A Capsule Network-based Embedding Model for Knowledge Graph Completion and Search Personalization

Dai Quoc Nguyen[1], **Thanh Vu**[2], **Tu Dinh Nguyen**[1], **Dat Quoc Nguyen**[3], **Dinh Phung**[1]
[1]Monash University, Australia; [3]The University of Melbourne, Australia
[2]The Australian e-Health Research Centre, CSIRO, Australia
[1]{dai.nguyen,tu.dinh.nguyen,dinh.phung}@monash.edu
[2]thanh.vu@csiro.au; [3]dqnguyen@unimelb.edu.au

Abstract

In this paper, we introduce an embedding model, named CapsE, exploring a capsule network to model relationship triples *(subject, relation, object)*. Our CapsE represents each triple as a 3-column matrix where each column vector represents the embedding of an element in the triple. This 3-column matrix is then fed to a convolution layer where multiple filters are operated to generate different feature maps. These feature maps are reconstructed into corresponding capsules which are then routed to another capsule to produce a continuous vector. The length of this vector is used to measure the plausibility score of the triple. Our proposed CapsE obtains better performance than previous state-of-the-art embedding models for knowledge graph completion on two benchmark datasets WN18RR and FB15k-237, and outperforms strong search personalization baselines on SEARCH17.

1 Introduction

Knowledge graphs (KGs) containing relationship triples *(subject, relation, object)*, denoted as *(s, r, o)*, are the useful resources for many NLP and especially information retrieval applications such as semantic search and question answering (Wang et al., 2017). However, large knowledge graphs, even containing billions of triples, are still incomplete, i.e., missing a lot of valid triples (West et al., 2014). Therefore, much research efforts have focused on the knowledge graph completion task which aims to predict missing triples in KGs, i.e., predicting whether a triple not in KGs is likely to be valid or not (Bordes et al., 2011, 2013; Socher et al., 2013). To this end, many embedding models have been proposed to learn vector representations for entities (i.e., *subject*/head entity and *object*/tail entity) and relations in KGs, and obtained state-of-the-art results as summarized by Nickel et al. (2016a) and Nguyen (2017). These embedding models score triples *(s, r, o)*, such that valid triples have higher plausibility scores than invalid ones (Bordes et al., 2011, 2013; Socher et al., 2013). For example, in the context of KGs, the score for *(Melbourne, cityOf, Australia)* is higher than the score for *(Melbourne, cityOf, United Kingdom)*.

Triple modeling is applied not only to the KG completion, but also for other tasks which can be formulated as a triple-based prediction problem. An example is in search personalization, one would aim to tailor search results to each specific user based on the user's personal interests and preferences (Teevan et al., 2005, 2009; Bennett et al., 2012; Harvey et al., 2013; Vu et al., 2015, 2017). Here the triples can be formulated as *(submitted query, user profile, returned document)* and used to re-rank documents returned to a user given an input query, by employing an existing KG embedding method such as TransE (Bordes et al., 2013), as proposed by Vu et al. (2017). Previous studies have shown the effectiveness of modeling triple for either KG completion or search personalization. However, there has been no single study investigating the performance on both tasks.

Conventional embedding models, such as TransE (Bordes et al., 2013), DISTMULT (Yang et al., 2015) and ComplEx (Trouillon et al., 2016), use addition, subtraction or simple multiplication operators, thus only capture the linear relationships between entities. Recent research has raised interest in applying deep neural networks to triple-based prediction problems. For example, Nguyen et al. (2018) proposed ConvKB—a convolutional neural network (CNN)-based model for KG completion and achieved state-of-the-art results. Most of KG embedding models are constructed to modeling entries at the same dimension of the given triple, where presumably each dimension captures some relation-specific attribute of entities. To the

Proceedings of NAACL-HLT 2019, pages 2180–2189
Minneapolis, Minnesota, June 2 - June 7, 2019.

best of our knowledge, however, none of the existing models has a "deep" architecture for modeling the entries in a triple at the same dimension.

Sabour et al. (2017) introduced capsule networks (CapsNet) that employ capsules (i.e., *each capsule is a group of neurons*) to capture entities in images and then uses a routing process to specify connections from capsules in a layer to those in the next layer. Hence CapsNet could encode the intrinsic spatial relationship between a part and a whole constituting viewpoint invariant knowledge that automatically generalizes to novel viewpoints. Each capsule accounts for capturing variations of an object or object part in the image, which can be efficiently visualized. Our high-level hypothesis is that embedding entries at the same dimension of the triple also have these variations, although it is not straightforward to be visually examined.

To that end, we introduce CapsE to explore a novel application of CapsNet on triple-based data for two problems: KG completion and search personalization. Different from the traditional modeling design of CapsNet where capsules are constructed by splitting feature maps, we use capsules to model the entries at the same dimension in the entity and relation embeddings. In our CapsE, $\boldsymbol{v}_s$, $\boldsymbol{v}_r$ and $\boldsymbol{v}_o$ are unique k-dimensional embeddings of s, r and o, respectively. The embedding triple $[\boldsymbol{v}_s, \boldsymbol{v}_r, \boldsymbol{v}_o]$ of *(s, r, o)* is fed to the convolution layer where multiple filters of the same 1×3 shape are repeatedly operated over every row of the matrix to produce k-dimensional feature maps. Entries at the same dimension from all feature maps are then encapsulated into a capsule. Thus, *each capsule can encode many characteristics in the embedding triple to represent the entries at the corresponding dimension*. These capsules are then routed to another capsule which outputs a continuous vector whose length is used as a score for the triple. Finally, this score is used to predict whether the triple *(s, r, o)* is valid or not.

