Received 13 June 2015
Revised 13 June 2015
Accepted 24 June 2015

GeTFIRST: ontology-based keyword search towards semantic disambiguation

Hoang-Minh Nguyen, Hong-Quang Nguyen,
Khoi-Nguyen Tran and Xuan-Vinh Vo
School of Computer Science and Engineering,
International University – VNUHCM, Ho Chi Minh City, Viet Nam

Abstract

Purpose – This paper aims to improve the semantic-disambiguation capability of an information-retrieval system by taking advantages of a well-crafted classification tree. The unstructured nature and sheer volume of information accessible over networks have made it drastically difficult for users to seek relevant information. Many information-retrieval methods have been developed to address this problem, and keyword-based approach is amongst the most common approach. Such an approach is often inadequate to cope with the conceptualization associated with user needs and contents. This brings about the problem of semantic ambiguation that refers to the disagreement in meaning of terms between involving parties of a communication due to polysemy, leading to increased complexity and lesser accuracy in information integration, migration, retrieval and other related activities.

Design/methodology/approach – A novel ontology-based search approach, named GeTFIRST (short for *G*raph-*e*mbedded *T*ree *F*ostering *I*nformation *R*etrieval *Sys**T*em), is proposed to disambiguate keywords semantically. The contribution is twofold. First, a search strategy is proposed to prune irrelevant concepts for accuracy improvement using our Graph-embedded Tree (GeT)-based ontology. Second, a path-based ranking algorithm is proposed to incorporate and reward the content specificity.

Findings – An empirical evaluation was performed on United States Patent And Trademark Office (USPTO) patent datasets to compare our approach with full-text patent search approaches. The results showed that GeTFIRST handled the ambiguous keywords with higher keyword-disambiguation accuracy than traditional search approaches.

Originality/value – The search approach of this paper copes with the semantic ambiguation by using our proposed GeT-based ontology and a path-based ranking algorithm.

Keywords Web search and information extraction, Metadata and ontologies, Web data integration

Paper type Research paper

1. Introduction

Most information retrieval (IR) systems would face the problem of semantic ambiguation that emerged in 1980s (Kent, 1989) and has been intensively investigated in the field of database (Saltor *et al.*, 1995; Hull, 1997; Hudson *et al.*, 1994; Ventrone, 1991), information systems (Kashyap and Sheth, 1997), information and schema integration (Hakimpour and Geppert, 2001; Lahr and Barr, 2011),

This research is funded by International University VNUHCM under the grant number SV2014-IT-01.

International Journal of Web Information Systems
Vol. 11 No. 4, 2015
pp. 442-467

1744-0084
DOI 10.1108/IJWIS-06-2015-0019

Geographic Information System (Bhattacharjee and Ghosh, 2014; Ying *et al.*, 2013) and Decision Support Systems (Cantrell, 2013). Semantic ambiguation, as an instance of semantic heterogeneity (Halevy, 2005; Sheth and Larson, 1990), refers to the explicit or implicit disagreement in the meaning of terms used by parties in communication. It is a root cause of complexity and inaccuracy in information integration, migration and retrieval.

A common manifestation of semantic ambiguation is in irrelevant results returned from typical search engines (Google Google Inc., 2015; Bing Yahoo! Search Engine Company, 2015; Yahoo! Microsoft Corporation, 2015) which could be demonstrated through the following scenario. Suppose that three users *A*, *B* and *C* initiate a search for the same keyword "mouse"; however, each user implicitly thinks of this keyword with different meanings: *A* wants to find a "computer mouse" for his new laptop, *B* wants to find information about some kind of "mouse" for her mammal analysis research, while *C* wants to find "mouse" characters in Walt Disney cartoons for her entertainment. Because search engines could not identify and match the requested meaning of a user with their indices, disagreement will arise: the search engines would return a mix of results with different meanings that partially satisfies the request of each user, as illustrated by Google search engine in Figure 1, or a set of results that is completely irrelevant to the request.

Semantic ambiguation lies at the core of many problems in information integration, migration and retrieval. In enterprise information integration, semantic ambiguation forms a primary challenge in accessing and analyzing data located on different data sources with different schemas. In the field of Deep Web indexing and querying, the wide variety in design of forms to access Deep Web data, representing an ambiguation in semantics, is the key contributor to the complexity of the field (Halevy, 2005).

Semantic disambiguation – the act of resolving semantic ambiguation – involves two activities:

Notes: (i) Mammal; (ii) computer device; (iii) cartoon character

Figure 1. Semantic ambiguation in IR – the keyword "mouse" results in three different concepts

(1) detecting a relative position between meaning of terms used by parties in the communication (i.e. detecting whether by the same term, the parties implicitly refer to the same, similar or completely different meanings); and

(2) reconciling these positions to solve the disagreement.

This relative position could be established through matching the meanings of the terms used by different users with a common domain-specific ontology – a knowledge representation showing concepts, organized into domains, and relationships between them. Therefore, defining and constructing a domain-specific ontology as a reference is the principle for solving semantic disagreement.

From the above observations, we designed a Graph-embedded Tree (GeT)-based ontology – a domain-specific ontology – to address such semantic disagreement (Vinh *et al.*, 2014). Even though the GeT-based ontology enables to capture better semantic meanings, new challenges emerged as concepts should be retrieved in more semantically enhanced manner. The task of retrieving appropriate candidate concepts is important because it is primary step toward many other tasks for managing concepts, such as concepts ranking. However, such a task could not be easily done using traditional keyword-based approaches. The keyword-based approaches often rely on inverted indices by extracting terms from documents whereas the concepts in an ontology are structured in different manner. On the other hand, despite having more similar patterns in term of concepts' retrieval, existing ontology-based retrieval approaches often either restrict users within limited query expressions or overlook the semantic ambiguation to some certain extent. In this paper, we address the severity of the semantic ambiguation by proposing a new ontology-based search approach, named GeTFIRST (short for *G*raph-*e*mbedded *T*ree *F*ostering *I*nformation *R*etrieval *Sys**T*em). Our contribution is twofold. First, we propose a novel search strategy to reduce a number of polysemous keywords by navigating through the GeT-based ontology. Second, we propose a novel path-based ranking algorithm including PageRank algorithm (Haveliwala, 1999; Page *et al.*, 1999) that supplements GeTFIRST to yield better concept disambiguation ability. Our method is evaluated using a dataset from US Patent Classification System (USPC) (US Patent and T Office, 2015); the experimental results confirmed advantages in disambiguation capability of GeTFIRST comparing to existing patent search methods.

This paper is structured as follows. Section 2 opens a discussion on the related ontology-based IR approach toward semantic disambiguation. Sections 3 revisits the construction of GeT data structure for ontology construction. Section 4 presents an overall architecture of GeTFIRST, consisting of GeT-based ontology and a light-weight search engine. Section 5 reviews a set of stages for constructing a GeT-based ontology. Our search strategy is proposed in Section 6, including a path-ranking algorithm. Section 7 presents our experiments, followed by Section 8, which concludes the paper with a summary of our proposed approach and future work.

2. Related work

When modeling an ontology-based IR system, an important objective is to utilize knowledge from a domain-specific ontology to obtain more comprehensive answers on a semantic basis. Most existing IR systems working in conjunction with an

ontology could be largely characterized in two paradigms: keyword-based or ontology-based.

One of the mainstream is concerned with improving existing keyword-based IR using a pre-built ontology like KIMO (Kiryakov *et al.*, 2004; Castells *et al.*, 2007). Such a semantic annotation system relies on traditional full-text search engines for indexing, retrieving and ranking tasks. Even though there were evidences of better disambiguation capability from the enhanced semantic version, this approach requires a plural of extra layers for integration between an ontology and a context-unawareness IR.

In the study by Mayfield and Finin (2003), an innovative inference approach combines a knowledge base into an existing keyword-based search engine to improve relevant results of text retrieval. The inference engine in OWLIR is useful for query expansion using rules created from a manual task of collecting non-redundant semantic markup. However, OWLIR expects its users to perform semantic search with RDF triple wildcards, trading its query expressiveness for more semantic controls.

