A Deep Architecture for Content-based Recommendations Exploiting Recurrent Neural Networks

Alessandro Suglia
Department of Computer Science
University of Bari Aldo Moro
alessandro.suglia@gmail.com

Claudio Greco
Department of Computer Science
University of Bari Aldo Moro
claudiogaetanogreco@gmail.com

Cataldo Musto
Department of Computer Science
University of Bari Aldo Moro
cataldo.musto@uniba.it

Marco de Gemmis
Department of Computer Science
University of Bari Aldo Moro
marco.degemmis@uniba.it

Pasquale Lops
Department of Computer Science
University of Bari Aldo Moro
pasquale.lops@uniba.it

Giovanni Semeraro
Department of Computer Science
University of Bari Aldo Moro
giovanni.semeraro@uniba.it

ABSTRACT

In this paper we investigate the effectiveness of Recurrent Neural Networks (RNNs) in a top-N content-based recommendation scenario. Specifically, we propose a *deep* architecture which adopts Long Short Term Memory (LSTM) networks to jointly learn two embeddings representing the items to be recommended as well as the preferences of the user. Next, given such a representation, a logistic regression layer calculates the relevance score of each item for a specific user and we returns the top-N items as recommendations.

In the experimental session we evaluated the effectiveness of our approach against several baselines: first, we compared it to other *shallow* models based on neural networks (as Word2Vec and Doc2Vec), next we evaluated it against state-of-the-art algorithms for collaborative filtering. In both cases, our methodology obtains a significant improvement over all the baselines, thus giving evidence of the effectiveness of deep learning techniques in content-based recommendation scenarios and paving the way for several future research directions.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → **Neural networks**;

KEYWORDS

Recommender Systems, Deep Learning, Recurrent Neural Networks, Content Representation

ACM Reference format:
Alessandro Suglia, Claudio Greco, Cataldo Musto, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. 2017. A Deep Architecture for Content-based Recommendations Exploiting Recurrent Neural Networks. In *Proceedings of UMAP '17, Bratislava, Slovakia, July 09-12, 2017,* 10 pages.
DOI: http://dx.doi.org/10.1145/3079628.3079684

1 INTRODUCTION

According to *Barry Schwartz*, people deal with the so-called *paradox of choice* [49]. This means that we can't psychologically and physiologically make an informed choice in scenarios where we face several alternatives and we have to discern among all the available options. Nowadays, in the *Big Data era*, this issue is much more felt. Given the plethora of different information sources today available, it is difficult to effectively find what we are looking for, thus an intelligent system which knows exactly what we like and what we want may be very useful to speed-up daily jobs and improve human activities. Systems able to deal with the *information overload problem* are called "recommender systems" [42]. They have the effect of guiding the user in a personalized way to interesting or useful objects in a large space of possible options. As reported in [8], recommender system techniques can be divided in different classes by exploiting several representations of users and items to generate recommendations.

Among the available paradigms, the recent exponential growth of textual data, made available through open knowledge sources as Wikipedia or Freebase[1], gave new lymph to the research in the area of Content-based Recommender Systems (CBRS) [29]. Indeed, such systems need *textual content* since they rely on the descriptions of the items to provide users with recommendations. This process is typically carried out by matching the features describing the items a user liked, which are stored in her *profile*, with those describing the items currently available the user did not rate yet.

It immediately follows that the effectiveness of content-based recommendation strategies tremendously depend on the way the items as well as the preferences of the target user are modeled [13], since a poor representation certainly loses a lot of relevant information and leads the algorithm to generate poor recommendations. It is not by chance that most of the research effort in the area of CBRS has been carried out to introduce *semantics-aware representations* [14], as those based on the use of ontologies [9, 30] or distributional semantics [35, 37], and to evaluate richer sets of features, as those available in Linked Open Data cloud [15, 32, 34].

At the same time, deep learning [28] architectures gained more and more attention, since they obtained outstanding performance in a broad range of machine learning-related scenarios, ranging from classical tasks as machine translation [2] and speech recognition

[1]http://www.freebase.com

[21] to very *visionary* ones, as learning a textual description of a given image[2] or beating human players in challenging games like *Go* [50].

Several work tried to exploit such architectures also to model *textual content*: the most popular attempt towards this direction is certainly Word2Vec (W2V) [31], which adopts a *shallow* deep model consisting of a two-layers neural network to learn dense vector-space representations of words, called *word embeddings*. Moreover, other models as Recurrent Neural Networks (RNNs) [43] and Long-Short Term Memory Networks (LSTMs) [24] have been recently investigated as well, and results [21, 53] showed that deep learning architectures tend to overcome the performance of state of the art algorithms.

As a consequence, in this article we follow this research direction and we investigate the effectiveness of techniques based on *deep learning* to represent textual content in a content-based recommendation scenario. To this end, we introduce a novel architecture, *Ask Me Any Rating (AMAR)*, which jointly learns a user and an item embedding. The former is used to model the user preferences and the latter exploits LSTMs to learn the item embedding from the item description. The resulting representations are used to feed a logistic regression layer to calculate the relevance of a certain item for the target user.

Even if the use of LSTMs in machine learning tasks can't be considered as *new*, up to our knowledge this is the first attempt evaluating the use of such architectures for *content-based recommendations*. This work significantly extends the preliminary results presented in [33] by providing more details on the components of the architecture and by carrying out a more solid and extensive experimental evaluation, which now includes different datasets, more combination of parameters, and more challenging baselines.

To sum up, in this paper we provide the following contributions:

- We propose a novel architecture based on deep neural networks to provide users with content-based recommendations;
- We evaluate the impact of LSTMs to represent textual data in a content-based recommendation scenario;
- We validate our methodology by comparing it to several baselines against two state of the art datasets;

The rest of the article is organized as follows. In Section 2 we give an overview of the literature in the area of Deep Learning and CBRS. Next, the methodology we propose is depicted in Section 3: first, we provide some basics of RNNs, then we describe the modules constituting our deep architecture. Finally, the findings emerging from the experimental evaluation and the conclusions of this work are given in Section 4 and 5, respectively.

