Together is Better: Hybrid Recommendations Combining Graph Embeddings and Contextualized Word Representations

Marco Polignano
marco.polignano@uniba.it
University of Bari Aldo Moro
Bari, Apulia, Italy

Cataldo Musto
cataldo.musto@uniba.it
University of Bari Aldo Moro
Bari, Apulia, Italy

Marco de Gemmis
marco.degemmis@uniba.it
University of Bari Aldo Moro
Bari, Apulia, Italy

Pasquale Lops
pasquale.lops@uniba.it
University of Bari Aldo Moro
Bari, Apulia, Italy

Giovanni Semeraro
giovanni.semeraro@uniba.it
University of Bari Aldo Moro
Bari, Apulia, Italy

ABSTRACT

In this paper, we present a *hybrid* recommendation framework based on the combination of *graph embeddings* and *contextual word representations*. Our approach is based on the intuition that each of the above mentioned representation models heterogeneous (and equally important) information, that is worth to be taken into account to generate a recommendation. Accordingly, we propose a strategy to combine both the features, which is based on the following steps: first, we separately generate graph embeddings and contextual word representations by exploiting state-of-the-art techniques. Next, these embeddings are used to feed a *deep architecture* that learns a hybrid representation based on the combination of the single groups of features. Finally, we exploit the resulting embedding to identify suitable recommendations. In the experimental session, we evaluate the effectiveness of our strategy on two datasets and results show that the use of a *hybrid* representation leads to an improvement of the predictive accuracy. Moreover, our approach overcomes several competitive baselines, thus confirming the validity of this work.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; • **Computing methodologies** → *Natural language processing*.

KEYWORDS

graph embeddings, BERT embeddings, USE embeddings, deep learning, recommender systems

ACM Reference Format:

Marco Polignano, Cataldo Musto, Marco de Gemmis, Pasquale Lops, and Giovanni Semeraro. 2021. *Together is Better:* Hybrid Recommendations Combining Graph Embeddings and Contextualized Word Representations. In *Proceedings of ACM RecSys 2021 - ACM RecSys 2021: 15th ACM Conference on Recommender Systems, September 27th - October 1st 2021, 2021, Amsterdam, Netherland and Online.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3460231.3474272

ACM RecSys 2021, Sept. 27th - Oct. 1st 2021, 2021, Amsterdam and Online

ACM ISBN 978-1-4503-8458-2/21/09…$0.00
https://doi.org/10.1145/3460231.3474272

1 INTRODUCTION

The ever-increasing amount of data and information available on the web makes it always more important to develop systems that efficiently support users in finding and consuming only relevant elements. Among the different solutions for dealing with this task, *recommender systems (RS)* emerged over the years. Their effectiveness is confirmed by several shreds of evidence: as an example, 35% of Amazon's revenues are generated through its recommendation engine[1], and many companies frequently report claims that such systems contribute from 10% to 30% of total revenues [18]. Similar success stories regarding other web companies such as Facebook, Spotify and Netflix further confirm that RS nowadays play a central role to improve the effectiveness of web platforms.

In short, RS use the preferences expressed by a user, explicitly or implicitly, to filter the available information and proactively propose elements that might be of interest for her. Several recommendation models have been proposed in the scientific literature. Most of them have been developed by using the so-called *wisdom of the crowd*. In particular, collaborative filtering approaches [27] are based on the idea that by observing the activities of a group of users similar to the active one, it is possible to identify a set of recommendations relevant to her. Instead, content-based approaches [11, 34] deal with user and product characteristics to identify the set of properties that are of interest for the user. Hybrid approaches [6, 39] try to put together the ideas behind collaborative and content-based RSs. In particular, they use both the wisdom of the crowd and the properties of users and products with a comprehensive approach. Hybrid recommender systems combine the best features of content-based and collaborative and deliver highly accurate results. However, one of the main issues related to the development of hybrid recommendation models concerns the *representation* of both content-based and collaborative features. *How can we represent (and combine) both these groups of features in a hybrid recommender system?*

In this work, we address such a research question and we propose a hybrid recommendation model based on the combination of *graph embeddings* and *contextual word representation*. The first are based on the use of graphs as a tool to represent products, concepts

[1] http://www.mckinsey.com/industries/retail/our-insights/how-retailers-can-keep-up-with-consumers

and information including their properties and relationships, and showed very good performance in several machine learning tasks [16]. Roughly speaking, graph embedding techniques [7] take a graph as input and produce a vector-space representation of each *node* as output. By referring to a recommendation scenario, they produce a vectorial representation of users and items such that users sharing similar preferences (and items voted by the same users, respectively) are close in the space. The second is based on the idea of using machine learning as a tool to create compact numerical representations of terms and emerge as an evolution of widespread *word embedding* (WE) techniques such as Word2Vec [29]. In particular, recent *contextual word representation* [40] learn a vector space representation based on word usage in large corpora of textual data, and produce a representation such that terms sharing a similar meaning are close in the space. However, differently from static WE techniques, contextual word representations take into account the *context* in which a word is used. In particular, such techniques allow generating a different vector representation of the word depending on the terms that co-occur with it in the specific sentence we are going to encode. This process is possible because of the use of large pre-trained language models, which can learn highly transferable and task-agnostic properties of a language [14]. As shown by several work, these techniques obtained outstanding results in many natural language processing-related scenarios [44].

In this work we tried to take the best out of these techniques since we combined them into a *hybrid* recommendation model. Our methodology is based on the following steps: first, graph embeddings and contextual word representations are *separately* generated by exploiting the previously mentioned techniques. Next, such pre-trained embeddings are used to feed a *deep architecture* that combines the single groups of features in a unique hybrid representation through a *concatenation layer*. Finally, the resulting representation is exploited to identify suitable recommendations. The novelty of the proposed approach lies: *(i)* the combination of graph embedding and contextual word representations in a single recommendation model. In particular, the adoption of contextual word representation is a particularly new contribution in the area of hybrid recommender systems; *(ii)* the analysis of the effectiveness of different strategies to concatenate item and user representations in a hybrid recommendation model. Another strong point of the proposed methodology lies in the adoption of *pre-trained* embeddings learnt by exploiting exogenous state-of-the-art techniques. In this way, we can lighten the load on our deep architecture, reduce training time and increase the scalability of the model.

In the experimental session, we evaluate the effectiveness of our recommendation strategy on two different datasets and the results show that the combination of graph embeddings and contextual word representations leads to an improvement of the predictive accuracy. Moreover, our approach overcomes several competitive baselines, thus confirming the validity of the intuitions behind this work.

To sum up, the contributions of this article can be summarized as follows:

- We introduce a recommendation framework inspired by *deep architectures*, that learns a hybrid representation of users and items based on collaborative and content-based features;
- We design different strategies to combine graph embeddings and contextual word representations into a single embedding;
- We evaluate our strategy against two datasets, by ensuring the reproducibility of the whole experimental protocol.

The rest of the article is organized as follows: Section 2 describes related work; Section 3 presents the adopted methodology; Section 4 discusses the findings emerging from the experimental evaluation, while conclusions and future work are sketched in Section 5.

2 RELATED WORK

Recommender systems [37] represent one of the most disrupting technologies that appeared on the scene in the last decade [23]. Basically, such systems acquire information about user needs, interests and preferences and tailor their behavior based on such information, thus supporting people in several decision-making tasks [35, 36].

The current state of the art is the consequence of a very long path: indeed, early research was characterized by the sharp dichotomy between *collaborative filtering algorithms* [27], based on the analysis of the ratings expressed by the users, and *content-based techniques* [11], which exploited textual information (*e.g.*, the plot of a movie) and descriptive features of the items. Subsequently, *hybrid RSs* [6] emerged as a mean to combine both the approaches. Indeed, they are based on the intuition that content-based and collaborative features provide heterogeneous and complementary evidence, and can equally contribute to generate accurate recommendations. In this work, we fall into this research line and we propose a *hybrid recommendation framework* that combines together collaborative features based on *graph embeddings* and content-based features based on *contextual word representations*.