In summary, our main contributions from this paper are as follows:

• We propose an embedding model CapsE using the capsule network (Sabour et al., 2017) for modeling relationship triples. To our best of knowledge, our work is the first consideration of exploring the capsule network to knowledge graph completion and search personalization.

• We evaluate our CapsE for knowledge graph completion on two benchmark datasets WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova and Chen, 2015). CapsE obtains the best mean rank on WN18RR and the highest mean reciprocal rank and highest Hits@10 on FB15k-237.

• We restate the prospective strategy of expanding the triple embedding models to improve the ranking quality of the search personalization systems. We adapt our model to search personalization and evaluate on SEARCH17 (Vu et al., 2017) – a dataset of the web search query logs. Experimental results show that our CapsE achieves the new state-of-the-art results with significant improvements over strong baselines.

2 The proposed CapsE

Let $\mathcal{G}$ be a collection of valid factual triples in the form of *(subject, relation, object)* denoted as *(s, r, o)*. Embedding models aim to define a *score function* giving a score for each triple, such that valid triples receive higher scores than invalid triples.

We denote $\boldsymbol{v}_s$, $\boldsymbol{v}_r$ and $\boldsymbol{v}_o$ as the k-dimensional embeddings of s, r and o, respectively. In our proposed CapsE, we follow Nguyen et al. (2018) to view each embedding triple $[\boldsymbol{v}_s, \boldsymbol{v}_r, \boldsymbol{v}_o]$ as a matrix $\boldsymbol{A} = [\boldsymbol{v}_s, \boldsymbol{v}_r, \boldsymbol{v}_o] \in \mathbb{R}^{k \times 3}$, and denote $\boldsymbol{A}_{i,:} \in \mathbb{R}^{1 \times 3}$ as the i-th row of $\boldsymbol{A}$. We use a filter $\boldsymbol{\omega} \in \mathbb{R}^{1 \times 3}$ operated on the convolution layer. This filter $\boldsymbol{\omega}$ is repeatedly operated over every row of $\boldsymbol{A}$ to generate a feature map $\mathbf{q} = [\mathsf{q}_1, \mathsf{q}_2, ..., \mathsf{q}_k] \in \mathbb{R}^k$, in which $\mathsf{q}_i = g\left(\boldsymbol{\omega} \cdot \boldsymbol{A}_{i,:} + b\right)$ where $\cdot$ denotes a dot product, $b \in \mathbb{R}$ is a bias term and g is a non-linear activation function such as ReLU. Our model uses multiple filters $\in \mathbb{R}^{1 \times 3}$ to generate feature maps. We denote $\boldsymbol{\Omega}$ as the set of filters and $\mathsf{N} = \mid \boldsymbol{\Omega} \mid$ as the number of filters, thus we have N k-dimensional feature maps, for which each feature map can capture one single characteristic among entries at the same dimension.

We build our CapsE with two single capsule layers for a simplified architecture. In the first layer, we construct k capsules, wherein entries at the same dimension from all feature maps are encapsulated into a corresponding capsule. Therefore, each capsule can capture many characteristics among the entries at the corresponding dimension in the embedding triple. These characteristics are generalized into one capsule in the second layer which produces a vector output whose length is used as the score for the triple.

The first capsule layer consists of k capsules, for which each capsule $i \in \{1, 2, ..., k\}$ has a vector

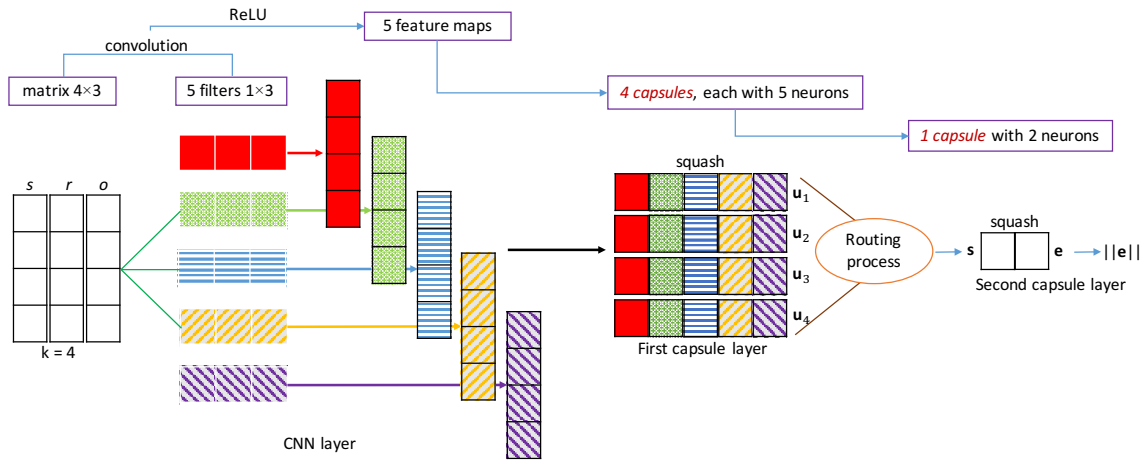

Figure 1: An example illustration of our CapsE with $k = 4$, $\mathsf{N} = 5$, and $d = 2$.