The ontology-based IR approach refers to an IR system that directly exploits the semantic graph, among which takes the advantages of common lexical resources like Wordnet (Volk *et al.*, 2003; Klapaftis and Manandhar, 2005; Bouramoul *et al.*, 2012) and Thesaurus (Wielinga *et al.*, 2001; Jarmasz and Szpakowicz, 2001). These are particularly representatives of the use of explicit conceptual descriptions. They promise a better semantic-enhanced manner in acquiring candidate concepts. However, as most existing work revolving around generic dictionaries which are willow and sparse conceptualizations, it often fails to exploit more sophisticated semantic relationships between concepts.

In fact, our approach to puzzle out semantic heterogeneity is close to the proposed approaches in the studies by Ma and Tian (2015), Mukhopadhyay *et al.* (2007). In the study by Mukhopadhyay *et al.* (2007), a comprehensive search engine was proposed that could improve speed significantly by taking out irrelevant domains of ambiguous concepts, but it restricted users for a given query syntax. Such approach is often prone to limited expressive power of query languages in comparison to ontology query languages. Furthermore, ranking functions in both approaches were not explained explicitly. Our approach complements (Mukhopadhyay *et al.*, 2007) with a search strategy specifically designed for obtaining and ranking relevant results without much efforts on user restrictions.

Swoogle (Ding *et al.*, 2004) is another interesting IR system for semantic Web search's purposes. Nevertheless, as Swoogle focuses more on the aspect of retrieving semantic meaning at the document level, it does not address individual polysemous word which could cause misleading during its ranking phase.

The existing approaches have their merits and caveats. Building an existing search engine would deliver more practical results, but impose more challenges in terms of scalability, as these approach are mainly dependent on the testing objects. Furthermore, such a task involves effort intensity as the heterogeneity problem should be handled across different tools. On the other hand, the other approach requires users' efforts to understand the semantic nature of the input keywords before the actual search could be carried out; it is often frustrated as users might feel reluctant to adapt new machinery syntax other than natural languages. In this paper, a novel retrieval approach is proposed to utilize the GeT-based ontology, without a large burden of integration. With

our proposed method, we aim to reach the optimal balance between accuracy, flexibility and effort intensity.

3. GeT data structure

Because GeT (Vinh *et al.*, 2014) lies at the heart of our retrieval approach, it should be revisited in this section. GeT is an integrated structure between a hierarchy and a graph, motivated by a wide variety of hierarchical Classification Schemes established by domain experts, a limitation in expressive capability of a hierarchical structure and difficulty in modeling explicit domains in a graph structure. By integrating a graph into a hierarchy, GeT aims to retain the hierarchical structure for an explicit domain boundary representation and a natural matching with established Classification Schemes while extending the expressive capability with embedded-graphs of concepts and relationships.

A wide variety of Classification Schemes exists in different fields and domains, sharing the defining characteristics of quality: as a majority of Classification Schemes were manually created and maintained by leading experts in the field, they hold a wealth of highly accurate domain knowledge.

Classification Schemes could be roughly divided into two groups:

(1) generic schemes, providing an overview and classification for a large number of domains from technology to science and business (e.g. Patent Classification Schemes such as US Patent and T Office, 2015; Cooperative Patent Classification (CPC), 2015; Organization, 2015); and

(2) specific schemes, providing descriptions and classification for only a number of related domains in a specific field, e.g. Association for Computing Machinery Computing Classification System (Association for Computing Machinery, 2012).

Most Classification Schemes follow a hierarchical structure (Figure 2), which could be modeled by sets of classification items and subclass relationships as follows:

$$CS = \{I, SR\}$$
$$I = \{i_i \mid i_i \text{ is a classification item in the scheme}\}$$
$$SR = \{(i_1, i_2) \mid i_1, i_2 \in I \text{ and } i_1 \text{ is parent of } i_2\}$$

Additionally, each classification item i_i could possess an optional natural language description and an optional ordered or unordered set of belonged documents.

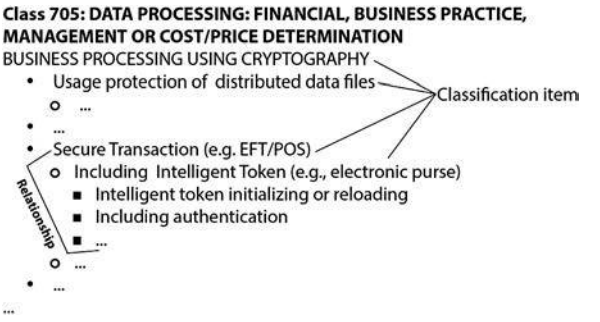

Figure 2.
Sample of Class 705 from USPC

The hierarchical structure consisting of distinct classification items linked by parent–child relationships highly simplifies the representation of domain boundaries and inter-domain relationships; thus, giving it an edge in modeling domain-specific knowledge. However, such an advantage is limited by the extensive use of a natural language in the hierarchy, making semantics of knowledge and intra-domain relationships implicit to computer agents. Therefore, to represent domain-specific knowledge that could be processed by computers, the hierarchical structure must be further extended with more expressive structures. As a result, a graph-based ontology is chosen.

Graph-based ontologies represent knowledge of the field as a graph of concepts in which each concept is a node and relationships between them as edges:

$$\begin{aligned} O &= \{V, E\} \\ V &= \{c_i \mid c_i \textit{ is a concept in the ontology}\} \\ SR &= \{(r,\ i_1,\ i_2) \mid c_1, c_2 \in V \textit{ and } c_1 \textit{ has relation } r \textit{ with } c_2\} \end{aligned}$$

Compared to hierarchical Classification Schemes, a graph-based ontology provides better expressive capability and computer understandability through the explicit representation of concepts and relationships as, respectively, nodes and edges in a graph. However, it lacks the ability to cluster knowledge into explicit domains, which is a natural capability of a hierarchical structure. A graph-based ontology also has limitations in modeling polysemy of terms, as the set of nodes V could not accept duplicated concepts.

Based on strengths and limitations of a hierarchy and a graph, we propose an integrated structure called *Graph-embedded Tree*. In GeT, graphs are used in place of classification items from a hierarchical structure to represent knowledge of specific domains, through concepts and relationships between them. Graphs in GeT are linked to form a hierarchy, representing inter-domain relationships. GeT is defined as follows:

$$\begin{aligned} GT &= \{G, SR\} \\ G &= \{g_i \mid g_i \textit{ is a graph ontology},\ g_i = \{V_i, vr_i, E_i\}\} \\ SR &= \{(g_1, g_2) \mid g_1 \textit{ is the parent class of } g_2\} \\ V &= \{c_i \mid c_i \textit{ is concept in ontology}\} \\ & \quad vr \in V,\ vr \textit{ is the main concept of a graph ontology} \\ E &= \{(r,\ c_1, c_2) \mid c_1, c_2 \in V \textit{ and } c_1 \textit{ has relationship } r \textit{ with } c_2\} \end{aligned}$$

In GeT, concepts are defined as nouns or noun-phrases that represent a concrete of an abstract class of entities in the domain. For example, from the classification items in group "Secure Transaction" of USPC Class 705 (Figure 3), we could identify the following concepts: "Secure Transaction", "EFT", "POS", "Intelligent Token", "Electronic Purse", "Intelligent Token Initializing", "Intelligent Token Reloading" and "Authentication".

