applied sciences

MDPI

Article

Similarity Calculation via Passage-Level Event Connection Graph

Ming Liu [1,2,3,4], Lei Chen [4,*] and Zihao Zheng [2]

[1] State Key Laboratory of Communication Content Cognition, People's Daily Online, Beijing 100733, China
[2] Harbin Institute of Technology, School of Computer Science and Technology, Harbin 150001, China
[3] Peng Cheng Laboratory, Shenzhen 518055, China
[4] International Business and Management Research Center, Beijing Normal University, Zhuhai 519087, China
* Correspondence: chenlei@bnuz.edu.cn

Abstract: Recently, many information processing applications appear on the web on the demand of user requirement. Since text is one of the most popular data formats across the web, how to measure text similarity becomes the key challenge to many web applications. Web text is often used to record events, especially for news. One text often mentions multiple events, while only the core event decides its main topic. This core event should take the important position when measuring text similarity. For this reason, this paper constructs a passage-level event connection graph to model the relations among events mentioned in one text. This graph is composed of many subgraphs formed by triggers and arguments extracted sentence by sentence. The subgraphs are connected via the overlapping arguments. In term of centrality measurement, the core event can be revealed from the graph and utilized to measure text similarity. Moreover, two improvements based on vector tunning are provided to better model the relations among events. One is to find the triggers which are semantically similar. By linking them in the event connection graph, the graph can cover the relations among events more comprehensively. The other is to apply graph embedding to integrate the global information carried by the entire event connection graph into the core event to let text similarity be partially guided by the full-text content. As shown by experimental results, after measuring text similarity from a passage-level event representation perspective, our calculation acquires superior results than unsupervised methods and even comparable results with some supervised neuron-based methods. In addition, our calculation is unsupervised and can be applied in many domains free from the preparation of training data.

Keywords: text similarity calculation; passage-level event connection graph; vector tuning; graph embedding

Citation: Liu, M.; Chen, L.; Zheng, Z. Similarity Calculation via Passage-Level Event Connection Graph. *Appl. Sci.* **2022**, *12*, 9887. https://doi.org/10.3390/app12199887

Academic Editors: Dionisis Margaris and Stefanos Ougiaroglou

Received: 18 May 2022
Accepted: 26 September 2022
Published: 1 October 2022

1. Introduction

The fast advance of web technology causes an explosive increase of web data. Text is one of the most prevailing data formats across the web, which enables lots of text-based analysis tools to be provided to help users ease the way to process texts. Text similarity calculation is one of the fundamental text processing tasks, which is also the bottleneck of many web applications, such as news recommendation, Q&A system, etc. Traditional text similarity calculations can be roughly divided into two classes. One is supervised based, which maps two texts into a high-dimensional space and finally makes two similar texts close in the form of vector representation. The other one is unsupervised based, which often treats text as a sequence of word pieces and scores the similarity between two texts in terms of word concurrence plus the order of the sequence or ignoring it. Except word concurrence, some other statistical features are also utilized such as TF/IDF or Mutual Information. Between the two kinds of calculations, supervised ones always own high performance, since they can accurately draw the boundary between similar texts and

dissimilar texts with the help of training data. The emergence of neural-based algorithm enhances the performances of supervised methods to a higher level, which gain a huge advantage beyond unsupervised ones. Accordingly, their high-quality results are over dependent on training data. When the domain changes, the performances of supervised ones degrade sharply. Unsupervised ones do not suffer from this limitation, since they do not refer to any transcendental knowledge and are free from training data. Thus, they do not fear domain transferring. The types of texts across the web are countless. We cannot collect all types of texts as training data to let supervised methods go through at advance. Therefore, it is reasonable to design an unsupervised similarity calculation, which can be applied in any domain.

Text is often used to record events. Reading text, we know what is happening and what is the end. An event just indicates something happens in some place at some time. Traditional event extraction tasks, like ACE [1] and FrameNet [2], treat events occurring in sentence-level (which means event is fine-grained). The events stated by different sentences are independent. Events extracted at sentence level cannot be directly utilized to deal with passage-level task. Text similarity calculation is a typical passage-level task. The similarity between two texts depends on their overall content similarity. We should take a high-level view over all the events mentioned in one text. Such as one text has one main topic, from passage-level, though there are many events stated by the sentences in one text, one text only has a core event. The other events serve the core event, as explaining the core event or completing the details of the core event. That indicates the core event mostly decides the similarity between two texts, and the other events play an auxiliary role. For example, the similarity between two following articles (Two articles are, respectively, https://www.reuters.com/article/us-asia-storm/super-typhoon-slams-into-china-after-pummeling-philippines-idUSKCN1LW00F, and https://www.wunderground.com/cat6/Typhoon-Mangkhut-Causes-Heavy-Damage-Hong-Kong-China-and-Macau, accessed on 17 May 2022. These two articles can be accessed till the pages are deleted) is high, since both take the event of the damage of "Mangosteen" typhoon as the core event, though one details the degree of the damage and the other does not.

As text similarity calculation is a passage-level task and concerns whether two texts stress the same core event or not, we should take the entire text into consideration to locate the core event. Anyway, most of events cannot be fully stated by only one sentence. They may cross several sentences, even the nonadjacent sentences. Like financial events, e.g., investment and debt, the arguments of those events spread all over the entire text. For this reason, traditional sentence-level event extraction methods are not appropriate to extract the core event from a passage level. This paper just constructs a graph, namely event connection graph, to cover the multiple relations among the events mentioned in one text. This graph is composed of a set of polygons. Each polygon is formed by one trigger as its center and some arguments surrounding this center. The trigger and the arguments are extracted by a sentence-level event extraction method. To value the nodes in the graph, PageRank is adopted. The nodes of the largest values are treated as the core event, and text similarity is calculated according to the correlation between two core events, respectively, extracted from two texts. Moreover, two improvements based on vector tuning are proposed to better model the event connection graph. One is to detect the semantically similar triggers and link them to fully cover the relations among events. The other is to embed the global content carried by the entire event connection graph into the core event to let text similarity be partially guided by the full-text content.

To sum up, the contributions of this paper can be summarized as follows:

1. This paper proposes a novel event connection graph to model the events and their mutual relations mentioned in one text. This graph is composed of some polygons, and each polygon represents a sentence-level event. Via PageRank, the core event can be extracted to represent the main content of the text, and further utilized to calculate text similarity.

2. Two improvements are provided to enhance the completeness and effectiveness of the constructed event connection graph. One is to tune the vector representation of

the trigger to find and link more related events, which enables the generation of a more comprehensive event connection graph. The other is to embed the information carried by the entire event connection graph into the core event to make similarity result more rational.

3. As shown by experimental results, our similarity calculation obtains superior results than unsupervised methods by a large margin, and even comparable results with supervised neuron-based methods. Typically, our calculation is unsupervised. It can be applied in any domain without the dilemma of domain transferring.

Though our similarity calculation can combine the merits from supervised and unsupervised similarity calculations. Our calculation has time issues needed to be further solved. In particular, our calculation needs to form a passage-level event representation. This kind of operation needs extra time. Thus, though our calculation has higher accuracy, it is not fit to online applications, especially some time-insensitive applications.

Our paper has six sections. Section 1 is introduction, which briefly introduces the motivation of our work and summarizes its contributions. Section 2 shows some related research. Section 3 gives a brief overview of our work at first, and then details the process used to construct the event connection graph and the approach used to value the nodes in the graph. Section 4 tells two improvements on our event connection graph based on vector tuning. Section 5 designs some experiments to illustrate the high quality of our similarity calculation. Section 6 presents the conclusions and gives some future works.

2. Related Work

The rapid advance of internet technology brings the explosive increase of web data. Facing the massive amount of data, internet users require automatic data analysis and processing tools. Text is one of the most prevailing data formats on the web. Thus, many web applications are designed aiming at processing textual data. Almost all the text related applications treat text similarity calculation as their fundamental module. Such as text clustering [3], machine dialogue [4], product recommendation [5], Q&A [6], those applications take text similarity calculation as the key component. In general, the methods for text similarity calculation can be partitioned into two categories. One is supervised based which is guided by annotated training samples. The other one is unsupervised based free from annotations.

Supervised type often treats texts as points mapping to the high-dimensional space. A classification function is trained to separate points into similar and dissimilar two groups. Some other methods turn classification to a rank problem, which learn score functions to discriminate similar points from dissimilar ones. The advantage of supervised type is brought from the guidance of training data. Due to training process, supervised type often acquires high performance. Text is encoded as a vector for calculating convenience. Before the appearance of deep neuron network, one-hot vector is widely used. Only one entry has non-zero value. This kind of encoder generates high-dimensional and sparse vectors, which degrades the quality of many text-oriented applications [7]. The proposal of word embedding changes this dilemma. Word embedding compresses one-hot vector into a densely distributional vector with low dimension. Skip-gram [8], CBOW [9], GloVe [10], ELMo [11] are typical exemplars. The neuron-based models, such as CNN [12], GRU [13], LSTM [14], or the pre-trained language models such as Transformer [15], GPT [16], BERT [17], XL-NET [18], Roberta [19] can produce more reasonable text representation on the basis of word embedding. The overlapping degree decides the similarity between texts, whereas, only depending on word concurrence or word alignment cannot fully express the semantic similarity between texts. To better model the interaction between texts, attention mechanism is taken, which considers the relevance of non-aligned parts across the input sequences. The widely applied attentions are multiple layer attention in [20] and co-attention in [21]. Basically, supervised text similarity calculations own high performance, especially after the application of neuron-based models. However, they are easily distorted by training data. They have to make a hypothesis about the distribution of input data in terms of the transcendental knowledge implicitly provided by training data. There is no way to collect

enough training data to let supervised calculations go through in advance, especially for the neural-based methods, since their explosive parameters require massive data for fully training. For this reason, supervised calculations are appropriate to deal with domain data and can hardly be transferred. In our paper, we hope to design a text similarity calculation, which can fit to the texts in any type and from any domain. Therefore, we try to design an unsupervised text similarity calculation.

