Maximum Common Subgraph Guided Graph Retrieval: Late and Early Interaction Networks
(Appendix)

A Potential limitations of our work

(1) In this paper, we have considered only the structural information encoded in the graphs. Consequently, all the nodes and edges are assigned the same label. However, in practice the graph nodes and edges may contain rich features, which must be taken into account when computing MCS scores. For example, certain applications may prohibit matching of graph components with different labels, which would potentially disallow many of the alignments currently proposed by our model. Additionally, in many real world applications involving knowledge graphs, we observe hierarchical relationships amongst entity types, which can further constraint the space of possible alignments. While our current formulations do allow for the encoding of node and edge features, this is insufficient as is to ensure such constraint-based matching.

(2) We consider MCS between two graphs. Some applications in chemical science [69] involve the computation of maximum common subgraph between three or more graphs.

B Neural parameterizations of GNN and Sinkhorn

We have already described the GNN network GNN_θ which was used for early interaction model (Section 3). Here, we describe the neural architecture of GNN_θ used in our late interaction models (Section 2) and GS_ϕ modules, used in both our late and early interaction models.

B.1 Neural parameterization of GNN_θ

Our GNN GNN_θ module is used in both LMCES and LMCCS. Given a graph $G = (V, E)$ (G can be either a query G_q and G_c), it uses $R = 5$ recurrent propagation layers to encode the node and edge embeddings. It consists of three modules, as follows.

(1) $\textsc{FeatureEncoder}_\theta$: This converts the input node features $\boldsymbol{z}_u$ into the initial node embeddings $\boldsymbol{h}_u$. In this work, we have focused solely on the structural aspects of MCS scoring. Hence, all nodes are assigned the same initial feature $\boldsymbol{z}_u = [1]$ (a single-element vector with a 1 — we want it to be oblivious to local features). The initial features are mapped to a $\text{dim}(\boldsymbol{h}_u(0)) = 10$ dimensional embedding vector using a single $\mathbb{R}^{1\times 10}$ linear layer.

$$\boldsymbol{h}_u(0) = \textsc{FeatureEncoder}_\theta(\boldsymbol{z}_u) \quad \forall\, u \in V \tag{19}$$

Having computed the initial embedding $\boldsymbol{h}_u(0)$, it computes the embeddings $\{\boldsymbol{h}_u(r) \,|\, r \in [R]\}$ described as follows:

(2) $\textsc{MessagePassing}_\theta$: Given a pair of node embedding vectors $\boldsymbol{h}_u(r), \boldsymbol{h}_v(r)$ as input, this generates a message vector $\boldsymbol{m}_{uv}(r)$ of dimension 20 using a linear layer. A simple sum aggregation is used on the incoming messages to any node, to obtain $\overline{\boldsymbol{m}}_{uv}(r)$.

$$\boldsymbol{m}_{uv}(r) = \textsc{MessagePassing}_\theta(\boldsymbol{h}_u(r), \boldsymbol{h}_v(r)) \quad \forall\, (u,v) \in E \tag{20}$$

$$\overline{\boldsymbol{m}}_u(r) = \textstyle\sum_{v \in \text{nbr}(u)} \boldsymbol{m}_{uv}(r) \quad \forall u \in V \tag{21}$$

(3) $\textsc{UpdateEmbedding}_\theta$: This uses the aggregated incoming messages $\overline{\boldsymbol{m}}$, to update the current node embedding using a Gated Recurrent Unit (GRU), as proposed in [70]. Here, the current node embeddings are treated as the hidden context of the GRU, which are updated using input $\overline{\boldsymbol{m}}$.

$$\boldsymbol{h}_u(r+1) = \textsc{UpdateEmbedding}_\theta(\boldsymbol{h}_u(r), \overline{\boldsymbol{m}}_u(r)) \quad \forall\, u \in V \tag{22}$$

Furthermore, in our early cross interaction model XMCS, the cross graph signal $\boldsymbol{\Delta}$ is concatenated to the message aggregation $\overline{\boldsymbol{m}}$, while being provided as GRU input. Given this framework, we obtain the node embedding matrices $\boldsymbol{H}_\bullet(r)$, used in LMCES, by gathering all the node embeddings $\boldsymbol{h}_u(r) \forall\, u \in V$. Similarly, we obtain the edge embedding matrices $\boldsymbol{M}_\bullet(R)$, used in LMCCS, by gathering the message vectors $\boldsymbol{m}_{uv}(R) \forall\, (u,v) \in E$.