The GeT concepts are linked by relationships of different types. A relationship type describes semantics of the relationship between involving concepts and a set of additional attributes. A concrete link between specific concepts in the ontology following a relationship type r is a relationship instance of type r. To reduce the

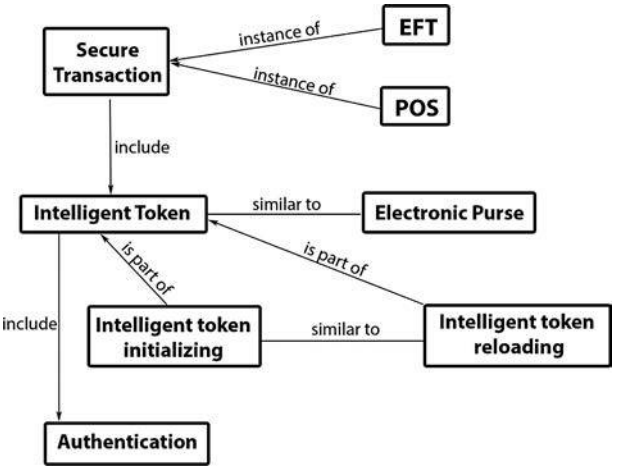

Figure 3.
USPC Class 705's Graph-based Ontology

complexity, GeT only accepts binary relationships, as they are the most common and basic type of relationships, into which any other types of relationship could be translated (Navathe and Elmasri, 2000) (Figure 4).

In GeT, each relationship type has two important attributes:

(1) *Hierarchical* characteristic indicates whether a relationship type forms a hierarchy between involving concepts. For instance, in *A* "is-part-of" *B* relationship, *A* and *B* form a hierarchy with *B* located at higher level, while in *A* "is-equivalent-to" *B*, the concepts *A* and *B* are on the same level.

(2) *Distributable* characteristic indicates whether the relationship type could be distributed between the related concepts of the same level. For example, consider the scenario where we have an equivalence relationship "is-similar-to" between

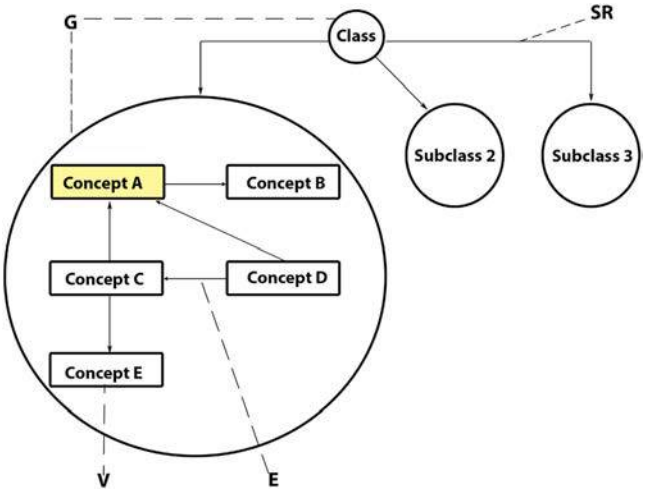

Figure 4.
Structure of the GeT

A and *B*, and a hierarchical relationship "is-a-part-of" between *A* and *C*. Assume that the hierarchical relationship is distributable, we could derive a new "is-a-part-of" relationship instance between *C* and *B* because *A* and *B* are two related concepts of the same level (Figure 5).

In conclusion, GeT integrates the hierarchical structure used by Classification Schemes with the graph structure used by ontologies to combine their merits and negate limitations. By introducing graphs of concepts in place of classification items, GeT makes the semantics of the domain knowledge explicit to computer agents. By introducing hierarchical relationships between graphs of different domains, GeT improves the graph structure with the ability to represent explicit domain boundaries and polysemy. The combination of the explicit domain representation and computer-understandability make GeT a suitable structure for domain-specific ontologies constructed from established Classification Schemes.

4. GeT-based system architecture

In this section, an ontology-based retrieval architecture is modeled for semantic disambiguation, comprising a light-weight IR system, called *GeTFIRST*, adopted a GeT-based ontology to perform semantic search. Intuitively, the process of modeling GeTFIRST could be broken down into two phases as depicted in Figure 6:

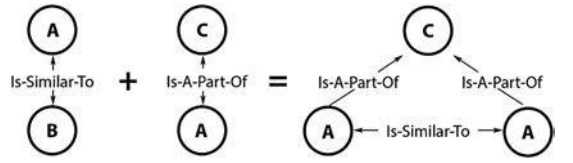

Figure 5.
Distributable characteristic in relationship

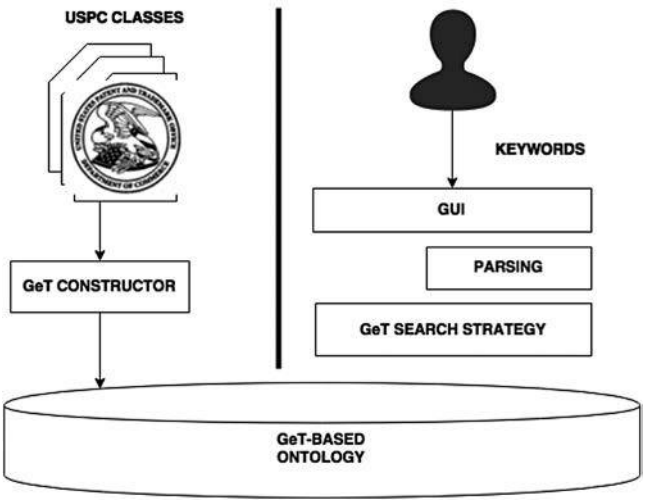

Figure 6.
Overview of GeTFIRST system architecture

(1) In Phase 1, we generally deal with generating appropriated rules to extract information from a specific classification scheme (e.g. USPC Class 705) and populate such data into a directed-graph database, called GeT-based ontology. These steps are to serve for the second phase where the actual retrieval is involved. GeT-based ontology construction will be revisited in Section 5.

(2) In Phase 2, we develop a light-weight GeTFIRST to facilitate the semantic search process. GeTFIRST has to deal with couple issues in semantic domains to achieve better disambiguation ability. In specific, a search strategy of the system will be discussed in Section 6.

5. Phase 1: building GeT-based ontology

A wide variety of established Classification Schemes covering different fields and domains exists around the world. Created and maintained by domain experts, these hierarchical schemes contain wealth amount of accurate domain knowledge. To capture high-quality knowledge and explicit domain representation of Classification Schemes for semantic disambiguation purpose, GeT ontology structure was introduced. In this section, we propose *GeTConstruction* (*GeTC*) – a method to construct GeT-based domain-specific ontologies from established Classification Schemes.

A principle underlying GeTC is the utilization of relationship types as a building block for the resulting ontology. Instead of a complete schema for mapping with input data and constructing an ontology, GeTC only requires the definition of relationship types that an ontology author would like to use in the resulting GeT-based ontology, along with syntactical forms that these types would assume in the input data. From these syntactical forms and patterns, concepts and relationship instances could be detected and linked automatically to form a completed domain-specific ontology.

GeTC consists of three stages (Figure 7):

(1) *Relationship Types Construction* stage is responsible for defining relationship types and their syntactical patterns to use in the construction.

(2) *Relationship Instances Construction* stage identifies occurrences of relationship types in the classification scheme to extract concepts and build relationship instances.

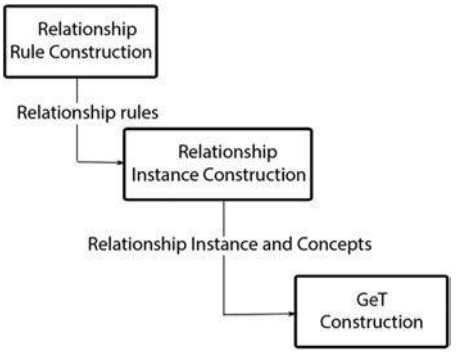

Figure 7.
GeT construction

(3) *GeTC* stage utilizes the detected relationship instances and hierarchical structure of input Classification Scheme to construct a GeT-based domain-specific ontology in a suitable representation format.

GeTC is iterative and flexible. Throughout the construction process, the ontology author could return to the first stage and propose additional relationship types at any moment. These new building blocks could be processed and combined into existing work with limited complexity.

5.1 Relationship types construction

Relationship Types Construction stage consists of three phases (Figure 8) aiming to define a set of relationship types and their syntactical patterns as building block for GeT-based ontology. Starting from the input Classification Scheme, a set of candidate relationship types would be defined, ranked and selected based on predefined thresholds to produce a minimal set of relationship types that is sufficient to construct ontology with less complexity.

