Learning to Recommend Items to Wikidata Editors

Kholoud AlGhamdi(✉), Miaojing Shi, and Elena Simperl

King's College London, London, UK
{kholoud.alghamdi,miaojing.shi,elena.simperl}@kcl.ac.uk

Abstract. Wikidata is an open knowledge graph built by a global community of volunteers. As it advances in scale, it faces substantial challenges around editor engagement. These challenges are in terms of both attracting new editors to keep up with the sheer amount of work, and retaining existing editors. Experience from other online communities and peer-production systems, including Wikipedia, suggests that personalised recommendations could help, especially newcomers, who are sometimes unsure about how to contribute best to an ongoing effort. For this reason, we propose a recommender system *WikidataRec* for Wikidata items. The system uses a hybrid of content-based and collaborative filtering techniques to rank items for editors relying on both item features and item-editor previous interaction. A neural network, named neural mixture of representations, is designed to learn fine weights for the combination of item-based representations and optimize them with editor-based representation by item-editor interaction. To facilitate further research in this space, we also create two benchmark datasets, a general-purpose one with 220, 000 editors responsible for 14 million interactions with 4 million items, and a second one focusing on the contributions of more than 8, 000 more active editors. We perform an offline evaluation of the system on both datasets with promising results. Our code and datasets are available at https://github.com/WikidataRec-developer/Wikidata_Recommender.

1 Introduction

Wikidata is an open knowledge graph built by a global community of volunteers [40]. Since its launch in 2012 it reached more than 90 million items and 24, 000 active editors[1]. Manual contributions are core to Wikidata [40]: editors add new content, keep it up-to-date, model knowledge as graph items and properties, and decide on all rules of content creation and management. However, as Wikidata advances in scale, it faces substantial challenges around editor engagement [34].

Experience from other online communities and peer-production systems, including Wikipedia, Quora and others, suggests that one way to improve engagement is through *recommendations* [7,9,44]. This is especially true for editors who are relatively new to Wikidata, who need to overcome so-called 'legitimate peripheral participation' (LPP) effects to continue to engage [28]. According to LPP,

[1] www.wikidata.org/wiki/Wikidata:Statistics/en.

A. Hotho et al. (Eds.): ISWC 2021, LNCS 12922, pp. 163–181, 2021.
https://doi.org/10.1007/978-3-030-88361-4_10

newcomers are more likely to become members of a community if they are provided with suggestions of (typically lower-risk) tasks they could carry out to further the goals of the community [19].

At the moment, looking for relevant items to edit in Wikidata remains challenging because of the sheer number of options available [34]. The suggested and open tasks page[2] gives a useful but daunting overview of the various ways in which people could contribute to Wikidata. Many task lists are automatically generated, without taking into account aspects such as editor tenure, previous editing history or interests. This lack of focus is also said to prevent editors from developing reinforced editing habits, which increases the likelihood of dropouts [34]. Seasoned editors use tools such as *QuickStatements*[3] and *Watchlist*[4] to organise their work, but such tools become relevant only once editors have identified productive ways to contribute.

For these reasons, we propose *WikidataRec*, a recommender system for Wikidata items. The system uses content-based and collaborative filtering techniques to rank items for editors relying on both item features and item-editor previous interaction. Collaborative filtering is a representative approach to address recommendation task with implicit feedback, which learns item-centric and editor-centric edit representations using matrix factorization [17] from item-editor interaction data. We adopt it in this work and further facilitate it with additional information from item content and relations, where we learn these representations from by sentence embedding model ELMo [26] and graph embedding model TransR [20], respectively. To combine the multiple representations, we introduce the neural mixture of representations (NMoR), a neural network inspired by mixture of experts [13] that produces fine weights for different item-based representations. It is optimized over the editor-based representation to rank items to editors. The proposed *WikidataRec* demonstrates a large capacity of leveraging items' contents, relations, and edit history between editors and items into the recommender.

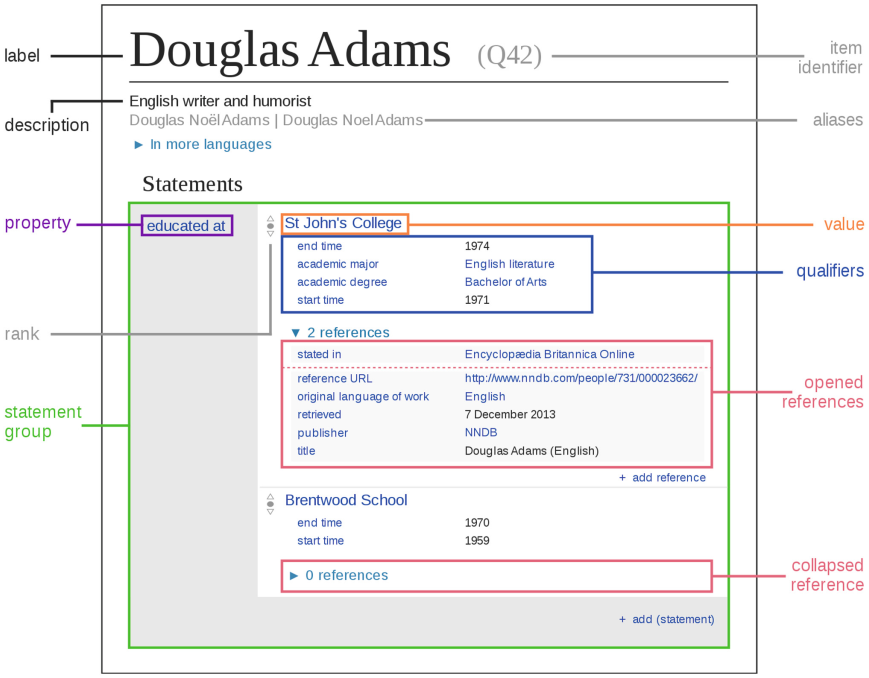
To facilitate further research in this space, we also create two benchmark datasets, a general-purpose one with $220,000$ editors responsible for 14 million interactions with 4 million items, and a more focused one with contributions of more than $8,000$ active editors with more than 200 edits each. We evaluate the system on these datasets against several baselines and analyse the impact of different features and levels of data sparsity on performance. The results are promising, though challenges remain because of unbalanced participation, sparse interaction data and little explicit information about editors' interests and feedback.