B.2 Neural parameterization of GS_ϕ

The Gumbel-Sinkhorn network is used to generate soft-permutation matrices (or doubly stochastic matrices) based on some input matrix. (One may regard the input as analogous to logits and the output as analogous to a softmax.) In principle, the input matrix is driven by some neural network and then it is fed into an iterative GUMBELSINKHORN operator. Given a matrix $\boldsymbol{U}$, $\textsc{GumbelSinkhorn}(\cdot)$ returns a soft-permutation matrix using following differentiable iterative process:

$$\textsc{GumbelSinkhorn}^0(\boldsymbol{U}) = \exp(\boldsymbol{U}/\zeta) \tag{23}$$

$$\textsc{GumbelSinkhorn}^{(t+1)}(\boldsymbol{U}) = \mathrm{ColDivide}\Big(\mathrm{RowDivide}\big(\textsc{GumbelSinkhorn}^t(\boldsymbol{U})\big)\Big) \tag{24}$$

$$\textsc{GumbelSinkhorn}(\boldsymbol{U}) = \lim_{t\to\infty} \textsc{GumbelSinkhorn}^t(\boldsymbol{U}) \tag{25}$$

ColDivide and RowDivide depict the iterative (row and column) normalization across columns and rows, *i.e.*, $[\mathrm{ColDivide}(\mathbf{X})]_{i,j} = X_{i,j}/\sum_k X_{i,k}$ and $[\mathrm{RowDivide}(\mathbf{X})]_{i,j} = X_{i,j}/\sum_k X_{k,j}$. The final output in Eq. (25) is also given by the solution of the following optimization problem:

$$\textsc{GumbelSinkhorn}(\boldsymbol{U}) = \operatorname*{argmax}_{\boldsymbol{P}\in\mathcal{B}} \langle \boldsymbol{P}, \boldsymbol{U}\rangle - \zeta \sum_{i,j} P_{i,j} \log P_{i,j} \tag{26}$$

where $\mathcal{B}$ is the set of doubly stochastic matrices lying in a Birkhoff polytope and ζ is the temperature parameter. As $\zeta \to 0$, one can show that $\textsc{GumbelSinkhorn}(C)$ approaches a hard-permutation matrix.

We use $\mathrm{GS}_\phi(\boldsymbol{H}_q(r), \boldsymbol{H}_c(r)) = \textsc{GumbelSinkhorn}(FF_\phi(\boldsymbol{H}_q(r))FF_\phi(\boldsymbol{H}_c(r))^T)$ in both our late (Eqs. (4), (9)) and early interaction (Eq. (9)) models. Here, FF_ϕ is linear-ReLU-linear network which consists of a 10×16 linear layer, a ReLU activation function, and a final 16×10 linear layer. We perform 20 iterations of row and column normalizations, with a temperature of $\zeta = 0.1$.

C Additional details about experimental setup

In this section, we provide further details about the experimental setup, including the baseline models, hyperparameters, datasets and computing resources.

C.1 Dataset generation

We obtain our seven datasets from the repository of graphs maintained by TUDatasets [71], for the purpose of benchmarking GNNs. Amongst them, MSRC is a computer vision dataset, DD is a Bioinformatics dataset, and the remaining are Small Molecules datasets.

We sample the corpus graphs G_c and some seed query graphs G'_q independently of each other using the Breadth First Search (BFS) sampling strategy used in previous works [72, 11]. To sample G_c or G'_q, we first randomly choose a starting node u in a graph present in the dataset, drawn uniformly at random. We implement a randomized BFS traversal to obtain node sets of size $|V| \in [10, 15]$ in the vicinity of the starting node u. Finally, the subgraph induced by this set of nodes, gives us the sampled seed query or corpus graph. Using this process we generate first generate 800 , corpus graphs. Subsequently, we sample 500 seed query graphs under the constraint that it should be subgraph isomorphic to a fraction η of the available set of corpus graphs. We set $\eta \in [0.1, 0.4]$ similar to prior works [72, 11]. Here we used the Networkx implementation of VF2 algorithm [73] for determining subgraph isomorphism. Subsequently, we augment the seed query graph G'_q, with randomly connected nodes and edges, which gives us the final query graph G_q for MCS computation. This procedure ensures that we have a significant variation in MCS sizes across the set of corpus graphs, which is a desirable condition for any retrieval setup.

The corresponding ground truth MCS values for each of the 400000 query-corpus pairs, are generated using the combinatorial formulations for exact MCCS and MCES, as described in Section 2.1. We split the set of 500 query graphs into 60% training, 20% test and 20% validation splits.

C.2 Baseline Implementations

We use the available PyTorch implementations of all the baselines, *viz.*, SimGNN[1], GraphSim[2], GOTSim[3], NeuroMatch[4], GEN[5] and GMN[6]. In order to ensure a fair comparison between our models and the baselines, we have ensured the following:

1. Across all models, the node embedding dimension is fixed to 10.
2. The exact same dataset is fed as input to all models, and the same early stopping mechanism used for best model selection.
3. NeuroMatch uses anchor node annotations, which are excluded so as to ensure fair comparison with other models.
4. GEN and GMN compute the Euclidean distance between the query and corpus graph vectors, which is incompatible with MCS scoring. Therefore, we add an additional linear layer on top of their outputs, so as to help them better predict the MCS ground truths.

Furthermore, SimGNN, GraphSim and GOTSim implement their neural scoring function on one graph pair at a time. This is prohibitively slow during training. Therefore, similar to previous work [11], we implement a batched version of the available implementations of these three baselines, so as to achieve the requisite speedup.

C.3 Hyperparameter details

The node embedding size is specified as 10 across all models. During training our model and all the baselines model use early stopping with patience parameter $n_p = 50$. This means that if the Mean Squared Error (MSE) loss on the validation set does not decrease for 50 epochs, then the training process is terminated and the model with the least validation MSE is returned. In all cases, we train with a batch size of 128, using an Adam optimizer with learning rate 10^{-3} and weight decay 5×10^{-4}.

