ConcreteGraph: A Data Augmentation Method Leveraging the Properties of Concept Relatedness Estimation

Yueen Ma, Zixing Song, Chirui Chang, Yue Yu, Irwin King
The Chinese University Of Hong Kong
{yema21, zxsong, crchang9, yueyu9, king}@cse.cuhk.edu.hk

Abstract

The concept relatedness estimation (CRE) task is to determine whether two given concepts are related. Although existing methods for the semantic textual similarity (STS) task can be easily adapted to this task, the CRE task has some unique properties that can be leveraged to augment the datasets for addressing its data scarcity problem. In this paper, we construct a graph named ConcreteGraph (Concept relatedness estimation Graph) to take advantage of the CRE properties. For the sampled new concept pairs from the ConcreteGraph, we add an additional step of filtering out the new concept pairs with low quality based on simple yet effective quality thresholding. We apply the ConcreteGraph data augmentation on three Transformer-based models to show its efficacy. Detailed ablation study for quality thresholding further shows that even a limited amount of high-quality data is more beneficial than a large quantity of unthresholded data. This paper is the first one to work on the WORD dataset and the proposed ConcreteGraph can boost the accuracy of the Transformers by more than 2%. All three Transformers, with the help of ConcreteGraph, can outperform the current state-of-the-art method, Concept Interaction Graph (CIG), on the CNSE and CNSS datasets.

1 Introduction

Concept relatedness estimation (CRE) is the task of determining whether two concepts are related. A Wikipedia entry, a news article, or a mathematical definition can all be considered a concept. Table 1 shows 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”, one should label them as a related pair; but if “Open-source software” and “Landscape architecture” were given, one should mark them unrelated. CRE plays an important role in a wide range of applications, such as information retrieval (Busch et al., 2012; Teevan et al., 2011), document clustering (Aswani Kumar and Srinivas, 2010), plagiarism detection (Muangprathub et al., 2021), etc. In recent years, the amount of concepts has been rapidly growing, and it becomes unfeasible to assess the relatedness of every concept pair manually. Therefore, automated concept relatedness estimation has been attracting much interest.

In traditional settings, the concept similarity matching (CSM) task is closely related to CRE and often considered as a formal concept analysis (FCA) task, where a concept is formally defined as a pair of sets: a set of objects and a set of attributes in a given domain (Formica, 2006). But such a definition of a concept becomes less suitable for today’s CRE problems because structured concepts are scarce while unstructured text documents of concepts are ubiquitous. With the recent introduction of the more difficult CRE task, the definition of a concept is generalized to any text document that describes a concept.

The challenge of CRE lies in the unstructured long concept documents and the limited amount of training data. Because of the structure of a concept in FCA, the methods for CSM were restricted to using ontology, Tversky’s ratio and rough set (Formica, 2006, 2008; Lombardi and Sartori, 2006; Wang and Liu, 2008), etc. However, most of to-
day's concepts are written in natural language without any explicit mathematical structure. Traditional CSM methods become less suitable for such types of data. With the popularity of deep neural networks (DNNs) (LeCun et al., 2015), many DNN models for NLP are now capable of processing unstructured text inputs. Recurrent Neural Networks (RNNs) (Schuster and Paliwal, 1997) were built upon recurrent units, like LSTM (Hochreiter and Schmidhuber, 1997) or GRU (Cho et al., 2014), but they often suffer from the vanishing gradient problem (Hochreiter, 1998). This problem was recently solved by Transformers (Vaswani et al., 2017). Therefore, in this paper, we use Transformers as our backbone models, such as BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b) and XLNet (Yang et al., 2019).

To address the data scarcity issue, we also propose a novel data augmentation method called ConcreteGraph (Concept relatedness estimation Graph). Most NLP data augmentation methods focus on sentence-level strategies, including paraphrasing, noising and sampling (Li et al., 2022). For example, one can augment a sentence by replacing words with their synonyms, deleting certain parts of a sentence, and inserting a short phrase. However, those typical NLP data augmentation methods do not take advantage of the unique properties that CRE has. Therefore, we build the ConcreteGraph utilizing three types of CRE properties: reflexivity, commutativity and transitivity. From this graph, new relationships can be discovered between any sampled concept pairs. The original relationships provided by a CRE dataset describe the immediate neighborhood, but ConcreteGraph enables us to obtain new related concept pairs from the multi-hop neighborhood and new unrelated concept pairs from different graph components. For instance, in the Table 1 example, if it is additionally given that “Open-source software” is related to “Creative Commons license”, then we can find a new two-hop relationship between “GNU General Public License” and “Creative Commons license”, which did not exist in the set of relationships provided.

Despite the theoretical potential of ConcreteGraph, the new relationships vary in quality. Namely, some paths between two sampled ConcreteGraph nodes may have low quality because some edges have low relatedness scores or the path lengths are too long. Therefore, we need to filter out such concept pairs in practice during the data augmentation process. We introduce two simple yet effective quality thresholds to eliminate the harmful concept pairs and only keep the high-quality ones.