Unsupervised similarity calculations free from training data. They model input data all by their natural distribution. Some untrained score functions are taken to measure text similarity based on distribution similarity. Euclidean distance [22], KL divergence [23], and Entropy [24] are some widely used score functions. Joint functions are also proposed to integrate previous scores [25,26]. Due to missing training data, the features used by score functions are some statistical values provided by raw texts after word segmentation and stemming, such as TF/IDF [27], TextRank [28], and LDA [29]. Some recent works try to turn unsupervised similarity calculation into a supervised task. An iterative process is adopted to take the output cases as training data in turn [30]. This kind of calculations suffers from cold-starting issue, which needs to set initial similarity values beforehand, and the final results drop a lot on the inappropriate initialization.

Table 1 just summarizes the difference between supervised text similarity calculation and unsupervised text similarity calculation.

Table 1. Comparison of supervised and unsupervised similarity calculation.

Feature/Method	Supervised	Unsupervised
Training data	Results derived from training data	Free from training data
Performance	Higher performance	Lower performance
W/O semantics	Semantic embedding via representation	Lack of understanding semantics
Domain transfer	Hard	Easy

As indicated by the pervious table, it can be observed that these two kinds of similarity calculations both have corresponding merits and drawbacks. Supervised methods have higher performances due to its importing of training data. However, using training data is hard to alter domain. This situation causes that the performances of supervised methods drop sharply when domain changes. On the contrary, unsupervised methods have lower performances, while their performances do not drop along with domain alteration. In this situation, we try to propose an unsupervised similarity calculation to combine both merits of supervised and unsupervised methods.

Features taken by previous calculations are words or word spans which contribute mostly to score functions (applied in supervised ones) or own some prominent distribution compared with other features (applied in unsupervised ones). Though among supervised calculations, some algorithms may learn a semantic embedding on word level or text level in terms of training data to help model the semantics in input text [31–33]. They all ignore a fact that most of web texts are used to record events. One text should tell one core event. The other mentioned events either help explain the core event or provide some details of the core event (such as time, place, or related events). In fact, the core event mostly decides the similarity between two texts. In other words, if one event is stressed by two texts meanwhile, these two texts are similar at a high possibility. Thus, the task of calculating text similarity can be fulfilled by comparing the discrepancy of the core events, respectively, extracted from two texts. The core event represents the main content of one text. It should be extracted from a passage-level viewpoint.

Event extraction and representation have been studied during a long time. As the most famous event extraction tasks, MUC (Message Understanding Conference) [34] and ACE (Automatic Content Extraction) [35] have been held for about 30 years. The definition of event in MUC and ACE is sentence-level with trigger as key element and arguments as supplementary details. Traditional event extraction tasks assume that an event can be fully expressed by a single sentence. It can be extracted without taking other sentences

into consideration. Since an event can be formatted as trigger and arguments, traditional sentence-level event extraction methods can be separated into two successive steps. The former step is called event detection (or trigger extraction), which aims to detect events and classify event type. The latter step is called argument extraction, which aims to acquire the arguments related to the trigger, such as time, location, subject, and object, etc. The algorithms designed for sentence-level event extraction are not appropriate to extract passage-level events, since they aim to learn a better representation for single sentences [36,37] and not to model the relations among events across sentences.

As told before, traditional event extraction methods treat sentences independently and extract events from a single sentence. Though it has been proposed something called cross-sentence event extraction methods. While their object is still to extract events from a single sentence, their highlight is to take the adjacent sentences in a sliding window into consideration during extracting process [38,39]. Obviously, the cross-sentence event extraction methods are not suitable to extract core events, since they also miss the operation of modeling the relations among events from a passage angle. Therefore, this paper designs an event connection graph to cover the relations among all the events mentioned in one text. Via graph centrality measurements, the core event can be extracted and used to calculate text similarity.

3. Model Details

3.1. Task Overview

The objective of text similarity calculation is easy to be defined. As given two texts s_1 and s_2, we hope to obtain the similarity value between s_1 and s_2. Different from traditional calculations, this paper aims to calculate text similarity in terms of measuring whether s_1 and s_2 mention the same core event or not. A graph, noted as $G(V, E)$, is constructed to model the relations among events, where V is node set and E is arc set. It is called event connection graph. The nodes in V are just triggers and arguments. Those triggers and arguments are extracted sentence by sentence to represent a serial of sentence-level events [40]. While the arcs in G represent the relations among events. Since the core event represents the main content of one text, it should be surrounded by the other events. As turning events to nodes and relations among events to arcs to form a graph, the nodes, which represent the core event, should locate at the graph's center. Via some graph centrality measurement, such as PageRank, we can easily locate the core event. It is worth noting that comparing the nodes (extracted by sentence-level event extraction methods), the arcs play an important role to decide the quality of similarity results. As shown in the experiments, we adopt several popular sentence-level event extraction methods, but it can hardly see the difference in accuracy. Therefore, we provide two improvements based on vector tuning to complete the constructed event connection graph to involve more arcs.

3.2. Graph Construction

This section details the approach used to construct the event connection graph. Here we borrow the method shown in [40] to extract fine-grained sentence-level events. Each sentence-level event is formed as a polygon with trigger as its center and arguments as its surrounding nodes. The arguments are listed in the order how they appear in the sentence. The trigger and the arguments are connected by arcs. Figure 1 is an example of one polygon formed from the sentence "The President of USA communicates with Chinese Leader on the phone about North Korea issue". It is straightforward that trigger is the key element, since event type and argument template are both decided by trigger. Therefore, we put trigger at the center of the polygon to represent its pivotal position and put the arguments surrounding the trigger. As shown in Figure 1, the trigger "communicate" (after stemming) is put at the center of the polygon and the arguments related to this trigger are put surrounding the center in the order how they appear in the sentence.

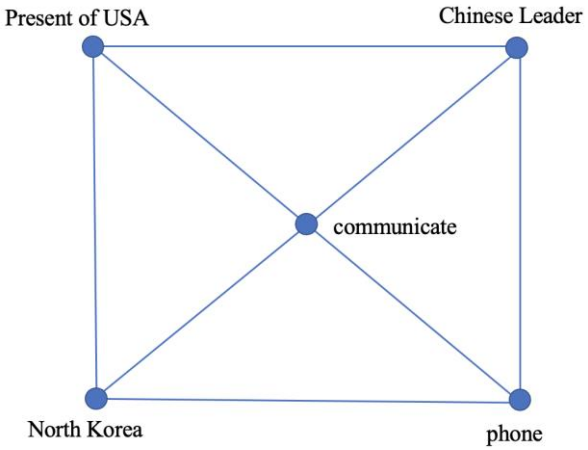

Figure 1. The polygon formed from the example sentence.

To model the relations among events, we just simply connect the polygons via the overlapping arguments to form an event connection graph. Figure 2 shows an example event connection graph formed from the following four sentences. In Section 4, we further propose a way to find semantically similar triggers to reveal deeper relations among events.

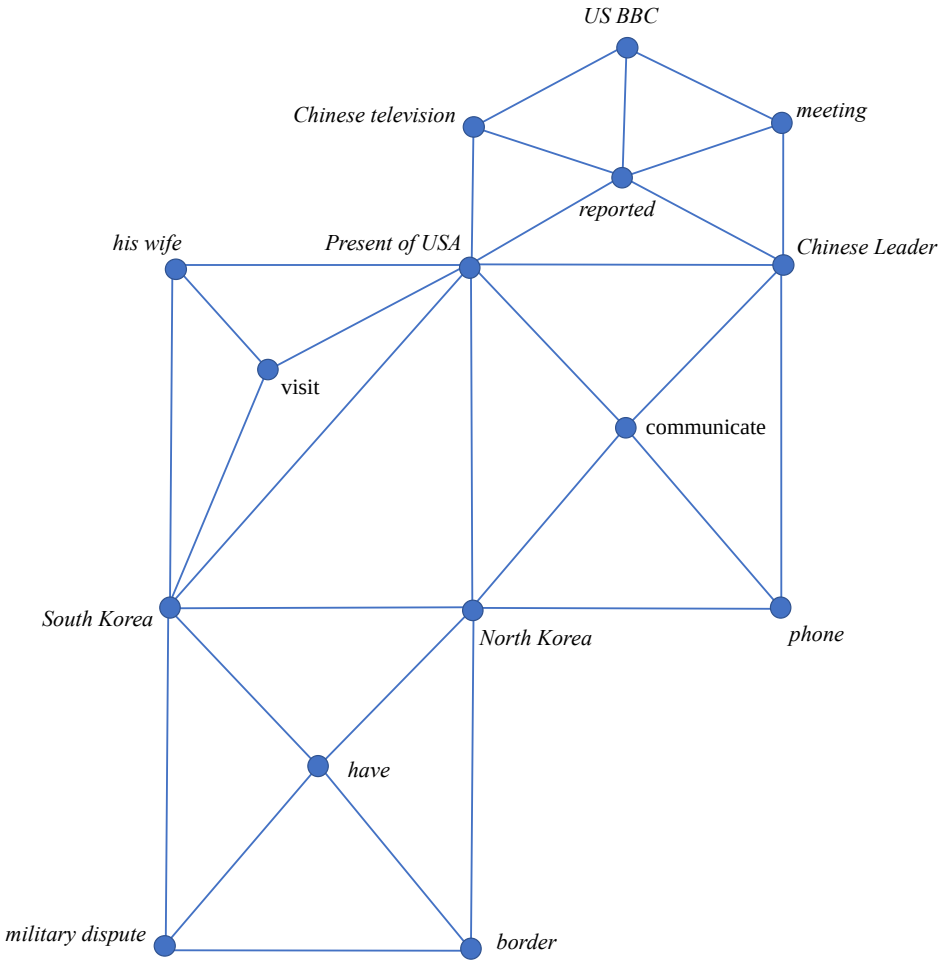

Figure 2. The event connection graph formed from given sentences.