For LMCCS computation as mentioned in Eq. (10), we tune the model across a range of temperature values $\lambda \in [0.05, 50]$ using cross-validation on the validation set. For each dataset, the temperatures resulting in the best performance, are reported in Tables 6.

MCCS	MM	MR	FM	FR	DD	COX2	MSRC
λ	0.7	0.1	0.8	1.4	10	1.1	1

Table 6: Values of best temperature λ for each dataset.

C.4 Evaluation Metrics

Given corpus graphs $C = \{G_c\}$, query graphs $Q = \{G_q\}$ and their gold MCES and MCCS values $\{y_{\text{MCES}}(G_q, G_c)\}$ and $\{y_{\text{MCCS}}(G_q, G_c)\}$. For each query graph, G_q, we compute three evaluation metrics based on the model predictions $\{s_\Lambda(G_q, G_c)\}$ with Λ being the set of trainable parameters and the ground truth MCS scores $\{y(G_q, G_c)\}$ (y can be either y_{MCES} or y_{MCCS}).

Mean Square Error (MSE): It evaluates how close the model predictions are to the ground truth, and a lower MSE value indicates a better performing model.

$$\text{MSE} = \frac{1}{|Q|} \sum_{G_q \in Q} \frac{1}{|C|} \sum_{G_c \in C} (y(G_q, G_c) - s_\Lambda(G_q, G_c))^2 \tag{27}$$

Kendall-Tau correlation (Ktau) [74]: Here, we track the number of concordant pairs N_q^+ where the model and ground truth rankings agree, and the number of discordant pairs N_q^- where they disagree. A better performing model will have a larger number of concordant predictions, which

---

[1] `https://github.com/benedekrozemberczki/SimGNN`

[2] `https://github.com/khoadoan/GraphOTSim`

[3] `https://github.com/khoadoan/GraphOTSim`

[4] `https://github.com/snap-stanford/neural-subgraph-learning-GNN`

[5] `https://github.com/Lin-Yijie/Graph-Matching-Networks`

[6] `https://github.com/Lin-Yijie/Graph-Matching-Networks`

	MCES	MSE (lower is better)						
		MSRC	MM	FR	MR	FM	COX	DD
Late	SimGNN	0.910±0.044	0.302±0.009	0.355±0.021	0.337±0.015	0.331±0.014	0.281±0.012	1.210±0.073
	GraphSim	0.629±0.028	0.274±0.010	0.282±0.012	0.274±0.010	0.261±0.009	0.249±0.009	0.881±0.081
	GOTSim	0.496±0.020	0.343±0.046	0.326±0.020	0.320±0.031	0.359±0.047	0.328±0.015	0.628±0.037
	NeuroMatch	0.582±0.082	0.308±0.059	0.282±0.055	0.795±0.606	0.604±0.322	0.269±0.072	2.827±2.281
	IsoNet	0.276±0.007	0.225±0.009	0.220±0.008	0.209±0.007	0.253±0.017	0.182±0.007	0.333±0.059
	GEN	0.426±0.020	0.311±0.017	0.273±0.012	0.284±0.013	0.324±0.023	0.277±0.021	0.568±0.110
	LMCES	0.232±0.008	0.167±0.005	0.170±0.005	0.162±0.005	0.163±0.004	0.140±0.004	0.223±0.006
Early	GMN	0.269±0.006	0.184±0.004	0.181±0.005	0.178±0.004	0.189±0.005	0.155±0.006	0.273±0.008
	XMCS	0.226±0.007	0.154±0.003	0.162±0.005	0.154±0.004	0.160±0.003	0.132±0.004	0.220±0.005

	MCES	MSE (lower is better)						
		MSRC	MM	FR	MR	FM	COX	DD
Late	SimGNN	0.100±0.020	0.360±0.032	0.337±0.036	0.233±0.026	0.316±0.033	0.289±0.021	0.136±0.023
	GraphSim	0.088±0.018	0.283±0.023	0.290±0.029	0.221±0.019	0.255±0.024	0.325±0.027	0.123±0.020
	GOTSim	0.165±0.025	0.416±0.044	0.340±0.034	0.330±0.034	0.321±0.028	0.318±0.023	0.202±0.027
	NeuroMatch	0.352±0.088	0.376±0.052	0.326±0.042	0.351±0.191	0.295±0.085	0.984±0.689	0.624±0.205
	IsoNet	0.086±0.015	0.237±0.016	0.244±0.019	0.191±0.017	0.218±0.018	0.253±0.018	0.124±0.019
	GEN	0.171±0.025	0.366±0.029	0.344±0.033	0.290±0.030	0.356±0.030	0.309±0.022	0.197±0.025
	LMCCS	0.068±0.012	0.174±0.015	0.179±0.016	0.134±0.012	0.173±0.017	0.177±0.012	0.097±0.016
Early	GMN	0.101±0.015	0.200±0.015	0.216±0.020	0.156±0.015	0.193±0.018	0.176±0.011	0.137±0.016
	XMCS	0.071±0.014	0.168±0.014	0.163±0.018	0.131±0.013	0.168±0.017	0.153±0.009	0.102±0.017

Table 7: Performance measured using **mean square error (MSE)** with standard error, of our models and state-of-the-art baselines on 20% test set, for all seven datasets. Top-half and bottom-half report results for MCES and MCCS respectively. Numbers in green (blue) indicate the best performers among early (late) interaction models. Numbers in red1 (yellow) indicate second best performers for early (late) interaction models.

results in a higher correlation score.

$$\text{KTau} = \frac{1}{|Q|} \sum_{G_q \in Q} \frac{N_q^+ - N_q^-}{\binom{|C|}{2}} \tag{28}$$

In practice we use scipy implementation of KTau to report all the numbers in our paper.

