Incorporating compactness to generate term-association view snippets for ontology search

Weiyi Ge\textsuperscript{a,\,*}, Gong Cheng\textsuperscript{b}, Huiying Li\textsuperscript{a}, Yuzhong Qu\textsuperscript{b}

\textsuperscript{a}School of Computer Science and Engineering, Southeast University, Nanjing 210096, China
\textsuperscript{b}State Key Laboratory for Novel Software Technology, Nanjing University, Nanjing 210093, China

**A R T I C L E   I N F O**

**Article history:**
Received 16 September 2011
Received in revised form 28 June 2012
Accepted 18 July 2012
Available online 2 November 2012

**Keywords:**
Snippet generation
Ontology search
Term association graph
Maximal $r$-radius subgraph
Group Steiner

**A B S T R A C T**

A query-relevant snippet for ontology search is useful for deciding if an ontology fits users’ needs. In this paper, we illustrate a good snippet in a keyword-based ontology search engine should be with term-association view and compact, and propose an approach to generate it. To obtain term-association view snippets, a model of term association graph for ontology is proposed, and a concept of maximal $r$-radius subgraph is introduced to decompose the term association graph into connected subgraphs, which preserve close relations between terms. To achieve compactness, in a query-relevant maximal $r$-radius subgraph, a connected subgraph thereof with a small graph weight is extracted as a sub-snippet. Finally, a greedy method is used to select sub-snippets to form a snippet in consideration of query relevance and compactness without violating the length constraint. An empirical study on our implementation shows that our approach is feasible. An evaluation on effectiveness shows that the term-association view snippet is favored by users, and the compactness helps reading and judgment.

© 2012 Elsevier Ltd. All rights reserved.

**1. Introduction**

Web ontologies described in RDF Schema and the OWL Web Ontology Language provide shared terms\textsuperscript{1} for describing things, and thus enable semantic interoperability of data across different applications. When developing Semantic Web applications or constructing new ontologies, developers are recommended to reuse or extend popular ontologies, as it can enhance data interoperability, reduce cost and accelerate development progress. As of today, a growing number of ontologies have emerged on the Web (Campinas et al., 2011). Therefore, ontology search becomes an important issue for Semantic Web applications. Furthermore, there may be numerous ontologies whose textual description is matched with the keyword query, and even some of them contain a large number of terms. It is inefficient for users to inspect each ontology returned. Instead, the search engine should provide a query-relevant part of ontology, instead of the entire one. It is usually called *snippet* in Web search.

When searching for ontologies, query-relevant terms are usually what users want to reuse or extend. But just listing part (all) of these query-relevant terms as snippet is not good enough to help users to judge the relevance. The semantic relations among them are very important to confirm if the ontology is a good candidate. For example, in Fig. B.1, given a query “person employee”, Person and Employee are two query-relevant terms in an ontology, and the relation rdfs:subClassOf between them is a very useful supplement for judgement. Furthermore, an indirect relation between two terms may exist. For example, in Fig. B.1, there is an indirect relation between Person and AcademicStaff via Employee. But, when two

\textsuperscript{1} In this paper, terms are classes, properties or individuals identified by URIs.

\textsuperscript{*} Corresponding author. Tel./fax: +86 25 89680923.
E-mail addresses: geweiyi@gmail.com (W. Ge), gcheng@nju.edu.cn (G. Cheng), huiyingli@seu.edu.cn (H. Li), yzqu@nju.edu.cn (Y. Qu).
terms are connected with a long distance, the relation between them is weak and thus not closely-related, e.g., member and Conference in Fig. B.1. Besides, a snippet does not have enough space to display such a long relation either. Therefore, query-relevant terms with direct relations or indirect but close relations are preferred in a snippet.

For an ontology, there exist numerous snippets which contain not only query-relevant terms but also close relations between them. But for the sake of reading and judgement, we hope to find a compact one, i.e., the sum of the length of all relations is as small as possible. For example, in Fig. B.1, if Person and AcademicStaff are two query-relevant terms, we should find a relation from Person to AcademicStaff via Employee, and there should be no other terms included in the snippet due to the compact requirement. Furthermore, term associations (i.e., direct relations) are with different preferences (e.g., the relation rdfs:subClassOf is preferred than the relation foaf:knows for ontology search).2 With regard to this, the compactness requires the sum of the weight of all relations is as small as possible, where the weight of all relations is measured by the sum of the reciprocal of the preference of each constituent term association.