output $\mathbf{u}_i \in \mathbb{R}^{\mathsf{N}\times 1}$. Vector outputs $\mathbf{u}_i$ are multiplied by weight matrices $\mathbf{W}_i \in \mathbb{R}^{d\times \mathsf{N}}$ to produce vectors $\hat{\mathbf{u}}_i \in \mathbb{R}^{d\times 1}$ which are summed to produce a vector input $\mathbf{s} \in \mathbb{R}^{d\times 1}$ to the capsule in the second layer. The capsule then performs the non-linear squashing function to produce a vector output $\mathbf{e} \in \mathbb{R}^{d\times 1}$:

$$\mathbf{e} = \mathsf{squash}\left(\mathbf{s}\right) \quad ; \quad \mathbf{s} = \sum_i c_i \hat{\mathbf{u}}_i \quad ; \quad \hat{\mathbf{u}}_i = \mathbf{W}_i \mathbf{u}_i$$

where $\mathsf{squash}\left(\mathbf{s}\right) = \frac{\|\mathbf{s}\|^2}{1+\|\mathbf{s}\|^2}\frac{\mathbf{s}}{\|\mathbf{s}\|}$, and c_i are coupling coefficients determined by the routing process as presented in Algorithm 1. Because there is one capsule in the second layer, we make only one difference in the routing process proposed by Sabour et al. (2017), for which we apply the softmax in a direction from all capsules in the previous layer to each of capsules in the next layer.[1]

for *all capsule* $i \in$ *the first layer* **do**
  $b_i \leftarrow 0$
for iteration = *1, 2, ..., m* **do**
  $\mathbf{c} \leftarrow \mathsf{softmax}\left(\mathbf{b}\right)$
  $\mathbf{s} \leftarrow \sum_i c_i \hat{\mathbf{u}}_i$
  $\mathbf{e} = \mathsf{squash}\left(\mathbf{s}\right)$
  for *all capsule* $i \in$ *the first layer* **do**
    $b_i \leftarrow b_i + \hat{\mathbf{u}}_i \cdot \mathbf{e}$

Algorithm 1: The routing process is extended from Sabour et al. (2017).

[1]The softmax in the original routing process proposed by Sabour et al. (2017) is applied in another direction from each of capsules in the previous layer to all capsules in the next layer.

We illustrate our proposed model in Figure 1 where embedding size: $k = 4$, the number of filters: $\mathsf{N} = 5$, the number of neurons within the capsules in the first layer is equal to N, and the number of neurons within the capsule in the second layer: $d = 2$. The length of the vector output $\mathbf{e}$ is used as the score for the input triple.

Formally, we define the score function f for the triple (s, r, o) as follows:

$$f\left(s, r, o\right) = \|\mathsf{capsnet}\left(g\left(\left[\boldsymbol{v}_s, \boldsymbol{v}_r, \boldsymbol{v}_o\right] * \boldsymbol{\Omega}\right)\right)\|$$

where the set of filters $\boldsymbol{\Omega}$ is shared parameters in the convolution layer; $*$ denotes a convolution operator; and $\mathsf{capsnet}$ denotes a capsule network operator. We use the Adam optimizer (Kingma and Ba, 2014) to train CapsE by minimizing the loss function (Trouillon et al., 2016; Nguyen et al., 2018) as follows:

$$\mathcal{L} = \sum_{(s,r,o)\in\{\mathcal{G}\cup\mathcal{G}'\}} \log\left(1 + \exp\left(-t_{(s,r,o)} \cdot f\left(s, r, o\right)\right)\right)$$

$$\text{in which, } t_{(s,r,o)} = \begin{cases} 1 \text{ for } (s, r, o) \in \mathcal{G} \\ -1 \text{ for } (s, r, o) \in \mathcal{G}' \end{cases}$$

here $\mathcal{G}$ and $\mathcal{G}'$ are collections of valid and invalid triples, respectively. $\mathcal{G}'$ is generated by corrupting valid triples in $\mathcal{G}$.

3 Knowledge graph completion evaluation

In the knowledge graph completion task (Bordes et al., 2013), the goal is to predict a missing entity given a relation and another entity, i.e, inferring a head entity s given (r, o) or inferring a tail entity o given (s, r). The results are calculated based on ranking the scores produced by the score function f on test triples.

3.1 Experimental setup

Datasets: We use two recent benchmark datasets WN18RR (Dettmers et al., 2018) and FB15k-237 (Toutanova and Chen, 2015). These two datasets are created to avoid reversible relation problems, thus the prediction task becomes more realistic and hence more challenging (Toutanova and Chen, 2015). Table 1 presents the statistics of WN18RR and FB15k-237.

Dataset	#E	#R	#Triples in train/valid/test		
WN18RR	40,943	11	86,835	3,034	3,134
FB15k-237	14,541	237	272,115	17,535	20,466

Table 1: Statistics of the experimental datasets. #E is the number of entities. #R is the number of relations.

Evaluation protocol: Following Bordes et al. (2013), for each valid test triple (s, r, o), we replace either s or o by each of all other entities to create a set of corrupted triples. We use the "Filtered" setting protocol (Bordes et al., 2013), i.e., not taking any corrupted triples that appear in the KG into accounts. We rank the valid test triple and corrupted triples in descending order of their scores. We employ evaluation metrics: mean rank (MR), mean reciprocal rank (MRR) and Hits@10 (i.e., the proportion of the valid test triples ranking in top 10 predictions). Lower MR, higher MRR or higher Hits@10 indicate better performance. Final scores on the test set are reported for the model obtaining the highest Hits@10 on the validation set.

