Matching Article Pairs with Graphical Decomposition and Convolutions

Bang Liu[†], **Di Niu**[†], **Haojie Wei**[‡], **Jinghong Lin**[‡], **Yancheng He**[‡], **Kunfeng Lai**[‡], **Yu Xu**[‡]
[†]University of Alberta, Edmonton, AB, Canada
{bang3, dniu}@ualberta.ca
[‡]Platform and Content Group, Tencent, Shenzhen, China
{fayewei, daphnelin, collinhe, calvinlai, henrysxu}@tencent.com

Abstract

Identifying the relationship between two articles, e.g., whether two articles published from different sources describe the same breaking news, is critical to many document understanding tasks. Existing approaches for modeling and matching sentence pairs do not perform well in matching *longer* documents, which embody more complex interactions between the enclosed entities than a sentence does. To model article pairs, we propose the *Concept Interaction Graph* to represent an article as a graph of concepts. We then match a pair of articles by comparing the sentences that enclose the same concept vertex through a series of encoding techniques, and aggregate the matching signals through a graph convolutional network. To facilitate the evaluation of long article matching, we have created two datasets, each consisting of about 30K pairs of *breaking news* articles covering diverse topics in the open domain. Extensive evaluations of the proposed methods on the two datasets demonstrate significant improvements over a wide range of state-of-the-art methods for natural language matching.

1 Introduction

Identifying the relationship between a pair of articles is an essential natural language understanding task, which is critical to news systems and search engines. For example, a news system needs to cluster various articles on the Internet reporting the same breaking news (probably in different ways of wording and narratives), remove redundancy and form storylines (Shahaf et al., 2013; Liu et al., 2017; Zhou et al., 2015; Vossen et al., 2015; Bruggermann et al., 2016). The rich semantic and logic structures in longer documents have made it a different and more challenging task to match a pair of articles than to match a pair of sentences or a query-document pair in information retrieval.

Traditional term-based matching approaches estimate the semantic distance between a pair of text objects via unsupervised metrics, e.g., via TF-IDF vectors, BM25 (Robertson et al., 2009), LDA (Blei et al., 2003) and so forth. These methods have achieved success in query-document matching, information retrieval and search. In recent years, a wide variety of deep neural network models have also been proposed for text matching (Hu et al., 2014; Qiu and Huang, 2015; Wan et al., 2016; Pang et al., 2016), which can capture the semantic dependencies (especially sequential dependencies) in natural language through layers of recurrent or convolutional neural networks. However, existing deep models are mainly designed for matching sentence pairs, e.g., for paraphrase identification, answer selection in question-answering, omitting the complex interactions among keywords, entities or sentences that are present in a longer article. Therefore, article pair matching remains under-explored in spite of its importance.

In this paper, we apply the divide-and-conquer philosophy to matching a pair of articles and bring deep text understanding from the currently dominating sequential modeling of language elements to a new level of graphical document representation, which is more suitable for longer articles. Specifically, we have made the following contributions:

First, we propose the so-called *Concept Interaction Graph* (CIG) to represent a document as a weighted graph of concepts, where each concept vertex is either a keyword or a set of tightly connected keywords. The sentences in the article associated with each concept serve as the features for local comparison to the same concept appearing in another article. Furthermore, two concept vertices in an article are also connected by a weighted edge which indicates their interaction strength. The CIG does not only capture the essen-

tial semantic units in a document but also offers a way to perform anchored comparison between two articles along the common concepts found.

Second, we propose a divide-and-conquer framework to match a pair of articles based on the constructed CIGs and graph convolutional networks (GCNs). The idea is that for each concept vertex that appears in both articles, we first obtain the local matching vectors through a range of text pair encoding schemes, including both neural encoding and term-based encoding. We then aggregate the local matching vectors into the final matching result through graph convolutional layers (Kipf and Welling, 2016; Defferrard et al., 2016). In contrast to RNN-based sequential modeling, our model factorizes the matching process into local matching sub-problems on a graph, each focusing on a different concept, and by using GCN layers, generates matching results based on a holistic view of the entire graph.

Although there exist many datasets for sentence matching, the semantic matching between longer articles is a largely unexplored area. To the best of our knowledge, to date, there does not exist a labeled public dataset for long document matching. To facilitate evaluation and further research on document and especially news article matching, we have created *two labeled datasets*[1], one annotating whether two news articles found on Internet (from different media sources) report the same breaking news event, while the other annotating whether they belong to the same news story (yet not necessarily reporting the same breaking news event). These articles were collected from major Internet news providers in China, including Tencent, Sina, WeChat, Sohu, etc., covering diverse topics, and were labeled by professional editors. Note that similar to most other natural language matching models, all the approaches proposed in this paper can easily work on other languages as well.

Through extensive experiments, we show that our proposed algorithms have achieved significant improvements on matching news article pairs, as compared to a wide range of state-of-the-art methods, including both term-based and deep text matching algorithms. With the same encoding or term-based feature representation of a pair of articles, our approach based on graphical decomposition and convolutions can improve the classification accuracy by 17.31% and 23.09% on the two datasets, respectively.

[1]Our code and datasets are available at: https://github.com/BangLiu/ArticlePairMatching

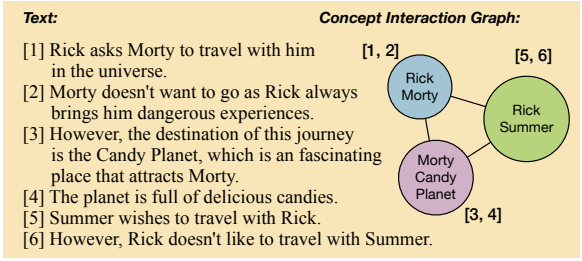

Figure 1: An example to show a piece of text and its Concept Interaction Graph representation.