As shown by several shreds of evidence (see [17, 30, 53], just to name a few), recommendation models exploiting graph embedding techniques outperform typical baselines. Based on the findings of previous research [33], this work exploits *translation models* such as TransE [5] and TransH [49]. More details about the algorithms will be provided next.

In parallel, *contextual word representations* [40] emerge as an evolution of widespread *word embedding* (WE) techniques. Even if WE techniques have been largely used to provide users with recommendations [32], the adoption of contextual word representations such as BERT [13], Elmo [4], Universal Sentence Encoder (USE) [9] in recommendation tasks is poorly investigated. As an example, an approach that exploits ELMo embeddings based on item features and user reviews is presented in [42], while the exploitation of BERT embeddings in a recommendation task is discussed in [8]. In this work, USE and BERT were chosen as contextual word representation methods since they emerged as the best-performing techniques in [19], where a benchmark for a paper RS is presented.

With respect to the above mentioned literature, that *separately* exploits *graph embeddings* and *contextual word representation*, in this work we proposed a strategy to learn a hybrid representation

that combines both the sources of information through a deep architecture.

However, the idea of exploiting deep architectures to learn a vector-space representation of users and items is not completely new. As an example, a deep architecture for Neural Collaborative Filtering, which applies the principles of deep learning to the collaborative filtering framework is presented in [21]. However, differently from [21], which is only based on collaborative features, we also encoded in the model content-based features coming from contextual word representations.

Another distinctive trait of our work lies in the adoption of a *concatenation layer* to combine the different groups of features in a deep architecture. This intuition, which is also widely adopted in multi-modal deep learning [41], was also investigated in [21]. Another similar architecture is presented in [31]. In this case, the model learns a representation for textual content by exploiting Recurrent Neural Networks. With respect to this work, we preferred to exploit pre-trained embeddings learnt by BERT, which showed to overcome other techniques in several tasks [44]. Finally, our work shares some ideas with those presented in [52].

As for the attempts to combine graph-based and word embeddings, Wang et al. [46] propose to merge semantic-level and knowledge-level representations of news through their knowledge-aware convolutional neural network (KCNN). Results show that the usage of graph-based entity embedding and contextual embedding can improve AUC. The same authors recently investigated [47] the use of graph embeddings in a recommendation scenario based on many state-of-the-art datasets (*i.e.,* MovieLens-20M, BookCrossing, and Last.FM, Dianping-Food). Also in this case, their Graph Neural Networks with Label Smoothness Regularization demonstrated the effectiveness of graph embedding in RSs. Similarly, Cenikj et al. in [8] propose to enhance a graph-based RS for Amazon products with two state-of-the-art representation models: BERT and GraphSage. The results obtained by the authors encourage to follow the idea to merge graph and contextual word representation techniques. With respect to these pieces of work, we considered more contextual word representation techniques (*i.e.,* BERT and USE) and different graph embedding techniques *i.e.,* TransE and TransH). This allows to better assess the impact of the single methodologies on the overall accuracy of the models and to get more precise take-home messages concerning the adoption of such techniques and the effectiveness of their combination.

3 METHODOLOGY

In this section we describe our methodology to learn a hybrid representation of users and items. We first introduce some basics of our recommendation framework. Next, we describe the deep architecture that combines such embeddings to provide users with recommendations.

3.1 Basics of the Recommendation Framework

Given a set of users $U = \{u_1 \dots u_m\}$ and a set of items $I = \{i_1 \dots i_n\}$, the goal of the framework is to build a *hybrid* vector-space representation for each $u \in U$ and for each $i \in I$. Such representations, that can be formally defined as $\overrightarrow{u}$ and $\overrightarrow{i}$, are used to feed a deep architecture that identifies suitable recommendations. In particular, each $\overrightarrow{u}$ is encoded starting with a *collaborative* embedding $\overrightarrow{u_c}$ and a *content-based* embedding $\overrightarrow{u_t}$, and the same principle holds for the representation of $\overrightarrow{i}$, that can be encoded starting with $\overrightarrow{i_c}$ and $\overrightarrow{i_t}$. In our recommendation framework, these representations will be concatenated by using different strategies.

Graph Embedding techniques. Vectors $\overrightarrow{i_c}$ and $\overrightarrow{u_c}$ encode the *collaborative* information which can be obtained by mining the graph connecting the users to the items they like. Accordingly, we first create $G = (N, E)$, where $N = I \cup U$, and $E = \{e_1 \dots e_k\}$. E encodes the available ratings, so each e_i connects a $u \in U$ to a $i \in I$, based on the preferences expressed by u. Of course, we only represent *positive* ratings. Next, a graph embedding technique is run to represent nodes (that is to say, users and items) encoded in the above described graph as *vectors*. Due to space reasons, we can't go into details of all the available methods, and we suggest to refer to [7] for a comprehensive survey.

In this work we based on the findings of previous research [33], and we investigate *translation models* such as TransE [5] and TransH [49]. These models embed nodes and relations by seeing relations as *translations in the vector space*. It should be pointed out that, in our case, the only relation that is encoded in the graph is the *feedback* relation, used to connect the users to the items they like.

In order to build a vectorial representation of nodes, these models need to be trained. Training is carried out by providing as input some *positive triples*, expressed as (u, f, i), where u and i are nodes and f is a relation. Each triple represents a *fact* that exists in the graph, thus, in our specific setting, for each rating $r \in R$ indicating that user u expressed a feedback f on item i, a triple is included in the training set. In parallel, a set of *negative triples* (that is to say, facts that do not exist in the graph) is built by using *negative sampling* strategy [29]. Next, models are trained by minimizing a *pairwise ranking loss function*, which compares the scores (as defined below) of positive triples to the scores of negative ones. Given this general setting, TransE learns a representation of entities and relations so that $\overrightarrow{u} + \overrightarrow{f} \approx \overrightarrow{i}$. This is done by using as scoring function the formula $f(u, f, i) = D(u + f, i)$, where D is a distance function such as the L1 or the L2 norm. The second graph embedding technique we employed is TransH, which is an extension of TransE that can more effectively deal with *one-to-many* and *many-to-many* relations. By considering that in a recommendation setting each user typically rates more than one item (and viceversa), we can assume that TransH will likely produce a more precise representation of users and items. Formally, this is obtained by projecting entities on a hyperplane identified by the normal vector w_u. Accordingly, the score function becomes: $f(u, f, i) = D(u \perp + f, i \perp)$, where $u \perp = u - w_f^T u w_f$ and $i \perp = i - w_f^T i w_f$ and D is a distance function such as the L1 or the L2 norm.

As shown in [33], TransE and TransH can provide similar results. However, results may be dependant on the datasets, so we evaluated the effectiveness of the embeddings built by exploiting both techniques. For further details about TransE and TransH we suggest again to the original papers describing the methods ([5, 49]).

Contextual Word Representations. The second part of the representation is obtained by learning $\overrightarrow{i_t}$ and $\overrightarrow{u_t}$. These vectors encode *content-based* information which are obtained by processing descriptive features of the item, such as the plot of a movie. So, we assume that for each item $i \in I$ some textual content $text(i)$ is available (*e.g*, plot of a movie). We decided to encode such textual information in a vectorial form that might preserve the meaning of the content feature. As previously stated, we chose BERT[13] and USE [45] as encoding strategies. This choice is motivated by their ability to express syntactic and semantic information of words observed in extensive document collections. Contextual word representations have proven to be a key component of the state-of-the-art natural language processing (NLP) approaches that require text comprehension and formalization. For the sake of brevity, we just provide a synthetic overview of the techniques, but for further details, we suggest to refer to the original papers (*i.e.,* [9, 13]).

