A Method for Designing Domain-Specific Document Retrieval Systems using Semantic Indexing

ThanhThuong T. Huynh[1]
University of Information Technology
VietNam National University HCMC
Ho Chi Minh city, Viet Nam

TruongAn PhamNguyen[2]
University of Information Technology
VietNam National University HCMC
Ho Chi Minh city, Viet Nam

Nhon V. Do[3]
Ho Chi Minh City Open University
Ho Chi Minh city, Viet Nam

***Abstract*—Using domain knowledge and semantics to conduct effective document retrieval has attracted great attention from researchers in many different communities. Ultilizing that approach, we presents the method for designing domain-specific document retrieval systems, which manages semantic information related to document content and supports semantic processing in search. The proposed method integrates components such as an ontology describing domain knowledge, a database of document repository, semantic representations for documents; and advanced search techniques based on measuring semantic similarity. In this article, a model of domain knowledge for various information retrieval tasks, called The Classed Keyphrase based Ontology (CK-ONTO), will be presented in details. We also present graph-based models for representing documents together measures for evaluating the semantic relevance for usage in searching. The above methodology has been used in designing many real-world applications such as the Job-posting retrieval system. Evaluation with real-world inspired dataset, our methods showed noticeable improvements over traditional retrieval solutions.**

***Keywords*—Document representation; document retrieval system; graph matching; semantic indexing; semantic search; domain ontology**

I. Introduction

A. Indispensible Need for Semantic Document Retrieval System

In this Information Age, the need for better management of digitalized documents in various aspects of daily life is ever more pressing. In education for example, searching for documents in your particular area of interest is an indispensible need of learners. That raises the problem of building a system to manage digitalized document in the domain of interest and support searching based on document content or knowledge. In media and publication, the vast amount of online news published everyday are making it more and more difficult for any entity in charge of managing and dissecting all those news article in their particular domain. Even the internal clerical and administrative work flow of a single organization can produce large amount documents that are in need of better content-based book keeping.

Another challenging document retrieval task can be found in job-posting management. The special nature of job-postings, which are often quite short but packed to the rim with keywords in the domain make the content of those documents very difficult to search.

To provide for those needs, we propose a model to build a class of document retrieval systems that optimize to manage a collection of documents in the same domain. The key challenging for those systems is a high precision semantic based search engine, which would be the focal point of the work discussed in this article. We follow the recent trend of ontology based semantic search as well as graph based document representation, combined in a coherent system.

B. Ontology-based Document Retrieval

Nowadays, many researches attempt to implement some degree of syntactic and semantic analysis to improve the document retrieval performance. In contrast to keyword based systems, the result of semantic document retrieval is a list of documents which may not contain words of the original query but have similar meaning to the query. Therefore, the objects of searching are concepts instead of keywords and the search is based on space of concepts and semantic relationships between them. To analyze the content of queries and documents, one has to consider extracting basic units of information from documents, queries and interpreting them. The main idea behind semantic search solutions is using semantic resources of knowledge to resolve words / phrases ambiguities, thus facilitate the understanding of query and document.

Knowledge representation models as well as knowledge resources play an increasingly importance role in enhancing the intelligence of document retrieval systems, in supporting a variety of semantic applications. Semantic resources include taxonomies, thesauri, and formal ontologies, among which ontologies are getting the most attention. Ontologies have proved to be powerful solutions to represent knowledge, integrate data from different sources, and support information extraction. One of the more common goals in developing ontologies is to share common understanding of the structure of information among people and/or systems. That goal leads to the development of gigantic general knowledge resources like DBPedia [1] or Yago, etc. However, even with the help of those generic knowledge bases, it remains extremely challenging to build a semantic search system that can cope with real world adhoc query. The current trend in Document Retrieval researchs is to focus on retrieval tasks in a very specific domains. The focus allows knowledge bases to be more carefully prepared, and thus both the query and the document can be better interpreted.

Many domains now have standardized ontologies developed for them by communities of domain experts and researchers. Those ontologies are often publicly shared and can

be used in a variety of tasks, some well-known large-scale and up-to-date ontologies are: The MeSH and SNOMED in Medicine, PhySH in Physics, JEL in Economics , AGROVOC and AgriOnt [2] in Agriculture, CSO [3] in Computer Science, MSC in Mathematics, etc. However, often an ontology of the domain is not a goal in itself. Developing an ontology is akin to defining a set of data and their structure for other programs to use. Problem-solving methods, domain-independent applications use ontologies and knowledge bases as data. Sadly, few of those wonderful ontologies were built with the document retrieval task in mind.

The CK-ONTO [4] is an ontology model developed first and foremost for the task of document retrieval in a specific domain. We tried to built a model powerful enough to support various information retrieval tasks, yet lean and efficient enough so that a CK-ONTO knowledge base can be quickly constructed in a new domain. The next section in this article describes the architecture of CK-ONTO in detail and then discusses a sample knowledge base built on the CK-ONTO model.

C. Document Representation

Document representation (DR) plays an important role in many textual applications such as document retrieval, document clustering, document classification, document similarity evaluation, document summarization, that is documents are transformed in form of readable and understandable way by both human and computer. The challenging task is to find the appropriate representation of document as so to be capable of expressing the semantic information of the text.

In statistical approaches, documents are described as pairs (feature, weight). Such models are based on the assumption that documents and user queries can be represented by the set of their features as terms (a simple word or phrase). Additionally, weights or probabilities are assigned to such terms to produce a list of answers ranked according to their relevance to the user query.

Among the first, widespread representations are the Bag Of Words (BoW)and the Vector Space Model (VSM). The document retrieval approaches using these representations primarily based on the exact match of terms in the query and those in the documents, they do not address multiple meanings of same word and synonymy of words [5].

In order to address polysemy, synonymy and dimensionality reduction, researchers have proposed several methods such as Latent Semantic Analysis (also called Latent Semantic Indexing), Probabilistic Topic Models or Latent Topic Models. In topic models, e.g. Probabilistic Latent Semantic Indexing [6], Latent Dirichlet Allocation [7], Word2Vec [8], documents are represented as vectors of latent topics. A latent topic is a probability distribution over terms or a cluster of weighted terms. The length of topic vectors is much smaller than the vectors of traditional models. Such models assume that words which are close in meaning tend to occur in similar pieces of text (contexts). These approaches are also widely used because of their simplicity and usefulness for describing document features, however, some of their drawbacks include: Most of such techniques are largely based on the term frequency information, but lack the reflection of semantics of text, e.g. ignore the connections among terms, structural and semantic (or conceptual) information is not considered; The topic models do not consider the structure of topics and relationships among them and have limitations when representing complex topics; Besides, the representations might be difficult to interpret. The results which can be justified on the mathematical level, but have no interpretable meaning in natural language. The good formalisms should make them easy to understand their meaning and the results given by the system, and also how the system computed the results.

Semantic or conceptual approaches attempt to implement some degree of syntactic and semantic analysis; in other words, they try to reproduce to some degree of understanding of the natural language text. Such researches indicate that semantic information and knowledge-rich approaches can be used effectively for high-end IR and NLP tasks.

Given such problem, many studies have been directed to the designing of more complex and effective features which aim to achieve a representation based on more conceptual features than on words. The multi-word terms or sometimes called phrases can be used as features in document vectors/bags. Some of complex feature models are: Lemmas, N-grams, Nouns Phrases, (head, modifier, ... modifier) tuples which are complex phrases with syntactic relations like subject-verb-object or contain non adjacent words. Such features can be detected via pure statistical models. Unfortunately, such representations are derived automatically, thus the (few) errors in the retrieval process compensate in accuracy provided by the richer feature space.

The rapid growth of information extraction techniquies and popularity of large scale general knowledge bases, thesauri as well as formal domain ontologies brought some new forms of representing vectors. The i-th component of vector is the weight reflecting the relevance of the i-th concept (or entity) of the knowledge resource in the represented document. For instance, Explicit Semantic Analysis (ESA) [9] uses Wikipedia articles, categories, and relations between articles to capture semantics in terms of concepts. ESA expresses the meaning of text as a vector of Wikipedia concepts. Each Wikipedia concept corresponds to an article whose title is concept name. The length of vector is the number of concepts defined in Wikipedia (a few millions). Semantic relatedness of documents is measured by cosin of the angle between their vectors. Document representation can be enriched by adding the annotated entities in to the vector space model [10], [11]. In [12], a document is modeled as bag of concepts provided by entity linking systems, in which concepts correspond to entities in the DBpedia knowledge base or related Wikipedia articles. Instead of centering around concepts or entities and using an additional resource, the work in [13] treats entities equally with words. Both word based and entity based representations are used in ad-hoc document retrieval. Word based representations of query and document are standard bags of words. Entity based representations of query and document are bags of entities constructed from entity annotations. An entity linking system finds the entity mentions in a text and links each mention to a corresponding entity in the knowledge base.

The meaning of a document as expressed through knowledge base concepts (or entities) is easier for human interpre-

tation as opposed to topics of latent topic models. However, the length of vectors equals the number of concepts in the knowledge base, which could be very large. Most of these approaches relies on "flat" meaning representations like vector space models, more sophisticate but still do not exploit the relational knowledge and network structure encoded within wide-coverage knowledge bases.

In recent years, modeling text as graphs are also gathering attraction in many fields such as document retrieval, document similarity, text classification, test clustering, text summarization, etc. Graph based approach for information retrieval has been widely studied and applied to different tasks due to its clearly-defined theory foundations and good empirical performance.

Because this topic is studied by different communities from different viewpoints and for usage in different applications, a wide range of graph models have been proposed. They greatly vary in the types of vertices, types of edge relations, the external semantic resources, the methods to produce structured representations of texts, weighting schemes, as well as the many subproblems focused on, from the selection feature as vertex and detection relationships between features, to matching graphs and up to ranking results. The rich choices of available information and techniques raise a challenge of how to use all of them together and fully explore the potential of graphs in text - centric tasks.

In [17], the text is represented as a graph by viewing the selected terms from the text as nodes and the co-occurrence relationships of terms as edges. Edges direction are defined based on the position of terms that occur together in the same unit . The weight is assigned to each edge so that the strength of relationship between two terms can be measured. Such graph model have the capability of retaining more structural information in texts than the numerical vector, but they do not take into account the meanings of terms and semantic relations between them.

Many richer document representation schemes proposed in [14]–[16], in which semantic relationship between words is considered to construct graphs. Vertex denotes terms mapped to concepts and edge denotes semantic relations specified in a controlled vocabulary or thesaurus, like synonymy or anotomy.

The method in [18], [19] took advantage of the DBpedia knowledge base for fine-grained information about entities and their semantic relations, thus resulting in a knowledge-rich document models. In these models, nodes are the concepts extracted from the document through references to entities in DBpedia using existing tools such as SpotLight or TagME. Those nodes are then connected by semantic relations found in DBpedia. The edges are weighted so as to capture the degree of relevance between concepts within an ontology. The different between these two works is that [18] also applied their model in the 'entity ranking' task in addition to the shared 'document semantic similarity evaluation' task. Moreover, not only [19] weighted edges like [18], they also weights concepts using closeness centrality measure which reflects their relevance to the aspects of the document. Another note is that these works disregarded structural information of the text, the relationships between nodes are independent of the given text.

The major difficulties in modeling document content with graphs are the development of an automated system to extract graph representation of text and the computation time limitation (time complexity). Besides, there may be difficulties in finding maximum common subgraph (subgraph isomorphism) between two document graphs, that are able to catch the semantic similarity between documents. Graph matching can be also accomplished in polynomial time making it impractical for large data sets.

In yet another attempt at those difficulties, we employ the graph based approach for representing and retrieving document in a very specific domain, where a fine grain ontological knowledge base can help noticeably improve retrieval performance. Our approach would be evaluate extrinsicly, which means only the final performance of the system will be considered, the quality of every internal processes are not yet attested. Our contributions are thus listed as follows:

- We propose a framework for building a semantic document retrieval system in a specific domain. Our framework aims to provide a systemmatic approach to better rank documents against a user query, with the help of a semantic resource.
- We also propose an Ontology model for domain knowledge to support various information retrieval tasks
- Graph-based document models along with a method to produce structured representations of texts are presented
- A graph matching algorithm to evaluate the semantic relevance for usage in searching would be introduced
- Finally, we evaluate search performance with the dataset of Information Technology Job Posting in Viet Nam

The remaining sessions of this paper are organized as follows: Section 2 is about a kind of document retrieval systems, called Semantic Document Base System, system architecture and design process; Sections 3 and 4 introduce an ontology model describing knowledge about a particular domain, a graph-based semantic model for representating document content; Section 5 presents techniques in semantic search; Section 6 introduces experiment, applications and finally a conclusion ends the paper.