2 Concept Interaction Graph

In this section, we present our *Concept Interaction Graph* (CIG) to represent a document as an *undirected weighted graph*, which decomposes a document into subsets of sentences, each subset focusing on a different *concept*. Given a document $\mathcal{D}$, a CIG is a graph G_D, where each vertex in G_D is called a *concept*, which is a keyword or a set of highly correlated keywords in document $\mathcal{D}$. Each sentence in $\mathcal{D}$ will be attached to the *single* concept vertex that it is the most related to, which most frequently is the concept the sentence mentions. Hence, vertices will have their own sentence sets, which are disjoint. The weight of the edge between a pair of concepts denotes how much the two concepts are related to each other and can be determined in various ways.

As an example, Fig. 1 illustrates how we convert a document into a Concept Interaction Graph. We can extract keywords *Rick*, *Morty*, *Summer*, and *Candy Planet* from the document using standard keyword extraction algorithms, e.g., TextRank (Mihalcea and Tarau, 2004). These keywords are further clustered into three concepts, where each concept is a subset of highly correlated keywords. After grouping keywords into concepts, we attach each sentence in the document to its most related concept vertex. For example, in Fig. 1, sentences 1 and 2 are mainly talking about the relationship between *Rick* and *Morty*, and are thus attached to the concept (*Rick*, *Morty*). Other sentences are attached to vertices in a similar way. The attachment of sentences to concepts naturally dissects the original document into multiple disjoint sentence subsets. As a result, we have represented the original document with a graph of key concepts, each with a sentence subset, as well as

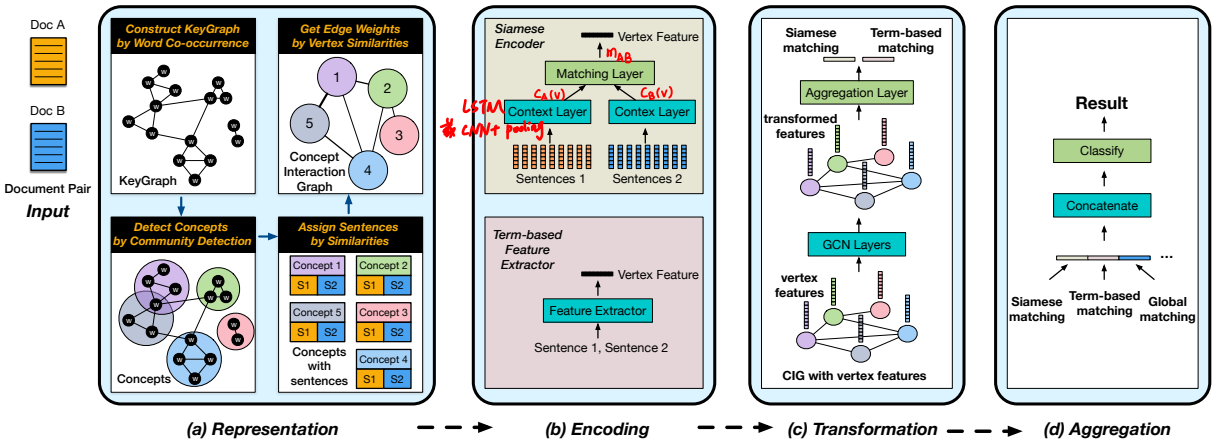

Figure 2: An overview of our approach for constructing the *Concept Interaction Graph* (CIG) from a pair of documents and classifying it by Graph Convolutional Networks.

the interaction topology among them.

Fig 2 (a) illustrates the construction of CIGs for a pair of documents aligned by the discovered concepts. Here we first describe the detailed steps to construct a CIG for a single document:

KeyGraph Construction. Given a document $\mathcal{D}$, we first extract the named entities and keywords by TextRank (Mihalcea and Tarau, 2004). After that, we construct a keyword co-occurrence graph, called *KeyGraph*, based on the set of found keywords. Each keyword is a vertex in the KeyGraph. We connect two keywords by an edge if they co-occur in a same sentence.

We can further improve our model by performing co-reference resolution and synonym analysis to merge keywords with the same meaning. However, we do not apply these operations due to the time complexity.

Concept Detection (Optional). The structure of KeyGraph reveals the connections between keywords. If a subset of keywords are highly correlated, they will form a densely connected subgraph in the KeyGraph, which we call a *concept*. Concepts can be extracted by applying community detection algorithms on the constructed KeyGraph. Community detection is able to split a KeyGraph G_{key} into a set of communities $C = \{\mathcal{C}_1, \mathcal{C}_2, ..., \mathcal{C}_{|C|}\}$, where each community $\mathcal{C}_i$ contains the keywords for a certain concept. By using overlapping community detection, each keyword may appear in multiple concepts. As the number of concepts in different documents varies a lot, we utilize the *betweenness centrality score* based algorithm (Sayyadi and Raschid, 2013) to detect keyword communities in KeyGraph.

Note that this step is optional, i.e., we can also use each keyword directly as a concept. The benefit brought by concept detection is that it reduces the number of vertices in a graph and speeds up matching, as will be shown in Sec. 4.

