Graph Component Contrastive Learning for Concept Relatedness Estimation

Yueen Ma¹, Zixing Song¹, Xuming Hu², Jingjing Li¹, Yifei Zhang¹, Irwin King¹

¹The Chinese University of Hong Kong,
²Tsinghua University
{yema21, zxsong, lijj, yfzhang, king}@cse.cuhk.edu.hk, hxm19@mails.tsinghua.edu.cn

Abstract
Concept relatedness estimation (CRE) aims to determine whether two given concepts are related. Existing methods only consider the pairwise relationship between concepts, while overlooking the higher-order relationship that could be encoded in a concept-level graph structure. We discover that this underlying graph satisfies a set of intrinsic properties of CRE, including reflexivity, commutativity, and transitivity. In this paper, we formalize the CRE properties and introduce a graph structure named ConcreteGraph. To address the data scarcity issue in CRE, we introduce a novel data augmentation approach to sample new concept pairs from the graph. As it is intractable for data augmentation to fully capture the structural information of the ConcreteGraph due to a large amount of potential concept pairs, we further introduce a novel Graph Component Contrastive Learning framework to implicitly learn the complete structure of the ConcreteGraph. Empirical results on three datasets show significant improvement over the state-of-the-art model. Detailed ablation studies demonstrate that our proposed approach can effectively capture the high-order relationship among concepts.

Introduction
Concept relatedness estimation (CRE) is the task of determining whether two concepts are related. In CRE, a concept can be a Wikipedia entry, a news article, a social media post, etc. Table 1 includes a pair of related concepts and an unrelated concept. In this example, when given the first two concepts, “Open-source software” and “GNU General Public License”, the goal of CRE is to label them as a related pair; but for “Open-source software” and “Landscape architecture”, one should label them as unrelated.

Existing CRE methods (Liu et al. 2019a) only consider the provided concept pairs, which are low-order pairwise relationships. However, we discover that there exist higher-order relationships in CRE that can be encoded by a concept-level graph structure, where the original pairs correspond to immediate neighbors (Ein-Dor et al. 2018; Liu et al. 2019a). The higher-order relationships are manifested as the multi-hop neighbors in the graph, which are validated by three types of intrinsic properties of CRE: reflexivity, commutativity, and transitivity. To construct the graph, we treat each concept as a node and add edges between any related concept pairs. In this paper, we name it ConcreteGraph (Concept relatedness Graph). It enables us to obtain new concept pairs that are not limited to low-order relationships. For example, one of the transitivity properties states that if concepts ($x_A, x_B$) are related and concepts ($x_B, x_C$) are related, then we can extract a new related concept pair ($x_A, x_C$), where $x_C$ is a 2-hop neighbor of $x_A$. Therefore, when we sample new concept pairs from the ConcreteGraph, the CRE properties are implicitly utilized.

The main challenge is how to take full advantage of the structural information of the ConcreteGraph. The most straightforward approach is to explicitly add new concept pairs to the dataset, which can be seen as a data augmentation method for alleviating the data scarcity issue. However, this approach exhibits a problem: there is an excessive amount of potential concept pairs. Assuming that we extract all possible related concept pairs by adding edges between any two connected concepts in the ConcreteGraph, we essentially turn every component of the ConcreteGraph into a complete subgraph. Because the edges in the original ConcreteGraph are sparse, the number of new related concept pairs grows quadratically with the number of concepts. It would become impractical for a model to learn all of them and, thus, we must only sample a subset of all possible concept pairs. If we sample completely at random, the quality of

Table 1: Examples of concept relatedness estimation from the WORD dataset.