The first phase in Relationship Type Construction defines a list of candidate relationship types. As GeTC is based on the GeT structure, only binary relationship types are acceptable. Relationship Type is defined as follows:

$$
\begin{aligned}
R &= \{Name,\ Sem,\ \{(F, FP)\},\ Att\} \\
Name &= identification\ of\ R \\
Sem &= description\ of\ semantic\ of\ R \\
F &= syntactical\ form\ of\ R\ in\ the\ input\ scheme \\
FP &= syntactical\ pattern\ of\ corresponding\ F \\
Att &= set\ of\ attributes\ of\ R
\end{aligned}
$$

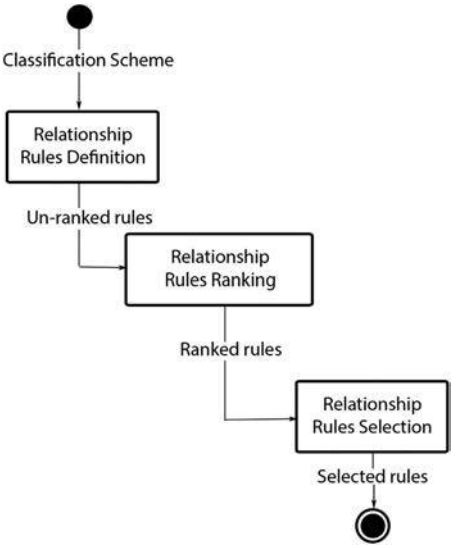

Figure 8.
Relationship types construction stage

Each relationship type is described by a name and a semantic (i.e. what is the meaning of this relationship between terms) in natural language. These descriptions are primarily for references.

Each relationship type could manifest itself in different syntactical forms in the Classification Scheme. A syntactical form describes the structure of phrases that belong to a specific relationship type, along with location of concepts in those phrases. In each phrase, there is one subject concept and at least one object concept. For example, in class 705 of USPC (Figure 2), the "equivalent" relationship type manifests itself in two forms: "Concept A or Concept B" and "Concept A/Concept B", where Concept A is the subject concept. Each syntactical form is formally defined with a syntactical pattern, which could be in a pattern language such as a regular expression. Matching between relationship rules and input data is performed with these patterns.

Finally, each relationship type could possess some special characteristics effecting the ontology construction. Following the GeT structure, relationship types in GeTC also have two key attributes: hierarchical and distributable. Algorithm for the first phase of Relationship Types Construction is given in Algorithm 1:

Algorithm 1 Relationship Types Construction
// Phase 1: Relationship Types Definition
for each relationship type to define **do**
Create an empty *R*
Set name, semantic and attributes
for each synstactical form of relationship type **do**
Create an empty (*F, FP*) tuple
Set the syntactical form to *F*
Set the corresponding pattern to *FP*
end for
end for

Relationship types created at the end of first phase in Relationship Type Construction are only candidates. To reduce the complexity of construction process and the resulting ontology, we need to perform ranking and removing uncommon relationship types to minimize the set of candidates.

The ranking is based on the occurrences of a relationship type in the Classification Scheme. Each relationship type will be matched with each classification item in the input scheme using the defined syntactical pattern. The number of occurrence for each candidate relationship type is kept track to rank the result at the end. Relationship types that fail to satisfy a predefined threshold will be removed. Remaining types form the set of selected relationship types would be used as building blocks to construct the GeT-based ontology. Algorithm 2 for ranking and selection is hereby presented:

Algorithm 2 Relationship Types Construction (cont)
// Phase 2: Relationship Types Ranking
for each *R* in candidate types set **do**
for each Item in Classification Scheme **do**
if found at least one match **then**
Increase the occurrence count for *R*
Continue with next *R* in candidate types set
end if

```
    end for
  end for
  // Phase 3: Relationship Types Selection
  for each R in candidate types set do
    if OccurenceCount of R < threshold by author then
      Remove R from the set
    end if
  end for
```

5.2 Relationship instance construction

The goal of Relationship Instance Construction is detecting occurrences of relationship types in the input Classification Schemes and constructing a set of relationship instances for each type. Relationship instances showing concepts and relationships between them form the graph-part of the resulting GeT-based ontology. Relationship Instance Construction has two phases (Figure 9). In Phase 1, each Relationship Type from Stage 1 is matched with each item in the Classification Scheme to detect and extract matching phrases. In Phase 2, subject and object concepts will be extracted from the detected phrases based on the syntactical patterns of Relationship Type to construct relationship instances. For phrases that contains a single subject and multiple objects, each pair of subject-object will form a distinct relationship instance. Algorithm 3 for this stage is hereby defined as follows:

```
Algorithm 3 Relationship Instances Construction
  // Stage 2: Relationship Instances Construction
  for each Item in the input Classification Scheme do
    for each R in relationship types set do
      Attempt to match each FP with Item
      if a match is found then
        Extract subject and object concepts
        Continue with next R in candidate types set
        for each object concept found in the phrase do
          Create an empty instance
          Add subject, R and object to instance
```

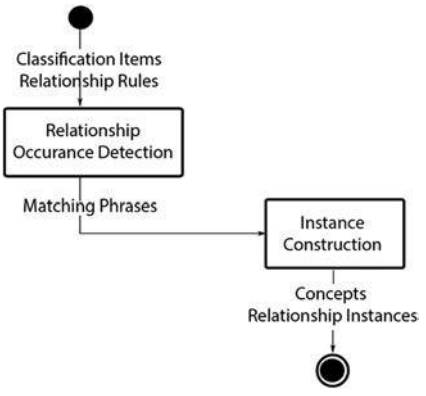

Figure 9.
Relationship instance construction stage

```
                Add instance into instances set
              end for
            end if
          end for
        end for
```

At the end of Stage 2, foundation for the graph-part in resulting GeT-based ontology has been formed with a set of concepts and a set of relationship instances.

5.3 GeT construction

The goal of GeT Construction stage is utilizing the relationship instances and hierarchical structure of the input Classification Scheme to complete the GeT-based ontology following descriptions in Section 3. GeT construction stage is also responsible for distributing distributable relationships.

As stated in GeT's definition, a GeT consists of two main sets: a set of graphs G and a set of subclass relationships between domains SR modeling the hierarchical structure. Inside each graph in G, we have a concepts set V and a relationships set E. The graph-part of the GeT-based ontology is constructed from the defined relationship instances, while hierarchical structure of GeT is constructed based on the structure of the input Classification Scheme.

Graph construction in GeTC follows the "depth-first-inside-first" strategy (Figure 11). "Depth-first" indicates the order of Classification Item selection for the graph construction, starting from deepest item in the scheme. "Inside-first" indicates that intra-domain relationships are constructed and distributed before inter-domain relationships between graphs are constructed. A recursive algorithm is used to realize this construction strategy (Algorithms 4 and 5):

```
Algorithm 4 Base Case of GeT Construction
  if item does not have children then
    Create an empty graph
    for each instance belonging to item do
      if concepts does not exist in nodes set V of G then
        Add concepts to V
      end if
      Create an empty edge
      Add subject, relationship type and object to edge
      Add edge to edges set E
    end for
    Add graph to graphs set G
    for each hierarchical instance belonging to item do
      for each node belonging to Item do
        if current node is not in instance then
          Create an empty edge
          Add subject of instance, relationship type and current node to edge
          Add edge to edges set E
        end if
      end for
    end for
  end if
```

Algorithm 5 Recursive Case of GeT Construction
if Base Case **then**
Execute Algorithm 4
else
for each child item of current item **do**
Construct graph for child item
end for
Construct graph for current item
for each graph of child item **do**
Create a tuple of current graph and graph of child item
Add tuple to subclass relationships set *SR*
end for
end if

At the end of the GeT Construction stage, a GeT-based, domain-specific ontology is created from the input Classification Scheme. The process is completed.