Sentence Attachment. After the concepts are discovered, the next step is to group sentences by concepts. We calculate the cosine similarity between each sentence and each concept, where sentences and concepts are represented by TF-IDF vectors. We assign each sentence to the concept which is the most similar to the sentence. Sentences that do not match any concepts in the document will be attached to a *dummy vertex* that does not contain any keywords.

Edge Construction. To construct edges that reveal the correlations between different concepts, for each vertex, we represent its sentence set as a concatenation of the sentences attached to it, and calculate the edge weight between any two vertices as the TF-IDF similarity between their sentence sets. Although edge weights may be decided in other ways, our experience shows that constructing edges by TF-IDF similarity generates a CIG that is more densely connected.

When performing article pair matching, the above steps will be applied to a pair of documents $\mathcal{D}_A$ and $\mathcal{D}_B$, as is shown in Fig. 2 (a). The only additional step is that we align the CIGs of the two articles by the concept vertices, and for each common concept vertex, merge the sentence sets from $\mathcal{D}_A$ and $\mathcal{D}_B$ for local comparison.

3 Article Pair Matching through Graph Convolutions

Given the merged CIG G_{AB} of two documents $\mathcal{D}_A$ and $\mathcal{D}_B$ described in Sec. 2, we match a pair of ar-

ticles in a "divide-and-conquer" manner by matching the sentence sets from $\mathcal{D}_A$ and $\mathcal{D}_B$ associated with each concept and aggregating local matching results into a final result through multiple graph convolutional layers. Our approach overcomes the limitation of previous text matching algorithms, by extending text representation from a sequential (or grid) point of view to a graphical view, and can therefore better capture the rich semantic interactions in longer text.

Fig. 2 illustrates the overall architecture of our proposed method, which consists of four steps: a) representing a pair of documents by a single merged CIG, b) learning multi-viewed matching features for each concept vertex, c) structurally transforming local matching features by graph convolutional layers, and d) aggregating local matching features to get the final result. Steps (b)-(d) can be trained end-to-end.

Encoding Local Matching Vectors. Given the merged CIG G_{AB}, our first step is to learn an appropriate *matching vector* of a fixed length for each individual concept $v \in G_{AB}$ to express the semantic similarity between $\mathcal{S}_A(v)$ and $\mathcal{S}_B(v)$, the sentence sets of concept v from documents $\mathcal{D}_A$ and $\mathcal{D}_B$, respectively. This way, the matching of two documents is converted to match the pair of sentence sets on each vertex of G_{AB}. Specifically, we generate local matching vectors based on both neural networks and term-based techniques.

Siamese Encoder: we apply a Siamese neural network encoder (Neculoiu et al., 2016) onto each vertex $v \in G_{AB}$ to convert the word embeddings (Mikolov et al., 2013) of $\{\mathcal{S}_A(v), \mathcal{S}_B(v)\}$ into a fixed-sized hidden feature vector $\mathbf{m}_{AB}(v)$, which we call the *match vector*.

We use a Siamese structure to take $\mathcal{S}_A(v)$ and $\mathcal{S}_B(v)\}$ (which are two sequences of word embeddings) as inputs, and encode them into two context vectors through the context layers that share the same weights, as shown in Fig. 2 (b). The context layer usually contains one or multiple bi-directional LSTM (BiLSTM) or CNN layers with max pooling layers, aiming to capture the contextual information in $\mathcal{S}_A(v)$ and $\mathcal{S}_B(v)\}$.

Let $\mathbf{c}_A(v)$ and $\mathbf{c}_B(v)$ denote the context vectors obtained for $\mathcal{S}_A(v)$ and $\mathcal{S}_B(v)$, respectively. Then, the matching vector $\mathbf{m}_{AB}(v)$ for vertex v is given by the subsequent aggregation layer, which concatenates the element-wise absolute difference and the element-wise multiplication of the two context vectors, i.e.,

$$\mathbf{m}_{AB}(v) = (|\mathbf{c}_A(v) - \mathbf{c}_B(v)|, \mathbf{c}_A(v) \circ \mathbf{c}_B(v)), \quad (1)$$

where $\circ$ denotes Hadamard product.

Term-based Similarities: we also generate another matching vector for each v by directly calculating term-based similarities between $\mathcal{S}_A(v)$ and $\mathcal{S}_B(v)$, based on 5 metrics: the TF-IDF cosine similarity, TF cosine similarity, BM25 cosine similarity, Jaccard similarity of 1-gram, and Ochiai similarity measure. These similarity scores are concatenated into another matching vector $\mathbf{m}'_{AB}(v)$ for v, as shown in Fig. 2 (b).

Matching Aggregation via GCN The local matching vectors must be aggregated into a final matching score for the pair of articles. We propose to utilize the ability of the Graph Convolutional Network (GCN) filters (Kipf and Welling, 2016) to capture the patterns exhibited in the CIG G_{AB} at multiple scales. In general, the input to the GCN is a graph $G = (\mathcal{V}, E)$ with N vertices $v_i \in \mathcal{V}$, and edges $e_{ij} = (v_i, v_j) \in E$ with weights w_{ij}. The input also contains a vertex feature matrix denoted by $X = \{\mathbf{x}_i\}_{i=1}^N$, where $\mathbf{x}_i$ is the *feature vector* of vertex v_i. For a pair of documents $\mathcal{D}_A$ and $\mathcal{D}_B$, we input their CIG G_{AB} (with N vertices) with a (concatenated) matching vector on each vertex into the GCN, such that the feature vector of vertex v_i in GCN is given by