2 RELATED WORK

The adoption of deep learning techniques in the area of Recommender Systems is a recent research trend. The first article investigating the effectiveness of such models is due to Salakhutdinov et al. [45], who proposed the use of Restricted Boltzmann Machines to model user-item interactions and to provide collaborative recommendations.

[2]http://googleresearch.blogspot.it/2014/11/a-picture-is-worth-thousand-coherent.html

However, despite its novelty, many deep learning architectures have been already investigated in literature: as an example, Wang et al. [57] used Stacked Denoising Autoencoders [56] to jointly learn collaborative and content-based features which are then incorporated in a standard collaborative filtering model. The same technique has been also adopted by Strub et al. [52]. Differently, the use of Convolutional Neural Network (CNNs) is proposed in [55], where the authors introduce a latent factor model for music recommendation whose features are learned by using CNNs. In this case, experiments showed a significant improvement with respect to the classical representation based on *bag of words*. A neural network model which combines wide and deep neural networks is presented in [10]. In particular they try to jointly learn a wide linear model which leverage co-occurences of hand-crafted features by cross-products transformation and a deep neural network in order to achieve both *memorization* and *generalization*, which are really useful properties for effective recommender systems. Differently from our approach it considers only contextual features associated to the user and does not learn a specific user embedding which can be able to capture latent relationships between user preferences. Another relevant attempt in the area of music recommendation based on deep learning is due to Wang et al. [58], who used deep belief networks to learn features from audio content and provide users with personalized recommendations. Finally, Zhang et al. [61] propose a deep learning architecture to perform collaborative filtering exploiting distributed representations.

All these contributions showed that models based on deep learning often overcome state-of-the-art algorithms. However, it did not emerge an architecture which overall performed the best, since each deep learning architecture has its own peculiarities and advantages. Clearly, a detailed overview of such models is out of the scope of this article: we suggest to refer to [5] for an extensive analysis.

The first distinguishing aspect of this article lies in the fact that none of the already proposed approaches exploit deep learning techniques to provide *pure* content-based recommendations. The most similar attempts towards this direction are due to Ozsoy et al. [39] and to Musto et al. [36], who recently proposed the use of Word2Vec [31] to learn word embeddings representing items and user profiles. Their experiments showed that CBRS based on word embeddings can obtain results comparable to those obtained by other content-based approaches and by algorithms for collaborative filtering based on matrix factorization. However, this technique can't be labeled as a as *real* deep learning approach since it relies on a two-layers neural network, while typically deep architectures should include four levels, at least.

In this article we investigate the effectiveness of Recurrent Neural Networks (RNNs) to model textual content for CBRS. The use of such models has been recently evaluated by Hidasi et al. [23], who proposed a novel approach for *session-based recommendations* based on RNNs. In this approach the authors adapt classic RNNs by introducing a novel ranking loss function and obtained a significant improvement over state of the art algorithms. As proved by the authors, such architectures can be useful to model *sequences of data* of arbitrary length, as the interactions occurring in a session. Similarly, we exploited this insight to model the content feeding a CBRS, as the *plot* of a movie, as a *sequence of terms*. This choice is due to the fact that the effectiveness of RNNs in natural-language related tasks has been already proved by several contributions [21, 51, 59], where they got results at least comparable to those obtained by other deep learning models as CNNs [62]. The effectiveness of RNNs for recommendation tasks is also proved by Florez et al. [17]. In this work ratings are predicted by exploiting a latent representation learnt from item descriptions by a RNN. Finally, another approach applied to item modeling is described in [18], in which CNNs have been applied to web pages and used to recommend interesting pages related to the one read by the user.

The use of LSTMs to model textual content in CBRS is the second distinguishing aspect of this work: the only similar contribution is due to Almahairi et al. [1], who used LSTMs to model textual content (textual *reviews*, in that case) to feed a collaborative recommendation algorithm. Another interesting work which tries to learn item representation directly from raw text description is presented in [3], which applies a bidirectional *Gated Recurrent Unit (GRU)* network to encode item description and an embedding for each tag associated to the item. This approach is really similar to the one we propose, but differently from our work here the authors decide to use a custom cross entropy function which is a linear combination of two different loss functions which try to optimize both the item representation and the tag representation for a given item.

3 METHODOLOGY

In this section we provide a comprehensive description of our approach based on deep learning: first, we introduce some basic concepts about RNNs and LSTMs, next we depict the architecture of our model.

3.1 Basics of RNNs and LSTMs

In the past years, deep learning has offered to scientists computational models organized in *multiple processing layers* able to learn from data using a general learning algorithm called *back-propagation* [28, 48]. Typically, these approaches combine different modules, each of which transforms the input representation to a representation which becomes progressively more abstract. By combining an adequate number of transformation layers and by using back-propagation to indicate how the network's parameters should change to improve their performance on the current task, these architectures are able to learn very complex functions and to discover intricate structures directly from the training data. As previously shown, these methods have dramatically improved the state of the art in several fields in a plenty of different machine learning tasks transforming the way by which common problems are tackled.

In this article, we focused on RNNs [43]. They are considered the main model able to manage *sequential data of arbitrary length.* It is a class of artificial neural networks in which the connections between the units allow the presence of *cycles.* Figure 1 describes the computational graph of a generic RNN, in its *compact* version and in its *unfolded* version in which the loop is represented as a series of repeated trasformations which involves the same parameters. Many recurrent neural networks in fact can be described with the

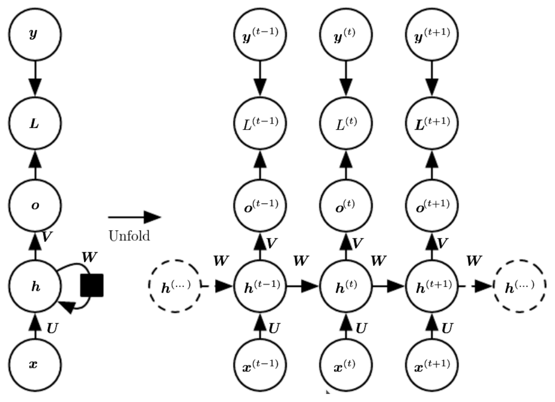


Figure 1: An example of a generic RNN: on the left the compact form of the computational graph is depicted, on the right there is its unfolded version computed in order to apply the *Backpropagation through time* algorithm (image adapted from [20]).