Training protocol: We use the common Bernoulli strategy (Wang et al., 2014; Lin et al., 2015b) when sampling invalid triples. For WN18RR, Pinter and Eisenstein (2018)[2] found a strong evidence to support the necessity of a WordNet-related semantic setup, in which they averaged pre-trained word embeddings for word surface forms within the WordNet to create synset embeddings, and then used these synset embeddings to initialize entity embeddings for training their TransE association model. We follow this evidence in using the pre-trained 100-dimensional Glove word embeddings (Pennington et al., 2014) to train a TransE model on WN18RR.

[2]Pinter and Eisenstein (2018) considered WN18RR and evaluated their M3GM model only for 7 relations as they employed the inverse rule model (Dettmers et al., 2018) for 4 remaining symmetric relations. Regarding a fair comparison to other models, we use the M3GM implementation released by Pinter and Eisenstein (2018) to re-train and re-evaluate the M3GM model for all 11 relations. We thank Pinter and Eisenstein (2018) for their assistance running their code.

We employ the TransE and ConvKB implementations provided by Nguyen et al. (2016b) and Nguyen et al. (2018). For ConvKB, we use a new process of training up to 100 epochs and monitor the Hits@10 score after every 10 training epochs to choose optimal hyper-parameters with the Adam initial learning rate in $\{1e^{-5}, 5e^{-5}, 1e^{-4}\}$ and the number of filters N in $\{50, 100, 200, 400\}$. We obtain the highest Hits@10 scores on the validation set when using N= 400 and the initial learning rate $5e^{-5}$ on WN18RR; and N= 100 and the initial learning rate $1e^{-5}$ on FB15k-237.

Like in ConvKB, we use the same pre-trained entity and relation embeddings produced by TransE to initialize entity and relation embeddings in our CapsE for both WN18RR and FB15k-237 ($k = 100$). We set the batch size to 128, the number of neurons within the capsule in the second capsule layer to 10 ($d = 10$), and the number of iterations in the routing algorithm m in $\{1, 3, 5, 7\}$. We run CapsE up to 50 epochs and monitor the Hits@10 score after each 10 training epochs to choose optimal hyper-parameters. The highest Hits@10 scores for our CapsE on the validation set are obtained when using $m = 1$, $\mathsf{N} = 400$ and the initial learning rate at $1e^{-5}$ on WN18RR; and $m = 1$, $\mathsf{N} = 50$ and the initial learning rate at $1e^{-4}$ on FB15k-237.

3.2 Main experimental results

Table 2 compares the experimental results of our CapsE with previous state-of-the-art published results, using the same evaluation protocol. Our CapsE performs better than its closely related CNN-based model ConvKB on both experimental datasets (except Hits@10 on WN18RR and MR on FB15k-237), especially on FB15k-237 where our CapsE gains significant improvements of $0.523 - 0.418 = 0.105$ in MRR (which is about 25.1% relative improvement), and $59.3\% - 53.2\% = 6.1\%$ absolute improvement in Hits@10. Table 2 also shows that our CapsE obtains the best MR score on WN18RR and the highest MRR and Hits@10 scores on FB15k-237.

Following Bordes et al. (2013), for each relation r in FB15k-237, we calculate the averaged number η_s of head entities per tail entity and the averaged number η_o of tail entities per head entity. If η_s <1.5 and η_o <1.5, r is categorized one-to-one (1-1). If η_s <1.5 and η_o ≥1.5, r is categorized one-to-many (1-M). If η_s ≥1.5 and η_o <1.5, r is

Method	WN18RR			FB15k-237		
	MR	MRR	H@10	MR	MRR	H@10
DISTMULT (Yang et al., 2015)	5110	0.425	49.1	254	0.241	41.9
ComplEx (Trouillon et al., 2016)	5261	**0.444**	50.7	339	0.247	42.8
ConvE (Dettmers et al., 2018)	4187	0.433	51.5	**244**	0.325	50.1
KBGAN (Cai and Wang, 2018)	–	0.213	48.1	–	0.278	45.8
M3GM (Pinter and Eisenstein, 2018)	1864	0.311	53.3	–	–	–
TransE (Bordes et al., 2013)	743⋆	0.245⋆	56.0⋆	347	0.294	46.5
ConvKB (Nguyen et al., 2018)	763⋆	0.253⋆	**56.7**⋆	254⋆	0.418⋆	53.2⋆
Our **CapsE**	**719**	0.415	56.0	303	**0.523**	**59.3**

Table 2: Experimental results on the WN18RR and FB15k-237 test sets. Hits@10 (H@10) is reported in %. Results of DISTMULT, ComplEx and ConvE are taken from Dettmers et al. (2018). Results of TransE on FB15k-237 are taken from Nguyen et al. (2018). Our CapsE Hits@1 scores are 33.7% on WN18RR and 48.9% on FB15k-237. Formulas of MRR and Hits@1 show a strong correlation, so using Hits@1 does not really reveal any additional information for this task. The best score is in bold, while the second best score is in underline. ⋆ denotes *our new results* for TransE and ConvKB, which are better than those published by Nguyen et al. (2018).