In a summary, we believe that a good snippet for ontology search is a term-association view information unit within a bounded size that effectively summarizes the query result for human reading and judgment. Firstly, term-association view means that a snippet for ontology search should contain not only query-relevant terms but also their close relations. Secondly, to be efficient for reading and judgment, we expect that each snippet is compact. That is, the sum of the weight of each relation is as small as possible. Finally, the snippet has a strict restriction on its size (the number of terms), as there are usually ten snippets to be shown in a page.

To achieve a term-association view snippet, we propose a notion of term association graph (TAG) to model the structure on terms in an ontology, where nodes stand for the terms of the ontology, and an edge between a pair of terms is associated with a set of so-called RDF sentences (Zhang et al., 2007) that characterize term associations. The weight of each edge represents the reciprocal of the preference of the term association. As terms with a long distance are not closely-related and a snippet does not have enough space to display such long relations, we decompose the original TAG into some maximal r-radius subgraphs, which preserves all close relations between terms.

To achieve compactness, in a query-relevant maximal r-radius subgraph, we want to find a subgraph of it with a small graph weight in terms of the sum of the weight of each edge. Firstly, a set of keywords is mapped to a collection of groups of nodes. Secondly, given a graph and a collection of groups, we should find a minimum-weight connected subgraph that contains at least one node from each group. This is a group Steiner problem. Therefore, we use an existing algorithm, and the result is called sub-snippet. Finally, to form a query-relevant and compact snippet with a size restriction, we select some sub-snippets and assemble them to form the result.

The rest of the paper is organized as follows. Section 2 introduces related work. Section 3 presents the method to construct a TAG for each ontology. Section 4 extracts all maximal r-radius subgraphs for each TAG. Section 5 generates a sub-snippet for each query-relevant maximal r-radius subgraph. Section 6 assembles these sub-snippets into the final snippet. Section 7 evaluates our approach. Section 8 gives the conclusion and future work.

2. Related work

2.1. Ontology search

Because of the importance of ontology search, many research papers focus on this topic. OntoKhoj (Patel et al., 2003), Swoogle (Ding et al., 2005), Ontosearch (Zhang et al., 2005), Watson (d’Aquin et al., 2007), Sindice (Tummarello et al.,

---

2 Please refer to Section 3.2 for detail.
2.3. Ontology summarization

In general, the problem of snippet generation for ontology search is how to compute a summary of an ontology from its description given as an RDF graph, biased towards a keyword query. A more general problem is automatic summarization for web data. Not like XML, which has a tree-like structure, an ontology can be represented as a graph. Therefore, their algorithm can not effectively generate snippets for ontology search.

Although ontology search is widely studied, its snippet generation is not always the primary topic. Swoogle (Ding et al., 2007) takes an ontology as a sequence of terms (in alphabetical order of their local names), and selects one query-relevant term and some of its neighboring terms as a snippet. Watson (d’Aquin et al., 2007) takes an ontology as a bag of terms, and returns a list of all terms whose local names are triggered by query keywords. Falcons (Cheng et al., 2011) takes an ontology as a bag of RDF sentences (Zhang et al., 2007) and extracts some of them with regards to salience, query relevance, and cohesion.

These approaches actually decompose an RDF graph into pieces (e.g. RDF sentences, terms) and select some of them into the snippet. Distinguished from them, our approach focus on generating a term-association view snippet, i.e. not only terms but also their close relations.

2.4. Keyword search on (semi-)structured data

There are many research work on keyword search on (semi-)structured data (Bhalotia et al., 2002; Kasneci et al., 2009; Luo et al., 2007; Li et al., 2008). Chen et al. (2009) give an overview of the state-of-the-art techniques. The Semantic Web community also uses them on RDF graphs (Ning et al., 2009; Tran et al., 2009). Their basic ideas are to model (semi-)structured data as a graph, and map query keywords to nodes in the graph, and then search and rank several subgraphs that connect these elements.