$$\mathbf{x}_i = (\mathbf{m}_{AB}(v_i), \mathbf{m}'_{AB}(v_i)).$$

Now let us briefly describe the GCN layers (Kipf and Welling, 2016) used in Fig. 2 (c). Denote the weighted adjacency matrix of the graph as $A \in \mathbb{R}^{N \times N}$ where $A_{ij} = w_{ij}$ (in CIG, it is the TF-IDF similarity between vertex i and j). Let D be a diagonal matrix such that $D_{ii} = \sum_j A_{ij}$. The input layer to the GCN is $H^{(0)} = X$, which contains the original vertex features. Let $H^{(l)} \in \mathbb{R}^{N \times M_l}$ denote the matrix of *hidden representations* of the vertices in the l^{th} layer. Then each GCN layer applies the following graph convolutional filter onto the previous hidden representations:

$$H^{(l+1)} = \sigma(\tilde{D}^{-\frac{1}{2}}\tilde{A}\tilde{D}^{-\frac{1}{2}}H^{(l)}W^{(l)}), \quad (2)$$

where $\tilde{A} = A + I_N$, I_N is the identity matrix, and $\tilde{D}$ is a diagonal matrix such that $\tilde{D}_{ii} = \sum_j \tilde{A}_{ij}$. They are the adjacency matrix and the degree matrix of graph G, respectively.

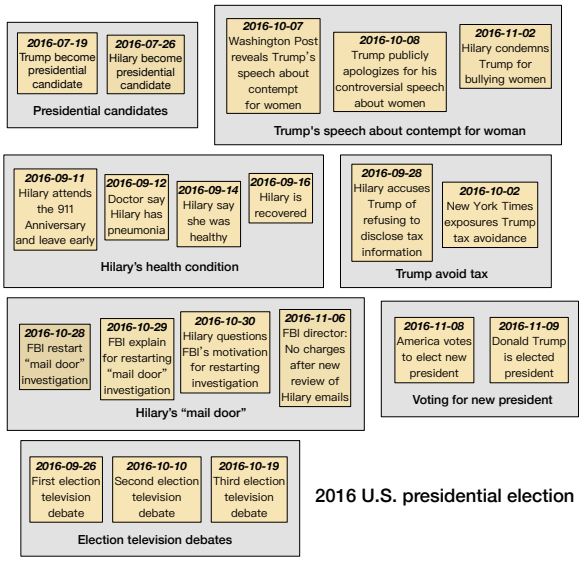

Figure 3: The events contained in the story "2016 U.S. presidential election".

$W^{(l)}$ is the trainable weight matrix in the l^{th} layer. $\sigma(\cdot)$ denotes an activation function such as sigmoid or ReLU function. Such a graph convolutional rule is motivated by the first-order approximation of localized spectral filters on graphs (Kipf and Welling, 2016) and when applied recursively, can extract interaction patterns among vertices.

Finally, the hidden representations in the final GCN layer is merged into a single vector (called a graphically merged matching vector) of a fixed length, denoted by $\mathbf{m}_{AB}$, by taking the mean of the hidden vectors of all vertices in the last layer. The final matching score will be computed based on $\mathbf{m}_{AB}$, through a classification network, e.g., a multi-layered perceptron (MLP).

In addition to the graphically merged matching vector $\mathbf{m}_{AB}$ described above, we may also append other global matching features to $\mathbf{m}_{AB}$ to expand the feature set. These additional global features can be calculated, e.g., by encoding two documents directly with state-of-the-art language models like BERT (Devlin et al., 2018) or by directly computing their term-based similarities. However, we show in Sec. 4 that such global features can hardly bring any more benefit to our scheme, as the graphically merged matching vectors are already sufficiently expressive in our problem.

4 Evaluation

Tasks. We evaluate the proposed approach on the task of ***identifying whether a pair of news articles report the same breaking news (or event) and whether they belong to the same series of news story***, which is motivated by a real-world news app. In fact, the proposed article pair matching schemes have been deployed in the anonymous news app for news clustering, with more than 110 millions of daily active users.

Note that traditional methods to document clustering include unsupervised text clustering and text classification into predefined topics. However, a number of breaking news articles emerge on the Internet everyday with their topics/themes unknown, so it is not possible to predefine their topics. Thus, supervised text classification cannot be used here. It is even impossible to determine how many news clusters there exist. Therefore, the task of classifying whether two news articles are reporting the same breaking news event or belong to the same story is critical to news apps and search engines for clustering, redundancy removal and topic summarization.

In our task, an "event" refers to a piece of breaking news on which multiple media sources may publish articles with different narratives and wording. Furthermore, a "story" consists of a series of logically related breaking news events. It is worth noting that our objective is fundamentally different from the traditional event coreference literature, e.g., (Bejan and Harabagiu, 2010; Lee et al., 2013, 2012) or SemEval-2018 Task 5 (Counting Events) (Postma et al., 2018), where the task is to detect all the events (or in fact, "actions" like shooting, car crashes) a document mentions.

In contrast, although a news article may mention multiple entities and even previous physical events, the "event" in our dataset always refers to the breaking news that the article intends to report or the incident that triggers the media's coverage. And our task is to identify whether two articles intend to report the same breaking news. For example, two articles "University of California system libraries break off negotiations with Elsevier, will no longer order their journals" and "University of California Boycotts Publishing Giant Elsevier" from two different sources are apparently intended to report the same breaking news event of UC dropping subscription to Elsevier, although other actions may be peripherally mentioned in these articles, e.g., "eight months of unsuccessful negotiations." In addition, we do not attempt to perform reading comprehension question answering tasks either, e.g., finding out how many killing incidents or car crashes there are in a year (SemEval-2018 Task 5 (Postma et al., 2018)).