BERT (acronym of Bidirectional Encoder Representations from Transformers) [13] is a language model based on the Transformer's deep learning architecture [43]. Its *base* version is designed as a stack of 12 encoding layers, each of which learns an increasingly precise and refined representation. Each layer is composed of a multi-head attention module, a normalization, and a feed-forward layer. In order to use BERT lo learn $\overrightarrow{i_t}$ we followed these steps: *(i)* We pre-processed $text(i)$ by making it lowercase and by removing punctuation, stopwords, and non-ASCII characters; *(ii)* We truncated the cleaned text we previously obtained to the first 128 words. This is due to a constraint of BERT, that can't accept longer input sentences; *(iii)* For each word $w \in cleaned\_text(i)$ we obtain its representation $\overrightarrow{bert(w)}$ by exploiting pre-trained BERT base model.In particular, we extract the output of the last encoding layer for word w; *(iv)* Finally, we obtain the embedding of item i as the centroid vector of the words w. Formally: $\overrightarrow{i_t} = \sum_{j=1}^{k} \overrightarrow{bert(w_j)}/k$.

In some cases, a centroid-based representation, as that we built based on BERT, is not the best approximation of the real meaning of a sentence. To tackle this problem, it is necessary to exploit deep neural models that learn a *sentence-based* representation, such as USE (Universal Sentence Encoder Model). USE is typically trained on *Wikipedia* by using a modified *Skip-thought strategy* [26]. The idea behind this approach is to predict the previous and the next sentence starting from the current one as input. In order to learn the model, each sentence s in the training set is used as input of a Transformer made of six transformers layers, each of which has a self-attention module followed by a feed-forward network. The model is the best solution for encoding sentences of different lengths without truncating or padding them. The last layer obtained from the model is a 512-dimensional vector what represents the whole sentence $\overrightarrow{s}$. Once the model is trained, we can use it to map a new sentence into the existing model. In particular, we use $cleaned\_text(i)$ as input of the Google USE model, and we obtain as output the item embedding $\overrightarrow{i_t}$ based on USE.

Once item embeddings are available, in both cases we adopted the following strategy to obtain a user profile $\overrightarrow{u_t}$: In particular, we borrowed from content-based RSs [11] that idea of building the profile as the centroid vector of the items positively rated by the user. Formally, for each user $u \in U$ all item embeddings of positively rated items $\overrightarrow{i_{rated}}$ are collected. Next, we calculate $\overrightarrow{u_t} = \sum_{j=1}^{z} \overrightarrow{i_{rated}}/z$, in order to build a vectorial representation of users.

3.2 Description of the Architecture

Once the single *pre-trained* embeddings are available, our deep architecture comes into play. As previously stated, it aims to: *(i)* learn a hybrid representation of users and items, based on graph and word embeddings; *(ii)* identify suitable recommendations, by predicting the interest of user u in items $i \in I$.

In order to also assess about the effectiveness of the single embedding strategies, we first defined a very simple architecture that we exploited as baseline in our experiments. In the following, we refer to this architecture as *basic*. Our *basic* architecture is shown in Fig. 1. Specifically, given one of the above mentioned representation techniques (graph embedding or word embedding, alternatively), we proceed through a single *concatenation* layer to merge the information regarding user and item. At this point, through a dense layer with a *sigmoid* activation function, we proceed to the prediction step. As shown in the Figure, no combination between different techniques is performed in this case.

However, such a representation does not take the best out of the *single* data sources. Accordingly, we propose different strategies to combine both data representations, and therefore we have designed an architecture that we called *hybrid*. The overall architecture is presented in Figure 2-3.

The process starts by encoding a representation of a user u and a item i. Such a representation is based on the embeddings $\overrightarrow{u_c}$, $\overrightarrow{u_t}$, $\overrightarrow{i_c}$, $\overrightarrow{i_t}$, learnt as described in Sec. 2.1. Then, these vectors are passed as input of a *dense* layer, in order to reduce their dimension and make them homogeneous. Next, the embeddings pass through the *core* of the whole architecture, which is a *concatenation* layer whose goal is to combine the single sources of information into a *hybrid* representation. In particular, in this work we introduce two different *concatenation strategies* (Fig. 2-3), defined as follows:

Entity-based concatenation: the idea behind this strategy is to first merge *user* embeddings learnt through different techniques, before concatenating them to *item* embeddings. The central insight guiding us in the development of this configuration is that merging different representations of the same entity in the first layer of the network could help to better represent the entity itself. Specifically, we think that heterogeneous information obtained by combining different groups of features could help the network to more easily identify the characterizing dimensions of items and users. To this end, we first use the Concatenation Layer (1) to put together vectors $\overrightarrow{u_c}$ and $\overrightarrow{u_t}$. Next, in Concatenation Layer (2) we merge $\overrightarrow{i_c}$ and $\overrightarrow{i_t}$. Finally, by using the Concatenation Layer (3) we further merge the representations based on different encoding strategies into a single vector, just before the top of our architecture.

Feature-based concatenation: in this case, we start by concatenating $\overrightarrow{i_c}$ and $\overrightarrow{u_c}$, that is to say, the concatenation of both graph embeddings through the Concatenation Layer (1). Next, we merge $\overrightarrow{i_t}$ and $\overrightarrow{u_t}$, that is to say, the concatenation of contextual word representations, by using the Concatenation Layer (2). Finally, also

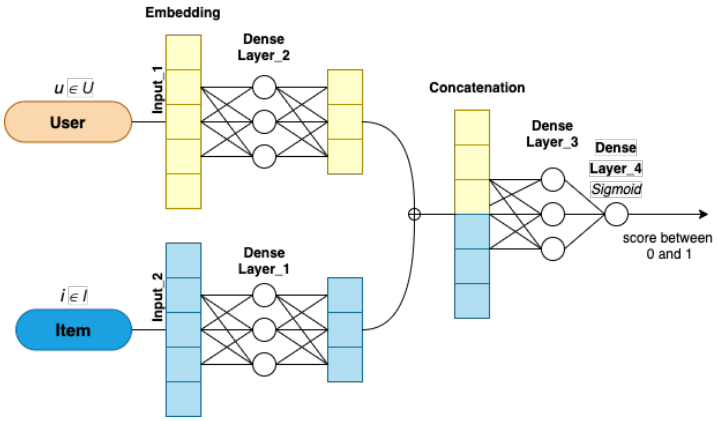

Figure 1: Our *basic* deep architecture to provide recommendations.