"South Korea and North Korea have a military dispute on the border."
"The President of USA visits South Korea with his wife."

"The President of USA communicates with Chinese Leader on the phone about North Korea issue."

"Chinese television and US BBC reported the meeting between the US president and Chinese leader respectively."

3.3. Node Evaluation

Typically, "If one author emphases a topic (or a clue), everything in his article is related to this topic (or clue)" [41,42]. It is straightforward to make an assumption that the core event in one text should be supported by the other events. If we project all the events mentioned in one text to a plane, the core event will be the point surrounded by the other event. In our paper, we just project events to a plane, while treating the event as a polygon which includes several nodes (i.e., trigger and arguments). Trigger is the key element in the event and decides event type. Thus, we put trigger at the center of the polygon. These two situations ensure that the center of the graph should be the core event. The remaining job becomes how to locate the center of the graph. PageRank, a popular centrality measurement, is chosen to fulfil this task. PageRank is proposed by Google and used to rank web pages in searching engine. The principle behind PageRank is random walk [43]. When one surfer randomly surfs on a graph, the node visited most frequently should be the central node (owning the largest PageRank value).

We just follow the traditional PageRank measurement. The only difference is to use the transition matrix (noted as A) formed from our event connection graph. The size of A is $v * v$. v denotes the number of nodes in the graph. Each entry in A is the transition probability from one node in the row to another node in the column. For example, given a surfer who travels on the event connection graph, if (i,j) is an arc, A_{ij} denotes the transition probability that this surfer visits j by jumping from i, and can be set as the reciprocal of the out degree of node i. On the opposite, if (i,j) is not an arc, this probability is 0.

Via PageRank, each node in the event connection graph has a value. This value indicates the importance of the node in the graph, which can be utilized to locate the core event. There are two kinds of nodes in the graph, i.e., trigger and argument. If the node of the largest PageRank value is a trigger, we then extract the trigger and the arguments belonging to this trigger as the core event. This way just treats the nodes in the polygon which takes the trigger of the largest value as its center as the core event. Otherwise, if the node of the largest value is an argument, we then output the nodes in all the polygons which take this argument as their intersection node. Figure 3 gives the PageRank values of the nodes in Figure 2. In this figure, the node "President of USA" owns the largest value (marked in red color). Since "President of USA" is an argument, we output the nodes in the polygons which take "President of USA" as their intersection node. The chosen nodes are marked in a yellow color in Figure 3. These nodes just indicate the core event expressed by the previous paragraph with four sentences. However, the chosen event is not accurate, since the main meaning of this paragraph is about the meeting of two leaders in USA and China.

Let S_i and S_j, respectively, denote the two sets which include the chosen nodes in the event connection graphs formed from the given texts, $text_i$ and $text_j$. To calculate the similarity between $text_i$ and $text_j$, we can form a matrix, noted as TS_{ij}. Each element in this matrix denotes the similarity between two chosen nodes in S_i and S_j, respectively. Since the node in the graph is either trigger word or argument word, it can be represented as vector via GloVe [10]. Their similarity can be measured by vector similarity via Cosine. Some triggers or arguments may be phrase (composing of several words). We then average the vectors of the words in that phrase as its vector representation. We take the mean of all the elements in TS_{ij} as the similarity between $text_i$ and $text_j$. The formula is shown as follows:

$$sim(text_i, text_j) = \sum_{k=1}^{n} TS_{ij}(k)/n \quad (1)$$

where n denotes the number of all the elements in TS_{ij}, and $TS_{ij}(k)$ denotes one element in TS_{ij}.

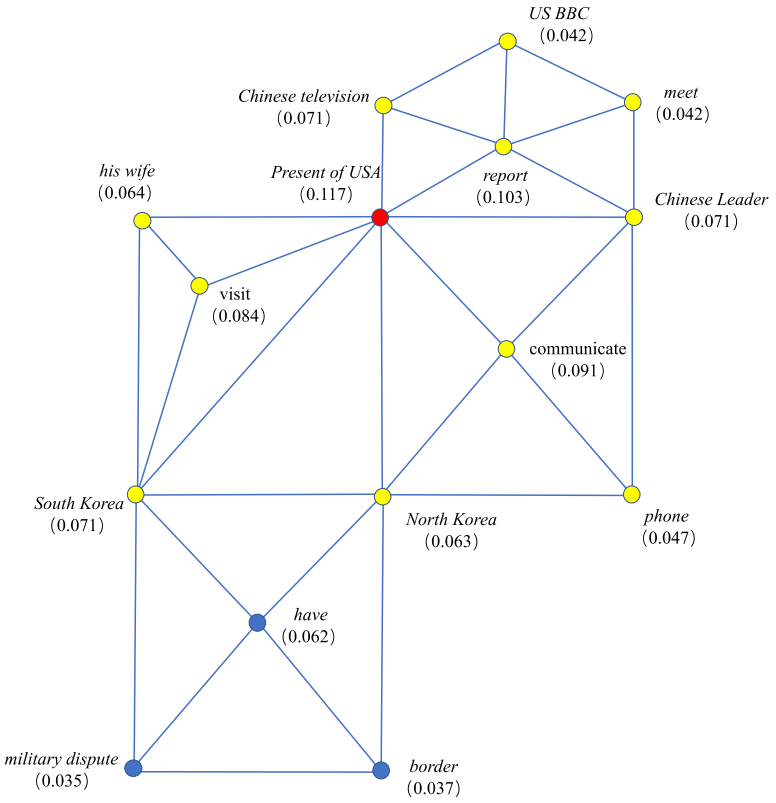

Figure 3. PageRank values of the nodes in the event connection graph.

The previously constructed event connection graph has two flaws. One is that it only uses the overlapping arguments shared by polygons to model the relations among events. This kind of relation is too vague and not sufficient, since the relations among events are mainly caused by triggers. We should provide a way to detect deeper relations among events. The other is that only the node of the largest value and its adjacent nodes are chosen as text representation. These nodes can cover the information expressed by the core event and some other events highly related to the core event. The rest of the events mentioned in the text can also add some supplementary details. These details also need to be considered in similarity calculation. For this reason, two improvements are made. One is to fine-tune the trigger vector to detect and connect more related events. The other is to tune the vectors of the nodes in the core event to let them integrate the information carried by the entire graph.

4. Two Improvements Made on Our Event Connection Graph

4.1. Tuning Trigger Words

The relations among events are mainly caused by trigger words. We should provide a way to find semantically similar triggers and link them in the event connection graph to involve more relations among events. It has been counted that about ninety percent of trigger words are nouns and verbs (or noun and verb phrases) [44,45]. The pre-trained word embedding injects semantics in vector representation and obtains this representation counting on whether two words own the similar contexts or not. However, in event related tasks, we cannot merely depend on pre-trained word embedding to reveal semantic similarity between triggers. Trigger and its arguments have some commonly used collocations, e.g., “kick football” and “play basketball”. That causes two trigger words which have high semantic similarity may have different contexts.

To find semantically similar trigger words, we can borrow the help from some synonym dictionaries, like VerbNet and WordNet, two manually formed synonym dictionaries. These two dictionaries put semantically similar words in one synset. The synsets are organized in hierarchy. Unfortunately, dictionary cannot cover all the possible semantically similar trigger pairs. Anyway, it is time-consuming and laborious to manually construct such kind of dictionary. Therefore, we should provide a way to find semantically similar triggers independent of manual dictionary. In this paper, we try to fine-tune the vector representations of the trigger words to let semantically similar triggers own close vector representations. Two triggers whose cosine similarity is beyond the threshold (0.8) are connected through an arc to involve more rational relations among events. Figure 4 shows the event connection graph after connecting similar triggers, i.e., visit and communicate, and report and communicate. Regarding the threshold (0.8), it is set based on sufficient experimental results shown in the experimental section. GloVe [10] trained on wiki data is used as the basic trigger embedding.

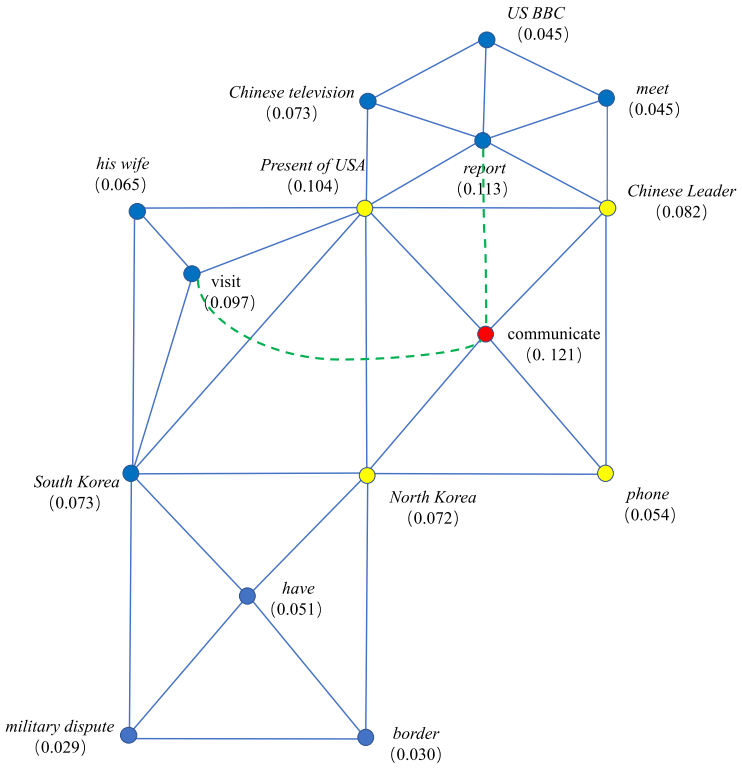

Figure 4. The event connection graph by linking semantically similar triggers.

As shown in Figure 4, with the inserted arc, dotted line with green color, the node of the largest value changes to "communicate". That indicates, after involving novel relation, the core event can be revealed more correctly.