The main contributions of our paper are as follows:

- We propose a novel model-independent data augmentation method based on ConcreteGraph. The data augmentation method leverages the unique properties of the concept relatedness estimation (CRE) task. Detailed experiments demonstrate its effectiveness for multiple datasets in two different languages, English and Chinese, and across three Transformer models, BERT, RoBERTa and XLNet;
- This paper is the first to work on the WORD dataset (Ein-Dor et al., 2018) that contains English Wikipedia concepts. Using three Transformer models along with our data augmentation method, this paper sets a strong baseline for future work on this dataset;
- Our method also achieves considerable improvement over the state-of-the-art model, Concept Interaction Graph (CIG) (Liu et al., 2019a), on two datasets of Chinese news articles, the CNSE dataset and the CNSS dataset.

2 Related Work

2.1 CRE Task

Although the concept relatedness estimation task is a relatively new task initiated by the Wikipedia Oriented Relatedness Dataset (WORD) dataset (Ein-Dor et al., 2018), similar tasks have existed for a long time. Originally, the concept similarity matching task was introduced for the concepts in formal concept analysis (FCA). In FCA, a concept is formally 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 methods (Formica, 2006, 2008), Tversky's-Ratio-based methods (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 FCA definition of a concept becomes less useful in recent applications because more concepts are described in plain text. Therefore, in the CRE task, the definition of a concept is generalized to
Figure 1: The overview of our ConcreteGraph data augmentation method. Each node represents a concept; solid-line edges correspond to related concept pairs and dashed-line edges denote unrelated concept pairs. The green node A is the source node from which we find shortest paths to other nodes using Dijkstra’s algorithm. Target nodes are highlighted in yellow. In this example, we use the minimum path score and the score threshold is 0.7. Therefore, the path between A and B is filtered out because the edge score between B and G is 0.5 < 0.7. The maximum path length is set to 2. Thus, the path of length 3 between A and D is removed. The two paths, A-G and A-C, satisfy the quality thresholds and they are treated as two new related concept pairs. There are not paths between A and E, A and F, so they are considered as two new unrelated concept pairs.

2.2 Data Augmentation

Data augmentation is to generate synthetic data based on the original data so that the training set is larger. It is useful when the amount of data is limited or the model is overfitting. Thus, augmenting data usually makes the trained model generalize better. Data augmentation was first commonly used in computer vision tasks (Shorten and Khoshgoftaar, 2019). Multiple data augmentation methods were then introduced for NLP tasks. An example of text classification is EDA (Wei and Zou, 2019), which involves synonym replacement, random insertion, random swap, random deletion. There is no existing data augmentation method for CRE.

ConcreteGraph is closely associated with knowledge graph but they are not identical (Ji et al., 2022). In a typical knowledge graph, each node represents a named entity that is made up of a short phrase or even a single noun; but a node in ConcreteGraph corresponds to a concept, which can contain multiple sentences. In addition, a knowledge graph edge usually has a type attribute but a ConcreteGraph edge does not.

3 Our Method

3.1 CRE Properties

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”. In this paper, we focus on concepts that are in the form of long documents.

The CRE task exhibits some unique properties that are rarely present in other typical NLP tasks.
To state these properties formally, we assume that there are 3 concepts $A$, $B$ and $C$. The similarity symbol “$\sim$” is used to represent that two concepts are “related”, while the dissimilarity symbol “$\napprox$” is used to connect two unrelated concepts.

**Property 1** (Reflexivity). $A$ is related to itself.

$$A \sim A.$$  

**Property 2** (Commutativity of Relatedness). If $A$ and $B$ are related, then $B$ and $A$ are related:

$$A \sim B \iff B \sim A.$$  

**Property 3** (Commutativity of Unrelatedness). If $A$ and $B$ are unrelated, then $B$ and $A$ are unrelated:

$$A \napprox B \iff B \napprox A.$$  

**Property 4** (Transitivity of Relatedness). If $A$ is related to $B$ and $B$ is related to $C$, then $A$ and $C$ are related:

$$A \sim B \land B \sim C \implies A \sim C.$$  

**Property 5** (Transitivity of Unrelatedness). If $A$ is related to $B$ but $B$ is unrelated to $C$, then $A$ and $C$ are unrelated:

$$A \sim B \land B \napprox C \implies A \napprox C.$$  

**Property 6**. If $A$ is unrelated to $B$ and $B$ is unrelated to $C$, no conclusion can be drawn about the relatedness between $A$ and $C$

$$A \napprox B \land B \napprox C \implies \varnothing.$$  

Strictly speaking, the property 6 means that we cannot determine whether $A$ and $C$ are related or not, given that we only know $A \napprox B$ and $B \napprox C$. Despite that, we can still be fairly confident that $A$ and $C$ are unrelated in practice if we know enough about what concepts are related to $A$ or $C$. Namely, if we also know that many other concepts, $D$, $E$, $F$, . . . , are related to $A$ but none of them are $C$, we can still be relatively confident that $A$ and $C$ are unrelated. For example, in Table 1, although it is not explicitly stated that “Landscape architecture” is unrelated to “Open-source software” in the WORD dataset, since we know well about the neighborhood of “Open-source software” (Figure 2), we can safely conclude that the two concepts are likely to be unrelated. Therefore, this property can be relaxed and used to produce more augmented data.

We do not use the property 1 as it is trivial and only yields related concept pairs, which can cause imbalance to the augmented dataset. It is listed here only for completeness.