II. Semantic Document Base System

A Semantic Document Base system (SDBS) is a computerized system focus on using artificial intelligence techniques to organize a text document repository on computer in an efficient way that supports semantic searching on the repository based on domain knowledge. It incorporates a repository (database) of documents in a specific domain, where content (semantics) based indexing is required, along with utilities designed to facilitate the document retrieval in response to queries. A SDBS considered here must have a suitable knowledge base used by a semantic index and search engine to obtain a better understanding and interpreting of documents and query as well as to improve search performance.

A semantic document base system has two main tasks:

- Offering multiple methods to retrieve documents from its database, especially the capability of semantic search for unstructured texts (i.e. the ability to exploit semantic connections between queries and documents, evaluate the matching results and rank them according to relevance).
- Storing and managing text documents and metadata, content based indexing to facilitate semantic search as well as managing the knowledge of a special domain for which the systems are developed.

Some other characteristics of a semantic document base system among the various kinds of document retrieval systems are as follows:

- A SDBS focuses on dealing with documents that belong to one particular domain, whereas existing knowledge resources in that domain can be exploited to improve system performance.
- A knowledge-rich document representation formalism as well as a framework for generating the structured representation of document content are introduced.
- A certain measure of semantic similarity between a query and a document is introduced.
- A proper consideration is imposed on the exploration of domain knowledge, the structural information and semantic information of texts, in particular, the occurrence of concepts and the relations existing between concepts.
- Offers a vast amount of knowledge in a specific area and assists in the management of knowledge stored in the knowledge base.

An overview of the system architecture is presented in Fig. 1. The structure of a SDB system considered here consists of some main components such as:

Semantic Document Base (SDB): This is a model for organizing and managing document repository on computer that supports tasks such as accessing, processing and searching based on document content and meaning. This model integrates components such as: (1) a collection of documents, each document has a file in the storage system, (2) a file storage system with the rules on naming directories, organizing the directory hierarchy and classifying documents into directories, (3) a database of collected documents based on the relational database model and Dublin Core standard (besides the common Dublin Core elements, each document may include some special attributes and semantic features related to its content), (4) an ontology partially describes the relevant domain knowledge and finally (5) a set of relations between these components.

Semantic Search engine: The system uses a special matching algorithm to compare the representations of the query and document then return a list of documents ranked by their relevance. Through the user interface, the search engine can interact with user in order to further refine the search result.

User Interface: Provide a means for interaction between user and the whole system. Users input their requirement for information in form of a sequence of keywords. It then displays search result along with some search suggestions for potential alternations of the query string.

Query Analyzer: Analyze the query then represent it as a "semantic" graph. The output of query analyzing process then be fed into search engine.

Semantic Collector and Indexing: Perform one crucial task in supporting semantic search, that is to obtain a richer understanding and representation of the document repository. The problems tackled in this module include keyphrase extraction and labeling, relation extraction and document modeling. This work presents a weighted graph based text representation model that can incorporate semantic information among keyphrases and structural information of the text effectively.

Semantic Doc Base Manager (including Ontology Manager): Perform fundamental storing and organizing task in the system.

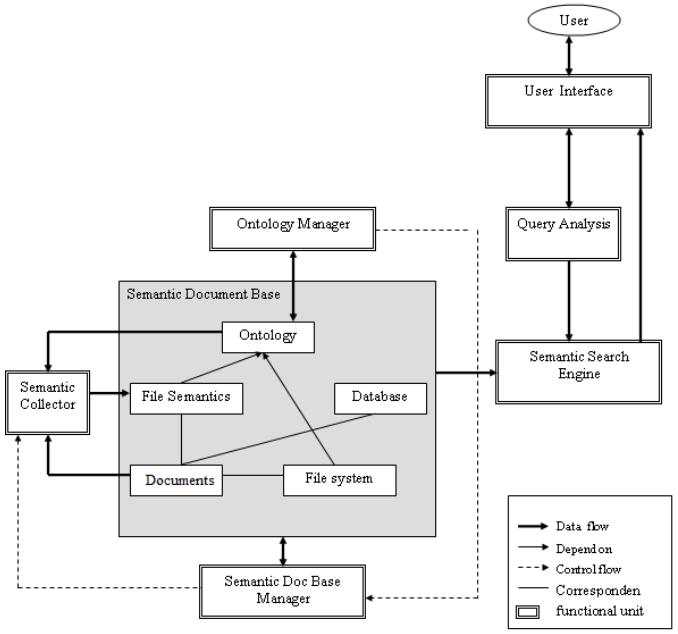

Fig. 1. Architecture of the SDB system

This paper describes the theoretical model of a semantic document base system by giving formal definitions to the "document representation" and the "similarity", with the occurrences of keyphrases, concepts and the semantic relations among them taken into consideration. Furthermore, there are some other important problems in a SDBS implementation point of view. The procedures as well as various kinds of data formats are described in order to implement the above model as a computerized system. The main models for representation of semantic information related to document's content will be presented in the next section.

III. The Classed Keyphrase based Ontology

Ontologies give us a modern approach for designing knowledge components of Semantic Information Retrieval Systems. Practical applications expect an ontology consisting of knowledge components: concepts, relations, and rules that support

symbolic computation and reasoning. In this article, we present an ontology model called Classed Keyphrase based Ontology (CK-ONTO). The CK-ONTO was made to capture domain knowledge and semantics that can be used to understand queries and documents, and to evaluate semantic similarity, first introduced in [20] and had some improvements in [4]. This ontology model was used to produce some practical applications in Information Retrieval. It can also be used to represent the total knowledge and to design the knowledge bases of some expert systems.

The preliminary CK-ONTO, however, was more of a lexical model than a fully structured Ontology. The central points in previous versions of CK-ONTO were the vocabulary of keyphrases (terms), as well as the internal relations between those keyphrases. Concepts and their structure received little attention.

In contrary, Gruber defined an ontology as an 'explicit specification of a conceptualization', which essentially means 'An ontology defines (specifies) the concepts, relationships, and other distinctions that are relevant for modeling a domain. The specification takes the form of the definitions of representational vocabulary (classes, relations, and so forth), which provide meanings for the vocabulary and formal constraints on its coherent use' [21].

Another definition of ontology was also given in [22]:

'An ontology may take a variety of forms, but necessarily it will include a vocabulary of terms, and some specification of their meaning. This includes definitions and an indication of how concepts are inter-related which collectively impose a structure on the domain and constrain the possible interpretations of terms.'

This paper presents a revised CK-ONTO model that is more on the line with contemporary ontology definitions. We still employ a vocabulary of keyphrases as the building block of our model but focus our efforts on structuralized concepts and their inter-relations. Ontologies must be both human-readable and machine-processable. Also, because they represent conceptual structures, they must be built with a certain composition.

Definition 1. *The Classed Keyphrase based Ontology (CK-ONTO), a computer interpretable model of domain knowledge for various information retrieval tasks, consists of four components:*

(K, C, R, Rules), where

- K is a set of keyphrases in a certain knowledge domain.
- C is a set of concepts in the domain.
- R is a set of relations that represent association between keyphrases in K or concepts in C.
- Rules is a set of deductive rules.

The structure of these components is presented in detail below, using the *Computer Science* domain as example:

A. A set of keyphrases: K

A keyphrase is an unequivocal phrase of relative importance in the domain. It can be a term that signifies a specific concept, an attribute of a concept or a unique entity in the domain.

Keyphrase is a linguistic unit structured as a word or a phrase. The syntactical classification of keyphrases yields three kinds: single keyphrase, compound keyphrase and modified keyphrase. A single keyphrase is either a single word or fixed phrase. For example, *computer, network, database, data structure, operating system, algorithm analysis and design, Arithmetic and logic unit, data mining* . A fixed phrase functions as a word, either with unique reference or as an idiom, is common in technical usage. The dividing line between a widely used ordinary phrase and a fixed phrase is not easy to determine. The degree of fixedness depends on frequency of occurrence and people's perception of the usage.

Compound keyphrases, on the other hand, are formed by two other keyphrases, or more. Based on the semantic of the relationship between constituents, compound keyphrases can be further classified as follows:

- Endocentric compound: one keyphrase is the 'head' and the others function as its modifiers, attributing a property to the head. For example: *database programming, network programming, document retrieval, wireless communication*.
- Dvanda compound: takes the form of multiple keyphrases concatenated together by using conjunctions, prepositions. For example, *data structures and algorithm, computer graphic and image processing*.

It is important to note that a single kephrase could be a complex combination of multiple words. But this 'combined word' contains only one keyphrase and thus can not be split into multiple keyphrases like a compound keyphrase.

A modified keyphrase, often consists of an adjective and a keyphrase, serves the same function as keyphrase. The adjective provides detail about, or modifies the original keyphrase. For example, *Low complexity, High complexity, classic Web content, rich multi-domain knowledge base*. There are numerous combinations created from this method, because there is no high stability so it may not have been collected in language dictionaries.

So, syntatically, we can consider the set of keyphrase K as $K = \{k|k \text{ is a keyphrase of knowledge domain}\}, K = K1 \cup K2 \cup K3$, in which, K1, K2, K3 are three sets of elements called single keyphrases, compound keyphrases and modified keyphrases, respectively.

On the semantic side, the set of keyphrases K can be partitioned into four subsets $K = K_A \cup K_E \cup K_C \cup K_N$ in which:

K_A, K_E, K_C are three subsets of keyphrases that imply attributes of some concepts, named entities (real-world objects such as persons, locations, organizations, products, etc.) or concepts respectively. And K_N is a set of keyphrases that have not been classifed. This semantic partition would prepare such set of keyphrases as the building block for other components of CK-ONTO discussed below. The partition is constructed by first identifying the relevant objects of the application domain, together with their relevant features.

B. A Set of Concepts: C

The main components of an ontology are concepts, relations, instances. A concept represents a set or class of entities (or objects, instances) or 'things' within a domain.

Concepts are basic cognitive units, each associated with a name and a formal definition providing an unambiguous meaning of the concept in the domain.A preferred label (name) is used for human readable purposes and in user interfaces. The matching and alignment of things is done on the basis of concepts (not simply labels) which means each concept must be defined. Concept can be defined by its intension and extensions. An extensional definition of a concept specifies a set of particular objects (also called instances) that the concept stands for. An intensional definition of a concept specifies its internal structure (attributes or slots) in either formal or informal way.

The definitional structure of each concept $c \in C$ can be modeled by *(cnames, Statement, Kbs, Attrs, Insts)*.

- $\emptyset \neq cnames \subseteq K_C$ is a set of keyphrases that can be used to name this concept. A $cnames$ is also called a synset which means a series of alternate labels to describe the concept. These alternatives include synonyms, acronyms that refer to the same concept.
- *Statement* is an informal (natural language) definition of this concept. For example, the *statement* of concept *PROGRAMING LANGUAGE* is *'A programming language is an artificial language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs that control the behavior of a machine and/or to express algorithms'*. The statement is a non-nullable human-readable string and does not need to be interpretable by computer.
- $Kbs \subseteq K$ is a set of "base" keyphrases where each keyphrase can be a descriptive feature of the concept. For example, concept *PROGRAMING LANGUAGE* can be described by the following base keyphrases: *artificial language, instructions, computer, program, algorithm*. The first place to look for base keyphrases could be the *Statement* of that concept.
- *Attrs* is either an empty set or a set of attributes of the class, describes its interior structure.
- Finally, *Insts* is an empty set or a set of instances. If *Attrs* is not empty, then each instance is a copy of the abstract concept with actual values for attributes. In case *Attrs* is an empty set, *Insts* would be a set of instance names which are keyphrases related to each other in certain semantics sense.

There are two most notable kinds of concepts. The first kind often refers to an area of interest in the domain, it is very difficult to define the exact attributes and instances of these concepts. Therefore, contents of these concepts would be described in our ontology through their base keyphrases and their relations to other concepts. Their attributes and instances would remain empty.

The second kind often refers to well-structured concepts, which means we can specify both their attributes and instances.

TABLE I. THE ATTRIBUTES OF CONCEPT *ALGORITHM*

Attribute name	type	range	sample value
isHeuristic	Boolean		true, false
isRecursive	Boolean		true, false
useDataStructure	Instance	{ARRAY, LIST, GRAPH, TREE}	liked list, stack, balanced tree, hash table, etc.
hasComplexity	Instance	{COMPLEXITY}	linear complexity, logarit complexity, exponential complexity, factorial complexity, etc.