<table>
<thead>
<tr>
<th>Related</th>
<th>Unrelated</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open-source software (OSS)</td>
<td>Landscape architecture</td>
</tr>
<tr>
<td>GNU General Public License (GNU GPL or simply GPL)</td>
<td>is the design of outdoor areas, landmarks, and structures to achieve environmental, social-behavioural, or aesthetic outcomes</td>
</tr>
</tbody>
</table>

Copyright © 2023, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.
the new concept pairs cannot be guaranteed. That is, when
the path length between two concepts is long, the quality of
their relationship tends to be low because the probability of
the existence of a noisy edge on the path becomes high. This
phenomenon is later confirmed by our experiments. A sim-
ples solution is to extract concept pairs from the local k-hop
neighborhood. But this limits the ConcreteGraph-based data
augmentation to contain only the local graph structure.

To this end, we introduce a novel Graph Component
Contrastive Learning (GCCL) framework that can implicitly
capture the complete structural information of the Con-
creteGraph. Rather than explicitly learning new concept
pairs, the objective of GCCL is that a concept treats its
own component as positive, whereas it forms negative pairs
with all other components. The representation of a concept
or a component is achieved using a shared backbone en-
coder. Thus, GCCL gives the encoder access to both lo-
cal and global structural information, which would be in-
tractable if we used the ConcreteGraph-based data augmen-
tation method. GCCL is inspired by Prototypical Contrastive
Learning (PCL) (Li et al. 2021), but the difference is that it
does not use k-means to find clusters. Each component of
the ConcreteGraph corresponds to a cluster, and therefore,
components in GCCL translate to prototypes in PCL.

Specifically, GCCL involves three main steps: (1) we use a
Transformer model as the backbone encoder to embed all
concepts and components before each epoch; (2) for each con-
ccept, we train the encoder to distinguish the concept’s
own component from all other components, which is defined
to be the GC-NCE objective; (3) along with the main GC-
NCE target, we also use a momentum encoder as in the Mo-
mentum Contrast (MoCo) framework (He et al. 2020; Chen
et al. 2020b) to optimize an InfoNCE loss (van den Oord,
Li, and Vinyals 2018). This InfoNCE loss contrasts a con-
cept with other concepts. It is added to the main GC-NCE
target to preserve the local smoothness of the overall GCCL
loss function (Li et al. 2021). We conduct comprehensive ex-
periments with three different Transformer models on three
datasets. The empirical results show that the GCCL frame-
work is capable of significantly improving their performance
and, when combined with the ConcreteGraph-based data
augmentation method, it can sometimes bring even more
improvement. We also conduct detailed ablation studies to
show the effectiveness of each module in our method. Our
code is available on Github1.

The main contributions of our paper are as follows:

- For CRE, we formally summarize its intrinsic properties
  that can be categorized into three types: reflexivity, com-
  mutativity, and transitivity. On their basis, we construct a
concept-level graph structure, ConcreteGraph, which en-
codes not only the provided concept pairs but also the
higher-order relationships between concepts. To the best
of our knowledge, we are the first to find such a graph
structure for the CRE task.

- We propose a novel Graph Component Contrastive
  Learning (GCCL) framework for taking full advantage
  of the local and global structural information of the Con-
  creteGraph. GCCL can be complemented by a novel
ConcreteGraph-based data augmentation method that ex-
plicitly provides local neighborhood information.

- Our method significantly improves over the state-of-the-
  art Concept Interaction Graph (CIG) model (Liu et al.
  2019a) on two datasets, CNSE and CNSS. We are also
  the first to apply deep neural networks to the WORD
  dataset (Ein-Dor et al. 2018).

Related Work

Concept Relatedness Estimation
The concept relatedness estimation (CRE) task stemmed
from the concept similarity matching (CSM) task in the area
of formal concept analysis (FCA). In FCA, a concept is for-
mally defined as a pair of sets: a set of objects and a set of
attributes in a given domain (Formica 2006). Methods for
assessing concept similarity include ontology-based meth-
ods (Formica 2006, 2008), Tversky’s-Ratio-based meth-
ods (Lombardi and Sartori 2006), rough-set-based methods
(Wang and Liu 2008), and semantic-distance-based methods
(Ge and Qiu 2008; Li and Xia 2011). The definition for con-
cepts in FCA is not suitable for the CRE task because CRE
concepts are long text documents and, thus, the CSM meth-
ods cannot be applied to the CRE task. Inspired by the gi-
ant success of introducing deep neural networks into natural
language processing applications (Li et al. 2019; Gao et al.
2020; Sun et al. 2022; Li et al. 2020), we adopt Transformer
models (Vaswani et al. 2017) to address the CRE task.