Let B_c denote the set including the synonymous trigger pairs sampled from VerbNet and WordNet. We tune the vector representations of the triggers according to the following formulas:

$$min\ O(B_c) = min(O_c(B_c) + R(B_c)) \tag{2}$$

$$O_c(B_c) = \sum_{(x_l,x_r)\in B_c} [\tau(att + x_l t_l - x_l x_r) + \tau(att + x_r t_r - x_l x_r)] \tag{3}$$

$$R(B_c) = \sum_{x_i \in B_c} \lambda \parallel x_i(int) - x_i \parallel_2 \tag{4}$$

where (x_l, x_r) denotes a synonymous word pair in B_c. t_l is one word, randomly sampled from the synset which x_l and x_r are not in. So is to t_r. att denotes the predefined deviation between the semantically similar word pair and the dissimilar one. It is set to 0.6. τ denotes max margin loss, noted as $max\ \tau(0, x)$. $x_i(int)$ denotes the pre-trained GloVe vector. λ is a

predefined regression parameter and is set to 0.0006. The predefined parameters are set according to [45].

As shown in Equation (2), the tuning formula has two parts. The former part (noted as $O_c(B_c)$) refers to Equation (3), which makes similar triggers own similar vector representations. The latter part (noted as $R(B_c)$) refers to Equation (4), which keeps the tuned vectors not far away from their pre-trained results. Since the pre-trained vectors are acquired from a large-scale corpus, we certainly do not want the tuned vectors to deviate from the pre-trained ones. If one trigger in the event connection graph is tuned, we then replace its original vector representation by the tuned one. In our tuning method, we only tune the vectors of the words included by VerbNet and WordNet, and do not extend the range outside the dictionaries. The reason is that the pre-trained vector representations are acquired from a large-scale corpus. Thus, they are credible until we have enough evidence to support that the pre-trained vector representations cannot calculate word similarity accurately. If one trigger is a phrase, we simply take the mean of the vectors through all the words in that phrase as its representation.

In English, the synonymous word pairs in VerbNet and WordNet can be used as training data. For the other languages, it is hard to find such kind of dictionary. We then use these two dictionaries as bridge to construct training data. We take VerbNet and WordNet as pivot dictionaries and utilize Google translation to translate the words in them into any language. However, one word in English can be translated to many words in the other language. Taking the synonymous word pair "undo" and "untie" in VerbNet for example, "undo" can be translated to five words in Chinese like "打开 (open)", "解开 (untie)", "拆开 (open)", "消除 (remove)", "取消 (cancel)", while "untie" can be translated to "解开 (untie)" and "松开 (loosen)". We finally obtain the possible translated word pairs in the number of 9 (except the duplicated one "解开 (untie)" and "解开 (untie)"). Among them, only the word pair "解开 (untie)" and "松开 (loosen)" is a rational synonym. To avoid incorrect translation, we introduce back translation, extensively used in unsupervised translation task to avoid semantic drift [46]. Following the idea of back translation, we only remain the translated word pairs which can be back translated to the exact same words in English. Also taking the word pair (undo, untie) for example, when we translate them in Chinese, we only remain the word pair (解开, 松开), since these two words can be translated back to undo and untie in English, respectively.

4.2. Node Representation via Graph Embedding

In Section 3, we only choose the nodes which can represent the core event as text representation to calculate text similarity. On one side, except the core event, the information carried by the other events (we call them supplementary events) also provide some useful information. We should not simply abandon them. On the other side, the information carried by the supplementary events is trivial compared with the core event. Thus, there is no need to choose nodes from the event connection graph to represent them. For this reason, we apply graph embedding to integrate the information carried by the supplementary events into the chosen nodes.

Graph embedding is conducted to embed the graph structure into node representation [47], which can make one node in the graph integrate the information carried by the entire graph. Graph embedding often has a clear target to achieve and the vector representations of the nodes are formed via a bunch of training data, while in our setting, we do not have a clear target to set objective function (to integrate information into the chosen nodes is not a clear target to set objective function) and certainly do not have any training data either. In this situation, we follow self-training approach used in word2vec, as shown in [48]. We take random walk to generate a set of paths and take these paths as contexts to adjust the vector representations of the nodes. The graph embedding process used to acquire node representation is shown as follows:

1. Taking $G_i(V, E)$ for example, the event connection graph formed from $text_i$, we treat one node in V as the starting point and choose the successive node via randomly

jumping to one of the adjacent nodes. Repeat this jump for l times. A path of length l can be obtained.

2. Repeat step 1 for m times on each node in V. We then get a path set (noted as PH) whose size is nm, where n denotes the size of V.
3. Each path in PH is treated as one training sample.

Let $v_1, v_2, \ldots, v_{i-1}, v_i, v_{i+1}, \ldots, v_{l-1}, v_l$ denote one path. The values of m and l are set according to [48]. Following self-training setting, we learn a vector representation for v_i to predicate the context of v_i. The loss function for it is:

$$\max P(v_{i-2}, v_{i-1}, v_{i+1}, v_{i+2}|v_i) = max \prod_{j=1}^{2} P(v_{i\pm j}|v_i) \quad (5)$$

where $v_{i-2}, v_{i-1}, v_{i+1}, v_{i+2}$ denote the context of v_i. Two fully-connected layers are used to train the node representation. Softmax is adopted as the output layer. During the training process, for the first iteration, the node v_i integrates the representations of its adjacent nodes, i.e., $v_{i-2}, v_{i-1}, v_{i+1}, v_{i+2}$. For the second iteration, v_i integrates the representations of the nodes which can be linked to v_i through the path whose length is less than 4. As the training process continues, the information carried by the entire graph can be integrated into v_i. After graph embedding, we replace the original vector representations of the nodes in the core event by the tuned ones to let the core event integrate the information carried by the entire text and recalculate the similarity between two texts using Equation (1).

5. Experiments and Analyses

5.1. Experimental Setting

Our similarity calculation aims to obtain the similarity value between two texts from a viewpoint of passage-level event representation. One text may mention several related events. An event connection graph is then constructed to model the relations among those events. In addition, two improvements based on vector tuning are provided to help better construct the event connection graph. Finally, the nodes, indicating the core event mentioned in one text, are chosen to represent the graph. It is worth noting that our calculation is unsupervised. It is not limited on any particular language and any particular domain. To test its compatibility in different languages and different tasks, we build testing corpora in three languages, i.e., English, Chinese, and Spanish. For English, there are many open tasks about text similarity measurement, such as paraphrase and query match in NLU (natural language understanding) [49,50]. We just choose these two tasks to test the performance of our similarity calculation. Ten thousand text pairs are sampled from the corpora for these two tasks. One half includes similar text pairs, and the other half includes dissimilar text pairs. The corpora for these two tasks only contain short sentences. Most of short sentences only mention one event. Our similarity calculation is designed on a passage-level representation perspective and chooses the core event to help accurately measure the similarity between two texts. It is more suitable to handle long text which mentions several events. The former two tasks cannot fully demonstrate the advantage of our calculation on dealing with long text. For this reason, we manually annotate a testing corpus including two thousand long text pairs from Daily news published in the latest one month. For Chinese, we build two testing corpora, one for short text and one for long text. The one for short text is provided by Alibaba company for query match task. The one for long text is manually annotated including two thousand long text pairs from Tencent news also published in the latest one month. For Spanish, there is no suitable open corpus for testing. We have to manually annotate one corpus including two thousand long text pairs from kaggle contest. Among all the manually annotated corpora, we set one half including similar text pairs and the other half including dissimilar text pairs.

The criterion used for evaluation is $F1$. The formulas are shown as follows:

$$P = \frac{r(n)}{t(n)} \quad (6)$$

$$R = \frac{r(n)}{a(n)} \tag{7}$$

$$F1 = 2 * \frac{P * R}{P + R} \tag{8}$$

where P is precision, which is measured by the correctly calculated similar text pairs (noted as $r(n)$) compared with the totally similar text pairs (noted as $t(n)$). R is recall, which is measured by the correctly calculated similar text pairs compared with the totally noted similar text pairs (noted as $a(n)$). $F1$ combines precision and recall together.

There are two kinds of corpora in the experiments. One kind is collected from open tasks, such as paraphrase and query match, with a large number. Sufficient annotated texts enable us to compare our calculation with some supervised similarity calculations. The other kind includes the manually collected long texts, which can be used to demonstrate the advantage of our calculation particularly on dealing with long texts. Regarding the large-scale corpora, we separate them into 8:1:1 for training set, development set, and test set. Three neuron-based algorithms are adopted as baselines in the following experiments. They are textCNN (one convolutional layer, one max-pooling layer, and one softmax output layer), Bi-LSTM (taking Bi-LSTM to encode text and softmax to output similarity value), Bi-LSTM+Bidirectional attention (taking Bi-LSTM to encode text and adding a Bidirectional attention layer to model the interaction between two input texts). In detail, we encode text via textCNN, Bi-LSTM, and Bi-LSTM+Bidirectional attention, respectively. Softmax layer is utilized to output a value to indicate the similarity between two texts. The pre-trained model, i.e., BERT base, is also taken as baseline (like machine reading, one of the fine-tuning tasks in BERT, we input two texts into BERT with a segmentation tag [SEP] and add a softmax layer on [CLS] to output similarity value).

Our calculation is unsupervised. Therefore, we also bring in some unsupervised baselines. We represent input text as vector via the following methods and apply Cosine as similarity function to calculate vector similarity as text similarity.

The applied unsupervised vector representations are listed as follows:

(1) Average: the mean of all the word vectors in the input text.
(2) TextRank+Average: take TextRank to choose keywords from input text and then treat the mean of the chosen keyword vectors as representation.
(3) TextRank+Concatenation: take TextRank to choose keywords and concatenate the word vectors of the chosen keywords to form a long vector.

All the word vectors are set via GloVe.

We also bring in two novel and high-performance text similarity calculating methods, Con-SIM [51] and RCMD [52]. They both based on powerful pertained language model. In addition, multi-head attention and cross-attention are both adopted to model deep interaction between two texts. These two algorithms have proved their high accuracy across some text related tasks. Between them, the first one takes context into consideration to model the training gap among different calculations, and the second one models the distance between sentences as the weighted sum of contextualized token distances.