The structure of an attribute and instance for these concepts is discussed below:

1) Attributes of a concept: Attributes (called properties or slots) are definitional constituents of concepts. Each instance of a concept will have the same set of attributes but with different values.

Each attribute $a \in Attrs$ is a triple $(attname, type, range)$, where $attname \in K_A$ is the naming keyphrase of the attribute. The $type$ of an attribute can be primitive data type in computer like string, integer, float, boolean, etc. For some attributes, the value could be an instance of another concept. In such case the *range* of such attribute would be a set of concepts from which instances can come. For example, some attributes of concept *ALGORITHM* are given in Table I.

2) Instances of a concept: Insts is the set of instances belonging to the concept, represents extensional components of the concept. All instances share the same structure as defined by the concept and thus can be model as a tuple $(instname, values)$ where $instname \in K \backslash K_A$ is the naming keyphrase of that instance and $values$ is the tuple of attribute values. In general, the sets of instances and attributes are expected to be disjoint. In case the concept has empty *Attrs* but non-empty *Insts*, each instance in *Insts* would consist of a name and an empty value set.

Some sample instances of concept *ALGORITHM* is given in Table II. Also, another example, the concept *PROGRAMMING* is described by Fig. 2.

TABLE II. SAMPLE INSTANCES OF CONCEPT *ALGORITHM*

instname	attribute	value
binary search	hasComplexity	logarithm
	useDataStructure	sorted array
	isHeuristic	false
	isRecursive	true
heap sort	hasComplexity	linearithmic
	useDataStructure	array
	isHeuristic	false
	isRecursive	false

C. A Set of Binary Relations on C - R_{CC}

The set of binary relations R is a tuple of two set $R = (R_{KK}, R_{CC})$.

A binary relation **r** on C is a subset of $C \times C$, i.e. a set of ordered pairs of concepts in C. It encodes the information of

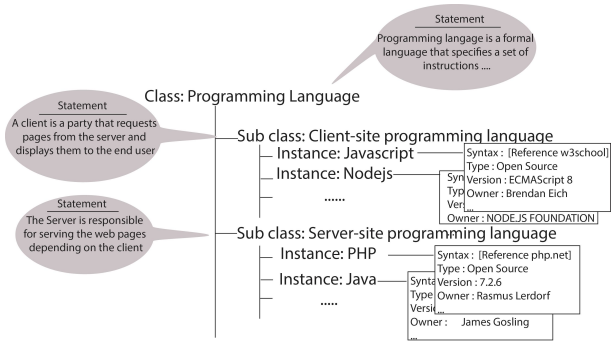

Fig. 2. An example of class *Programming language* in IT domain

relation: a concept c_1 is related to a concept c_2 if and only if the pair (c_1, c_2) belongs to the set. The statement $(c_1, c_2) \in r$ is read "concept c_1 is r-related to concept c_2", and is denoted by $c_1 r c_2$.

Each relation r will have an inverse denoted by r^{-1}, which is a relation with the order of two concepts reversed. In other words $\forall c_1, c_2 \in C, c_1 r c_2 \iff c_1 r^{-1} c_2$.

There are several kinds of semantic relations between concepts. The amount of relations may vary depending on the knowledge domain. These relations can be divided into two groups: hierarchical relations, non-hierarchical relations. So, relations also fall into two broad kinds:

1) Hierarchical relations among concepts: The most common forms of these are:

Hyponymy relation, also called 'is a' or 'kind of' relation links specific concepts with more general meaning ones, like *SORTING ALGORITHMS* is a more specific case of concept *ALGORITHMS*. We denote this relation as $r_{HYP} \in R_{CC}$. An interesting fact about this relation is that it can give us insights into the instances and attributes of concepts. Given two concepts $c_1, c_2 \in C$, it is possible to establish $c_1 r_{HYP} c_2$ if and only if the following conditions hold:

- Every instance of c_1 is also an instance of c_2

- Every attribute of c_2 is also an attribute of c_1

A class can include multiple sub classes or be included in other classes. A subclass is a class that inherits some properties from its superclass. The inheritance relationships of classes give rise to a hierarchical structure among classes.

Meronymy relation (r_{PART}), also known as 'a part of' or 'part-whole' or 'has a' relation, is another important hierarchical relation between concepts. For example, *CPU* is a part of *COMPUTER*.

Sub-topic relation (r_{SUB}) indicates that a concept is a sub area of another one like *ARTIFICIAL INTELLIGENCE* and *COMPUTER SCIENCE*, or, *LINKED DATA* and *SEMANTIC WEB*. While these so-called 'topical' concepts are hard to describe structurally, the capture of their hierachical relation play a vital role in many retrieval tasks.

2) Non-hierarchical relations : The three aforementioned hierarchical relations will incur three **'sibling' relations** denote as $r_{HYPSIB}, R_{PARTSIB}$ and r_{SUBSIB} respectively. Two concepts are sibling if they share a direct common parent in their hierarchy.

Domain-range relation, r_{RANGE}, links a concept to another concept in the *range* of its attributes. Given $c_1, c_2 \in C$, if there exists an attribute a of c_1 whose *type* is 'instance' and $c_2 \in range$ of a, we can say that $c_2 r_{RANGE} c_1$. For example, *COMPLEXITY* is in the range of attribute has Complexity of *ALGORITHM* , thus $(COMPLEXITY, ALGORITHM) \in r_{RANGE}$.

Depending on the domain knowledge, there exists many other types of non-hierarchical relationships that link concepts which are semantically related to each other without forming a hierarchy and no clear structural definition, such as Expansion, Cause, Influence, Instrument, Make, Possession, Source, Aim, Location, Temporal, Manner, Support, Beneficiary, Property, Agent, Circumstance, Related, etc.

Like binary relations general, our relations between concepts may have some properties like symmetric, transitive or reflexive, etc. A non-exhausted list of properties of relations in R_{CC} is given in Table III

TABLE III. PROPERTIES OF RELATIONS IN R_{CC}

relation	properties
Hierachical relations	transitive, reflexive, antisymetric
Domain-range relation	antisymetric
Sibling relations	transitive, reflexive, symetric

D. A Set of Binary Relations on K: R_{KK}

In addition to being a knowledge model of concepts and their relations, CK-ONTO also resembles a lexical model, in that it groups keyphrases together based on their meaning similarity and labels the semantic relations among keyphrases. This information is vital in many semantic retrieval tasks.

A binary relation **r** on K is a subset of $K \times K$. The statement $(x, y) \in r$ is read"keyphrase x is r-related to keyphrase y", and is denoted by xry. Keyphrases are interlinked by means of conceptual-semantic and lexical relations. There are three kinds of relations among keyphrases:

1) Equivalence relations: link keyphrases that have the same or similar meaning and can be used as alternatives for each other. There are two types of equivalence relations. The first one is 'abbreviation' relation, which links a short form or acronym keyphrase to its full form like *AI* and *Artificial Intelligence* or *Twittworking* and *Twitter networking*. This relation, denoted as r_{abbr}, is neither symmetric or transitive since two completely different keyphrases can share the same abbreviation, like *Best First Search* and *Breadth First Search* can both be abbreviated as *BFS*.

The other type of equivalence is synonymy relation, denoted as r_{syn},links keyphrases that can be used interchangably, like *Ontology Matching* and *Ontology Mapping*. This relation is fully symmetric and transitive, thus can be used to group keyphrases that share the same semantic meaning. The distinction between these two relations, therefore, should come from their semantical effects. If a short form keyphrase can

replace its full form ubiquitously with no additional disambiguation needed, that should be considered synonym rather than abbreviation.

When creating a synonymous groups of keyphrase, one should consider the spoke-and-hub model with one keyphrase serves as the centroid (hub) for the group and links to its synonymous keyphrase. The choice of hub keyphrase may not be trivial but the most popular keyphrase in the domain literature should be chosen in most cases.

2) Syntactical relations: that link compound keyphrase with its components. For dvanda compound, we have a simple 'formed by' relation (r_{formby}) from the compound keyphrase to each of its components. For endocentric compound, however, we have the 'head component' keyphrase and the 'modifier component' keyphrase, hence, there are 'headed by' relation (r_{headby}) and 'modified by' relation (r_{modby}) from an endocentric compound to its components respectively.

3) Semantic relations derived from concept relations: In information retrieval, there are many tasks that can be facilitated by the processing of terms and their relations, without any need for uncovering the structure of concepts. To better prepared our model for such tasks, we enrich R_{KK} with derived version of relations from R_{CC} including r_{hyp}, r_{part} and r_{sub} as hierarchical relations; $r_{hypsib}, r_{partsib}, r_{subsib}, r_{range}$ and $r_{related}$ as non-heirachical relations.

The exact keyphrase-keyphrase pair for each of these relations can be specified explicitly in addition to derivation from each element of R_{CC}. Since a keyphrase can express either a concept, an attribute or an instance, we would need some rules to deduce relations between keyphrases from relations between concepts. These rules will be discussed in the next section.

E. The Set of Rules

Rules is a set of deductive rules on facts related to keyphrases and concepts. A rule can be described as follows: $rule : \{f_1, f_2, ..., f_n\} \Rightarrow \{g_1, g_2, ..., g_m\}$ with $\{f_1, f_2, ..., f_n\}$ are hypothesis facts and $\{g_1, g_2, ..., g_m\}$ are goal facts of the rule.

Facts are concrete statements about 'properties of relations', 'relations between keyphrases' or 'relations between concepts'. The notations for each kind of facts are listed below:

Facts about properties of relations are written as $[< relation\ symbol > is < property >]$. For example, [r_{syn} is symmetric] means that the synonym relations between keyphrases is symmetric.

Facts about relations between keyphrases are written as $[< first\ keyphrase >< relation\ symbol >< second\ keyphrase >]$. For example, ['quick sort' r_{hyp} 'sorting algorithm'] means that keyphrase *quick sort* has hyponymy relation with keyphrase *sorting algorithm*.

Facts about relations between concepts are written as $[< first\ concept >< relation\ symbol >< second\ concept >]$. For example, ['EXPERT SYSTEMS' r_{SUB} 'ARTIFICIAL INTELLIGENCE'] means concept *EXPERT SYSTEMS* is a sub-topic of concept *ARTIFICIAL INTELLIGENCE*.

Some examples of rule include:

$\forall k_1, k_2, k_3 \in K, \forall r \in S_{R_{KK}}$ where $S_{R_{KK}}$ is aa set of symbols (or names) of the relations in R_{KK}

rule 1: if [r is symmetric] and $[k_1 r k_2]$ then $[k_2 r k_1]$

rule 2: if [r is transitive] and $[k_1 r k_2]$ and $[k_2 r k_3]$ then $[k_1 r k_3]$

rule 3: if $[k_1 r_{syn} k_2]$ and $[k_2 r k_3]$ then $[k_1 r k_3]$

Once keyphrases, classes and relations had been defined, rules should be described for constraint checking and inferring relation between two kephrases, between a keyphrase and a class, and between two classes. Moreover, rules also help (1) saving storage cost now that we don't have to manually store every single relationship, (2) enforce constraint and help reduce workload of a knowledge engineer when building ontology data, (3) the set of rules is an essential tool to deduce the direct or indirect relationships between keyphrases or concepts, the key step in evaluating the semantic similarity among keyphrases and concepts.

The Roles of CK-ONTO in Document Retrieval Systems

There are many ways to utilized CK-ONTO in different components of a document retrieval system.

- Document representation can be enriched.
  CK-ONTO can be viewed as a specific knowledge resource which be effective for language understanding tasks, i.e. can be used to understand and interpret queries and documents. In lexical models like WordNet, concepts correspond to senses of words. A concept in WordNet is represented as a synonym set and each synset is provided a textual definition, examples of its usage. Typical semantic relations between synsets include is-a relation, instance-of relation, part-of relation. In contrast, our CK-ONTO contains many different lexical and semantic relations between concepts or keyphrases. Keyphrases can refer well-structuralized concepts or specific entities.
  On the other hand, there are several existing general ontologies that can provide internal structural information about concepts or entities. However, they are massive in size, require additional disambiguation processing. Whereas CK-ONTO can facilitate quick, painless keyphrase extraction and graph-based document representation as pointed out in our previous iteration [4].
- Relevance evaluation between concepts or keyphrases is arguably the most common utilization of knowledge resources in retrieval systems. The semantic relevance between two concepts or keyphrases can be measured through their relations to other concepts. This measurement can then be used to expand query, ranking entities, representing document, semantic matching and so on. A good relevance evaluation strategy, tend to be specifically tuned to maximize the utilization of information provided in a specific resource. Therefore, we will propose a semantic relevance evaluation strategy based on CK-ONTO in the next section.