These approaches are designed for query answering. They are different from the purpose of a snippet for ontology search, which is used to support users’ quick reading and relevance judgment. Keyword search has no mandatory constraint on the
size of resulting graph, but the size for a snippet is strictly limited. Besides, they are interested in finding several answers and rank them. But in our case, we expect that there is only one relevant piece.

3. Term association graph

3.1. Ontology and RDF sentence

In RDF, identifiers consist of three disjoint sets: URIs (U), blank nodes (B) and literals (L). An RDF triple \( t \) has three parts: a subject (subj(\( t \))), a predicate (pred(\( t \))), and an object (obj(\( t \))), which is with the form \((\text{subj}(\( t \)), \text{pred}(\( t \)), \text{obj}(\( t \))) \in (U \cup B) \times U \times (U \cup B \cup L)\). An RDF graph is a set of RDF triples.

Definition 1. [Ontology] An ontology \( O \triangleq (\text{ID}, T, R) \) consists of:
- \( \text{ID} \), a namespace URI as its identification,
- \( T \), a set of terms that are classes, properties, or individuals identified by URIs in the namespace \( \text{ID} \),
- \( R \), an RDF graph describing the terms in \( T \).

For example, Table B.1 shows the original document format of an ontology serialized in RDF/XML syntax. In the following, we will use this ontology as an example to illustrate our approach. This document serializes an RDF graph\(^8\), which is shown in Fig. B.1. In this ontology, eleven terms in the namespace http://swrc.ontoware.org/ontology# are described.

In RDF, a blank node is used to identify a resource for which a URI or literal is not given. That is, it is treated as indicating the existence of a resource, without saying anything about the name of that resource. An important role of a blank node is to identify an anonymous class as a restriction. For example, in the dashed area of Fig. B.1, four RDF triples sharing a common blank node (\(_{\cdot}5\)) express that all member of a project are persons. Furthermore, from this example, we can find that it makes no sense if RDF triples sharing common blank nodes are separated. Therefore, we borrow the concept RDF sentence\(^{Zhang et al. (2007)}\) to encapsulate these triples as a whole.

In an RDF graph, two RDF triples are called \( B \)-connected if they share common blank nodes. \( B \)-connected is defined to be transitive.

Definition 2. (RDF Sentence) An RDF sentence, as a subgraph of an RDF graph, is a maximum set of \( B \)-connected RDF triples.

For example, there are eight RDF sentences in the RDF graph presented by Fig. B.1, listed on the right side (from \( S_1 \) to \( S_8 \)) of Fig. B.2.

3.2. Preference of RDF sentences

RDF sentences can be divided into two categories: those that define classes or properties (schema sentences), and those that describe data using these terms. In the context of ontology search, users focus more on schema sentences than data-describing ones, as they want to reuse or extend defined classes/properties. Unfortunately, a plain discrimination between them is not possible\(^{Hayes and Gutierrez (2004)}\). We use a simple heuristic rule to this issue.

An RDF sentence \( S \) is a schema sentence, if there is an RDF triple \( s \in S \) satisfying:
- \( \text{pred}(s) = \text{rdf:type} \) and \( \text{obj}(s) = C \), where \( C \) is a class defined by OWL or RDFS ontology\(^{\text{excluded}}\), or
- \( \text{pred}(s) = p \), where \( p \) is a non-annotation property defined by OWL or RDFS ontology\(^9\).

For example, \((a, \text{rdfs:seeAlso}, b)\) is not a schema sentence, as \text{rdfs:seeAlso} is an annotation property defined by OWL. \((a, \text{rdfs:type}, \text{foaf:Person})\) is not a schema sentence either, as \text{foaf:Person} is not a class defined by OWL or RDFS. Obviously, all sentences listed in the right hand side of Fig. B.2 are schema sentences.

For an RDF sentence \( S \), the following equation denotes its preference:

\[
\text{pref}_S(S) = \begin{cases} \alpha & \text{if } S \in \text{SchemaSents}, \\ 1 - \alpha & \text{if } S \notin \text{SchemaSents}, \end{cases}
\]

where \( 0.5 < \alpha < 1 \) and SchemaSents is the set of all schema sentences.

---

\(^7\) http://www.w3.org/TR/rdf-concepts/#dfn-blank-node.

\(^8\) For more information about the RDF graph and its RDF/XML syntax, please refer to http://www.w3.org/TR/REC-rdf-syntax/.

