# Multi-view Classification Model for Knowledge Graph Completion

**Wenbin Jiang** [1], **Mengfei Guo** [2]*,
**Yufeng Chen** [2], **Ying Li** [1], **Jinan Xu** [2], **Yajuan Lyu** [1], **Yong Zhu** [1]
[1]Baidu Inc., Beijing, China
[2]School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
{jiangwenbin, nicole, lvyajuan, zhuyong}@baidu.com
{guomengfei, jaxu, chenyf}@bjtu.edu.cn

## Abstract

Most previous work on knowledge graph completion conducted single-view prediction or calculation for candidate triple evaluation, based only on the content information of the candidate triples. This paper describes a novel multi-view classification model for knowledge graph completion, where multiple classification views are performed based on both content and context information for candidate triple evaluation. Each classification view evaluates the validity of a candidate triple from a specific viewpoint, based on the content information inside the candidate triple and the context information nearby the triple. These classification views are implemented by a unified neural network and the classification predictions are weightedly integrated to obtain the final evaluation. Experiments show that, the multi-view model brings very significant improvements over previous methods, and achieves the new state-of-the-art on two representative datasets. We believe that, the flexibility and the scalability of the multi-view classification model facilitates the introduction of additional information and resources for better performance.

## 1 Introduction

Knowledge graph (KG) is a typical kind of graph-structured knowledge base (KB). Nowdays, there exist many famous KGs such as YAGO (Suchanek et al., 2007), Freebase (Bollacker et al., 2008) and DBpedia (Lehmann et al., 2015). Large-scale KGs are widely used in many applications such as semantic searching (Kasneci et al., 2008; Schuhmacher and Ponzetto, 2014; Xiong et al., 2017), question answering (Zhang et al., 2016; Hao et al., 2017) and machine reading (Yang and Mitchell, 2017). A KG contains a set of triples indicating facts, each of which is composed of a *head* entity, a *tail* entity, and a *relation* indicating the relationship between the two entities. It is nearly impossible to collect a complete set of facts or triples for a KG, especially in open domains. In fact, many valuable valid triples are missing even for the existing well-built large-scale KGs such as Freebase (Socher et al., 2013; West et al., 2014). Many researchers devote their efforts to the problem of knowledge graph completion (KGC), the core operation of which is to evaluate the validity of candidate triples.

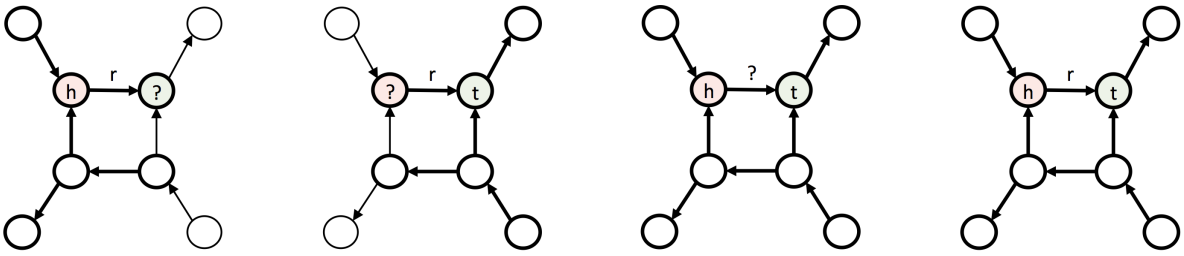
Previous work on KGC mainly include two groups, embedding-based methods and classification-based methods. Embedding-based models learn embeddings for entities and relations, and evaluate candidate triples based on the embeddings and specific distance metrics. Representative models include TransE (Bordes et al., 2013) and its extensions (Wang et al., 2014; Lin et al., 2015b; Ji et al., 2015; Nguyen et al., 2016), DistMult (Yang et al., 2015) and ComplEx (Trouillon et al., 2016). Classification-based models learn neural networks to evaluate the validity of candidate triples. Representative models include ConvE (Dettmers et al., 2018) and ConvKB (Nguyen, 2017). The major advantage of classification-based methods is that they directly model the evaluation of the validity of candidate triples, probably leading to better performance. Most of these previous work conducted single-view prediction based on content information, that is, evaluating a candidate triple according to a single distance metric or classification schema, resorting to information restricted in the scope of the candidate triple. We believe that multiple learning views for triple evaluation as well as context information of the candidate triple would contribute to better performance.

In this work, we propose for KGC a novel multi-view classification model, where multiple classifi-

*Joint first author. Guo participated in the optimization of this work during the internship in Baidu.



*Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 726–734




a) hr→t b) rt→h c) ht→r d) hrt→

Figure 1: Illustration of sub-graphs corresponding to the learning views. The colored nodes indicate the head and tail entities of the candidate triple. The bold nodes and edges are the elements in the retrieved sub-graphs. The question marks indicate the elements to be predicted.

cation views are performed to estimate the validity of a candidate triple, based on both content and context information of the triple. There are four classification views for candidate triple evaluation. Each of the first three views performs component prediction, where a specific component of the candidate triple is predicted according to the other two components as well as its nearby triples. The last view performs plausibility prediction, where the plausibility of the candidate triple is predicted according to its components as well as its nearby triples. The prediction conditions of these views investigate both content and context information of the candidate triple, that is, the components in the candidate triple, and the triples nearby the candidate triple. The content and context information can be represented as a sub-graph surrounding the candidate triple. These classification views are implemented by a unified neural network with shared embedding and encoding layers and separated prediction layers, and the classification predictions are integrated by a weighted integration procedure for better candidate triple evaluation. In the unified neural network, the sub-graphs indicating the content and context of the candidate triples are encoded in a sequencial manner, by converting the sub-graphs into sequential tree representations. It facilitates the utilization of advanced encoders such as BiLSTM or Transformer.

We experiment on two widely used benchmark datasets, FB15k-237 and WN18RR, specific versions of Freebase and WordNet. We find that the multi-view model achieves the new state-of-the-art, significantly outperforming pervious work on KGC. We also find that we can promote the efficiency of the multi-view model in realistic applications, by a coarse-to-fine strategy where the first two views are performed to give a list of candidates, and the overall model is then performed to evaluate these candidates. We believe that, the flexibility and the scalability of the multi-view classification model facilitates the introduction of additional information and resources for better performance.

## 2 Related Work

Most existing KGC models are based on KG embeddings, which aims at learning distributed representations for entities and relations in a KG. In these models, the candidate triples are evaluated by some specific distance metrics based on the embeddings. These models perform embedding learning with local information in individual triples, including translation-based models (Bordes et al., 2013; Wang et al., 2014; Lin et al., 2015b), semantic matching models (Yang et al., 2015; Nickel et al., 2016; Trouillon et al., 2016), and neural network models (Dettmers et al., 2018; Jiang et al., 2019; Nguyen, 2017). There also exist KGC models based on classification, where classifiers are learnt to evaluate the validity of candidate triples (Dettmers et al., 2018; Nguyen, 2017). Both kinds of previous work consider only one view, with simple distance metrics and classification operations. In contrast, multi-view learning enables the incorporation of much more views that utilize internal and external information for triple evaluation.

In recent years, many efforts were devoted to embedding learning based on non-local information such as multi-hop paths (Lin et al., 2015a; Das et al., 2017) and $k$-degree neighborhoods (Feng et al., 2016; Schlichtkrull et al., 2017). Some researchers also investigated graph embeddings in social network and other areas (Perozzi et al., 2014; Grover and Leskovec, 2016; Ristoski and Paulheim, 2016; Cochez et al., 2017). Compared with these work, our method not only learns em-

| View Type | Instance from $g_v$ | Instance from $g_v^-$ |
|---|---|---|
| **hr→t** | $g_{\mathbf{hr}\rightarrow\mathbf{t}} = \langle \mathcal{G}(h, r, ?), t \rangle$ | $g^-_{\mathbf{hr}\rightarrow\mathbf{t}} = \langle \mathcal{G}(\mathcal{S}(h, r, ?)), \mathbf{none} \rangle$, $\mathbf{s.t.}\ \mathcal{S}(h, r, ?) \notin \mathcal{KG}$ |
| **rt→h** | $g_{\mathbf{rt}\rightarrow\mathbf{h}} = \langle \mathcal{G}(?, r, t), h \rangle$ | $g^-_{\mathbf{rt}\rightarrow\mathbf{h}} = \langle \mathcal{G}(\mathcal{S}(?, r, t)), \mathbf{none} \rangle$, $\mathbf{s.t.}\ \mathcal{S}(?, r, t) \notin \mathcal{KG}$ |
| **ht→r** | $g_{\mathbf{ht}\rightarrow\mathbf{r}} = \langle \mathcal{G}(h, ?, t), r \rangle$ | $g^-_{\mathbf{ht}\rightarrow\mathbf{r}} = \langle \mathcal{G}(\mathcal{S}(h, ?, t)), \mathbf{none} \rangle$, $\mathbf{s.t.}\ \mathcal{S}(h, ?, t) \notin \mathcal{KG}$ |
| **hrt→** | $g_{\mathbf{hrt}\rightarrow} = \langle \mathcal{G}(h, r, t), \mathbf{true} \rangle$ | $g^-_{\mathbf{hrt}\rightarrow} = \langle \mathcal{G}(\mathcal{S}(h, r, t)), \mathbf{false} \rangle$, $\mathbf{s.t.}\ \mathcal{S}(h, r, t) \notin \mathcal{KG}$ |