2 Background and Related Work

2.1 Wikidata Data Model

Wikidata consists of structured records stored in form of entities, where each entity is allocated a separate page and described by a set of terms and statements [40].

[2] www.wikidata.org/wiki/Wikidata:Contribute/Suggested_and_open_tasks.
[3] https://quickstatements.toolforge.org/#/.
[4] www.wikidata.org/wiki/Help:Watchlist.



Douglas Adams (Q42)

English writer and humorist

Douglas Noël Adams | Douglas Noel Adams

► In more languages

Statements

educated at

St John's College

end time	1974
academic major	English literature
academic degree	Bachelor of Arts
start time	1971

▼ 2 references

stated in	Encyclopædia Britannica Online
reference URL	http://www.nndb.com/people/731/000023662/
original language of work	English
retrieved	7 December 2013
publisher	NNDB
title	Douglas Adams (English)

+ add reference

Brentwood School

end time	1970
start time	1959

► 0 references

+ add (statement)

Fig. 1. The structure of an Wikidata item. Source: [42]

There are two types of entities, items and properties, situated in different namespaces. The namespaces are identified by URIs using numbers and letters, with the letter Q relating to items and the letter P to properties. Although there are other namespaces in Wikidata, we focus only on these two, as they are most relevant for our recommender system. Items and property pages have a similar interface, starting with labels, descriptions and aliases, which are language-specific, and then moving on to statements, which are language-agnostic. Each statement contains a set of triplet edges (head, relation, tail), capturing the different types of relations between entities in the world, e.g., London $\xrightarrow{\text{capital of}}$ United Kingdom. Pages also include sitelinks connecting to Wikipedia articles or other Wikimedia projects [24]. Each page is indexed by a unique identifier. Figure 1 shows an example of a Wikidata item page.

2.2 Editors and Editing in Wikidata

Anyone can edit Wikidata. The literature tends to distinguish between two types of contributions: manual ones, carried out by registered or anonymous editors, and bots. Bots are used for repetitive editing tasks, but do not contribute to discussions, modeling decisions, or rules for content creation, and management [24]. Human editors may get involved in any type of activities, though some require specific access rights or ontology engineering skills [28,29]. Furthermore, there are two types of editing tasks which are higher-risk tasks and lower-risk ones. Edits are higher-risk if they affect a larger share of the graph, e.g. property editing; hence not all editors may create properties in Wikidata. Lower-risk edits are, for example adding/changing labels, descriptions, etc. Formally, the Wikidata editing process starts when an editor logs into the platform and contributes by inserting data on an item page using the basic editing interface (See Fig. 1). Every action performed by an editor is recorded on the so-called revision history page. In our current work, we focus on item edits rather than properties or other types of contributions. Our aim is to establish whether recommendations are technically feasible with the available data. Previous studies into the Wikidata community suggest that item edits are popular with less experienced editors, who are the main audience for personalised recommendations. Properties or other sorts of works are normally dealt by seasoned and active editors [28,29].

2.3 Recommending Tasks to Communities and Crowds Online

There are a lot of works proposing the use of personalized recommendations in online communities and peer-production systems [7,9,44]. Recommendations aim to make work more effective and increase retention, by matching open tasks to people's skills and interests [5]. As noted in the introduction, they can also help new members of a community find their way and contribute [9,28].

In Wikipedia, a first recommender system, SuggestBot, was proposed in [5]. It used article titles, links, and co-editing patterns as main features to recommend articles to editors. A more recent recommender system by [23] represents Wikipedia articles using Graph Convolutional Networks. Both works aim to recommend items (Wikipedia articles) to editors based on features of items and editors. There is no explicit feedback that would confirm an editor's interest in an article. Equally, there is very little information about editors beyond what they have edited so far [23]. Our system is similar in spirit to [5] and [23], though our neural approach incorporates additional item-based features and a mix of representations.

In community question-answering (CQA), several papers develop recommender systems to route users to questions they might be interested in answering, hence improving their engagement on the platform and reducing question answering time [7,21,37,38]. [7,37] model recommendation as a classification problem, using different machine learning techniques. They implement a hybrid approach with content and collaborative knowledge to address the sparsity problem. [21,38] apply graph embedding techniques to tackle the same problem. Similarly, we mix item and relational representations with collaborative filtering to solve in a novel application context.

Another related area is online crowdsourcing, where the aim is to allocate tasks published by a requester to an online crowd. [18,33] employ a probabilistic matrix factorization (PMF) to recommend suitable tasks to crowdworkers based on their previous activities, performance, and preferences. They handle the cold start problem by utilizing predefined categories (e.g., sentiment analysis, translation, image labeling) as additional features to improve the recommendation accuracy.

2.4 Evaluating Recommender Systems

Evaluation Methodologies. To evaluate the performance of a recommender system, *precision@k* and *recall@k* are the most common metrics for top-k recommendations task with implicit data [36]. *Precision@k* measures the relevance of the recommended items list, whereas *recall@k* gives insight about how well the recommender is able to recall all the items the editor has rated positively (or interacted with) in the test set [36]. Moreover, there is a set of metrics that cares about where the relevant item appears in the recommended list. For instance, *mean average precision(MAP)* and *mean average recall (MAR)* are popular metric for search engines and are applied to the recommendation task [2]. In these metrics, the relevant items are required to be placed as high on the recommendation list as possible [36]. There is also a family of metrics such as *catalog coverage* and *diversity* that pays special attention to editor experience [2]. The *catalog coverage* evaluates whether the recommender algorithm can generate recommendations with a high proportion of items, and the *diversity* evaluates how diverse the set of proposed recommendations within a single recommended list [2].

Recommender Datasets. Recommender datasets are available for items such as movies (using MovieLens)[5] or products (using the Amazon dataset)[6]. In other tasks, such benchmarks are not widely available. The systems discussed in Sect. 2.3 are mostly evaluated on custom-built datasets derived from the platforms' activity logs. This is also the case with our system, which to the best of our knowledge is the first of its kind for Wikidata. To encourage further research in this space, we hence provide two new datasets for the community to reuse, specified in Sect. 3.

