A Keyphrase Graph-Based Method for Document Similarity Measurement

ThanhThuong T. Huynh, TruongAn PhamNguyen, and Nhon V. Do

***Abstract*—Measuring similarity between texts is an essential task in a large variety of applications. Contemporary approaches for this task rely heavily on statistical and lexical information to represent text. They thus produce opaque and hard to interpret models that could be hard to adapt in some applications and hamper the user experience. To represent the text document more interpretable, we propose a graph-based semantic model that integrates more semantic information among keyphrases as well as the structural information of the text. The utilization of large knowledge bases (e.g. DBpedia, Wikipedia) makes available fine-grained information about concepts, entities, and their semantic relations, thus resulting in a knowledge-rich interpretation. The relevance evaluation between two documents can then be performed by calculating the semantic similarity between two keyphrase graphs that represent them. The final result comes close in performance to the specialized black-box methods particularly tuned to this task on a traditional dataset.**

***Index Terms*—Document representation, Graph-based document model, Keyphrase Extraction, Document similarity, Graph matching**

I. Introduction

Document representation is a fundamental task required in the vast majority of natural language processing systems, with applications as diverse as document retrieval, clustering, classification, document similarity assessment, and document summarization. The most challenging aspect of this task is capturing as much of the text's underlying semantic information as possible in a computer-readable model.

Early statistical approaches like Bag Of Words and Vector Space Models characterize documents as (features, weights) pairs, where the feature could be a single word or phrase extracted from the body of the text. Furthermore, in order to construct highly discriminative representations of various texts, such features are also provided with weights or probabilities. For the document similarity assessment problem, the methods that use the document representation in the manner mentioned above rely mostly on the exact match of the keywords identified as appearing in the two documents; they do not take into consideration diverse meanings of the same word or synonyms.

Many studies have built on this principle by providing more complicated and effective features that are based on additional conceptual aspects rather than just simple words. Lemmas, N-grams, Noun Phrases, phrases produced by syntactic relations like subject-verb-object, and phrases comprising non-contiguous words in texts are some well-known complex feature models. The last two models are also called (head, modifier, ... modifier) tuples. Unfortunately, representations derived from spaces of such more meaningful and semantically richer features are more difficult to generate automatically and errors in feature extraction can hinder the accuracy of tasks built on top of them. These techniques are also regarded to be overly reliant on term frequency and lack reflection on the semantic relationships between terms. Furthermore, neither the structural nor the semantic information included in the text has been studied in depth. Another disadvantage is that the size of the feature set or the number of dimensions of the vector will be quite enormous, requiring a large amount of storage space as well as consuming computational time.

Several Language models, such as Probabilistic Latent Semantic Indexing [1], Latent Dirichlet Allocation [2], and Word2Vec [3], provide another approach to dealing with synonymy, polysemy, and dimensionality reduction by modeling the document as a vector of hidden topics, instead of a vector of terms. These models assume that several chunks of text with comparable meanings (related to the same topic) are more likely to appear in a similar context and should be represented as vectors with close distance to each other. Topic vectors are substantially shorter than vectors used in traditional models. Topic models are also getting a lot of attention recently because of how simple and efficient they are in describing the features of a text. However, these models do not take into consideration the actual information structure of topics and the semantic links between topics and hence may be limited in their ability to describe complicated topics in complex domains. Furthermore, it is almost impossible for humans to interpret and grasp the theme of the text based on the existing representational structure. The results can be analyzed mathematically, but have no meaning that can be explained in natural language. Good formalisms enable the reader to capture their meaning, comprehend the system's output, and understand how the system computed those results.

For understanding the meaning of a text, semantic or conceptual approaches attempt to incorporate some degree of syntactic and semantic analysis. Semantic approaches emerge as a result of significant advancements in information extraction techniques and the growing use of large-scale general knowledge bases. Several novel forms of vectors for document representation are proposed that aim to capture the semantics of the text in terms of concepts rather than words. The vector's i^{th} component is a weight that indicates the relevance of the knowledge source's i^{th} concept (or entity)

Manuscript received July 26, 2021; revised March 17, 2022.

ThanhThuong T. Huynh is a PhD student and lecturer at the Faculty of Computer Science, University of Information Technology - Ho Chi Minh National University, Ho Chi Minh City, Vietnam (corresponding author to provide phone: +84985-538-775; e-mail: thuonghtt@uit.edu.vn).

TruongAn PhamNguyen is a lecturer at the Faculty of Computer Science, University of Information Technology - Ho Chi Minh National University, Ho Chi Minh City, Vietnam (e-mail: truonganpn@uit.edu.vn).

Nhon V.Do is an Associate Professor and the Dean of Faculty of Engineering ang Technology, Head of Department of Information Technology, Hong Bang International University, Ho Chi Minh City, Vietnam (e-mail: nhondv@hiu.vn).

to the content of the given document. By including the annotated entities into the vector space model, the document representation can be improved, as shown in works [4], [5].

The document in [7] is represented as a set of concepts extracted by an entity linking system, where the concepts are referenced to entities with related descriptive information documented in Wikipedia articles or to entities available in the DBpedia knowledge base. Rather than focusing solely on concepts or entities, and with the assistance of extra semantic resources, the work in [8] treats entities and words equally, i.e. takes into account both types of objects simultaneously. Entity-based representations are bags of entities created from entity annotations, whereas word-based representations are typical bags of words. An entity linker scans the body of text for mentions and connects each one to a relevant entity in the knowledge base.

Many other studies, such as Explicit Semantic Analysis (ESA) [6], leverage information acquired from Wikipedia articles, categories, and wikilinks (or internal links) connecting articles, thanks to which the meaning of the text can be expressed as a vector of weighted Wikipedia concepts. The concept name is the title of the relevant Wikipedia article. The semantic relevance of two documents is determined by the cosine of the angle formed by their representative vectors. In comparison to latent topics of Language models, the meaning of a document expressed by concepts (or entities) of the knowledge base is more natural, and easier to interpret and understand for humans. The length of such vectors, however, is equal to the number of concepts described in the knowledge base, which can be fairly large (possibly up to several million concepts) and computationally expensive. Most of these approaches are still based on ”flat” representations like the vector space model. Despite the fact that such new models still inherit the simplicity and more knowledge-rich than traditional models, they do not take advantage of relational knowledge and network structure encoded within wide-coverage knowledge bases.

Text modeling as graphs has gained popularity in recent years in a variety of fields, including document retrieval, document similarity, text classification, text clustering, text summarization, and so on. Because of well-defined theory foundations and good empirical performance, the graph-based technique has been widely utilized in a variety of text-centric tasks and a wide range of graph models have been proposed. The types of vertices, types of edge relations, external semantic resources, methods for generating structured representations of texts, and weighting schemes all differ significantly between these models. The abundance of available information and methodologies poses the challenge of figuring out how to combine them all and fully exploit the potential of graphs in document representation.

In [9], the text is modeled as a graph, with nodes representing terms that appear in the text and directed edges indicating co-occurrence relationships of terms. Each edge is given weight so that the relationship's strength can be assessed. The position of terms that occur together in the same unit determines the direction of edges. In comparison to a numerical vector, this graph model can retain more structural information in texts, however, the model ignores the meanings of terms and their semantic relationships.

The approach in [10], [11] obtains detailed information about entities and their semantic relationships from the DBpedia knowledge base, resulting in knowledge-rich document models. Nodes in these models are concepts that are related to the content of the document and referenced to entities in DBpedia using existing entity linking tools such as SpotLight or TagMe. Weighted edges connect related concepts based on semantic relationships found in DBpedia. Both works apply a graph model to the document similarity measurement task, but the former also additional demonstrate the benefits of this model through the task of entity ranking. Both works apply a graph model to the document similarity measurement task, but the former [10] also additional demonstrate the benefits of this model through the task of entity ranking. In addition to assigning weights to the edges in the same way as [10], the method in [11] also uses a closeness centrality measure to weight the concept nodes in order to reflect the relevance of these concepts to aspects of the document. It should be noted that these works have ignored the structural information of the text, the relationships between the nodes are not extracted from the given document itself, i.e. independent of the document structure.

The main challenges in graph-based document modeling are the requirement for automated techniques to generate text representations and the computational time complexity for the graph matching task. To capture the semantic similarity between texts, finding the maximum common subgraph (subgraph isomorphism) between two corresponding graphs is a huge challenge. Because graph matching could be performed in non-polynomial time, this makes it impracticable for big datasets.

Motivated by the previous work, this paper deals with the task of document similarity evaluation, where a more expressive way to represent the texts is proposed. Graph-based semantic models for representing document content take into account the incorporation of structural (syntactic) information in text and semantic information derived from various existing knowledge bases to improve task performance. Domain-specific or generic knowledge has been exploited to obtain fine-grained information about concepts and their semantic relations, thereby resulting in knowledge-rich document models.

This work's main contribution can be summarized as follows:

- Several keyphrase graph-based models are proposed to facilitate a more thorough understanding of the document's content along with a method for generating representations of the text. The task of graph-based keyphrase extraction, which is inherently involved as a crucial phase in the process of document representation is also presented. The weighting of graph elements such as keyphrase nodes and relational edges is influenced by a variety of factors and it is an essential treatment that contributes to reflecting how useful these components are as document descriptor features.
- We describe a method for determining the fundamental semantic similarity between two keyphrases, which can subsequently be utilized to construct other various similarity measures between sophisticated structured document representations.
- Finally, to estimate inter-document similarity, we introduce a graph matching technique based on the pairwise

similarity between keyphrase vertices and relational edges of the graphs.

We demonstrate how well a keyphrase graph-based document representation model is designed to help with estimating the degree of semantic similarity between two documents. Figure I provides an overview of our entire approach. Given two input texts, we first construct labeled keyphrase graphs with fully weighted nodes and edges (techniques detailed in section II-C) and then compute semantic document similarity based on the graph's features. The problem of document similarity measurement is formulated as a graph matching problem (as in section III). Since a keyphrase graph is made up of keyphrase nodes and relation edges, the graph similarity is naturally calculated by the pairwise similarity between the keyphrases and between the relationships. This approach not only outperforms baseline document similarity assessment methods, but it also competes with state-of-the-art methodologies for this problem, according to our experiments as in section IV. In addition, our keyphrase graph-based document models, as well as the graph construction techniques and, in particular those for estimating keyphrase similarity and graph matching, can be integrated as fundamental components in various document retrieval frameworks.

II. Document Representation based on Keyphrase Graphs

One of the focuses of the work is text representation, with the goal of transforming text documents into a structured format that computer programs may use to process the text's primary content. The problem is to find a suitable document representation capable of representing semantic information among keyphrases as well as the structural information of the text. Representing documents in a semantic approach requires going through a complex processing process in text semantic analysis. The first is the document analysis phase to extract basic information units from the document and represent the document by those information units. Information units can be words, or more complex phrases, concepts, and document content can be represented by a simple structure such as a set of words(or phrases) with weights or a more semantically rich form of a graph.

Graph-based approaches were chosen for the following main reasons: (i) they are universal in nature and can be utilized with any graph-like knowledge resource, whatever its specific vocabulary; (ii) they have demonstrated to be effective for language comprehension tasks. Understanding the content of a document requires an understanding of the key concepts and entities in the document, as well as how they relate to each other, and most of all, a graph is a mathematical structure capable of effectively modeling relationships along with important structural information. In recent years, the graph-based text representation model has been increasingly noticed and used individually in different problems of text mining fields such as Clustering and Classification, Information Extraction and Retrieval, Text Summarization, Topic Detection. The results of applying the graph models on English documents show that this kind of model has a lot of potential because it takes advantage of important information about the structure and semantic relationships that are not considered in the traditional models. Many graph models have been proposed such as semantic networks, Concept Graphs - CGs, improved CGs, Star Graphs, Frequency Graphs, Distance Graphs, Co-occurrence Graphs, etc. evaluated as having a lot of potential for use, having a clear and strong theoretical background and good experimental performance.

Each discipline builds these models in different ways due to different research objectives and means of use. Representation models and techniques may vary in: vertex types, relational edge types, semantic resources used, ways to create structured representations of text, weighting scheme for vertices and edges of the graph, as well as how to solve subproblems from extracting features as vertices, determining relationships between features, graph matching, and ranking result. The richness and diversity of existing information sources, techniques, and models present a new challenge in the research community: analyzing the applicability of existing models and techniques, thereby finding ways to apply, coordinate, improve and develop in order to exploit the full application potential of the graph approach and enhance the efficiency of solving the problems posed.

Our method is based on descriptive information encoded within the backend knowledge base. Using DBpedia as an underlying knowledge base, we describe a technique for generating structured representations of textual content, similar to [10], [11].