Pairwise Ranking Reward (PairRank): Here, we track the number of concordant pairs, and normalize by the maximum number of possible concordant ranking $N_{q,\max}^+$ in an ideal case. Hence, a higher value of indicates a more accurate model, with a maximum achievable value of 1. The final reported values are computed, across the set of query graphs $Q = \{G_q\}$, as follows:

$$\text{PairRank} = \frac{1}{|Q|} \sum_{G_q \in Q} \frac{N_q^+}{N_{q,\max}^+} \tag{29}$$

Furthermore, for each of the evaluation metrics, along with the average across query graphs, we also report the standard error.

C.5 Hardware and Software details

We implement our models using Python 3.8.5 and PyTorch 1.10.2. Training of our models and the baselines, was performed across servers containing Xeon E5-2620 2.10GHz CPUs, Nvidia Titan Xp-12 GB GPUs, Nvidia T4-16 GB GPUs, and Nvidia Quadro RTX 6000-48 GB GPUs. Running times are compared on the same GPU, averaged over 10 runs of each method.

C.6 License details

SimGNN repository is available under GNU license, while GEN, GMN and GOTSim are available under MIT license.

D Additional experiments

D.1 Results on comparison with SOTA methods along with standard error

In Table 1 in the main submission, we neither reported the result on DD dataset nor the standard error due to space constraints. In Tables 7– 8, we report results with standard error on all seven datsets. Here, the standard error is computed over the variation across all test queries. Moreover, in Table 9, we also report results of PairRank metric. They reveal similar observations as in Table 1.

	MCES	KTau (higher is better)						
		MSRC	MM	FR	MR	FM	COX	DD
Late	SimGNN	0.232±0.003	0.368±0.007	0.358±0.007	0.354±0.007	0.372±0.005	0.394±0.011	0.215±0.004
	GraphSim	0.461±0.004	0.432±0.008	0.458±0.006	0.454±0.005	0.500±0.006	0.403±0.009	0.570±0.003
	GOTSim	0.564±0.004	0.464±0.009	0.448±0.007	0.516±0.006	0.496±0.006	0.374±0.013	0.608±0.003
	NeuroMatch	0.632±0.005	0.488±0.010	0.516±0.007	0.548±0.008	0.535±0.007	0.514±0.011	0.667±0.008
	IsoNet	0.669±0.004	0.506±0.010	0.504±0.008	0.537±0.007	0.532±0.007	0.522±0.012	0.698±0.005
	GEN	0.627±0.005	0.416±0.013	0.468±0.011	0.456±0.012	0.456±0.015	0.466±0.014	0.635±0.010
	LMCES	0.691±0.004	0.577±0.006	0.588±0.006	0.598±0.005	0.610±0.004	0.574±0.009	0.724±0.004
Early	GMN	0.670±0.003	0.544±0.006	0.567±0.007	0.568±0.006	0.569±0.006	0.555±0.009	0.701±0.004
	XMCS	0.699±0.004	0.582±0.006	0.594±0.006	0.612±0.005	0.606±0.005	0.580±0.009	0.724±0.004

	MCES	KTau (higher is better)						
		MSRC	MM	FR	MR	FM	COX	DD
Late	SimGNN	0.125±0.008	0.281±0.006	0.308±0.007	0.313±0.008	0.299±0.006	0.366±0.008	0.115±0.009
	GraphSim	0.153±0.009	0.336±0.008	0.337±0.008	0.315±0.009	0.366±0.007	0.292±0.006	0.146±0.010
	GOTSim	-0.088±0.007	0.320±0.008	0.327±0.008	0.307±0.009	0.380±0.008	0.416±0.009	-0.092±0.007
	NeuroMatch	0.125±0.011	0.376±0.010	0.365±0.009	0.370±0.012	0.406±0.010	0.440±0.009	0.142±0.014
	IsoNet	0.185±0.013	0.381±0.012	0.388±0.012	0.351±0.014	0.402±0.011	0.406±0.009	0.168±0.015
	GEN	0.111±0.013	0.325±0.011	0.332±0.012	0.305±0.011	0.326±0.011	0.391±0.011	0.137±0.015
	LMCCS	0.248±0.013	0.451±0.012	0.438±0.014	0.406±0.014	0.457±0.012	0.487±0.012	0.212±0.015
Early	GMN	0.174±0.013	0.416±0.010	0.405±0.012	0.379±0.013	0.431±0.011	0.479±0.011	0.173±0.014
	XMCS	0.198±0.014	0.452±0.012	0.451±0.014	0.412±0.014	0.453±0.012	0.501±0.011	0.201±0.015

Table 8: Performance measured using **Kendall Tau Rank Correlation (KTau)** with standard error, of our models and state-of-the-art baselines on 20% test set, for all seven datasets. Top-half and bottom-half report results for MCES and MCCS respectively. Numbers in green (blue) indicate the best performers among early (late) interaction models. Numbers in red (yellow) indicate second best performers for early (late) interaction models.

