http://dl.acm.org/citation.cfm?id=646268.758890

[25] Florian Strub, Romaric Gaudel, and Jérémie Mary. 2016. Hybrid Recommender System Based on Autoencoders. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. ACM, New York, NY, USA, 11–16. https://doi.org/10.1145/2988450.2988456

[26] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*. ACM, 1096–1103.

[27] Denny Vrandečić and Markus Krötzsch. 2014. Wikidata: A Free Collaborative Knowledgebase. *Commun. ACM* 57, 10 (2014), 78–85. https://doi.org/10.1145/2629489

[28] Jeroen B. P. Vuurens, Martha Larson, and Arjen P. de Vries. 2016. Exploring Deep Space: Learning Personalized Ranking in a Semantic Space. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems (DLRS 2016)*. ACM, New York, NY, USA, 23–28. https://doi.org/10.1145/2988450.2988457

[29] Hao Wang, Naiyan Wang, and Dit-Yan Yeung. 2015. Collaborative Deep Learning for Recommender Systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '15)*. ACM, New York, NY, USA, 1235–1244. https://doi.org/10.1145/2783258.2783273

[30] Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. 2016. Collaborative Denoising Auto-Encoders for Top-N Recommender Systems. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining (WSDM '16)*. ACM, New York, NY, USA, 153–162. https://doi.org/10.1145/2835776.2835837

[31] Jingwei Xu, Yuan Yao, Hanghang Tong, Xianping Tao, and Jian Lu. 2015. *Ice-Breaking: Mitigating cold-start recommendation problem by rating comparison*. 3981–3987.