A. Labeled Keyphrase Graph

A Labeled Keyphrase Graph is a directed, finite, multigraph. The term "Multigraph" refers to the fact that a pair of nodes can be connected by several edges. Each node is a keyphrase, a significant, topical phrase chosen from the document's body. Multiple different directed edges can link a pair of keyphrase nodes. Keyphrases offer a concise overview of the content, and may thus be utilized as features in further processing.

The meaning of a document is the result of a reader's interpretation and understanding. This task requires far more information than the exact data included in such a document. In other words, the document structure alone cannot give sufficient information to comprehend the text's content. It is necessary to give the text an additional informative dimension, thereby allowing us to grasp the key concepts or entities mentioned in a document under consideration. As a result, each keyphrase node should be attached with labels to indicate that it may refer to well-defined concepts or unique entities contained in the knowledge base.

We choose DBpedia in this study because it has a substantial quantity of machine-readable knowledge, such as entities, classes, and fine-grained explicit semantic relationships at both the ontology and instance levels. Such a method, however, may be used to any other lexical or ontological resource, as long as it can be seen as a graph containing disambiguated entities or concepts and explicit semantic relationships. A labeled keyphrase graph is built on top of a DBpedia vocabulary and is subject to such vocabulary's specific constraints. We first provide a formal definition for a DBpedia vocabulary.

Definition 1. (DBpedia vocabulary) [12]

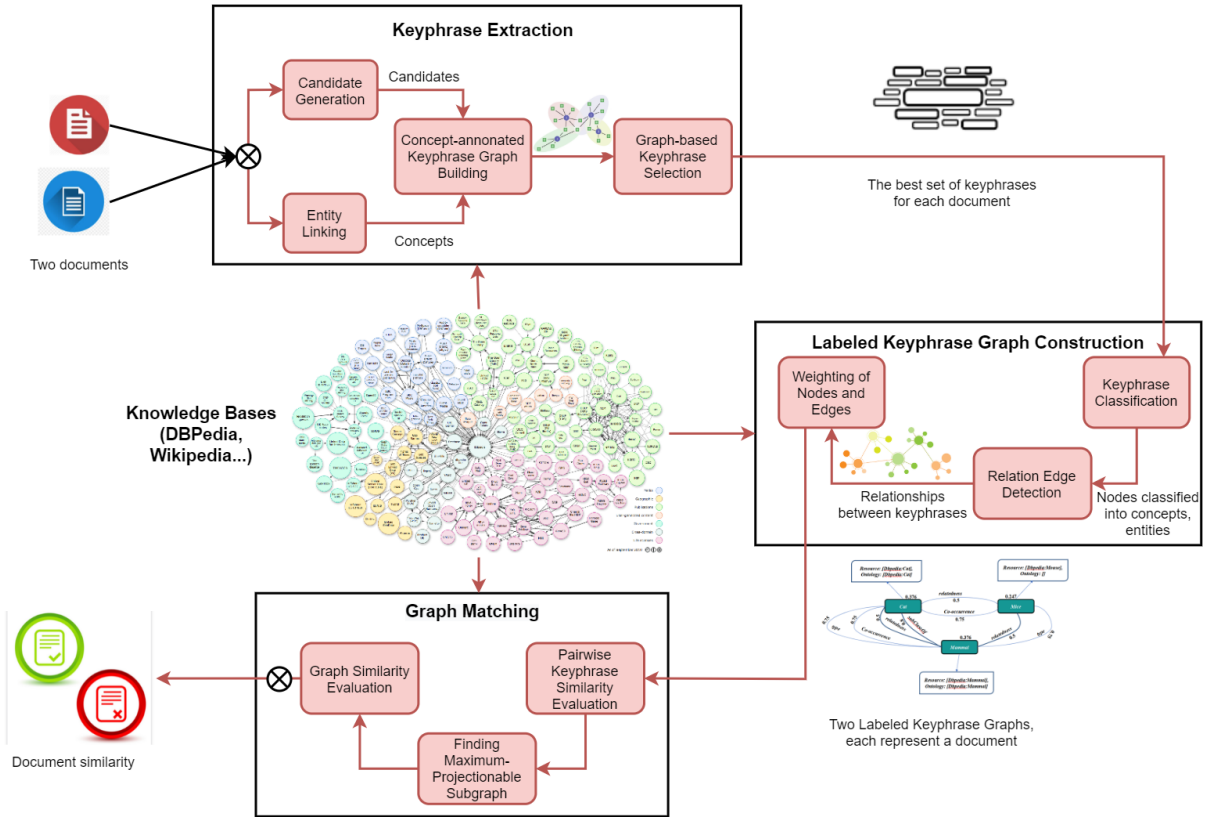

Fig. 1. Document similarity measurement workflow

A DBpedia vocabulary is a triple (T_O, T_R, R_{CC})satisfying the following conditions:

- T_O and T_R are finite, disjoint sets.
- $T_O = T_C \cup T_E$ is the full set of DBpedia's concepts T_C(also called classes) and entities T_E.
- T_R is the set of relation names (called relation symbols) found in DBpedia.
- $R_{CC} \subseteq T_O \times T_R \times T_O$ is the set of semantic relations between concepts and/or entities found in DBpedia.

The two sets T_O and T_R are said to be discrete because they must have no common element. $T_O = T_C \cup T_E$, where T_C is the full set of concepts included in The DBpedia ontology. The DBpedia ontology is a shallow cross-domain ontology developed as a result of a successful crowd-sourcing effort. There are currently 768 classes in ontology, which constitute a subsumption hierarchy and are defined by 3000 distinct attributes. Each concept $c \in T_C$ has a unique URI in the form http://dbpedia.org/ontology/Name. T_E is the set of DBpedia's entities. The DBpedia data set describes 6.0 million entities classified in a consistent ontology, including several classes such as person, place, work, music album, film, video game, organization, species, disease, other. Entities are given URIs that follow the pattern http://dbpedia.org/resource/Name, where Name is derived from the URL of the source Wikipedia article, which takes the form http://en.wikipedia.org/wiki/Name.

The DBpedia vocabulary may be regarded as a directed edge-labeled graph, with nodes representing entities or concepts and edges reflecting explicit semantic relationships between them; equivalently, T_O is a set of nodes, T_R is a set of edge labels, and $R_{CC} \subseteq T_O \times T_R \times T_O$ is a set of edges.

Understanding the content of a text entails not just identifying the significant keyphrases that appear in the document, but also determining the semantic links between them. As a result, modeling these relationships is required and a graph is a suitable tool for this task. Then, each edge reflects the relationship between the two keyphrases and is also labeled with the name of the relation.

The weighted keyphrase graph enables the depiction of semantic and structural links between keyphrases as well as the measurement of such keyphrase's importance in addition to the relationship strength which is unachievable with poor representative, traditional representation models. Each keyphrase in a given document can be assigned a weight, which represents an assessment of how useful it is as a document descriptor. We may wish to label each edge of a graph with a number that measures an important aspect of the edge. Similarly, each relation edge is also weighted (typically but not always statistical) that indicates the degree of membership or relevance between two corresponding keyphrases.

Definition 2. (Labeled Keyphrase Graph) [12]

Given a document d, a labeled keyphrase graph, which represents the document d (denoted as $labdocKG(d)$), defined over a DBpedia vocabulary $O = (T_O, T_R, R_{CC})$, is a tuple $G = (V, E, \phi, l_V, l_E, w_V, w_E)$ satisfying the following conditions:

- (V, E, ϕ) is a finite, directed multigraph called the underlying graph of G, denoted $graph(G)$. V is the non-empty set of keyphrases mentioned within the document, called keyphrase node set. E is a relation edge set.

$\phi : E \rightarrow \{(x,y)|(x,y) \in V^2, x \neq y\}$ an incidence function mapping every edge to an ordered pair of distinct nodes. The edge represents a semantic (conceptual) or syntactic relationship between its two adjacent nodes.
The two keyphrase nodes $k1, k2 \in V$ are connected if there exists a relationship $r \in T_R$ such that $(k1, r, k2) \in R_{CC}$. In addition to semantic relationships, the two nodes can also be connected if there exist some forms of syntactic relationship between them such as co-occurrence or grammatical relationships.

- $l_V : V \rightarrow \wp(T_O)$ and $l_E : E \rightarrow T_R \cup T_S$ are two labeling functions of the nodes and edges of $graph(G)$. Each node can be assigned by multiple labels and is therefore called a multi-label node. Every edge $e \in E$ is labeled with a relation name $l_E(e) \in T_R \cup T_S$. T_S is a set of names of syntactic relations used for labeling edges such as co-occurrence relation or other grammatical relations. $\wp(T_O)$ is the power set of set T_O.
- $w_V : V \rightarrow [0,1]$ and $w_E : E \rightarrow [0,1]$ are two mappings which describe the weighting of the nodes and edges of $graph(G)$. Nodes and edges are thus assigned weights so as to capture their different levels of specificity.

In several disciplines, graphs are widely employed to encode structural information, and graph matching is a challenging task. Subgraph matching, also known as subgraph isomorphism, is the issue of matching a graph to a portion of another graph. As a result, we are also attentive to subgraphs of a labeled keyphrase graph that are labeled keyphrase graphs themselves.

Definition 3. (Sub labeled keyphrase graph)
Let $G = (V, E, \phi, l_V, l_E, w_V, w_E)$ be a labeled keyphrase graph. A sub labeled keyphrase graph (sublabKG) of G is a labeled keyphrase graph $G' = (V', E', \phi', l_{V'}, l_{E'}, w'_V, w'_E)$ (also denoted as $G' \leq G$) such that: $V' \subseteq V$; $E' \subseteq E$; ϕ', l'_V, l'_E are the restrictions of ϕ, l_V, l_E to V', E' respectively; $\phi'(E') \subseteq V' \times V'$; and the weights of every nodes and edges of G' are equal to their counterparts in the super keyphrase graph G.

B. Keyphrase extraction

This section describes an approach that aims to automatically select from the body of the given document keyphrases which is relevant and cover the main content of the document. The objective of keyphrase extraction is to find a collection of excellent keyphrases that are both relevant to the primary themes mentioned in the document and cover those topics. This section discusses how WikiRank [13] might be used in the automated keyphrase extraction process. This extraction pipeline consists of three major steps: candidate generation, entity linking, and graph-based keyphrase selection. The first two steps have commonly been utilized to address keyphrase extraction (KE) in previous KE research.

The candidate generation task aims to generate a set of potential candidate phrases that are likely to be selected as document-specific keyphrases. Based on observations of human-generated keyphrases from several standard datasets for the keyphrase extraction task, keyphrases often take the form of noun phrases that appear in the document and consists of one or more noun combined with zero or more adjectives. As a result, the default pattern for extracting such noun phrases can be described as "a list of nouns" that may or may not be preceded by "a list of adjectives" with no additional Part-of-speech, using the following regular expressions: $< JJ|JJR|JJS > * < NN|NNS|NNP|NNPS > +$.

An NLP toolkit chunker is employed to partially parse sentences. A chunker would normally be initialized with the desired pattern. It then extracts words in sentences, assigns POS tags, and partially processes the sentence grammar to find phrases that resemble the input pattern. An example of this chunker in action can be seen in the paragraph below, candidates that matched the pattern were highlighted:

"*Mad cow disease* has killed 10,000 *cattle*, restricted the *export market* for *Britain's cattle industry* and raised *fears* about the *safety* of eating *beef*. The *government* insists that the *disease* poses only a *remote risk* to *human health*, but *scientists* still aren't certain what causes the disease or how it is transmitted."

In the next step, if each noun phrase identified in the above manner exists in the dictionary generated from Wikipedia page titles, that noun phrase will be considered a candidate. The second step in the automatic keyphrase extraction process is to associate the text with a backend knowledge base to gain a better understanding of the objects mentioned in the text. This task starts with identifying meaningful substrings in a document, called mentions, and associating each mention with related entities contained in the knowledge base. We conducted several experiments before deciding on an entity linking tool that would be appropriate for a keyphrase extraction solution.

Some notable entity association systems considered during testing are TagMe, DBpedia Spotlight, Illinois Wikifier, WAT Api, SWAT API, etc. In comparison to the other tools, our experimental findings demonstrate that integrating TagMe in the solution generates the best keyphrase extraction performance. Using TagMe, this phase produces a list of entities (also known as concepts), each of which comprises the mention and a link to a Wikipedia page that provides additional information for the meaning of the mention.