	MCES	PairRank (higher is better)						
		MSRC	MM	FR	MR	FM	COX	DD
Late	SimGNN	0.644±0.002	0.768±0.005	0.756±0.005	0.754±0.005	0.764±0.004	0.796±0.007	0.631±0.003
	GraphSim	0.785±0.003	0.814±0.005	0.828±0.004	0.824±0.003	0.852±0.004	0.803±0.005	0.846±0.002
	GOTSim	0.848±0.002	0.837±0.006	0.821±0.005	0.868±0.004	0.850±0.004	0.781±0.008	0.868±0.001
	NeuroMatch	0.891±0.002	0.854±0.006	0.870±0.005	0.890±0.005	0.877±0.005	0.887±0.005	0.904±0.004
	IsoNet	0.913±0.001	0.867±0.006	0.861±0.006	0.883±0.004	0.875±0.005	0.893±0.007	0.923±0.002
	GEN	0.887±0.003	0.801±0.009	0.835±0.008	0.825±0.008	0.821±0.010	0.851±0.009	0.885±0.005
	LMCES	0.927±0.001	0.921±0.003	0.921±0.004	0.927±0.003	0.930±0.002	0.933±0.003	0.939±0.001
Early	GMN	0.914±0.001	0.896±0.003	0.906±0.004	0.905±0.003	0.901±0.004	0.919±0.004	0.925±0.001
	XMCS	0.932±0.001	0.925±0.003	0.926±0.004	0.937±0.002	0.927±0.002	0.938±0.003	0.939±0.001

	MCES	PairRank (higher is better)						
		MSRC	MM	FR	MR	FM	COX	DD
Late	SimGNN	0.752±0.016	0.772±0.009	0.804±0.008	0.843±0.008	0.791±0.009	0.812±0.008	0.698±0.015
	GraphSim	0.810±0.017	0.810±0.007	0.828±0.007	0.841±0.008	0.843±0.007	0.748±0.006	0.774±0.017
	GOTSim	0.311±0.010	0.802±0.009	0.818±0.008	0.839±0.010	0.855±0.007	0.850±0.007	0.312±0.012
	NeuroMatch	0.716±0.021	0.844±0.007	0.849±0.006	0.892±0.007	0.875±0.006	0.869±0.006	0.724±0.020
	IsoNet	0.849±0.017	0.845±0.009	0.866±0.006	0.865±0.009	0.869±0.006	0.844±0.007	0.775±0.020
	GEN	0.667±0.020	0.801±0.009	0.823±0.010	0.831±0.009	0.804±0.009	0.829±0.008	0.671±0.023
	LMCCS	0.863±0.019	0.904±0.004	0.909±0.005	0.915±0.005	0.912±0.004	0.903±0.005	0.831±0.018
Early	GMN	0.818±0.017	0.878±0.006	0.881±0.006	0.895±0.006	0.893±0.005	0.898±0.005	0.805±0.017
	XMCS	0.863±0.019	0.903±0.004	0.919±0.004	0.925±0.005	0.910±0.004	0.916±0.005	0.854±0.017

Table 9: Performance measured using **Pairwise Ranking Reward (PairRank)** with standard error, of our models and state-of-the-art baselines on 20% test set, for all seven datasets. Top-half and bottom-half report results for MCES and MCCS respectively. Numbers in green (blue) indicate the best performers among early (late) interaction models. Numbers in red1 (yellow) indicate second best performers for early (late) interaction models.

D.2 Effect of MCS layer

In Table 2 in the main submission, we reported MSE on four datasets. In Tables 10– 11, we report results with standard error, on all seven datsets, which probe the effect of substituting the general purpose scoring layer with MCS customized scoring layer. Here, the standard error is computed over the variation across all test queries. They reveal similar insights as in Table 2.

D.3 Ablation Study

In Table 3 in the main submission, we reported MSE for an ablation study on three datasets. In Tables 12–13, we report results with standard error, on all seven datasets. Here, the standard error is computed over the variation across all test queries. We make the following observations:

MCES		**MSE (lower is better)**						
		MSRC	MM	FR	MR	FM	COX	DD
Late	GEN	0.426±0.020	0.311±0.017	0.273±0.012	0.284±0.013	0.324±0.023	0.277±0.021	0.568±0.110
	GEN (MCS)	0.284±0.008	0.181±0.005	0.179±0.005	0.169±0.004	0.177±0.004	0.153±0.006	0.298±0.008
	IsoNet	0.276±0.007	0.225±0.009	0.220±0.008	0.209±0.007	0.253±0.017	0.182±0.007	0.333±0.059
	IsoNet (MCS)	0.260±0.011	0.187±0.012	0.178±0.005	0.173±0.005	0.175±0.004	0.148±0.005	0.256±0.011
	LMCES	0.232±0.008	0.167±0.005	0.170±0.005	0.162±0.005	0.163±0.004	0.140±0.004	0.223±0.006
Early	GMN	0.269±0.006	0.184±0.004	0.181±0.005	0.178±0.004	0.189±0.005	0.155±0.006	0.273±0.008
	GMN (MCS)	0.228±0.005	0.155±0.003	0.158±0.004	0.157±0.003	0.162±0.003	0.134±0.003	0.217±0.004
	XMCS	0.226±0.007	0.154±0.003	0.162±0.005	0.154±0.004	0.160±0.003	0.132±0.004	0.220±0.005