- The use of the ontology can also be useful for query expansion by means of introducing related keyphrases (or entities, concepts) and their content to expand the query. A 'heavy' domain ontology is preferred for fine-grain and precise expansion. However, we are yet to conduct formal experiment to substantiate the usefuleness of CK-ONTO in supporting query expansion tasks. Only system-wide experiment results are discussed in this article.
- Ranking model can exploit the ontology for matching the representations of texts. This is among the last steps in a retrieval systems, to determine the order of search results. A ranking scheme relied on earlier versions of CK-ONTO can be found in [4].

To build a knowledge base in CK-ONTO model is a task best supervised by well-trained domain experts. The process often involves these following steps:

- Collect a set of keyphrases in the domain from existing resources like dicitonaries, thesauri, Wikipedia, etc.
- Scan the document repository for any keyphrases that could have been missed in the previous step.
- Identify concepts and define their structures in CK-ONTO model.
- Determine the possible relations among concepts and employ inference engine based on the set of rules to deduce any additional relations among concepts and keyphrases.

Since the performance of various retrieval tasks heavily relied on ontology quality, it's ineluctable to have manual tuning from a team of experts in the domain. We built a web-based CK-ONTO management tool to help co-ordinate the efforts among teams of users. A screenshot of that tool is given in Fig. 3.

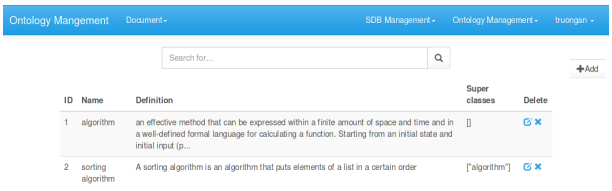

Fig. 3. A screenshot of CK-ONTO management tool

IV. Keyphrase Graphs for Document Representation

The work will focus on studying the method of text document representation, with the aim of converting documents into a structured form suitable for computer programs while still being able to describe core content of that text. We first briefly outline document representation formalism properties that we consider to be essential.

A. Requirements for a Document Representation Formalism

The content of document can be understood and interpreted in various ways. We are interested in document formalisms that comply, or aim at complying, with the following requirements:

- To allow for a structured representation of document content.
- To have a solid mathematically foundation.
- To allow users to have a maximal understanding and control over each step of the building process and use.

Document representation formalisms can be compared according to different criterias, such as expressiveness, formality, computational efficiency, ease of use, etc. A model is considered good if the following criterias are met:

1) Expressiveness: One of the fundamental challenges of text representation is the ability to represent information in text. The *Expresiveness* measures how "well" a representation can reflect the content of a document, i.e, what concepts and/or entities are mentioned in the document and what information can be inferred about them. A good representation has to capture both important structural information and semantic information, whereas structural information comprising of:

- The set of selected representative terms from text: A term is a simple word or phrase which helps to describe the content of document, and which may indeed occur in the document, once or several times (also called keywords, or keyphrases). Besides, "representative terms" can be more complex features like *n-grams*, *nouns phrases*, etc. extracted using various linguistic processing techniques
- Frequency of terms: the number of occurrences of terms in a document or in a collection of documents reflects their importance and specificity in the texts.
- The ordering information among terms.
- The co-occurrence of terms in different window sizes, i.e. terms can occurrence together in a sentence, a paragraph, or in a fixed window of n words and the evaluation for the strength of this relation. There is an assumption that if terms appear together in the units (as a sentence, different parts of a sentence) with a higher frequency, it means there is a close relationship between them, so thus the corresponding link should be weighted stronger.
- Location of terms in text: position information of terms at any content item (title, abstract, subtitle, content, etc.), at the beginning, middle or end of the text.

We define three levels of effectiveness in capturing structural information, described in Table IV.

Richer document representation schemes can be obtained by considering not only words or phrases but also semantic relations between them. The meaning of a document is the result of an interpretation done by a reader. This interpretation task needs much more information than the data contained in the document itself. The understanding the content of a document involves not only the determination of the main concepts mentioned in the document but also the determination of semantic relations between these concepts. Besides, the importance of representative concepts, how strongly they relate to each other should also be considered. The semantic information discussed in this paper is the meaning of a text derived

TABLE IV. LEVEL OF STRUCTURAL INFORMATION EXPRESSIVENESS

Criteria	Level 1	Level 2	Level 3
Model can capture structural information	Record the set of words appear in the document, with or without weighting parameter to indicate the importance of those words in the document.	Record set of phrases or features in the document along with their weights, location information	In addition to level 2, also record the co-occurence relation among features.
Example model	Bag of Words, Vector Space Models, etc.	Bag of complex features such as n-grams, Nouns phrases, (head, modifier, ..., modifier) tuples, etc.	Co-occurrence Graph based on the co-occurrence of feature terms in the document.

TABLE V. LEVEL OF SEMANTIC INFORMATION EXPRESSIVENESS

Criteria	Level 1	Level 2	Level 3
Model can capture semantic information	Represent document as a bag or vector of concepts (or entities) mentioned in the document with or without frequency weighting. Concepts are linked to an external semantic resource	Represent document as a bag or vector of concepts where relations between such concepts in the semantic resource are exploited in the weighting process.	Represent document as a graph of concepts with vertex weights reflecting the importance of concepts in document and edge weights representing the strength of relationship between two corresponding concepts. Difference kinds of relationships are recorded

from lexical semantics which are the underlying meanings of terms in the document and term relations or conceptual semantics which capture the cognitive structure of meaning. There are two main approaches to extracting semantic information. The first one employs Natural Language Processing techniques to parse the grammatical structure of the document into computer friendly representation. In this article, however, we will focus on the second approach, that is employing an external knowledge source to infer the meaning of document. The semantic information unearth using this approach may consist of:

- List of concepts or entities discussed in the document. Depending on the type of semantic resource being used, the structure of concepts may vary. In lexical models, concepts correspond to senses of words whereas concepts in knowledge models (abstract models of knowledge) stand for classes of real-world entities. Lexical concepts may refer to entities, classes, relations, attributes, or other senses of words and can be organized along lexical relationships in a lexical model. Knowledge models basically represent classes, attributes associated with these classes, and relations between classes.
- Relationships between concepts or entities reflected in the document. There are various kinds of association between concepts that raises a challenge of how to explore fully the potential of them and how to use some or all of them together.
- Weights associated with concepts (or entities) which reflect their relevance to the aspects or topics of the document.
- Weights associated with relationships between concepts which capture the strength of those relationships, i.e. the degree of associativity between concepts, how strongly related the two corresponding concepts are.

Levels of effectiveness in capturing semantic information may be considered as in Table V.

2) Formality: Components in a representation model have to be defined on a strong foundation with logically and mathematically sound notations. Further operations facilitated by the model also have to be well stated in the same notations so that they can be proved and implemented.

The formality is vital since it helps with the disambiguation and thus reduces error rate when using the model on real life data.

3) Computational efficiency: The specification language of the model has a simple structure but can represent knowledge domain and content of documents adequately. Users can employ it to represent, update, search, store easily as well as control over each step of the building process. Moreover, technical difficulty and utilization available tools or technologization should be considered. We are interested in representation formalisms that can be used for building systems able to solve real, complex problems. It is thus essential to anchor these formalisms in a computational domain having a rich set of efficient algorithms so that usable systems can be built. Due to the importance of natural language, a document representation formalism should allow the user to easily understand the results given by the system. The ability for describing the natural semantics is a good empirical criteria for delimiting the usability of the formalism.

Motivated by the previous work, this paper deals with the problem of document representation, provides a more expressive way to represent the texts for multiple tasks such as document retrieval, document similarity evaluation, etc. We propose graph based semantic models for representing document content which consider the incorporation of structural (syntactic) information and semantic information in texts to improve performance. Exploiting domain specific or general knowledge have been studied for acquiring fine - grained information about concepts and their semantic relations, thus resulting in knowledge-rich document models.

B. Modeling Document as Graph over Domain Knowledge

This subsection is devoted to an intuitive introduction of Keyphrase Graphs. The graph-based document representation formalism is introduced in detail. This formalism is based on a graph theoretical vision and complies with the main principles delineated in the previous subsection. Document Representation has long been recognized as a central issue in Document Retrieval. Very generally speaking, the problem

is to symbolically encode text document in natural language in such a way that this encoded document can be processed by a computer to obtain intelligent understanding.

We use the term "keyphrase graphs" (KGs in short) to denote the family of formalisms and use specific terms, e.g. simple keyphrase graph, weighted keyphrase graph, full weighted keyphrase graph —for notions which are mathematically defined in this paper.

A simple keyphrase graph is a finite, directed, muligraph. "Multigraph" means that a pair of nodes may be linked by several edges. Each node is a keyphrase that occurs and of relative importance in the domain. Edges express relationships that hold between these keyphrases. Each edge has a label. An edge is labeled by a relation name. A simple keyphrase graph is built relative to an ontology called CK-ONTO and it has to satisfy the constraints enforced by that ontology.

Definition 2. *Let $O = (K, R_{KK})$ be a sub-model derived from a domain ontology in the CK-ONTO formalism. A simple keyphrase graph (KG) defined over O, is a tuple (V, E, ϕ, l_E) where:*

- $V \subset K$ is the non-empty, finite set of keyphrases, called set of vertices or nodes of the graph.
- E is a set of directed edges.
- $\phi : E \to \{(x, y)|(x, y) \in V^2, x \neq y\}$ an incidence function mapping every edge to an ordered pair of distinct vertices. The edge represents a semantic (conceptual) relationship between its two adjacent vertices. The two vertices $k1, k2 \in V$ are connected if there exits a relation $r \in R_{KK}$ such that $(k1, k2) \in r$.
- $l_E : E \to T_R$ is a labeling function for edges. Every edge $e \in E$ is labeled with a relation name $l_E(e) \in T_R$. T_R is a set of names of binary relations found in R_{KK}.

O is composed of two sets: a set of keyphrases and a set of binary relations between keyphrases and can be considered as a rudimentary ontology. In contrast to lexical resourses like WordNet, our ontology contains many different, well-controlled semantic relations. In some works, it is assumed that O has a specific structure, such as a graph, thus a simple keyphrase graph can be viewed as a subgraph of O. A KG has nodes representing defined keyphrases in the domain ontology and edges representing semantic relationships found in the ontology between these keyphrases. Keyphrase nodes can refer to concepts or specific entities of domain knowledge. Important differences between the keyphrase graph model and other semantic networks are to be pointed out:

Compared to Conceptual Graph (CG), the structure of Keyphrase Graph is leaner. CGs are buit on a vocabulary of three pairwise disjoint sets: the ordered set of concept types, the set of relation symbols, and the set of typed individual markers. A concept type can be considered as a class name of all the entities having this type. In KG definition, on the contrary, the vocabulary K is a mixture of concepts' names (the counterpart of concept types), entities' names (the equivalence of individual markers) and many other things. A concept node in CG refers to either a specific entity, labeled by a pair (type, marker), or an unspecified entity with just the 'type' label. On the other hand, the nodes in KG are labeled with only the keyphrase. This is possible because the majority of information about each keyphrase can be inferred from CK-ONTO, we can get by with fewer annotations on keyphrase graph nodes.

Since the definition of CGs does not specified any relationship among concepts beyond simple a-kind-of relations. The determination of possible semantic relationships between concept types in CGs must use some complex natural language processing techniques and external resources. Whereas for keyphrase graphs, relationships can be quickly found by exploiting information about relations within the ontology or deducing from them.

Recently, various graph models use general kwowledge bases (e.g. DBpedia, Freebase) as the backend ontologies. Such knowledge bases contain knowledge about concepts or real-world entities such as descriptions, attributes, types, and relationships, usually in form of knowledge graphs. They share the same spirit with controlled vocabulary but are created by community efforts or information extraction systems, thus have a large scale, wide-coverage [23].

Due to such wide-coverage, when comparing to a domain specific ontology like CK-ONTO, those general knowledge bases often have a higher degree of conceptual overlapping and ambiguity. Thus various disambiguation techniques are required when using those knowledge bases, an unnecessary burden for retrieval tasks in a specific domain.