following equation:

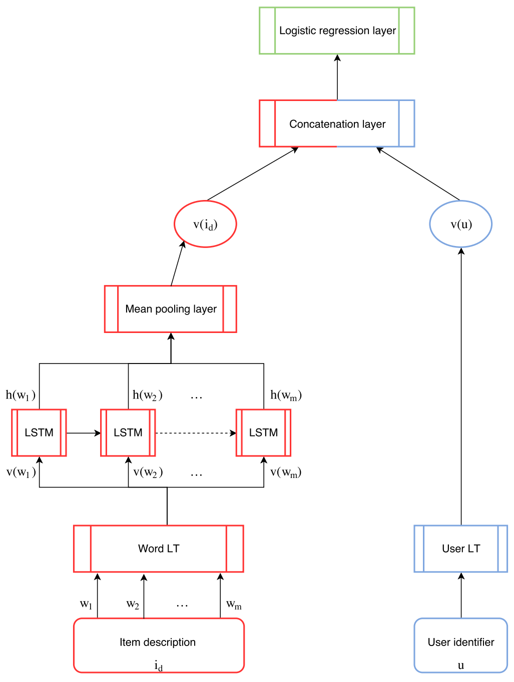
$$h^{(t)} = f(h^{(t-1)}, x^{(t)}, \theta)$$

where $h^{(t)}$ represents the state of the RNN at the timestep t which is the results of the application of a generic transformation f applied considering the previous state of the RNN $h^{(t-1)}$, the current input $x^{(t)}$ and the network parameters $\boldsymbol{\theta}$ which are shared through each timestep $t = 1, 2, \dots, T$. As a consequence, this class of networks can carry out the training process by taking into consideration the information coming from earlier states, as well. However, due to gradient computation applied to the same shared parameters $\boldsymbol{\theta}$, the RNNs suffer of the well-known problem of the *vanishing/exploding gradient* [6]. Indeed, especially when long-term dependencies have to be learned, the back-propagation algorithm can make the parameters too small (or too huge), thus making impossible to learn complex functions. As a consequence, LSTMs [24] emerged to solve such a problem. They are a specialization of RNNs which use cells with a more complex structure that allows the neural network to make more elaborate computations and interpretations. Several variants of the original architecture originally introduced in [24] have been proposed in literature [22]. However, in this work we exploited the most widely used one, presented in [19].

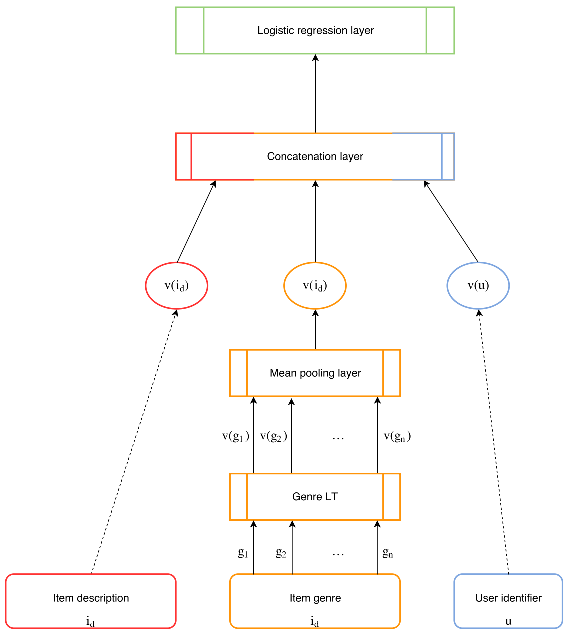
As previously shown, beyond solving the vanishing/exploding gradient problem, this structure well-performed in several tasks where short and long-term dependencies have to be learnt. In the following, we will show how LSTMs have been used to learn an embedding for the textual content describing the items recommended by a CBRS.

3.2 Architecture

The architecture we propose is partially inspired by the model based on LSTMs recently proposed in a Question Answering (QA) scenario [60]. Our choice is based on a simple insight: as in QA, given a question, an answer is provided on the ground of the available facts, CBRS can provide suggestions on the ground of the *description* of the available items given a *profile* of the user. Accordingly, we



(a) *AMAR* base architecture



(b) *AMAR* Extended architecture

Figure 2: Two instances of the AMAR recommender system framework: both the figures depict the forward pass of the neural network which represents how the input data are processed through each layer obtaining a representation which is used to estimate the probability of a like for the given item

decided to exploit the *analogy* between question and user profile (and, clearly, between answers and recommendations) to design the following architecture, named *Ask Me Any Rating (AMAR)*.

AMAR Base. The proposed architecture implements a content-based recommender system able to predict a score $s(u, i)$ which defines the probability that a user u would *like* a specific item i. In a nutshell, our approach is based on two different modules which jointly learn continuous vector representations for each user $u \in U$ and for each item $i \in I$ that are used to feed a classifier which generates the preference estimation.

Generally speaking, our architecture (see Figure 2a) consists of the following layers:

- **Lookup table layer:** extracts from a specific weight matrix the vector representation associated to a given user (or item, respectively);
- **LSTM module:** an RNN network with LSTM units;
- **Mean pooling layer:** calculates the mean of the input vectors;
- **Concatenation layer:** concatenates the input vectors;
- **Logistic regression layer:** exploits logistic regression to calculate the relevance of the item for a certain user.

An important component of our architecture is the *lookup table layer* which generates dense representations of its input. Given a set of elements A, each of them can be represented as a d-dimensional vector contained in a $W \in \mathbb{R}^{|A| \times d}$ embedding matrix. Let $e_j \in \mathbb{R}^{|A|}$, which is all zeros except for the j-th index in which there is the value 1. For each $a \in A$ we assign an index j, $1 \leq j \leq |A|$. We define the *embedding* of a generic element a as the vector representation $v(e_j)$ given by $e_j^\top W$. Intuitively, $v(e_j)$ can be intended as the j-row of the related embedding matrix.