Table 1: Instance generation for each learning view. The first/second part in an instance is used as the input/output for classification. The function $\mathcal{G}$ retrieves the sub-graph surrounding the candidate triple with the maximum height and width limitations. The function $\mathcal{S}$ receives a tuple and returns a randomly corrupted tuple that not exists in the KG, by randomly replacing a known component which is not denoted by the question mark. The operator $\in$ incidates that a tuple is *equal to* or *inside of* a triple.

beddings for individual entities and relations based on non-local information, but also obtains representations for sub-graphs resorting to complicated neural encoders. This manner probably brings better KGC performance by leveraging global information more effectively.

## 3 Method: Multi-view Classification

A knowledge graph $\mathcal{KG}$ contains a set of triples indicating facts, $\{(h, r, t)\} \subseteq \mathcal{E} \times \mathcal{R} \times \mathcal{E}$. Each triple $(h, r, t)$ consists of two entities $h$ and $t$ referred to the subject and object of the triple, and a relation $r$ referred to the relationship between the two entities. $\mathcal{E}$ and $\mathcal{R}$ indicates the possible entity set and the possible relation set, respectively. The fundamental problem for KGC is to define a candidate triple evaluation model $f : \mathcal{E} \times \mathcal{R} \times \mathcal{E} \rightarrow \mathbb{R}$, giving each candidate triple $(h, r, t)$ a score indicting the validity of the triple.

### 3.1 Classification Views

We adopt a multi-view classification model for KGC, where a candidate triple is evaluated from four different views. The first three views adopt the generative methodology, each view predicts a specific component of the candidate triple according to the other two components and the nearby triples. The last view adopts the discriminative methodology, it predicts the plausibility of the whole triple according to its components as well as its nearby triples. In the prediction conditions of these views, the components in the candidate triple are content information inside the triple, and the triples nearby the candidate triple are context information outside the triple.

In details, the first view **hr→t** predicts $t$ based on $h$, $r$ and their context, the second view **rt→h** predicts $h$ based on $r$, $t$ and their context, the third view **ht→r** predicts $r$ based on $h$, $t$ and their context, and the fourth view **hrt→** predicts the plausibility given $h$, $r$, $t$ and their context. We denote the view set as $\mathcal{V}$, containing the four views mentions above. These views evaluate the candidate triple from different viewpoints and can be integrated to give better prediction.

In the prediction condition of each view, the context information includes the entities and relations nearby the candidate triple, and excludes the entities and relations that can only be reached by way of the entity or relation to be predicted. The content and context can be jointly represented as the sub-graph surrounding the candidate triple. For each of the first three views, the entity of relation to be predicted is replaced by a specific placeholder. The sub-graph can be extracted by breadth-first traversal from the candidate triple, without passing by the entity or relation to be predicted. In the traversal procedure, two hyperparameters $d$ and $w$ are introduced to restrict the depth and width of the sub-graph. Specifically, $d$ defines the maximum distance between an entity and the candidate triple, and $w$ defines the maximum branch count when passing by an entity.

The sub-graphs can be linearized as sequences of of symbols with paired brackets in specific positions. The linearization facilitates the sequential encoding of graphic structures, which is proved to be effective and efficient in syntactic parsing. Table 1 shows the learning views and Figure 1 shows the content and context information for each view.

### 3.2 Instance Generation

Given a learning view $v \in \mathcal{V}$, we define a pair of instance generation functions, $g_v$ and $g_v^-$, to generate positive and negative classification instances for a candidate triple under this view. The instances are used as classification instances for triple evaluation. In an instance $\langle x, y \rangle$, the source part $x$ is a linearized sequence representing a sub-graph, and the target part $y$ is a label indicating an entity, a