MCES		**MSE (lower is better)**						
		MSRC	MM	FR	MR	FM	COX	DD
Late	GEN	0.171±0.025	0.366±0.029	0.344±0.033	0.290±0.030	0.356±0.030	0.309±0.022	0.197±0.025
	GEN (MCS)	0.076±0.014	0.226±0.020	0.195±0.018	0.161±0.014	0.204±0.021	0.180±0.012	0.108±0.019
	IsoNet	0.086±0.015	0.237±0.016	0.244±0.019	0.191±0.017	0.218±0.018	0.253±0.018	0.124±0.019
	IsoNet (MCS)	0.088±0.016	0.230±0.020	0.225±0.019	0.161±0.014	0.206±0.021	0.195±0.014	0.119±0.018
	LMCCS	0.068±0.012	0.174±0.015	0.179±0.016	0.134±0.012	0.173±0.017	0.177±0.012	0.097±0.016
Early	GMN	0.101±0.015	0.200±0.015	0.216±0.020	0.156±0.015	0.193±0.018	0.176±0.011	0.137±0.016
	GMN (MCS)	0.070±0.014	0.178±0.016	0.173±0.018	0.125±0.011	0.164±0.018	0.154±0.011	0.098±0.016
	XMCS	0.071±0.014	0.168±0.014	0.163±0.018	0.131±0.013	0.168±0.017	0.153±0.009	0.102±0.017

Table 10: Performance measured using **mean square error (MSE)** with standard error, showing effect of replacing the general-purpose scoring layers with **new layers customized to MCS** on most competitive baselines, *viz.*, GEN and IsoNet (late interaction models) and GMN, across all seven datasets. Numbers in green (red) indicate the best (second best) performers for early interaction models. Numbers in blue (yellow) indicate the best (second best) performers for late interaction models. The proposed modification improves performance of all baselines. However our models outperform them, even after modifying their layers, in most cases.

MCES		**KTau (higher is better)**						
		MSRC	MM	FR	MR	FM	COX	DD
Late	GEN	0.627±0.005	0.416±0.013	0.468±0.011	0.456±0.012	0.456±0.015	0.466±0.014	0.635±0.010
	GEN (MCS)	0.666±0.004	0.556±0.006	0.565±0.007	0.584±0.006	0.590±0.005	0.555±0.010	0.690±0.004
	IsoNet	0.669±0.004	0.506±0.010	0.504±0.008	0.537±0.007	0.532±0.007	0.522±0.012	0.698±0.005
	IsoNet (MCS)	0.677±0.004	0.569±0.006	0.570±0.007	0.585±0.005	0.589±0.005	0.565±0.009	0.708±0.004
	LMCES	0.691±0.004	0.577±0.006	0.588±0.006	0.598±0.005	0.610±0.004	0.574±0.009	0.724±0.004
Early	GMN	0.670±0.003	0.544±0.006	0.567±0.007	0.568±0.006	0.569±0.006	0.555±0.009	0.701±0.004
	GMN (MCS)	0.693±0.004	0.583±0.006	0.592±0.006	0.598±0.005	0.606±0.005	0.573±0.010	0.727±0.004
	XMCS	0.699±0.004	0.582±0.006	0.594±0.006	0.612±0.005	0.606±0.005	0.580±0.009	0.724±0.004

MCES		**KTau (higher is better)**						
		MSRC	MM	FR	MR	FM	COX	DD
Late	GEN	0.111±0.013	0.325±0.011	0.332±0.012	0.305±0.011	0.326±0.011	0.391±0.011	0.137±0.015
	GEN (MCS)	0.223±0.013	0.421±0.011	0.418±0.013	0.385±0.014	0.424±0.011	0.477±0.010	0.218±0.015
	IsoNet	0.185±0.013	0.381±0.012	0.388±0.012	0.351±0.014	0.402±0.011	0.406±0.009	0.168±0.015
	IsoNet (MCS)	0.187±0.013	0.440±0.011	0.404±0.011	0.386±0.013	0.434±0.011	0.480±0.011	0.182±0.014
	LMCCS	0.248±0.013	0.451±0.012	0.438±0.014	0.406±0.014	0.457±0.012	0.487±0.012	0.212±0.015
Early	GMN	0.174±0.013	0.416±0.010	0.405±0.012	0.379±0.013	0.431±0.011	0.479±0.011	0.173±0.014
	GMN (MCS)	0.192±0.014	0.450±0.011	0.443±0.014	0.408±0.014	0.458±0.012	0.499±0.011	0.198±0.016
	XMCS	0.198±0.014	0.452±0.012	0.451±0.014	0.412±0.014	0.453±0.012	0.501±0.011	0.201±0.015

Table 11: Performance measured using **Kendall Tau Rank Correlation (KTau)** with standard error, showing effect of replacing the general-purpose scoring layers with **new layers customized to MCS** on most competitive baselines, *viz.*, GEN and IsoNet (late interaction models) and GMN, across all seven datasets. Numbers in green (red) indicate the best (second best) performers for early interaction models. Numbers in blue (yellow) indicate the best (second best) performers for late interaction models. The proposed modification improves performance of all baselines. However our models outperform them, even after modifying their layers, in most cases.