5.2. Experimental Results

The following experiments are conducted in six aspects. Section 5.2.1 shows the experiments to test the rationality of the threshold setting in our calculation. Section 5.2.2 demonstrates the performances of our calculation on the condition that the event connection graph is constructed by different event extraction methods. Section 5.2.3 shows some sampled examples to explicitly demonstrate the ability of our calculation. Section 5.2.4 shows the results of our calculation comparing with the supervised and unsupervised baseline algorithms on the testing corpora. Section 5.2.5 shows the ablation results to see the enhancement brought from two improvements provided in Section 4. In Section 5.2.6, we add an experiment to prove that our calculation is not limited on any particular language and any particular domain.

5.2.1. Testing on Threshold

In Section 4, we provide a vector tunning-based method to find semantically similar triggers and link them to form a comprehensive event connection graph. Two triggers whose semantic similarity value is beyond the threshold (set as 0.8) can be connected in the graph. The following figure just demonstrates and explains the rationality of the threshold setting. It shows *F*1 values when the threshold changes from 0.1 to 1.0. This experiment is designed to see the rationality of the chosen of the threshold setting.

As shown in Figure 5, the calculating results change along with the variety of the value of the threshold. All the curves have the similar trend across different corpora. They all reach the perk at the value of 0.8 (or close to it). The reason can be explained based on the principle behind word embedding. For most of word pairs, if two words are semantically similar, their pre-trained vector representations are close. The vector representation of trigger is initialized via GloVe, one kind of word embedding; thus, we can take the value between two triggers measured by vector distance to decide whether two triggers are semantically similar or not. Trigger decides event type. Two similar triggers just indicate two related or similar events. However, as indicated in Section 4.1, due to the situation that there are some commonly used collocations in linguistics, the previous conclusion (similar triggers have close vectors) is not always true. Thus, in Section 4.1, we tune the pre-trained vector representations of the triggers via the training data sampled from synonym dictionaries to make semantically similar triggers own close vectors. Based on the tuning operation, finding a threshold to decide whether two triggers are similar becomes feasible. As shown in Figure 5, 0.8 is a reasonable choice, where the performance curves reach the peak through all the testing corpora. When the threshold exceeds 0.8, the performance curves even drop. This is because when the threshold enlarges, some similar triggers are missed to be connected. The relations among events, especially the relations among the semantically similar events, cannot be fully covered by the event connection graph. It finally causes that the extracted event may not be the core event. Furthermore, missing the connections between similar events, the extracted event cannot integrate the information carried by the entire event connection graph. These two situations lead to the drop of the performance curves.

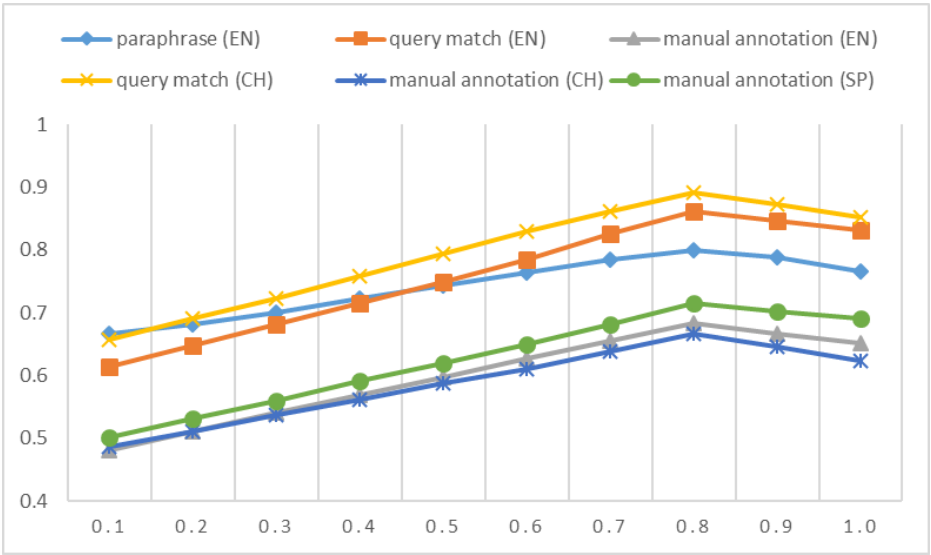

Figure 5. *F*1 values when we change the threshold from 0.1 to 1.0. The threshold is utilized to decide whether to connect two triggers in the event connection graph or not.

5.2.2. Comparison of Different Event Extraction Methods

Our similarity calculation needs to construct an event connection graph to reveal the core event to calculate text similarity. In our paper, this graph is constructed by the

sentence-level event extraction method shown in [40]. We note this method as OneIE as it is called in [40]. There raises a doubt that whether different event extraction methods affect the final calculating results or not. Therefore, we design an experiment to see the similarity calculation results with event connection graphs constructed by different event extraction methods across all the testing corpora. The following table just illustrates the performances of our calculation on the condition that the event connection graph is constructed via several popular sentence-level event extraction methods. The event connection graph includes both trigger and argument; thus, the chosen event extraction methods should jointly extract trigger and argument meanwhile. We choose BeemSearch [53], JointTransition [54], and ODEE-FER [55] as baselines. BeemSearch is one of the classic joint event extraction methods, which encodes text via one-hot feature and applies local and global features to label trigger and argument meanwhile. JointTransition and ODEE-FER are both based on neuron model. The significant difference between them is that JointTransition applies transition model to characterize the relation between trigger and argument, while ODEE-FER integrates latent variable into neuron model to extract open-domain event free from event schema predefinition. Multi-task learning framework is utilized in ODEE-FER to identify trigger and argument concurrently. The corpora for paraphrase, query match, and manually annotated are abbreviated as Para, Q&Q, and MA, which are also used in the following tables.

As shown in Table 2, it can be found that different event extraction methods do not affect the performances of our calculation much. This situation is due to the following two reasons. First of all, the construction of event connection graph is only the first step in our calculation. The extracted triggers and arguments are subsequently measured to indicate their importance via centrality measurement. During the measuring process, the incorrectly extracted triggers and arguments can be eliminated. In Table 2, we also add an extreme case, noted as Extreme listed in the last row, where we treat a verb in the sentence as trigger and noun as argument. If there is more than one trigger in the sentence, we construct polygon for each trigger following the approach shown in Section 3. It can be observed that the result obtained from Extreme is a little different from the ones obtained from the other methods. That indicates we do not need the precise event extraction results, as long as the extracted results contain enough triggers and arguments. Furthermore, two improvements proposed in Section 4 can also help remove the adverse effect brought from the incorrectly extracted triggers and arguments. In detail, one improvement tunes the vector representation of the trigger, and links the semantically similar triggers in the event connection graph. Regarding the triggers incorrectly extracted, they are little related to the core event mentioned in the text. Thus, these triggers do not locate at the center of the event connection graph. After the measuring process, these triggers will be valued with little weights. They will not be chosen as the core event to measure text similarity. The other improvement is to integrate the information carried by the entire text into the chosen nodes via graph embedding. In some cases, even if the incorrectly extracted triggers and arguments are chosen as the core event, after graph embedding, the information carried by the entire text can be integrated into the incorrectly extracted triggers and arguments. This way can also alleviate the adverse effect brought from the incorrect extraction.

Table 2. The results of our calculation obtained on the condition that the event connection graph is constructed via several popular event extraction methods (highest values in bold).

Methods	English			Chinese		Spanish
	Para	Q&Q	MA	Q&Q	MA	MA
BeemSearch	0.81	0.84	0.66	0.85	0.62	0.67
JointTransition	0.79	0.83	0.64	0.84	0.63	0.68
ODEE-FER	**0.82**	0.85	**0.68**	0.86	0.65	**0.71**
OneIE	**0.82**	**0.86**	**0.68**	**0.89**	**0.67**	**0.71**
Extreme	0.77	0.82	0.63	0.82	0.61	0.66

To clearly see the effects brought from different event extraction methods, we also draw a histogram figure to illustrate the calculating results obtained by different sentence-level event extraction methods. In addition, to compare their results in two tasks of text similarity calculation and event extraction, we show the results obtained by different event extraction methods in both tasks in two colors. In this figure, each algorithm corresponds to two columns. The column with blue color indicates the text similarity calculating results, which is averaged across all the corpora including open tasks and manual annotation. The column with orange color indicates the event extracting results, which is averaged across the corpora from ACE and FrameNet, two popular event extraction tasks.

As shown in Figure 6, it is hard to see the performance gap among all the event extraction methods in text similarity calculation task. However, the event extraction methods adopted in the experiment perform differently in the event extraction task. As shown in this figure, ODEE-FER performs much better than the other event extraction methods. Anyway, in this experiment, we also design a simple method called Extreme. This method just simply treats a verb in the sentence as trigger and a noun as argument. It is easy to assume that Extreme should have a poor performance in the event extraction task. The result also proves this assumption. Surprisingly, Extreme obtains almost the same result with the other event extraction methods in the text similarity calculation task. That indicates we do not need the precise event extraction results, as long as the extracted results contain enough triggers and arguments.

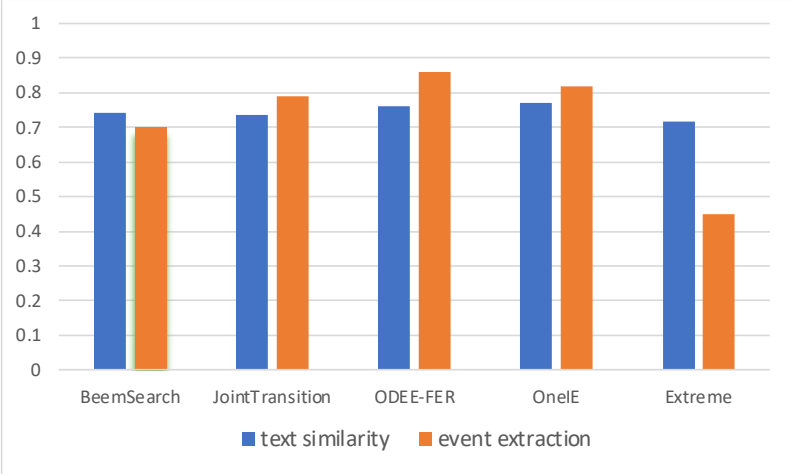

Figure 6. The histogram to see the results obtained by different event extraction methods (blue color indicates text similarity calculation task and orange color indicates event extraction task).