Formally, given a set of items I and a set of users U, our architecture learns a d_i-dimensional embedding for each $i \in I$ and a d_u-dimensional embedding for each $u \in U$. This process is carried out according to the workflow presented in Figure 2a.

Each user u is given in input to a *user lookup table (User LT)* which generates a d_u-dimensional *user embedding* $v(u)$ considering a weight matrix W_u. Each word $w_1, w_2, \ldots, w_m$ of the item description i_d associated to the item i is given in input to a *word lookup table (Word LT)* which generates a d_w-dimensional embedding $v(w_j)$ for each word w_j, $1 \leq j \leq m$ considering a weight matrix W_w. Clearly, at the first step of the learning process the weight matrixes W_u and W_w are initialized according to a predefined probability distribution.

Next, these word representations $v(w_k)$ are sequentially passed through an *LSTM network* with s hidden units which generates for each of them a s-dimensional latent representation $h(w_j)$ using an LSTM cell. The motivation behind the adoption of LSTM is twofold: first, as stated in the Introduction, several work already showed that LSTM can overcome shallow models (see [21, 51, 54]). Moreover, we chose to adopt LSTM since such networks are very effective when sequences of input have to be modeled. Given that the textual description of the items can be easily viewed as a sequence of terms, it was straightforward for us to investigate the adoption of LSTMs in a content-based recommendation scenario. Once the latent representation are computed by the LSTM network, the *item embedding* $v(i)$ is obtained by a *mean pooling layer* which averages the latent representations $h(w_k)$ for all the words in the textual description of the item.

The resulting representations $v(u)$ and $v(i)$ are then concatenated through a *concatenation layer*. The resulting (d_u+d_i)-dimensional feature vector is given as input to a *logistic regression layer* which finally predicts the score $s(u,i)$. Formally, these operations can be expressed by the following expression:

$$y = \text{sigmoid}(W_{ih}[v(i_u), v(i_d)] + b_{ih})$$

where $W_{ih} \in \mathbb{R}^{(d_u+d_{i_d})\times 1}$ is a weight matrix, $b_{ih} \in \mathbb{R}$ is a bias term and the square brackets denote the concatenation operation between two vectors. The *logistic regression layer* is able to learn in its parameters W_{ih} and b_{ih} relevant relationships between the user and the rated items useful for the preference estimation $s(u,i)$.

In order to perform the *top-N* recommendation task, the recommender system should generate a list of suggestions ordered by the item relevance with regard to the user profile. The presented architecture provides recommendations for a given user u by sorting items in decreasing order according to the score $s(u,i)$ for each item i.

AMAR Extended. As previously stated, a relevant research line in the area of CBRS is to investigate the effectiveness of features different from those extracted from the simple textual description of the items. Accordingly, we provided AMAR with a very *modular* and *extensible* architecture, in order to make the model able to learn a representation of the items $i \in I$ by also exploiting features different from the textual description.

This architecture, called *AMAR Extended*, extends the previous one by adding a module able to process supplementary features associated to each item. The *AMAR* Extended architecture proposed in this work associates a list of genres $g_1, g_2, \ldots, g_n$ to each item, as shown in figure 2b. Each genre is passed in input to a *genre lookup table (Genre LT)* which generates a d_g-dimensional embedding $v(g_k)$ for each genre g_k, $1 \leq k \leq n$ considering a weight matrix W_g. Next, a *mean pooling layer* averages the resulting representations $v(g_k)$ by giving a *genres embedding* $v(i_g)$ which is concatenated to the user and item embeddings to evaluate the recommendation score.

Clearly, in this preliminary attempt we chose to extend the architecture by only modeling information about the *genre* of the items, since we can assume that such an information is available for all the domains, but different set of features (as the *cast* of a movie or the *author* of a book) can be easily introduced without modifying the logic which inspired the architecture.

4 EXPERIMENTAL EVALUATION

In the experimental session we evaluated the effectiveness of the above described deep architecture against several baselines, in the task of *top-N recommendation* leveraging binary user feedback. First, we compared it to other models based on (shallow) neural networks *(Experiment 1)*, next we evaluated it against state-of-the-art techniques such as algorithms for matrix factorization, user-user and item-item collaborative filtering *(Experiment 2)*.

4.1 Experimental Design

Datasets: Experiments were carried out against two state-of-the-art datasets, as Movielens 1M (ML1M)[3] and DBbook. The first is a a widespread dataset for movie recommendations, while the latter focuses on book recommendations and is exploited in the *ESWC 2014 Linked-Open Data-enabled Recommender Systems challenge*[4]. Some statistics about the datasets are provided in Table 1.

A quick analysis of the table shows two datasets very different from each other. ML1M is larger than DBbook, but it is less sparse so it is more suitable for collaborative filtering-based algorithms. On the other side, DBbook is sparser and unbalanced towards negative preferences (only 45.85% of positive ratings), and this makes the recommendation task very challenging.

Protocol: Experiments were performed by adopting different protocols for the chosen datasets. As regards ML1M, we first applied a *binarization procedure* to convert the original dataset preferences expressed on a 5-point Likert scale, to 0 or 1. In particular, we considered as *positive* preferences only those ratings equal to 4 or 5, as *negative* all the others. After having applied the above-mentioned binarization procedure, we carried out a k-fold cross-validation by splitting the original dataset in five different folds. For each fold, we maintained the ratio between positive and negative ratings per user. On the other side, the single training/test split which was used in the ESWC 2014 Challenge is exploited for DBbook. Similarly, no processing was needed for the ratings since they were already available as *binarized*.

The effectiveness of our algorithm is evaluated by calculating the F1-Measure of the recommendation list [42]. In order to make our experiments reproducible, all the metrics are calculated by using the *RiVal toolkit* [44] following the *TestRatings* strategy [4]. The final F1@K measure for each algorithm is computed averaging the F1@K measure obtained on each fold. Statistical significance is assessed by exploiting *Mann-Whitney U* test (*P-value*=0.05), chosen after running the *Shapiro-Wilk* test[5], which revealed the non-normal distribution of the data. *P-values* are calculated by using the *RiVal toolkit*, as well.