1. LMCES (final layer), where the relevance score is computed using only the embeddings of the R^{th} layer, is outperformed by LMCES in 12 out of 14 cases across MSE and KTau.
2. LMCCS (no gossip), where we remove the gossip network and compute $s(G_q, G_c) = \sum_{i,j} \min(\boldsymbol{A}_q \odot \boldsymbol{M}_q, \boldsymbol{P}^{(R)}\boldsymbol{A}_c \odot \boldsymbol{M}_c\boldsymbol{P}^{(R)})_{i,j}$, is consistently the worst performed amongst all three MCCS late interaction variants.
3. LMCCS (no NOISE FILTER) where we set $\tau_t = 0$ in Eq. (10), is outperformed by LMCES in 12 out of 14 cases across MSE and KTau.
4. There is no clear winner between XMCS, and XMCS (all layers) where we compute the relevance score in Eq. (17) using embeddings from all R layers. We observe that the performance scores of

	MCES	MSE (lower is better)						
		MSRC	MM	FR	MR	FM	COX	DD
Late	LMCES (final layer)	0.237±0.008	0.175±0.005	0.170±0.005	0.166±0.004	0.167±0.004	0.140±0.004	0.226±0.005
	LMCES	0.232±0.008	0.167±0.005	0.170±0.005	0.162±0.005	0.163±0.004	0.140±0.004	0.223±0.006
Early	XMCS (all layers)	0.224±0.008	0.154±0.004	0.165±0.004	0.152±0.004	0.155±0.003	0.128±0.003	0.223±0.004
	XMCS	0.226±0.007	0.154±0.003	0.162±0.005	0.154±0.004	0.160±0.003	0.132±0.004	0.220±0.005

	MCES	MSE (lower is better)						
		MSRC	MM	FR	MR	FM	COX	DD
Late	LMCCS (no gossip)	0.166±0.017	0.241±0.017	0.240±0.017	0.187±0.014	0.237±0.019	0.224±0.015	0.293±0.027
	LMCCS (no NOISE FILTER)	0.068±0.012	0.194±0.016	0.206±0.024	0.140±0.013	0.177±0.016	0.205±0.018	0.104±0.018
	LMCCS	0.068±0.012	0.174±0.015	0.179±0.016	0.134±0.012	0.173±0.017	0.177±0.012	0.097±0.016
Early	XMCS (all layers)	0.069±0.014	0.181±0.016	0.171±0.016	0.129±0.013	0.172±0.017	0.156±0.010	0.103±0.017
	XMCS	0.071±0.014	0.168±0.014	0.163±0.018	0.131±0.013	0.168±0.017	0.153±0.009	0.102±0.017

Table 12: Performance measured using **mean square error (MSE)** with standard error, on the four variants of our models considered for **Ablation study**. (i) LMCES (final layer) where the relevance score is computed using only the embeddings of the R^{th} layer, (ii) LMCCS (no gossip), where we remove the gossip network and compute $s(G_q, G_c) = \sum_{i,j} \min(\boldsymbol{A}_q \odot \boldsymbol{M}_q, \boldsymbol{P}^{(R)} \boldsymbol{A}_c \odot \boldsymbol{M}_c \boldsymbol{P}^{(R)})_{i,j}$, (iii) LMCCS (no NOISE FILTER) where we set $\tau_t = 0$ in Eq. (10) and (iv) XMCS (all layers) where we compute the relevance score in Eq. (17) using embeddings from all R layers. Numbers in green (red) indicate the best (second best) performers for early interaction models. Numbers in blue (yellow) indicate the best (second best) performers for late interaction models.

	MCES	KTau (higher is better)						
		MSRC	MM	FR	MR	FM	COX	DD
Late	LMCES (final layer)	0.692±0.004	0.562±0.006	0.581±0.006	0.594±0.005	0.599±0.005	0.566±0.009	0.720±0.004
	LMCES	0.691±0.004	0.577±0.006	0.588±0.006	0.598±0.005	0.610±0.004	0.574±0.009	0.724±0.004
Early	XMCS (all layers)	0.700±0.004	0.587±0.006	0.580±0.006	0.610±0.005	0.613±0.005	0.582±0.010	0.722±0.004
	XMCS	0.699±0.004	0.582±0.006	0.594±0.006	0.612±0.005	0.606±0.005	0.580±0.009	0.724±0.004

	MCES	KTau (higher is better)						
		MSRC	MM	FR	MR	FM	COX	DD
Late	LMCCS (no gossip)	0.132±0.011	0.418±0.011	0.380±0.012	0.360±0.013	0.415±0.011	0.476±0.011	0.091±0.008
	LMCCS (no NOISE FILTER)	0.241±0.013	0.436±0.011	0.433±0.013	0.392±0.014	0.446±0.011	0.477±0.011	0.213±0.015
	LMCCS	0.248±0.013	0.451±0.012	0.438±0.014	0.406±0.014	0.457±0.012	0.487±0.012	0.212±0.015
Early	XMCS (all layers)	0.201±0.014	0.455±0.012	0.448±0.014	0.410±0.015	0.453±0.012	0.500±0.011	0.196±0.015
	XMCS	0.198±0.014	0.452±0.012	0.451±0.014	0.412±0.014	0.453±0.012	0.501±0.011	0.201±0.015