6. Phase 2: GeTFIRST modeling

Traditional search engines often build an inverted index using keywords extracted from the documents. However, in GeT-based ontology, the concepts have no such contents. Therefore, we model a different retrieval process that could be broken down into three sub-phases (Figure 10):

(1) In the first sub-phase, user supplies keywords interactively through a friendly graphical user interface.

(2) During the second sub-phase, a concept parser, which recognizes the concepts of user input using GeT-based ontology, is invoked.

(3) The final sub-phase is mainly concerning how system handle search process. To handle such process, we propose a GeT search strategy that is the core of our GeTFIRST. The strategy begins by accepting outputs from the parser. It is a combination of two algorithms, named *Path Retrieval* and *Path Ranking*. While the former retrieves the corresponding path to each target concepts, the latter performs actual ranking algorithm using supplied paths from Path Retrieval.

6.1 GeT search strategy

GeT search strategy plays an essential role in GeTFIRST's concept retrieval. As a concept has no content to be indexed, an algorithm is developed for indexing by recording paths from the roots to the target concepts. In this manner, a single path could be seen as a virtual document wherein terms are concepts along the path. The only difference is that whereas every document is identified with a different ID, path's

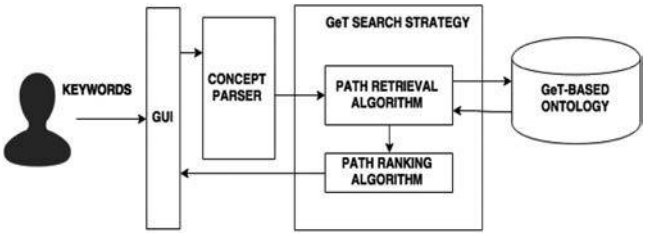

Figure 10.
Phase 2 – GeTFIRST Modeling

identification in GeTFIRST is uniquely determined by concepts within it. For example, if two classes contains the same number of concepts, they are then treated equally, but not identically. Such determination further allows us to fully observe semantic similarity between the concepts. In this section, we decided to convey our strategy using intuitive examples extracted from USPC. In general, the GeT search strategy is modeled based on two assumptions (Figure 11):

(1) Query from users should include at least a matched concept in GeT-based ontology. Otherwise, an empty result is expected.

(2) Path Retrieval algorithm always begins at top-most generic class, also acknowledged as root class. In other words, a root class is a particular class instance of a Classification Scheme. (e.g. class 705 of USPC or G06Q2220 of CPC).

6.1.1 Path retrieval algorithm. A path is fundamental measurement unit in GeTFIRST. In specific, a path P is an infinite tuple of concepts c from root class to a destination. A path should satisfy the following characteristics:

- A generalized form of path P's ID is $(c_1, c_2, \ldots, c_i)$ where $c_i \in V$.
- A concept in a path might have one or a plural of paths leading to it.
- A path has its natural inclusive characteristic. For example, $P_A \subset P_B$ or $P_A \supset P_B$.
- A path is uniquely identified by every concept that it passes through. In Figure 12, a simple case is drawn where the path is identified as $P(c_{root}, c_B, c_C)$.

In Figure 13, more sophisticated problems (hereby defined as multi-paths problems) arise when a concept C could be reachable from either $c_{Root\ 1}$ or $c_{Root\ 2}$ via four different paths. This problem is undoubtedly referred as semantic ambiguation, as a concept existed in various domains. Using traditional keyword-based approaches, the context of the terms are difficult to identify. However, because paths in each domain are stored

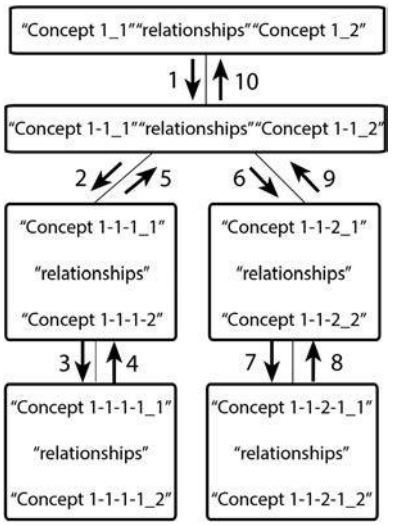

Figure 11.
Depth-first-inside-first construction strategy

semantically in GeT-based ontology, we could fairly tackle the issue of which path should be more related to the user queries.

Both single path and multi-paths are already handled in Path Retrieval, an algorithm of GeT search strategy. The objective of Path Retrieval is to construct a global alphabetically sorted dictionary of concepts and associate each of these concepts with its posting lists. A posting list is a set of path IDs that concept belongs to; the idea of posting list is borrowed from the inverted index algorithm. As stated in Section 3, our concepts are made of nouns or noun-phrases, sorting by alphabet would allows us to speed up searching process. For example, in Figure 13, the posting list of the target concept *C* could be constructed using Algorithm 6:

Algorithm 6 Path Retrieval Algorithm
Input: (*user_concepts*) where *user_concepts* are parsed concepts.
Output: map of alphabetically sorted paths *path_map*(*PI*), where *PI* is a set of path IDs.

```
for each class root_i in set of class schedules S do
  for each concept_j in user_concepts do
    Find path_list from class root_i to concept_j
    for each path_k in path_list do
      Add path_k to path_map
    end for
  end for
end for
```

The resulting posting list for *C* contains:

- P_1's ID is ($c_{Root\ 1}$, c_D, c_C)
- P_2's ID is ($c_{Root\ 1}$, c_B, c_E, c_C)
- P_3's ID is ($c_{Root\ 2}$, c_B, c_E, c_C)
- P_4's ID is ($c_{Root\ 2}$, c_C)

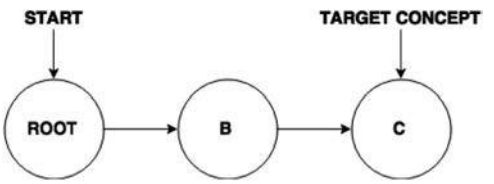

Figure 12.
Single path to a target concept from the Root

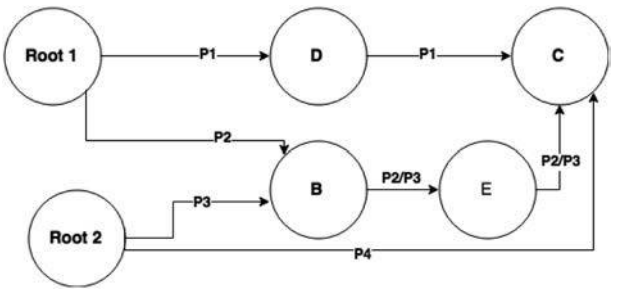

Figure 13.
Multi-paths to the target concept *C*

6.1.2 Path ranking algorithm. As the sheer volume of new concepts increases, there is a need to effectively find and correlate the emerging concepts in more manageable form, which leads to a ranking task based on semantic similarity. Semantic similarity in GeTFIRST follows the idea of the concept counting methods. The similarity between two concepts could then be measured by a length of the path linking the concepts or by concept's position in the taxonomy (Rada *et al.*, 1989; Richardson *et al.*, 1994; Li *et al.*, 2003; Leacock and Chodorow, 1998). Because our GeT data structure is essentially a hierarchical tree, we measure the distance between the concept nodes in GeT with respect to its corresponding root. In addition, the shorter distance result would have higher relevancy as similar to Rada *et al.* (1989). The reason for electing shorter distance is reasonable, particularly in the patent industry in which a classification search (United States Patent And Trademark Office (USPTO), 2015) often requires a manual top-to-bottom scan before a new patent is classified. Following this manner, the top-most relevant concept should be recommended first to the users. In other words, Path Ranking algorithm would rank relevancy based on traveled distance from the root to the target concept using a path map directly fed from the output of Path Retrieval algorithm.