Items Representation: To feed our content-based deep architecture, we collected *textual descriptions* of the items. Specifically, we used a ML1M mapping[16] to provide each movie with textual features. For each item, we used the mapping to obtain the *Wikipedia* page from which are extracted the first two sections or the abstract of the entity using the property *abstract* of the *DBpedia ontology*[6]. If a reference for a given item is not present in the used mapping[7], the IMDb synopsis was used otherwise the concatenation of the user reviews. On the other side, for DBbook we collect the *plots* of the books. In this case, data were made available during the ESWC 2014 Challenge. In both cases, textual descriptions were processed by removing stop words and performing tokenization.

[3]http://grouplens.org/datasets/movielens/
[4]http://challenges.2014.eswc-conferences.org/index.php/RecSys
[5]http://en.wikipedia.org/wiki/Shapiro-Wilk_test
[6]http://dbpedia.org/ontology/
[7]Some URIs are incorrect in the original mapping and for this reason not resolvable.

	ML1M	DBbook
Users	6,040	6,181
Items	3,301	6,733
Ratings	1,000,209	72,372
Sparsity	94.98%	99.83%
Positive Ratings	57.22%	45.85%
Avg. ratings/user ± σ	157.11±182.93	11.70±5.85
Median/mode per user	90/23	11/5
Avg. ratings/item ± σ	293.98±393.88	10.74±27.14
Median/mode per item	147/1	4/1

Table 1: Description of the datasets.

To feed AMAR Extended, we also collected information about the *genres* of both movies and books. In the first case, we provided each movie with the genres already available in ML1M data. As regards DBbook, we queried DBpedia and we used as genres the objects associated to the property literaryGenre or genre of the DBpedia ontology. If none of them were present, we used the objects associated to the property dcterms:subject.

Baselines: in order to confirm the effectiveness of our approach, we compared it with several state-of-the-art techniques, as *collaborative filtering*, *traditional content-based recommender system* and *embedding-based recommender systems*. A brief description of each algorithm follows:

Popularity a non-personalized technique which recommends most-popular items to users. Specifically, the algorithm suggest the items with the highest ratio of positive ratings over the total number of ratings.

U2U k-nearest neighbor user-based collaborative filtering [41], where the preference estimation is computed according to the preference expressed by users similar to the one to whom the suggestions will be generated.

I2I k-nearest neighbor item-based collaborative filtering [47], uses similarities between the rating patterns of items to estimate the preference of a given user for a given item.

BPRMF a matrix factorization model for item recommendation based on *Bayesian Personalized Ranking optimization criterion (BPR-Opt)* [40].

WRMF a weighted matrix factorization algorithm based on the *Alternating Least Squares (ALS)* learning method [25].

BPRSlim *Sparse Linear Methods (SLIM)* for item recommendation using *BPR-Opt* [38].

TF-IDF *Vector Space Model (VSM)* with *TF-IDF* weighting scheme [46] is used to represent each item description. The user representation is the centroid of the items rated positively by the user. The preference estimation score is computed as the cosine similarity between the item representation and the user representation.

Word2Vec (W2V) additional baselines are proposed using W2V representation learn from both the item descriptions using different W2V training strategies (*SG* and *CBOW*) and pre-trained word embeddings such as *Word2vec Google News*[8] *(W2V-PRE)*. The recommendation score is generated using the *cosine similarity* between the item represented by averaging word embeddings of its description and the user profile represented by averaging positively rated item representations, by following the recommendation pipeline proposed by Musto et al. in [36].

GloVe Exploited the pre-trained *GloVe Wikipedia 2014 + Gigaword 5*[9] word embeddings to generate recommendations according to the same procedure used for *W2V*.

Doc2Vec (D2V) , also known as *Paragraph Vector* [27], is another embedding-based algorithm. In particular, item representations are learnt from the item descriptions corpus and they are used to generate the user profile, averaging those that are positively rated by a given user. The recommendation score is computed as in the previous case, using the generated item description and user representation.

As regards the implementation details, *collaborative filtering* and *Popularity* baselines are available in MyMediaLite[10] and Scikit-learn[11], while approaches based on word embeddings are implemented by using the code available in the *Gensim* library for W2V[12] and D2V[13].

Overview of the Parameters: AMAR and AMAR Extended are trained by using RMSProp as optimizer. The parameter α is set to 0.9 and learning rate is set to 0.001. Models are trained for 25 epochs, by setting batch sizes to 1536 for ML1M and to 512 for DBbook, respectively. As cost function we choose the *binary cross entropy*[14], which is typically used for classification tasks since it tries to minimize the errors in the classification. Due to the computational costs requested by the models, the dimension of the learned embeddings d_w and d_u are fixed to 10. For the *LSTM module* we fixed s to 10.

As regards the CBRS based on W2V and D2V, we learn embeddings of dimension 300 and 500 by using the *Skip-Gram (SG)* and the *Continuous Bag-of-Words (CBOW)* methodologies for the former, and the *Distributed Memory Model (DM)* and the *Distributed Bag-of-Words (DBOW)* methodologies for the latter. Word embeddings are learnt by using the textual description of the items as corpus. Descriptions are gathered as previously described.

[8]https://code.google.com/archive/p/word2vec/

[9]http://nlp.stanford.edu/projects/glove/

[10]http://www.mymedialite.net/documentation/item_prediction.html

[11]http://scikit-learn.org/

[12]https://radimrehurek.com/gensim/models/word2vec.html

[13]https://radimrehurek.com/gensim/models/doc2vec.html

[14]https://en.wikipedia.org/wiki/Cross_entropy

	ML1M	
	F1@5	**F1@10**
W2V-SG-300	0.490	0.580
W2V-SG-500	0.485	0.580
W2V-CBOW-300	0.490	0.581
W2V-CBOW-500	0.487	0.579
W2V-PRE	*0.500*	*0.587*
GloVe	0.485	0.575
D2V-SG-300	0.479	0.571
D2V-SG-500	0.481	0.574
D2V-DBOW-300	0.482	0.574
D2V-DBOW-500	0.481	0.573
AMAR	0.555	0.641
AMAR+	**0.558**	**0.644**

Table 2: Results of Experiment 1 on ML1M data. The best-performing baseline is reported in italics while the overall best configuration is highlighted in bold.