3 Wikidata Recommender Datasets

3.1 From Wikidata Dumps to Relational Tables

Wikidata dumps are readily available as JSON, RDF and XML[7]. In our work we use the JSON and XML ones from July 1, 2019. The JSON dump contains all Wikidata pages without their edit history, which is available as XML. We parse the JSON data to extract all Wikidata items along with their corresponding identifiers, labels, descriptions, and statements. For the text data (i.e. label and description), we fetch only the English language version and discard data for other languages, because English is the best covered language in Wikidata [15]. Also, we ignore items' metadata about aliases and sitelinks as we do not use them as features in our system. We transform the parsed data into CSV and import it into a PostgreSQL database (as a *Wikidata-items-content* table); further processed the raw data containing the items' statements (i.e. claims) in

[5] https://grouplens.org/datasets/movielens/.
[6] https://jmcauley.ucsd.edu/data/amazon/.
[7] https://dumps.wikimedia.org/wikidatawiki/.

similar way (as a *Wikidata-items-relations* PostgreSQL table). Next, we work on the XML dump to extract editing information for each individual editor. We only focus on edits that are performed on an item's namespace and ignore all other non-item-related actions, as we do not model this information in our system. For each edit, the following information is kept: who carried out the edit, the item being edited, the timestamps of the edit, and the comment indicating the specific action executed by the editor on the item. They are stored in PostgreSQL (as a *Wikidata-editors-edits* table).

3.2 Sampling and Cleaning

We randomly sample 14 million editing activities performed by $221,353$ editors who are human registered (i.e. non-bots). We sort these activities in ascending order based on timestamps. We refer to this as the *Wikidata-14M* dataset. We also want to test the system on a denser dataset that contains the editors who interacted with a larger number of items in their editing history and the items that received edits by a high number of editors. The *Wikidata-14M* dataset is filtered by keeping editors who edited at least 200 unique items during their tenure and items that have been edited by at least 5 different editors. We refer to it as the *Wikidata-Active-Editors* dataset. We then removed some outliers from both datasets - 2.14% of editors in *Wikidata-14M* and 5% in *Wikidata-Active-Editors* edited a very large number of items in a very short time relative to the size of their contributions. This is considered as the case for institutional accounts that publish their data via Wikidata [39]. We remove those accounts and their edits from the datasets, as they are not the main target editors for recommendations. We use both datasets to evaluate our system. The first one *Wikidata-14M* depicts the actual diversity of editing activities in Wikidata with a mix of active and more casual editors. The second one *Wikidata-Active-Editors* focuses on more active editors and items - we use it to understand the effects of data sparsity on recommender performance.

Table 1. Number of items, editors and interactions in the two Datasets.

Dataset	# editors	# items	# interactions
Wikidata-14M	221,353	4,881,720	14,045,523
Wikidata-Active-Editors	8,024	381,784	3,272,086

Table 2. Statistics of the distribution of editing activities among editors and items.

	Wikidata-14M			Wikidata-Active-Editors		
	Median	Mean	Std	Median	Mean	Std
# items/editor	1	72	534	900	1,244	2,197
# edits/editor	2	143	1,413	1,045	4,666	32,338
# editors/item	1	2	3.7	-	-	-

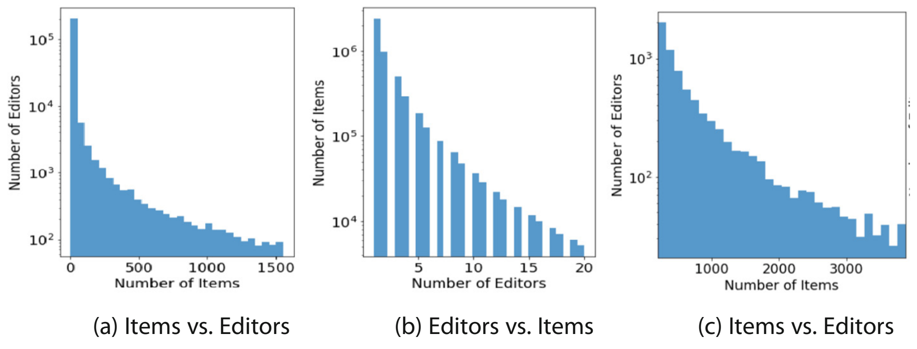


Fig. 2. Editing Activities Distribution in *Wikidata-14M* (a and b) and in *Wikidata-Active-Editors* (c).

3.3 Datasets Description

Table 1 reports some key descriptive statistics of the two datasets, *i.e.* number of editors, items and interactions. Figure 2a, 2b and 2c characterise the distribution of edits in the two datasets. For *Wikidata-14M*, Fig. 2a shows the number of items edited by each editor - most editors edited only a few items ($< 1,000$); while fewer than 100 editors (out of $221,353$) edited more than $1,500$ items. Table 2 illustrates this skewed distribution: the median values are much smaller than the mean values, both per editor and per item. It is in line with observations from previous studies in Wikidata and other large-scale community-driven knowledge engineering initiatives [16,29,34], which attest that a small core of contributors are responsible for a majority of the work. This also gives us an opportunity to increase retention in the long tail of editors (novice editors mostly) through recommendations. We also examine items that are edited by multiple editors, which is useful in collaborative filtering: Fig. 2b shows that most items have been edited only by small number of editors; the highest number of editors per item is 20. This implies the issue of cold-start for new items, we address it by mining item content and relation information for recommendation (see Sect. 4).

We observe similar, though less pronounced effects for *Wikidata-Active-Editors* (see Fig. 2c and Table 2): the interaction data between editors and items are denser, e.g. the mean and median of the number of items per editor is $1,244$ and 900, respectively, showing a less skewed distribution. Notice we do not have the corresponding Fig. 2b (Editors vs. Items) for *Wikidata-Active-Editors*, because all active editors are filtered and selected, making this statistic no longer meaningful.

4 Wikidata Recommender System

4.1 Problem Statement

Our problem is defined as follows: given a set of N editors, a set of M items, and an interaction matrix $A^{N\times M} = \{a_{ij}\}$, where each matrix entry a_{ij} is either 1 or

0 indicating whether editor i has edited item j. The task is learning to estimate the preference scores of editors to items so as to rank and recommend new items to editors. Solving this problem is not straightforward: as discussed in Sect. 3, there exists a high number of items but a low number of interactions between items and editors; many items have only a few edits. Standard approaches using collaborative filtering [35] rely primarily on the item-editor interaction data and do not perform well in our scenario. Intuitively, there are two ways to improve it: by including more information from either editors (e.g. their interests or activities) or items (e.g. their content or relations). In this paper we focus on the latter, as for the former, there is very little descriptive information available for Wikidata editors, a known issue also present in Wikipedia [5,23]. Of course one could try to collect additional information about editors from their activities in Wikidata discussions or contributions to other WikiProjects. This however is not the scope of this paper as we decide to steer away from it to avoid the potential ethical implications.