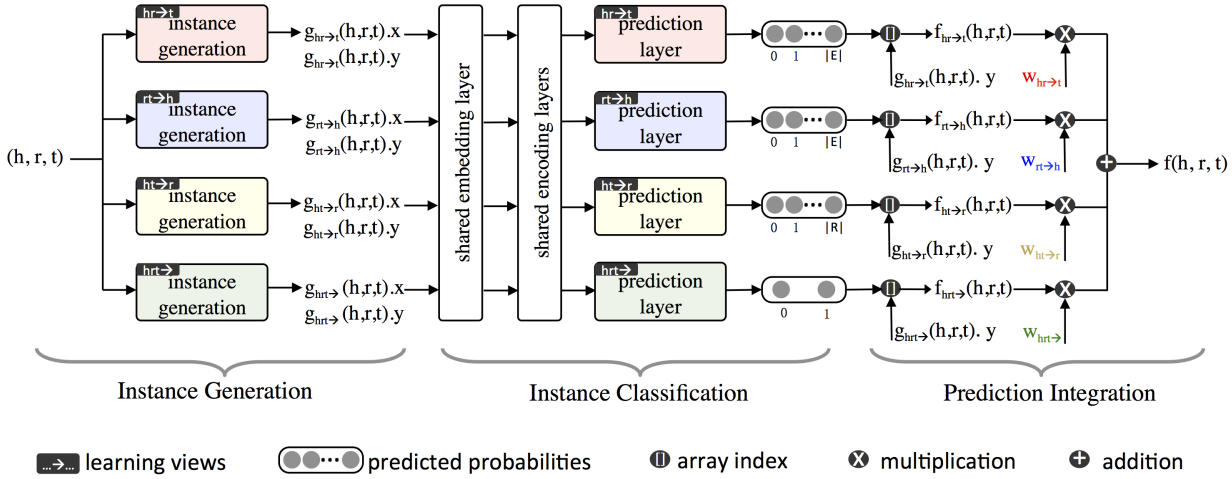



Figure 2: The overall multi-task learning architecture for the multi-view learning model.

relation or a boolean symbol. They correspond to the input and output for the learning of the classification models. For a given triple and a given view $v$, we always generate one positive view instance, but only generate a negative instance with a certain frequency $\rho_v$. The frequencies for the first three views should be much smaller than 1 in order to balance the instances with respect to the classification labels.

The positive instances are generated directly according to the schemas of the views. The negative instances are necessary for the learning of the triple evaluation model especially for the forth view. The source part of a negative instance can be generated by replacing a random component in the tuple with a random symbol of the same type, to satisfy the condition that the changed tuple is not *equal to* or *inside of* a triple in the KG. The target part for a negative instance is **none** for the first three views, and **false** for the fourth view. Table 1 shows the instance generation functions and their instances.

For each learning view, both positive and negative instances generated from the training triples are used for training, while only positive instances generated from the candidate triple are needed for testing. The classification models for the learning views can be trained with separated classifiers or in a multi-task framework. To promote the information sharing and interaction between learning views, we realized the multi-view model in a multi-task learning architecture, where each subtask takes charge of a specific learning view. In the multi-task architecture, the instances for a training or testing triple are simultaneously assigned to the sub-tasks according to their corresponding views. The details for realization will be described in the next section.

### 3.3 Triple Evaluation

Given a candidate triple, four classification instances are generated for the learning views by the corresponding positive instance generation functions. The evaluation given by each learning view is obtained by evaluating the corresponding instance with the corresponding classifier. The evaluation given by the whole multi-view model is the weightedly summation of the evaluations given by these views:

$$f(h,r,t) = \sum_{v \in \mathcal{V}} \mathbf{w}_v f_v(h,r,t)$$

The function $f_v$ and the hyperparameter $\mathbf{w}_v$ indicate the view-specific evaluation function and its weighting coefficient, respectively.

The view-specific evaluation function invokes the classification model of the view with the source part of the instance as input, and returns the prediction score corresponding to the target part of the instance:

$$f_v(h,r,t) = \sum_{v \in \mathcal{V}} \mathbf{w}_v \mathcal{F}_v(g_v^+(h,r,t) \cdot x)[g_v^+(h,r,t) \cdot y]$$

The function $\mathcal{F}$ indicates the classification procedure of the sub-task corresponding to a specific learning view, it takes the source part of the instance as input and gives the prediction scores

on all possible labels. The operator $\cdot$ indexes the source or target part of the instance, and the operator $[\,]$ indexes the score corresponding to the target part.

For each triple in the testing set, we should compare its validity with those of the candidate triples, which are generated by replacing the head or tail entity with another entity. This means that, for a KG with millions of entities, millions of candidate triples should be evaluated by the multi-view model for each testing triple. To promote the efficiency of the multi-view model, we adopt a coarse-to-fine strategy in testing, where the first or second view is performed to give a list of $k$-best candidates, and the overall model is then performed to evaluate these candidates.