	DBbook	
	F1@5	**F1@10**
W2V-SG-300	0.541	0.645
W2V-SG-500	0.542	0.645
W2V-CBOW-300	0.539	0.645
W2V-CBOW-500	0.540	0.645
W2V-PRE	0.548	*0.655*
GloVe	*0.552*	*0.655*
D2V-DM-300	0.540	0.652
D2V-DM-500	0.537	0.653
D2V-DBOW-300	0.54	0.654
D2V-DBOW-500	0.54	0.654
AMAR	0.564	0.662
AMAR+	**0.565**	**0.662**

Table 3: Results of Experiment 1 on DBbook data. The best-performing baseline is reported in italics while the overall best configuration is highlighted in bold

On the other side, I2I and U2U are evaluated by setting the neighborhood size to 30, 50 and 80, while the matrix factorization algorithms are run by learning 10, 30 and 50 latent factors.

Source code: The source code of our content-based recommendation algorithm based on deep learning has been published on GitHub[15]. The recommendation framework is implemented using *Torch7* [12] exploiting the *NVIDIA CUDA* libraries bindings to run the code on a Titan X GPU. The *LSTM network* architecture used in this work is based on the *SeqLSTM* module implemented in the *rnn package*[16].

4.2 Discussion of the results

Results of *Experiment 1* on ML1M and DBbook data are provided in Table 2 and 3, respectively. Each line reports the name of the configuration and its parameters, as the size of the embedding and the training methodology. The configuration based on pre-trained word vectors is reported as W2V-PRE, while AMAR Extended is reported as AMAR+ for the sake of brevity.

Results clearly show that AMAR and AMAR+ are able to significantly outperform all the baselines based on word embeddings on both the datasets. The improvement is particularly huge on ML1M (+6% over the best-performing baseline), but the gap is statistically significant on DBbook as well. As regards the baselines, neither the size of the vectors nor the training methodology significantly affects the results, since the gaps are never statistically significant. Moreover, it is worth to note that the best configuration is that based on pre-trained word embeddings. This means that the quality of learnt embeddings strictly depends on the amount (and on the quality) of the data that feed the learning algorithm. Indeed, in this case the embeddings pre-trained by Google and those pre-trained using *GloVe* beat those trained on the textual content gathered from the dataset, even if those embeddings are based on *exogenous knowledge* which is wider and not strictly related to a specific recommendation domain.

[15]https://github.com/nlp-deepcbrs/amar

[16]https://github.com/Element-Research/rnn#rnn.SeqLSTM

However, regardless this interesting finding, this first experiment showed the usefulness of exploiting *more complex* deep architectures for content-based recommendation tasks, since results showed that the introduction of LSTMs and the development of a deep architecture as AMAR can lead to more effective recommendations. This is an expected behavior, since neural networks are able to learn models from huge number of ratings and to extract relevant features from item descriptions.

Furthermore, as previously stated, both RNNs and LSTMs are particularly useful to model sequences of data as the *textual description* of the items, thus it is not surprising that a better representation of textual content can lead to a better accuracy of the recommendation algorithm. Indeed, the embedding-based baselines are not able to fill the gap due to their ineffectiveness to represent items and users by a simple mean of pre-trained word embeddings. In addition, in this particular setting we are interested in modeling the whole sequence of terms which describes the item rather than each single word given a window of contextual terms (as Word2Vec does).

Moreover, it is worth to note that these results are obtained with a very *light* optimization of the parameters of the models. It is likely that by introducing regularization techniques, tuning hyperparameters and using early stopping procedures better performances may be obtained. Indeed, given the difficulty of *RNNs* to converge to satisfying minima, more epochs are typically required to obtain better results.

Next, the insight of extending the original architecture by introducing extra features as the *genre* lead to promising results. Indeed, we obtained an improvement (even if *not significant*) on both datasets. This supports the idea that more (and diverse) *semi-structured* features describing the items to be recommended can further improve the accuracy of the suggestions. Clearly, further analysis are needed to assess which extra features are worth to be used and how they do affect the overall results, but these preliminary results pave the way to several developments of our architecture which will be investigated in the future.

	ML1M	
	F1@5	**F1@10**
Popularity	0.425	0.525
I2I-30	0.432	0.527
I2I-50	0.431	0.528
I2I-80	0.430	0.527
U2U-30	0.427	0.525
U2U-50	0.425	0.524
U2U-80	0.424	0.523
BPRMF-10	0.424	0.522
BPRMF-30	0.425	0.524
BPRMF-50	0.424	0.524
WRMF-10	0.419	0.520
WRMF-30	0.423	0.522
WRMF-50	0.425	0.525
BPRSlim	*0.446*	*0.548*
AMAR	0.555	0.641
AMAR+	**0.558**	**0.644**

Table 4: Results of Experiment 2 on ML1M data. The best-performing baseline is reported in italics while the overall best configuration is highlighted in bold.

	DBbook	
	F1@5	**F1@10**
Popularity	0.532	*0.645*
TF-IDF	0.532	*0.645*
I2I-30	*0.536*	0.64
I2I-50	0.534	0.64
I2I-80	0.531	0.64
U2U-30	0.536	0.639
U2U-50	0.528	0.636
U2U-80	0.522	0.634
BPRMF-10	0.507	0.631
BPRMF-30	0.508	0.631
BPRMF-50	0.507	0.631
WRMF-10	0.514	0.632
WRMF-30	0.518	0.635
WRMF-50	0.519	0.636
BPRSlim	0.511	0.632
AMAR	0.564	0.662
AMAR+	**0.565**	**0.662**

Table 5: Results of Experiment 2 on DBbook data. The best-performing baseline is reported in italics while the overall best configuration is highlighted in bold.