Despite the fact that TagMe provides an enormous number of annotations, it also offers various optional parameters to aid in filtering the results. TagMe, in particular, assigns a confidence score for each annotation it returns, keeping in mind that this score does not represent the entity's relevance to the input text. Confidence scores can be used to filter out annotations that fall below a certain threshold. On the other hand, other tools may be more accurate but there is no way to compensate for their insufficient number of annotations. Our method depends on such annotations to further screen candidates, it is more convenient to have redundant annotations.

A concept-annotated keyphrase graph that represents the document d generated using a list of candidates K extracted by the candidate generation task and a list of concepts C detected by the entity linking task. We then use this graph to select potential candidates that have many connections with important concepts in the document.

Definition 4. (Concept-annotated keyphrase graph)
A concept-annotated keyphrase graph , which represents the document d (denoted as $aKG(d)$), is a tuple $G =$

(V_k, V_c, E, w) satisfying the following conditions:

- (V_k, V_c, E) is a finite,bipartite, undirected graph.
- $V_k \subseteq K$ is a non-empty set of candidate nodes, $V_c \subseteq C$ is a set of concept nodes
- The node set of the graph is $V = V_k \cup V_c, V_k \cap V_c \neq \emptyset$
- E is a set of undirected edges. Nodes of the bipartite graph are divided into two non-empty, disjoint sets V_k and V_c , with two different kinds of nodes. Following that, each edge connects one node from V_k and one node from V_c.
- Given a candidate $k \in V_k$ and a concept $c \in V_c$, there exists an edge from vertex k to vertex c iff the concept name or a mention of c can be found in k, in other words, it is the substring contained in k. Mention of a concept is a word or phrase that appears in the body of the document and is annotated to that concept by the entity linking task.
- $w : V_k \cup V_c \rightarrow \mathbb{R}^+$ is a weighting function for the graph's nodes. Such weights indicate an assessment of the candidates' and concepts' effectiveness as a descriptor of the document in terms of distinguishing it from other documents in the collection.

The weight associate with the concept node $c \in V_c$ of the graph reflects the importance of the concept within a given document according to the frequency of mentions of c in the whole document. However,c is a concept that is annotated from the target knowledge base (e.g. Wikipedia), so there are many cases that such concept name does not appear fully in the document. Therefore, the weight of the concept c is calculated through all the mentions of this concept, i.e. through the sum of the times all the annotated mentions to the same concept c occur in the given document.

For example, consider the document from DUC-2001 Dataset LA030889-0163 with the content as follows:

> *"Canadian Coach Charlie Francis, who claimed that sprinter Ben Johnson's urine sample at the Seoul Olympics was spiked with a banned steroid, told an acquaintance in Seoul that Johnson had worried that he might test positive. Lynda Huey, who was at Seoul working for NBC-TV and as a physical therapist for some American athletes, said Tuesday that Francis had bragged to her about Johnson's preparations for a showdown against U.S. sprinter Carl Lewis. Huey said she had known Francis since 1980 when he and sprinter Angella Taylor Issajenko stayed at her home in Los Angeles. Huey said she had seen Francis on a practice track at Seoul and he had greeted her as an old friend. "Charlie came over to me and we started talking," Huey said. "We were talking about how Ben might do. Charlie said, 'Ben's more afraid of failing the drug test than he is of Carl Lewis.' He was bragging." Huey said she is tired of hearing Francis, who has been in Toronto testifying at a Canadian inquiry into drug use in sport, say that Johnson was clean at the Olympics. Francis testified that Johnson was not taking the steroid, stanozolol, before the Games and that a mysterious person might have slipped something into Johnson's beverage in the drug-testing area before the sprinter gave his urine specimen. Clean or not, Huey said, "Francis must have had some reason to think Ben may not pass the test." JULIE CART".*

A part of the concept-annotated keyphrase graph of the above sample document is shown in Figure II-B (The rectangles are called candidate keyphrases, and the circles are called annotated concepts) and concepts annotated along with their weights are shown in Table I

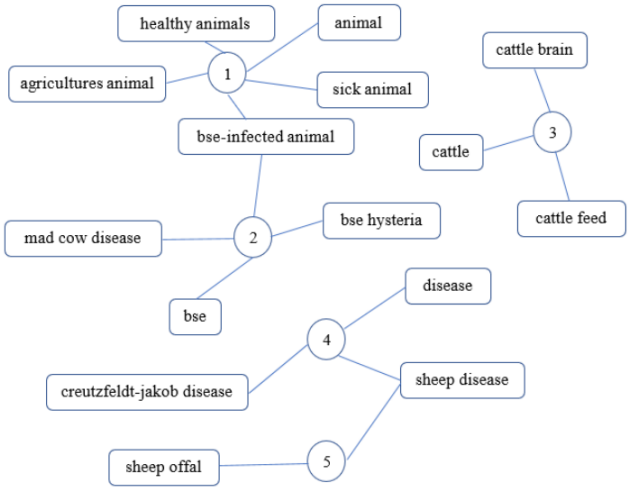

Fig. 2. An excerpt of concept-annotated keyphrase graph corressponding to above document

TABLE I
CONCEPTS ANNOTATED ALONG WITH THEIR WEIGHTS CORRESPONDING TO FIGURE II-B

ID	Concept name	Frequency
1	Animal	6
2	bovine spongiform encephalopathy	14
3	cattle	9
4	disease	16
5	sheep	5

Given the concept-annotated keyphrase graph $G = (V_k, V_c, E, w)$ that represents the document d and constructed in the previous phase. Let n be the desired number of keyphrases. We adjusted the technique provided in [13], as shown in Algorithm 1, to choose a list of prioritized keyphrases. The following are the primary steps involved in the key selection procedure:

The first is to weigh each candidate based on the set of concepts adjacent to that candidate in the graph. Let $W_p(p)$ be the weight of candidate p and defined as: $W_p(p) = \sum_{c \in PC} W_c(c)$, where $PC = \{c \in V_c | \ c \ adjacent \ to \ p\}$. The step of weighting each candidate can be quite time-consuming, especially for long documents that often have thousands of candidates. We can reduce the processing cost by pruning the initial graph before performing the calculation by applying some heuristics as follows: The candidate whose $tf \times idf$ weight is too low can be eliminated from the graph; We can also remove candidates that are not even associated to any concepts or ones are only connected to a single concept with extremely low weight.

The next step is to identify the candidate with the highest weight, add it to the result set, and delete the selected keyphrase from the list of candidates. In the third step, the weight of each c concept, (denoted by $W_c(c)$) adjacent to the selected keyphrase is reduced by a ratio of α. In particular, $W_c(c) = W_c(c)/\alpha$, with α is a parameter that can be chosen experimentally. The value of such rate determines

Algorithm 1 Generate a list of keyphrases for the text

Input :
$G = (V_k, V_c, E, w)$ is the concept-annotated keyphrase graph that represents the given document d, n: The number of desired keyphrases
Output: A list of ranked keyphrases Q
Algorithm

```
Q ← ∅
P ← V_k /* P is a list of candidates */
C ← V_c /* C is a list of concepts */
while |Q| < n do
    /* Calculate the weight of candidate
       based on the set of concepts
       adjacent to that candidate in the
       graph */
    ∀p ∈ P , W_p(p) ← 0 /* W_p(p) is the weight of
       candidate p */
    foreach p ∈ P do
        PC = {c ∈ C| c adjacent to p}
        W_p(p) = Σ_{c∈PC} W_c(c)
    end
    q = FindMaxWeight(P) /* Find the candidate
       in P with maximum weight W_p */
    Q = Q ∪ {q} /* Add q to the result set */
    P = P \ {q}  /* Remove the selected
       keyphrase from the set of
       candidates */
    /* Reduce the weight of concepts
       adjacent to the selected keyphrase
       */
    foreach c ∈ C do
        if c is adjacent to q then
            W_c(c) = W_c(c)/α /* α is a parameter
               that can be selected through
               experiment */
        end
    end
end
return 0 /* There is no more keyphrase to
   visit */
```

how much weight reduction of a concept after one of its adjacent candidates is extracted. This step is necessary to avoid extracting multiple candidates that are adjacent to the same concept. A group of near-synonym candidates is frequently associated with a concept, i.e. candidates which refer to the same concept are more likely to be different names for that concept or one candidate is a substring of the other one, it is sufficient to extract only one of the candidates. The process will be repeated until all n keyphrases have been selected or until there are no more candidates to visit.

All the different parameters of the solution such as the suitable entity linking tool, the threshold value of $tf \times idf$ for pruning the candidates, and the ratio of concept weight reduction after each selection in the aforementioned method should all be taken into account and the optimal set of parameters must be chosen for the best performance. A few experiments are conducted on the DUC-2001 dataset [15] which is widely used for the keyphrase extraction task. As a result, TagMe is used to annotate the text and our experiments with various values of the confidence score on the dataset reveal that a score of 0.11 yields the best keyphrase extraction results. There is no one-size-fits-all value for the $tf \times idf$ cut-off threshold. Instead, we create a lookup table based on the number of candidates, e.g. documents with less than 150 keyphrases should be treated with a threshold of 0.26, whereas documents with 150 to 200 candidates should be cut-off at 0.82. Finally, a one-third reduction in weight (i.e., values of the parameter $alpha = 1.5$) allows for improved task performance.

C. Graphical representation of documents

The main idea behind using a labeled key graph to represent a document is to associate keyphrases in the document with entities or concepts in the knowledge base, and then to explore the structural information in the text and semantic information between key phrases in the knowledge base, from which the document can be interpreted. The task of creating a labeled keyphrase graph of the text will be carried out in three primary steps: key classification, relation edge detection, and weighting of the graph's vertices and edges, given a set of keyphrases identified in the previous stage.

Keyphrase node is a keyphrase that appears in the body of the text and may correspond to concepts or entities defined in the knowledge base. As a result, the nodes are divided into two types: concepts and entities. A concept node can refer to a class in DBpedia ontology, whereas an entity node relates to a DBpedia entity. If any class in the DBpedia ontology has a name that includes the given keyphrase, or if the class name is a substring of the keyphrase, this keyphrase is classified as a concept (class). The keyphrase "power," for example, can denote the concept"Power Station," which has the URI http://dbpedia.org/ontology/PowerStation. The DBpedia SPARQL endpoint may also be used to query the whole DBpedia Ontology. On the other hand, using TagMe, you can associate each keyphrase with a set of referent Wikipedia entities. Through the ID of the Wikipedia page obtained from TagMe, we can retrieve a unique corresponding entity in DBpedia via dbo:wikiPageID and SPARQL endpoint. For instance, the keyphrase "air defense radar systems" contains two mentions, which are "air defense" and "radar". Both of these mentions have been annotated on two related Wikipedia pages with the IDs of such Wikipedia pages being 146640 and 25676 respectively. Then, using a simple query, we can derive the respective entity names "Anti-aircraft warfare" and "Radar". Then we can use a simple query to get the corresponding entity name "Anti-aircraft warfare" and "Radar".

Consider the following two scenarios for determining if two keyphrase nodes can be connected by edges:

- Two keyphrases can be connected by an edge if there is a relation between them specified on the DBpedia vocabulary, such edge obtained from DBpedia will be labeled by the name of the relation.
- If two keyphrases appear together in the same sentence, then several syntactic parsing techniques are applied to determine the type of syntactic relationship between the two keyphrases.
  In case the possible syntactic relationship between the two keyphrases cannot be identified, the co-occurrence

relationship is recorded. Edges constructed in this manner are called edges obtained from the document structure.

There are many various sorts of relationships that can exist between two keyphrase nodes and therefore many different types of edges. However, we will first focus on only five different sorts of connections in this research, including:

- The Hyponymy relationship between two concepts can be derived from hierarchical structures in the DBpedia ontology (denoted "subClassOf")
- The Relatedness relationship between two entities is obtained based on the calculation of relevance according to the formula Eq. 2 and returns a value greater than a certain threshold (denoted "relatedness_entity")
- The rdf:type property in DBpedia helps to detect the type relationship between a concept and an entity (denoted by "entity_type_concept")
- There exists a synonym relationship between two keyphrases when they are annotated to the same entity by an entity linking tool (called "alt_name")
- There will be a Co-occurrence Relationship between two keyphrases that appear together in the same sentence in the document.

We give the keyphrase nodes in the graph weights to represent how important they are to the document's meaning. The three weighting methods considered for use include closeness centrality, $tf \times idf$ scheme, and standard PageRank.

With this in mind, the primary themes of the document can be reflected by a group of semantically closely related keyphrases. As a result, the closeness centrality of each keyphrase node can inform us how significant the keyphrase is in describing the core content of the document. To determine the weight of each keyphrase node, we calculate its closeness centrality over the graph, which is based on the 'closeness property' of such node to all other ones in the graph. Closeness centrality is a method of discovering nodes that can efficiently disseminate information throughout a graph. The average farness (inverse distance) of a node to all other nodes is measured by its closeness centrality. The highest-scoring node has the shortest distances between itself and all other nodes. A node's importance in a graph increases as it gets closer to the center. Therefore, based on finding the shortest paths between all pairs of nodes, the algorithm calculates the sum of each node's shortest distances to all other nodes. The resulting sum is then inverted to establish the node's closeness centrality score.