## 4 Realization: Multi-task Architecture

We implement the multi-view learning in a multi-task architecture, where each sub-task takes charge of a specific learning view. The multi-task learning strategy enables information sharing and interaction between the sub-tasks, thus leading to better performance.

### 4.1 Overall Pipeline

We design a unified neural multi-task learning architecture for the multi-view model. The overall procedure of the multi-task architecture is shown in Figure 2. The overall procedure is composed of three stages, instance generation, instance classification and prediction integration. The instance generation stage takes as input the given triple, and generates classification instances for all learning views by the instance generation functions. The instance classification stage takes as input the source parts of these instances, and predicts the labels for each input with the corresponding view-specific classification model. The prediction integration stage takes as input the predictions of all the classification models, and computes the overall training cost and evaluation score according to the target parts of the instances. Note that we need not compute the overall evaluation score for training, nor generate the negative instances for testing.

In the instance classification stage, all the classification models follow the same pipeline composed of embedding, encoding and predicting. For predicting, these models adopt separated predicting layers due to their essentially different learning objects. For embedding and encoding, these models adopt the shared layers following the conventional strategy in NLP multi-task learning work. This is reasonable because the relationship between an instance and its components is analogous to that between a sentence and its words. The architecture in Figure 2 shows the multi-task learning architecture with shared embedding and encoding layers.

We add a specific symbol indicating the learning view at the beginning of the source part of the instance. This is similar to the idea in multilingual NMT that a specific markup is added at the beginning of a source language sentence to indicate the target language. The marked source parts of the instances are input into the same encoding layer. According to the added markups, the neural network learns and applies different information propagation regularities for instances of different views, while sharing network parameters as much as possible.

### 4.2 Neural Classifier

We use multi-layer Transformer as the encoding layers and logistic regression with softmax as the classification layers. Given the source part of an instance, $x = (x_1, x_2, ..., x_n)$, which is a sequence of entities and relations with paired brackets indicating an linearized sub-graph, we construct the representation for each element $x_i \in x$ as:

$$\mathbf{h}_i^0 = \mathbf{x}_i^e + \mathbf{x}_i^p$$

where $x_i^e$ is the element embedding and $x_i^p$ the position embedding, indicating the current element and its position in the sequence, respectively. We feed these representations into a stack of $L$ successive Transformer encoders as:

$$\mathbf{h}_i^l = \text{Transformer}(\mathbf{h}_i^{l-1}),\ l = 1, 2, ..., L$$

where $\mathbf{h}_i^l$ is the hidden state of $x_i$ after the $l$-th encoding layer. We omit the detailed description of Transformer since it is already ubiquitous recently.

The representation used for the subsequent classification layer is the concatenation of the final hidden states corresponding to the components of the triple for evaluation. Note that for the first three views, one of the three components is a placeholder. The training procedure aims to find the parameters minimizing the cross-entropy loss:

$$\mathcal{L}(\theta) = \sum_{z \in \mathcal{KG}} \sum_{v \in \mathcal{V}} \sum_{\langle x, y \rangle \in \{g_v^+(z), g_v^-(z)\}} \mathcal{C}(\mathcal{F}_v(x, \theta), y)$$

| Setting | FB15k-237 | | | | | | WN18RR | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Content | | | +Context | | | Content | | | +Context | | |
| | MR | MRR | H@10 | MR | MRR | H@10 | MR | MRR | H@10 | MR | MRR | H@10 |
| $\mathcal{V}$ - hr→t | 161 | .267 | .431 | 209 | .289 | .485 | 2420 | .408 | .477 | 2262 | .412 | .498 |
| $\mathcal{V}$ - rt→h | 155 | .277 | .443 | 178 | .296 | .476 | 3318 | .377 | .437 | 3573 | .393 | .473 |
| $\mathcal{V}$ - ht→r | 150 | .294 | .468 | 215 | .310 | .481 | 2824 | .424 | .491 | 2713 | .462 | .522 |
| $\mathcal{V}$ - hrt→ | 156 | .290 | .475 | 161 | .335 | .492 | 3011 | .421 | .477 | 2713 | .436 | .509 |
| $\mathcal{V}$ | 139 | .330 | .491 | 151 | .359 | .521 | 2193 | .446 | .526 | 2210 | .484 | .540 |

Table 2: The contributions of the individual views to the overall model, evaluated on the development sets.

| | | FB15k-237 | WN18RR |
|---|---|---|---|
| **Statistics** | # entity | 14,541 | 40,943 |
| | # relation | 237 | 11 |
| **Partition** | Train | 272,115 | 86,835 |
| | Develop | 17,535 | 3,034 |
| | Test | 20,466 | 3,134 |

Table 3: The statistics of FB15k-237 and WN18RR, including number of entities, relations, and triples in each partition.