Figure 2: The neighborhood of the concept “Open-source software” in the extracted ConcreteGraph

### 3.2 ConcreteGraph Data Augmentation

**ConcreteGraph** To make use of the CRE properties in practice, we can build a graph to piece together the pairwise relationships from the dataset and then sample new concept pairs from this graph. We name it “ConcreteGraph” (concept relatedness graph). An overview of the steps in the data augmentation method is shown in Figure 1.

The structure of the ConcreteGraph is easy to understand: One can simply treat the concepts as the nodes and all related concept pairs as the edges. The annotation of the raw relatedness in the WORD dataset is a decimal score ranging from 0 to 1, which is the average of the binary answers of multiple annotators. Therefore, this relatedness score is higher when more annotators agree on the relatedness of the concept pair. However, this score cannot be directly used in a shortest-path algorithm. A high relatedness score should mean a short distance. To obtain the suitable distance measure, we introduce three mappings from the relatedness score to the distance:

$$d(A, B) = \begin{cases} 
1 - s(A, B) & \text{linear} \\
(1 - s(A, B))^2 & \text{quadratic} \\
\frac{1}{1 + s(A, B)} - 1 & \text{reciprocal}
\end{cases} \quad (1)$$

where $d(A, B) \geq 0$ is the distance between the concept $A$ and the concept $B$ and $s(A, B) > 0$ is their relatedness score. As we desired, all three kinds of distances decrease when the relatedness score increases. When $s(A, B) = 0$, $A$ and $B$ are unrelated, we simply do not add an edge to these two concepts, and thus the distance between them is implicitly set to $+\infty$. 

![ConcreteGraph Diagram]
Algorithm 1 ConcreteGraph Sampling

Input: ConcreteGraph \( G \), dataset \( D \), thresholds \( T \)
Output: newPair, newLabel

1: Sample \( A, B \) from \( G \)
2: if \( (A, B) \) in \( D \) then
   3: return NULL, NULL
4: else
5: \( path \) ← dijkstra\( (G, A, B) \)
6: if \( path \) == NULL then
7: return \( (A, B) \), “unrelated”
8: else
9: if \( \text{quality}(path) \) > \( T \) then
10: return \( (A, B) \), “related”
11: else
12: return NULL, NULL
13: end if
14: end if
15: end if

Sampling  After building the ConcreteGraph, we can then sample new concept pairs that did not exist in the original dataset. The high-level idea is to pick two random concept nodes and check whether there is a path between them. If so, we assess the quality of their relationship according to several criteria; otherwise, they are a pair of unrelated concepts. By doing those steps, we are taking advantage of the commutativity properties 2, 3, the transitivity properties 4, 5, 6.

Commutativity is used if we sample two nodes that already provided by the dataset but in a different order. For example, if \( (A, B) \) is provided and we sample \( (B, A) \). Although the ConcreteGraph is not a directed graph, commutativity is still useful in data augmentation. The reason is that Transformers are aware of permutations, \((A, B)\) and \((B, A)\) are different inputs for Transformers. Transitivity is used when the path between the two sampled concepts has at least two edges. For example, if the path is \( A \to B \to C \), then the new concept pair \( (A, C) \) is justified by the transitivity properties.

The algorithm can be expressed more formally in the pseudocode algorithm 1. The dataset \( D \) is a list of concept pairs and each of the concept pairs is a two-element tuple; To check whether there is a path between two concepts, we use Dijkstra’s algorithm \( \text{dijkstra}(\cdot, \cdot, \cdot) \) with the distance \( d(\cdot, \cdot) \) in Eq(1) as the edge weight; The \text{quality}(\cdot) function maps the path between the sampled concept pair to a set of scalars that are quality measures of the path; the “thresholds” variable \( T \) is a set of values for ensuring the quality of relationship between the sampled concept pair. In this paper, we developed two simple yet effective quality measures: path length and path score.

Path Length  Path length of a path is the number of edges. The edge weights along the path are not taken into account. When the path length is too long, the connection between the concept pair becomes “risky”. That is, when there are many edges on the path, the probability of the existence of a “bad” edge is high, which degrades the quality of the path. Thus, we prefer paths that are not too long.

Path Score  Path score is the aggregation of all relatedness scores of the edges. A high path score should be assigned to a high-quality path and vice versa. We have also developed three methods to calculate the path score: mean, minimum, and product. “Mean” is the average of all edge scores; “minimum” is to find the minimum edge score along the path; “product” is the product of all edge scores.

In Algorithm 1, after we successfully sample a new concept pair, we also need to check whether it has been found before or it is already in the dataset \( D \). The sampling is unsuccessful if the algorithm returns NULL. We run the ConcreteGraph sampling algorithm until the sampling success rate drops to near zero, which gives us a \( \sim 10\% \) data augmentation ratio in practice (without considering commutativity). If we also add \((B, A)\) as a new concept pair when we successfully sample \((A, B)\), the augmentation ratio is \( \sim 20\% \).

The reason why we have multiple ways to calculate the distance and the path score is that we will apply our ConcreteGraph data augmentation on three Transformer models. These models work best with difference distance and path score measures, as we show in our experiments.

3.3 Transformers for CRE

To test the effectiveness of our ConcreteGraph data augmentation method, we finetune three Transformer models on our augmented dataset: BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b) and XLNet (Yang et al., 2019). Their respective configuration details can be found in Appendix:Table 7, 8, 9. XLNet is the current state-of-the-art model for the STS task.