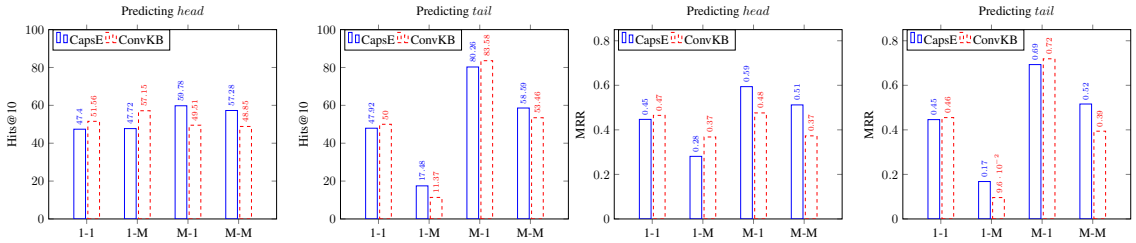

Figure 2: Hits@10 (in %) and MRR on the FB15k-237 test set w.r.t each relation category.

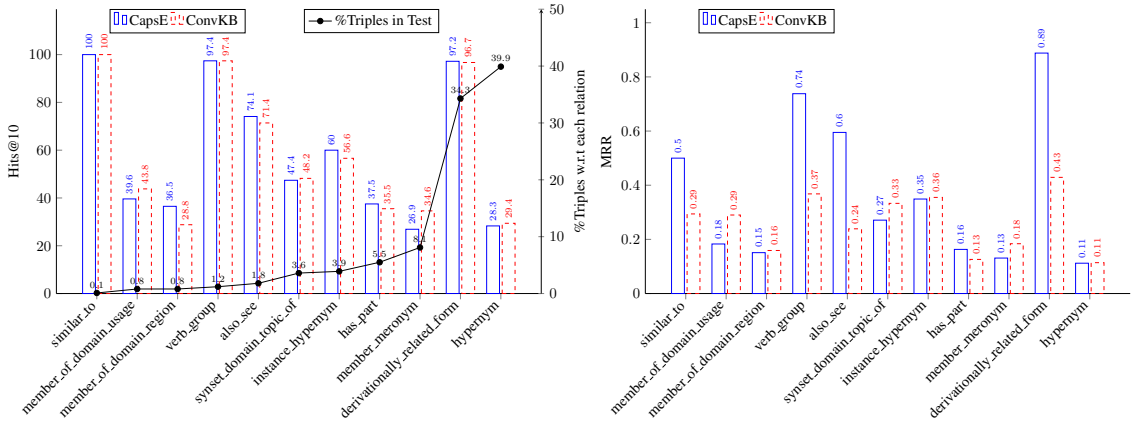

Figure 3: Hits@10 and MRR on the WN18RR test set w.r.t each relation. The right y-axis is the percentage of triples corresponding to relations.

categorized many-to-one (M-1). If $\eta_s \geq 1.5$ and $\eta_o \geq 1.5$, r is categorized many-to-many (M-M). As a result, 17, 26, 81 and 113 relations are labelled 1-1, 1-M, M-1 and M-M, respectively. And 0.9%, 6.3%, 20.5% and 72.3% of the test triples in FB15k-237 contain 1-1, 1-M, M-1 and M-M relations, respectively.

Figure 2 shows the Hits@10 and MRR results for predicting head and tail entities w.r.t each relation category on FB15k-237. CapsE works better than ConvKB in predicting entities on the "side M" of triples (e.g., predicting *head* entities in M-1 and M-M; and predicting *tail* entities in 1-M and M-M), while ConvKB performs better than CapsE in predicting entities on the "side 1" of triples (i.e., predicting *head* entities in 1-1 and 1-M; and predicting *tail* entities in 1-1 and M-1).

Figure 3 shows the Hits@10 and MRR scores w.r.t each relation on WN18RR. *also_see*, *similar_to*, *verb_group* and *derivationally_related_form* are symmetric relations which can be considered as M-M relations. Our CapsE also performs better than ConvKB on these 4 M-M relations. Thus, results

m	10	20	30	40	50
1	**48.37**	**52.60**	**53.14**	**53.33**	**53.21**
3	47.78	52.34	52.93	52.99	52.86
5	47.03	52.25	45.80	45.99	45.76
7	40.46	45.36	45.79	45.85	45.93

Table 3: Hits@10 on the WN18RR validation set with $\mathsf{N} = 50$ and the initial learning rate at $1e^{-5}$ w.r.t each number of iterations in the routing algorithm m and each 10 training epochs.

shown in Figures 2 and 3 are consistent. These also imply that our CapsE would be a potential candidate for applications which contain many M-M relations such as search personalization.

We see that the length and orientation of each capsule in the first layer can also help to model the important entries in the corresponding dimension, thus CapsE can work well on the "side M" of triples where entities often appear less frequently than others appearing in the "side 1" of triples. Additionally, existing models such as DISTMULT, ComplEx and ConvE can perform well for entities with high frequency, but may not for rare entities with low frequency. These are reasons why our CapsE can be considered as the best one on FB15k-237 and it outperforms most existing models on WN18RR.

Effects of routing iterations: We study how the number of routing iterations affect the performance. Table 3 shows the Hits@10 scores on the WN18RR validation set for a comparison w.r.t each number value of the routing iterations and epochs with the number of filters $\mathsf{N} = 50$ and the Adam initial learning rate at $1e^{-5}$. We see that the best performance for each setup over each 10 epochs is obtained by setting the number m of routing iterations to 1. This indicates the opposite side for knowledge graphs compared to images. In the image classification task, setting the number m of iterations in the routing process higher than 1 helps to capture the relative positions of entities in an image (e.g., eyes, nose and mouth) properly. In contrast, this property from images may be only right for the 1-1 relations, but not for the 1-M, M-1 and M-M relations in the KGs because of the high variant of each relation type (e.g., symmetric relations) among different entities.