Furthermore, to evaluate the effectiveness of keyphrases in distinguishing a document from other documents in the same collection, there are several weighting methods assuming that: the best descriptor of the document may include keyphrases that appear frequently in the document but are rarely present in other documents. Among those techniques, the weighting scheme $tf \times idf$ is one of the most popular.

A node can be associated with a specific entity, which has its own Wikipedia article, as a result, the node weight can be calculated based on the PageRank of the corresponding Wikipedia article. In other words, many entities are described by Wikipedia articles and cross-references appearing in Wikipedia pages allow a directed graph structure to be established. This structure is perfectly suited when used to calculate PageRank scores, which are considered important weights for entities.

Weights are also assigned to directed edges in the graph to reflect the strength of the relationship as well as capture the degree of semantic relevance between keyphrases, which in turn also contributes to the specification of the topics of the document. For example, for a co-occurrence relationship, the weight assigned to this relationship is usually calculated based on the following assumption: Within a document, the higher the frequency of co-occurrence of two keyphrases in the same sentence, the stronger the connection between them. However, the frequency of each individual keyphrase occurring in the text might be quite low in some types of documents, thus two keyphrases seldom appear together in a sentence more than once. The weight allocated to an edge may thus be defined as the frequency of co-occurrence in a sentence of both of its adjacent nodes throughout the whole collection of documents.

As another example, the weight of a semantic relation is a measure of the semantic similarity between keyphrases directly linked by this relation. The weight value can be determined manually or through some experiment and be subject to some additional constraints, such as Keyphrases connected by a synonymy relationship will have a higher degree of semantic similarity than keyphrases linked just by a hierarchical relationship, and non-hierarchical relationships are given less weight than hierarchical relationships.

After continuously completing the aforementioned processes, an entire labeled keyphrase graph structure associated with the provided text can be generated by employing selected feature keyphrases, along with their labels, directed relation edges between pairs of keyphrases, and the full weights of vertices and edges. For example, consider the document #20 from LP50 dataset with extracted keyphrases whose referent concepts/entities and corresponding weightes are shown in Table II: "The *United Nations* was determined that its *showpiece environment summit* - the biggest conference the world has ever witnessed - should be staged in *Africa*. The venue, however, could not be further removed from the grim realities of life in the rest of Africa. Johannesburg's exclusive and formerly whites-only suburb of *Sandton* is the *wealthiest neighbourhood* in the continent. Just a few kilometres from Sandton begins the *sprawling Alexandra township*, where nearly a million people live in squalor. Organisers of the conference, which begins today, seem determined that the two worlds should be kept as far apart as possible. Tight security surrounds Sandton's convention centre and *five-star hotels*, where world leaders will debate poverty, the *environment* and sustainable development while enjoying lavish hospitality."

III. Graph-based document similarity evaluation

A. Semantic similarity between two keyphrases

Documents are usually considered semantically close when they are described by similar concepts or by closely related concepts. As a result, one of the core tasks for the challenge of document similarity measurement is assessing the semantic similarity between two concepts. This work takes advantage of DBpedia's Knowledge Graph and adopts a method from [16], termed wpath, to measure semantic similarity between concepts. This method exploits the structural

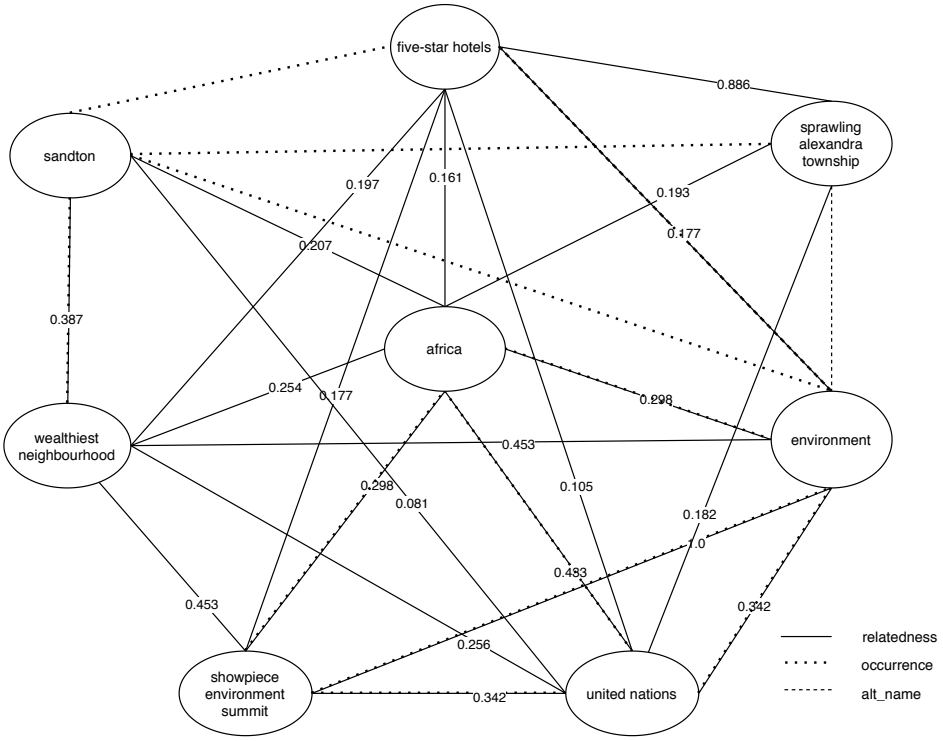

Fig. 3. An excerpt of keyphrase graph representing the document #20

TABLE II
LIST OF KEYPHRASES OF THE DOCUMENT #20 AND THEIR REFERENT CONCEPTS/ENTITIES

Keyphrase	Concepts and entities	Weights: centrality, tf, idf and pagerank score
united nations	http://dbpedia.org/resource/United_Nations	1.78, 1, 4.64, 0.10
showpiece environment summit	http://dbpedia.org/resource/Natural_environment	1.92, 1, 5.64, 0.16
africa	http://dbpedia.org/resource/Africa	1.78, 2, 4.06, 0.10
sandton	http://dbpedia.org/resource/Sandton	2.02, 3, 5.64, 0.19
sprawling alexandra township	http://dbpedia.org/resource/Alexandra,_Gauteng, http://dbpedia.org/ontology/Ship, 'http://dbpedia.org/ontology/Town	1.75, 1, 5.64, 0.08
five-star hotels	http://dbpedia.org/resource/Hotel_rating, http://dbpedia.org/resource/Hotel, http://dbpedia.org/ontology/Star, http://dbpedia.org/ontology/Hotel	1.78, 1, 5.64, 0.08
environment	http://dbpedia.org/resource/Natural_environment	2.10, 2, 3.64, 0.23
wealthiest neighbourhood	http://dbpedia.org/resource/Wealth, http://dbpedia.org/resource/Neighbourhood	1.61, 1, 5.64, 0.05

information of the semantic network such as the shortest path length between concepts in the graph, combined with the leverage of the Information Content (IC) of the concept to evaluate the strength of the path.

Definition 5. (Pairwise concept similarity) The semantic similarity between two concepts c_i and c_j (also known as Pairwise concept similarity) is defined over a DBpedia vocabulary $O = (T_O, T_R, R_{CC})$ by the following formula:

$$sim_{wpath}(c_i, c_j) = \frac{1}{1 + length(c_i, c_j) * k^{IC(c_{lcs})}} \quad (1)$$

where $length(c_i, c_j)$ signifies the length of the shortest path between two concepts c_i and c_j in O. The shorter the path between two concepts, the closer they are semantical. General concepts are more likely to appear in a given context and are therefore assumed to provide less information than specialized concepts. A concept with a higher probability of occurrence is less surprising and less informative. The less likely a concept is to appear, the more surprising and informative it is. As a result, the amount of information conveyed by a concept in a particular context is referred to as its information content.

Information Content of a concept c, denoted by $IC(c)$, measures the informativeness of a concept as the inverse of its probability of occurrence and is therefore defined as follows: $IC(c) = -\log Prob(c)$ where $Prob(c) = \frac{|entities(c)|}{N}$, N is the total number of entities in O, and $entities(c)$ is the set of entities with the type c.

We shall, however, calculate the information content of the most recent common ancestor of the two concepts rather than the information content of each concept separately. The symbol c_{lcs} represents the Least Common Subsumer (LCS) of concepts c_i and c_j found in the given DBpedia taxonomy of concepts (classes)(derived from O thanks to the is-a relation), i.e. c_{lcs} is the most specific concept which is an

ancestor of both. When two concepts share a more specific concept (i.e. higher the IC of their LCS), it indicates that they share more information in common and therefore more similar. The IC of the LCS indicates the amount of common information content shared by the two concepts and has a certain degree of contribution or influence to the semantic similarity between the two concepts. The parameter $k \in (0,1]$ denotes the contribution of the LCS's IC and experimenting yielded the most efficient value of 0.8 for k.

Definition 6. (Pairwise entity similarity) Given two entities e_i and e_j, let $In(e_i)$ and $In(e_j)$ indicate the two sets of incoming links to each Wikipedia page of e_i and e_j respectively, the semantic relatedness between two entities e_i and e_j (also called pairwise entity similarity) is given as follows:

$$rel(e_i,e_j) = 1 - \frac{\log(max(|In(e_i)|,|In(e_j)|)) - \log(|In(e_i) \cap In(e_j)|)}{\log(|W|) - \log(min(|In(e_i)|,|In(e_j)|))} \quad (2)$$

where W is the set of all entities included in the knowledge base.

The semantic relatedness of two entities in the knowledge base reveals how semantically related they are to one another. We utilize a basic metric based on the amount of shared incoming links to quantify the pairwise entity similarity of such entities, as shown in the above formula Eq 2, which is similar in spirit to [10]. Shared incoming links are English Wikipedia pages that link to both certain entities. The number of incoming links indicates the prominence or popularity of an article in Wikipedia in relation to an entity. The greater the number of shared incoming links, the closer the two entities are to sharing the same context. Therefore, the relevance of the two entities will increase as more of their common links are added.

According to the above-mentioned method of creating a Labeled Keyphrase Graph that represents the content of a document, each keyphrase node can be associated with a set of concepts along with a set of entities derived from a back-end knowledge base. As a result, assessing the similarity between two related concept sets, as well as the similarity between the two associated entity sets, can be used to determine the semantic similarity between two given keyphrases.

Definition 7. (Groupwise concept similarity)

Given two keyphrases k_1 and k_2, let $C_1 = \{c_{11}, c_{12}, ..., c_{1p}\}$ and $C_2 = \{c_{21}, c_{22}, ..., c_{2q}\}$ be two sets of concepts that referred to k_1 and k_2, respectively. The "conceptual" similarity between two keyphrases k_1 and k_2 is determined through the similarity of two groups of concepts associated with k_1 and k_2 (also known as Groupwise concept similarity), and is calculated according to the following formula:

$$gcs(k_1,k_2) = \max\{sim_{wpath}(c_{1i},c_{2j}) | c_{1i} \in C_1, c_{2j} \in C_2\} \quad (3)$$

By calculating the semantic similarity of each possible pair of concepts (c_{1i}, c_{2j}), derived from C_1 and C_2,i.e. $c_{1i} \in C_1$ and $c_{2j} \in C_2$, the similarity of the two groups of concepts is determined by the maximum value of such pairwise concept similarities. In the same way, the groupwise entity similarity would be determined.

Definition 8. (Groupwise entity similarity)

Given two keyphrase k_1 and k_2, let $E_1 = \{e_{11}, e_{12}, ..., e_{1p}\}$ and $E_2 = \{e_{21}, e_{22}, ..., e_{2q}\}$ are the set of entities that are associated to k_1 and k_2, respectively. The "entity" similarity between two keyphrases k_1 and k_2 is defined as the similarity of two groups of entities associated with those two keyphrases (also known as Groupwise entity similarity) given by the following formula:

$$gec(k_1,k_2) = \max\{rel(e_{1i},e_{2j}) | e_{1i} \in E_1, e_{2j} \in E_2\} \quad (4)$$

The maximum value of the groupwise entity similarity and the groupwise concept similarity results in the semantic similarity between two keyphrases.