4.2 Approach

We introduce a Wikidata recommender system *WikidataRec* which is a hybrid model combining item content and relation information with collaborative filtering [17] by means of mixture of experts (MoE) [13] (Fig. 3). The matrix factorization (MF) decomposes the item-editor interaction matrix A into editor-centric and item-centric edit representations, denoted by e_i and v_j for editor i and item j respectively. Item content representation, denoted by c_j, is obtained by the sentence embedding model ELMo [26]; and item relation representation, denoted by r_j, is obtained by the graph embedding model TransR [20] building upon the Wikidata knowledge graph. To combine multiple item-based representations, we introduce a neural mixture of representations (NMoR) inspired by MoE, in which we utilize a soft-gating to assign different weights to each representation. More specifically, the network takes four inputs e_i, v_j, c_j and r_j, where v_j, c_j and r_j are added with weights (w_v, w_c, w_r) produced from a soft-gating branch whose input is the concatenation of v_j, c_j and r_j. The merged item representation is fed into an element-wise dot product layer together with e_i to predict the preference score x_{ij}. At the training stage, x_{ij} is optimized with the cross-entropy loss over the ground truth label in A; while at testing, it is used to recommend items to the editors. The whole process can be written as,

$$x_{ij} = e_i(w_v \cdot v_j + w_c \cdot c_j + w_r \cdot r_j)^\intercal. \tag{1}$$

Notice item and editor representations are used in the network as fixed representations (i.e., they are not updated during the training process). They are learned in advance by MF, ELMo, and TransR, respectively. We decide not to jointly tune these representations for efficiency and simplicity reasons, as it is not lightweight to integrate all the models in a whole. We will work on joint learning in our future work.

Below we first specify the generation of e_i, v_j, c_j and r_j and then introduce the neural mixture of representations for w_v, w_c and w_r.

4.3 Editor-Centric and Item-Centric Edit Representations

Editing activities are summarized in the interaction matrix A. Inspired by [17], we use matrix factorization to decompose A into the editor's latent representation matrix $E^{N \times Z}$ and the item's latent representation matrix $V^{M \times Z}$. This is achieved by approximating the target matrix A via the matrix product of two low-rank matrices E and V:

$$\widehat{A} = EV^{\intercal} \tag{2}$$

Each row e_i in E can be seen as a latent representation of editor i in terms of its edits. Similarly, each row v_j of V describes an item j with respect to its edits by different editors. Thus, the prediction formula from Eq. (2) can also be written as:

$$\widehat{a_{ij}} = e_i v_j^{\intercal} \tag{3}$$

In order to single out E, V from A, we utilize Bayesian Personalized Ranking (BPR) [32] to optimize $\widehat{A}$ over it. If we look at Eq. (1), $\widehat{a_{ij}}(e_i v_j^{\intercal})$ is its first term.

4.4 Item Content Representation

To generate item content representation c_j, we need an embedding model that can learn both semantic and syntactic representations from text. There are various embedding models that learn item representations from words, sentences, or paragraphs. Among them, word embedding models such as Word2Vec [22], GloVe [25] and FastText [14] are widely used for many recommendation tasks [8], but they have limitations: the order of words is ignored, which leads to the loss

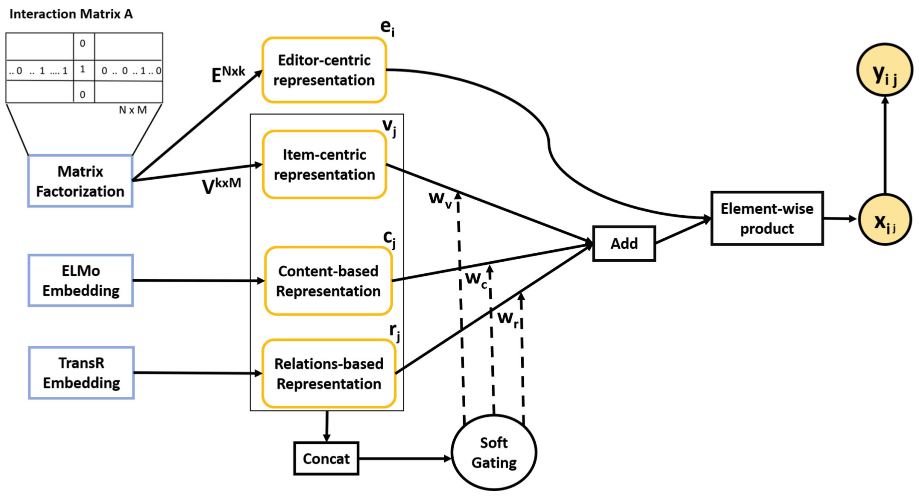


Fig. 3. The architecture of *WikidataRec*.

of the syntactic and semantic meaning in sentences [41]. We therefore considered two state-of-the-art sentence embedding models instead, specifically Embeddings from Language Models (ELMo) and Sentence-BERT. ELMo [26] employs a bi-directional deep LSTM network which takes an entire sentence as input, assigns representation to each word in the sentence, then takes the average of the vectors of words to output a Z-dimensional vector for the input sentence. Sentence-BERT [31] relies on a bi-directional Transformer network [31] which learns from fix-sized semantically meaningful sentences. We preferred ELMo to Sentence-BERT to generate c_j because of the high computational costs of the latter, particularly on large corpora [30]. Each item in Wikidata has a very short description which naturally serves as input sentence to ELMo. The implementation details are in Sect. 5.1.