Table 1: Description of evaluation datasets.

Dataset	Pos Samples	Neg Samples	Train	Dev	Test
CNSE	12865	16198	17438	5813	5812
CNSS	16887	16616	20102	6701	6700

As a typical example, Fig. 3 shows the events contained in the story *2016 U.S. presidential election*, where each tag shows a breaking news event possibly reported by multiple articles with different narratives (articles not shown here). We group highly coherent events together. For example, there are multiple events about Election television debates. One of our objectives is to identify whether two news articles report the same event, e.g., a yes when they are both reporting *Trump and Hilary's first television debate*, though with different wording, or a no, when one article is reporting *Trump and Hilary's second television debate* while the other is talking about *Donald Trump is elected president*.

Datasets. To the best of our knowledge, there is no publicly available dataset for long document matching tasks. We created two datasets: the Chinese News Same Event dataset (CNSE) and Chinese News Same Story dataset (CNSS), which are labeled by professional editors. They contain long Chinese news articles collected from major Internet news providers in China, covering diverse topics in the open domain. The CNSE dataset contains $29,063$ pairs of news articles with labels representing whether a pair of news articles are reporting about the same breaking news event. Similarly, the CNSS dataset contains $33,503$ pairs of articles with labels representing whether two documents fall into the same news story. The average number of words for all documents in the datasets is 734 and the maximum value is 21791.

In our datasets, we only labeled the *major* event (or story) that a news article is reporting, since in the real world, each breaking news article on the Internet must be intended to report some specific breaking news that has just happened to attract clicks and views. Our objective is to determine whether two news articles intend to report the same breaking news.

Note that the negative samples in the two datasets are not randomly generated: we select document pairs that contain similar keywords, and exclude samples with TF-IDF similarity below a certain threshold. The datasets have been made publicly available for research purpose.

Table 1 shows a detailed breakdown of the two datasets. For both datasets, we use 60% of all the samples as the training set, 20% as the development (validation) set, and the remaining 20% as the test set. We carefully ensure that different splits do not contain any overlaps to avoid data leakage. The metrics used for performance evaluation are the accuracy and F1 scores of binary classification results. For each evaluated method, we perform training for 10 epochs and then choose the epoch with the best validation performance to be evaluated on the test set.

Baselines. We test the following baselines:

- *Matching by representation-focused or interaction-focused deep neural network models*: DSSM (Huang et al., 2013), C-DSSM (Shen et al., 2014), DUET (Mitra et al., 2017), MatchPyramid (Pang et al., 2016), ARC-I (Hu et al., 2014), ARC-II (Hu et al., 2014). We use the implementations from MatchZoo (Fan et al., 2017) for the evaluation of these models.

- *Matching by term-based similarities*: BM25 (Robertson et al., 2009), LDA (Blei et al., 2003) and SimNet (which is extracting the five text-pair similarities mentioned in Sec. 3 and classifying by a multi-layer feedforward neural network).

- *Matching by a large-scale pre-training language model*: BERT (Devlin et al., 2018).

Note that we focus on the capability of long text matching. Therefore, we do not use any short text information, such as titles, in our approach or in any baselines. In fact, the "relationship" between two documents is not limited to "whether the same event or not". Our algorithm is able to identify a general relationship between documents, e.g., whether two episodes are from the same season of a TV series. The definition of the relationship (e.g., same event/story, same chapter of a book) is solely defined and supervised by the labeled training data. For these tasks, the availability of other information such as titles can not be assumed.

As shown in Table 2, we evaluate different variants of our own model to show the effect of different sub-modules. In model names, "CIG" means that in CIG, we directly use keywords as concepts without community detection, whereas "CIG$_{cd}$" means that each concept vertex in the CIG contains a set of keywords grouped via community detection. To generate the matching vector on each

Table 2: Accuracy and F1-score results of different algorithms on CNSE and CNSS datasets.

Baselines	CNSE		CNSS		Our models	CNSE		CNSS	
	Acc	F1	Acc	F1		Acc	F1	Acc	F1
I. ARC-I	53.84	48.68	50.10	66.58	**XI**. CIG-Siam	74.47	73.03	75.32	78.58
II. ARC-II	54.37	36.77	52.00	53.83	**XII**. CIG-Siam-GCN	**74.58**	**73.69**	**78.91**	**80.72**
III. DUET	55.63	51.94	52.33	60.67	**XIII**. CIG_{cd}-Siam-GCN	73.25	73.10	76.23	76.94
IV. DSSM	58.08	64.68	61.09	70.58	**XIV**. CIG-Sim	72.58	71.91	75.16	77.27
V. C-DSSM	60.17	48.57	52.96	56.75	**XV**. CIG-Sim-GCN	**83.35**	**80.96**	**87.12**	**87.57**
VI. MatchPyramid	66.36	54.01	62.52	64.56	**XVI**. CIG_{cd}-Sim-GCN	81.33	78.88	86.67	87.00
VII. BM25	69.63	66.60	67.77	70.40	**XVII**. CIG-Sim&Siam-GCN	84.64	**82.75**	89.77	90.07
VIII. LDA	63.81	62.44	62.98	69.11	**XVIII**. CIG-Sim&Siam-GCN-Sim^g	84.21	82.46	**90.03**	**90.29**
IX. SimNet	71.05	69.26	70.78	74.50	**XIX**. CIG-Sim&Siam-GCN-BERT^g	**84.68**	82.60	89.56	89.97
X. BERT fine-tuning	81.30	79.20	86.64	87.08	**XX**. CIG-Sim&Siam-GCN-Sim^g&BERT^g	84.61	82.59	89.47	89.71

vertex, "Siam" indicates the use of Siamese encoder, while "Sim" indicates the use of term-based similarity encoder, as shown in Fig. 2. "GCN" means that we convolve the local matching vectors on vertices through GCN layers. Finally, "BERT^g" or "Sim^g" indicates the use of additional global features given by BERT or the five term-based similarity metrics mentioned in Sec. 3, appended to the graphically merged matching vector $\mathbf{m}_{AB}$, for final classification.