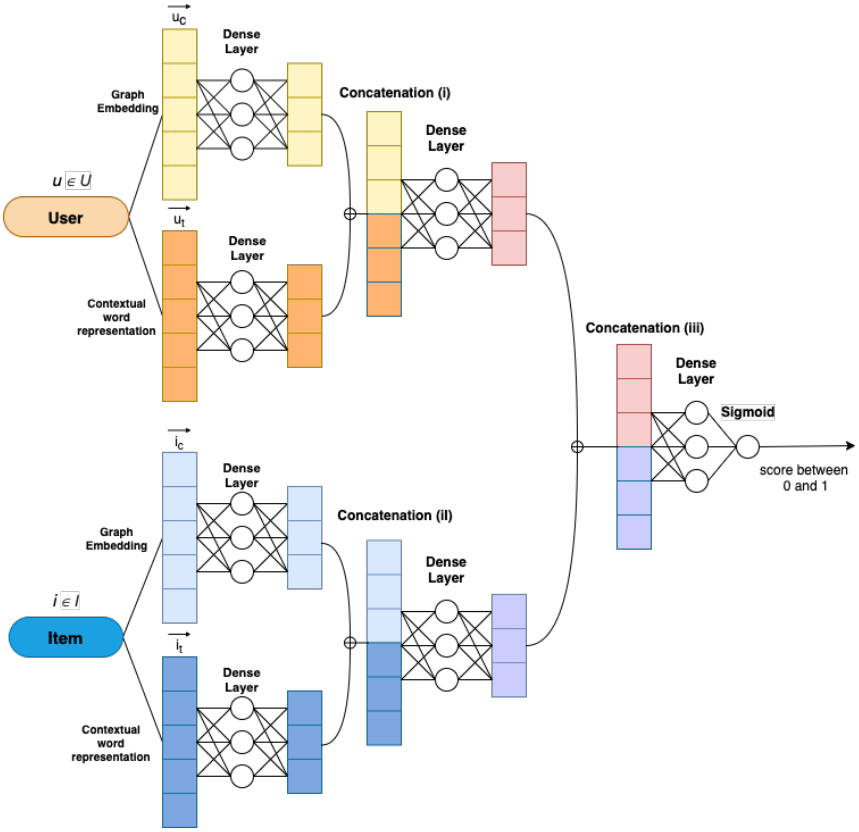

Figure 2: Our *hybrid* deep architecture based on the Entity-based concatenation model.

in this strategy, we use the Concatenation Layer (3) for merging weights coming from different encoding strategies. In other terms, we decided to merge together the embeddings based on the type of algorithm which generated them. In this case, the model relies on the idea that first concatenating the embeddings of the different entities (*i.e.,* items and users) obtained by the same representation can help the network to strengthen the learning of the relationships between them. Such relationships could lead to a more accurate prediction of the relevance of a recommendation for a specific user. Of course, many other concatenation strategies can be designed based on the combination of different groups of features. For the sake of simplicity, in this work we just evaluate these methods. The analysis of further combinations is left as future work.

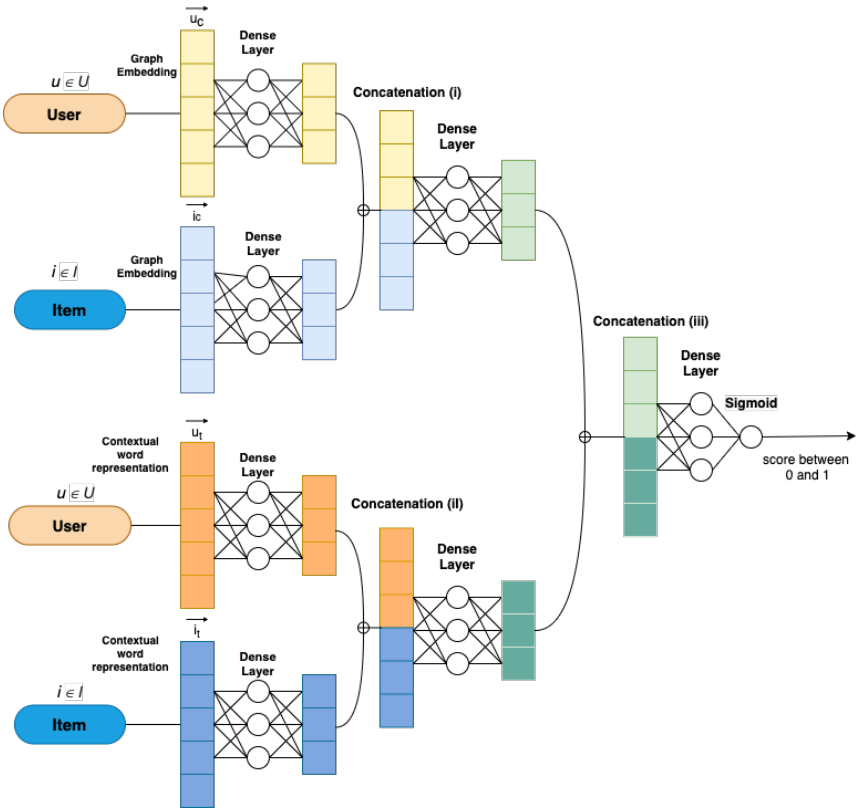

Figure 3: Our *hybrid* deep architecture based on the Feature-based concatenation model.

Obtaining the Recommendations. However, regardless the specific concatenation strategy which is adopted, a simple concatenation of the original features is not enough to learn the underlying relationships among the features, so the hybrid vectors are passed again through a new *dense layer* just before the final prediction. As a final layer, we used a last dense layer of size 1 having a *sigmoid* function as activation function. This layer estimates the probability that the item i is relevant for user u. Before making predictions, the architecture needs to be trained by exploiting all the ratings in the form (u, i) available in the dataset. Once the model is learned, it can predict to what extent user u would like unseen items $j \in I$. More details about the training process are provided in the next section.

4 EXPERIMENTAL EVALUATION

In the experimental session we evaluated the effectiveness of our methodology in the task of item recommendation. In particular, experiments were designed to answer to the following research questions: *how do the different strategies to combine graph embeddings and contextual word representations perform in a recommendation task?* **(Experiment 1)**; *how does our hybrid representation perform with respect to other state-of-the-art techniques?* **(Experiment 2)**

4.1 Experimental Design

Datasets. Experiments were carried out in a *movie recommendation* and in a *book recommendation* scenario. In the former case, Movielens 1M (ML1M)[2] was exploited as dataset, while in the latter we used DBbook dataset[3]. Table 1 depicts some statistics about the datasets. Generally speaking, 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. For both the datasets, we used a 80%-20% training-test split. Data were split in order to maintain the ratio between positive and negative ratings. As for `MovieLens-1M` we considered as *positive* only the ratings equal to 4 and 5 out of 5. As for DBbook, ratings were provided in a binary format (positive/negative). The predictive accuracy of the algorithms was evaluated on top-5 and top-10 recommendation list, calculated by following the *TestRatings* strategy [3]. For the sake of brevity, we just present the result of the top-5 list since the analysis on top-10 leads to similar findings. In order to obtain the *top-k* items, we first predict through our deep architecture interest scores for all the items i in the Test Set of user u, then we rank the items based on the output of the neural network.

Configurations. We evaluate six different configurations of the framework on varying of the encoding techniques as well as the

[2]http://grouplens.org/datasets/movielens/
[3]http://challenges.2014.eswc-conferences.org/index.php/RecSys

	Users	Items	Ratings	%Positive	Sparsity	Avg. ratings/user	Avg. ratings/item	Content feature
ML1M	6,040	3,883	1,000,209	57.51%	96.42%	157.11±182.93	293.98±393.88	plot
DBbook	6,181	6,733	72,372	45.85%	99.83%	11.70±5.85	10.74±27.14	abstract

Table 1: Statistics of the datasets

concatenation strategy. In particular, we evaluated the representation learnt based on TransE, TransH, BERT and USE by exploiting both the *entity-based* and *feature-based* concatenation strategies. Moreover, in order to assess the effectiveness of our *hybrid* representation, we also evaluate an alternative implementation of our deep architecture where the embeddings based on just *one* technique (*e.g.,* just graph embeddings or just contextual word representations) are combined as showed in Fig. 1. In the following, we refer to this configuration as *basic* configuration.

Overview of the Parameters. Our models were trained for 25 epochs, by setting batch sizes to 512 for ML1M and to 1536 for DBbook, respectively. The parameter α is set to 0.9 and learning rate is set to 0.001. As cost function we choose the *binary cross entropy*. As optimizer, we used ADAM for ML1M and RMSprop for for DBbook. In the *basic* configuration (see Fig. 1) the dimension of dense layers is set to 64 neurons for the layers labeled as *dense_layer_1* and *dense_layer_2*, and it is set to 32 for *dense_layer_3*. Finally, the dimension of the last layer is set to 1 neuron in order to make predictions. A similar setting has been set for hybrid configurations (see Fig. 2-3). In this case dense layers from 1 to 6 are set to 64 neurons, while the others follow the same size introduced in the previous configuration. Finally, the dimension of the embeddings was set to 768 for BERT and 512 for USE. Due to space reasons we do not report more results, but our experiments showed that different dimensions of the embeddings did not significantly affect the overall accuracy.