CRE is also related to tasks such as semantic textual simi-
larity (Cer et al. 2017; Zhang and Zhu 2019, 2020), text simi-
larity (Thijs 2019), text relatedness (Tsatsaronis, Varlamis,
and Vazirgiannis 2014), text matching (Jiang et al. 2019),
and text classification (Hu et al. 2020, 2021a,b; Liu et al.
2022; Li et al. 2022). The current state-of-the-art model for
semantic textual similarity is XLNet (Yang et al. 2019), but
there is still no work that applies deep neural networks to the
WORD dataset. With the advancement of Graph Neu-
ral Networks (GNNs) (Song and King 2022; Song, Zhang,
and King 2022; Song et al. 2021, 2022; Zhang et al. 2022a),
Liu et al. (2019a) introduced the Concept Interaction Graph
(CIG) method to match news article pairs along with two
new datasets, CNSE and CNSS. CIG is the current state-of-
the-art model on these two datasets.

CRE can play an important role in a wide range of ap-
plications, such as information retrieval (Busch et al. 2012;
Teevan, Ramage, and Morris 2011), document clustering
(Aswani Kumar and Srinivas 2014), plagiarism detection
(Muangprathub, Kajornkasirat, and Wanichsombat 2021),
etc. Therefore, CRE has been attracting much interest lately.

Contrastive Learning
Since the introduction of contrastive learning (Chopra, Had-
sell, and LeCun 2005), many variants have been developed.
InfoNCE (van den Oord, Li, and Vinyals 2018) aims to find a
positive in a group of noise samples. MoCo (He et al. 2020;
Chen et al. 2020b) provides a queue-based framework for
utilizing data from previous batches with the addition of a
Figure 1: Overview of our proposed framework for concept relatedness estimation. (a) After collecting the concept pairs into the ConcreteGraph, the concepts are divided into multiple components, e.g., $c_1$ and $c_2$. For simplicity, concepts are represented with their indices, e.g., the node with index A represents concept $x_A$; edges connect related concepts: solid black edges are true annotations, and dashed gray edges correspond to new related concept pairs from ConcreteGraph-based data augmentation; disconnected pairs are unrelated ones, such as $(x_A, x_F)$. (b) Under the Graph Component Contrastive Learning (GCCL) framework, the GC-NCE objective aims to distinguish each concept’s own component from other components (in this example, the concept is $x_A$, highlighted in yellow); the MoCo objective is to contrast the momentum embedding of each concept with the momentum embeddings of a queue of other concepts. The positive embeddings are colored in green and the negative ones are in red. $\theta^{T-1}$ and $\theta^T$ are both the parameters of the main encoder but from two different epochs; $\theta^T_K$ is the parameters of the momentum encoder from the MoCo framework.

Our Method

Concept Relatedness Estimation

The concept relatedness estimation (CRE) task is to predict whether two given concepts are related or unrelated. Thus, it is a binary classification task with two labels “related” and “unrelated”. Formally, given a set of concepts $X = \{x_i | i \in \mathbb{N}_n\}$ with an index set $\mathbb{N}_n = \{1, \ldots, n\}$ and a set of known pairwise binary labels $Y = \{y_{i,j} | (i,j) \subset \mathbb{N}_n \times \mathbb{N}_n\}$, the CRE task is to learn a parameterized encoder $f_\theta$ that can estimate the true relatedness $y_{i,j}$ of two concepts $x_i$ and $x_j$. In this paper, we focus on concepts that are long documents.

CRE Properties. The CRE task exhibits some unique intrinsic properties that are rarely present in typical NLP tasks.