Here, we use $\mathcal{F}$ to indicate the feedforward procedure, $\mathcal{C}$ to indicate the cross-entropy cost function, and $\mathcal{KG}$ to indicate the set of training triples. In the testing procedure, only positive instances are used for a testing triple. The testing procedure evaluates a triple by integrating the four views as mentioned before.

## 5 Experiments

### 5.1 Datasets and Evaluation Protocol

We evaluate the multi-view model on two widely used benchmark datasets, FB15k-237 and WN18RR, which are subsets of two common datasets FB15k and WN18. The original FB15k and WN18 are easy for KGC due to the reversible relations, it could not reflect the real performance of KGC models. Therefore, researchers create FB15k-237 and WN18RR to fix the reversible relation problem, and make the KGC task more realistic (Toutanova and Chen, 2015; Dettmers et al., 2018). The statistics of the datasets are summarized in Table 3.

The purpose of KGC is to predict a missing entity given a relation and another entity. Following Bordes et al. (2013), for every testing triple, we replace the head or tail entities with all entities existed in the knowledge graph, and rank these triples in ascending order according to the triple evaluation function, following the *filtered* setting protocol which does not consider any corrupted triples that appear in the original KG. Following (Nguyen, 2017), we use three common evaluation metrics, mean rank (MR), mean reciprocal rank (MRR), and the proportion of the valid test triples ranking in top $n$ predictions (H@n) with $n \in \{1, 3, 10\}$.

### 5.2 Details for Training and Testing

The multi-view model is trained with instances generated from the training triples, and is used to evaluate the instances generated from the testing triples. There are parameters to be tuned in the procedures of instance generation, model training, and model testing.

For the definition of the subgraph indicating the content and context of a triple, the maximum depth $d$ and width $w$ will be determined in the developing procedure. For the transformer used for classification, the number of Transformer blocks is $L = 6$, the number of self-attention heads is $A = 4$, and the hidden size and the feed-forward size are $D = 256$ and $2D = 512$, respectively. The dropout strategy is applied on embedding and encoding layers with dropout rate $0.5$. We adopt the Adam algorithm (Kingma and Ba, 2014) for tuning with a learning rate $\eta = 5 \times 10^{-4}$. The multi-view model is trained with batch size $B = 256$ for at most $1000$ epochs. For the coarse-to-fine prediction strategy in the testing procedure, the number $k$ of best candidates given by the first or second view is determined on the development set. We choose $\rho = [0.001, 0.001, 0.01, 1.0]$ for negative instance generation, $d = 2$ and $w = 3$ for sub-graph retrieval, and $\mathbf{w} = [0.30, 0.30, 0.25, 0.15]$ for view combination by grid search experiments on development sets. The above models are implemented on PaddlePaddle[1].

[1] https://github.com/PaddlePaddle/Paddle

| Model | FB15k-237 | | | | | WN18RR | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | MR | MRR | H@1 | H@3 | H@10 | MR | MRR | H@1 | H@3 | H@10 |
| R-GCN+ | - | .249 | .151 | .264 | .417 | - | - | - | - | - |
| KB-LRN | 209 | .309 | .219 | - | .493 | - | - | - | - | - |
| ConvE | 246 | .316 | .239 | .350 | .491 | 5277 | .460 | .390 | .430 | .480 |
| ConvR | - | .350 | .261 | .385 | .528 | - | .475 | .443 | .489 | .537 |
| RotatE | 177 | .338 | .241 | .375 | .533 | 3340 | .476 | .428 | .492 | .571 |
| TuckER | - | .358 | .266 | .394 | .544 | - | .470 | .443 | .482 | .526 |
| pLogicNet | 173 | .332 | .237 | .367 | .524 | 3408 | .441 | .398 | .446 | .537 |
| SimpleClassification | 161 | .307 | .223 | .382 | .525 | 2193 | .446 | .393 | .456 | .522 |
| MultiView | **134** | .320 | **.276** | **.412** | **.544** | **1738** | .463 | **.462** | **.494** | .549 |

Table 4: Performance of multi-view learning compared with previous methods, on the testing sets of FB15k-237 and WN18RR. R-GCN+: (Schlichtkrull et al., 2017), KB-LRN: (Garcia-Duran and Niepert, 2017), ConvE: (Dettmers et al., 2018), ConvR: (Jiang et al., 2019), RotatE: (Sun et al., 2019), TuckER: (Balažević et al., 2019), pLogicNet: (Qu and Tang, 2019). SimpleClassification: multi-view model based on simple classification (**hr→t and rt→h**), MultiView: multi-view model with all components (**hr→t + rt→h + ht→r + hrt→**) .