Source Code and Baselines. The source code of our hybrid recommendation algorithm is available on GitHub[4]. Moreover, in Experiment 2 we compared our methodology to several baselines based on Deep Neural Networks implemented into the NeuRec library[5], such as:

- **(*MLP*) Multi-Layer Perceptron** [21]: the authors present a multimodal deep learning model for developing an item-based collaborative filtering system. In particular, they propose a deep network that combines the descriptive features of items and users by concatenating them. A standard multi-layer perceptron is used to learn the interaction between latent features. In order to run this model we set the number of learning epocs equal to 100 and the batch size to 256;
- **(*NeuMF*) Neural Matrix Factorization** [21]: is an item-based collaborative filtering deep learning model able to combine matrix factorization and multi-layer perceptron. In particular, the authors learn separate embeddings for matrix factorization and multi-layer perceptron and combine them by concatenating their last hidden layer. We configure the model to generate embedding of 16 dimensions and run the model for 100 epochs of learning and 256 elements as a batch;
- ***(NAIS)* Neural Attentive Item Similarity** [20]: it is an item-based collaborative filtering system. NAIS is an attention network, which is capable of distinguishing which historical items in a user profile are more important for a prediction. It uses a strong representation power by adding to state-of the art models only a few additional parameters brought by the attention network. The model has been run for 8 training epochs with an embedding size of 16 and a batch size of 256 elements;
- ***(DeepICF)* Deep Item-based Collaborative Filtering** [51]: is a deep learning model for item-based collaborative filtering that can model higher-order relationships among items. They use multiple nonlinear layers above the pairwise interaction modeling to learn higher-order item relations. We run the model by setting batch size to 16 dimensions, 16 training epochs and a batch size of 256 items;
- ***(FISM)* Factored Item Similarity Models [25]**: is an item-based method for generating top-N recommendations that learns the item-item similarity matrix as the product of two low dimensional latent factor matrices. The optimal parameter configuration of the model is obtained by setting an embedding size of 16, 16 learning epochs and 256 as batch size;
- ***(DAE)* Denoising Auto-Encoder** [50]: is a method for item-based top-N recommendation that utilizes the idea of Denoising Auto-Encoders. The model learns correlations between the user's item preference by training on a corrupted version of the known preference set by using a one-hidden-layer neural network. The model has been trained for 1000 training epochs on batch size of 256 elements;
- ***(CFGAN)* Collaborative Filtering Framework based on Generative Adversarial Networks** [10]: is a GAN-based collaborative filtering (CF) framework to provide higher accuracy in recommendation. CFGAN aims at generating a purchase vector with real-valued elements, by distinguishing between the vectors generated by the model and the real ones from the ground truth. It has been configured in order to train for 100 epochs with a batch of 64 items;
- ***(WRMF)* Collaborative Filtering for Implicit Feedback Datasets** [22]: is a collaborative filtering approach for implicit feedback dataset. In particular the authors estimate the relevance of an item for each user by using a confidence level. A latent factor that uses these scores has been consequently used. The model has been configured for working with an embedding size of 8 and 300 epochs of training.

Moreover, in order to compare our approach to other methods exploiting *side information*, we also considered the following baselines available in the Elliot framework [1] [6]:

[4]https://github.com/ANONYMOUS
[5]https://github.com/wubinzzu/NeuRec
[6]https://github.com/sisinflab/elliot

- ***kNN***: is the well known nearest neighbor classifier. The kNN classifier search for the k closest elements from a given element to be classified. These elements are commonly known as nearest neighbors. The item label is consequently assigned according to the class of its neighbors [38]. In our experiment we investigate the **item-knn**, the **user-knn** and their hybrid variants: **att-item-knn**, **att-user-knn**. In particular attribute-based KNN approaches can access other types of side information in the form of item attributes [15]. For our purposes, we enriched each item with ontological, categorical and factual information coming from extracted from DBpedia by selecting, from the knowledge base, the item properties. We configured the model for using a cosine similarity measure and a number of neighbor equal to 100.
- ***VSM***: is the classic vector space model. Every document is represented as a vector of term weights, where each weight indicates the degree of association between the document and the term [38]. In our run the features used for training the model are the same side information already used for the kNN classifiers. Also in this case we used the cosine as similarity metric.
- ***kaHFM***: is a Knowledge-aware Hybrid Factorization Machines RS. The model extends classic factorization machines by using the semantic information encoded in a knowledge graph. More details can be found in [2]. In our run the model use the same side information already used for the kNN classifiers.

For sake of simplicity we reported here only the *best* parameters of the baselines used for running them in the reported comparison. Indeed, for each baseline model we run a grid-search optimization by using F1-measure as score function.

Evaluation Metrics. In order to evaluate the effectiveness of the models we used *Precision*, *Recall* and *F1 score* [12]. These metrics allow us to evaluate the model by considering whether or not what is suggested is relevant to the user. Their weakness is that they do not consider the position of the items in the list of suggestions as an element to be evaluated. Accordingly, we decided to extend our metrics by calculating mean reciprocal ranking (*MRR*) and the normalized discounted cumulative gain (*nDGC*) [24, 48] scores. Both of them are ranking measures that reward recommendation lists where relevant items are put in the first positions. Given that in our task a binary score is available for each item (like or dislike), we used the binary relevance version of nDCG [48].

4.2 Discussion of the Results

In Experiment 1 we evaluated the effectiveness of our hybrid representation on varying of the concatenation strategy of $\overrightarrow{u}$ and $\overrightarrow{i}$ and of the encoding techniques. As shown in Table 2 and Table 3, the first outcome emerging from the experiment concerns the effectiveness of our *hybrid* representations, since all the configurations we took into account overcame the *basic* representation strategies. Just as a recall, basic representation exploits the pre-trained embeddings based on *TransE*, *TransH*, *BERT*, *USE* without any combination or concatenation. This outcome can be observed for F1 measure (see Table 2), MRR and nDCG (see Table 3). In particular, by considering the ML1M dataset, our best configuration (TransH + BERT with an entity based concatenation) shows an increment in performance ranging from 1.44% (if compared to TransH alone) to 8.33% (if compared to USE alone, in terms of F1 measure). Similar findings can be observed for nDCG and MRR. In the first case, we observe an increase of performances of 1.78% if considering TransH and 10.31% if considering USE. Finally, as for MRR, it emerged a gap in performances of 1.08% comparing with TransH and 6.33% by comparing results with USE.

As for DBbook, TransE + BERT with feature-based concatenation emerged as the best-performing configuration. In this case, by considering F1-measure, we can observe an increase of 0.8% w.r.t. TransE 4.64% w.r.t. USE. The trend observed for nDCG is equivalent. Moreover, we also noted an increase in performance w.r.t. both TransE (0.65% increase) and USE (4.6% increase). Finally, by analyzing MRR, we observe a slightly smaller difference. In fact, we keep only an increase in performance of 0.77% considering TransE and 3.8% considering USE.

Overall, it emerges that *feature-based concatenation* obtained the best results. Positive findings also emerge for entity-based concatenation, which overcame the basic representations. As regards the techniques, configurations based on BERT obtained the best results on both the datasets. The result obtained does not surprise us, in fact as known in the literature, the use of a bi-directional Transformer in the BERT model is able to better represent relationships between terms and obtain more effective results [28]. Conversely, as for the graph embedding techniques, TransH obtained the best results on MovieLens, while TransE overcame the other configurations on DBbook. This is probably due to the characteristics of the dataset: indeed, MovieLens data are more *dense*, thus more *one-to-many* relations are encoded in the dataset. Accordingly, it is likely that the ability of TransH to better learn these relationships leads to a better representation. This outcome is also confirmed on the basic configurations exploiting graph embeddings.