Definition 9. (Pairwise keyphrase similarity)

Given two keyphrases k_1 and k_2, the semantic similarity between two keyphrases k_1 and k_2, (called Pairwise keyphrase similarity) is defined as:

$$sim(k_1,k_2) = \max\{gcs(k_1,k_2), gec(k_1,k_2)\} \quad (5)$$

The algorithm 2 calculate this similarity value.

Algorithm 2 Calculate the semantic similarity between two keyphrases

Input :
C_1: The set of concepts linked to keyphrase K_1
C_2: The set of concepts linked to keyphrase K_2
E_1: The set of entities linked to keyphrase K_1
E_2: The set of entities linked to keyphrase K_2
Output: The similarity value between K_1 and K_2

```
Algorithm sim ← 0 foreach i ∈ C1 do
    foreach j ∈ C2 do
        s ← similarity(i, j)   /* According to
            formula 1                          */
        if s > sim then
            sim ← s
        end
    end
end
foreach i ∈ E1 do
    foreach j ∈ E2 do
        s ← similarity(i, j)   /* According to
            formula 2                          */
        if s > sim then
            sim ← s
        end
    end
end
return sim
```

B. Semantic similarity between two keyphrase graphs

A fundamental concept when using KG for the task of document similarity measurement is homomorphism, also known as projection. A KG projection is a mapping between two keyphrase graphs that preserves the structure of KG. More specifically, a projection from the keyphrase graph H to the keyphrase graph G is built based on the mappings going from the set of vertices and the set of edges of H to the set of vertices and the set of edges of G where allowing maps adjacent vertices in H to adjacent vertices in G. Following

that, several essential definitions for graph matching and calculating the similarity of two graphs are required.

Definition 10. (KG projection) [12]
Let $G = (V_G, E_G, \phi_G, l_{V_G}, l_{E_G}, w_{V_G}, w_{E_G})$ and $H = (V_H, E_H, \phi_H, l_{V_H}, l_{E_H}, w_{V_H}, w_{E_H})$ are two labeled keyphrase graphs. A KG projection from G to H consists of an ordered pair $\Pi = (f,h)$ of two mappings $f : V_G \to V_H$, $h : E_G \to E_H$, that satisfy the following conditions:

- f and h are injective functions
- The projection preserves the relationships between nodes of G, i.e. for all $e \in E_G, f(adj_i(e)) = adj_i(h(e))$, $adj_i(e)$ denotes the i^{th} node adjacent to edge e.
- $\forall k \in V_G, sim(k, f(k)) \neq 0$, where $sim(k, f(k)) \in [0,1]$ is the semantic similarity between such two keyphrases

The definition of the (total) KG projection provides a useful means through which we can assess the relevance between two pieces of text modeled by the keyphrase graphs. However, some texts can be considered related even if only part of them are similar. In other words, if two documents are represented by two keyphrase graphs and the two corresponding graphs are still said to "match" if there exist one or more pieces of content in the first text that are related in meaning (ie refer to the same topic or related topics) with the content in the other text. Thus, graphs are said to have a match when they contain the same vertices or vertices in one graph are semantically related to some vertices in the other graph. Therefore, it is more feasible to find a projection from only part of one keyphrase graph to another keyphrase graph. We call this a partial KG projection.

Definition 11. There is a partial KG projection from a labeled keyphrase graph H to a labeled keyphrase graph G iff there exists a KG projection from H', a sub labeled keyphrase graph of H (H' $\leq$ H), to G.

Definition 12. (Maximum-projectionable subgraph)
Given G and H are two labeled keyphrase graphs. A labeled keyphrase graph g is a maximum-projectionable subgraph of G and H, denoted as mps(G, H), if the following conditions are satisfied:

- $g \leq G$
- There exists a KG projection $\Pi = (f,h)$ from g to H, which is also a partial KG projection from G to H.
- There is no other sub labeled keyphrase graph g', which satisfies two first conditions, such that $|V_{g'}| > |V_g|$.

Definition 13. (Similarity between two keyphrase graphs)
Let $G = (V_G, E_G, \phi_G, l_{V_G}, l_{E_G}, w_{V_G}, w_{E_G})$ and $H = (V_H, E_H, \phi_H, l_{V_H}, l_{E_H}, w_{V_H}, w_{E_H})$ are two labeled keyphrase graphs representing documents. The semantic similarity between graphs G and H is defined as follows:

$$Sim(G,H) = \beta \times \frac{\sum_{k}^{V_g} sim(k, f(k)) \times w_V(k, f(k))}{max(|V_G|,|V_H|)} + (1-\beta) \times \frac{\sum_{e}^{E_g} w_E(e, h(e))}{max(|E_G|,|E_H|)} \quad (6)$$

where $w_V(k, f(k)) = \frac{min(w_{V_G}(k), w_{V_H}(f(k)))}{max(w_{V_G}(k), w_{V_H}(f(k)))}$, $w_E(e, h(e)) = \frac{min(w_{E_G}(e), w_{E_H}(h(e)))}{max(w_{E_G}(e), w_{E_H}(h(e)))}$. The similarity between two relation edges in two keyphrase graphs is simply defined as a ratio between the weights of the lower-valued edge and the higher-valued edge. This definition allows the similarity between two edges to be calculated regardless of the difference in the type of the relationship between the two edges.

The symbol $g = mps(G,H)$ signifies the maximum-projectionable subgraph of G and H. There may be numerous KG projections from g to H, however, the attached projection is the one with the maximum value of $\sum_k^{V_g} sim(k, f(k)) \times w_V(k, f(k))$. It is possible to have a total KG projection between two documents' graphs even if the two documents are not completely related. The valuation of this projection should not be the maximum. However, there may not be any total projections between the two graphs even though they are related, and then it is necessary to consider the partial projections between them. The parameter $\beta \in (0,1)$ in formula 6 is a user-defined artificial coefficient that indicates the percentage contribution of the vertex matching in the comparison with the edge matching to the final graph matching result.

Given two labeled keyphrase graphs G and H as inputs, finding a maximum-projectionable subgraph is a computational task aimed at determining whether G contains some projectionable subgraph such that there exists a KG projection from this subgraph to a corresponding subgraph of H and then derive the largest of all the found compatible subgraphs of G. Extracting a maximum-projectionable subgraph from two graphs is a difficult task. The bruce force approach is to first generate all possible partial KG projections for the two given graphs, and then find the projection with the largest subgraph of G. A major challenge has arisen since computing in this way leads to an NP-complete problem. To overcome this difficulty, we do not aim to find the exact solution of the *mps* search problem as well as prove the correctness of the solution mathematically. Instead, we just leverage the heuristic approach to graph matching and estimate the semantic similarity of the two graphs, as shown in AlgorithmIII-B.

The process of evaluating the semantic similarity between two graphs can be explained by the following main processing steps:

- First, for each edge e in G, we find the best projection from e to H. Let $h(e)$ be an edge of H corresponding to e under this projection, $Sim(e, h(e))$ reaches the maximum value compared to other possible projections. Next, store e and $h(e)$ in the two result graphs which are subgraphs of G and H respectively. The endpoints of the edge e are added a queue Q.
- The second phase involves removing a vertex p from the queue Q and performing vertex expansion, which entails finding all vertices adjacent to p in G, as well as all vertices adjacent to $f(p)$ in H(attached to the corresponding adjacent edges).
  Given two sets of edges acquired after the aforementioned expansion stage as input, a Weighted Bipartite Matching algorithm is used to select a set of pairs of edges from the input that is determined to be the best match in terms of semantic similarity between two relation edges. Two vertices of each pair belong to two different graphs. The ratio between the weights of the lower-valued edge and the higher-valued edge is simply

Algorithm 3 Evaluate the semantic similarity between two keyphrase graphs

input : Two keyphrase graph H and G
output: The the similarity between G and H

```
result ← 0
foreach edge E in G do
    let E′ be the most suitable projection of E in H
    f (E) ← E′
    h (E.source) ← E′.source
    h (E.destination) ← E′.destination
    Q ← Empty Queue
    Q.enQueue (E.source)
    Q.enQueue (E.destination)
    while Q is not Empty do
        kg ← Q.deQueue ()
        kh ← h (kg)
        foreach e where e is adjacent edge to kg and f (e)
         is null do
            let e′ be the most suitable projection of e that is
              connected to kh
            f (e) ← e′
            h (e.destination) ← e′.destination
            Q.enQueue (e.destination)
        end
    end
    result = max (result, Sim(G,H))
end
return result
```

calculated to determine how similar the two edges are. The match between these two edge sets must ensure the maximum number of matched edge pairs in the result set. This necessitates us giving up the pair of edges with the highest similarity in order to choose the pair with the lower similarity and allow another pair of edges to be included in the result. In terms of overall interest, this trade-off can get the maximum score.
The endpoints of the edges of G from the previous step's set of pairs are further added to Q and the selected edges are added to the result graphs. If Q is empty but there are still vertices in G that are not visited, then enqueue these vertices to Q in turn. This expansion will be repeated until Q is empty and all vertices of G have been considered. When no more expansion can be made, the maximum projection has been found.

- In the third step, the formula Eq.6 is used to calculate the semantic similarity between G and H. Besides the similarity value, the output also has a result graph $mps(G,H)$ which is a temporary maximum-projectionable subgraph of G.
- Finally, the whole process is repeated, starting from the first step with another edge of G, resulting in a different value of semantic similarity between two graphs, which is compared to the currently stored value, then the bigger value is selected and the storage is updated for the next comparison.

IV. Experiments

Extensive experiments are required not only to evaluate the efficacy of the proposed method but also to determine the appropriate parameters for the underlying formulas and algorithms. The keyphrase extraction task is the subject of our first series of experiments. We can only begin experimenting to compare our approach against other document similarity measurement strategies when the best suitable keyphrase extraction method has been established. We will delve into the process of finding optimal parameters in the next section. This section will report the evaluation of our approach on the LP50 dataset.

A. Experiment settings

The LP50 dataset is a classic for the task of document similarity measurement. There are 50 news articles in this collection derived from the Australian Broadcasting Corporation (ABC), resulting in 1225 pairs of documents. These texts range in length from 51 to 126 words and cover a broad variety of topics. The similarity between two documents in each of the 1225 pairs was graded on a scale of 1 to 5 by 8 to 12 different assessors. The ultimate score for each pair was obtained by averaging all of the judged scores. The agreement between system outputs and human assessments is measured using the Pearson linear correlation coefficient. This statistic provides a single percentage value that reflects the overall performance across the entire dataset.

We have a significant number of published benchmark results for comparison due to the sheer popularity of the LP50 dataset in the research community. The following are several notable state-of-the-art methods for this task: Concepts Learned [19], Explicit Semantic Analysis (ESA) [6], Concept graph similarity (ConceptGraphSim) [10], Context Semantic Analysis (CSA) [18], and graph model (GED) [11].

Aside from the above methods that were tested individually with the LP50, some authors had also investigated linear combinations of findings from different methods such as WikiWalk and ESA [17], CSA combined with Latent Semantic Analysis (LSA, and CSA combined with ESA. Inspired by these studies, we performed both including testing our method separately, and besides also trying some linear combinations of the original method with reimplementation of ESA. Our method's best results were obtained, utilizing the configuration in Table III.

Last but not least, we also test some of the latest Deep learning Language models like *Roberta* [20] and *MPNet* [21]. These models represent the state-of-the-art Deep learning approach to build robust language models that required a large corpus but very little human annotation to train. These pre-trained models can then be used to produce text embedding for a variety of NLP tasks. Though none of these models was trained on the document similarity measuring task, they are comparable with our method's long-term goal - that is - to create a robust document representation that can be easily adapted to use in document retrieval systems.

To adapt these two models for the inter-document similarity problem, we use Transformers library to produce sentence embedding vectors for each document in LP50 datasets. Then compute the pairwise cosine similarity between those vectors to determine document similarity.

B. Benchmark result and discussion

All results are shown in the Table IV. For a long time, Concepts Learned has been the dominating champion, though

TABLE III
PARAMETERS AND THEIRS OPTIMAL VALUES

Parameter	Value
Candidate pattern (section II-B)	$<JJ\|JJR\|JJS>*<NN\|NNS\|NNP\|NNPS>+$
Entity Linker (section II-B)	TagMe with confident score 0.11
weight reduction rate α (section II-B)	1.5
Keyphrase node weight (section II-C)	closeness centrality+tf+idf*pagerank (gives the best results compared to many other combinations)
Types of edge (section II-C)	Co-occurr, relatedness_entity, subClass, entity_type_concept, alt_name
β in Equation 6	0.71

TABLE IV
RESULTS ON THE LP50 DATASET (PEARSON R CORRELATION COEFFICIENT)

Method	Pearson
CSA [18]	0.62
GED [11]	0.63
MPNet [21]	0.63
CSA + LSA [18]	0.65
Roberta [20]	0.65
ESA paper [6]	0.72
CSA + ESA [18]	0.72
ConceptGraphSim [10]	0.74
WikiWalk + ESA [17]	0.77
ConceptGraphSim + ESA [10]	0.78
KGM (+TagMe)	0.79
KGM (+LE-TagMe)	0.80
Concepts Learned [19]	0.80
KGM (+LE-TagMe) + ESA reimplement	0.81

its lead over is just approximately one or two percent and it is likely overfitted for the LP50 dataset.