4 Search personalization application

Given a *user*, a submitted *query* and the *documents* returned by a search system for that query, our approach is to re-rank the returned documents so that the more relevant documents should be ranked higher. Following Vu et al. (2017), we represent the relationship between the submitted query, the user and the returned document as a *(s, r, o)*-like triple *(query, user, document)*. The triple captures how much interest a user puts on a document given a query. Thus, we can evaluate the effectiveness of our CapsE for the search personalization task.

4.1 Experimental setup

Dataset: We use the SEARCH17 dataset (Vu et al., 2017) of query logs of 106 users collected by a large-scale web search engine. A log entity consists of a user identifier, a query, top-10 ranked documents returned by the search engine and clicked documents along with the user's dwell time. Vu et al. (2017) constructed short-term (session-based) user profiles and used the profiles to personalize the returned results. They then employed the SAT criteria (Fox et al., 2005) to identify whether a returned document is relevant from the query logs as either a clicked document with a dwell time of at least 30 seconds or the last clicked document in a search session (i.e., a SAT click). After that, they assigned a $relevant$ label to a returned document if it is a SAT click and also assigned $irrelevant$ labels to the remaining top-10 documents. The rank position of the $relevant$ labeled documents is used as the ground truth to evaluate the search performance before and after re-ranking.

The dataset was uniformly split into the training, validation and test sets. This split is for the purpose of using historical data in the training set to predict new data in the test set (Vu et al., 2017). The training, validation and test sets consist of 5,658, 1,184 and 1,210 relevant (i.e., valid) triples; and 40,239, 7,882 and 8,540 irrelevant (i.e., invalid) triples, respectively.

Evaluation protocol: Our CapsE is used to re-rank the original list of documents returned by a search engine as follows: (i) We train our model and employ the trained model to calculate the score for each (s, r, o) triple. (ii) We then sort the scores in the descending order to obtain a new ranked list. To evaluate the performance of our proposed model, we use two standard evaluation metrics: mean reciprocal rank (MRR) and Hits@1.[3] For each metric, the higher value indi-

[3]We re-rank the list of top-10 documents returned by the

cates better ranking performance.

We compare CapsE with the following baselines using the same experimental setup: (**1**) SE: The original rank is returned by the search engine. (**2**) CI (Teevan et al., 2011): This baseline uses a personalized navigation method based on previously clicking returned documents. (**3**) SP (Bennett et al., 2012; Vu et al., 2015): A search personalization method makes use of the session-based user profiles. (**4**) Following Vu et al. (2017), we use TransE as a strong baseline model for the search personalization task. Previous work shows that the well-known embedding model TransE, despite its simplicity, obtains very competitive results for the knowledge graph completion (Lin et al., 2015a; Nickel et al., 2016b; Trouillon et al., 2016; Nguyen et al., 2016a, 2018). (**5**) The CNN-based model ConvKB is the most closely related model to our CapsE.

Embedding initialization: We follow Vu et al. (2017) to initialize user profile, query and document embeddings for the baselines TransE and ConvKB, and our CapsE.

We train a LDA topic model (Blei et al., 2003) with 200 topics only on the *relevant* documents (i.e., SAT clicks) extracted from the query logs. We then use the trained LDA model to infer the probability distribution over topics for every returned document. We use the topic proportion vector of each document as its document embedding (i.e. $k = 200$). In particular, the z^{th} element ($z = 1, 2, ..., k$) of the vector embedding for document d is: $\boldsymbol{v}_{d,z} = \mathrm{P}(z \mid d)$ where $\mathrm{P}(z \mid d)$ is the probability of the topic z given the document d.

We also represent each query by a probability distribution vector over topics. Let $\mathcal{D}_q = \{d_1, d_2, ..., d_n\}$ be the set of top n ranked documents returned for a query q (here, $n = 10$). The z^{th} element of the vector embedding for query q is defined as in (Vu et al., 2017): $\boldsymbol{v}_{q,z} = \sum_{i=1}^{n} \lambda_i \mathrm{P}(z \mid d_i)$, where $\lambda_i = \frac{\delta^{i-1}}{\sum_{j=1}^{n} \delta^{j-1}}$ is the exponential decay function of i which is the rank of d_i in D_q. And δ is the decay hyper-parameter ($0 < \delta < 1$). Following Vu et al. (2017), we use $\delta = 0.8$. Note that if we learn query and document embeddings during training, the models will overfit to the data and will not work for new queries and documents. Thus, after the initialization process, we fix (i.e., not updating) query and document embeddings during training for TransE, ConvKB and CapsE.

In addition, as mentioned by Bennett et al. (2012), the more recently clicked document expresses more about the user current search interest. Hence, we make use of the user clicked documents in the training set with the temporal weighting scheme proposed by Vu et al. (2015) to initialize user profile embeddings for the three embedding models.