To state these properties formally, we assume three concepts $x_A$, $x_B$, and $x_C$. The similarity symbol “∼” is used to denote that two concepts are “related”, while the dissimilarity symbol “≁” is used to connect two unrelated concepts.

**Property 1** (Reflexivity). A concept $x_A$ is related to itself: $x_A \sim x_A$.

**Property 2** (Commutativity of Relatedness). If $x_A$ and $x_B$ are related, then $x_B$ and $x_A$ are related: $x_A \sim x_B \iff x_B \sim x_A$.

**Property 3** (Commutativity of Unrelatedness). If $x_A$ and $x_B$ are unrelated, then $x_B$ and $x_A$ are unrelated: $x_A \not\sim x_B \iff x_B \not\sim x_A$.

**Property 4** (Transitivity of Relatedness). If $x_A$ is related to $x_B$ and $x_B$ is related to $x_C$, then $x_A$ and $x_C$ are related: $(x_A \sim x_B) \land (x_B \sim x_C) \implies x_A \sim x_C$.

**Property 5** (Transitivity of Unrelatedness). If $x_A$ is related to $x_B$ but $x_B$ is unrelated to $x_C$, then $x_A$ and $x_C$ are unrelated: $(x_A \sim x_B) \land (x_B \not\sim x_C) \implies x_A \not\sim x_C$.

ConcreteGraph. To make use of the CRE properties in practice, we can build a graph to encode both the pairwise relationships from the dataset and the higher-order relationships based on those properties. We name it ConcreteGraph (Concept relatedness Graph). An illustration of how to construct the ConcreteGraph is shown in Figure 1(a). Each node in ConcreteGraph represents a concept, and edges are only added between related concept pairs. The construction of the ConcreteGraph takes $O(|Y|)$ time as we need to iterate over all labeled concept pairs. Since it is a binary classification task, we do not need to set the edge weights. By the
nature of CRE, the ConcreteGraph usually has multiple connected components, where each component contains a set of related concepts. To identify the components of the ConcreteGraph, we can use breadth-first search (BFS), which takes \( O(V + E) \) time where \( V \) is the number of nodes and \( E \) is the number of edges.

**Data Augmentation**

The ConcreteGraph enables a straightforward data augmentation method for addressing the data scarcity problem. The CRE properties automatically come into play when we sample new concept pairs from the ConcreteGraph. Commutativity is used if we sample two nodes that are already provided by the dataset but in a different order. For example, if \((x_A, x_B)\) is provided and we sample \((x_B, x_A)\), then this new pair is validated by the commutativity of relatedness property (Property 2) or the commutativity of unrelatedness property (Property 3). Transitivity is utilized when the path between the two sampled concepts has at least two edges or when there is no path between them. For example, if there is a path \( x_A \rightarrow x_B \rightarrow x_C \), then the new concept pair \((x_A, x_C)\) is justified by the transitivity of relatedness property (Property 4). When two concepts are sampled from two different components, the transitivity of unrelatedness property (Property 5) is employed to prove that they are unrelated.

In theory, any two concepts within the same graph component are related; any two concepts from two different components are unrelated. But it is intractable to use all possible concept pairs. For example, the largest component of the ConcreteGraph of the WORD dataset (Ein-Dor et al. 2018) has 4,301 nodes. We could sample up to 9,247,150 related concept pairs from it, which is impractical for a Transformer model to learn. Therefore, we must only use a subset of those concept pairs. An intuitive heuristic is to pick a concept pair with a short path length. If the path length is too long, the connection between the concept pair becomes “risky”. Namely, when there are more edges on the path, the probability of the existence of a noisy edge becomes higher, which could degrade the quality of the path. We later prove this theory with an empirical ablation study in Sub-section “Ablation Study of Data Augmentation”. In practice, we obtain the \( k \)-hop neighborhood for every node, where \( k \) is a small integer, such as 2 or 3. We use a subset of the nodes in this local neighborhood as related concepts by setting a target augmentation ratio. Unrelated concept pairs can be produced by simply sampling concepts from other components.