4.5 Item Relational Representation

Item relational reprentations are built upon the Wikidata graph over items. We model item relations as a directed labeled graph, where it has a set of nodes to represent items, a set of edges (unweighted) to represent relations between items, and labels to capture the type of relations (see Sect. 2.1). Over different graph embedding models [3,20,43], we adopt the TransR [20], a representative approach for heterogeneous graphs, to learn low-dimensional (Z in this work) relational embedding of items, r_j. It builds entity and relation embeddings by regarding a relation as a translation from head entity to tail entity [20]. Different from other graph embedding methods [3,43] which normally assume the embedding of entities and relations within the same space, TransR represents entities and relations in distinct spaces and projects entities from entity space to relation space via a projection matrix. For this reason, it is selected to model the heterogeneity of items and their relations in the Wikidata graph. Items with similar relations have similar embedding in TransR. The implementation details are in Sect. 5.1.

4.6 Neural Mixture of Representations

Having e_i, v_j, c_j, and r_j ready, we present our neural mixture of representations (NMoR) to combine them for Wikidata recommendation. e_i, v_j, c_j, and r_j are fed into NMoR in parallel as fixed representations, where they are not updated during training but learned in advance (as noted in Sect. 4.2). Item representations v_j, c_j and r_j are added with weights w_v, w_c and w_r generated by an additional soft-gating branch (see Fig. 3). Soft-gating is used to distinguish from hard-gating in MoE. In hard-gating, weights are either 0 or 1 for each expert. Soft-gating can be seen as probabilities combining different experts. The motivation for assigning different weights to different item-based representations is to control their respective contribution to the final prediction x_{ij} (Eq. 1). These weights are also Z-dimensional vectors as with v_j, c_j, and r_j such that they are tailored to every dimension of feature representations. The soft-gating branch takes the concatenation of v_j, c_j and r_j of size $1 \times \mathrm{Z} \times 3$ (corresponding to rows, columns, and channels of the tensor) as input; three 1D convolutional layers (kernel size 1 × 1, filters 1024) followed by ReLu are applied to process the tensor along its channels; the number of their output channel remains 3. The output of the last convolutional layer is thus of size 1 × Z × 3, where each channel corresponds to the weight for certain item-based representation (w_v, w_c or w_r). At every column of the tensor, it is softmaxed over channels such that the three values at each column are summed to 1. The merged item representation $w_v \cdot v_j + w_v \cdot c_j + w_r \cdot r_j$ is passed through the element-wise dot product layer with e_i, their similarity value x_{ij} is hence computed. To optimize x_{ij}, we adopt the binary cross-entropy loss,

$$L = -(y_{ij} \log(\sigma(x_{ij})) + (1 - y_{ij}) \log(1 - \sigma(x_{ij}))), \tag{4}$$

where x_{ij} is transformed into a probability using the sigmoid activation function $\sigma(\cdot)$; y_{ij} is the ground truth label of either 0 or 1 in A indicating whether interaction between the item j and editor i has been observed (*i.e.* positive instances) or not (*i.e.* negative instances). Since A is sparse, we randomly select negative instances to make its number the same to that of positive instances. x_{ij} indicates how likely item j is relevant to editor i, we can use it to recommend items to editors at testing.

Traditional weight tuning approach like line search is only suitable for optimizing a limited number of weights, for instance, one parameter for each item-based representation. The proposed NMoR optimizes weights in a neural network which enables a refined weight steering on every dimension of the item-based representation.

5 Experiments

5.1 Experimental Setup and Implementation Details

We run experiments on the two datasets *Wikidata-14M* and *Wikidata-Active-Editors* introduced in Sect. 3. Both datasets are structured in an editor-item-edits format. For each dataset, similar to [11,12], we follow the hold-out strategy to select 80% items associated with each editor to constitute the training set and use the remaining 20% as the test set. Within the training set, we set aside 10% as validation data for hyper-parameter tuning. In *Wikidata-14M* dataset, 25% of the editors edited only between 2 to 6 different items during their editing tenure on the selected dates; these are cold-start editors. Therefore, we exclude these editors from the training set and include them only in the test set. We do this because the editing data for this type of editor are rather too sparse to be informative during training. However, we use this data in the test set, as we want to evaluate the performance of *WikidataRec* on cold-start editors. Z for feature representations is set to 1024.

NMoR: To learn NMoR, the batch size is set to 32. The model is learned with Adam optimizer with a learning rate of 0.001 for 100 epochs.

ELMo: To generate the item content representations, we start from the *Wikidata-items-content* table in our datasets, and then follow the following steps using Python's Spacy: 1) tokenize and normalize the content (label and description) of each item; 2) remove stopwords, punctuation and single-letter words; 3) use part-of-speech tagging and retained only nouns and adjectives. We pass the resulting corpus as input to the ELMo model. ELMo is pre-trained on the 1 Billion Word Benchmark that contains about $800M$ tokens of general-purpose data from WMT 2011 [4]. This is very comprehensive such that fine-tuning on Wikidata is no longer necessary. Following [10], we simply forward each Wikidata item descriptor to ELMo to obtain the item-content representation. Notice ELMo produces multi-scale output which we average them.

TransR: To generate the item relational representations, we first build the labeled directed graph $G(V, E, P)$ from the *Wikidata-items-relations* (Sect. 3.1), where V signifies Wikidata items (the head of the triplet), E inter-item relations (the tail of the triplet), and P relations' types (the relation of the triplet). We use the graph as input to train the TransR model. The original triplets in the graph are the positive instances. We sample a few of them to replace either their heads or tails with wrong components to create negative triplets. TransR is trained with both positive and negative triplets using binary cross-entropy loss. It is optimized with the SGD optimizer for 10 epochs; the batch size is 128 and learning rate is 0.01.

5.2 Evaluation Protocols

For evaluation, in order to simulate the practical recommendation scenario in Wikidata, we follow the spirit in [23] to carry out the test: for each editor i and the item-editor interaction matrix A in the test set, we 1) split the items of editor i into half-half where the editor-centric edit representation e_i along with the representations of item-centric edit, content, and relations v_j, c_j and r_j are obtained by excluding the second-half items in A in our model; 2) exclude the editor i from A to generate the representations of item-centric edits, item-content and item relations for the second half of items in our model; 3) randomly select 50 negative items that were not edited by editor i, where we can obtain their corresponding sub-matrix of A and obtain their item-based representations; 4) feed (e_i, v_j, c_j and r_j) obtained from step 1 to our NMoR along with the item-based representations obtained from steps 2 and 3 to obtain their ranking scores.