Definition 3. *Let $O = (K, R_{KK})$ be a sub-model derived from CK-ONTO. A weighted keyphrase graph (wKG) defined over O, is a tuple $(V, E, \phi, l_E, w_V, w_E)$ where:*

- (V, E, ϕ, l_E) is the simple keyphrase graph.
- $w_V : V \to \mathbb{R}^+$ and $w_E : E \to \mathbb{R}^+$ are two mappings describing the weighting of the vertices and edges.

In some works, not all keyphrases or all relations are equally informative, so numerical weights associated with them are necessary. Such weight might represent for example cost, length, capacity, descriptive importance or degree of associativity, depending on the problem at hand.

Graphs are commonly used to encode structural information in many fields, and graph matching is an important problem in these fields. The matching of a graph to a part of another graph is called subgraph matching problem or subgraph isomorphism problem. So, we are interested here in subgraphs of a KG that are themselves KGs.

Definition 4. *Let $G = (V, E, \phi, l_E)$ be a simple keyphrase graph. A sub keyphrase graph (subKG) of G is a simple keyphrase graph $G' = (V', E', \phi', l'_E)$ (denoted as $G' \leq G$) such that: $V' \subseteq V$, $E' \subseteq E$, ϕ', l'_E are the restrictions of ϕ, l_E to E' respectively, and $\phi'(E') \subseteq V' \times V'$. Conversely, the graph G is called a super keyphrase graph of G'.*

Definition 5. *Let $G = (V, E, \phi, l_E, w_V, w_E)$ be a weighted keyphrase graph. A sub weighted keyphrase graph (sub-wKG) of G is a weighted keyphrase graph $G' = (V', E', \phi', l'_E, w'_V, w'_E)$ (also denoted as $G' \leq G$) such that: $(V', E', \phi', l'_E) \leq (V, E, \phi, l_E)$ and the weights of every vertices and edges of G' are equal to their counterparts in the super keyphrase graph G.*

A subKG of G can be obtained from G only by repeatedly deleting an edge or an isolated vertex.

Keyphrase graphs are building blocks for representing different kinds of texts, e.q. used for the semantic representation of documents and queries. Keyphrases are the most relevant phrases that best characterize the content of a document. Keyphrases provide a brief summary of the content, and thus be used to index the document and as features in further search processing. Furthermore, understanding the document content involves not only the determination of the main keyphrases occur in that document but also the determination of semantic relationships between these keyphrases. Therefore, each document can be represented by a compact graph of keyphrases in which keyphrases are connected to each other by semantic relationships. Nodes represent keyphrases extracted from the document through references to explicit keyphrases in a domain ontology. We can assign a weight to each keyphrase in the given document, representing an estimate of its usefulness as a descriptor of the document. Similarly, each relation edge in the document graph also allocated a weight (usually but not necessarily statistical) which reflects the membership degree between two direct keyphrases. This is a distinctive feature of weighted keyphrase graphs: they allow to represent semantic and structural links between keyphrases and measure the importance of keyphrases along with the strength of relationships whereas poor representation models cannot.

Definition 6. *Let $O = (K, R_{KK})$ be a sub-model derived from CK-ONTO. Given a document d which belongs to a collection D of documents in a specific knowledge domain. A weighted keyphrase graph, which represents the document d (denoted as $docKG(d)$), defined over O, is a tuple $(V, E, \phi, l_E, w_V, w_E)$ where:*

- $(V, E, \phi, l_E, w_V, w_E)$ is a weighted keyphrase graph whose vertices and edges can be weighted with some statistical or linguistic criterion.
- (l_E, w_E) are two labeling functions for edges of the graph. Every edge $e \in E$ is labeled by a pair $(l_E(e), w_E(e))$ where $l_E(e)$ is a name of semantic relation in R_{KK} , $w_E(e)$ is the weight assigned to the current edge . This weight is a measure of semantic similarity between two keyphrases.
- w_V is a labeling function for vertices of the graph Each keyphrase vertex $k \in V$ is assigned a weight $w(k, d)$, which is a measure of how effective the keyphrase k is in distinguishing the document d from others document in the collection

The most expressive keyphrase graph is called full weighted keyphrase graph. The basic idea of the extension from weighted keyphrase graph to full weighted keyphrase graph is that there are various kinds of association between keyphrase vertices considered. We consider different types of relationships among keyphrases and their environment in the domain ontology as well as in the documents.

Definition 7. *Let $O = (K, R_{KK})$ be a sub-model derived from CK-ONTO. Given a document d which belongs to a collection D of documents in a specific knowledge domain. A full weighted keyphrase graph, which represents the document d (denoted as $fulldocKG(d)$), defined over O, is a tuple $(V, E_1, E_2, \phi_1, \phi_2, l_{E_1}, l_{E_2}, w_V, w_E)$ satisfying the following conditions:*

- $(V, E_1, \phi_1, l_{E_1}, w_V, w_E)$ is a weighted keyphrase graph representing d .
- E_2 is a set of directed edges representing syntactic relationships between keyphrase vertices (the edge set of graph is $E = E_1 \cup E_2$) and $\phi_2 : E_2 \rightarrow \{(x, y)|(x, y) \in V^2, x \neq y\}$ maps every edge to an ordered pair of distinct vertices. In addiction to semantic relationships, the two keyphrase vertices $k1, k2 \in V$ can also be connected if there exits some forms of syntactic relationship between them such as co-occurrence or grammatical relationships.
- $l_{E_2} : E_2 \rightarrow T_S$ is a labeling function for edges in E_2. T_S is a set of names of binary syntactic relations used for labeling such edges.
- $w_E : E \rightarrow \mathbb{R}^+$ is used for weighting edges. Such weights capture the degree of relevance between keyphrases in the graph.
- Two keyphrases are connected by co-occurence relationship if they appear in the same sentence. The edge connecting them is labeled "co-occurrence", its direction is based on the order in which those two keyphrases appear. The weight of such edge reflects how strongly the two keyphrase related and could be measured by the frequency they appear together.
- The syntactic relationship is a special kind of co-occurence relationship, when grammatical roles of the two keyphrase can be inferred. The label, direction and weight of edge in case may vary depending on the domain knowledge and the parsing technique.

C. Weighted Keyphrase Graph Construction

1) A general framework for document graph generation: We present a method to generate the structured representation of textual content using CK-ONTO as the backend ontology. The key idea of document representation by a keyphrase graph is to link the keyphrases in the document text to concepts/entities of a domain ontology in the CK-ONTO formalism, and to explore the semantic and structural information among them in the ontology as well as in the text body.

Given an input text document d, the process of generating a full weighted keyphrase graph *fulldocKG(d)* representing d consists of the following stages:

- Step 1: Extract keyphrases in the text d, that correspond to defined keyphrases in the knowledge base CK-ONTO. This step is in iteself an active research problem, resulting in a variety of existing tools. However, in some specific domains, human intervention is still unavoidable to form a concise list of vertices of the graph. Then weights will be assigned to each vertex and some popular weights like tf, idf¸, ect. are good starting point.
- Step 2: Connect the extracted keyphrase vertices using their semantic and/or structural relationships. Each

pair of keyphrases k_i and k_j are connected by an edge in two cases: 1) If they are directly linked by a relation defined on CK-ONTO, that relation name is also used to label the edge. 2) If they occur together in a sentence, syntactic parsing techniques are employed to determine the syntactical relation between them, otherwise they only have simple "co-occurrence" relation.
Based on the observation that the core aspects of a document should be a set of closely related keyphrases, the strength of associations among keyphrases are used for the representation to better reflect the semantics of the text. The weight on the directed edge r connecting k_i and k_j reflects the strength of relationship between two keyphrases, based on their features and relationships in the domain ontology. Moreover, keyphrases that frequently appear together in a document or in many documents of the collection tend to have stronger links between them. This kind of association reflects how often two keyphrases share contexts. However, the exact formula for edge's weight may vary depending on the type of the document.

- Step 3: If a group of synonym keyphrases are extracted, remove all but the one with highest weight and update the weight of this keyphrase.
- Step 4: Compute the weight for each edge to evaluate the strength of the corresponding relation.

A query may be specified by the user as a set of keyphrases or in natural language. In the latter case, the query can be processed exactly like a miniature document in similar manner. A natural language query can receive the usual processing, i.e., keyphrase extraction, relationship identification, etc. transforming it into a graph of keyphrases.

2) Assigning weights to keyphrase vertices and relation edges: Each keyphrase vertex k of the keyphrase graph representing the document d is assigned a weight w(k, d), which is a measure of how effective the keyphrase k is in distinguishing the given document d from other documents in the same collection. There are many strategies to weight keyphrase nodes and a variety of weighting schemes have been used. The exact scheme for automatic generation of weights may vary depending on the characteristics of the document repository. The formulas below were used in some of our applications and are listed here for examplary purpose.

The weight associate with the keyphrase node k of the keyphrase graph docKG(d), representing an estimate of the usefulness of the given keyphrase as a descriptor of the document d, is computed by:

$$w(k,d) = tf(k,d) \times idf(k,D) \times ip(k,d) \tag{1}$$

The "term frequency" tf(k,d) is the frequency of occurrence of the keyphrase k within the given document d, that reflects the importance of the keyphrase within a given document according to the number of times it appears in the document, is computed by:

$$tf(k,d) = c + (1-c)\frac{n(k,d)}{max(\{n(k',d)|k' \in d\})} \tag{2}$$

where $n(k,d)$ is the number of occurrences of the keyphrase k in the document d. Parameter $c \in [0,1]$ is the predefined minimum tf value for every keyphrase. This parameter reflects one's confident in the keyphrase extraction process, that means any keyphrase extracted must have a certain value of importance as a descriptor of the document and in the worst case it should have a tf of at least c.

In large (long) documents like books and thesis, some 'popular' keyphrases can appear a thousand fold more times than a more specific keyphrase, leading to a very low frequency for this specific keyphrase. This parameter also help prevent keyphrases from being overshadowed in large documents. The value of c is chosen through experimenting and can be fine-tuned to suit different specific applications.

The "Inverse document frequency" idf(k, D) is a measure of how widely the keyphrase k is distributed over the given collection of documents D and computed by:

$$idf(k,D) = log\left(\frac{|D|}{1+|\{d \in D, k \in d\}|}\right) \tag{3}$$

where $|D|$ is the total number of documents in the collection and $|\{d \in D, k \in d\}|$ is the number of documents where the keyphrase k appears.

The "Importance of Position" ip(k,d) represents an estimate of the importance of keyphrase k in document d based on the location of occurrence of k in the document, is defined as:

$$ip(k,d) = a + (1-a)\frac{\sum_{i \in A} w_i}{\sum_i w_i} \tag{4}$$

in which, w_i is the weight assigned for the ith component of document d , representing the importance of ith component of document structure. The set of the index of all components in which k appear defined as $A = \{x|n_x(k,d) > 0\}$, on top of that we can defined Parameter $a = max(w_i|i \in A)$ as the weight of the most important component where k appears, also serves as the predefined minimum value for $ip(k,d)$. The number of a document's component and the weight for each component is different for each type of document. In a paper, for example, the title and abstract are much more important in helping readers quickly grasp the general meaning of the text, so the keyphrases appear in these components are always considered to be more significant and should have the largest weight.

By adopting $tf \times idf \times ip$ weighting scheme, such weighting scheme assumes that the best descriptors of a given document will be the keyphrases that occur often in the document and very rarely in other documents and they are likely to occur in important content items of the document (such as title, subtitle, abstract, etc.).

Similarly, weights are also assigned to relation edges in the graph. The weight on the directed edge r connecting k_i and k_j reflects the strength of the relationship between pair

of keyphrases. Commonly, if keyphrases appear together in a sentence with a higher frequency (within given document), it means there is a stronger link between them. However, in some types of documents, the number of times that keyphrases occur in the texts could be low, so k_i and k_j rarely co-occur more than once. Therefore, the weight assigned to an edge can be considered by the relative frequency of co-occurrence of its both adjacent keyphrase vertices (in a sentence) over the given collection. Thus, the formula for edge's weight may vary depending on the type of the document. An example forumla will be given in Section **??**.

We demonstrate the benefits of these semantic representations in the following search task:

V. Graph based Document Retrieval

This paper deals with the problem of document representation for the task of ad-hoc document retrieval. The main task is to retrieve a ranked list of (text) documents from a fixed corpus in response to free-form keyword queries. In this work, the query and documents are modeled by enhanced graph-based representations. We define several semantic similarity measures which consider both semantic and statistical information in documents to improve search performance.

A. Semantic Relevance Evaluation

Relevance evaluation between the target query and documents is done by calculating the semantic similarity between two keyphrase graphs that represent them. A keyphrase graph is constituted by keyphrase nodes and relation edges, so the similarity between two keyphrase graphs is calculated by means of their pairwise similarity.