Hyper-parameter tuning: For our CapsE model, we set batch size to 128, and also the number of neurons within the capsule in the second capsule layer to 10 ($d = 10$). The number of iterations in the routing algorithm is set to 1 ($m = 1$). For the training model, we use the Adam optimizer with the initial learning rate $\in \{5e^{-6}, 1e^{-5}, 5e^{-5}, 1e^{-4}, 5e^{-4}\}$. We also use ReLU as the activation function g. We select the number of filters $\mathsf{N} \in \{50, 100, 200, 400, 500\}$. We run the model up to 200 epochs and perform a grid search to choose optimal hyper-parameters on the validation set. We monitor the MRR score after each training epoch and obtain the highest MRR score on the validation set when using $\mathsf{N} = 400$ and the initial learning rate at $5e^{-5}$.

We employ the TransE and ConvKB implementations provided by Nguyen et al. (2016b) and Nguyen et al. (2018) and then follow their training protocols to tune hyper-parameters for TransE and ConvKB, respectively. We also monitor the MRR score after each training epoch and attain the highest MRR score on the validation set when using margin = 5, l_1-norm and SGD learning rate at $5e^{-3}$ for TransE; and $\mathsf{N} = 500$ and the Adam initial learning rate at $5e^{-4}$ for ConvKB.

4.2 Main results

Table 4 presents the experimental results of the baselines and our model. Embedding models TranE, ConvKB and CapsE produce better ranking performances than traditional learning-to-rank search personalization models CI and SP. This indicates a prospective strategy of expanding the triple embedding models to improve the ranking quality of the search personalization systems. In particular, our MRR and Hits@1 scores are higher than those of TransE (with relative improvements of 14.5% and 22% over TransE, respectively). Specifically, our CapsE achieves the highest performances in both MRR and Hits@1 (our improvements over all five baselines are statistically

search engine, so Hits@10 scores are same for all models.

Method	MRR	H@1
SE [⋆]	0.559	38.5
CI [⋆]	0.597	41.6
SP [⋆]	0.631	45.2
TransE [⋆]	0.645	48.1
TransE (ours)	0.669	50.9
ConvKB	$0.750_{+12.1\%}$	$59.9_{+17.7\%}$
Our CapsE	$\mathbf{0.766}_{+14.5\%}$	$\mathbf{62.1}_{+22.0\%}$

Table 4: Experimental results on the test set. [⋆] denotes the results reported in (Vu et al., 2017). Hits@1 (H@1) is reported in %. In information retrieval, Hits@1 is also referred to as P@1. The subscripts denote the relative improvement over our TransE results.

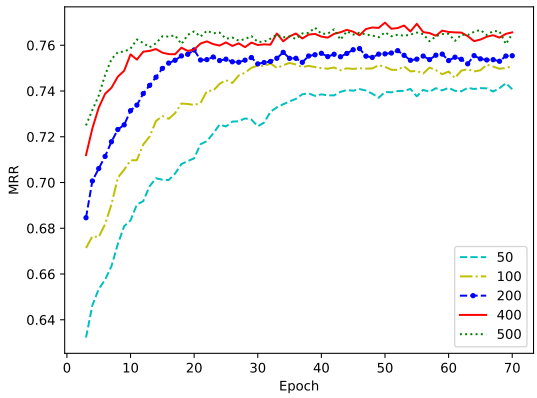

Figure 4: Learning curves on the validation set with the initial learning rate at $5e^{-5}$.

significant with $p < 0.05$ using the *paired t-test*).

To illustrate our training progress, we plot performances of CapsE on the validation set over epochs in Figure 4. We observe that the performance is improved with the increase in the number of filters since capsules can encode more useful properties for a large embedding size.

5 Related work

Other transition-based models extend TransE to additionally use projection vectors or matrices to translate embeddings of s and o into the vector space of r, such as: TransH (Wang et al., 2014), TransR (Lin et al., 2015b), TransD (Ji et al., 2015) and STransE (Nguyen et al., 2016b). Furthermore, DISTMULT (Yang et al., 2015) and ComplEx (Trouillon et al., 2016) use a tri-linear dot product to compute the score for each triple. Moreover, ConvKB (Nguyen et al., 2018) applies convolutional neural network, in which feature maps are concatenated into a single feature vector which is then computed with a weight vector via a dot product to produce the score for the input triple. ConvKB is the most closely related model to our CapsE. See an overview of embedding models for KG completion in (Nguyen, 2017).

For search tasks, unlike classical methods, personalized search systems utilize the historical interactions between the user and the search system, such as submitted queries and clicked documents to tailor returned results to the need of that user (Teevan et al., 2005, 2009). That historical information can be used to build the *user profile*, which is crucial to an effective search personalization system. Widely used approaches consist of two separated steps: (1) building the user profile from the interactions between the user and the search system; and then (2) learning a ranking function to *re-rank* the search results using the user profile (Bennett et al., 2012; White et al., 2013; Harvey et al., 2013; Vu et al., 2015). The general goal is to re-rank the documents returned by the search system in such a way that the more relevant documents are ranked higher. In this case, apart from the user profile, dozens of other features have been proposed as the input of a learning-to-rank algorithm (Bennett et al., 2012; White et al., 2013). Alternatively, Vu et al. (2017) modeled the potential *user*-oriented relationship between the submitted query and the returned document by applying TransE to reward higher scores for more relevant documents (e.g., clicked documents). They achieved better performances than the standard ranker as well as competitive search personalization baselines (Teevan et al., 2011; Bennett et al., 2012; Vu et al., 2015).