As in BERT, we use special tokens, [CLS] and [SEP], to accommodate the two concept documents. The maximum sequence length of all three models is 512 because we experiment with their base configuration. To deal with long documents, we use the following strategy to create input sequences: If both documents are longer than 255
tokens, we keep the first 255 tokens; if one of the documents is shorter than 255 tokens but the whole sequence of the two concept documents is still longer than 512 tokens, we truncate the longer document. For example, the input sequence could look like \([\text{[CLS]}, a_{1:300}, \text{[SEP]}, b_{1:210}]\) if concept \(A\) is 450-token long and concept \(B\) is 210-token long. In this case, we only keep the first 300 tokens of the concept \(A\) as it is the longer one. The resulting sequence has 300+210+2=512 tokens where “2” corresponds to the two special tokens.

A fully-connected layer with logistic activation takes as input the Transformer hidden state of \([\text{CLS}]\) and produces the prediction of the relatedness probability of the two concepts. Binary cross-entropy loss is used.

4 Experiments

4.1 Datasets

We experiment with three datasets in two languages: English and Chinese. The Wikipedia Oriented Relatedness Dataset (WORD) dataset (Ein-Dor et al., 2018) is recently developed to focus on English concepts from Wikipedia. It is made up of 19,176 pairs of concepts. The Chinese News Same Event (CNSE) dataset and the Chinese News Same Story (CNSS) dataset were together introduced by Liu et al. (2019a). They both contain news articles from the major internet news providers in China. The statistics of the three datasets can be seen in Appendix: Table 10.

4.2 Implementation Details

We set the learning rate of the Transformer blocks at \(1 \times 10^{-5}\) and the learning rate of the final fully-connected layer at \(1 \times 10^{-3}\). We used 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. Those dataset splits are fixed throughout the experiments for all models. The models are trained for at most 5 epochs, depending on the model and the dataset, and the last checkpoint is used for evaluation.

4.3 Performance Comparison on the WORD dataset

The results of the performance comparison are summarized in Table 2. We use accuracy and F1 as the two most representative performance metrics. For the WORD dataset, we include two traditional algorithms, BM25 (Robertson, 2009) and LDA (Blei et al., 2003). In the BM25-based relatedness estimation algorithm (Appendix: Algorithm 2), we use BM25 to query the source concept in the test set and check whether the target concept is a match. If the concept pair is a match, then it is a related concept pair. In the LDA-based relatedness estimation algorithm (Appendix: Algorithm 3), we first train the LDA model to learn what topics exist in the training set; then we obtain the topic distributions of concept pairs in the test set with the trained LDA model and calculate the cosine similarity between the topic distributions, which is the concept relatedness estimation.

We developed a baseline based on ELMo (Peters et al., 2018) and graph convolutional network (GCN) (Fey and Lenssen, 2019) because CIG (Liu et al., 2019a) is also a GCN but it only works on CNSE and CNSS. We build one graph for each concept document, where each node corresponds to the sentence embedding from ELMo and each edge links two nodes if they are consecutive sentences (next to each other in the original concept document) or similar sentences based on the Agglomerative Clustering algorithm provided by Scikit-learn (Pedregosa et al., 2011).

To illustrate the benefit of ConcreteGraph data augmentation, we train BERT (Devlin et al., 2019), RoBERTa (Liu et al., 2019b), and XLNet (Yang et al., 2019) on the original dataset and our augmented dataset (“w/ ConcreteGraph”). As we can see that the three Transformer models significantly

---

### Table 2: Performance comparison on WORD, CNSE and CNSS and the effect of ConcreteGraph data augmentation.

| Model      | WORD | CNSE | CNSS |
|------------|------|------|------|
|            | Acc  | F1   | Acc  | F1   | Acc  | F1   |
| LDA (Blei et al., 2003) | 49.34 | 46.44 | 51.74 | 47.54 | 51.89 | 50.29 |
| BM25 (Robertson, 2009) | 51.87 | 41.31 | 59.56 | 40.25 | 50.22 | 44.04 |
| GCN*       | 62.96 | 50.96 | 71.81 | 54.54 | 62.49 | 52.94 |
| BERT (Devlin et al., 2019) | 75.21 | 73.88 | 76.22 | 78.88 | 84.51 | 84.98 |
| w/ ConcreteGraph | (82.96, (51.4) | (80.82, (51.4) | (82.96, (51.4) | (82.96, (51.4) | (82.96, (51.4) | (82.96, (51.4) |
| RoBERTa    | 76.01 | 74.88 | 79.22 | 78.88 | 85.72 | 85.98 |
| w/ ConcreteGraph | (82.20, (53.3) | (81.31, (50.9) | (80.96, (50.8) |
| XLNet (Yang et al., 2019) | 75.58 | 74.17 | 79.22 | 78.88 | 85.72 | 85.98 |
| w/ ConcreteGraph | (82.10, (53.4) | (81.47, (51.4) | (81.34, (51.4) |

* Since CIG (Liu et al., 2019a) only works on CNSE and CNSS, we developed a sentence-level GCN model with ELMo features for WORD. The numbers in brackets show the improvement caused by ConcreteGraph data augmentation.
outperform the non-deep-learning baselines (LDA, BM25), which is expected as LDA and BM25 were initially designed for document retrieval. GCN improves over LDA and BM25 in accuracy and F1 score mainly because of the pretrained ELMo model. Our ConcreteGraph data augmentation can further improve the Transformer models by $\sim 2.5\%$ in accuracy and $\sim 1 – 5\%$ in F1.