\(^9\) Five annotation properties are predefined by OWL, namely: \text{owl:versionInfo}, \text{rdfs:label}, \text{rdfs:comment}, \text{rdfs:seeAlso}, and \text{rdfs:isDefinedBy}. (http://www.w3.org/TR/owl-ref/#Annotations.
3.3. Term association

Given an ontology, we are interested in only the relations between its constituent terms but not blank nodes or literals. To achieve this, we propose a notion of term association that operates on the concept of RDF sentence.

Fig. B.2. Term association graph (TAG) of the RDF graph in Fig. B.1. $S_1, \ldots, S_8$ are eight RDF sentences.

3.3. Term association

Given an ontology, we are interested in only the relations between its constituent terms but not blank nodes or literals. To achieve this, we propose a notion of term association that operates on the concept of RDF sentence.
Definition 3. (Term Association) For two terms \( t_1 \) and \( t_2 \) and an RDF graph \( R \), term association \( \Xi(t_1, t_2, R) \) is a set of RDF sentences in \( R \), in each of which there is a directed path connecting \( t_1 \) and \( t_2 \) whose arcs exclude \texttt{rdfs:type}.

For example, in Fig. B.2, there is a directed path connecting \texttt{Project} and \texttt{Person} in \( S_5 \). So, the term association between \texttt{Project} and \texttt{Person} is \( S_5 \).

The preference of a term association \( \Xi(t_1, t_2, R) \) is given by:

\[
\text{pref}_{\Xi}(\Xi(t_1, t_2, R)) = \sum_{S \in \Xi(t_1, t_2, R)} \text{pref}_{\Xi}(S)
\]

We transform the RDF graph of an ontology into a new term association graph that characterizes all the associations between its constituent terms.

Definition 4. (Term Association Graph (TAG)) Given an ontology \( \mathcal{O} \), its term association graph \( G(\mathcal{O}) \equiv (V, E, \text{Lbl}_V, \text{Lbl}_E, \text{Wt}_E) \) consists of:

- \( V \triangleq T(\mathcal{O}) \), the set of nodes,
- \( E \subseteq V \times V \), the set of undirected edges, where \((v_1, v_2) \in E \) if and only if \( \Xi(v_1, v_2, R(\mathcal{O})) \neq \emptyset \),
- \( \text{Lbl}_V : V \mapsto 2^W \), a function that labels each \( v \in V \) with a set of words collected from its local name\(^{10}\) and labels,\(^{11}\) where \( W \) is a set of words collected from local names and labels of all terms,
- \( \text{Lbl}_E : E \mapsto \mathbb{R} \), a function that labels each \( (v_1, v_2) \in E \) with \( \Xi(v_1, v_2, R(\mathcal{O})) \), and
- \( \text{Wt}_E : E \mapsto \mathbb{R} \), a weighting function, \( \text{Wt}_E((v_1, v_2)) = \frac{\text{pref}_{\Xi}(\Xi(v_1, v_2, R(\mathcal{O})))}{\text{pref}_{\Xi}(\Xi(v_1, v_2, R(\mathcal{O})))} \).

For example, Fig. B.2 presents the TAG in correspondence to the RDF graph given by Fig. B.1 (edge weights are not shown).

4. Maximal \( r \)-radius subgraph

4.1. Motivation

In the Introduction, we have roughly explained the reason why TAG should be decomposed into maximal \( r \)-radius subgraphs. In this subsection, we will give a solid discussion.

Firstly, Cheng and Qu, 2008 find that every pair of terms can be connected within a small distance in most situations. On the contrary, if two terms are separated by a long distance, they are not closely-related. Therefore, we should only find term relations within close distances, which is a requirement for the term-association view. In other words, a maximum number of hops should be applied when traversing a TAG to find relations. Secondly, a snippet has a strict restriction on its size. There is usually no space to display a long-distance relation. Even if such long-distance relation can be contained in the snippet, there is no space to display any other information. Thus, the snippet will have a narrow coverage of the topic, especially the document has more than one topic. Last but not least, if we directly search compact and query-relevant sub-graphs as snippets on a TAG, most effective but not efficient algorithms cannot be used, as a TAG may have an arbitrary size.