Next, in *Experiment 2* we compare our content-based approach to other baselines in the area of recommender systems. Results reported in Table 4 and 5 show that both AMAR and AMAR+ can significantly outperform all the state-of-the-art algorithms we consider. The improvement is even larger than that observed in Experiment 1, since the average gap between AMAR and the best-performing baseline is between 10 and 13% on ML1M and around 3% on DBbook.

These results further prove the effectiveness of our strategy, since also widespread and well-performing techniques based on matrix factorization are outperformed by AMAR and AMAR+. Another interesting finding of Experiment 2 is that our deep architecture is the only configuration able to overcome the simple popularity-based baseline on DBbook. As previously stated, this dataset is very challenging since it is very sparse and many users have just a few positive ratings, thus it is very difficult to model users' preferences in a classic recommendation task. These characteristics of the dataset lead to poor performance all the other algorithms we take into account, except our deep architecture which is able to significantly overcome it. This interesting outcome further validates the results obtained by our framework and confirms the effectiveness of deep architectures for content-based recommendation tasks.

5 CONCLUSIONS AND FUTURE WORK

In this article we present AMAR, a modular and extensible architecture exploiting deep neural networks to provide users with content-based recommendations.

The model is based on LSTM networks, a particular class of RNNs particularly able to deal with sequences of data of arbitrary length, as the content describing the items to be recommended. Specifically, we develop an approach which jointly learns two embeddings representing both the items to be recommended as well as the preferences of the users. Given such representations, recommendations are provided by exploiting a logistic regression layer which calculates the likelihood that a user will like a certain item. In the experimental evaluation we compare our approach to several baselines, and results show that our deep architecture is able to significantly overcome both algorithms based on (shallow) neural networks as W2V and D2V as well as widespread and well-performing techniques for collaborative filtering and matrix factorization.

We plan to extend the proposed work along three different directions: optimization, architecture and additional features. Regarding optimization, training could be improved by applying early stopping and regularization techniques, by using different weight initialization schemes and by designing a proper cost function for the *top-N recommendation.* An additional improvement could be obtained by doing hyperparameter optimization. From the architectural perspective, different neural network architectures could be used to generate item representations to better extract relevant features from item descriptions. Another improvement could be obtained by adding more dense layers to the *concatenation layer* for capturing more complex relations between features to lead to a better classification. Furthermore, we will also investigate the adoption of other kind of neural network architectures to model the *textual features*, such as *Convolutional Neural Networks (CNNs)* which show their effectiveness in a broad range of natural language-related tasks [7, 26] and RNNs equipped with Gated Recurrent Units (GRUs) [11], which have a simpler structure than *LSTMs* but show comparable results. Preliminary results investigating the integration of further semi-structured features as the *genre* of the items to be recommended lead to little improvements in the overall accuracy of the system, thus encouraging us to design different variants of AMAR+ modeling additional information coming from important data silos such as Linked Open Data or Web and social media.

REFERENCES

[1] A. Almahairi, K. Kastner, K. Cho, and A. C. Courville. 2015. Learning Distributed Representations from Reviews for Collaborative Filtering. In *Proceedings of the 9th ACM Conference on Recommender Systems, RecSys 2015, Vienna, Austria, September 16-20, 2015.* 147–154.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* (2014).

[3] Trapit Bansal, David Belanger, and Andrew McCallum. 2016. Ask the GRU: Multi-task Learning for Deep Text Recommendations. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16).* ACM, New York, NY, USA, 107–114.

[4] Alejandro Bellogin, Pablo Castells, and Ivan Cantador. 2011. Precision-oriented evaluation of recommender systems: an algorithmic comparison. In *Proceedings of the fifth ACM conference on Recommender systems.* ACM, 333–336.

[5] Y. Bengio. 2009. Learning deep architectures for AI. *Foundations and trends® in Machine Learning* 2, 1 (2009), 1–127.

[6] Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning long-term dependencies with gradient descent is difficult. *Neural Networks, IEEE Transactions on* 5, 2 (1994), 157–166.

[7] Phil Blunsom, Edward Grefenstette, and Nal Kalchbrenner. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics.* ACL, 655–665.

[8] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12, 4 (2002), 331–370.

[9] Iván Cantador, Alejandro Bellogín, and Pablo Castells. 2008. Ontology-based personalised and context-aware recommendations of news items. In *Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology-Volume 01.* IEEE Computer Society, 562–565.

[10] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, and others. 2016. Wide & Deep Learning for Recommender Systems. *arXiv preprint arXiv:1606.07792* (2016).

[11] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[12] Ronan Collobert, Koray Kavukcuoglu, and Clément Farabet. 2011. Torch7: A matlab-like environment for machine learning. In *BigLearn, NIPS Workshop.*

[13] Marco De Gemmis, Leo Iaquinta, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. 2010. Learning preference models in recommender systems. In *Preference Learning.* Springer, 387–407.

[14] Marco de Gemmis, Pasquale Lops, Cataldo Musto, Fedelucio Narducci, and Giovanni Semeraro. 2015. Semantics-Aware Content-Based Recommender Systems. In *Recommender Systems Handbook.* Springer, 119–159.

[15] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, and Davide Romito. 2012. Exploiting the web of data in model-based recommender systems. In *Proceedings of the sixth ACM conference on Recommender systems.* ACM, 253–256.

[16] Tommaso Di Noia, Roberto Mirizzi, Vito Claudio Ostuni, Davide Romito, and Markus Zanker. 2012. Linked open data to support content-based recommender systems. In *Proceedings of the 8th International Conference on Semantic Systems.* ACM, 1–8.

[17] Omar U Florez. 2014. Deep Learning of Semantic Word Representations to Implement a Content-Based Recommender for the RecSys ChallengeâĂŹ14. In *Semantic Web Evaluation Challenge.* Springer, 199–204.

[18] Jianfeng Gao, Patrick Pantel, Michael Gamon, Xiaodong He, and Li Deng. 2014. Modeling Interestingness with Deep Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL.* 2–13.

[19] F. A Gers, J. Schmidhuber, and F. Cummins. 2000. Learning to forget: Continual prediction with LSTM. *Neural computation* 12, 10 (2000), 2451–2471.