1) Semantic similarity between two keyphrases: This subsection will discuss a method to estimate the similarity between two keyphrases, the most basic components in CK-ONTO, from which other similarity metric can be built upon.

Let $\alpha : K \times K \rightarrow [0, 1]$ be the mapping to measure semantic similarity between two keyphrases. Value 1 represents the equivalence between two keyphrases and value 0 corresponds to the lack of any semantic link between them. To calculate the value of α we first have to present some preliminary definitions:

Definition 8. *Given a knowledge domain modeled by CK-ONTO O = (K, C, R, Rules) and two keyphrases $k, k' \in K$, the keyphrase k' is called **directly reachable** from the keyphrase k if there exists a relation $r \in R_{KK}$ such that $(k, k') \in r$ (or written as k r k'). We can also said that k' is directly reachable from k by r.*

*When k' is directly reachable from k by relation $r \in R_{KK}$, the triplet (k, r, k') could be assigned a decimal number in the interval (0.0 ... 1.0], denoted as $val(k, r, k')$. This number stands for the **axiomatic similarity degree** of k and k' according to r.*

The similarity degree of two keyphrases linked by a relation depends mostly on that relation. For example, two keyphrase linked by $synonym$ relation must have much larger similarity degree than two keyphrases linked by $hyponym$ relation. On the other hand, two pairs of keyphrases linked by the same relation may have slightly different semantic similarity. This value should be established by a panel of experts in the given domain adhering to some constraints, for example:

- $\forall k_1, k_2, k_3, k_4, k_5, k_6 \in K$, if $k_1 r_i k_2, k_3 r_j k_4, k_5 r_t k_6$, where r_i is a equivalence relation, r_j is a hierarchical relation and r_t is a non-hierarchical relation then $val(k_1, r_i, k_2) > val(k_3, r_j, k_4) > val(k_5, r_t, k_6)$
- $\forall k, k' \in K$ if $k r_i k'$ where $r_i \in \{r_{syn}, r_{abbr}\}$ then $val(k, r_i, k') \approx 1$

Definition 9. *Given a knowledge domain modeled by CK-ONTO O = (K, C, R, Rules) and two keyphrases $k, k' \in K$, the keyphrase k' is **reachable** from the keyphrase k if there is a chain of keyphrases $k_1, k_2, ..., k_n$ with $k_1 = k$ and $k_n = k'$ such that k_{i+1} is directly reachable from k_i, for i = 1,..., n-1.*

Let $R_{KK} = \{r_1, r_2, ..., r_m\}$ be a set of binary relations on K, sequence of integers $S = (s_1, s_2, ..., s_{n-1}), s_i \in [1, m], r_{s_i} \in R_{KK}$, the notation $(k_1 r_{s_1} k_2, k_2 r_{s_2} k_3, ...k_{n-1} r_{s_{n-1}} k_n)$, called a **path** of length n-1 from k to k' in CK-ONTO, denotes a finite sequence of relations which joins a sequence of distinct keyphrases and obtained from the reachable relation between k and k'. $(r_{s_1}, r_{s_2}, ..., r_{s_{n-1}})$ is the relation sequence of the path and $(k_1, k_2, ..., k_n)$ is the keyphrase sequence of the path.

Definition 10. *Given a path $(k_1 r_{s_1} k_2, k_2 r_{s_2} k_3, ...k_{n-1} r_{s_{n-1}} k_n)$ from k_1 to k_n in CK-ONTO, the **weight** of such path is defined by the formula*

$$V(k_1 r_{s_1} k_2, k_2 r_{s_2} k_3, ...k_{n-1} r_{s_{n-1}} k_n) = \prod_{1}^{n-1} val(k_i, r_{s_i}, k_{i+1})$$

Definition 11. *For all $k, k' \in K$, the mapping α measuring semantic similarity between k and k' would be defined as follows:*

- $\alpha(k, k') = 1$ if $k = k'$
- $\alpha(k, k') = 0$ if k' is not reachable from k
- $\alpha(k, k') = Max(\{V(P) \mid P \text{ is a path from k to k'}\})$ otherwise

There may exist many paths from k to k' and the value of $\alpha(k, k')$ would be the maximum weight of those paths. So to calculate $\alpha(k, k')$ we have to solve the **maximum weight path problem**, which is to find the path of maximum weight from keyphrase k to k'.

However, one may note that if we extend an existing path by adding one more relation and keyphrase to it, its weight will be multiplied by a number between 0 and 1, thus will likely to decrease. Therefore, our maximum weight path problem is indeed a special case of shortest path problem which can be solved quite easily.

The algorithm 1 is a modified version of the classic Dijkstra algorithm that can calculate $alpha$ between two keyphrase. The typical complexity of Djkstra algorithm implement using binary heap is $O((|E| + |V|) * log|V|)$ whereas in our case, $|E| = \sum_{r \in R_{KK}} |r|$ and $|V| = |R_{KK}| * |K|$

Algorithm 1 Calculate semantic similarity between two keyphrase k_1 and k_2

Data: $O = (K, C, R, Rules)$ - the knowledge domain modeled by CK-ONTO, where $R = (R_{KK}, R_{CC})$
Input : Two keyphrases $k_1, k_2 \in K$
Output: The semantic similarity $\alpha(k_1, k_2)$
Q ← Empty Priority Queue /* Each item in Q is a $\{keyphrase, value\}$ pair and item with maximum value is at the front of the queue */
visited ← Empty hash table /* Used to keep track of visited keyphrase */
Q.enQueue($\{k_1, 1\}$)
while Q *is not empty* **do**
  $\{k, value\} \leftarrow$ Q.DeQueue()
  visited.insert(k)
  if $k = k_2$ **then**
    return value
  else
    foreach *relation* r *in* R_{KK} **do**
      foreach *keyphrase k' in K where* $k\ r\ k'$ **do**
        /* We consider every keyphrase k' with whom k have relationship r */
        $nextValue \leftarrow value \times val(k, r, k')$
        if visited.Contain(k') = *false* **then**
          Q.enQueue($\{k', nextValue\}$)
        end
      end
    end
  end
end
return 0 /* There is no more keyphrase to visit */

2) Semantic similarity between two relations: When dealing with the determination of possible relationships between keyphrases, one may notice that there could be more than one way to making sense of the relation between a pair of keyphrases. For example, when two keyphrases that occur in the same sentence, one can try to deduce their relation in terms of grammatical role in the sentence or just simply leave them as having 'co-occurence' relation, whatever suits the application at hand. Another example is the 'kind-of' relation and 'subtopic' relation. They are sometimes interchangeable (depend on how one categorizes the set of keyphrase). This notion of interchangeability between relations gives rise to the demand for semantic similarity evaluation between two relations:

Let $\beta : T_R \cup T_S \times T_R \cup T_S \rightarrow [0, 1]$ be a mapping which allows to value the semantic similarity between two relations. T_R is a set of relation names found in R_{KK} and T_S is a set of names of syntactic relations between keyphrases. Because the number of relations is small, we can determine the values of β through an arbitrary pre-defined lookup table. Although the expression of this function can be determined arbitrarily (even the values of β can manually been chosen), some constraints should be considered, for example:

- $\forall r \in T_R \cup T_S, \beta(r, r) = 1$.
- $\beta(synonymy, abbreviation) = 1$.
- Relations that are in the same group (such as Hierarchical relations) should have more semantically likeness than relations in different groups.

3) Semantic similarity between two keyphrase graphs: The fundamental notion for studying and using KG is homomorphism, also called a projection. A KG projection is a mapping between two KGs that preserves the KG structure and provides means to evaluate the relevance between two KGs. More concretely, a projection from a KG H to a KG G is a function from the nodes of H to the nodes of G, which respects their structure, i.e. it maps adjacent vertices to adjacent vertices.

Definition 12. *Let* $H = (V_H, E_H, \phi_H, l_{E_H})$ *and* $G = (V_G, E_G, \phi_G, l_{E_G})$ *be two simple keyphrase graphs defined over the same* $O = (K, R_{KK})$ *of CK-ONTO. A KG projection from H to G is an ordered pair* $\Pi = (f, g)$ *of two mappings* $f : E_H \rightarrow E_G$, $g : V_H \rightarrow V_G$ *satisfying the following conditions:*

- f and g are injective functions.
- The projection preserves the relationships between vertices of H, i.e. for all $e \in E_H, g(adj_i(e)) = adj_i(f(e))$, $adj_i(e)$ denotes the ith vertex adjacent to edge e.
- $\forall e \in E_H, \beta(l_{E_H}(e), l_{E_G}(f(e))) \neq 0$.
- $\forall k \in V_H, \alpha(k, g(k)) \neq 0$.

The following condition can be set if desired: $\forall r, r' \in T_R \cup T_S$ where $r \neq r', \beta(r, r) \neq 0$. This condition allows that there exists a projection from any relation edge to any other one.

The definition of KG projection provides the vessel through which we can evaluate the relevance between two piece of texts represented by keyphrase graphs. However, some texts can be considered as related to each other even if only a portion of them are similar. Therefore, it could be more feasible to find a projection from only a portion of keyphrase graph to another keyphrase graph. We call this a partial projection:

Definition 13. *There is a partial projection from a keyphrase graph H to a keyphrase graph G if there exists a projection from H', a sub keyphrase graph (subKG) of H (H' ≤ H), to G.*

Below described formula allows valuation of one projection. In valuation formula of the projection from H to G, H is a query graph and G is a document graph.

Definition 14. *Let H is a keyphrase graph of the query q and G is a keyphrase graph of the document d and H' ≤ H. A valuation of a partial projection* Π *from H' to G is defined in formula (5):*

$$v(\Pi) = \frac{|V_{H'}|/|V_H| \sum_{k \in V_{H'}} w(g(k), d) \cdot \alpha(k, g(k)) + \sum_{e \in E_{H'}} \beta(e, f(e)) \cdot w(e)}{|V_{H'}| + |E_{H'}|} \quad (5)$$

The main idea of a searching method is the semantic relevance calculation between a query and a document. Therefore, it is necessary to evaluate the similarity between two keyphrase graphs that represent them. There can be a (total) KG projection from the query graph to document graph even if the document does not perfectly fit the query. The valuation of this projection will not be maximum. However, there may not be any total projection between the two graphs even though they may be related, and then partial projections between them are necessary. The result of relevance evaluation would be the maximum value of those partial projections.

Definition 15. *Let H is a keyphrase graph of the query q and G is a keyphrase graph of the document d. Semantic similarity between two keyphrase graphs H and G is defined as:* $Rel(H,G) = Max(\{v(\Pi)|\Pi$ *is a partial projection from H' to G,* $H' \leq H)\}$

The problem of finding a partial projection between two keyphrase graphs such that the value of projection is maximized is posed. The process for finding the maximum partial projection between two keyphrase graphs is very complicated. The general way to calculate Rel(H,G) is to start with finding all sub keyphrase graphs of H and then for each sub keyphrase graph H' of H to find every projections from H' to G and return the maximum evaluation value of all projections. Unfortunately, the computation involved in this way may be a NP-complete problem. In this paper, we do not follow the definition of maximum partial projection in a mathematical way as well as find the optimal solution.

Fig. 4 and 5 shows a document graph and the best projection from a query with the relevance ratio of 53.7%.