6.1.3 Initial ranking algorithm. Initially, the algorithm scores each path individually. A path's value P (or path's score) is reasonably determined by the total sum of weight factors of any concepts visible to the path. To effectively score a path, we need both formula and procedure in which such formula will be embedded. First, we propose a formula for computing a path's score as follows:

$$P = \sum_{i=0}^{n} S_i \tag{1}$$

Where:

P = is the score for a specific path;
n = is the total number of concepts from the root to the target concepts; and
S_i = is the concept c_i's score where c_i in P.

In GeTFIRST, every concept is treated equally in a path and its weight is set to 1 in Formula 1, the formula could be rewritten as in Formula 2:

$$P = n * S \tag{2}$$

Where:

P = is the score for a specific path;
n = is the total number of concepts from the root to the target concepts; and
S = is the concept c_i's score in where c_i in P.

Using Formula 2, a path is accumulatively calculated to measure the ranking position. Suppose a single term query "c_1, c_2, …, c_n" is given, we propose a procedure of calculation as follows:

- First, we fetch paths map which is created after the execution of Path Retrieval algorithm as an input.

- Second, for each path$_i$ to c_i, we compute corresponding path$_i$'s score.
- Finally, the algorithm stops after c_n is processed.

To capture our algorithm's intuition, we apply it in an empirical patent domain's scenario (Figure 14) where relevant paths leading to concept "Having Program" will be ranked. In this scenario, a partial map from USPC is extracted for observation. The procedure is described in the following two steps:

(1) A paths map to "Having Program" is fed from Path Retrieval including $P_1(c_{c705}, c_{bp}, c_{up}, c_{cp}, c_{hpi})$ and $P_2(c_{c380}, c_{vc}, c_{cp}, c_{hpi})$. Note that concepts' names are shortened to enhance readability.

(2) We calculate path's score using the proposed formula 2. Then $P_1 = 5, P_2 = 4$.

Since $P_1 > P_2$, we choose P_2 as the shorter path that a patent examiner would be likely to check first during the classification search and P_2 will be ranked higher in GeT-FIRST. The drawback of this algorithm is that it overlooks the inclusive nature when multiple concepts are available within a single path. Given a user query "Network Tokens" in Figure 15, our initial procedure will recognize P_1 to "Net-work" and P_2 to "Token" separately without the awareness of its semantic relationship. In GeT-based ontology, "Token" has an outer relationship with "Network"; this inclusive relationship should be further exploited to achieve adequate results for semantically related concepts instead of retrieving it individually. Therefore, we propose an improved semantic version of Path Ranking Algorithm.

6.1.4 Improved ranking algorithm. We present the improved version of the ranking algorithm as in Algorithm 7:

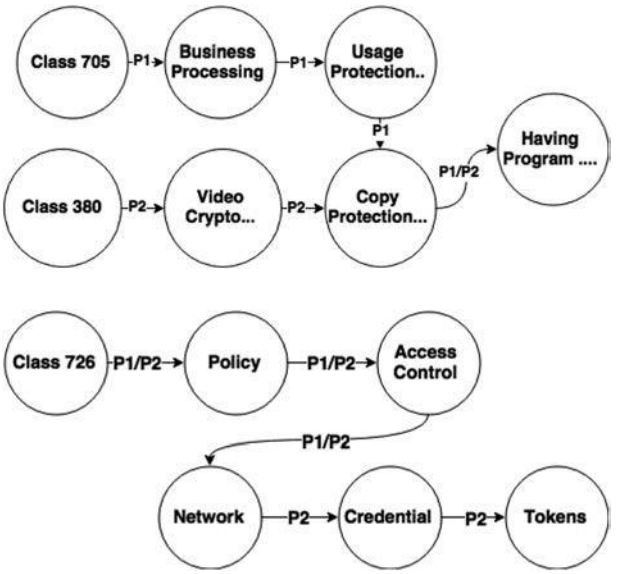

Figure 14. Single concept in Multiple Paths

Figure 15. P_1 is overlapped with P_2

Algorithm 7 Path Ranking Algorithm
Input: (*path_map*) where *path_map* is produced by from SeMP
Output: sorted map of ranked path_map(*PI, PS*), where *PI* is set of path ids and *PS* is set of corresponding scores.

```
for each path_k in path_map do
  path_k score = calculate-WF(path_k)
end for
for each path_k in path_map do
  if ∃path_k, path_m∈path_map and path_k ⊂ path_m then
    path_m score = path_m score + calculate-WF(path_k)
    Remove path_k
  end if
  if ∃path_k, path_m and path_k ⊇ path_m then
    path_k score = path_k score + calculate-WF(path_m)
    Remove path_m
  end if
end for
```

In this version, a semantic validation step is added to initial procedure to assess the superior-subordinate relationship between two paths:

(1) *Superior Relation*: $P_a \supset P_b$, if P_a is superior to P_b.

(2) *Subordinate Relation*: $P_a \subset P_b$, if P_a is subset of P_b.

Following this validation, if P_1 is shorter than P_2, then P_1 is considered as superior to P_2 and vice verse. Finally, we would only need to increase the ranking score of the most specific path.

6.1.5 Re-evaluating concept's score. The ranking formula proposed in Formula 2 does not fully reflect the codependent nature within classifications system, particularly in the patent industry. Intuitively, the more one class is referred by the others, the less priority its concepts should be ranked. In fact, the most popular class tends to include generic concepts that could be reusable by many other classes and, thus, would be insignificant to be class's representative. Observing that the class hierarchy also shares similar characteristics as important pages, we embed the PageRank Algorithm to compute concept's score within its specific domain. The formula notion is changed to adapt for the class instance, denoted as Class Ranking (denoted as *CR*), we compute the class score as follows:

$$S = \frac{1}{CR(A)} = \frac{1}{(1 - d) + d * \left(\sum_{i=1}^{n} \frac{CR(T_i)}{C(T_i)} \right)} \tag{3}$$

Where:

$CR(A)$ = is the Class Rank of Class A in which concept S could be found;
$CR(T_i)$ = is the Class Rank of Class T_i that links to Class A;
$C(T_i)$ = is the number of outbound links on Class T_i; and
d = is a damping factor between 0 and 1.

Replacing Formula 3 by Formula 2, we obtain our finalized ranking formula:

$$P = n_A * \frac{1}{(1-d) + d * \left(\sum_{i=1}^{n} \frac{CR(T_i)}{C(T_i)} \right)} \quad (4)$$

Where n_A is the total number of concepts from the root to the target concepts in Class A.

Figure 16 depicts a subset of classes' references extracted from USPC. The damping factor d is recommended at 0.85 (Haveliwala, 1999) and thus, $d = 0.85$ is also chosen for our experiment. Further research would possibly take into account the clarification on appropriate damping factor in patent classification system. Because the collection of patent classes is relatively small, such factors could be observed:

$$CR(380) = 0.5 + 0.5 * \left(\frac{CR(726)}{2} + \frac{CR(705)}{2} \right) = 0.909$$
$$CR(705) = 0.5 + 0.5 * \left(\frac{CR(726)}{2} + \frac{CR(380)}{2} \right) = 0.909$$
$$CR(726) = 0.5 + 0.5 * \frac{CR(380)}{2} = 0.727$$
$$CR(902) = 0.5 + 0.5 * \frac{CR(705)}{2} = 0.727$$

7. Evaluation

In this paper, our extended datasets of more than 400 classes are exploited to assess our contributions on wider aspects:

- to evaluate our proposed GeTFIRST search approach through revisiting the severity of the semantic ambiguation issue in most generic patent search engines; and
- to confirm the advantages in semantic disambiguation capability using the GeTFIRST.