Implementation Details. We use Stanford CoreNLP (Manning et al., 2014) for word segmentation (on Chinese text) and named entity recognition. For Concept Interaction Graph construction with community detection, we set the minimum community size (number of keywords contained in a concept vertex) to be 2, and the maximum size to be 6.

Our neural network model consists of word embedding layer, Siamese encoding layer, Graph transformation layers, and classification layer. For embedding, we load the pre-trained word vectors and fix it during training. The embeddings of out of vocabulary words are set to be zero vectors. For the Siamese encoding network, we use 1-D convolution with number of filters 32, followed by an ReLU layer and Max Pooling layer. For graph transformation, we utilize 2 layers of GCN (Kipf and Welling, 2016) for experiments on the CNSS dataset, and 3 layers of GCN for experiments on the CNSE dataset. When the vertex encoder is the five-dimensional features, we set the output size of GCN layers to be 16. When the vertex encoder is the Siamese network encoder, we set the output size of GCN layers to be 128 except the last layer. For the last GCN layer, the output size is always set to be 16. For the classification module, it consists of a linear layer with output size 16, an ReLU layer, a second linear layer, and finally a Sigmoid layer. Note that this classification module is also used for the baseline method SimNet.

As we mentioned in Sec. 1, our code and datasets have been open sourced. We implement our model using PyTorch 1.0 (Paszke et al., 2017). The experiments without BERT are carried out on an MacBook Pro with a 2 GHz Intel Core i7 processor and 8 GB memory. We use L2 weight decay on all the trainable variables, with parameter $\lambda = 3 \times 10^{-7}$. The dropout rate between every two layers is 0.1. We apply gradient clipping with maximum gradient norm 5.0. We use the ADAM optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.8,\ \beta_2 = 0.999,\ \epsilon = 10^8$. We use a learning rate warm-up scheme with an inverse exponential increase from 0.0 to 0.001 in the first 1000 steps, and then maintain a constant learning rate for the remainder of training. For all the experiments, we set the maximum number of training epochs to be 10.

4.1 Results and Analysis

Table 2 summarizes the performance of all the compared methods on both datasets. Our model achieves the best performance on both two datasets and significantly outperforms all other methods. This can be attributed to two reasons. First, as the input of article pairs are re-organized into Concept Interaction Graphs, the two documents are aligned along the corresponding semantic units for easier concept-wise comparison. Second, our model encodes local comparisons around different semantic units into local matching vectors, and aggregate them via graph convolutions, taking semantic topologies into consideration. Therefore, it solves the problem of matching documents via divide-and-conquer, which is suitable for handling long text.

Impact of Graphical Decomposition. Comparing method XI with methods I-VI in Table 2, they all use the same word vectors and use neu-

ral networks for text encoding. The key difference is that our method XI compares a pair of articles over a CIG in per-vertex decomposed fashion. We can see that the performance of method XI is significantly better than methods I-VI. Similarly, comparing our method XIV with methods VII-IX, they all use the same term-based similarities. However, our method achieves significantly better performance by using graphical decomposition. Therefore, we conclude that graphical decomposition can greatly improve long text matching performance.

Note that the deep text matching models I-VI lead to bad performance, because they were invented mainly for sequence matching and can hardly capture meaningful semantic interactions in article pairs. When the text is long, it is hard to get an appropriate context vector representation for matching. For interaction-focused neural network models, most of the interactions between words in two long articles will be meaningless.

Impact of Graph Convolutions. Compare methods XII and XI, and compare methods XV and XIV. We can see that incorporating GCN layers has significantly improved the performance on both datasets. Each GCN layer updates the hidden vector of each vertex by integrating the vectors from its neighboring vertices. Thus, the GCN layers learn to graphically aggregate local matching features into a final result.

Impact of Community Detection. By comparing methods XIII and XII, and comparing methods XVI and XV, we observe that using community detection, such that each concept is a set of correlated keywords instead of a single keyword, leads to slightly worse performance. This is reasonable, as using each keyword directly as a concept vertex provides more anchor points for article comparison . However, community detection can group highly coherent keywords together and reduces the average size of CIGs from 30 to 13 vertices. This helps to reduce the total training and testing time of our models by as much as 55%. Therefore, one may choose whether to apply community detection to trade accuracy off for speedups.

Impact of Multi-viewed Matching. Comparing methods XVII and XV, we can see that the concatenation of different graphical matching vectors (both term-based and Siamese encoded features) can further improve performance. This demonstrates the advantage of combining multi-viewed matching vectors.

Impact of Added Global Features. Comparing methods XVIII, XIX, XX with method XVII, we can see that adding more global features, such as global similarities (Sim^g) and/or global BERT encodings (BERT^g) of the article pair, can hardly improve performance any further. This shows that graphical decomposition and convolutions are the main factors that contribute to the performance improvement. Since they already learn to aggregate local comparisons into a global semantic relationship, additionally engineered global features cannot help.