**Graph Component Contrastive Learning**

We introduce Graph Component Contrastive Learning (GCCL) to take full advantage of local and global structural information of the ConcreteGraph, which is neglected by previous CRE methods. Our ConcreteGraph-based data augmentation only provides concept pairs from a local \( k \)-hop neighborhood, and it is intractable to include all concept pairs that satisfy the CRE properties. In fact, it is harmful to explicitly extract all possible concept pairs as augmented data due to the quality problem caused by long paths. GCCL captures global structural information by learning the representations of the components in the ConcreteGraph and contrasting a concept’s own component against other components. In this way, all of the related and unrelated concept pairs are preserved implicitly because each component embedding aggregates a group of related concept embeddings. Our method also has a connection to clustering methods: if we add edges for all possible related pairs, the ConcreteGraph itself would become a cluster graph because all the components become complete subgraphs. For that reason, a ConcreteGraph component in GCCL is equivalent to a cluster (prototype) in Prototypical Contrastive Learning (PCL) (Li et al. 2021). The difference is that GCCL does not rely on clustering algorithms to find clusters.

The overview of our GCCL framework is illustrated in Figure 1(b). The main objective of GCCL is to treat the embedding of a concept’s own component embedding as the positive and the embeddings of all other components as negatives. This allows the model to learn representations for both individual concepts and their components, which implicitly encode the global graph structure. Formally, we assume that the ConcreteGraph has a set of \( r \) components \( C = \{c_1, c_2, \ldots, c_r\} \), where each component itself is a set of concepts \( c_s = \{x_{s1}, x_{s2}, \ldots, x_{sp}\} \) and \( p \) is the number of concepts in \( c_s \). Same as PCL, we assume that \( f_\theta(x_i) \) follows an isotropic Gaussian distribution around the component embedding:

\[
p(x_i; c_s, \theta) = \frac{\exp \left(-\frac{(f_\theta(x_i) - f_\theta(c_s))^2}{2\sigma^2} \right)}{\sum_{j=1}^{r} \exp \left(-\frac{(f_\theta(x_i) - f_\theta(c_j))^2}{2\sigma^2} \right)},
\]

where \( x_i \in c_s \) and \( \sigma^2 \) is the variance. We also overload \( f_\theta \) with the input of a component, such that we get the representation of a component using a shared backbone encoder. In this paper, the component embedding is simply the element-wise average of the embeddings of its concepts:

\[
f_\theta(c_s) = \frac{1}{p} \sum_{k=1}^{p} f_\theta(x_{s_k}).
\]

Theoretically, we could add an extra Graph Neural Network (GNN) layer (Wu et al. 2021) on top of the concept embeddings to get the component embedding. However, because the components are complete subgraphs, GNN layers would quickly over-smooth the concept embeddings, which is close to their average. Thus, we do not use GNN layers to obtain the concept embeddings.

After applying \( L_2 \)-normalization to both the concept embeddings and the component embeddings, we have \( (f_\theta(x_i) - f_\theta(c_s))^2 = 2 - 2f_\theta(x_i) \cdot f_\theta(c_s) \). Then the maximum log-likelihood estimation for \( \theta \) optimizes a novel Graph-Component-NCE (GC-NCE) loss \( L_{GC-NCE} \):

\[
\theta^* = \arg \max_\theta \sum_{i=1}^{n} \log p(x_i; C, \theta) = \arg \min_\theta \sum_{i=1}^{n} -\log \frac{\exp \left(f_\theta(x_i) \cdot f_\theta(c_s)/\phi_s \right)}{\sum_{j=1}^{n} \exp \left(f_\theta(x_i) \cdot f_\theta(c_j)/\phi_j \right)} \triangleq \arg \min_\theta L_{GC-NCE},
\]

where

\[
\phi_s = \sum_{k=1}^{p} \|f_\theta(x_{s_k}) - f_\theta(c_s)\|_2\frac{p}{p \log (p + \alpha)}
\]
is a concentration estimation that replaces the variance in Eq. 1, which is also introduced in PCL; $\alpha$ is a smoothing hyper-parameter for preventing $\phi$ from becoming excessively large when the component size $p$ is small. When $\phi$ is small, the concept embeddings within the same component are pulled closer together and, thus, have a high concentration level.