A set of ten predefined keywords ranked by the complexity of appearance was supplied to both GeTFIRST and USPTO's patent search engine (US Patent and T Office, 2015) to gather evaluative metrics and compare their approaches in handling ambiguation. Based on these preparations, we performed two evaluation activities:

(1) Using predefined keywords as in Table I, we initiated search on PIRS. In specific, we tracked the number of results directly from PIRS. On the other

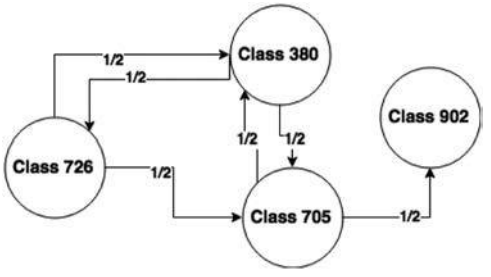

Figure 16.
Classes in USPC share its importance

Table I.
USPTO's results for experimental queries

Experimental query	USPTO		Un-ranked GeTFIRST		Ranked GeTFIRST	
	No. of results	No. of domains	No. of results	No. of domains	No. of results	No. of domains
1. Copy protection	64,849	10	3,365	4	3,365	4
2. Key management	1,66,024	10	3,051	2	3,051	2
3. Task assignment	31,893	10	917	2	917	2
4. Charge determination	1,39,512	10	1,602	3	1,602	3
5. Secure transaction	39,461	10	1,447	1	1,447	1
6. Smart card	41,970	10	1,151	4	1,151	1
7. business processing using cryptography	7,772	10	5,271	1	86	1
8. Advertisement avoiding fraud	406	10	410	13	410	4
9. Usage protection including third party for collecting payment	17	10	2,922	2	461	1
10. Usage protection having program ID	0	10	1,173	3	444	1

hand, because the domains of a particular keyword could not be accurately determined using PIRS due to vast resource of patents, we used Patent Classifier (2015) as an alternative solution. Patent Classifier is a prediction tool that allows user to search for appropriate classes in USPC given a free-text query. Its accuracy could reach up to 84 per cent given a training set of 2,179,828 patents. Because all of our test queries were reasonably classified by Patent Classifier, we decided to approximately acquire domains using Patent Classifier. Despite the fact that an ambiguous word could exist in a larger number of classes using PIRS, Patent Classifier only returns top ten relevant classes.

(2) We perform a GeTFIRST search approach without any supports from external resources. Within GeT-FIRST, we also compared the effectiveness of our proposed semantic ranking algorithm against the un-ranked version.

Finally, both approaches are then served as evaluation to the severity of the ambiguation problems in an ordinary syntactical-based search engine. Our results are given in Table I.

A key point could be observed from the results. The probability that all search results belong to a single domain is very small (in our result, there was no such case). Even though, the total observed domains was not revealed in Patent Classifier itself, the fact that the number of returned domains for a single keyword is more than one and may have the tendency of an increase with the number of results represent severe ambiguation problem. This semantic issue often leads to lower accuracy and user dissatisfaction in an ordinary syntactical-based search engine.

The results from PIRS represent a common characteristic of syntactical-based full-text IR: a high recall with a low accuracy and no semantic disambiguation. With the advanced classification tool Patent Classifier, PIRS returned 64,849 documents, from ten different classes, that contain the keyword "copy protection". Without semantic and domain awareness, PIRS mixed all documents from various domains into a single result list and left the semantic disambiguation to manual efforts of user. As the number of results increases, this approach is unrealistic, frustrating and might lead to loss of useful information.

Table I also shows results from a different approach to semantic disambiguation used by GeTFIRST. Instead of performing full-text syntactical matching with each indexed document, GeTFIRST searches for matching concepts in GeT-based ontology and retrieves relevant documents in its document set. The knowledge about exact concept's location in GeT-based ontology helps GeTFIRST to organize results into groups according to their domain and provides meaningful description for each group to support users.

We used a logarithm function to present more readable results in Figures 17 and 18 due to a large gap between the results (Table I). In Figure 17, PIRS was unable to produce a good recall and witnessed a down-trend pattern when the user query became more complicated. Meanwhile, GeTFIRST generally showed more stable performances in the total returned results. As in the query "advertisement avoiding fraud" which potentially has semantic relation "is-part-of", unlike the un-ranked GeTFIRST where all documents within matching concepts were aggressively returned, the ranked version only gave bonus to the most specific concept and thus, improved the overall retrieval performance.

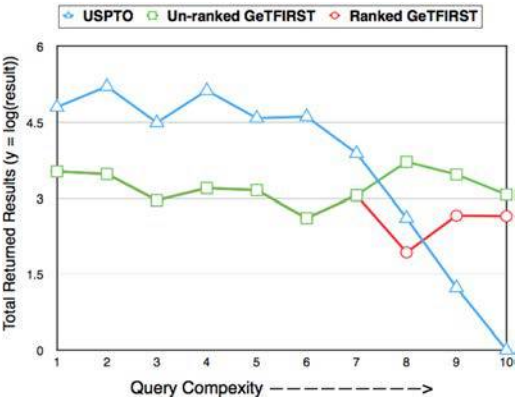

Figure 17.
PIRS vs GeTFIRST – Overall Results

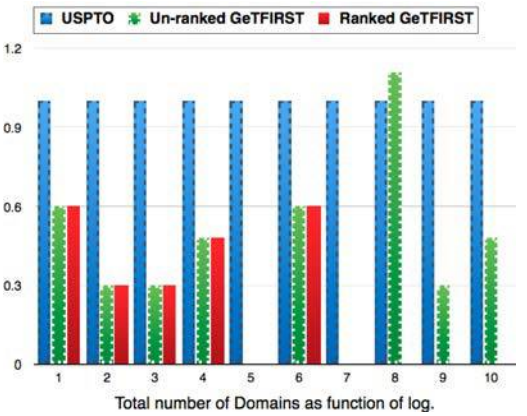

Figure 18.
PIRS vs GeTFIRST – Total Domains

In Figure 18, GeTFIRST outperformed PIRS, thanks to its rich set of semantic awareness. In "advertisement avoiding fraud", while USPTO's users might struggle with mixed domains for such keywords, unranked version of GeTFIRST successfully identified only 13 the semantic paths led from each domain. Nevertheless, the ranked version of GeT-FIRST was more greedy for the most specific class, it yielded even better disambiguation ability. As the query structure became sophisticated, the ranked GeTFIRST was able to classify the correct classification while the other approaches showed signs of semantic confusion.

8. Conclusion

Semantic ambiguation refers to the conflict between meanings of terms used by different parties in a communication. It could be resolved through establishing and reconciling relative positions of parties in the communication by matching their terms' meanings to a common domain-specific ontology. Therefore, a prerequisite for the semantic disambiguation problems is the combination of a domain-specific ontology and a suitable retrieval strategy.

Integrating the domain-specific ontology into an IR system faces three common obstacles:

(1) many layers added for transparency if a keyword-based IR is chosen;

(2) limitations of existing ontology-based IRs; and

(3) there are no proper standard criteria to evaluate efficiency among set of ontology-based IRs.

To address the above problems, we chose to approach our GeT-based ontology using our proposed GeTFIRST approach to exploit the expressive capability and computer understandability of our concept – relationship graphs. The advantages in disambiguation capability of GeT-ontology search strategy were confirmed through the evaluation on USPC Classification Schemes.

Our proposed approach, GeTFIRST, could be further improved in the future work. First, the current scope of the GeT-ontology-based retrieval is limited to the classification titles for pivoting operations. This could be expanded to documents linking to each class. Second, the automation in GeTC could be extended by introducing a set of starting heuristics to detect and suggest relationship types automatically. Finally, the investigation could be performed to incorporate GeT-based ontology into the solution for semantic disambiguation in different fields beyond the IR.

References

Association for Computing Machinery (2012), "The 2012 ACM computing classification system", available at: www.acm.org/ (accessed 4 May 2015).

Bhattacharjee, S. and Ghosh, S. (2014), "Automatic resolution of semantic heterogeneity in gis: an ontology based approach", *Advanced Computing, Networking and Informatics*, Springer International Publishing Switzerland, Switzerland, Vol. 1, pp. 585-591.

Bouramoul, A., Kholladi, M. and Doan, B.L. (2012), "An ontology-based approach for semantics ranking of the web search engines results", in 3rd IEEE International Conference on Multimedia Computing and Systems (ICMCS'2012), Tangier, pp. 797-802.