5.2.3. Case Study

In our similarity calculation, we need to extract some nodes from the event connection graph to represent the core event mentioned in the text. In the following table, we just show the extracted nodes (i.e., triggers and arguments) from some sampled texts. We sample 6 texts from our English testing corpora, three for long texts (noted as A1, A2, A3) and three for short texts (noted as S1, S2, S3). The short texts are news caption. The contents of the chosen texts can be found in (A1: https://www.bbc.com/news/technology-52391759; A2: http://news.bbc.co.uk/sport2/hi/football/europe/8591081.stm; A3: https://www.bbc.com/news/business-52467965; S1: https://www.bbc.com/news/uk-51259479; S2: https://www.bbc.com/news/business-44789823; S3: https://www.bbc.com/news/business-52466864 (accessed on 17 May 2022). These texts can be accessed till the pages are deleted). Since keywords can also be treated as the content representation of the text, we also show the keyword extraction results (the top five keywords) obtained by LDA and TextRank, two popular unsupervised keyword extraction algorithms.

As shown in Table 3, among the chosen words, some are the same across different extraction methods while some are distinct. Taking the contents of the sampled texts into

consideration, the words extracted by our calculation can exactly cover the core event mentioned in the sampled texts, though the number of extracted words is often less the number from the other two keyword extraction methods. On the contrary, the words extracted by TextRank and LDA are not always related to the core event. This situation is obvious for long texts. For example, for A1, its core event is "Apple iPhone has a software leak on email app". The keywords extracted from this text via TextRank and LDA both include "ZecOps". This word repeats many times in A1. Thus, it is chosen as keyword, whereas this word indicates the source where the news is published. It is not the part of the core event. The reason to this situation is that traditional keyword extraction methods often take the shallow statistics, such as frequency or distribution, to measure word importance. Such an approach causes "ZecOps" to be incorrectly chosen. In our calculation, when measuring the importance of one word, we consider the effect of the event which includes this word. In detail, only if the event is emphasized by one text, the word included by this event can be treated as the representation of this text. In the text of long length, there may mention several events. The frequently occurring words may not be included by the core event, such as the words "ZecOps" and "Rooney" included by A1 and A2. Thus, they may be extracted incorrectly. Regarding the texts of short length, they only include one or two sentences. A few events are mentioned. The frequently occurring words are mostly included by the core event. Thus, among the short texts, the words extracted by the three methods are almost same.

Table 3. The extracted words from the sampled texts (* marks that less than five words can be extracted).

Methods	TextRank	LDA	Ours
		Long Texts	
A1	ZecOps, Apple, mail, leak, hacker	Apple, mobile, ZecOps, bug, hacker	Apple, mail, software, leak, *
A2	Rooney, soccer, Bayern Munich, injury, champion	soccer, Bayern Munich, beat, Man Utd, Rooney	Bayern Munich, beats, Man Utd, *
A3	Barclay, bank, economic, coronavirus, profit	Barclay, coronavirus, bank, pandemic, work	coronavirus, pandemic, costs, £2.1bn, *
		Short Texts	
S1	Carmaker, Tesla, build, factory, Shanghai	Carmaker, Tesla, build, factory, Shanghai	Tesla, build, factory, Shanghai, *
S2	Kobe Bryant, death, BBC, TV news, mistake	BBC, apologize, footage, mistake, *	BBC, apologize, footage, mistake, *
S3	Coronavirus, economy, sink, pandemic, shutdown	Coronavirus, economy, sink, shutdown, *	Economy, sink, pandemic, shutdown, *

5.2.4. Comparison of Different Algorithms

The following table shows the results of comparing our similarity calculation with the supervised and unsupervised baseline algorithms. The supervised baseline algorithms include textCNN, LSTM, LSTM+Bidirectional attention (abbreviated as LSTM+BIA), and BERT-base. They are conducted only on the large-scale testing corpora including short sentences, since these corpora have enough data to form training set. The unsupervised baseline algorithms include Average (abbreviated as AVE), TextRank+Average (abbreviated as TR+AVE), and TextRank+Concatenation (abbreviated as TR+CON). The details of the baseline algorithms are already told at the beginning of Section 5.

For unsupervised similarity calculations, to test their similarity calculating results, we first represent input text as vector. This vector is averaged from all the word vectors through input text, or some chosen word vectors by TextRank algorithm. Then, Cosine similarity is used to decide whether two texts are similar or not. For supervised similarity calculations, we also represent input text as vector, while this vector is formed by different encoder models like LSTM, Bi-LSTM, or pretrained models (BERT or RoBERTa). Then, two

vectors obtained by different encoders are sent to MLP layer to output a probability in terms of softmax to indicate similarity results. All the testing algorithms output a value to measure the similarity between two texts. We record the similarity values of all the text pairs in the testing corpora via the given algorithms and take the mean of all the values as the threshold to decide whether two texts are similar or not. To make the obtained results more persuasive, we add significant test. We separate each testing corpus into ten parts, and record calculating results in each part. Two-tail paired t-test is applied to determine whether the results obtained by different algorithms over the ten times' calculations are significantly different or not. We set three significant levels as 0.01, 0.05, and 0.1 (labelled as ***, **, and *).

As shown in Table 4, we list the results obtained in different languages and in different tasks. It can be found that supervised algorithms overwhelm unsupervised algorithms by a large margin in all the testing corpora. The reason is straightforward, since supervised algorithms can utilize training data to obtain a reasonable hyperplane to separate similar text pairs from dissimilar ones. Correspondingly, unsupervised algorithms cannot acquire any transcendental guidance to help model the discrimination between similar and dissimilar text pairs. They only depend on data's natural distribution, thus, obtain lower performance. Anyway, unsupervised algorithms only choose some words with prominent distribution or aggregate all the words in the text to generate text representation, whereas long text has many words which are little related to the main content. This situation causes unsupervised algorithms obtain extremely lower performances on the manually annotated corpora which include only long texts. It can be found that our calculation obtains comparable results with supervised baseline algorithms and performs much better than unsupervised baseline algorithms especially on the corpora including long texts. The reason is totally due to our event connection graph. Based on this graph, the nodes (or words), which can represent the core event mentioned in one text, can be finally extracted. The unrelated noisy words are ignored when calculating text similarity. For this reason, we can acquire accurate similarity results on both long and short texts. Besides, to further improve performance, graph embedding is used to encode the information carried by the entire graph into the chosen nodes to make the chosen nodes carry the global information expressed by entire text. The significant test results also prove the reliability of the high performance of our calculation.

Table 4. The comparison of our calculation with the baseline algorithms (highest values in bold. ***, **, and * indicate three significant levels as 0.01, 0.05, and 0.1.).

Methods		English			Chinese		Spanish
		Para	Q&Q	MA	Q&Q	MA	MA
supervised	TextCNN	0.84 ***	0.82 **	—	0.86 ***	—	—
	LSTM	0.83 **	0.84 **	—	0.83 **	—	—
	LSTM+BIA	0.83 ***	0.87 **	—	0.89 ***	—	—
	BERT-base	0.85 ***	0.89 ***	—	0.91 ***	—	—
	Con-SIM	0.87 ***	0.90 ***	—	0.92 ***	—	—
	RCMD	**0.90** ***	**0.92** ***	—	**0.93** ***	—	—
unsupervised	AVE	0.58 *	0.61 **	0.42 *	0.63 **	0.49 *	0.48 **
	TR+AVE	0.65 **	0.66 **	0.49 **	0.68 **	0.55 **	0.59 **
	TR+CON	0.69 *	0.71 ***	0.53 **	0.66 **	0.53 **	0.57 **
	Ours	0.82 ***	0.86 ***	**0.68** ***	0.89 ***	**0.67** ***	**0.71** ***

In the experiments, there are also two SOTA text similarity calculations, Con-SIM and RCMD. They both obtain extraordinary results. Moreover, it can be observed that RCMD even obtains over 90% F1 value across all the testing corpora. These two methods both have two layers. The lower layer is text encoder, where Con-SIM takes BERT as the encoder and RCMD takes a more powerful model (RoBERTa) as encoder. The difference is that RoBERTa has more parameters; thus, RCMD obtains higher performance. The upper level is the interaction layer, where Con-SIM takes hierarchical interactive attention and

RCMD takes two matrixes to model local interaction (inner sentence) and global interaction (outer sentence) to obtain sentence matching results. As indicated in the experiments, our proposed method obtains lower performance than those SOTA methods. However, it is easy to be explained. Those SOTA methods all take pretrained models to encode input texts with massive parameters. Anyway, those methods need to consume training data to adjust model to deal with domain-specific data. As domain changed, these methods are easy to be distorted as shown in the experiments. Compared with those methods, our proposed method is unsupervised based. Thus, it keeps its performance across domains. Anyway, with the help of passage-level document representation, our proposed method can obtain high performance. Though the performance is lower than the ones with pretrained models, its performance is much higher than the baseline unsupervised methods.

5.2.5. Ablation Results

In Section 4, we provide two improvements on our calculation. One is to detect and link similar triggers to involve the relations among the similar or related events into the event connection graph. The other is node representation via graph embedding, which lets the representations of the chosen nodes integrate the information carried by the entire event connection graph. In the following table, we record the results obtained by our calculation in the following two settings. One is with or without linking similar triggers. The other is with or without node representation.