Deep transfer-learning approaches like Roberta and MPNet also achieved decent performance, beating CSA and GED alone but were in turn beaten by some combination of CSA, ESA, and other concept-based approaches.

Our method, named Keyphrase Graph Matching (abbreviated KGM), on the other hand, can achieve performance comparable to the most advanced state-of-the-art methods, as well as the same high level of task performance as the leading method, "Learned Concepts".

Besides, by combining our approach with the ESA reimplementation, we can get ahead of the "Concepts Learned" by a little percentage in performance, but that is too small of an improvement to be noticeable, therefore we will not invest too much effort in re-implementing other methods in combination with our technique in a similar manner.

When looking more closely at the example where our method scores the lowest, we notice a recurring pattern where the human accessor would judge the similarity of two documents very much differently from the computer. Given two paragraphs in the LP50 dataset:

First paragraph:

> *Washington has sharply rebuked Russia over bombings of Georgian villages, warning the raids violated Georgian sovereignty and could worsen tensions between Moscow and Tbilisi. "The United States regrets the loss of life and deplores the violation of Georgia's sovereignty," White House spokesman Ari Fleischer said. Mr Fleischer said US Secretary of State Colin Powell had delivered the same message to his Russian counterpart but that the stern language did not reflect a sign of souring relations between Moscow and Washington*

Second paragraph:

> *A U.S-British air raid in southern Iraq left eight civilians dead and nine wounded, the Iraqi military said Sunday. The military told the official Iraqi News Agency that the warplanes bombed areas in Basra province, 330 miles south of Baghdad. The U.S. Central Command in Florida said coalition aircraft used precision-guided weapons to strike two air defense radar systems near Basra "in response to recent Iraqi hostile acts against coalition aircraft monitoring the Southern No-Fly Zone.*

Our system would give these paragraphs a 1.6/5 similarity score because they mention two different wars, in two different time frames with completely different combatants. However, both paragraphs have a similar topic of collateral damage in a war between the world's superpowers against a smaller country. And this similar theme carries a much larger score of 3.9 from human accessors of the LP50 dataset. Now, not only that this underlying theme is hard to extract by computer, but it is also very subjective to evaluate the degree of similarity, as the next example will show:

First paragraph:

> *Hunan province remained on high alert last night as thunderstorms threatened to exacerbate the flood crisis, now entering its fifth day and with 108 already dead and hundreds of thousands evacuated. On the flood frontline at Dongting Lake, the water level peaked at just under 35m on Saturday night, then eased about 3cm during the day under a hot sun, with temperatures reaching 35C. But with the lake still brimming at dangerously high levels, and spilling over the top of its banks in some places, locals were fearful that a thunderstorm and high winds forecast to hit the region last night would damage the dikes. About 1800km of dikes around the lake are all that stand between 10 million people in the surrounding farmland and disaster.*

Second paragraph:

> *The river Elbe surged to an all-time record high Friday, flooding more districts of the historic city of Dresden as authorities scrambled to evacuate tens of thousands of residents in the worst flooding to hit central Europe in memory. In the Czech Republic, authorities were counting the cost of the massive flooding as people returned to the homes and the Vlava river receded, revealing the full extent of the damage to lives and landmarks.*

Our system is surely at the error when it gave these examples a score of 1 out of 5. It over penalized the difference in types of disaster (thunderstorm vs natural flood) and location while failing to notice that the similar impact of flooding caused by those natural events on human life is a major talking point of both paragraphs. However, the human assessors' generosity in giving these two paragraphs a whopping 4.6 similarity score is also a debatable decision. Such a high score would indicate the two paragraphs were discussing the same event, which they clearly did not.

These two examples served to illustrate both the difficult and subjective nature of the document similarity evaluation problem. Therefore, it is worth reverberating that our main

goal in this research is not revolutionizing the document similarity measurement task. The major objective is to contribute to the diversification of approaches to this problem so that the novel proposed method is good enough for assessing document similarity while remaining flexible enough to be easily adaptable for a variety of other related tasks in the field of document retrieval.

V. Effect of Parameter Selection Strategies

A. Parameters in the keyphrase extraction experiment

As we mentioned briefly in section II-B, the optimal parameters for keyphrase extraction method spawn from exploratory experiments on the DUC-2001 dataset [15]. This dataset is among the first benchmark for a keyphrase extraction method with 308 news articles on 30 different topics with an average length of 700 words per document. Each document was manually labeled with at most 10 keyphrases by two different annotators, resulting in 2488 keyphrases for the entire corpus.

The system's extraction result for each document can be evaluated using the prototypical metrics: precision, recall, and F-score. In our experiment, we use the average F-score of 308 documents as the sole performance indicator for each combination of parameters' values. In the preceding sections, we looked at the reasoning behind these parameters and how they might affect overall performance.

We first start the experiment with a baseline configuration for all the parameters. The best feasible value for each parameter is then discovered by holding the other parameters constant while testing with a wide range of alternative values for the parameter under consideration. We keep the value that brought the best F-score then repeat that process for the next parameters.

The first to be examined was the keyphrase candidate pattern. The keyphrase extracting process mainly uses patterns of noun phrases which are mostly combinations of adjectives and nouns. In addition, we also consider incorporating various types of words (parts of speech) to form a variety of potential noun phrase patterns such as nouns, verbs, adjectives, adverbs, conjunctions, and even prepositions. Adjectives, in particular, are mentioned with all three different degrees – positive degree, comparative degree, superlative degree of adjectives, while verbs also include present and past participle forms. The second parameter is the choice of entity linking tool and the corresponding confidence threshold. Since some tools might be more suitable for certain domains and application contexts than others, it is beneficial to experiment with different tools for not only pure performance metrics but for consistency as well. The third parameter is the $tf \times idf$ (term frequency and inverse document frequency) cut-off value. One of the heuristics employed to improve task performance is that candidates with too low $tf \times idf$-values can be eliminated before the graph-based keyphrase selection phase starts operating. The final parameter in algorithm 1 indicates "how much should we reduce the weight of concept after we extract one of its adjacent keyphrase candidates".

The final configuration will then be compared against other State of the art keyphrase extraction methods. The result shown in Table VI cemented the effectiveness of our keyphrase extraction process.

TABLE V
Parameters and theirs tested values in the keyphrase extraction task

Parameter	Tested Values
Candidate pattern	$<JJ>*<NN.*>$, $<NN.*><RB><NN.*>$, $(<JJ><NN.*>)+$, $<JJ>*<NN.*><CC>?<NN.*>$, $<VBG><NN.*>+$, $<NN.*><IN><NN.*>+$, $<VBG><NN.*>+$, $<NN.*><IN>?<NN.*>$, $<JJ>?<CD><NN.*>+$, $<JJ><NN.*><IN><NN.*><CC><NN.*>$, $<VBN><JJ><NN.*>+$, $(<NN.*>+)<VBG>$, $<JJ>*<NN.*><IN><VBG><JJ>$, $<CD><NN.*>+$, $(<JJ>*<NN.*>+<IN>)?<JJ>*<NN.*>+$, $(JJ\|JJR\|JJS)*(NN\|NNS\|NNP\|NNPS)+$, $(JJ\|JJR\|JJS\|VBG\|VBN)*$ $(NN\|NNS\|NNP\|NNPS\|VBG)+$
Entity Linking System	SingleRank, Topical PageRank, WikiRank, Our system (+ Spotlight), Our system (+ TagMe with the different confident scores in [0,1])
$tf \times idf$ cut-off threshold	(0,1]
Weight reduction rate α	[1, 2]

TABLE VI
Results of Keyphrase Extraction task on the DUC-2001 dataset

Method	F score
WikiRank paper [13]	27.53
WikiRank reimplement	20.93
Pattern(adj+noun) [14]	27.7
Chunk(+participle) [14]	27.9
Our system (+ Spotlight)	23.62
Our system (+ TagMe)	28.44

Of those parameters, the entity linker tool and the $tf \times idf$ threshold value were the hardest to do experiments since we have too many combinations to choose from. Instead, we tried to find the best linker first and figure out the $tf \times idf$ cut-off value later. To find a good linker, we compared their raw entity taggers output with the golden dataset, treating all entities/candidates as if they are keyphrases. This comparison resulted in very low precision and hence low F-score so we would focus more on recall instead. Our little experiment shows that TagMe with a confidence of 0.11 provides the best recall and F-score.

The $tf \times idf$ cut-off value was even more tricky to choose. Since most entity taggers sort their output by the value of $tf \times idf$ score. It may seem trivial to choose a cut-off value that would result in a more manageable amount of candidates. However, the scatter plot in Figure 4 clearly shows a messy and chaotic relationship between the cut-off value and the number of candidates extracted.

Each dot represents a single document in DUC2001, the y-coordinate shows the best $tf \times idf$ threshold to extract keyphrase on such document and the x-coordinate shows

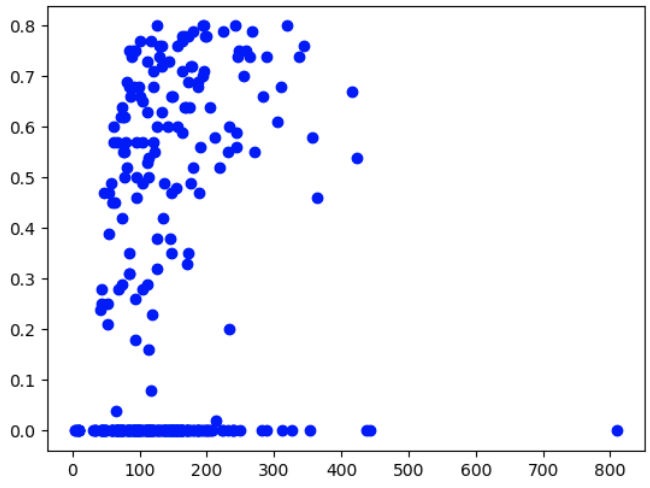

Fig. 4. Optimal TF*IDF threshold for each document in DUC2001

the corresponding number of extracted candidates. Through observation, we discover that it would work best to set different cut-off values for documents based on the number of candidates detected in those documents. For example, some guideline specifically for our dataset is that documents with less than 50 candidates may have the cut-off value of 0.26, documents with between 100 and 150 candidates should have the cut-off value of 0.24, etc...

With all those hard parameters chosen, we chose the last parameter by repeating our experiment and changing this parameter by a small variance in each step. We were able to find a local optimum of F-score if the weight of the concept is reduced by one and a half times after one of the candidate vertices adjacent to it in the graph is extracted.

B. Parameters in the document similarity measurement experiment

In the proposed solution of the document similarity measurement task, there are many parameters that need to be tested in order to choose the best set of parameters. Firstly, each document is modeled as a keyphrase graph, where nodes can be connected to each other by some of the five types of relationships such as alt_name, subClassOf, entity_type_concept, co-occurrence, and finally relatedness_entity. Therefore, it is necessary to determine which sets of relations give the best performance. For the relatedness_entity relation, we also need to choose the threshold value to conclude whether two keyphrase vertices are related to each other. Moreover, each relationship type should be assigned a weight that indicates the comparable expressive power of the relationship in the same graph over others. As discussed in section II-C, we investigated three potential methods for assigning weights to the keyphrase nodes in the keyphrase graph, namely $tf \times idf$ scheme, closeness centrality, and standard PageRank. We tried several simple combinations of the three weights. We tried also several different values of the coefficient β in formula 6 for measuring the semantic similarity between graphs.