Finally, in order to validate the results, we performed a Wilcoxon signed-rank test. The statistical evaluation demonstrated that there is a difference in performances among our best configuration and *basic* approaches by considering $p < 0.01$. Based on these results, we can state that the intuition of learning a *hybrid representation* to encode information about user and items into a deep recommendation framework was supported by the results. A statistically significant gap also emerged by comparing the best-performing configuration to most of the alternative settings. In particular, as for MovieLens, 6 out of 8 comparisons led to a significant gap in terms of F1, NDCG and MRR. As for DBbook, we noted a relevant pattern since the best configuration significantly overcame all the other configurations, with the exception of those exploiting *feature-based concatenation*. In this case, we can state that this concatenation strategy, regardless the specific techniques which are adopted, led to the best results in terms of F1-measure. As a final observation, it should be pointed out that our methodology relies on *pre-trained* embeddings, thus we proved that it is possible to avoid heavyweight training of deep neural networks and to equally obtain accurate recommendations.

Next, in Experiment 2, we compared our best-performing configuration in terms of F1-measure ('Our Best' in Table 4), nDCG and MRR ('Our Best' in Table 5) to several competitive baselines.

Configurations			Movielens 1M - TOP 5			DBbook - TOP 5		
			Prec.	***Recall***	***F1***	***Prec.***	***Recall***	***F1***
BASIC	**Graph Embedding**	*TransE*	0.7722	0.4492	0.5680 *	0.6268	0.5540	0.5881 $^{\Delta}$
		TransH	0.7722	0.4494	0.5682 *	0.6252	0.5514	0.5860 *
	Contexual Word Representations	*BERT*	0.7692	0.4487	0.5668 *	0.6234	0.5510	0.5849 *
		USE	0.7120	0.4248	0.5321 *	0.6050	0.5324	0.5664 *
HYBRID	**Entity-based concatenation**	*TransE + BERT*	0.7744	0.4512	0.5702 *	0.6257	0.5530	0.5872 *
		TransH + BERT	0.7747	0.4511	0.5702 *	0.6274	0.5533	0.5880 *
		TransE + USE	0.7847	0.4544	0.5756	0.6226	0.5500	0.5840 *
		TransH + USE	0.7839	0.4543	0.5752	0.6250	0.5548	0.5878 $^{\Delta}$
	Feature-based concatenation	***TransE + BERT***	0.7805	0.4523	0.5727 *	**0.6304**	**0.5593**	**0.5927**
		TransH + BERT	**0.7857**	**0.4551**	**0.5764**	0.6230	0.5494	0.5839 *
		TransE + USE	0.7776	0.4522	0.5718 *	0.6298	0.5581	0.5918
		TransH + USE	0.7773	0.4516	0.5713 *	0.6287	0.5574	0.5909

Table 2: Results of our proposed framework on ML1M and DBbook dataset by using Precision, Recall and F1 measure. Values marked with * show a difference in score, compared with our best configuration, statistically validated by using the Wilcoxon signed rank test with p-value<0.01. $^{\Delta}$ shows statistical valid differences with p-value< 0.05.

Configurations			Movielens 1M - TOP 5		DBbook - TOP 5	
			nDCG	***MRR***	***nDCG***	***MRR***
BASIC	**Graph Embedding**	*TransE*	0.8205 *	0.8803 *	0.7437 $^{\Delta}$	0.8116
		TransH	0.8207 *	0.8805 *	0.7415 $^{\Delta}$	0.8102
	Contextual Word Representations	*BERT*	0.8183 *	0.8794 *	0.7404 *	0.8106
		USE	0.7572 *	0.8370 *	0.7154 *	0.7879 *
HYBRID	**Entity-based concatenation**	*TransE + BERT*	0.8235 *	0.8825 *	0.7430 $^{\Delta}$	0.8130
		TransH + BERT	0.8243 *	0.8836 *	0.7453	0.8156
		TransE + USE	0.8334	0.8869	0.7406 *	0.8109
		TransH + USE	0.8326	0.8867	0.7422 *	0.8102
	Feature-based concatenation	***TransE + BERT***	0.8286 *	0.8827 *	**0.7485**	**0.8179**
		TransH + BERT	**0.8353**	**0.8900**	0.7394 *	0.8100
		TransE + USE	0.8272 *	0.8845 *	0.7483	0.8150
		TransH + USE	0.8261 *	0.8840 *	0.7476	0.8147

Table 3: Results of our proposed framework on ML1M and DBbook dataset by using nDCG and MRR. Values marked with * show a difference in score, compared with our best configuration, statistically validated by using the Wilcoxon signed rank test with p-value<0.01. $^{\Delta}$ shows statistical valid differences with p-value< 0.05.

Models	Movielens 1M - TOP 5			Diff. F1	DBbook - TOP 5			Diff. F1
	Precision	**Recall**	**F1**		**Precision**	**Recall**	**F1**	
CFGAN	0.7381	0.4291	0.5427 *	-5.85%	0.5752	0.5008	0.5354 *	-9.66%
DeepICF	0.7197	0.4287	0.5373 *	-6.78%	0.597	0.5177	0.5545 *	-6.44%
MLP	0.7188	0.4289	0.5372 *	-6.80%	0.5960	0.5179	0.5542 *	-6.49%
WRMF	0.7145	0.4250	0.5330 *	-7.53%	0.6017	0.5216	0.5588 *	-5.72%
FISM	0.7095	0.4229	0.5299 *	-8.07%	0.5970	0.5177	0.5545 *	-6.44%
NAIS	0.5673	0.3699	0.4478 *	-22.31%	0.5583	0.4861	0.5197 *	-12.32%
DAE	0.6120	0.3889	0.4756 *	-17.49%	0.5723	0.4945	0.5306 *	-10.48%
NeuMF	0.5269	0.3546	0.4239 *	-26.46%	0.5675	0.4937	0.5281 *	-10.90%
item-knn	0.6628	0.3638	0.4697 *	-18.51%	0.5687	0.4786	0.5198 *	-12.30%
user-knn	0.6608	0.3594	0.4656 *	-19.22%	0.5613	0.4716	0.5126 *	-13.51%
att-item-knn	0.5762	0.3391	0.4269 *	-25.94%	0.4585	0.3656	0.4068 *	-31.36%
att-user-knn	0.6587	0.3661	0.4706 *	-18.36%	0.5484	0.4561	0.4980 *	-15.98%
VSM	0.5776	0.3424	0.4299 *	-25.42%	0.4779	0.3878	0.4282 *	-27.75%
kaHFM	0.6046	0.3475	0.4413 *	-23.44%	0.5917	0.5037	0.5442 *	-8.183%
Our Best	**0.7857**	**0.4551**	**0.5764**	—	**0.6304**	**0.5593**	**0.5927**	—

Table 4: Results of our proposed framework on ML1M and DBbook dataset by using Precision, Recall and F1 measure. Values marked with * show a difference in score, compared with our best configuration, statistically validated by using the Wilcoxon signed rank test with p-value<0.01.