TITLE: **Frontend Engineer - Core**
- 5+ years experience building highly-scalable interactive web applications (e-commerce preferred)
- Expert knowledge of JavaScript
- Strong knowledge of HTML5 & CSS3.
- Knowledge of Angular & Reat are definitely a plus
- Strong familiarity of server-side web technologies such as Nodejs, Python, Ruby, JSP, etc.
- Experience writing object-oriented code, especially in Javascript
- Experience working with database technologies
- Experience working in a test-driven development
- Familiar with Agile methodologies
- Experience working with open source technologies is required and contribution to open source systems is a plus

Fig. 4. An excerpt from a job posting (document)

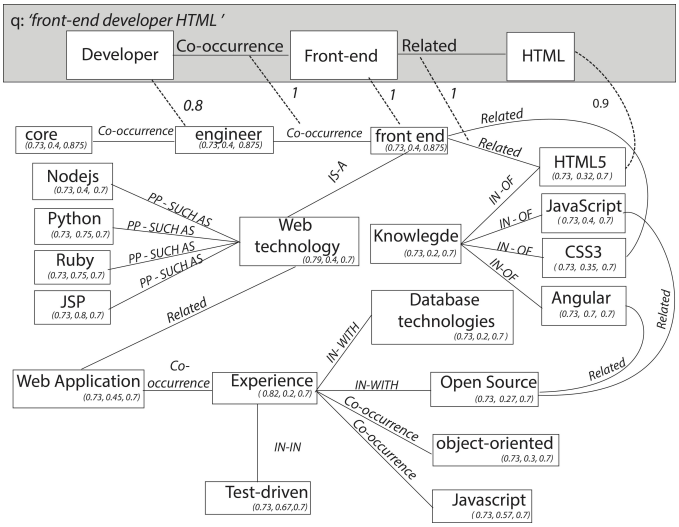

Fig. 5. An excerpt of keyphrase graph corresponding to above document and an example of keyphrase graph matching

Algorithm 2 Find every sub keyphrase graph of KG

Function `findAllSubKG(`*subkg, kg, minSize*`)`
- **input :** $subkg$ the collection of all sub keyphrase graph - passed by reference
- **input :** kg the orginal keyphrase graph - passed by value
- **input :** $minSize$ the minimum number of keyphrase in a sub keyphrase grap - default to 1
- **Result:** All keyphrase graph of kg will be stored in $subkg$

if `Count( Vertices(`kg`)) >` $minSize$ **then**
- **foreach** *keyphrase* k *in* `Vertices(`kg`)` **where** k *has no relation* **do**
  - $tmp \leftarrow kg$
  - tmp`.RemoveKeyphrase(`k`)`
  - $subkg \leftarrow subkg \cap \{tmp\}$
  - `findAllSubKG(`*subkg, tmp, minSize*`)`
- **end**
- **foreach** *relation* r *in* `Relations(`kg`)` **do**
  - $tmp \leftarrow kg$
  - tmp`.RemoveKeyphrase(`k`)`
  - $subkg \leftarrow subkg \cap \{tmp\}$
  - `findAllSubKG(`*subkg, tmp, minSize*`)`
- **end**

end

B. Semantic Search Algorithm

With all the similarity measurement defined, the next ingredient for the semantic search system would be the algorithms to effectively calculate all those measurement. First we have to find all sub kepyrase graph of the query keyphrase graph. Since query keyphrase graphs are usually small, about 6 vertices or less, we can exhaustively search for all sub KG using algorithm 2

Exhaustively search for all projections between two keyphrase graph however is not a trivial task, so we opted for a heuristic approach as presented in algorithm 5.

VI. Application and Experiment

This section discuss the hand-on experience in building a semantic document retrieval system with SDB framework. We present a few most notable experiment systems we have built, especially the newest - it job posting retrieval system and how we evaluate its retrieval performance.

The section also discuss the experiment and evaluation setup for our SDB framework. The contemporary trend is evaluating each key tasks in the systems using standardized dataset. This line of evaluation would allow for easier comparison between approaches as well as help pointing weakpoints for future refinements. However, this paper want to strive for

Algorithm 3 Evaluate all projections from keyphrase graph h to larger keyphrase graph g

input : keyphrase graph h
input : a smaller keyphrase graph g
output: The maximum relevance value of all projection from g to a subKG of h

```
isolateProjection ← Maximum weight matching from all isolated keyphrase in g to isolated keyphrase in h
result ← 0
matchComplete ← TRUE
foreach relation rh in h do
    foreach relation rg in g where β(rh, rg) > 0 do
        /* We consider every keyphrase k'
           with whom k have relationship r */
        if α(rh.sourcce, rg.source) = 0 or
         α(rh.destination, rg.destination) = 0 then
            continue /* source and destination
               keyphrase of rh and rg have no
               relevance */
        end
        projection ← Empty matching
        projection (rh) ← rg
        projection (rh.source) ← rg.source
        projection (rh.destination) ← rg.destination
        Q ← Empty Queue
        Q.enQueue(rg.source)
        Q.enQueue(rg.destination)
        while Q is not Empty do
            kg ← Q.deQueue()
            kh ← projection (kh)
            hNeighbors ← { adjacent keyphrase vertices i
             from kh in h where projection (i) is null }
            gNeighbors ← { adjacent keyphrase vertices i
             from kg in g where projection (i) is null }
            if gNeighbors not = ∅ then
                matched ← the maximum weight matching
                 from gNeighbors to hNeighbors
                if matchednot = null then
                    projection ← matched ∪ projection
                    Q.enQueue (gNeighbors)
                end
                else
                    matchComplete ← FALSE
                    break
                end
            end
        end
        if matchComplete not =FALSE then
            projection ← matched ∪ isolateProjection
            result = max (result, evaluate (projection))
        end
    end
end
return result
```

real-world applications with extrinsicly evaluating. Therefore an application-specific dataset that can simulate real-world documents and queries may be a better setup.

A. Meet ITJPRS: An IT Job Posting Retrieval System

The prime motivation for this system is to help job-seekers, people who are interested in another career opportunity, in searching for the most relevant job description on various job posting websites.

We target the Information Technology job posting domain for this systems due to the sheer amount of job postings available online, as well as a large number of potential users. Especially in Viet Nam, where the Tech Industry is fast growing and oversee a high job switching rate.

The special nature of job postings also provides interesting challenges for retrieval systems. Most job postings are very brief but contain a lot of keywords and catchphrases. They also do not conform to formal grammar and as our experiment will later show, traditional text retrieval systems have a lot of struggle with them.

While building the system as well as the experiment settings, we focus solely on the job's description. Special information about employment conditions, like salary, benefits, work hours, etc., if ever mentioned in the job posting, are not given any special consideration.

Our userbase demographic survey reveals three groups of job-seekers. The first group includes people interested in information technology domain but haven't completed or even received any training. They are not really looking for new position, and only want to take a peek and the available opportunities in this field and thus they do not have any particular information need and tend to throw trending keywords at the retrieval system. While our system may serve this group of users, we do not really focus efforts on their usecase.

The second group of users are people looking for their first job in the field. This group have a rough sketch of their information need but struggle to find the best keywords to describe it. While we provided some filters and suggestions to help them narrow down the retrieved results. We don't evaluate the retrieval performance in their usecase.

Our focal group of users are experienced job-seekers who have worked for at least a year or more than one jobs in Information Technology industry. This group can describe their information need effectively both in natural language as well as through selected keywords. They are the dominant demographic group in our assessors forces, helped us forming the experiment scenario and evaluated our system performance.

B. Design SDB for ITJPRS

The IT Job-posting retrieval system are built using SDB framework, the blue print design for this system can found in Fig. 6. Some important steps are discussed in detail below:

1) Building IT Jobs knowledge base: The first step in building a knowledge base in CK-ONTO formalism is to collect the set of keyphrases in the domain. Our starting point would be other reputable open-access resources. Many lexical

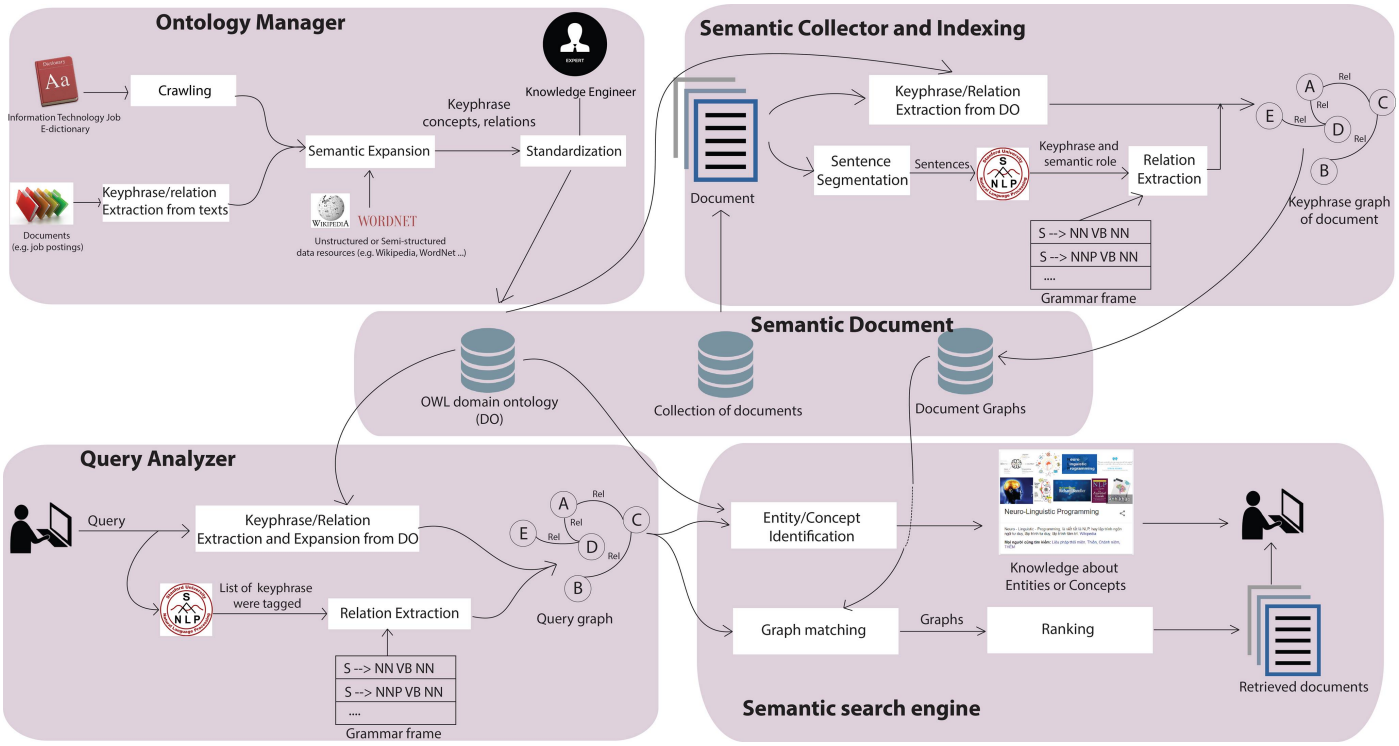

Fig. 6. Architecture of the IT Job posting retrieval system

resources provide a list of keyphrase in a domain along with some manner or categorization for those keyphrase.

Another source we used was the website whatis.techtarget.com, which provides an extensive and up-to-date list of 'terms' in information technology domain, organized in a hierarchy of 'topics'.

Another source of keyphrases is the name of softwares and other Information Technology toolkits deployed in enterprise environment. We notice that a considerable amount of job postings often require hands-on experience with a foray of tools and softwares, many of which are yet to be registered as a term in other lexical resources. Therefore, we also included the list of softwares we found on trustradius.com, a review aggregate service with a hefty list of softwares organized into many categories.

We then cross-referenced with Wikipedia to acquire the definitions of terms as well as the relations among terms. All the data from those sources was indispensible to our knowledge engineers when building the knowledge base.

2) Building weighted keyphrase graphs to represent job posting: Building a keyphrase graph to represent a job posting follows the general framework described in Section IV-C1. However, the challenging characteristics of job postings would dictate some special attention when connecting keyphase vertices in the graph and assigning weighs for those edges.

To determine syntactical relationships among keyphrases that appear in the same sentence, we perform POS tagging using the Stanford Parser on that sentence with special care to make sure the Pos-Tagger won't break keyphrases down into multiple normal words. Then we devise a list of syntactical rules to determine the relationships between tagged keyphrases. The nodes and edges will be assigned weights using the same formulas presented in Section IV-C1 with the parameter c in 'term frequency' formula set to 1.

We allocated each edge of the graph a weight coming from its frequency information in the whole document repository. It is assumed that if two keyphrase vertices connected by the same relationship occur in a lot of document graphs then we can safely say that this relationship between them should be strong and a large weight should be assigned to the corresponding edge. Given an edge e in the document graph $docKG(d)$ connects two keyphrases k_1, k_2, e is labeled with a relation symbol r, and thus can denoted as $e = (k_1, r, k_2)$. The example formula for calculating the weight of e is given below:

$$w(e) = \frac{tf(e, D)}{Max(\{tf(e', D)|e' \in KG(D)\})} \quad (6)$$

in which, $tf(e, D)$ is the number of documents in D where its keyphrase graph contains e (thus it is a "global" statistic) and $KG(D)$ is the set of keyphrase graphs that each represents a document in D.

C. Evaluating Job Posting Retrieval Performance

1) Experiment setup: We evaluate our system performance in ad hoc search, the most standard retrieval task, in which a system aims to collect a list of job-postings that are relevant to an arbitrary user's information need. Our model users are experienced job-seekers in Information Technology domain, who frequently look for and read job-postings, and thus are quite familiar with keyphrases in the domain.