Table 13: Performance measured using **Kendall Tau correlation (KTau)** with standard error, on the four variants of our models considered for **Ablation study** mentioned in Table 12. Numbers in green (red) indicate the best (second best) performers for early interaction models. Numbers in blue (yellow) indicate the best (second best) performers for late interaction models.

both variants are quite close to each other in most cases, with XMCS gaining a slight edge over XMCS (all layers) in 14 out of 28 cases.

D.4 Interpretabilty

For both MCES and MCES, our models propose a soft alignment between the nodes of the query-corpus pair. We use the Hungarian algorithm on top of it to obtain an injective mapping $\boldsymbol{P}$, which is depicted by matching node colors in the example graph pairs in Figure 2 and Figure 3. Subsequently, we compute the adjacency matrix of the MCS graph under the proposed alignment as $\min(\boldsymbol{A}_q, \boldsymbol{P}\boldsymbol{A}_c\boldsymbol{P}^\top)$. For LMCES, we indicate the edges of the proposed MCS graph in **thick black**. For LMCCS, we further apply TARJANSCC, to identify the largest connected component, whose edges are again indicated in **thick black**. In Figure 2, we present one example each, of the proposed alignments in the MCES and MCCS settings. For MCES, there are 10 overlapping edges under the proposed node alignment, which are in two disconnected components of 7 and 3 edges. For MCCS, the proposed node alignment identifies a set of connected 8 connected nodes, common to both graphs, which are connected by **thick black** edges.

In Figure 3, we present an example graph pair, where XMCS is able to identify a larger common connected component, as compared to LMCCS. In the alignment shown on the left, we see that there are 6 nodes in the connected component, identified under the node alignment proposed by LMCCS. On the other hand, on the right we observe that XMCS is able to propose a node alignment, which leads to the emergence of a connected component with 10 nodes.

(a) LMCES (b) LMCCS

Figure 2: Example graph pairs with node colors indicating the proposed alignments by our models. In Panel (a), we show MCES setup. Here, Nodes are aligned to maximize the number of overlapping edges (indicated as **thick black** edges). We observe two disconnected components in the MCES setup. In Panel (b), we show MCCS setup. Here, Nodes are aligned to identify largest connected component (identified by **thick black** edges).

(a) LMCCS (b) XMCS

Figure 3: Example graph pairs with node colors indicating the proposed alignments by our models LMCCS and XMCS, for the MCCS setup. In Panel (a), LMCCS proposes an alignment which results in an MCCS score of 6. In Panel (b), XMCS proposes an alignment on the same query-corpus pair, which results in an MCCS score of 10. The connected nodes in both cases are identified by **thick black** edges.

D.5 Training and inference times

Here, we report both the training and inference time (in seconds). The training time is computed for each batch of size 128 (which is the fixed hyperparameter used for the numbers reported in the paper). Inference time is computed for the entire test set of 100 query graphs and 800 corpus graphs, with the maximum possible batch size allowed by our GPU - Nvidia TITAN X (Pascal).

Method	Training Time per batch (in secs)	Inference time on test(in secs)
GEN	0.037	5.937
SimGNN	0.073	24.671
GraphSim	0.125	23.284
NeuroMatch	0.027	6.532
GOTSim	0.259	72.590
IsoNet	0.069	13.956
GMN	0.426	99.542
LMCES	0.109	28.129
LMCCS	0.074	13.776
XMCS	0.159	31.101

Table 14: Training and inference time

We observe that: (1) Among the late interaction models, both LMCCS and LMCES are significantly faster than GOTSim. This is because GOTSim uses a combinatorial solver, which does not allow for batched processing and results in significant slowdown. (2) LMCES is comparable to GraphSim and SimGNN, in both training and inference times, while affording significantly better performance. (3) LMCCS is comparable to IsoNet in training and inference times, and is significantly faster than SimGNN, GraphSim and GOTSim. (4) GEN and NeuroMatch are significantly faster than all late interaction models in terms of training and inference times. However, as shown in Table 1 of the main paper, both these models are outperformed by LMCCS,LMCES and IsoNet in most of the datasets. (5) Among the early interaction models, XMCS is 3X faster than GMN. The reason for this is explained in lines 338-344 in our main paper.

It is true that we performed experiments on graphs of size < 20, driven by the needs of practical graph retrieval applications like molecular fingerprint detection, object detection in images, etc. However, our method can easily scale beyond $|V| = 20$, as follows:

Inference Time (in sec)	Current Size	$\lvert V\rvert = 30$	$\lvert V\rvert = 50$	$\lvert V\rvert = 70$	$\lvert V\rvert = 100$
LMCES	0.036	0.048	0.052	0.071	0.106
LMCCS	0.022	0.031	0.039	0.042	0.064
XMCS	0.067	0.071	0.085	0.095	0.131

Table 15: Scalability for large graphs