In addition to the main GC-NE objective of GCCL, we also use the InfoNCE (van den Oord, Li, and Vinyals 2018) objective under the MoCo framework (He et al. 2020; Chen et al. 2020b). The idea of InfoNCE is to distinguish the embedding of a concept from the embeddings of a set of noise concepts. MoCo introduces a dynamic dictionary based on a queue data structure for storing the embeddings from a momentum encoder $f_\theta_m$ in a first-in-first-out (FIFO) fashion, which enables the decoupling between the batch size and the amount of negatives. This is especially useful since we use large Transformer models as encoders. The MoCo objective is constructed by contrasting the query embedding $f_\theta(x_i)$ from the main encoder and the key embeddings $f_\theta_m(x_j)$ from the momentum encoder. The momentum encoder $f_\theta_m$ is the moving average of the main encoder $f_\theta$ with a momentum coefficient $m \in (0, 1); \theta_k \leftarrow m \theta_k + (1 - m) \theta_q$.

Similar to InfoNCE, the MoCo loss treats the query and key embeddings of a concept as the positive pair, and the key embeddings of other concepts in the MoCo queue are negatives to the query embedding of the current concept. That leads to the MoCo objective:

$$L_{MoCo} = \sum_{i=1}^{n} - \log \frac{\exp (f_\theta(x_i) \cdot f_{\theta_m}(x_i)/\tau)}{\sum_{j=0}^{Q} \exp (f_\theta(x_i) \cdot f_{\theta_m}(x_j)/\tau)},$$  \hspace{1cm} (4)

where the denominator includes one positive and Q negatives in the MoCo queue; $\tau$ is a temperature hyper-parameter. The purpose of adding this additional MoCo objective is to preserve local smoothness of the loss function.

$\text{Li et al.} (2021)$: the GC-NCE objective (Eq. 2) and the CRE objective (Eq. 5) both pull the embeddings of related concepts closer, those concept embeddings might become very similar after finetuning, which can cause poor generalization; the MoCo objective preserves the distinction of individual concept embeddings by contrasting a concept against all other concepts.

In this paper, we use Transformer (Vaswani et al. 2017) as the encoder $f_\theta$. After passing the concepts into $f_\theta$ and obtaining their concept embeddings, the concept relatedness is estimated as follows:

$$h(x_i, x_j; \theta) = f_\theta(x_i) \oplus f_\theta(x_j) \oplus \left| f_\theta(x_i) - f_\theta(x_j) \right|$$

$$p(x_i, x_j) = \sigma(W h(x_i, x_j; \theta))$$

$$L_{CRE} = \mathcal{L}_{CE}(p(x_i, x_j), y_{i,j}),$$  \hspace{1cm} (5)

where $\oplus$ is the concatenation operation; $|f_\theta(x_i) - f_\theta(x_j)|$ is the element-wise absolute value of the difference of the two vectors; $\sigma$ is a sigmoid activation function; $W$ is a trainable parameter matrix; $\mathcal{L}_{CE}$ is the binary cross-entropy loss.

The overall objective $\mathcal{L}$ combines the GC-NCE loss (Eq. 2), the MoCo loss (Eq. 4) and the CRE loss (Eq. 5):

$$\mathcal{L} = \mathcal{L}_{CRE} + \beta \mathcal{L}_{GCCL} = \mathcal{L}_{CRE} + \beta(\mathcal{L}_{GC-NCE} + \mathcal{L}_{MoCo}),$$

where $\beta$ is the hyper-parameter that adjusts the effect of the GCCL loss $\mathcal{L}_{GCCL}$.