Model Size and Parameter Sensitivity: Our biggest model without BERT is XVIII, which contains only $\sim$34K parameters. In comparison, BERT contains 110M-340M parameters. However, our model significantly outperforms BERT.

We tested the sensitivity of different parameters in our model. We found that 2 to 3 layers of GCN layers gives the best performance. Further introducing more GCN layers does not improve the performance, while the performance is much worse with zero or only one GCN layer. Furthermore, in GCN hidden representations of a size between 16 and 128 yield good performance. Further increasing this size does not show obvious improvement.

For the optional community detection step in CIG construction, we need to choose the minimum size and the maximum size of communities. We found that the final performance remains similar if we vary the minimum size from 2$\sim$3 and the maximum size from 6$\sim$10. This indicates that our model is robust and insensitive to these parameters.

Time complexity. For keywords of news articles, in real-world industry applications, they are usually extracted in advance by highly efficient off-the-shelf tools and pre-defined vocabulary. For CIG construction, let N_s be the number of sentences in two documents, N_w be the number of unique words in documents, and N_k represents the number of unique keywords in a document. Building keyword graph requires $\mathcal{O}(N_s N_k + N_w^2)$ complexity (Sayyadi and Raschid, 2013), and betweenness-based community detection requires $\mathcal{O}(N_k^3)$. The complexity of sentence assignment and weight calculation is $\mathcal{O}(N_s N_k + N_k^2)$. For graph classification, our model size is not big and can process document pairs efficiently.

5 Related Work

Graphical Document Representation. A majority of existing works can be generalized into four categories: word graph, text graph, concept graph, and hybrid graph. Word graphs use words in a document as vertices, and construct edges based on syntactic analysis (Leskovec et al., 2004), co-occurrences (Zhang et al., 2018; Rousseau and Vazirgiannis, 2013; Nikolentzos et al., 2017) or preceding relation (Schenker et al., 2003). Text graphs use sentences, paragraphs or documents as vertices, and establish edges by word co-occurrence, location (Mihalcea and Tarau, 2004), text similarities (Putra and Tokunaga, 2017), or hyperlinks between documents (Page et al., 1999). Concept graphs link terms in a document to real world concepts based on knowledge bases such as DBpedia (Auer et al., 2007), and construct edges based on syntactic/semantic rules. Hybrid graphs (Rink et al., 2010; Baker and Ellsworth, 2017) consist of different types of vertices and edges.

Text Matching. Traditional methods represent a text document as vectors of bag of words (BOW), term frequency inverse document frequency (TF-IDF), LDA (Blei et al., 2003) and so forth, and calculate the distance between vectors. However, they cannot capture the semantic distance and usually cannot achieve good performance.

In recent years, different neural network architectures have been proposed for text pair matching tasks. For representation-focused models, they usually transform text pairs into context representation vectors through a Siamese neural network, followed by a fully connected network or score function which gives the matching result based on the context vectors (Qiu and Huang, 2015; Wan et al., 2016; Liu et al., 2018; Mueller and Thyagarajan, 2016; Severyn and Moschitti, 2015). For interaction-focused models, they extract the features of all pair-wise interactions between words in text pairs, and aggregate the interaction features by deep networks to give a matching result (Hu et al., 2014; Pang et al., 2016). However, the intrinsic structural properties of long text documents are not fully utilized by these neural models. Therefore, they cannot achieve good performance for long text pair matching.

There are also research works which utilize knowledge (Wu et al., 2018), hierarchical property (Jiang et al., 2019) or graph structure (Nikolentzos et al., 2017; Paul et al., 2016) for long text matching. In contrast, our method represents documents by a novel graph representation and combines the representation with GCN.

Finally, pre-training models such as BERT (Devlin et al., 2018) can also be utilized for text matching. However, the model is of high complexity and is hard to satisfy the speed requirement in real-world applications.

Graph Convolutional Networks. We also contributed to the use of GCNs to identify the relationship between *a pair of* graphs, whereas previously, different GCN architectures have mainly been used for completing missing attributes/links (Kipf and Welling, 2016; Defferrard et al., 2016) or for node clustering or classification (Hamilton et al., 2017), but all within the context of a *single* graph, e.g., a knowledge graph, citation network or social network. In this work, the proposed Concept Interaction Graph takes a simple approach to represent a document by a weighted undirected graph, which essentially helps to decompose a document into subsets of sentences, each subset focusing on a different sub-topic or concept.

6 Conclusion

We propose the *Concept Interaction Graph* to organize documents into a graph of concepts, and introduce a divide-and-conquer approach to matching a pair of articles based on graphical decomposition and convolutional aggregation. We created two new datasets for long document matching with the help of professional editors, consisting of about 60K pairs of news articles, on which we have performed extensive evaluations. In the experiments, our proposed approaches significantly outperformed an extensive range of state-of-the-art schemes, including both term-based and deep-model-based text matching algorithms. Results suggest that the proposed graphical decomposition and the structural transformation by GCN layers are critical to the performance improvement in matching article pairs.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. *The semantic web*, pages 722–735.

Collin Baker and Michael Ellsworth. 2017. Graph methods for multilingual framenets. In *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*, pages 45–50.

Cosmin Adrian Bejan and Sanda Harabagiu. 2010. Unsupervised event coreference resolution with rich linguistic features. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 1412–1422. Association for Computational Linguistics.