Models	Movielens 1M - TOP 5		Diff.	Diff.	DBbook - TOP 5		Diff.	Diff.
	nDCG	MRR	nDCG	MRR	nDCG	MRR	nDCG	MRR
CFGAN	0.7838 *	0.8556 *	-6.17%	-3.87%	0.6815 *	0.7697 *	-8.95%	-5.89%
DeepICF	0.7663 *	0.8450 *	-8.26%	-5.06%	0.7037 *	0.7792 *	-5.98%	-4.73%
MLP	0.7659 *	0.8445 *	-8.31%	-5.11%	0.7042 *	0.7810 *	-5.91%	-4.51%
WRMF	0.7616 *	0.8410 *	-8.82%	-5.51%	0.7107 *	0.7856 *	-5.04%	-3.94%
FISM	0.7565 *	0.8392 *	-9.43%	-5.71%	0.7037 *	0.7792 *	-5.98%	-4.73%
NAIS	0.6098 *	0.7291 *	-27.00%	-18.08%	0.6617 *	0.7552 *	-11.59%	-7.66%
DAE	0.6539 *	0.7578 *	-21.72%	-14.85%	0.6769 *	0.7663 *	-9.57%	-6.31%
NeuMF	0.5683 *	0.6896 *	-31.96%	-22.52%	0.6719 *	0.7629 *	-10.22%	-6.72%
item-knn	0.7252 *	0.8539 *	-13.18%	-4.06%	0.6855 *	0.7922 *	-8.41%	-3.14%
user-knn	0.7199 *	0.8516 *	-13.82%	-4.32%	0.6784 *	0.7890 *	-9.36%	-3.53%
att-item-knn	0.6208 *	0.7571 *	-25.68%	-14.93%	0.5691 *	0.7376 *	-23.97%	-9.81%
att-user-knn	0.7216 *	0.8543 *	-13.61%	-4.01%	0.6657 *	0.7867 *	-11.06%	-3.81%
VSM	0.6281 *	0.7710 *	-24.81%	-13.37%	0.5894 *	0.7478 *	-21.26%	-8.57%
kaHFM	0.6506 *	0.7734 *	-22.11%	-13.10%	0.7038 *	0.7801 *	-5.97%	-4.62%
Our Best	**0.8353**	**0.8900**	—	—	**0.7485**	**0.8179**	—	—

Table 5: Results of our proposed framework on ML1M and DBbook dataset by nDCG and MRR. Values marked with * show a difference in score, compared with all the baselines, statistically validated by using the Wilcoxon signed rank test with p-value<0.01.

As previously stated, we took into account both *deep learning* techniques, *hybrid* approaches and methods exploiting *side information*. In the first case, we aimed to assess the effectiveness of our deep architecture w.r.t. other deep recommender systems. In the latter, we aimed to evaluate the intuition of combining state-of-the-art embedding encoding both *collaborative* and *content-based* information. All these models represent the current state of the art in several recommendation tasks. In this case, it is possible to observe for both the datasets a relevant increase in the F1-measure. In particular, configuration based on TransH and BERT overcomes CFGAN, the best deep learning baselines on ML1M dataset, by 5.85%. Similarly, TransE and BERT overcomes WRMF by 5.72% for the DBbook dataset. About nDCG and MRR, we observed a comparable result. TransH and BERT overcomes CFGAN by 6.17% for nDCG and by 3.87% for MRR on ML1M dataset. TransE and BERT overcomes WRMF by 5.04% for nDCG and by -3.94% for MRR on DBbook dataset. Interesting results are also observed by considering baseline based on hybrid techniques. In particular, the results obtained are in line with those obtained by deep learning baselines and, however, lower than those obtained from our best configurations, both considering the F1 measure and the nDCG and MRR. This definitely confirms the effectiveness of our approach and supports the underlying hypotheses of this work. Also in this case, we decided to validate the obtained results by performing a Wilcoxon signed-rank test. The results of the test showed that the differences observed between our best methods and baselines are statistically significant ($p < 0.01$).

5 CONCLUSIONS AND FUTURE WORK

In this paper we presented a *hybrid* recommendation framework based on the combination of *graph embeddings* and *contextual word representations*. In particular, we evaluated the possibility of combining graph-based (i.e., *TransH*, *TransE*) and contextual word representation (i.e., *BERT*, *USE*) techniques through two concatenation strategies: entity-based and feature-based. In the first one, we followed the intuition that by concatenating more representations of the same entity in the first level of the network, we could help the neural model in better generalizing the intrinsic characteristics of the entity. In the feature-based approach, we instead tried to show that concatenating homogeneous representations of users and items (i.e., created with the same representation technique) could help the network better learn the relationships between these entities. Results showed that hybridization techniques of user and item representations yield excellent results compared to many baselines and compared to the use of a single content representation technique. In particular, the *feature-based concatenation* approach has shown the most promise. Indeed, the configuration using TransH + BERT proved to be the best for the M1M dataset, while TransE + BERT was the winner for the DBbook dataset. These results confirm the validity of the intuition behind the proposed framework. As future work, we will extend the model by also encoding features coming from *knowledge graphs* such as DBpedia or Freebase. Moreover, we will extend our deep architecture by introducing attention mechanisms, and we will evaluate more content representation strategies in order to learn an even more precise representation of users and items.

ACKNOWLEDGMENTS

This work has been supported by Apulia Region, Italy through the project "Un Assistente Dialogante Intelligente per il Monitoraggio Remoto di Pazienti" (Grant n. 10AC8FB6) in the context of "Research for Innovation - REFIN", and by the project "DECiSION" (code: BQS5153) under the programme INNONETWORK - POR Puglia FESR-FSE 2014–2020. We would also like to thank Pierpaolo Ventrella for having contributed to executing some experiments reported in the article.

REFERENCES

[1] Vito Walter Anelli, Alejandro Bellogín, Antonio Ferrara, Daniele Malitesta, Felice Antonio Merra, Claudio Pomo, Francesco M. Donini, and Tommaso Di Noia. 2021. Elliot: a Comprehensive and Rigorous Framework for Reproducible Recommender Systems Evaluation. *CoRR* abs/2103.02590 (2021).

[2] Vito Walter Anelli, Tommaso Di Noia, Eugenio Di Sciascio, Azzurra Ragone, and Joseph Trotta. 2019. How to make latent factors interpretable by feeding