### 4.4 Performance Comparison on the CNSE dataset and the CNSS dataset

For the CNSE dataset and the CNSS dataset, we also use BM25 (Robertson, 2009) and LDA (Blei et al., 2003) as two representative baselines among traditional methods. BERT, RoBERTa, and XLNet are also finetuned on the two datasets and their augmented versions.

The Concept Interaction Graph (CIG) model (Liu et al., 2019a) is the current state-of-the-art model. It is based on extracting a concept graph for each article. The concepts in those concept graphs are different from the CRE concepts in this paper. They are the concepts within an article, similar to named entities. One major problem with this model is that there is a limit to the size of the concept graphs, i.e., the number of concepts in a graph. If the concept graph exceeds the limit, the model simply discards the article pair. Their performance measurements excluded those excessively large graphs. By doing so, they are practically working with easier subsets of the original datasets, which causes inaccurate measurements of GIC’s performance. We corrected the accuracy and the F1 score by adding the skipped pairs as wrong predictions for accuracy and false negatives for F1 score (equivalent to treating them as false positives because of the property of harmonic mean). CIG is trained with its default training settings.

The results are also collected in Table 2 and we can see that our Transformer models that are finetuned on augmented datasets achieve better accuracy and F1 score on both the CNSE dataset and the CNSS dataset, outperforming all existing baselines. On both datasets, the accuracy is improved by $\sim 2\%$ on average and the F1 score is improved by $\sim 1 – 5\%$.

### 4.5 Ablation Study of Data Augmentation

We divided our data augmentation into two main parts: data augmentation using the commutativity properties 2, 3 and data augmentation based on the transitivity properties 4, 5 and the relaxed version of property 6. We trained our model in four different settings to study the effect of the two independent data augmentation methods: no data augmentation ("No Aug"), only commutativity data augmentation ("Comm"), only transitivity data augmentation ("Trans") and commutativity + transitivity data augmentation ("Both").

The ablation study on the WORD dataset for each Transformer is included in Table 3. We compare their performance using accuracy, F1, and area under curve (AUC). We can observe that more performance gain is brought by transitivity. Although commutativity doubles the size of the dataset while transitivity only augments the dataset by $\sim 5 – 10\%$, commutativity does not provide new concept pairs and, thus, it cannot improve the performance a lot. Commutativity is mainly helpful to compensate the fact that Transformers are aware of the permutations of the two input documents. That is, $(A, B)$ and $(B, A)$ are different inputs to Transformers.

In theory, Transformers should be able to implicitly learn the same set of new concept pairs as provided by transitivity. But in practice, this is hard to achieve as the structure of the ConcreteGraph is not easy to learn. For example, the biggest component (connected subgraph) in WORD’s ConcreteGraph has 4,301 nodes, and we can sample up to 9,247,150 concept pairs from it. Such amount of potential new concept pairs cannot be perfectly captured by Transformers implicitly.

More detailed ablation study of BERT based on additional metrics, precision, recall and specificity, is shown in Table 4. We can see that the ConcreteGraph data augmentation sacrifices precision and specificity for better recall. By the definition of those metrics (Appendix:Eq(2)), it means that there

---

**Table 3: Ablation study of each component in ConcreteGraph on the WORD dataset**

| Method | Accuracy | Precision | Recall | Specificity | F1 | AUC |
|--------|----------|-----------|--------|-------------|----|-----|
| No Aug | 75.21    | 82.16     | 65.09  | 85.56       | 72.63 | 84.53 |
| Comm   | 76.52    | 76.42     | 77.43  | 75.59       | 76.92 | 85.16 |
| Trans  | 76.56    | 74.02     | 82.64  | 80.10       | 78.09 | 85.24 |
| Both   | 78.17    | 80.10     | 75.57  | 80.81       | 75.77 | 85.53 |

**Table 4: Detailed ablation study for BERT**

| Method | Accuracy | Precision | Recall | Specificity | F1 | AUC |
|--------|----------|-----------|--------|-------------|----|-----|
| No Aug | 75.21    | 82.16     | 65.09  | 85.56       | 72.63 | 84.53 |
| Comm   | 76.52    | 76.42     | 77.43  | 75.59       | 76.92 | 85.16 |
| Trans  | 76.56    | 74.02     | 82.64  | 80.10       | 78.09 | 85.24 |
| Both   | 78.17    | 80.10     | 75.57  | 80.81       | 75.77 | 85.53 |
Table 5: Ablation study of each component in ConcreteGraph on the WORD dataset