**Experiments**

We conduct experiments on three CRE datasets: WORD (Ein-Dor et al. 2018), CNSE and CNSS (Liu et al. 2019a). We compare the performance of three Transformer models, including BERT (Devlin et al. 2019), RoBERTa (Liu et al. 2019b) and XLNet (Yang et al. 2019). We also include four latest Graph Neural Network (GNN) models (Chen et al. 2020a; Zeng et al. 2021; Wang and Zhang 2022; Liu et al. 2021).
2019a) as baselines. Detailed ablation studies show the effect of different modules of our method.

Datasets

The Wikipedia Oriented Relatedness Dataset (WORD) dataset (Ein-Dor et al. 2018) collects English concepts from Wikipedia, including 19,176 concept pairs. The Chinese News Same Event (CNSE) dataset and the Chinese News Same Story (CNSS) dataset were both created by Liu et al. (2019a), containing 29,063 and 33,503 news article pairs from the major internet news providers in China, respectively. We use the official dataset split for WORD whose train-test ratio is approximately 2:1. Since CNSE and CNSS do not provide an official dataset split, they are split randomly with a train-dev-test ratio of 7:2:1. These dataset splits are fixed throughout different experiments across all models.

Experimental Setup

We use the AdamW optimizer (Loshchilov and Hutter 2019) with learning rate $\eta = 1 \times 10^{-5}$ and $\epsilon = 1 \times 10^{-8}$, following a linear schedule. The Transformer models are trained for 5 epochs. For MoCo (He et al. 2020; Chen et al. 2020b), we use queue size $Q = 32$, momentum coefficient $m = 1 - 1 \times 10^{-4}$, and temperature $\tau = 0.1$. We use $\beta = 0.1$ for the overall loss. CIG (Liu et al. 2019a) is trained with its default hyper-parameters. Experiments are conducted on four Nvidia TITAN V GPUs.

Performance Comparison

The performance comparison of different models is summarized in Table 2. Three performance metrics are used: accuracy, F1, and AUC. We compare our method with different GNNs and Transformers.

<table>
<thead>
<tr>
<th>Objective</th>
<th>BERT (Devlin et al. 2019) (CRE only)</th>
<th>w. MoCo</th>
<th>w. GC-NCE</th>
<th>w. GCCL (GC-NCE &amp; MoCo)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acc</td>
<td>76.85</td>
<td>77.10</td>
<td>75.58</td>
<td>77.83</td>
</tr>
<tr>
<td>F1</td>
<td>76.12</td>
<td>76.79</td>
<td>78.06</td>
<td>77.59</td>
</tr>
<tr>
<td>AUC</td>
<td>84.86</td>
<td>84.82</td>
<td>85.81</td>
<td>85.86</td>
</tr>
</tbody>
</table>

Table 3: The ablation study of the GCCL objective.
performance improvement from the two objectives stacks.

In Figure 2, we also visualize the concept embeddings with t-SNE (Van der Maaten and Hinton 2008) to qualitatively illustrate the effect of GCCL. GCCL better represents the concepts by taking the global structure of the ConcreteGraph into account: the embeddings of the concepts in the same component are closer to each other using GCCL and the embedding space of GCCL is better organized. This gives us an insight into how GCCL helps the main CRE objective: GCCL pulls related concepts together and pushes away unrelated components.

Ablation Study of Data Augmentation
To understand whether the quality or the quantity of augmented data better helps the model performance, we experiment with different $k$ values and target augmentation ratios, as summarized in Table 4. When the target augmentation ratio is specified, the sampling process of new related concept pairs stops when the target augmentation ratio is reached, regardless of whether all possible related concept pairs are sampled. If there is no target augmentation ratio ($\infty$), we find all possible related concept pairs for the given $k$. In both cases, we keep sampling unrelated concept pairs until the number of unrelated pairs is equal to that of related pairs. This ensures the balance of the augmented dataset.