factorization machines with knowledge graphs. In *International Semantic Web Conference*. Springer, 38–56.
[3] 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*. 333–336.
[4] Iz Beltagy, Kyle Lo, and Arman Cohan. 2019. SciBERT: A pretrained language model for scientific text. *arXiv preprint arXiv:1903.10676* (2019).
[5] Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Advances in neural information processing systems* 26 (2013), 2787–2795.
[6] Robin Burke. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction* 12, 4 (2002), 331–370.
[7] Hongyun Cai, Vincent W Zheng, and Kevin Chen-Chuan Chang. 2018. A comprehensive survey of graph embedding: Problems, techniques, and applications. *IEEE Transactions on Knowledge and Data Engineering* 30, 9 (2018), 1616–1637.
[8] Gjorgjina Cenikj and Sonja Gievska. 2020. Boosting Recommender Systems with Advanced Embedding Models. In *Companion Proceedings of the Web Conference 2020* (Taipei, Taiwan) *(WWW '20)*. Association for Computing Machinery, New York, NY, USA, 385–389. https://doi.org/10.1145/3366424.3383300
[9] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018).
[10] Dong-Kyu Chae, Jin-Soo Kang, Sang-Wook Kim, and Jung-Tae Lee. 2018. Cfgan: A generic collaborative filtering framework based on generative adversarial networks. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 137–146.
[11] 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.
[12] Félix Hernández Del Olmo and Elena Gaudioso. 2008. Evaluation of recommender systems: A new approach. *Expert Systems with Applications* 35, 3 (2008), 790–804.
[13] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. (2019).
[14] Kawin Ethayarajh. 2019. How Contextual are Contextualized Word Representations? Comparing the Geometry of BERT, ELMo, and GPT-2 Embeddings. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, Hong Kong, China, 55–65. https://doi.org/10.18653/v1/D19-1006
[15] Zeno Gantner, Steffen Rendle, Christoph Freudenthaler, and Lars Schmidt-Thieme. 2011. MyMediaLite: A free recommender system library. In *Proceedings of the fifth ACM conference on Recommender systems*. 305–308.
[16] Palash Goyal and Emilio Ferrara. 2018. Graph embedding techniques, applications, and performance: A survey. *Knowledge-Based Systems* 151 (2018), 78–94.
[17] László Grad-Gyenge, Attila Kiss, and Peter Filzmoser. 2017. Graph embedding based recommendation techniques on the knowledge graph. In *Adjunct Publication of the 25th Conference on User Modeling, Adaptation and Personalization*. 354–359.
[18] J Grau. 2009. Personalized product recommendations: Predicting shoppers' needs.
[19] Hebatallah A Mohamed Hassan, Giuseppe Sansonetti, Fabio Gasparetti, Alessandro Micarelli, and Joeran Beel. 2019. BERT, ELMo, USE and InferSent Sentence Encoders: The Panacea for Research-Paper Recommendation?. In *RecSys (Late-Breaking Results)*. 6–10.
[20] Xiangnan He, Zhankui He, Jingkuan Song, Zhenguang Liu, Yu-Gang Jiang, and Tat-Seng Chua. 2018. Nais: Neural attentive item similarity model for recommendation. *IEEE Transactions on Knowledge and Data Engineering* 30, 12 (2018), 2354–2366.
[21] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural Collaborative Filtering. In *Proceedings of the 26th International Conference on World Wide Web* (Perth, Australia) *(WWW '17)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, CHE, 173–182. https://doi.org/10.1145/3038912.3052569
[22] Yifan Hu, Yehuda Koren, and Chris Volinsky. 2008. Collaborative filtering for implicit feedback datasets. In *2008 Eighth IEEE International Conference on Data Mining*. Ieee, 263–272.
[23] Dietmar Jannach, Markus Zanker, Alexander Felfernig, and Gerhard Friedrich. 2010. *Recommender systems: an introduction*. Cambridge University Press.
[24] Kalervo Järvelin and Jaana Kekäläinen. 2017. IR evaluation methods for retrieving highly relevant documents. In *ACM SIGIR Forum*, Vol. 51. ACM New York, NY, USA, 243–250.
[25] Santosh Kabbur, Xia Ning, and George Karypis. 2013. Fism: factored item similarity models for top-n recommender systems. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*. 659–667.
[26] Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. *Advances in neural information processing systems* 28 (2015), 3294–3302.
[27] Yehuda Koren and Robert Bell. 2015. Advances in collaborative filtering. In *Recommender systems handbook*. Springer, 77–118.
[28] Xiaofei Ma, Zhiguo Wang, Patrick Ng, Ramesh Nallapati, and Bing Xiang. 2019. Universal text representation from BERT: an empirical study. *arXiv preprint arXiv:1910.07973* (2019).
[29] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[30] Cataldo Musto, Pierpaolo Basile, and Giovanni Semeraro. 2019. Hybrid Semantics-Aware Recommendations Exploiting Knowledge Graph Embeddings. In *International Conference of the Italian Association for Artificial Intelligence*. Springer, 87–100.
[31] Cataldo Musto, Tiziano Franza, Giovanni Semeraro, Marco de Gemmis, and Pasquale Lops. 2018. Deep content-based recommender systems exploiting recurrent neural networks and linked open data. In *Adjunct Publication of the 26th Conference on User Modeling, Adaptation and Personalization*. 239–244.
[32] Makbule Gulcin Ozsoy. 2016. From word embeddings to item recommendation. *arXiv preprint arXiv:1601.01356* (2016).
[33] Enrico Palumbo, Giuseppe Rizzo, Raphaël Troncy, Elena Baralis, Michele Osella, and Enrico Ferro. 2018. Translational models for item recommendation. In *European Semantic Web Conference*. Springer, 478–490.
[34] Marco Polignano, Pierpaolo Basile, Marco de Gemmis, and Giovanni Semeraro. 2019. Social tags and emotions as main features for the next song to play in automatic playlist continuation. In *Adjunct Publication of the 27th Conference on User Modeling, Adaptation and Personalization*. 235–239.
[35] Marco Polignano, Pierpaolo Basile, Gaetano Rossiello, Marco de Gemmis, and Giovanni Semeraro. 2017. Learning inclination to empathy from social media footprints. In *Proceedings of the 25th Conference on User Modeling, Adaptation and Personalization*. 383–384.
[36] Marco Polignano, Fedelucio Narducci, Marco de Gemmis, and Giovanni Semeraro. 2021. Towards Emotion-aware Recommender Systems: an Affective Coherence Model based on Emotion-driven Behaviors. *Expert Systems with Applications* 170 (2021), 114382.
[37] Paul Resnick and Hal R Varian. 1997. Recommender systems. *Commun. ACM* 40, 3 (1997), 56–58.
[38] Francesco Ricci, Lior Rokach, and Bracha Shapira. 2011. Introduction to recommender systems handbook. In *Recommender systems handbook*. Springer, 1–35.
[39] Giovanni Semeraro, Pasquale Lops, and Marco Degemmis. 2005. WordNet-based user profiles for neighborhood formation in hybrid recommender systems. In *Fifth International Conference on Hybrid Intelligent Systems (HIS'05)*. IEEE, 6–pp.
[40] Noah A Smith. 2020. Contextual word representations: putting words into computers. *Commun. ACM* 63, 6 (2020), 66–74.
[41] Nitish Srivastava and Ruslan Salakhutdinov. 2014. Multimodal learning with deep boltzmann machines. *The Journal of Machine Learning Research* 15, 1 (2014), 2949–2980.
[42] P. Stakhiyevich and Z. Huang. 2019. Building user Profiles Based on user Interests and Preferences for Recommender Systems. In *2019 IEEE International Conferences on Ubiquitous Computing Communications (IUCC) and Data Science and Computational Intelligence (DSCI) and Smart Computing, Networking and Services (SmartCNS)*. 450–455. https://doi.org/10.1109/IUCC/DSCI/SmartCNS.2019.00101
[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*. 5998–6008.
[44] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2018. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461* (2018).
[45] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. 2019. Glue: A multi-task benchmark and analysis platform for natural language understanding. In *7th International Conference on Learning Representations, ICLR 2019*.
[46] Hongwei Wang, Fuzheng Zhang, Xing Xie, and Minyi Guo. 2018. DKN: Deep knowledge-aware network for news recommendation. In *Proceedings of the 2018 world wide web conference*. 1835–1844.
[47] Hongwei Wang, Fuzheng Zhang, Mengdi Zhang, Jure Leskovec, Miao Zhao, Wenjie Li, and Zhongyuan Wang. 2019. Knowledge-aware graph neural networks with label smoothness regularization for recommender systems. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 968–977.
[48] Yining Wang, Liwei Wang, Yuanzhi Li, Di He, Wei Chen, and Tie-Yan Liu. 2013. A theoretical analysis of NDCG ranking measures. In *Proceedings of the 26th annual conference on learning theory (COLT 2013)*, Vol. 8. 6.
[49] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 28.
[50] 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*. 153–162.

[51] Feng Xue, Xiangnan He, Xiang Wang, Jiandong Xu, Kai Liu, and Richang Hong. 2019. Deep item-based collaborative filtering for top-n recommendation. *ACM Transactions on Information Systems (TOIS)* 37, 3 (2019), 1–25.

[52] Fuzheng Zhang, Nicholas Jing Yuan, Defu Lian, Xing Xie, and Wei-Ying Ma. 2016. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 353–362.

[53] Yuhang Zhang, Jun Wang, and Jie Luo. 2020. Knowledge Graph Embedding Based Collaborative Filtering. *IEEE Access* 8 (2020), 134553–134562.