As shown in Table 5, it can be found that two improvements both enhance the performance of our calculation. Between them, node representation brings more boost. The advance brought from these two improvements is easy to be explained. Regarding the advance brought from the linkage of similar triggers, since we find similar triggers via tuning their vector representations and further link them, our event connection graph can cover more relations among events. Via this comprehensive graph, we can locate the core event more accurately. Regarding the advance brought from node representation, the pre-trained vector representations of the chosen nodes can only express their inherent information, i.e., only representing the local information carried by the chosen nodes. After we tune node representation via graph embedding, the vector representations of the chosen nodes can integrate the information carried by the entire event connection graph. Via these nodes, text similarity can be calculated more accurately. Anyway, the representations of the chosen nodes after graph embedding can cover both the inherent information themselves and the information carried by the global graph. It can bring more boost on the performance than the linkage of similar triggers.

Table 5. The ablation results of our calculation (highest values in bold).

Methods	English			Chinese		Spanish
	Para	Q&Q	MA	Q&Q	MA	MA
Only via graph	0.76	0.80	0.55	0.81	0.56	0.57
+linking triggers	0.78	0.82	0.59	0.85	0.60	0.62
+node representation	0.81	0.84	0.63	0.87	0.65	0.69
+linking triggers and node representation	**0.82**	**0.86**	**0.68**	**0.89**	**0.67**	**0.71**

5.2.6. Task Transferring

Text similarity calculation is a fundamental component of many artificial intelligence applications. We cannot predefine the domain and the task where these applications are applied. We then add a test to compare the capacities of different algorithms in the transferring scenario across different tasks and languages. The testing corpora are collected from different domains and different languages. Regarding supervised algorithms, we train them on one corpus and test them on another corpus. There are three kinds of corpora in English, two in Chinese, and one in Spanish. For English, we combine two corpora as

training set and test the algorithms on the remaining corpus. For Chinese, we train the algorithms on one corpus and test them on the other corpus. For Spanish, since we only have one corpus, we do not test the algorithms in this language.

To conduct the experiments on the task transferring scenario, we take supervised algorithms and unsupervised algorithms in two ways. Regarding unsupervised algorithms, since they do not have training stage, we run them directly on each corpus and record the results. Anyway, since both languages (English and Chinese) have the corpus about query match task, we test all the algorithms on this task while training on one language and testing on the other language (noted as E-C and C-E indicating English to Chinese and Chinese to English). In this test, all the algorithms are given cross-lingual word embeddings trained on the corpus formed via sentence alignment [56]. We also add significant test to see the credibility of the obtained results.

As shown in Table 6, it can be found that supervised algorithms degrade much compared with the results shown in Table 4. In Table 4, the results are obtained in the situation that training and testing are performed on the same corpus. This phenomenon indicates that task transferring (or corpus changing) deeply affects the performance of supervised algorithms. The reason is obvious. Since supervised algorithms count on the transcendental knowledge (this knowledge indicates data distribution assumption) derived from training corpus to deal with novel data, they are easy to be distorted by the other corpus which owns diverse distribution. On the contrary, there is no training corpus for unsupervised algorithms. Thus, they do not make any assumption about data distribution, which causes they are not affected by task transferring (or corpus changing). Our calculation is one kind of unsupervised algorithms. It keeps high quality across all the corpora. The significant test results prove the reliability of the high quality of our calculation.

Table 6. The results of all the algorithms in the transferring scenario (highest values in bold. ***, **, and * indicate three significant levels as 0.01, 0.05, and 0.1.).

Methods		English			Chinese		C-E	E-C
		Para	Q&Q	MA	Q&Q	MA	Q&Q	Q&Q
supervised	TextCNN	0.61 ***	0.60 **	0.49 **	0.47 *	0.51 *	0.31 **	0.27 **
	LSTM	0.56 **	0.59 **	0.51 *	0.54 **	0.52 **	0.23 *	0.22 *
	LSTM+BIA	0.54 **	0.53 **	0.47 *	0.55 **	0.56 **	0.26 *	0.24 *
	BERT-base	0.57 ***	0.55 ***	0.53 **	0.49 ***	0.51 **	0.48 **	0.41 **
	Con-SIM	0.52 ***	0.53 ***	0.51 *	0.47 ***	0.50 *	0.44 **	0.38 **
	RCMD	0.53 ***	0.56 ***	0.54 *	0.46 ***	0.53 **	0.42 **	0.43 **
unsupervised	AVE	0.58 *	0.61 **	0.42 *	0.63 **	0.49 *	0.61 **	0.63 **
	TR+AVE	0.65 **	0.66 **	0.49 **	0.68 **	0.55 **	0.66 **	0.68 **
	TR+CON	0.69 *	0.71 ***	0.53 **	0.66 **	0.53 **	0.71 ***	0.66 **
	Ours	**0.82** ***	**0.86** ***	**0.68** ***	**0.89** ***	**0.67** ***	**0.86** ***	**0.89** ***

6. Conclusions and Future Work

Text similarity calculation is a fundamental task for many high-level artificial intelligence applications, such as text clustering, text summarization, and Q&A. Traditional similarity calculations are conducted in terms of either making two similar texts close in a high-dimensional space (supervised methods) or measuring the number of concurrent words shared by two texts (unsupervised methods). They ignore a fact that, in many scenarios, text is used to record events. Text similarity is mostly decided by whether two texts mention the same core event or not. This paper just proposes a novel text similarity calculation via constructing an event connection graph to disclose the core event mentioned in one text. To better model the relations among events, we tune the vectors of the triggers to detect related events and link them in the event connection graph. This approach can locate the core event more accurately. The nodes which can represent the core event are chosen and utilized to measure text similarity. Moreover, we adopt graph embedding to

tune the vectors of the chosen nodes to integrate the global information carried by the entire text into the chosen nodes. This way can further boost the performance of our calculation. Experimental results prove the high performance of our similarity calculation.

Though our paper can combine the merits from supervised and unsupervised similarity calculations and can be applied in many text-related downstream applications which need text similarity as their component. Our calculation has time issue needed to be further solved. In particular, our calculation needs to form a passage-level event representation. This kind of operation needs extra time. Thus, though our calculation has higher accuracy, it is not fit to online applications, especially some time-insensitive applications.

One issue needed to be mentioned is that, to link semantically similar triggers to let our event connection graph cover more relations among events, we need to predefine a threshold to decide whether two triggers are similar or not. As shown in the experiments, this parameter setting is not optimal for some corpus. It is chosen via balancing the results across all the testing corpora. Therefore, in the future work, we hope to set it dynamically. The other work we hope to carry out is to improve efficiency. The process of graph construction is time-consuming. We hope to construct some template graphs at advance. During the calculating stage, we choose the corresponding template graph via some matching score and complete the matched template graph using some specific words chosen from input text.

Author Contributions: M.L. and Z.Z. conceived and designed the study, interpreted the data, and drafted the manuscript. L.C. guided the study and revised the manuscript. All authors critically revised the manuscript, agree to be fully accountable for ensuring the integrity and accuracy of the work, and read and approved the final manuscript before submission. All authors have read and agreed to the published version of the manuscript.

Funding: The research in this article is supported by the National Key Research and Development Project (2021YFF0901600), the Project of State Key Laboratory of Communication Content Cognition (A02101), the National Science Foundation of China (61976073, 62276083), and Shenzhen Foundational Research Funding (JCYJ20200109113441941).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Acknowledgments: I would like to express my gratitude to all those who have helped me during the writing of this thesis, including all the co-authors. I also appreciate the help from the reviewers' directions.

Conflicts of Interest: This submission is an extension version of a conference paper written by the same authors in NLPCC, however, I confirm that over 60% part including methods and experiments are added. Besides, the results are improved by the novel proposed method. All authors are aware of the submission and agree to its review. There is no conflict of interest with Associate Editors. The coverage of related work is appropriate and up-to-date.

References

1. Ji, H.; Grishman, R. Refining event extraction through cross-document inference. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Columbus, OH, USA, 16–18 June 2008; pp. 254–262.
2. Baker, C.; Fillmore, C.; Lowe, J. The berkeley framenet project. In Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics, Montreal, QC, Canada, 10–14 August 1998; pp. 86–90.
3. Jacksi, K.; Ibrahim, R.; Zeebaree, S.; Zebari, R.; Sadeeq, M. Clustering documents based on semantic similarity using HAC and K-mean algorithms. In Proceedings of the 2020 International Conference on Advanced Science and Engineering, Duhok, Iraq, 23–24 December 2020; pp. 205–210.
4. Huang, X.; Qi, J.; Sun, Y.; Zhang, R. Mala: Cross-domain dialogue generation with action learning. In Proceedings of the 34th AAAI Conference on Artificial Intelligence, New York, NY, USA, 7–12 February 2020; pp. 7977–7984.