Since there are so many different parameter values to choose from, if we are to test every combination of the possible values for these parameters, the number of test cases

TABLE VII
PARAMETERS AND THEIRS TESTED VALUES IN THE DOCUMENT SIMILARITY MEASUREMENT TASK

Order	Parameter	Tested Values	Default value
1	The threshold value of the relatedness between two keyphrase nodes	(0,1]	None, this parameter is the first to get tuned
2	Types of edge	All subsets of Co-occurr, relatedness_entity, subClass, entity_type_concept, alt_name	Use all five types of edge
3	Keyphrase node weight	All combinations with two operators * and + of weights such as closeness centrality, tf. idf, pagerank	All node get the weight of value 1 (maximum weight)
4	Relation edge weight	(0,1] assigned value for edge of types: subClass, entity_type_concept and alt_name	alt_name had weight 1, subClass and entity_type_concept weight 0.8
5	β in Equation 6	[0,1]	1

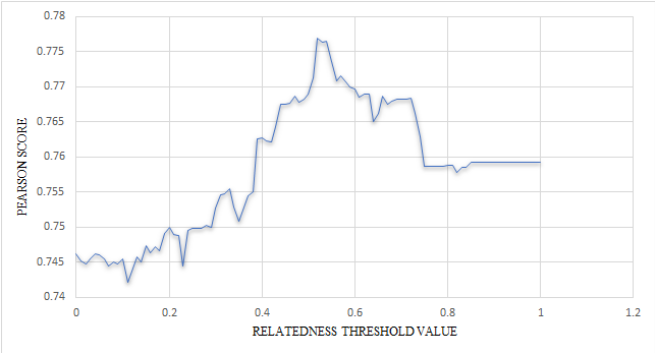

Fig. 5. Pearson score for different related threshold values

is inconceivably large. Therefore, we tried to find locally optimized results by tuning one parameter at a time. We start with arbitrary chosen defaults for all parameters given in Table VII. Then we tried out every possible value for one parameter while keeping the rest rigid, searching for the value that would bring the best Pearson score. After that, we would lock this parameter in the new-founded best value and repeat the experiments for the next parameter and so forth.

The first parameter to be tuned is the keyphrase relatedness threshold value, since this parameter affects the number of edges and the size of the keyphrase graph, we preferred to deal with it sooner rather than later. Starting at the threshold of 0 and an increment step of 0.01, we found the Pearson score peaked at a threshold of 0.52 as shown in Figure 5.

The next parameter would be the types of edges to be included in the graph since this parameter will finalize the graph size. There were 5 types of edge, which meant 31 combinations of types of edge to consider, all presented in Figure 6. One can notice that this parameter makes the Pearson score fluctuate in a quite wide margin. Also, it's to our surprise that the best score was achieved using only

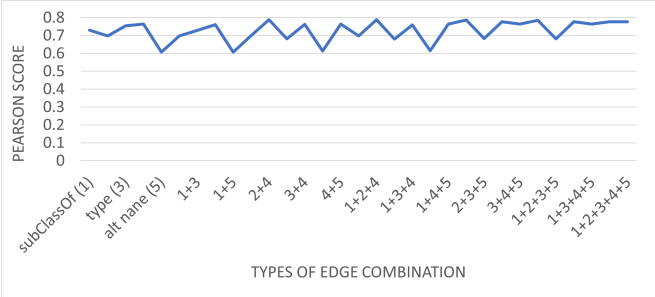

Fig. 6. Pearson score for different combinations of types of edge

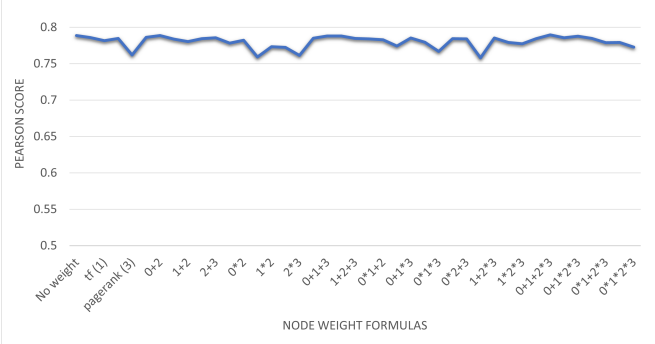

Fig. 7. Pearson score for different formulas of keyphrase node weighting

two types of edge, none of which come from the knowledge base, but rather from the structural information within the document.

Since the best types of edge retain two types of edge, both of which got their own weighting formula, we do not need to tune the weighting strategies for types of edge that we will not use. The next parameter will be a weight combining strategy for each keyphrase node. There were 4 kinds of weight for nodes, and thus there were quite a lot of formulas using different combinations of 4 variables to choose from. However, we did not feel the need to exhaustively test every possible formula under the sky, as one can see in Figure 7: Of all the 41 formulas we did try out, the fluctuation of the final Pearson score is within 0.03, a too-small margin to matter. Even our default choice of no weighting at all (treating all nodes equally with the weight of 1) proved to be more than good enough. Thus we settle upon the best value out of the 41 and move on to the final parameter.

The β parameter in control whether node similarity or edge similarity will have more influence in the similarity measurement between two graphs. We assumed that the node will be much more influential, which turned out to be true but not to a degree we anticipated. Ranging β from 1 (completely disregarded edge similarity) to 0.3 yields a near-identical Pearson score (within 0.05 of each other). The score, however, drops sharply with β lower than 0.3, to the point of unusable. Therefore we chose the locally best β of 0.71 and thus found the optimized parameter shown in Table III.

VI. Conclusion

We presented graph-based document models and a knowledge-rich strategy for producing expressive interpre-

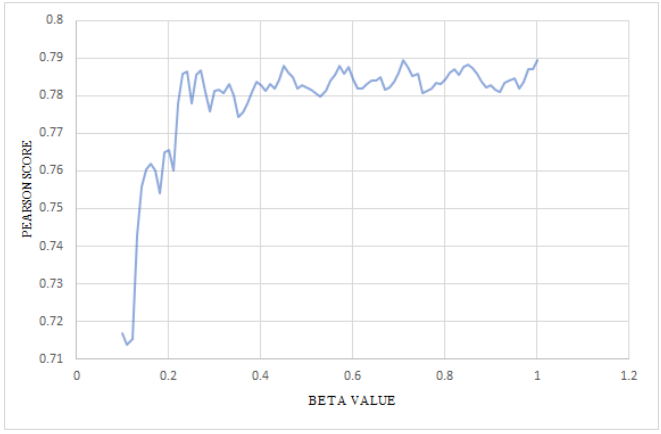

Fig. 8. Pearson score for different values of β

tations of texts in this work. By extracting representative keyphrases from texts and inferring their explicit semantic associations from the enormous volumes of structured human knowledge stored in existing knowledge bases, document graphs can be constructed. Keyphrases are linked to concepts, disambiguated entities in DBpedia and Wikipedia to facilitate the understanding of semantics. Each node and edge of the graph are assigned weights so that the usefulness of keyphrases and the strength of relationships may be assessed.

DBPedia is a large-scale knowledge base that covers a wide range of disciplines and includes numerous detailed, explicit semantic relations between entities and concepts. Our method, however, has just used basic relations like subClassOf, relatedness_entity, entity_type_concept, and alt_name. We intend to examine a combination of several types of connections between keyphrases in future work.

The paper also presented the idea of applying WikiRank to aid with keyphrase extraction. Several candidates who are less likely to be keyphrases can be eliminated with the help of Wikipedia and TagMe, thus reducing noise in the findings. However, the extraction of both a keyphrase and its abbreviation is regarded as an error. Longer, cross-concept candidates appeared to be preferred by the key extraction algorithm, which is a significant issue in some datasets. It might help by adjusting the weights that each candidate contributes to its adjacent concepts, especially when it is linked to multiple concepts. Furthermore, due to the nature of Wikipedia, this method is likely to perform well in the general domain and on news articles, but for the domain-specific corpus, such as a repository of scientific articles belonging to a certain knowledge domain, we will need to exploit other knowledge bases.

The wide range of available weighting schemes and techniques also posed a challenge in terms of figuring out how to combine them and fully exploit the potential of keyphrase graphs for better performance. We believe that by integrating more features and learning their weights, performance can be enhanced even more, but we will leave that for future work.

By extracting the maximum-projectionable subgraph between two related keyphrase graphs, a novel graph matching technique was provided, which can then be used to evaluate the semantic distance between two documents. This technique can also be applied to a variety of textual tasks, includ-

ing document retrieval, document classification, and entity ranking, in addition to measuring document similarity. The algorithms for calculating the similarity between keyphrase graphs might need some efficency improvements, however, when facing larger corpus.

APPENDIX

In this section, we will introduce an Entity Linking Method called Lookup Entity. This method relies on entity names taken from the DBpedia knowledge base to identify entities that are related to the content of the document. The Lookup Entity method allows to identify entities and annotate them to entities contained in DBpedia. A hybrid method that combines Lookup Entity and TagMe is also presented, named LE-TagMe. To begin, certain key concepts in the solution will be described explicitly.

Definition 14. The main entity, in the text, is an entity whose mention cannot be joined with the preceding or following words to produce a new entity. A child entity is one whose mention can be combined with surrounding words to form a new entity, which is then referred to as its parent entity. A child entity can have multiple parent entities. The parent entity is not necessarily the main entity.

Definition 15. The names of different entities might be duplicated, that is, a name can refer to many separate entities, we call entities detected from the same name as unidentified entities. Entities discovered from different names, i.e. they do not share the same name, are called uniquely identified entities.

Consider a sentence “The Hubble Telescope is a space telescope that was launched into low Earth orbit in 1990 and remains in operation”. The entity ”Hubble Space Telescope” is annotated with the mention ”Hubble Telescope.” Because the related mention cannot be merged with words preceding or following it to produce another entity, this entity is considered the main entity. Mention ”Telescope” is annotated to an entity ”Telescope,” which is a child entity because its mention ”Telescope” can be joined with ”Hubble” to generate a new entity, ”Hubble Space Telescope.”

The following is a model of the entity extraction problem: Given a set of entity names derived from the DBpedia knowledge base and a document *d*. The returned result of the problem is the main entity set found in the text d. Each main entity e in the result has information including 1) *Name* is the official name of the entity on DBpedia, 2) *Sent* is the ordinal number of the sentence (considered in the whole body of document) containing the entity, 3) *StartToken* is the index of the start token of the mention in document d referring to entity e, 4) *EndToken* is the index of the end token of the mention in document d referring to entity e, and 5) *ListSub* is a collection of child entities of e.

The main steps in the process of extracting entities using the Lookup Entity method include:

- Step 1: Collect all official names and alternative names of each entity found on the DBpedia knowledge base.
- Step 2: Using the Spacy tool, process token separation, label tokens, determine the position of the token-containing sentence in the text. The output of this processing phase will be used as the starting point for entity identification in the next stage.
- Step 3: Separate all of the entities in the document into nine bags.
- Step 4: Identify the main entities, the child entities from the nine entity bags.

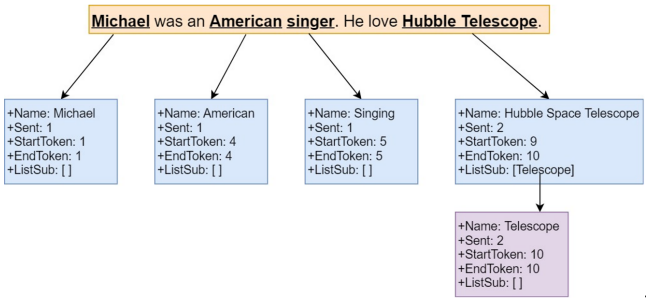

Fig. 9. An example of entity extraction result using Lookup Entity

Each entity can have multiple names. Therefore, to distinguish entities, in addition to using identifiers (abbreviated as IDs), DBpedia also uses another attribute, which is the official name. This type of name is unique, functioning like an ID that helps distinguish entities from each other. An ID is often made up of a string of numbers while an official name often is a combination of alphabetic elements. Names that refer to the same entity, which are not official names, are referred to as alternative names. Barack Obama, for example, is the official name, whereas the collection including Obama, President Obama, Barack Hussein Obama II are the alternate names. UIT stands for Ho Chi Minh City University of Information Technology. ”Vietnam National University, Ho Chi Minh City,” commonly known as VNU-HCM, National University of Ho Chi Minh City, or Ho Chi Minh City National University.

The properties ”is dbo:wikiPageRedirects of” and ”is dbo:wikiPageDisambiguates of” are used by DBpedia to classify entity names. ”is dbo:wikiPageRedirects of” stores a name that is not similar to the name of any other entity, while ”is dbo:wikiPageDisambiguates of” stores ambiguous names that cannot navigate to a specific official name. Because such a name could refer to a variety of different entities, it’s impossible to determine which one the writer is referring to.

Natural language processing tools such as NLTK, Stanford CoreNLP, Spacy, and others are currently available. Spacy was chosen as the document processing tool in this work since it is currently among the most high performance, and there are many additional natural language processing tools being developed around it, which is a favorable factor for future research improvement.