### 5.3 Main Results and Analysis

We verify the effectiveness of the multi-view model, by investigating the contributions of the learning views to the overall model. Table 2 shows the performance on the development sets of the two datasets. Note that for each experimental setting, the model is retrained on the classification instances generated according to the views in the setting. We find that each of the learning views contributes to the final performance, and context information brings further improvement.

The performance of the multi-view model on the testing sets of the two datasets is shown in Table 4, where the performance of methods in previous work is also listed. The multi-view learning model achieves the new state-of-the-art on both benchmark datasets. Compared with previous work, it gives significantly better MR on both datasets. It reveals that in the multi-view model, the answers are high in the ranked lists on average. Considering that it does not use any optimization tricks, we think that it still has potential for further improvement by intruding additional information and resources, such as pre-trained embeddings, text descriptions and surface morphologies of entities and relations. We also find that, the simple classification model based on the first two views, which brutally predict the head and tail entities according to the rest components, achieves very promising results. In other words, the first two views lead to simple but effective classification-based KGC models.

The simple classification model works very fast in evaluation of candidate triples, since direct prediction of the missing entities is equivalent to evaluating thousands of candidate triples simultaneously. We can adopt a coarse-to-fine strategy in realistic applications. It pre-selects the $k$-best candidates by the first two views, and reranks the candidates by the whole multi-view model. Figure 3 shows the experimental results. The quality of the candidate list is measured with recall, indicating the percentage of the instances for which the candidate lists contain the answers. We find that the pre-selection of 2000-best list achieves very high recalls on the two datasets, especially on FB15k-237. Therefore, we can safely filter out most of the candidates with little loss of final precision. It facilitates the introduction of more features in the multi-view model by restricting the search space to a small but precise $k$-best list.

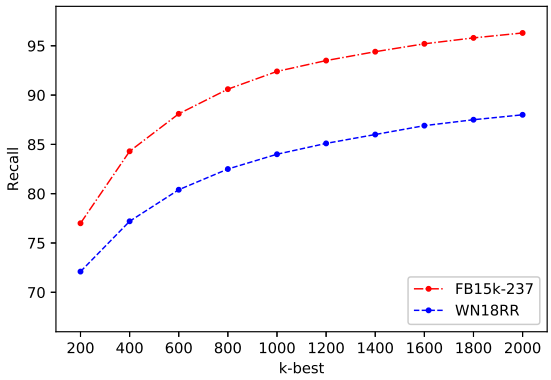



Figure 3: The recall curves of $k$-best pre-filtering.

## 6 Conclusion

We propose a novel multi-view classification model for knowledge graph completion, where multiple classification views are performed based on both content and context information for candidate triple evaluation. The multi-view model is implemented with a simple and unified multi-task learning architecture where the parameters are shared across all the learning views. It achieves the new state-of-the-art although without using any optimization tricks. The multi-view model can be improve from two perspectives in the future. First, the multi-view model can leverage more kinds of information and resources for better performance, such as the descriptions of the entities and relations, as well as related information in external knowledge bases. Second, the multi-task learning architecture can introduce different kinds of neural networks to better model different kinds of information, for example, sequential neural networks for sequences and graph neural networks for graphs.

## Acknowledgments

The authors Chen and Xu were also supported by the National Nature Science Foundation of China (No. 61876198, 61976015, 61370130 and 61473294), the Fundamental Research Funds for the Central Universities (No. 2018YJS025), the Beijing Municipal Natural Science Foundation (No. 4172047), and the International Science and Technology Cooperation Program of China under Grant No. K11F100010. We sincerely thank Quan Wang in Baidu for the enlightening suggestions in the research procedure, and the anonymous reviewers for their valuable comments and suggestions.

## References

Ivana Balažević, Carl Allen, and Timothy M. Hospedales. 2019. TuckER: Tensor factorization for knowledge graph completion. In *Proceedings of EMNLP-IJCNLP*.

Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. 2008. Freebase: A collaboratively created graph database for structuring human knowledge. In *Proceedings of ICMD*.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. *Proceedings of NIPS*.

Michael Cochez, Petar Ristoski, Simone Paolo Ponzetto, and Heiko Paulheim. 2017. Global rdf vector space embeddings. In *Proceedings of ISWC*.

Rajarshi Das, Arvind Neelakantan, David Belanger, and Andrew McCallum. 2017. Chains of reasoning over entities, relations, and text using recurrent neural networks. In *Proceedings of EACL*.