| Model | Distance Mapping | Path Score | Score Threshold | Max Length | Acc  | F1   | AUC |
|-------|------------------|------------|----------------|------------|------|------|-----|
| BERT  | Recip Min 0.1    | 77.42      | 76.94          | 85.49      |
|       | Recip Min 0.3    | 77.55      | 76.85          | 85.31      |
|       | Recip Min 0.5    | 77.58      | 77.75          | 85.42      |
|       | Recip Min 0.7    | 78.17      | 77.77          | 85.93      |
|       | Recip Min 0.9    | 78.10      | 77.66          | 86.18      |
|       | Linear Min 0.1   | 76.53      | 77.19          | 84.52      |
|       | Linear Min 0.5   | 75.69      | 76.67          | 83.84      |
|       | Linear Min 0.7   | 76.63      | 77.43          | 85.40      |
|       | Linear Mean 0.7  | 76.06      | 73.86          | 84.91      |
|       | Linear Prod 0.7  | 78.02      | 77.12          | 86.06      |
| RoBERTa| Quad Min 0.7     | 76.63      | 75.60          | 84.49      |
| XLNet | Recip Min 0.3    | 78.21      | 78.21          | 85.48      |
|       | Linear Min 0.3   | 76.56      | 74.02          | 82.04      |
|       | Linear Mean 0.3  | 76.56      | 76.09          | 84.25      |

Table 5: Ablation study of each component in ConcreteGraph on the WORD dataset

are more false positives but much less false negatives when we use data augmentation. That is, the transformers are more lenient on making positive predictions. Such trade-off is worthwhile because, for instance, the harmonic mean of precision and recall (F1) of “Both” becomes higher than that of “No Aug”.

5 Effect of Path Quality Functions

We use quality functions to measure the quality of the relationships between sampled concept pairs. Detailed ablation study for each component in our ConcreteGraph data augmentation algorithm is included in Table 5. The highlight colors show the changed component, and the other components in those rows remain the same. For example, only the score threshold (in blue) is changed in the first 5 rows.

The distance mapping affects which path the Dijkstra’s algorithm chooses given two sampled concept pairs (highlighted in yellow), which, in turn, can influence the path score. We experiment with three approaches to calculate the path score (highlighted in orange). Once we obtain the path score, we use a score threshold to filter out low-quality paths (highlighted in blue). We also filter the paths based on their path length, which is the number of edges on the path regardless of the edge weights or the edge scores (highlighted in green).

According to the experiment results, the reciprocal distance mapping outperforms the other two mappings. One unique property of reciprocal distance mapping is that when the edge score approaches 0, the distance approaches infinity. Therefore, it penalizes low-score edges much more than the other two distance mappings do. Thus, it is the best distance mapping for BERT and XLNet. Quadratic distance mapping is limited to the range from 0 to 1, which is the same as linear distance mapping, but it also penalizes edges more when the score is close to 0.

For calculating the path score, all three models perform the best when minimum edge score is used. This is reasonable because whenever there is an edge with a low relatedness score, the connectivity between the two concept nodes becomes weak. Other path score measures, product and mean, might ignore a low-score edge if other edges on the path all have high scores. When minimum edge score is combined with a score threshold, we are able to remove weak relationships and only keep the high-quality paths.

Maximum length is an independent quality measure of the score threshold. It is simply the number of edges on the path. By limiting the maximum length, we eliminate long dependencies, which are more “risky” than short paths. That is, it is more likely to have a low-score edge when the path is long.

Table 6 shows the performance when no quality function is used (“Without Quality Thresholds”). When we run a tenfold data augment, the performance in fact decreases significantly, which indicates that not every new concept pair is beneficial to the performance. If we augment the dataset by the same amount as the “With Quality Thresholds” setting (3rd row and 4th row), the performance is not as bad. But by including higher-quality concept pairs, we can boost the performance even further (5th row and 6th row).

6 Conclusion

Concept relatedness estimation is a recently introduced task that has a wide range of applications. Many typical NLP data augmentation methods can be applied on CRE, but the unique properties of CRE are underexplored. ConcreteGraph takes advantage of such CRE properties and can boost the performance of Transformers even further.
References

Ch. Aswani Kumar and S. Srinivas. 2010. Concept lattice reduction using fuzzy k-means clustering. Expert Systems with Applications, 37(3):2696–2704.

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

Michael Busch, Krishna Gade, Brian Larson, Patrick Lok, Samuel Luckenbill, and Jimmy Lin. 2012. Earlybird: Real-time search at twitter. In 2012 IEEE 28th International Conference on Data Engineering, pages 1360–1369.

Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. arXiv preprint arXiv:1708.00055.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the properties of neural machine translation: Encoder-decoder approaches. In SSST@EMNLP, pages 103–111. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In NAACL-HLT (1), pages 4171–4186. Association for Computational Linguistics.

Liat Ein-Dor, Alon Halfon, Yoav Kantor, Ran Levy, Yosi Mass, Ruty Rinott, Eyal Shnarch, and Noam Slonim. 2018. Semantic relatedness of wikipedia concepts - benchmark data and a working solution. In LREC. European Language Resources Association (ELRA).

Matthias Fey and Jan E. Lenssen. 2019. Fast graph representation learning with PyTorch Geometric. In ICLR Workshop on Representation Learning on Graphs and Manifolds.

Anna Formica. 2006. Ontology-based concept similarity in formal concept analysis. Information sciences, 176(18):2624–2641.

Anna Formica. 2008. Concept similarity in formal concept analysis: An information content approach. Knowledge-based systems, 21(1):80–87.

Jike Ge and Yuhui Qiu. 2008. Concept similarity matching based on semantic distance. In SKG, pages 380–383. IEEE Computer Society.