5. Kieu, B.; Unanue, I.; Pham, S.; Phan, H.; Piccardi, M. Learning neural textual representations for citation recommendation. In Proceedings of the 25th International Conference on Pattern Recognition, Milan, Italy, 10–15 January 2021; pp. 4145–4152.
6. Chen, Y.; Zhou, M.; Wang, S. Reranking answers for definitional QA using language modeling. In Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics, Sydney, Australia, 17–21 July 2006; pp. 1081–1088.
7. Turian, J.; Ratinov, L.; Bengio, Y. Word representations: A simple and general method for semi-supervised learning. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, 11–16 July 2010; pp. 384–394.
8. Mikolov, T.; Sutskever, I.; Chen, K.; Corrado, G.; Dean, J. Distributed representations of words and phrases and their compositionality. In Proceedings of the Advances in Neural Information Processing Systems, Lake Tahoe, NV, USA, 5–10 December 2013; pp. 3111–3119.
9. Mikolov, T.; Chen, K.; Corrado, G.; Dean, J. Efficient estimation of word representations in vector space. In Proceedings of the 1st International Conference on Learning Representations, Scottsdale, AZ, USA, 2–4 May 2013; pp. 1–12.
10. Pennington, J.; Socher, R.; Manning, C. GloVe: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1532–1543.
11. Peters, M.; Neumann, M.; Iyyer, M.; Gardner, M.; Clark, C.; Lee, K.; Zettlemoyer, L. Deep contextualized word representations. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, New Orleans, LA, USA, 1–6 June 2018; pp. 2227–2237.
12. Chen, J.; Dai, X.; Yuan, Q.; Lu, C.; Huang, H. Towards interpretable clinical diagnosis with Bayesian network ensembles stacked on entity-aware CNNs. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 3143–3153.
13. Cho, K.; van Merriënboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F.; Schwenk, H.; Bengio, Y. Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, Doha, Qatar, 25–29 October 2014; pp. 1724–1734.
14. Behera, R.K.; Jena, M.; Rath, S.K.; Misra, S. Co-LSTM: Convolutional LSTM model for sentiment analysis in social big data. *Inf. Process. Manag.* **2020**, *58*, 102435. [CrossRef]
15. Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A.; Kaiser, L.; Polosukhin, I. Attention is all you need. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 5998–6008.
16. Radford, A.; Narasimhan, K.; Salimans, T.; Sutskever, I. Improving Language Understanding by Generative Pre-Training. 2018. Available online: https://S3-Us-West-2.Amazonaws.Com (accessed on 31 December 2021).
17. Devlin, J.; Chang, W.; Lee, K.; Toutanova, K. BERT: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Minneapolis, MN, USA, 2–7 June 2019; pp. 4171–4186.
18. Yang, Z.; Dai, Z.; Yang, Y.; Carbonell, J.; Salakhutdinov, R.; Le, V. XLNET: Generalized autoregressive pretraining for language understanding. In Proceedings of the Advances in Neural Information Processing Systems, Vancouver, BC, Canada, 8–14 December 2019; pp. 5753–5763.
19. Liu, Y.; Ott, M.; Goyal, N.; Du, J.; Joshi, M.; Chen, D.; Levy, O.; Lewis, M.; Zettlemoyer, L.; Stoyanov, V. RoBERTa: A robustly optimized BERT pretraining approach. *arXiv* **2019**, arXiv:1907.11692.
20. Pan, Y.; Yao, T.; Li, Y.; Mei, T. X-Linear attention networks for image captioning. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 14–19 June 2020; pp. 10971–10980.
21. Liu, Y.; Zhang, X.; Zhang, Q.; Li, C.; Huang, F.; Tang, X.; Li, Z. Dual self-attention with co-attention networks for visual question answering. *Pattern Recognit.* **2021**, *117*, 107956. [CrossRef]
22. Lee, L.H.; Wan, C.H.; Rajkumar, R.; Isa, D. An enhanced Support Vector Machine classification framework by using Euclidean distance function for text document categorization. *Appl. Intell.* **2011**, *37*, 80–99. [CrossRef]
23. Zhu, H.; Zhang, P.; Gao, Z. K-means text dynamic clustering algorithm based on KL divergence. In Proceedings of the 17th IEEE/ACIS International Conference on Computer and Information Science, Singapore, 6–8 June 2018; pp. 659–663.
24. Huang, A. Similarity measures for text document clustering. In Proceedings of the 6th New Zealand Computer Science Research Student Conference, Hamilton, New Zealand, 14–18 April 2008; pp. 9–56.
25. Atoum, I.; Otoom, A. Efficient Hybrid Semantic Text Similarity using Wordnet and a Corpus. *Int. J. Adv. Comput. Sci. Appl.* **2016**, *7*, 124–130. [CrossRef]
26. Gomaa, W.; Fahmy, A. A survey of text similarity approaches. *Int. J. Comput. Appl.* **2013**, *68*, 13–18.
27. Robertson, S. Understanding inverse document frequency: On theoretical arguments for IDF. *J. Doc.* **2004**, *60*, 503–520. [CrossRef]
28. Mihalcea, R.; Tarau, P. Textrank: Bringing order into text. In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, Barcelona, Spain, 25–26 July 2004; pp. 404–411.
29. Blei, D.; Ng, A.; Jordan, M. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
30. Pavlinek, M.; Podgorelec, V. Text classification method based on self-training and LDA topic models. *Expert Syst. Appl.* **2017**, *80*, 83–93. [CrossRef]
31. Sharif, U.; Ghada, E.; Atlam, E.; Fuketa, M.; Morita, K.; Aoe, J.-I. Improvement of building field association term dictionary using passage retrieval. *Inf. Process. Manag.* **2007**, *43*, 1793–1807. [CrossRef]

32. Dorji, T.C.; Atlam, E.-S.; Yata, S.; Fuketa, M.; Morita, K.; Aoe, J.-I. Extraction, selection and ranking of Field Association (FA) Terms from domain-specific corpora for building a comprehensive FA terms dictionary. *Knowl. Inf. Syst.* **2010**, *27*, 141–161. [CrossRef]
33. Malkiel, I.; Ginzburg, D.; Barkan, O.; Caciularu, A.; Weill, J.; Koenigstein, N. Interpreting BERT-based Text Similarity via Activation and Saliency Maps. In Proceedings of the 2022 ACM Web Conference, Lyon, France, 25–29 April 2022; pp. 3259–3268.
34. Grishman, R.; Sundheim, B. Message understanding conference-6: A brief history. In Proceedings of the 16th International Conference on Computational Linguistics, Copenhagen, Denmark, 5–9 August 1996; pp. 466–471.
35. Doddington, G.; Mitchell, A.; Przybocki, M.; Ramshaw, L.; Strassel, S.; Weischedel, R. The Automatic Content Extraction (ACE) program-tasks, data, and evaluation. In Proceedings of the 4th International Conference on Language Resources and Evaluation, Centro Cultural de Belem, Lisbon, 24–30 May 2004; pp. 837–840.
36. Yang, S.; Feng, D.; Qiao, L.; Kan, Z.; Li, D. Exploring pre-trained language models for event extraction and generation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 169–175.
37. Wang, Z.; Wang, X.; Han, X.; Lin, Y.; Hou, L.; Liu, Z.; Li, P.; Li, J.; Zhou, J. CLEVE: Contrastive pre-training for event extraction. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, Online, 1–6 August 2021; pp. 6283–6297.
38. Du, X.; Cardie, C. Document-Level event role filler extraction using multi-granularity contextualized encoding. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Washington, DC, USA, 5–10 July 2020; pp. 8010–8020.
39. Hu, Z.; Liu, M.; Wu, Y.; Xu, J.; Qin, B.; Li, J. Document-level event subject pair recognition. In Proceedings of the 9th CCF International Conference on Natural Language Processing and Chinese Computing, Zhengzhou, China, 14–18 October 2020; pp. 283–293.
40. Lin, Y.; Ji, H.; Huang, F.; Wu, L. A joint neural model for information extraction with global features. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Washington, DC, USA, 5–10 July 2020; pp. 7999–8009.
41. Liao, S.; Grishman, R. Using document level cross-event inference to improve event extraction. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, Uppsala, Sweden, 11–16 July 2010; pp. 789–797.
42. Atlam, E.-S.; Elmarhomy, G.; Morita, K.; Fuketa, M.; Aoe, J.-I. Automatic building of new Field Association word candidates using search engine. *Inf. Process. Manag.* **2006**, *42*, 951–962. [CrossRef]
43. Lai, D.; Lu, H.; Nardini, C. Finding communities in directed networks by PageRank random walk induced network embedding. *Phys. A Stat. Mech. Its Appl.* **2010**, *389*, 2443–2454. [CrossRef]
44. Li, P.; Zhou, G.; Zhu, Q.; Hou, L. Employing compositional semantics and discourse consistency in Chinese event extraction. In Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, Jeju Island, Korea, 12–14 July 2012; pp. 1006–1016.
45. Amir, H.; Béatrice, D. Word embedding approach for synonym extraction of multi-word terms. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation, Miyazaki, Japan, 7–12 May 2018; pp. 297–303.
46. Davenport, E.; Cronin, B. Knowledge management: Semantic drift or conceptual shift? *J. Educ. Libr. Inf. Sci.* **2000**, *1*, 294–306. [CrossRef]
47. Grover, A.; Leskovec, J. Node2vec: Scalable feature learning for networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 855–864.
48. Sasano, R.; Korhonen, A. Investigating word-class distributions in word vector spaces. In Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Online, 5–10 July 2020; pp. 3657–3666.
49. Kennington, C. Enriching language models with visually-grounded word vectors and the lancaster sensorimotor norms. In Proceedings of the 25th Conference on Computational Natural Language Learning, Online, 10–11 November 2021; pp. 148–157.
50. Mysore, S.; Cohan, A.; Hope, T. Multi-vector models with textual guidance for fine-grained scientific document similarity. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Seattle, Washington, DC, USA, 10–15 July 2022; pp. 4453–4470.
51. Sun, X.; Meng, Y.; Ao, X.; Wu, F.; Zhang, T.; Li, J.; Fan, C. Sentence similarity based on contexts. *Trans. Assoc. Comput. Linguist.* **2022**, *10*, 573–588. [CrossRef]
52. Lee, S.; Lee, D.; Jang, S.; Yu, H. Toward interpretable semantic textual similarity via optimal transport-based contrastive sentence learning. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Dublin, Ireland, 22–27 May 2022; pp. 5969–5979.
53. Li, Q.; Ji, H.; Huang, L. Joint event extraction via structured prediction with global features. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, Sofia, Bulgaria, 4–9 August 2013; pp. 73–82.
54. Zhang, J.; Qin, Y.; Zhang, Y.; Liu, M.; Ji, D. Extracting entities and events as a single task using a transition-based neural model. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, Macao, China, 10–16 August 2019; pp. 5422–5428.
55. Liu, X.; Huang, H.; Zhang, Y. Open domain event extraction using neural latent variable models. In Proceedings of the 57th Conference of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 2860–2871.
56. Levy, O.; Søgaard, A.; Goldberg, Y. A strong baseline for learning cross-lingual word embeddings from sentence alignments. In Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics, Valencia, Spain, 3–7 April 2017; pp. 765–774.