Tim Dettmers, Pasquale Minervini, Pontus Stenetorp, and Sebastian Riedel. 2018. Convolutional 2d knowledge graph embeddings. In *Proceedings of AAAI*.

Jun Feng, Minlie Huang, Yang Yang, and Xiaoyan Zhu. 2016. Gake: Graph aware knowledge embedding. In *Proceedings of COLING*.

Alberto Garcia-Duran and Mathias Niepert. 2017. Kblrn: End-to-end learning of knowledge base representations with latent, relational, and numerical features. In *arXiv preprint abs/1709.04676*.

Aditya Grover and Jure Leskovec. 2016. node2vec: Scalable feature learning for networks. In *Proceedings of SIGKDD*.

Yanchao Hao, Yuanzhe Zhang, Kang Liu, Shizhu He, Zhanyi Liu Hua Wu, and Jun Zhao. 2017. An end-to-end model for question answering over knowledge base with cross-attention combining global knowledge. In *Proceedings of ACL*.

Guoliang Ji, Shizhu He, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Knowledge graph embedding via dynamic mapping matrix. In *Proceedings of ACL-IJCNLP*.

Xiaotian Jiang, Quan Wang, and Bin Wang. 2019. Adaptive convolution for multi-relational learning. In *Proceedings of NAACL-HLT*.

Gjergji Kasneci, Fabian M Suchanek, Georgiana Ifrim, Maya Ramanath, and Gerhard Weikum. 2008. Naga: Searching and ranking knowledge. *Proceedings of ICDE*.

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

Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Soren Auer, and et al. 2015. Dbpedia-a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic Web*.

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 EMNLP*.

Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015b. Learning entity and relation embeddings for knowledge graph completion. In *Proceedings of AAAI*.

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

Dat Quoc Nguyen, Kairit Sirts, Lizhen Qu, and Mark Johnson. 2016. Stranse: a novel embedding model of entities and relationships in knowledge bases. In *Proceedings of NAACL-HLT*, pages 460–466.

Maximilian Nickel, Lorenzo Rosasco, and Tomaso Poggio. 2016. Holographic embeddings of knowledge graphs. In *Proceedings of AAAI*.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. 2014. Deepwalk: Online learning of social representations. In *Proceedings of SIGKDD*.

Meng Qu and Jian Tang. 2019. Probabilistic logic neural networks for reasoning. *arXiv:1906.08495*.

Petar Ristoski and Heiko Paulheim. 2016. Rdf2vec: Rdf graph embeddings for data mining. In *Proceedings of ISWC*.

Michael Schlichtkrull, Thomas Kipf, Peter Bloem, Ivan Titov Rianne van den Berg, and Max Welling. 2017. Modeling relational data with graph convolutional networks. In *arXiv:1703.06103*.

Michael Schuhmacher and Simone Paolo Ponzetto. 2014. Knowledge-based graph document modeling. In *Proceedings of WSDM*, pages 543–552.

Richard Socher, Danqi Chen, Christopher D Manning, and Andrew Ng. 2013. Reasoning with neural tensor networks for knowledge base completion. *NIPS*, pages 926–934.

Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge. *Proceedings of WWW*, pages 697–706.

Zhiqing Sun, Zhi-Hong Deng, Jian-Yun Nie, and Jian Tang. 2019. Rotate: Knowledge graph embedding by relational rotation in complex space. *arXiv:1902.10197*.

Kristina Toutanova and Danqi Chen. 2015. Observed versus latent features for knowledge base and text inference. In *Proceedings of the ACL Workshop on Continuous Vector Space Models and their Compositionality*, pages 57–66.

Theo Trouillon, JohannesWelbl, Sebastian Riedel, Eric Gaussier, and Guillaume Bouchard. 2016. Complex embeddings for simple link prediction. *Proceedings of ICML*, pages 2071–2080.

Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge graph embedding by translating on hyperplanes. In *Proceedings of AAAI*, 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 WWW*, pages 515–526.

Chenyan Xiong, Russell Power, and Jamie Callan. 2017. Explicit semantic ranking for academic search via knowledge graph embedding. *Proceedings of WWW*, pages 1271–1279.

Bishan Yang and Tom Mitchell. 2017. Leveraging knowledge bases in lstms for improving machine reading. In *Proceedings of ACL*.

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 ICLR*.

Yuanzhe Zhang, Kang Liu, Shizhu He, Guoliang Ji, Zhanyi Liu, Hua Wu, and Jun Zhao. 2016. Question answering over knowledge base with neural attention combining global knowledge information. *arXiv:1606.00979*.