Therefore, we take a TAG as an unweighted graph and give an algorithm to decompose the graph into pieces called maximal \( r \)-radius subgraphs, in which the greatest distance between any pair of vertices, i.e. diameter,\(^{12}\) is limited to \( 2r \). And any pair of terms whose distance is within \( 2r \) must be contained in some maximal \( r \)-radius subgraphs.

With this decomposition, we ensure that long-distance relations are eliminated from our snippet and subgraphs are of moderate sizes (Experiments in Section 7.1.1 demonstrates the usability for reducing the graph scales). Thus, more topics can be covered, and more sophisticated algorithms can be applied. Furthermore, maximal \( r \)-radius subgraphs are the basic units for data organization. And keywords in the query can locate them quickly with an index.

4.2. Subgraph decomposition

Li et al. (2008) introduce an approach to break a large connected graph into pieces of maximal \( r \)-radius graphs such that each graph has a radius of exact \( r \). But in our context, a TAG may be disconnected, and thus it contains connected components whose radiuses are less than \( r \). If only \( r \)-radius graphs are extracted, these small connected components will be lost. So, we introduce the concept of \( r \)-radius subgraph, which allows subgraphs whose radiuses less than \( r \).

Definition 5. (\( r \)-Radius subgraph) \( G_i \) is an \( r \)-radius subgraph of \( G \), iff \( G_i \) is a subgraph of \( G \) and the radius of \( G_i \) is less than or equal to \( r \).

\(^{10}\) http://www.w3.org/TR/REC-xml-names/#dt-localname.
\(^{11}\) Values of the property \texttt{rdfs:label}.
\(^{12}\) For the definitions of distance, diameter, and radius, please refer to the Wikipedia: http://en.wikipedia.org/wiki/Distance_(graph_theory). An obvious conclusion in the graph theory is that: the diameter is at least the radius and at most twice the radius.
For a given graph, there are exponential number of $r$-radius subgraphs. Therefore, the concept of maximal $r$-radius subgraph is introduced.

**Definition 6.** (Maximal $r$-radius subgraph) Given a graph $G$ and an $r$-radius subgraph $G_i$ of $G$, $G_i$ is called a maximal $r$-radius subgraph if there is no other $r$-radius subgraph $G_j$ such that $G_i$ is a subgraph of $G_j$.

Next, we propose an algorithm $\text{MRR\_extraction}$ (Algorithm 1) to compute all maximal $r$-radius subgraphs of a graph. Generally, lines 2–6 generate some $r$-radius subgraph candidates, and lines 8–21 pick all maximal $r$-radius graphs from them. Specifically, for each node in the graph, line 3 finds a set of all nodes that it can reach in $r$ steps, and line 4 generates a subgraph of $G$ induced by this set of nodes. For every two induced subgraphs, lines 11–17 check whether all nodes in the small graph are contained in the large one. If true, remove this small one in line 19.

**Algorithm 1.** The algorithm $\text{MRR\_extraction}$ to extract all maximal $r$-radius subgraphs from the graph $G$

```
1 $G$ is a set;
2 foreach node $v_i$ in $V(G)$ do
3     $V_i := \text{breadth\_first\_search}(G, v_i, r);$  
4     $G_i$ is a subgraph of $G$ induced by $V_i;$  
5     insert $G_i$ into $G;$
6 end
7 $n := |G|;$
8 for every pair of graphs $G_i$ and $G_j$ in $G$ do
9     $G_i$ is the graph with fewer nodes;
10    $G_j$ is the graph with more nodes;
11    $\text{isSubSet} = \text{true};$
12    foreach node $v_i$ in $V(G_i)$ do
13        if $v_i \notin V(G_j)$ then
14            $\text{isSubSet} := \text{false};$
15            break;
16        end
17    end
18    if $\text{isSubSet} = \text{true}$ then
19        remove $G_i$ from $G;$
20    end
21 end
22 return $G;$
```

**Theorem 1.** For a given graph, the algorithm $\text{MRR\_extraction}$ generates all maximal $r$-radius subgraphs in $O(n^3)$, where $n$ is the number of nodes in the graph.

This theorem ensures the correctness and effectiveness of our algorithm in extracting all maximal $r$-radius subgraphs. It has a straight-forward proof, which is provided in A. Besides, an obvious observation from the algorithm is that the number of all maximal $r$-radius subgraphs is less than or equal to the number of nodes of the original graph.