A typical test collection for text retrieval system consists of 3 parts: (1) a collection of documents, (2) a set of sample queries and (3) the golden standard relevance assessment that states which document is relevant to which query by a group of human accessory experienced in the domain.

2) Documents: For our document collection, we collected job postings on the website stackoverflow.com[1] during three months of summer, 2018. To assert the high quality of collected documents, we only download job-postings that filled in all following fields: title, job overview, company's name, expected salary, technology, job descriptions, benefit and company overview. A total of 2500 job postings was downloaded in HTML format, we then parsed them into plain texts for the retrieval system to process.

3) Topics: We format our sample queries in a similar fashion to TREC "topics". Each topic represents an information need from users and contains a title field and a narrative field. The title contains between one to five keyphrases that best describe the information need. This is the data that was given to the system as a search query. The narrative field is a natural language statement that gives a concise description of the information need and potential relevant job-postings. This field is used to co-ordinate our assessors, making sure all assessors have the same understanding of each topic to judge its relevance to documents.

To make sure the information need in our experiment reflect real world situations, half of our topics was inspired by suggestions from popular search engines. Our assessors would input one keyphrase into the search engine then scan the suggestions for valid job-seeker's need and build a topic around them. Since most search engines will suggest queries as you type based on previous search request history they received, those suggestions give an insight to real queries submitted by a broad user-base. Around 50 topics were built in this way. Another 50 topics were synthesized by our accessors, based on their own experience in job seeking as well as in coporate recruiting process.

4) Relevance assessing: The relevance assessments are the combining factor that turn documents and topics into a test collection. We told our assessors to assume that they have the information need described in the topic and they are 'between jobs'. If there is a reasonable chance they would apply for the opening described in the job posting, that job posting is to be marked as 'relevant', otherwise, that posting is to be marked as 'irrelevant'. Assessors are also told to look at job title, overview and description only, information like company's name, benefits and working conditions are hidden from assessors.

It is a well known fact that the relevance is highly subjective, the assessments may vary not only across assessors but also vary for the same assessor across different times. To circumvent this, we schedule our assessors to work only on a subset of topics that he/she feels most comfortable with. We make sure those subsets overlap so that each topic-document pair is assessed by at least five assessors. To avoid assessing fatigue and to ensure that documents are assessed independently from each others, assessors are told to work on a batch of 500 documents at a time. They would assess one topic across 500 documents, then go on to the next topic. Only when they complete their set of topics that they comeback to judge the first topic across another 500 documents.

Working in this manner, it took our assessors about six months to complete their work. We then combine assessors' opinion in a majoritarianism manner. A document is relevant to a query only if more than half the number of assessors agree it is relevant.

5) Evaluation results and discussion: The classic recall and precision index are used to evaluate the effectiveness of the our document retrieval system. We compared our system against Lucene, a traditional search engine that has been long established as the baseline for information retrieval. The verbatim installation of Lucene however, got abysmal performance with only single digit precision overall as seen in Table VI. This is owing the characteristics of job postings we mentioned before. While some jobs may have vastly different job descriptions. In Lucene's eye, a good response for the query 'front-end web developer' could be job-postings for 'junior mobile developer' or 'senior game developer' or anything contain the term 'develope'.

To dewindle this challenge, we also run Lucene with our customized tokenizer to make sure that Lucene can recognize keyphrases in the domain. This 'Lucene + CK-tokenizer' method achieved a drastical improvement in precision while maintained a decent recall rate and would serve as the new baseline for our comparisions.

Another improvement that can be done on behalf of Lucene is to perform query expansion using our knowledge base before passing the keyphrase sets to Lucene. We experimented to find out the best limit for the expansion, starting off with keyphrases that have 'equivalence' relationships with the original query, then keep adding keyphrases while watching the performance record. It is observed that F1-score would peak out with the inclusion of both 'equivalence' keyphrases and 'hyponymy' keyphrases, including evermore keyphrases would just diminish the precision. This 'Lucence + CKQe' experiment helps evaluating the potent of our CK-ONTO model in boosting the performance of traditional simple baseline retrieval method.

For our method, we performed one extra experiment besides the final method presented in this article. We created an SDB system that represents job-postings using the form of keyphrase graph with only semantic relation edges. That means even if two keyphrases appear in the same sentence in the document, they will not be linked by an edge if their relationships cannot be found in the knowledge base. This 'SDB+docKG' experiment helps attesting the potential of combining semantic relationships and syntatical relationships.

TABLE VI. PERFORMANCE OF JOB SEEKING SYSTEM (IN PERCENTAGE)

Model	Precision	Recall	F-score
SDB + fulldocKG	77.1	77.8	77.4
SDB + docKG	70.3	71.9	71.1
Lucene	8.7	98.5	16.0
Lucene + CKTokenizer	43.7	58.5	50.0
Lucene+ CKQe	45.1	70.3	54.9

[1] stackoverflow.com/jobs

TABLE VII. PROTOTYPE KNOWLEDGE BASE METRICS

statistic	Computer Science KB	IT-Jobs KB	Labor & Employment KB
keyphrases	15968	6755	2764
concepts	10946	4356	1523
keyphrase relationships	192089	40757	20347

One can observe that our models can maintain better performance compare to two other models. While the Lucence combine with query expansion model can provide quite high recall, it still falls short in precision and F measurements.

D. Others Applications Facilitated by SDB Framework

Throughout the development of SDB, we have implemented and tested it in three document retrieval systems:

- *The learning resource repository management system* [20] (educational assistance program) in the University of Information Technology HCM City, Vietnam. This system employs our first version of CK-ONTO to provide semantic search on a repository of English documents (mostly textbooks) in Computer Science domain.
- *The Vietnamese online news aggregating system* [24] in Labor and Employment domain alongside Public Investment and Foreign Investment domain. This system periodically aggregates news articles and provides semantic search capability. It was used Binh Duong Department of Information and Communications, Viet Nam.

Corresponding to those two systems, we built two prototype knowledge bases in CK-ONTO model: Computer Science KB, and Labor & Employment KB. The size of those knowledge bases are described in Table VII.

The prebuilt knowledge bases was used when extracting keyphrases from documents in order to help with the disambiguation of terms. After that, they also helped with determining the relations between keyphrases and forming a graph based representation of documents, which will be used in various retrieval tasks later on. Also, knowledge bases was used when processing queries that users put into the systems. They enable query expansion to include more relevance keyphrases into the search, and support interactive search by suggesting user with potential keyphrases. And finally, the most important use of knowledge base in document retrieval would be to estimate semantic similarity between keyphrases and between concepts. These semantic similarity metrics would be the basis for determining the relevance between document and query or between documents, which is the essence of semantic search.

VII. CONCLUSIONS

In this paper, we proposed a method for designing a kind of document retrieval systems, called Semantic Document Base Systems (SDBS). A semantic document base system is distinguished from a traditional document retrieval system by its capability of semantic search on a content-based indexed document repository in a specific domain.

The Classed Keyphrase based Ontology (CK-ONTO in short) was made to capture domain knowledge and semantics that can be used to understand queries and documents, and to evaluate semantic similarity. CK-ONTO contains keyphrases of relative importance in the domain, which is the building block for other components. Another main component is a set of concepts with definitional structures to provide an unambiguous meaning of the concept in the domain. In addition to being a knowledge model of concepts and their relations, CK-ONTO also resembles a lexical model, in that it groups keyphrases together based on their meaning similarity and labels the semantic relations among keyphrases. Finally, there is a set of rules for constraint checking and inferring relation between two kephrases, between a keyphrase and a class, and between two classes. The structure of CK-ONTO is general and can be easily extended to fit different knowledge domains as well as different kind of applications.

To model document content and to design measures along with algorithms for evaluating the semantic relevance between a query and documents, keyphrase graph - based models and weighting schemes were proposed. Each document can be represented by a compact graph of keyphrases in which keyphrases are connected to each other by semantic relationships. A distinctive feature of weighted keyphrase graphs: they allow to represent semantic and structural links between keyphrases and measure the importance of keyphrases along with the strength of relationships whereas poor representation models cannot. Relevance evaluation between the target query and documents is done by calculating the semantic similarity between two keyphrase graphs that represent them. We defined a KG-projection between two KGs along with necessary formulas and algorithms to evaluate the similarity between them.

The proposed design method has been applied in a foray of applications, the latest of which is IT Job-posting retrieval system. The designing process of that system was presented in depth along side with experimental setup and dataset preparing and evaluating process.

As future work, we are planning on building a public gateway to provide access to our aforementioned knowledge bases. Moreover, we are revising said knowledge bases as to enable linking data between our knowledge bases and others knowledge sources on Semantic Web. Finally, we are resolved to incrementally update the CK-ONTO model and periodically release new versions. A few elements of CK- ONTO that still in need of additional work are the inferring rule and a formal reasoning engine to go along with it. Besides tools to help knowledge engineer through automation of some tasks are in dire need. Moreover, the rich choices of available weighting schemes and techniques also raise a challenge of how to incorporate them together and fully explore the potential of keyphrase graphs for better retrieval performance. And finally, the algorithms to calculate similarity between keyphrase graphs can also use some improvements.

ACKNOWLEDGMENTS

This research is funded by Vietnam National University HoChiMinh City (VNU-HCM) under grant number C2018-26-08

REFERENCES

[1] Bizer, Christian, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. "DBpedia-A crystallization point for the Web of Data." Web Semantics: science, services and agents on the world wide web 7, no. 3 (2009): 154-165.

[2] Ngo, Quoc Hung, Nhien-An Le-Khac, and Tahar Kechadi. "Ontology Based Approach for Precision Agriculture." In International Conference on Multi-disciplinary Trends in Artificial Intelligence, pp. 175-186. Springer, Cham, 2018.

[3] Salatino, Angelo A., Thiviyan Thanapalasingam, Andrea Mannocci, Francesco Osborne, and Enrico Motta. "The computer science ontology: a large-scale taxonomy of research areas." In International Semantic Web Conference, pp. 187-205. Springer, Cham, 2018.

[4] ThanhThuong T. Huynh, Nhon V. Do, TruongAn PhamNguyen, and NgocHan T. Tran. "A Semantic Document Retrieval System with Semantic Search Technique Based on Knowledge Base and Graph Representation." In SoMeT, pp. 870-882. 2018.

[5] 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.

[6] 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.

[7] 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.

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

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

[10] 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.

[11] Hadas Raviv, Oren Kurland, and David Carmel. 2016. 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.

[12] Faezeh Ennsan, Ebrahim Bagheri, *Document Retrieval Model Through Semantic Linking*, ACM, WSDM, 2017.

[13] 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, Japan [doi¿10.1145/3077136.3080768]

[14] S S Sonawane, P A Kulkarni, *Graph based Representation and Analysis of Text Document: A Survey of Techniques*, International Journal of Computer Applications 96(19):1-8, 2014.

[15] Faguo Zhou, Fan Zhang and Bingru Yang, *Graph-based text representation model and its realization*, In Natural Language Proceeding and knowledge Engineering (NLP-KE), 2010, pp 1-8.

[16] Francois Rousseau, Michalis Vazigiannis, *Graph-of-word and TW-IDF: New Approach to Ad Hoc IR*, Proceedings of the 22nd ACM international conference on Conference on information and knowledge management 2013, pp. 59-68.

[17] 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.

[18] 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.

[19] Yuan Ni, Qiong Kai Xu, Feng Cao, *Semantic Documents Relatedness using Concept graph representation*, ACM, WSDM, 2016.

[20] Nhon V. Do, ThanhThuong T. Huynh, and TruongAn PhamNguyen. "Semantic representation and search techniques for document retrieval systems." In Asian Conference on Intelligent Information and Database Systems, pp. 476-486. Springer, Berlin, Heidelberg, 2013.

[21] Gruber, Tom. Ontology. springer US, 2009.

[22] M. Uschold, M. King, S. Moralee, and Y. Zorgios, The Enterprise Ontology, The Knowledge Engineering Review, 13(1):31-89, 1998.

[23] Chenyan Xiong, Jamie Callan, Tie-Yan Liu, *Word-Entity Duet Representations for Document Ranking*, SIGIR'17, August 7-11, 2017, Shinjuku, Tokyo, Japan, ACM 2017.

[24] Nhon V. Do, Vu Lam Han, and Trung Le Bao. "News Aggregating System Supporting Semantic Processing Based on Ontology." In Knowledge and Systems Engineering, pp. 285-297. Springer, Cham, 2014.