The evaluation criteria is to measure how well *WikidataRec* ranks the correct items against the negative items for a given editor. Therefore, we employ *Precision@k*, *Recall@k*, and *MAR@k* (see Sect. 2.4). We are also interested in the diversity of the recommended items, as editing a different set of items is a noted behavior in Wikidata community [27,34]. Having a diverse recommendation would allow the editors to discover a wide range of items for the editing.

Table 3. Precision@k and Recall@k results comparison between our model and state-of-the-art for *Wikidata-14M*.

	Precision @k		Recall @k		
	5	10	50	100	200
GMF	0.096	0.071	0.135	0.183	0.274
BPR-MF	0.050	0.043	0.093	0.135	0.190
eALS	0.048	0.027	0.087	0.112	0.154
YouTube-DNN	0.105	0.082	0.142	0.204	0.305
WikidataRec	**0.120**	**0.113**	**0.215**	**0.243**	**0.337**

5.3 Results on Wikidata-14M

Comparison to State of the Art. We compare *WikidataRec* against the following methods on *Wikidata-14M*: ***BPR-MF*** [32] is a representative collaborative filtering model that uses matrix factorization (MF) as the underlying predictor and is optimized with a pairwise ranking loss. It is suitable for recommendation scenarios with no explicit editor feedback and personalised ranked recommendation results. ***eALS*** [12] is also a MF-based method that is optimized with square loss. It treats all unobserved interactions as negative instances and weights them non-uniformly by the item's popularity. ***GMF*** [11] is a neural network-based collaborative filtering which implements MF with cross-entropy loss. It embeds each item and editor in the network and computes their element-wise dot product to predict the relevance score. ***YouTube-DNN*** [6] is a neural recommender for YouTube videos, using deep candidate generation and ranking networks. It uses a hybrid of users' activities and content information of users and items and directly learns their low dimensional representations. In this paper, we adapt YouTube-DNN with our item content and relational representations.

Table 3 shows the results: *WikidataRec* achieves the best performance over all with both accurate and diverse recommendations. First, it outperforms the collaborative-filtering (CF) methods GMF, eALS and BPR-MF by a large margin. CF methods only rely on the item-editor interactions data without taking into account the information of items themselves, whereas adding additional item content and relation information significantly improves the recommendation performance in our model (see Table 4). Our model also beats Youtube DNN. Youtube-DNN learns item's content and relational embedding in a neural network with random initialization while we employ additional state-of-the-art embedding models (i.e., ELMo and TransR) to generate them separately which yields superior performance.

Ablation Study. We ablate different components of *WikidataRec* on the *Wikidata-14M* to understand the effect of each one.

Item Contents and Relations. To justify the importance of item content and relations in *WikidataRec*, we start with the original collaborative filtering

Table 4. Ablation study of *WikidataRec* on *Wikidata-14M.*

	Precision @k		Recall @k			MAR@k		Diversity
	5	10	50	100	200	5	10	10
CF (BPR-MF)	0.050	0.043	0.093	0.135	0.190	0.069	0.107	0.252
+ item content	0.104	0.086	0.189	0.199	0.287	0.108	0.170	0.502
+ item relations	**0.120**	**0.113**	**0.215**	**0.243**	**0.337**	**0.133**	**0.224**	**0.567**
WikidataRec w/o NMoR	0.085	0.072	0.165	0.209	0.284	0.090	0.191	0.521
WikidataRec w/ NMoR	**0.120**	**0.113**	**0.215**	**0.243**	**0.337**	**0.133**	**0.224**	**0.567**

Table 5. Ablation study of embedding methods for item content on *Wikidata-14M.*

	Embedding model	Precision@5	Recall@50	**Recall@100**
BPR-MF +	Word2Vec	0.060	0.123	0.158
	FastText	0.029	0.069	0.115
	ELMo	**0.104**	**0.189**	**0.199**

(CF) technique, BPR-MF, and gradually add item content and relational representations using NMoR. The results are in Table 4 which show that values in every metric are increased by considering item content and relations. Particularly, item content representations provide rich information in terms of words and semantic meanings in items, which contribute more in performance increase; item relational representations in contrast contribute less. We can note each feature's contributions from the amount of increase when each of them is added. We suggest the reason as these relations are learned from the Wikidata graph, which is unevenly and sparingly connected, as discussed in [29].

Neural Mixture of Representations. To ablate the proposed NMoR, Table 4 further illustrates the results of *WikidataRec* with and without using it. For the latter, the item-based representations are added with no weights. WikidataRec with NMoR performs significantly better than WikidataRec without NMoR.

Embedding Models for Item Content. We compare the ELMo model with two text embedding models, Word2Vec [22] and FastText [14] to generate item content representations. Word2Vec and FastText are lightweight models which we train them from scratch using our Wikidata. The results are in Table 5: ELMo outperforms Word2Vec and FastText clearly. ELMo employs the deep bidirectional Language Model (biLM), which provides a very rich representation about the word tokens and captures the semantic and syntactic meaning of words. This is not the case in the Word2Vec and FastText who utilize shallow neural networks.

5.4 Results on Wikidata-Active-Editors

We ran another ablation study of our model on the second dataset, *Wikidata-Active-Editors*, introduced in Sect. 3.3. *Wikidata-Active-Editors* is a subset of

Table 6. Ablation study of *WikidataRec* on *Wikidata-Active-Editor*.

	Precision @k		Recall @k			MAR@k		Diversity
	5	10	50	100	200	5	10	10
CF (BPR-MF)	0.079	0.055	0.139	0.260	0.236	0.115	0.193	0.353
+ item content	0.143	0.109	0.253	0.302	0.349	0.157	0.251	0.565
+ item relations	**0.164**	**0.131**	**0.289**	**0.342**	**0.391**	**0.179**	**0.297**	**0.596**
WikidataRec w/o NMoR	0.132	0.073	0.196	0.295	0.323	0.134	0.245	0.553
WikidataRec w/ NMoR	**0.164**	**0.131**	**0.289**	**0.342**	**0.394**	**0.179**	**0.297**	**0.596**

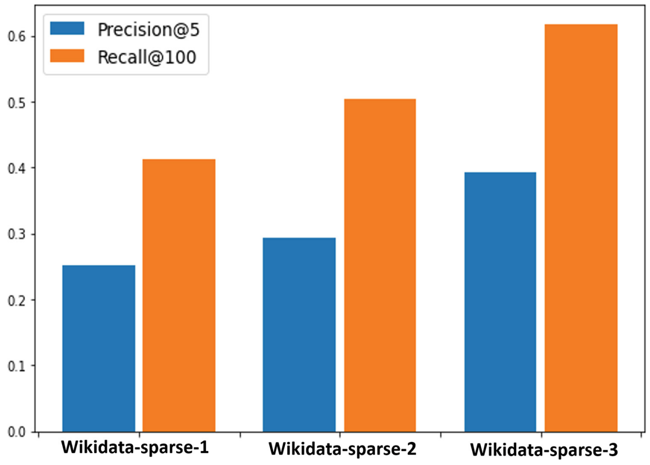


Fig. 4. *WikidataRec* with various sparsity levels.