Spacy splits document d into n tokens $T = \{t_1, t_2, \ldots, t_n\}$ (as an example in Figure 10). Each token $t_i \in T$ contains the following information: $\{StartChart_i, EndChart_i, sttToken_i, POS_i, Sent_i, isStopWord_i\}$, where $StartChart_i$ is the position of the starting character of the token t_i in text d, $EndChart_i$ is the position of the end character of tokent_i in text d, $sttToken_i$ indicates the ordinal number of the token in text d, POS_i is the label of t_i (eg PROPN, VERB, ADP, NOUN,...) , $Sent_i$ indicates

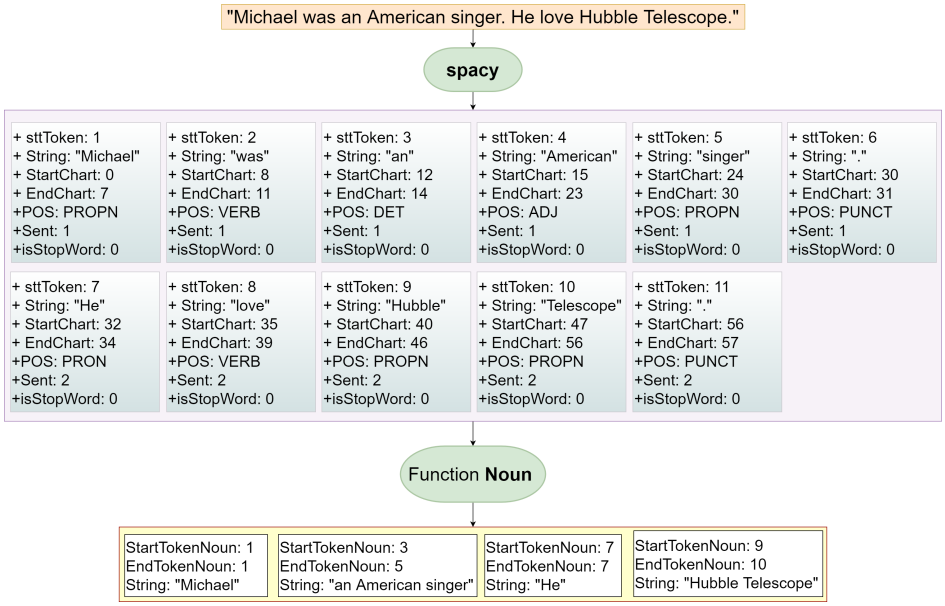

Fig. 10. An example of identifying noun phrases in a piece of a document

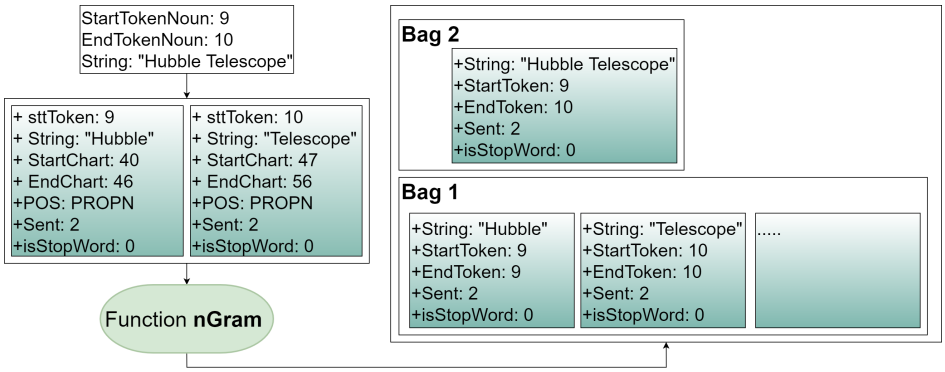

Fig. 11. An example of n-gram splitting of noun phrases

the ordinal number of the sentence containing the token, and finally $isStopWord_i$ indicates whether the token is a stopword (using Spacy's stopword list).

The entity recognition process consists of two phase. The first one is to identify the noun phrases in the document. For simplicity, we consider phrases without verbs, without "," and "." are noun phrases. The next phase is to separate the n-grams of the noun phrases given above and divide them into 9 bags. Splitting n-grams where n is a large value takes a lot of resources. It is easy to see that, most of the names of the entities will be kept short to facilitate communication. Therefore, we choose to separate n-grams with n = 9. The units that make up n-grams will be tokens. These n-grams will be divided into 9 different bags, the i-th bag contains n-grams with n = i, as Figure 11 illustrates. An n-gram that exactly matches the entity name is annotated to the corresponding entity. Here are some notes about the task of entity annotation: If the n-gram exactly matches an alternative name, try to navigate the alternative name to a certain official name. The n-gram is annotated as an "identified entity" if this task is completed successfully. If it is not possible to convert the alternative name to an official name, the n-gram is classified as an "unidentified entity". Additionally, if the n-gram is a stopword in the 1-st bag, it is not labeled as an entity.

A parent entity, as previously said, will contain child entities; in other words, a child entity will undoubtedly dwell in a parent entity. The parent entity will definitely have a longer mention than all mentions of its child entities. To discover the parent entities, we begin by looking for mentions with a large number of tokens. Child entities are those that are contained within these parent entities. An entity without a parent entity is known as the main entity. To be able to determine if an entity is contained within another entity, we employ a dynamic programming technique to determine whether or not an entity's mention was an element of another mention.

The Lookup Entity method merely identifies entities by performing a match based on entity names. Meanwhile, the meaning of a word depends on the context in which it appears. As a result, in addition to the name-related features of the entity, the contextual features of the entity also need to be exploited. TagMe is an entity annotator that exploits additional contextual information. We will combine Lookup Entity and TagMe (as Figure 12) to enrich and improve the performance of this task. In case, for the same mention, TagMe and Lookup Entity may generate two different annotation results. At this time, we do not consider the correctness

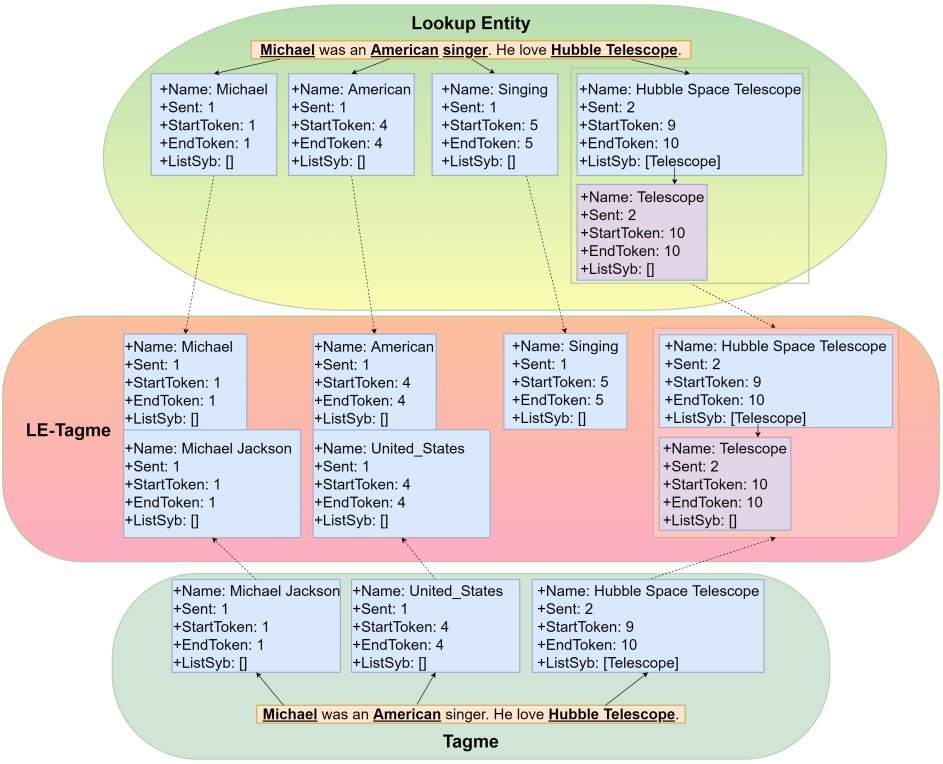

Fig. 12. An example of an entity annotation result using LE-TagMe

of the annotation results because each method has its own advantages. Both results will be kept in order to take benefit of them. The TagMe has an advantage when the annotated entity is already defined in detail on the knowledge base. However, the number of entities is in fact extremely large and increases over time so that one can only define typical entities. Trying to refer to a certain entity is sometimes detrimental because not all entities are available on the knowledge base. Lookup Entity overcomes this drawback because it can return results including unidentified entities.

REFERENCES

[1] Thomas Hofmann, "Probabilistic Latent Semantic Indexing", Proceedings of the Twenty-Second Annual International SIGIR Conference on Research and Development in Information Retrieval (SIGIR-99), 1999.

[2] Blei David M., Ng Andrew Y., Jordan Michael I. Lafferty John. "Latent Dirichlet Allocation". Journal of Machine Learning Research. 3 (4–5): pp. 993–1022. doi:10.1162/jmlr.2003.3.4-5.993.

[3] Mikolov Tomas, et al. "Efficient Estimation of Word Representations in Vector Space", 2013. arXiv:1301.3781

[4] Chenyan Xiong , Jamie Callan , Tie-Yan Liu, "Bag-of-Entities Representation for Ranking", Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval, September 12-16, 2016, Newark, Delaware, USA.

[5] Hadas Raviv, Oren Kurland, and David Carmel, "Document retrieval using entity-based language models", Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2016), ACM, 65–74, 2016.

[6] Gabrilovich, Evgeniy, Markovitch, Shaul,"Computing Semantic Relatedness using Wikipedia-based Explicit Semantic Analysis", IJCAI International Joint Conference on Artificial Intelligence. Vol. 6, 2007.

[7] Faezeh Ensan, Ebrahim Bagheri,"Document Retrieval Model Through Semantic Linking", ACM, WSDM, 2017.

[8] Chenyan Xiong , Jamie Callan , Tie-Yan Liu, "Word-Entity Duet Representations for Document Ranking", Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, August 07-11, 2017, Shinjuku, Tokyo.

[9] Jianging Wu, Zhaoguo Xuan and Donghua Pan,"Enhancing text representation for classification tasks with semantic graph structures", International Journal of Innovative Computing, Information and Control Volume 7, Number 5(B), 2011.

[10] Yuan Ni, Qiong Kai, Xu Feng Cao. "Semantic Documents Relatedness using Concept Graph Representation", WSDM '16 Proceedings of the Ninth ACM International Conference on Web Search and Data Mining, Pages 635-644, ACM, 2016.

[11] Michael Schuhmacher, Simone Paolo Ponzetto, "Knowledge-based graph document modeling", WSDM '14 Proceedings of the 7th ACM international conference on Web search and data mining, Pages 543-552, 2014.

[12] ThanhThuong T. Huynh, TruongAn N.Pham, Nhon V.Do, "Keyphrase Graph in text representation for document similarity measurement", in Proceedings of The 19th International Conference on New Trends in Intelligent Software Methodologies, Tools, and Techniques (SoMeT2020), IOS Press, 2020.

[13] Yang Yu, Vincent Ng, "WikiRank: Improving Keyphrase Extraction Based on Background Knowledge," Information Retrieval (cs.IR), 2018.

[14] Le, T. T. N., Nguyen, M. L., and Shimazu, A. , "Unsupervised Keyphrase Extraction: Introducing New Kinds of Words to Keyphrases", Lecture Notes in Computer Science, 665–671, 2016.

[15] Wan X. and Xiao J., "Single document keyphrase extraction using neighborhood knowledge", In Proceedings of the 23rd National Conference on Artificial Intelligence - Volume 2, AAAI'08, pages 855–860. AAAI Press, 2008.

[16] Ganggao Zhu, and Carlos A. Iglesias, "Computing Semantic Similarity of Concepts in Knowledge Graphs", IEEE Transactions on Knowledge and Data Engineering 29.1,pages 72-85, 2017.

[17] E. Yeh, D. Ramage, C. D. Manning, E. Agirre, and A. Soroa, "Wikiwalk: Random walks on wikipedia for semantic relatedness", In Proceedings of the 2009 Workshop on Graph-based Methods for Natural Language Processing, 2009.

[18] Benedetti, F., Beneventano, D., Bergamaschi, S., and Simonini, G., "Computing inter-document similarity with Context Semantic Analysis", Information Systems, 2018.

[19] L. Huang, D. Milne, E. Frank, and I. H. Witten, "Learning a concept-based document similarity measure", Journal of the Association for Information Science and Technology, 2012.

[20] Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. "Roberta: A robustly optimized bert pretraining approach." arXiv preprint arXiv:1907.11692, 2019

[21] Song, Kaitao, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. "Mpnet: Masked and permuted pre-training for language understanding." arXiv preprint arXiv:2004.09297, 2020.