Cantrell, S.J. (2013), "Ontology-based search engine in support of a decision support system", US Patent No: US 20130246382 a1.

Castells, P., Fernandez, M. and Vallet, D. (2007), "An adaptation of the vector-space model for ontology-based information retrieval", *Knowledge and Data Engineering, IEEE Transactions on*, Vol. 19 No. 2, pp. 261-272.

Cooperative Patent Classification (CPC) (2015), "International patent classification systems", available at: www. cooperativepatentclassification.org/index.html (accessed 1 June 2015).

Ding, L., Finin, T., Joshi, A., Pan, R., Cost, R.S., Peng, Y., Reddivari, P., Doshi, V. and Sachs, J. (2004), "Swoogle: a search and metadata engine for the semantic web", *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management, CIKM '04, Washington, DC*, ACM New York, NY, pp. 652-659.

Google Inc. (2015), "Google", available at: www.google.com (accessed 4 May 2015).

Hakimpour, F. and Geppert, A. (2001), "Resolving semantic heterogeneity in schema integration", *FOIS '01 Proceedings of the International Conference on Formal Ontology in Information Systems-Volume, Ogunquit, ME*, ACM New York, NY, pp. 297-308.

Halevy, A. (2005), "Why your data won't mix", *Queue -Semi-structured Data*, Vol. 3 No. 8, pp. 50-58.

Haveliwala, T. (1999), "Efficient computation of pagerank", Technical Report 1999-1931, Stanford InfoLab, Stanford, CA.

Hudson, A.R., Bright, M.W. and Pakzad, S. (1994), "Automated resolution of semantic heterogeneity in multidatabases", *ACM Transactions on Database Systems*, Vol. 2 No. 19, pp. 212-253.

Hull, R. (1997), "Managing semantic heterogeneity in databases: a theoretical prospective", *Proceedings of the Sixteenth ACM SIGACT-SIGMOD-SIGACT Symposium on Principles of Database Systems*, *Tucson, AZ*, ACM New York, NY, pp. 51-61.

Jarmasz, M. and Szpakowicz, S. (2001), "Roget's Thesaurus: a lexical resource to treasure", *NAACL WordNet and Other Lexical Resources Workshop*, Pittsburgh, PA, pp. 186-188.

Kashyap, V. and Sheth, A. (1997), *Cooperative Information Systems – Trends and Directions: Semantic Heterogeneity in Global Information Systems: The Role of Metadata, Context, and Ontologies*, Academic Press, The Blvd, Kidlington OX5 1GB.

Kent, W. (1989), "The many forms of a single fact", *Thirty-Fourth IEEE Computer Society International Conference: Intellectual Leverage, Digest of Papers*, IEEE, pp. 438-443.

Kiryakov, A., Popov, B., Terziev, I., Manov, D. and Ognyanoff, D. (2004), "Semantic annotation, indexing, and retrieval", *Web Semantics: Science, Services and Agents on the World Wide Web*, Vol. 2 No. 1, pp. 49-79.

Klapaftis, I. and Manandhar, S. (2005), "Google & Wordnet based word sense disambiguation", *Proceedings of the Workshop on Learning and Extending Ontologies by Using Machine Learning Methods*, *Bonn, Germany*.

Lahr, N.B. and Barr, G.C. (2011), "System and methods for searching and displaying ontology-based data structures", US Patent No: US 20110264680 A1.

Leacock, C. and Chodorow, M. (1998), "Combining local context and wordnet similarity for word sense identification", *WordNet: An Electronic Lexical Database*, Vol. 49 No. 2, pp. 265-283.

Li, Y., Bandar, Z. and Mclean, D. (2003), "An approach for measuring semantic similarity between words using multiple information sources", *Knowledge and Data Engineering, IEEE Transactions*, Vol. 15 No. 4.

Ma, S. and Tian, L. (2015), "Ontology-based semantic retrieval for mechanical design knowledge", *International Journal of Computer Integrated Manufacturing*, Vol. 28 No. 2, pp. 226-238.

Mayfield, J. and Finin, T. (2003), "Information retrieval on the Semantic web: integrating inference and retrieval", *Proceedings of the SIGIR Workshop on the Semantic Web*, *Toronto, Canada*.

Microsoft Corporation (2015), "Bing search engine".

Mukhopadhyay, D., Banik, A., Mukherjee, S., Bhattacharya, J. and Kim, Y. (2007), "A domain specific ontology based semantic web search engine", *The 7th International Workshop MSPT 2007 Proceedings, Youngil Publication, Republic of Korea*, pp. 81-89.

Navathe, S.B. and Elmasri, R. (2000), *Fundamentals of Database Systems* (5th edition), Pearson New York , NY.

Organization, W.I.P. (2015), "International patent classification", available at: www.wipo.int (accessed 1 June 2015).

Page, L., Brin, S., Motwani, R. and Winograd, T. (1999), "The pagerank citation ranking: bringing order to the web", Technical Report 1999-1966, Stanford InfoLab, Stanford, CA.

Patent Classifier (2015), "Patent classifier service", available at: http://patentclassifier.com (accessed 1 June 2015).

Rada, R., Mili, H., Bicknell, E. and Blettner, M. (1989), "Development and application of a metric on semantic nets", *Systems, Man and Cybernetics, IEEE Transactions on*, Vol. 19 No. 1, pp. 17-30.

Richardson, R., Smeaton, A. and Murphy, J. (1994), "Using wordnet as a knowledge base for measuring semantic similarity between words", *Proceedings of AICS Conference*, *Trinity College*, *Dublin, Ireland*.

Saltor, F., Garcia-Solaco, M. and Castellanos, M. (1995), *Object-Oriented Multidatabase Systems: Semantic Heterogeneity in Multidatabase Systems*, Prentice Hall International (UK), Hertfordshire.

Sheth, A.P. and Larson, J.A. (1990), "Federated database systems for managing distributed, heterogeneous and autonomous databases", *ACM Computing Surveys*, Vol. 3 No. 22, pp. 183-236.

United States Patent And Trademark Office (USPTO) (2015a), "Handbook of classification", available at: www.uspto.gov (accessed 1 June 2015).

United States Patent And Trademark Office (USPTO) (2015b), "Classification standards and development", available at: www.uspto.gov (accessed 1 June 2015).

Ventrone, V. (1991), "Semantic heterogeneity as a result of domain evolution", *ACM SIGMOD Record*, Vol. 4 No. 20, pp. 16-20.

Vinh, V.X., Nguyen, H.Q. and Tran, K.N. (2014), "Get-based ontology construction for semantic disambiguation", *Proceedings of the 16th International Conference on Information Integration and Web-based Applications; Services*, *Hanoi, Vietnam*, ACM New York, NY, pp. 445-453.

Volk, M., Vintar, S. and Buitelaar, P. (2003), "Ontologies in cross-language information retrieval", *Proceeding of the Workshop Ontologie-basiertes Wissensmanagement (WOW 2003)*, *Luzern, Switzerland*.

Wielinga, B.J., Schreiber, A.T., Wielemaker, J. and Sandberg, J.A.C. (2001), "From thesaurus to ontology", *Proceedings of the 1st International Conference on Knowledge Capture, K-CAP '01*, *Victoria, Canada*, ACM New York, NY, pp. 194-201.

Yahoo Inc. (2015), "Yahoo Search – Web Search", available at: https://search.yahoo.com (accessed 1 June 2015).

Ying, L., Huimin, Z., Hui, L. and Chengli, Z. (2013), "Ontology-based knowledge representation for resolution of semantic heterogeneity in GIS", *Proceedings of the 2013 International Conference on Management of e-Commerce and e-Government*, *Kunming*, pp. 336-340.

Further reading

Europe Patent Office (2015), "Espacenet – Patent Search", available at: http://worldwide.espacenet.com/ (accessed 1 June 2015).

Corresponding author

Hong-Quang Nguyen can be contacted at: nhquang@hcmiu.edu.vn