Wikidata-14M, focusing on active editors and frequently edited items. The results are in Table 6, in which we note improved results on *WikidataRec* by adding item content and relations. In particular, content-based information contributed most towards the performance of the model. On the other hand, the contribution of relational information is less than that of content information. Furthermore, the results show that *WikidataRec* with NMoR performs better than *WikidataRec* without NMoR.

Dataset Sparsity. We further study the sparsity of the dataset in terms of its item-editor interaction data. Wikidata is very sparse in nature: a small number of interactions between items and editors are observed. *Wikidata-Active-Editors* is denser than *Wikidata-14M*, but its sparsity is still 99.90%, which means only 1% of interactions are observed over all the possible connections from every item to every editor in the dataset. To study the influence of the dataset sparsity, we extract three sub-datasets from *Wikidata-Active-Editors* with different sparsity yet roughly the same size (about 6000 editors and 10,000 items): *Wikidata-sparse-1* with 99.81% sparsity, *Wikidata-sparse-2* with 99.68% sparsity, and *Wikidata-sparse-3* with 99.27%. The editors in each dataset have edited more than 200 different items, and each item has edited by more than 16 different editors. Figure 4 show the Precision@5 and Recall@10 on the three subsets. It shows that when the sparsity decreases (from 99.81% to 99.27%, data gets denser), our model performance increases. This again verifies the conclusion in [1].

5.5 Discussion and Analysis of Results on both Datasets

Comparing the results of the two datasets (Table 4 and Table 6), we summarize that: 1) adding item content and relations with NMoR improves the performance on both datasets; 2) WikidataRec performs better on the *Wikidata-Active-Editors* than on the *Wikidata-14M*; the former is with denser editing data, so having less sparse data is likely to improve the results. This finding is consistent with the conclusion of [1].

6 Conclusion and Future Work

We present *WikidataRec*, a hybrid recommender model that recommends Wikidata items to editors based on their past editing activities. The work is motivated by Wikidata's quest for more (and more engaged) editors to keep up with a knowledge graph of growing size and complexity. As the first work of its kind, our focus is on establishing technical feasibility and providing a benchmark for future recommendation research in Wikidata. We do so with solid system implementation and benchmark datasets. Our model is informed by related research in content-based and collaborative filtering in similar verticals, which cannot rely upon explicit feedback for the recommendation task. It uses state-of-the-art models for representation learning and operates by means of a mixture of experts [13]. We employ ELMo [26] for items' content-based representations and TransR [20] for items' relations-based representations. The results, though far from perfect, are promising and could be considered a baseline for future Wikidata recommender work.

Based on our experiments, we make three claims for the recommendation task in Wikidata: 1) Collaborative filtering is not enough to recommend Wikidata items, as editing data is very sparse; however, adding item content and relational representations significantly enhances performance; 2) Not each item representation contributes to the final predictions equally. We show how to optimise the weights with NMoR; 3) While our model works better on the denser datasets extracted from *Wikidata-Active-Editors* data, showing that there is room for improvement.

We plan to extend the work in several directions. First, we plan to run editor studies, including interviews with more or less experienced editors to learn about their current ways to choose what they work on and perhaps uncover how existing technical affordances and interfaces influence such decisions. Second, we aim to experiment with other recommender models, particularly sequential learning and time-sensitive models that encode the temporal aspect and distinguish between older and more recent editor interests, as well as with ways to elicit more information about the editors in a responsible way. One option here would be to recruit new editors for a study in which they would explicitly consent to us collecting such information or would provide additional information about their interests themselves. Also, since the data of Wikidata can be represented as item-item relation graph and editor-item graph, exploring graph neural networks (GNNs) for a recommendation would be an interesting future direction. The advantages provided by the GNNs would provide great potential to advance our recommendation task. Third, topic-recommendation is another area that might interests the editors in Wikidata more than item recommendations. We plan to investigate this area. In addition, we are going to conducting an editor-centric evaluation would provide a space to examine and compare the feasibility and effectiveness of the two algorithms (topic vs. item recommendations).

References

1. Adomavicius, G., Zhang, J.: Impact of data characteristics on recommender systems performance. ACM Trans. Manag. Inf. Syst. (TMIS) (2012)
2. Aggarwal, C.C., et al.: Recommender systems (2016)
3. Bordes, A., Usunier, N., Garcia-Duran, A., Weston, J., Yakhnenko, O.: Translating embeddings for modeling multi-relational data. In: Neural Information Processing Systems (2013)
4. Chelba, C., et al.: One billion word benchmark for measuring progress in statistical language modeling. arXiv preprint arXiv:1312.3005 (2013)
5. Cosley, D., Frankowski, D., Terveen, L., Riedl, J.: SuggestBot: using intelligent task routing to help people find work in Wikipedia. In: International Conference on Intelligent User Interfaces (2007)
6. Covington, P., Adams, J., Sargin, E.: Deep neural networks for Youtube recommendations. In: ACM Conference on Recommender Systems (2016)
7. Dror, G., Koren, Y., Maarek, Y., Szpektor, I.: I want to answer; who has a question?: Yahoo! answers recommender system. In: ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (2011)
8. Esmeli, R., Bader-El-Den, M., Abdullahi, H.: Using Word2Vec recommendation for improved purchase prediction. In: International Joint Conference on Neural Networks (2020)
9. Freyne, J., Jacovi, M., Guy, I., Geyer, W.: Increasing engagement through early recommender intervention. In: ACM Conference on Recommender Systems (2009)
10. Hassan, H.A.M., Sansonetti, G., Gasparetti, F., Micarelli, A., Beel, J.: BERT, ELMO, USE and InferSent sentence encoders: the panacea for research-paper recommendation? In: RecSys (Late-Breaking Results) (2019)
11. He, X., Liao, L., Zhang, H., Nie, L., Hu, X., Chua, T.S.: Neural collaborative filtering. In: The International Conference on World Wide Web (2017)
12. He, X., Zhang, H., Kan, M.Y., Chua, T.S.: Fast matrix factorization for online recommendation with implicit feedback. In: International ACM SIGIR Conference on Research and Development in Information Retrieval (2016)
13. Jacobs, R.A., Jordan, M.I., Nowlan, S.J., Hinton, G.E.: Adaptive mixtures of local experts. Neural Comput. (1991)