6 Conclusion

We propose CapsE—a novel embedding model using the capsule network to model relationship triples for knowledge graph completion and search personalization. Experimental results show that our CapsE outperforms other state-of-the-art models on two benchmark datasets WN18RR and FB15k-237 for the knowledge graph completion. We then show the effectiveness of our CapsE for the search personalization, in which CapsE outperforms the competitive baselines on the dataset SEARCH17 of the web search query logs. In addition, our CapsE is capable to effectively model many-to-many relationships. Our code is available at: https://github.com/daiquocnguyen/CapsE.

Acknowledgements

This research was partially supported by the ARC Discovery Projects DP150100031 and DP160103934. The authors thank Yuval Pinter for assisting us in running his code.

References

Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyuk, and Xiaoyuan Cui. 2012. Modeling the impact of short- and long-term behavior on search personalization. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 185–194.

David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022.

Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26*, pages 2787–2795.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning Structured Embeddings of Knowledge Bases. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 301–306.

Liwei Cai and William Yang Wang. 2018. KBGAN: Adversarial Learning for Knowledge Graph Embeddings. In *Proceedings of The 16th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, page to appear.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2D Knowledge Graph Embeddings. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence*, pages 1811–1818.

Steve Fox, Kuldeep Karnawat, Mark Mydland, Susan Dumais, and Thomas White. 2005. Evaluating implicit measures to improve web search. *ACM Transactions on Information Systems*, 23(2):147–168.

Morgan Harvey, Fabio Crestani, and Mark J. Carman. 2013. Building user profiles from topic models for personalised search. In *Proceedings of the ACM International Conference on Information and Knowledge Management*, pages 2309–2314.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge Graph Embedding via Dynamic Mapping Matrix. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 687–696.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yankai Lin, Zhiyuan Liu, Huanbo Luan, Maosong Sun, Siwei Rao, and Song Liu. 2015a. Modeling Relation Paths for Representation Learning of Knowledge Bases. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 705–714.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence Learning*, pages 2181–2187.

Dai Quoc Nguyen, Tu Dinh Nguyen, Dat Quoc Nguyen, and Dinh Phung. 2018. A Novel Embedding Model for Knowledge Base Completion Based on Convolutional Neural Network. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 327–333.

Dat Quoc Nguyen. 2017. An overview of embedding models of entities and relationships for knowledge base completion. *arXiv preprint*, arXiv:1703.08098.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016a. Neighborhood Mixture Model for Knowledge Base Completion. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 40–50.

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016b. STransE: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–466.

Maximilian Nickel, Kevin Murphy, Volker Tresp, and Evgeniy Gabrilovich. 2016a. A Review of Relational Machine Learning for Knowledge Graphs. *Proceedings of the IEEE*, 104(1):11–33.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016b. Holographic Embeddings of Knowledge Graphs. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 1955–1961.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543.

Yuval Pinter and Jacob Eisenstein. 2018. Predicting Semantic Relations using Global Graph Properties. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1741–1751.

Sara Sabour, Nicholas Frosst, and Geoffrey E Hinton. 2017. Dynamic routing between capsules. In *Advances in Neural Information Processing Systems*, pages 3859–3869.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning With Neural Tensor Networks for Knowledge Base Completion. In *Advances in Neural Information Processing Systems 26*, pages 926–934.

Jaime Teevan, Susan T. Dumais, and Eric Horvitz. 2005. Personalizing search via automated analysis of interests and activities. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 449–456.

Jaime Teevan, Daniel J. Liebling, and Gayathri Ravichandran Geetha. 2011. Understanding and predicting personal navigation. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 85–94.

Jaime Teevan, Meredith Ringel Morris, and Steve Bush. 2009. Discovering and using groups to improve personalized search. In *Proceedings of the ACM International Conference on Web Search and Data Mining*, pages 15–24.

Kristina Toutanova and Danqi Chen. 2015. Observed Versus Latent Features for Knowledge Base and Text Inference. In *Proceedings of the 3rd Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66.

Théo Trouillon, Johannes Welbl, Sebastian Riedel, Éric Gaussier, and Guillaume Bouchard. 2016. Complex Embeddings for Simple Link Prediction. In *Proceedings of the 33nd International Conference on Machine Learning*, pages 2071–2080.

Thanh Vu, Dat Quoc Nguyen, Mark Johnson, Dawei Song, and Alistair Willis. 2017. Search personalization with embeddings. In *Proceedings of the European Conference on Information Retrieval*, pages 598–604.

Thanh Vu, Alistair Willis, Son Ngoc Tran, and Dawei Song. 2015. Temporal latent topic user profiles for search personalisation. In *Proceedings of the European Conference on Information Retrieval*, pages 605–616.

Quan Wang, Zhendong Mao, Bin Wang, and Li Guo. 2017. Knowledge Graph Embedding: A Survey of Approaches and Applications. *IEEE Transactions on Knowledge and Data Engineering*, 29(12):2724–2743.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, pages 1112–1119.

Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. 2014. Knowledge Base Completion via Search-based Question Answering. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 515–526.

Ryen W. White, Wei Chu, Ahmed Hassan, Xiaodong He, Yang Song, and Hongning Wang. 2013. Enhancing personalized search by mining and modeling task behavior. In *Proceedings of the World Wide Web conference*, pages 1411–1420.

Bishan Yang, Wen-tau Yih, Xiaodong He, Jianfeng Gao, and Li Deng. 2015. Embedding Entities and Relations for Learning and Inference in Knowledge Bases. In *Proceedings of the International Conference on Learning Representations*.