Table 2: The results of *AUC* and *F1* in CTR prediction.

Model	MovieLens-20M		Book-Crossing		Last.FM	
	AUC	*F1*	*AUC*	*F1*	*AUC*	*F1*
SVD	0.963 (-1.5%)	0.919 (-1.4%)	0.672 (-8.9%)	0.635 (-7.7%)	0.769 (-3.4%)	0.696 (-3.5%)
LibFM	0.959 (-1.9%)	0.906 (-2.8%)	0.691 (-6.4%)	0.618 (-10.2%)	0.778 (-2.3%)	0.710 (-1.5%)
LibFM + TransE	0.966 (-1.2%)	0.917 (-1.6%)	0.698 (-5.4%)	0.622 (-9.6%)	0.777 (-2.4%)	0.709 (-1.7%)
PER	0.832 (-14.9%)	0.788 (-15.5%)	0.617 (-16.4%)	0.562 (-18.3%)	0.633 (-20.5%)	0.596 (-17.3%)
CKE	0.924 (-5.5%)	0.871 (-6.5%)	0.677 (-8.3%)	0.611 (-11.2%)	0.744 (-6.5%)	0.673 (-6.7%)
RippleNet	0.968 (-1.0%)	0.912 (-2.1%)	0.715 (-3.1%)	0.650 (-5.5%)	0.780 (-2.0%)	0.702 (-2.6%)
KGCN-sum	**0.978**	**0.932***	**0.738**	**0.688***	0.794 (-0.3%)	0.719 (-0.3%)
KGCN-concat	0.977 (-0.1%)	0.931 (-0.1%)	0.734 (-0.5%)	0.681 (-1.0%)	**0.796***	**0.721***
KGCN-neighbor	0.977 (-0.1%)	**0.932***	0.728 (-1.4%)	0.679 (-1.3%)	0.781 (-1.9%)	0.699 (-3.1%)
KGCN-avg	0.975 (-0.3%)	0.929 (-0.3%)	0.722 (-2.2%)	0.682 (-0.9%)	0.774 (-2.8%)	0.692 (-4.0%)

* Statistically significant improvement by unpaired two-sample t-test with $p = 0.1$.

- **LibFM + TransE** extends LibFM by attaching an entity representation learned by TransE [1] to each user-item pair.
- **PER** [22] treats the KG as heterogeneous information networks and extracts meta-path based features to represent the connectivity between users and items.
- **CKE** [23] combines CF with structural, textual, and visual knowledge in a unified framework for recommendation. We implement CKE as CF plus a structural knowledge module in this paper.
- **RippleNet** [18] is a memory-network-like approach that propagates users' preferences on the KG for recommendation.

4.3 Experiments Setup

In KGCN, we set functions g and f as inner product, σ as *ReLU* for non-last-layer aggregator and *tanh* for last-layer aggregator. Other hyper-parameter settings are provided in Table 1. The hyper-parameters are determined by optimizing *AUC* on a validation set. For each dataset, the ratio of training, evaluation, and test set is $6:2:2$. Each experiment is repeated 3 times, and the average performance is reported. We evaluate our method in two experiment scenarios: (1) In click-through rate (CTR) prediction, we apply the trained model to predict each interaction in the test set. We use *AUC* and *F1* to evaluate CTR prediction. (2) In top-K recommendation, we use the trained model to select K items with highest predicted click probability for each user in the test set, and choose *Recall@K* to evaluate the recommended sets. All trainable parameters are optimized by Adam algorithm. The code of KGCN-LS is implemented under Python 3.6, TensorFlow 1.12.0, and NumPy 1.14.3.

The hyper-parameter settings for baselines are as follows. For SVD, we use the unbiased version (i.e., the predicted rating is modeled as $r_{pq} = \mathbf{p}^\top \mathbf{q}$). The dimension and learning rate for the four datasets are set as: $d = 8$, $\eta = 0.5$ for MovieLens-20M, Book-Crossing; $d = 8$, $\eta = 0.1$ for Last.FM. For LibFM, the dimension is $\{1, 1, 8\}$ and the number of training epochs is 50. The dimension of TransE is 32. For PER, we use manually designed user-item-attribute-item paths as features (i.e., "user-movie-director-movie", "user-movie-genre-movie", and "user-movie-star-movie" for MovieLens-20M; "user-book-author-book" and "user-book-genre-book" for Book-Crossing, "user-musician-date_of_birth-musician" (date of birth is discretized), "user-musician-country-musician", and "user-musician-genre-musician" for Last.FM). For CKE, the dimension of the three datasets are 64, 128, 64. The training weight for KG part is 0.1 for all datasets. The learning rate are the same as in SVD. For RippleNet, $d = 8$, $H = 2$, $\lambda_1 = 10^{-6}$, $\lambda_2 = 0.01$, $\eta = 0.01$ for MovieLens-20M; $d = 16$, $H = 3$, $\lambda_1 = 10^{-5}$, $\lambda_2 = 0.02$, $\eta = 0.005$ for Last.FM. Other hyper-parameters are the same as reported in their original papers or as default in their codes.

4.4 Results

The results of CTR prediction and top-K recommendation are presented in Table 2 and Figure 2, respectively (SVD, LibFM and other variants of KGCN are not plotted in Figure 2 for clarity). We have the following observations:

- In general, we find that the improvements of KGCN on book and music are higher than movie. This demonstrates that KGCN can well address sparse scenarios, since Book-Crossing and Last.FM are much sparser than MovieLens-20M.
- The performance of KG-free baselines, SVD and LibFM, are actually better than the two KG-aware baselines PER and CKE, which indicates that PER and CKE cannot make full use of the KG with manually designed meta-paths and TransR-like regularization.
- LibFM + TransE is better than LibFM in most cases, which demonstrates that the introduction of KG is helpful for recommendation in general.
- PER performs worst among all baselines, since it is hard to define optimal meta-paths in reality.
- RippleNet shows strong performance compared with other baselines. Note that RippleNet also uses multi-hop neighborhood structure, which interestingly shows that capturing proximity information in the KG is essential for recommendation.

The last four rows in Table 2 summarize the performance of KGCN variants. The first three (sum, concat, neighbor) correspond to different aggregators introduced in the preceding section, while the last variant KGCN-avg is a reduced case of KGCN-sum where