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

Daniel Bruggermann, Yannik Hermey, Carsten Orth, Darius Schneider, Stefan Selzer, and Gerasimos Spanakis. 2016. Storyline detection and tracking using dynamic latent dirichlet allocation. In *Proceedings of the 2nd Workshop on Computing News Storylines (CNS 2016)*, pages 9–19.

Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering. In *Advances in Neural Information Processing Systems*, pages 3844–3852.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Yixing Fan, Liang Pang, JianPeng Hou, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2017. Matchzoo: A toolkit for deep text matching. *arXiv preprint arXiv:1707.07270*.

William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*.

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.

Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management*, pages 2333–2338. ACM.

Jyun-Yu Jiang, Mingyang Zhang, Cheng Li, Mike Bendersky, Nadav Golbandi, and Marc Najork. 2019. Semantic text matching for long-form documents.

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

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Heeyoung Lee, Angel Chang, Yves Peirsman, Nathanael Chambers, Mihai Surdeanu, and Dan Jurafsky. 2013. Deterministic coreference resolution based on entity-centric, precision-ranked rules. *Computational Linguistics*, 39(4):885–916.

Heeyoung Lee, Marta Recasens, Angel Chang, Mihai Surdeanu, and Dan Jurafsky. 2012. Joint entity and event coreference resolution across documents. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 489–500. Association for Computational Linguistics.

Jure Leskovec, Marko Grobelnik, and Natasa Milic-Frayling. 2004. Learning sub-structures of document semantic graphs for document summarization.

Bang Liu, Di Niu, Kunfeng Lai, Linglong Kong, and Yu Xu. 2017. Growing story forest online from massive breaking news. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 777–785. ACM.

Bang Liu, Ting Zhang, Fred X Han, Di Niu, Kunfeng Lai, and Yu Xu. 2018. Matching natural language sentences with hierarchical sentence factorization. In *Proceedings of the 2018 World Wide Web Conference on World Wide Web*, pages 1237–1246. International World Wide Web Conferences Steering Committee.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of 52nd annual meeting of the association for computational linguistics: system demonstrations*, pages 55–60.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *Proceedings of the 2004 conference on empirical methods in natural language processing*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World*

Wide Web, pages 1291–1299. International World Wide Web Conferences Steering Committee.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese recurrent architectures for learning sentence similarity. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Paul Neculoiu, Maarten Versteegh, Mihai Rotaru, and Textkernel BV Amsterdam. 2016. Learning text similarity with siamese recurrent networks. *ACL 2016*, page 148.

Giannis Nikolentzos, Polykarpos Meladianos, François Rousseau, Yannis Stavrakas, and Michalis Vazirgiannis. 2017. Shortest-path graph kernels for document similarity. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1890–1900.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

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

Adam Paszke, Sam Gross, Soumith Chintala, and Gregory Chanan. 2017. Pytorch: Tensors and dynamic neural networks in python with strong gpu acceleration. *PyTorch: Tensors and dynamic neural networks in Python with strong GPU acceleration*.

Christian Paul, Achim Rettinger, Aditya Mogadala, Craig A Knoblock, and Pedro Szekely. 2016. Efficient graph-based document similarity. In *European Semantic Web Conference*, pages 334–349. Springer.

Marten Postma, Filip Ilievski, and Piek Vossen. 2018. Semeval-2018 task 5: Counting events and participants in the long tail. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 70–80.

Jan Wira Gotama Putra and Takenobu Tokunaga. 2017. Evaluating text coherence based on semantic similarity graph. In *Proceedings of TextGraphs-11: the Workshop on Graph-based Methods for Natural Language Processing*, pages 76–85.

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

Bryan Rink, Cosmin Adrian Bejan, and Sanda M Harabagiu. 2010. Learning textual graph patterns to detect causal event relations. In *FLAIRS Conference*.

Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389.

François Rousseau and Michalis Vazirgiannis. 2013. Graph-of-word and tw-idf: new approach to ad hoc ir. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 59–68. ACM.

Hassan Sayyadi and Louiqa Raschid. 2013. A graph analytical approach for topic detection. *ACM Transactions on Internet Technology (TOIT)*, 13(2):4.

Adam Schenker, Mark Last, Horst Bunke, and Abraham Kandel. 2003. Clustering of web documents using a graph model. *SERIES IN MACHINE PERCEPTION AND ARTIFICIAL INTELLIGENCE*, 55:3–18.

Aliaksei Severyn and Alessandro Moschitti. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 373–382. ACM.

Dafna Shahaf, Jaewon Yang, Caroline Suen, Jeff Jacobs, Heidi Wang, and Jure Leskovec. 2013. Information cartography: creating zoomable, large-scale maps of information. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1097–1105. ACM.

Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, pages 373–374. ACM.

Piek Vossen, Tommaso Caselli, and Yiota Kontzopoulou. 2015. Storylines for structuring massive streams of news. In *Proceedings of the First Workshop on Computing News Storylines*, pages 40–49.

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*, volume 16, pages 2835–2841.

Yu Wu, Wei Wu, Can Xu, and Zhoujun Li. 2018. Knowledge enhanced hybrid neural network for text matching. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ting Zhang, Bang Liu, Di Niu, Kunfeng Lai, and Yu Xu. 2018. Multiresolution graph attention networks for relevance matching. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 933–942. ACM.

Deyu Zhou, Haiyang Xu, and Yulan He. 2015. An unsupervised bayesian modelling approach for storyline detection on news articles. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1943–1948.