When $k$ is fixed at 2 and the augmentation ratio increases, the performance worsens. If we take it to the extreme by not setting a target ratio, the performance drops significantly. These results show that a large augmentation ratio does not always help because it shrinks the proportion of annotated concept pairs and the quality of sampled concept pairs never matches that of annotated concept pairs. When the augmentation ratio is fixed at 2 and $k$ is incremented, the performance also declines. This confirms our theory about how path length can affect the quality of sampled concept pairs. Namely, when the path length between two concepts is long, the existence of a noisy edge becomes more likely, which degrades the overall quality of the connection.

Table 4: The effect of the number of hops $k$ and the augmentation ratio in the ConcreteGraph-based data augmentation method. “Samp.” means sampled related pairs for training and “Poss.” means all possible related pairs in the $k$-hop neighborhood. The number of sampled unrelated pairs is equal to the number of sampled related pairs and omitted.

<table>
<thead>
<tr>
<th>$k$</th>
<th>Target Ratio</th>
<th>Related Pairs</th>
<th>Acc</th>
<th>F1</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Samp. Poss.</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>$\infty$</td>
<td>6.8K 6.8K</td>
<td>76.85</td>
<td>76.12</td>
<td>84.86</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>12.9K 107K</td>
<td><strong>77.42</strong></td>
<td><strong>76.81</strong></td>
<td><strong>85.75</strong></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>19.3K 107K</td>
<td>75.66</td>
<td>74.51</td>
<td>83.96</td>
</tr>
<tr>
<td>4</td>
<td>4</td>
<td>25.8K 107K</td>
<td>72.23</td>
<td>72.07</td>
<td>79.99</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>32.2K 107K</td>
<td>70.99</td>
<td>70.78</td>
<td>78.95</td>
</tr>
<tr>
<td>2</td>
<td>$\infty$</td>
<td>107K 107K</td>
<td>63.51</td>
<td>58.65</td>
<td>70.97</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>404K 401K</td>
<td>56.76</td>
<td>51.38</td>
<td>63.69</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>12.9K 1.4M</td>
<td>75.91</td>
<td>75.81</td>
<td>83.78</td>
</tr>
<tr>
<td>5</td>
<td>5</td>
<td>12.9K 3.9M</td>
<td>74.90</td>
<td>75.44</td>
<td>83.08</td>
</tr>
</tbody>
</table>

Figure 2: The t-SNE visualization of the original and GCCL concept embeddings. Each point corresponds to a concept embedding, and the concepts in the same component share the same color. Both sub-figures show 200 Components.

Conclusion
Concept relatedness estimation is an emerging task that has a wide range of applications. To the best of our knowledge, we are the first to discover the concept-level graph structure that unveils the high-order relationship among concepts. We name it ConcreteGraph and develop a novel data augmentation method based on it. However, because data augmentation cannot capture the global structural information of the ConcreteGraph, we integrate a novel Graph Component Contrastive Learning (GCCL) framework to encode the complete graph structure. Experimental results show that the data augmentation method can improve the performance by $\sim 1\%$, whereas the improvement from GCCL can be up to $\sim 3\%$. Detailed analysis shows that GCCL better organizes the embedding space where related concepts are close while unrelated concepts are separated. We also conduct an experiment to show that it is difficult for the ConcreteGraph-based data augmentation method to provide as much performance benefit as GCCL by simply sampling more concept pairs, due to the degradation of the data quality caused by the long path length.
Acknowledgements

The work described here was partially supported by grants from the National Key Research and Development Program of China (No. 2018AAA0100204) and from the Research Grants Council of the Hong Kong Special Administrative Region, China (CUHK 2410021, Research Impact Fund, No. R5034-18).

References


Chen, M.; Wei, Z.; Huang, Z.; Ding, B.; and Li, Y. 2020a. Simple and Deep Graph Convolutional Networks. In ICML.