[20] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning.* MIT Press. http://www.deeplearningbook.org.

[21] A. Graves, A. Mohamed, and G. Hinton. 2013. Speech recognition with deep recurrent neural networks. In *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on.* IEEE, 6645–6649.

[22] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber. 2015. LSTM: A search space odyssey. *arXiv preprint arXiv:1503.04069* (2015).

[23] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. 2015. Session-based Recommendations with Recurrent Neural Networks. *arXiv preprint arXiv:1511.06939* (2015).

[24] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.

[25] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *Data Mining, 2008. ICDM'08. Eighth IEEE International Conference on.* Ieee, 263–272.

[26] Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882* (2014).

[27] Quoc V Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053* (2014).

[28] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. *Nature* 521, 7553 (2015), 436–444.

[29] Pasquale Lops, Marco De Gemmis, and Giovanni Semeraro. 2011. Content-based recommender systems: State of the art and trends. In *Recommender systems handbook.* Springer, 73–105.

[30] Stuart E Middleton, David De Roure, and Nigel R Shadbolt. 2009. Ontology-based recommender systems. In *Handbook on ontologies.* Springer, 779–796.

[31] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[32] Cataldo Musto, Pierpaolo Basile, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. 2017. Introducing linked open data in graph-based recommender systems. *Information Processing & Management* 53, 2 (2017), 405–435.

[33] Cataldo Musto, Claudio Greco, Alessandro Suglia, and Giovanni Semeraro. 2016. Ask Me Any Rating: A Content-based Recommender System based on Recurrent Neural Networks. (2016).

[34] Cataldo Musto, Pasquale Lops, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. 2016. Semantics-aware graph-based recommender systems exploiting linked open data. In *Proceedings of the 2016 Conference on User Modeling Adaptation and Personalization.* ACM, 229–237.

[35] Cataldo Musto, Fedelucio Narducci, Pasquale Lops, Giovanni Semeraro, Marco De Gemmis, Mauro Barbieri, Jan Korst, Verus Pronk, and Ramon Clout. 2012. Enhanced semantic tv-show representation for personalized electronic program guides. In *International Conference on User Modeling, Adaptation, and Personalization.* Springer Berlin Heidelberg, 188–199.

[36] C. Musto, G. Semeraro, M. de Gemmis, and P. Lops. 2016. Learning Word Embeddings from Wikipedia for Content-Based Recommender Systems. In *Advances in Information Retrieval.* Springer, 729–734.

[37] 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.* Springer, 381–392.

[38] Xia Ning and George Karypis. 2011. Slim: Sparse linear methods for top-n recommender systems. In *Data Mining (ICDM), 2011 IEEE 11th International Conference on.* IEEE, 497–506.

[39] Makbule Gulcin Ozsoy. 2016. From Word Embeddings to Item Recommendation. *arXiv preprint arXiv:1601.01356* (2016).

[40] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the twenty-fifth conference on uncertainty in artificial intelligence.* AUAI Press, 452–461.

[41] Paul Resnick, Neophytos Iacovou, Mitesh Suchak, Peter Bergstrom, and John Riedl. 1994. GroupLens: an open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM conference on Computer supported cooperative work.* ACM, 175–186.

[42] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2015. Recommender systems: introduction and challenges. In *Recommender Systems Handbook.* Springer, 1–34.

[43] D Rumelhart, G Hinton, and R Williams. 1986. Learning representations by back-propagating errors. Lett. Nat., 323: 533-536. (1986).

[44] Alan Said and Alejandro Bellogín. 2014. Rival: a toolkit to foster reproducibility in recommender system evaluation. In *Proceedings of the 8th ACM Conference on Recommender systems.* ACM, 371–372.

[45] R. Salakhutdinov, A. Mnih, and G. Hinton. 2007. Restricted Boltzmann machines for collaborative filtering. In *Proceedings of the 24th international conference on Machine learning.* ACM, 791–798.

[46] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management* 24, 5 (1988), 513–523.

[47] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. 2001. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th international conference on World Wide Web.* ACM, 285–295.

[48] Jürgen Schmidhuber. 2015. Deep learning in neural networks: An overview. *Neural Networks* 61 (2015), 85–117.

[49] Barry Schwartz. 2004. *The paradox of choice: Why more is less.* HarperCollins.

[50] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, and others. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489.

[51] Richard Socher, Christopher D Manning, and Andrew Y Ng. 2010. Learning continuous phrase representations and syntactic parsing with recursive neural networks. In *Proceedings of the NIPS-2010 Deep Learning and Unsupervised Feature Learning Workshop.* 1–9.

[52] F. Strub and J. Mary. 2015. Collaborative Filtering with Stacked Denoising AutoEncoders and Sparse Inputs. In *NIPS Workshop on Machine Learning for eCommerce*.

[53] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*. 3104–3112.

[54] K. S. Tai, R. Socher, and C. D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. *arXiv preprint arXiv:1503.00075* (2015).

[55] Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. 2013. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*. 2643–2651.

[56] P. Vincent, H. Larochelle, I. Lajoie, Y. Bengio, and P. Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *The Journal of Machine Learning Research* 11 (2010), 3371–3408.

[57] 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*. ACM, 1235–1244.

[58] X. Wang and Y. Wang. 2014. Improving content-based and hybrid music recommendation using deep learning. In *Proceedings of the ACM International Conference on Multimedia*. ACM, 627–636.

[59] T.-H. Wen, A. Heidel, H. Lee, Y. Tsao, and L.-S. Lee. 2013. Recurrent neural network based language model personalization by social network crowdsourcing.. In *INTERSPEECH*. 2703–2707.

[60] J. Weston, A. Bordes, S. Chopra, and T. Mikolov. 2015. Towards AI-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698* (2015).

[61] J. Zhang, H. Cai, T. Huang, and H. Xue. 2015. A Distributional Representation Model For Collaborative Filtering. *CoRR* abs/1502.04163 (2015). http://arxiv.org/abs/1502.04163

[62] X. Zhang, J. Zhao, and Y. LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in Neural Information Processing Systems*. 649–657.