Sepp Hochreiter. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. Int. J. Uncertain. Fuzziness Knowl. Based Syst., 6(2):107–116.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. Neural Comput., 9(8):1735–1780.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In Advances in neural information processing systems, pages 2042–2050.

Shaoxiong Ji, Shuiru Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. 2022. A survey on knowledge graphs: Representation, acquisition, and applications. IEEE Trans. Neural Networks Learn. Syst., 33(2):494–514.

Jyun-Yu Jiang, Mingyang Zhang, Cheng Li, Michael Bendersky, Nadav Golbandi, and Marc Najork. 2019. Semantic text matching for long-form documents. In The World Wide Web Conference, WWW ’19, page 795–806, New York, NY, USA. Association for Computing Machinery.

Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. 2015. Deep learning. nature, 521(7553):436–444.

Bohan Li, Yutai Hou, and Wanxiang Che. 2022. Data augmentation approaches in natural language processing: A survey. AI Open.

Wenjie Li and Qixiang Xia. 2011. A method of concept similarity computation based on semantic distance. Procedia Engineering, 15:3854–3859.

Bang Liu, Di Niu, Haojie Wei, Jinghong Lin, Yancheng He, Kunfeng Lai, and Yu Xu. 2019a. Matching article pairs with graphical decomposition and convolutions. In ACL (1), pages 6284–6294. Association for Computational Linguistics.

Yinhua Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019b. Roberta: A robustly optimized BERT pretraining approach. CoRR, abs/1907.11692.

Luigi Lombardi and Giuseppe Sartori. 2006. Concept similarity: An abstract relevance classes approach. In The 7th International Conference on Cognitive Modeling, Trieste, pages 190–195.

Jirapond Muangprathub, Siriwon Kajornkasirat, and Apirat Wanichsombat. 2021. Document plagiarism detection using a new concept similarity in formal concept analysis. Journal of Applied Mathematics, 2021:1–10.

Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text matching as image recognition. In AAAI, pages 2793–2799. AAAI Press.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. Journal of Machine Learning Research, 12:2825–2830.
A Model Configurations

Model specification for the transformer models.

| Model | Details | EN | CN |
|-------|---------|----|----|
| **ConcreteBERT architecture details (BERT base)** | | | |
| Hidden Activation | GELU | | | |
| Dropout Probability | 0.1 | | | |
| Hidden Size | 768 | | | |
| Intermediate Size | 3072 | | | |
| Max Position Embeddings | 512 | | | |
| Num Attention Heads | 12 | | | |
| Num Hidden Layers | 12 | | | |
| Vocab Size | 30522 (EN) | 21128 (CN) | | |

| **RoBERTa architecture details (RoBERTa base)** | | | |
| Hidden Activation | GELU | | | |
| Dropout Probability | 0.1 | | | |
| Hidden Size | 768 | | | |
| Intermediate Size | 3072 | | | |
| Max Position Embeddings | 514 | | | |
| Num Attention Heads | 12 | | | |
| Num Hidden Layers | 12 | | | |
| Vocab Size | 50265 (EN) | 21128 (CN) | | |

| **XLNet architecture details (XLNet base)** | | | |
| Hidden Activation | GELU | | | |
| Dropout Probability | 0.1 | | | |
| Hidden Size | 768 | | | |
| Intermediate Size | 3072 | | | |
| Num Attention Heads | 12 | | | |
| Num Hidden Layers | 12 | | | |
| Attention Head Size | 64 | | | |
| Vocab Size | 32000 (EN) | 32000 (CN) | | |

Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237. Association for Computational Linguistics.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional neural tensor network architecture for community-based question answering. In *IJCAI*, pages 1305–1311. AAAI Press.

Juan Ramos et al. 2003. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, volume 242, pages 29–48. Citeseer.

Zaragoza Robertson. 2009. The probabilistic relevance framework: BM25 and beyond. *Found. Trends Inf. Retr.*, 3(4):333–389.

Mike Schuster and Kuldip K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45(11):2673–2681.

Connor Shorten and Taghi M. Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *J. Big Data*, 6:60.

J. Teevan, D. Ramage, and M. Morris. 2011. #twittersearch: a comparison of microblog search and web search. In *WSDM ’11*.

B. Thijs. 2019. Paragraph-based intra- and inter-document similarity using neural vector paragraph embeddings. In *ISSI*.

George Tsatsaronis, Iraklis Varlamis, and Michalis Vazirgiannis. 2014. Text relatedness based on a word thesaurus. *CoRR*, abs/1401.5699.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI*, pages 2835–2841. AAAI Press.

Lidong Wang and Xiaodong Liu. 2008. A new model of evaluating concept similarity. *Knowledge-Based Systems*, 21(8):842–846.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *EMNLP/IJCNLP (1)*, pages 6381–6387. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS*, pages 5754–5764.
B Metrics

\[
\text{precision} = \frac{TP}{TP + FP}, \\
\text{recall} = \text{sensitivity} = \frac{TP}{TP + FN}, \\
\text{specificity} = \frac{TN}{TN + FP}, \\
\text{F1 score} = \frac{2 \cdot \text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}},
\]

where TP = true positives, FP = false positives, TN = true negatives, FN = false negatives.