14. Joulin, A., Grave, E., Bojanowski, P., Mikolov, T.: Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 (2016)
15. Kaffee, L.A., Piscopo, A., Vougiouklis, P., Simperl, E., Carr, L., Pintscher, L.: A glimpse into babel: an analysis of multilinguality in Wikidata. In: Proceedings of the 13th International Symposium on Open Collaboration, pp. 1–5 (2017)
16. Kanza, S., Stolz, A., Hepp, M., Simperl, E.: What does an ontology engineering community look like? A systematic analysis of the schema.org community. In: Gangemi, A., et al. (eds.) ESWC 2018. LNCS, vol. 10843, pp. 335–350. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-93417-4_22
17. Koren, Y., Bell, R., Volinsky, C.: Matrix factorization techniques for recommender systems. Computer (2009)
18. Kurup, A.R., Sajeev, G.: Task recommendation in reward-based crowdsourcing systems. In: International Conference on Advances in Computing, Communications and Informatics (2017)
19. Lave, J., Wenger, E.: Legitimate peripheral participation. Learning and Knowledge (1999)
20. Lin, Y., Liu, Z., Sun, M., Liu, Y., Zhu, X.: Learning entity and relation embeddings for knowledge graph completion. In: AAAI Conference on Artificial Intelligence (2015)
21. Liu, Z., Li, K., Qu, D.: Knowledge graph based question routing for community question answering. In: International Conference on Neural Information Processing (2017)
22. Mikolov, T., Chen, K., Corrado, G., Dean, J.: Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781 (2013)
23. Moskalenko, O., Parra, D., Saez-Trumper, D.: Scalable recommendation of Wikipedia articles to editors using representation learning. arXiv preprint arXiv:2009.11771 (2020)
24. Müller-Birn, C., Karran, B., Lehmann, J., Luczak-Rösch, M.: Peer-production system or collaborative ontology engineering effort: What is wikidata? In: International Symposium on Open Collaboration (2015)
25. Pennington, J., Socher, R., Manning, C.D.: Glove: Global vectors for word representation. In: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (2014)
26. Peters, M.E., et al.: Deep contextualized word representations. arXiv preprint arXiv:1802.05365 (2018)
27. Piscopo, A., Phethean, C., Simperl, E.: What makes a good collaborative knowledge graph: group composition and quality in Wikidata. In: International Conference on Social Informatics (2017)
28. Piscopo, A., Phethean, C., Simperl, E.: Wikidatians are born: paths to full participation in a collaborative structured knowledge base (2017)
29. Piscopo, A., Simperl, E.: Who models the world?: Collaborative ontology creation and user roles in Wikidata. In: The ACM on Human-Computer Interaction (2018)
30. Polignano, M., de Gemmis, M., Semeraro, G.: Contextualized BERT sentence embeddings for author profiling: the cost of performances. In: Gervasi, O., et al. (eds.) ICCSA 2020. LNCS, vol. 12252, pp. 135–149. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58811-3_10
31. Reimers, N., Gurevych, I.: Sentence-BERT: sentence embeddings using Siamese BERT-networks. arXiv preprint arXiv:1908.10084 (2019)
32. Rendle, S., Freudenthaler, C., Gantner, Z., Schmidt-Thieme, L.: BPR: Bayesian personalized ranking from implicit feedback. arXiv preprint arXiv:1205.2618 (2012)

33. Safran, M., Che, D.: Efficient learning-based recommendation algorithms for top-N tasks and top-N workers in large-scale crowdsourcing systems. ACM Trans. Inf. Syst. (TOIS) **37**(1), 1–46 (2018)
34. Sarasua, C., Checco, A., Demartini, G., Difallah, D., Feldman, M., Pintscher, L.: The evolution of power and standard Wikidata editors: comparing editing behavior over time to predict lifespan and volume of edits. Comput. Support. Coop. Work (CSCW) (2019)
35. Schafer, J.B., Frankowski, D., Herlocker, J., Sen, S.: Collaborative filtering recommender systems. In: The Adaptive Web (2007)
36. Shani, G., Gunawardana, A.: Evaluating recommendation systems. In: Recommender Systems Handbook (2011)
37. Sun, J., Vishnu, A., Chakrabarti, A., Siegel, C., Parthasarathy, S.: ColdRoute: effective routing of cold questions in stack exchange sites. Data Mining Knowl. Discov. (2018)
38. Sun, J., Zhao, J., Sun, H., Parthasarathy, S.: An end-to-end framework for cold question routing in community question answering services. arXiv preprint arXiv:1911.11017 (2019)
39. Turki, H., et al.: Wikidata: a large-scale collaborative ontological medical database. J. Biomed. Inf. (2019)
40. Vrandečić, D., Krötzsch, M.: Wikidata: a free collaborative knowledge base (2014)
41. Wang, B., Wang, A., Chen, F., Wang, Y., Kuo, C.C.J.: Evaluating word embedding models: methods and experimental results. APSIPA Trans. Sig. Inf. Process. (2019)
42. Wikipedia: Wikidata (2021). https://en.wikipedia.org/wiki/Wikidata. Accessed 26 June 2021
43. Xiao, H., Huang, M., Hao, Y., Zhu, X.: TransA: an adaptive approach for knowledge graph embedding. arXiv preprint arXiv:1509.05490 (2015)
44. Yang, L., Amatriain, X.: Recommending the world's knowledge: application of recommender systems at Quora. In: ACM Conference on Recommender Systems (2016)