C Dataset Statistics

Table 10: Statistics of the WORD dataset, the CNSE dataset and the CNSS dataset

|                  | WORD  | CNSE  | CNSS  |
|------------------|-------|-------|-------|
| Total Size       | 19,176| 29,063| 33,503|
| Positive Count   | 10,028| 12,865| 16,887|
| Negative Count   | 9,148 | 16,198| 16,616|
| Training Size    | 11,563| 20,342| 23,449|
| Validation Size  | 1,287 | 5,812 | 6,700 |
| Test Size        | 6,302 | 2,907 | 3,351 |

D Pseudo-code for the Algorithms Appeared in this Paper

The pseudo-code for baselines, LDA and BM25.

Algorithm 2 BM25-based relatedness estimation

Input: testPairs \((A_1, B_1, score_1), (A_2, B_2, score_2), (A_3, B_3, score_3) \ldots, (A_n, B_n, score_n)\), threshold \(T\)

Output: testAccuracy

1: \(score = \{\}\)
2: \(Documents = [A_1, B_1, A_2, B_2, \ldots, A_n, B_n]\)
3: testNum = 0
4: predictTrue = 0
5: for every pair \((A_i, B_i, score_i)\) in testPairs do
6: \(q = A_i\)
7: \(score[A_i] = \[]\)
8: for every \(d\) in Documents do
9: \(newScore = BM25(d, q)\)
10: \(score[A_i].append(newScore)\)
11: testNum + = 1
12: end for
13: \(q = B_i\)
14: \(score[B_i] = \[]\)
15: for every \(d\) in Documents do
16: \(newScore = BM25(d, q)\)
17: \(score[B_i].append(newScore)\)
18: testNum + = 1
19: end for
20: Normalize(score[A_i])
21: Normalize(score[B_i])
22: if \(score_i > 0\) then
23: \(\) if \(score[A_i][2i - 1] > T\) then
24: predictTrue + = 1
25: end if
26: \(\) if \(score[B_i][2(i - 1)] > T\) then
27: predictTrue + = 1
28: end if
29: else
30: \(\) if \(score[A_i][2i - 1] < T\) then
31: predictTrue + = 1
32: end if
33: \(\) if \(score[B_i][2(i - 1)] < T\) then
34: predictTrue + = 1
35: end if
36: end if
37: end for
38: testAccuracy = predictTrue/testNum
return testAccuracy

To calculate the score for a document, we use
the following function based on BM25:

\[
BM25(d, q) = \sum_i \log \frac{N - n(q_i) + 0.5}{n(q_i) + 0.5} \\
\frac{(k_1 + 1) \cdot tf(q_i, d)}{k_1(1 - b + b \cdot \frac{L_d}{L_{avg}}) + tf(q_i, d)} \cdot \frac{(k_2 + 1) \cdot tf(q_i, q)}{k_2 + tf(q_i, q)}
\]

(3)

where \(N\) is the total number of Documents, \(q_i\) is the i-th token in document \(q\), \(n(q_i)\) is the number of documents contains token \(q_i\), \(k_1, k_2, b\) are three parameters, \(tf(a, b)\) is the frequency of token \(a\) in \(b\), \(L_d\) is the length of document \(d\) and \(L_{avg}\) is the average length of all documents.

**Algorithm 3** LDA based relatedness estimation

**Input:** trainPairs \{\((A_1, B_1), (A_2, B_2), (A_3, B_3)\) \ldots\}, testPairs \{\((C_1, D_1, score_1), (C_2, D_2, score_2), (C_3, D_3) \ldots, (C_n, D_n)\)\}, threshold \(T\)

**Output:** testAccuracy

\[
\text{▷ First, we clean and tokenize the text in trainPairs and testPairs}
\]

1: \(trainDf = [A^t_1, B^t_1, A^t_2, B^t_2, \ldots, A^t_n, B^t_n]\)

2: \(testDf = [(C^t_1, D^t_1, score_1), (C^t_2, D^t_2, score_2), \ldots, (C^t_n, D^t_n, score_n)]\)

3: \(ldaModel = \text{trainLda}(trainDf)\)

4: \(testNum = 0\)

5: \(predictTrue = 0\)

6: \(\text{for every pair } (C^t_i, D^t_i, score_i) \text{ in testDf do } \text{▷}
\]

\[
\text{We can obtain the topic distribution with the model trained on docs from trainPairs}
\]

7: \(\text{distributionC} = ldaModel(C^t_i)\)

8: \(\text{distributionD} = ldaModel(D^t_i) \text{ ▷}
\]

\[
\text{We use the cosine similarity of the two topic distribution for the related estimation for the doc pair}
\]

9: \(\text{similarity} = \frac{\text{distributionC} \cdot \text{distributionD}}{||\text{distributionC}|| \cdot ||\text{distributionD}||}\)

10: \(testNum + = 1\)

11: \(\text{if } score_i > 0 \text{ then}
\]

12: \(\text{if } \text{similarity} >= T \text{ then}
\]

13: \(predictTrue + = 1\)

14: \(\text{end if}
\]

15: \(\text{else}
\]

16: \(\text{if } \text{similarity} < T \text{ then}
\]

17: \(predictTrue + = 1\)

18: \(\text{end if}
\]

19: \(\text{end if}
\]

20: \(\text{end for}
\]

21: \(\text{testAccuracy} = predictTrue/testNum\)

return testAccuracy