For example, for the TAG in Fig. B.2, we generate five maximal 1-radius subgraphs in Fig. B.3.

Besides, the following theorem (proof is in B) shows that all maximal $r$-radius subgraphs preserve all paths between closely-related terms.

**Theorem 2.** Given a graph $G$, its all maximal $r$-radius subgraphs $\text{MRR}(G)$ have two important properties:

1. Every pair of nodes in each maximal $r$-radius subgraph has a distance (i.e. the shortest path length) less than or equal to $2r$, i.e. $\forall G_i \in \text{MRR}(G), \forall v_i, v_j \in V(G_i), d(v_i, v_j) \leq 2r$, where $d(v_i, v_j)$ is the distance between $v_i$ and $v_j$.
2. All pairs of nodes within $2r$ must be contained in some maximal $r$-radius subgraph, i.e. $\forall v_i, v_j \in V(G) d(v_i, v_j) \leq 2r \implies \exists G_i \in \text{MRR}(G), v_i, v_j \in V(G_i)$.
5. Sub-snippet generation

5.1. Query-relevant maximal r-radius subgraph

For a query \( Q \), i.e. a set of keywords, a maximal \( r \)-radius subgraph \( G_i \) is query-relevant, if and only if there exists a keyword in \( Q \) that matches at least one node in \( G_i \). We call these keywords “relevant keywords”. Formally, \( G_i \) is query-relevant, iff
\[
\exists q \in Q, \exists v \in V(G_i), q \in \text{Lbl}_V(v).
\]
The relevant keywords of \( G_i \) is \( Q_i = \{q | q \in Q \land \exists v \in V(G_i), q \in \text{Lbl}_V(v)\} \).

For example, given the query \( Q = \{\text{conference, staff, person}\} \), all maximal \( r \)-radius subgraphs in Fig. B.3 are query-relevant. Relevant keywords in subgraph (c) are “person” and “staff”.

5.2. Sub-snippet

It is easy to find that each query-relevant maximal \( r \)-radius subgraph satisfies the requirement of term-association view. But it may be large and complex for reading. Therefore, we should tailor it to a smaller connected piece, where there should be at least one relevant node for each keyword. We call it sub-snippet. Here, the “connected” requirement ensures close relations are preserved in sub-snippets.

Definition 7. [Sub-snippet] Given a maximal \( r \)-radius subgraph \( G_i \) and a set of relevant keywords \( Q_i = \{q_1, \ldots, q_g\} \), a sub-snippet \( G_{\text{sub}} \) is a connected subgraph of \( G_i \) satisfying \( \forall q \in Q_i, \exists v \in V(G_{\text{sub}}) \text{ such that } q \in \text{Lbl}_V(v) \).

For example, Fig. B.3c is a sub-snippet with respect to the relevant keywords “person” and “staff”, as it is connected and nodes Person and AcademicStaff are matched with “person” and “staff”, respectively.

But there may be numerous sub-snippet candidates. We propose to provide users with the compactest one. According to the Formula 4, a compact sub-snippet (with large compactness) should be with a small graph weight, where a graph weight is the sum of weights of all edges:
\[
W_{G}(G) = \sum_{e \in E(G)} W_{\ell}(e)
\]

5.3. An algorithm for generating compact sub-snippets

In a maximal \( r \)-radius subgraph \( G_i \), for each relevant keyword \( q_i \in Q_i \), we have a node set \( V_i \subseteq V(G_i) \), where each \( v \in V_i \) satisfies \( q_i \in \text{Lbl}_V(v) \), i.e. its label is matched with \( q_i \). The node set \( V_i \) is called required node set with respect to \( q_i \). Then

\[\text{For a given TAG } G, W_{G}(G) \text{ is a constant. Therefore, a sub-snippet with a smaller graph weight must be more compact.}\]
the problem of finding the compact sub-snippet is transformed to the following problem: given a graph $G_i$ and a collection of required node sets $\{V_1, V_2, \ldots, V_g\}$, to find $G_{sub}$, a connected subgraph of $G_i$ that satisfies $\forall V_i, V_i \cap V(G_{sub}) \neq \emptyset$ and has the lowest graph weight. It is the group Steiner problem (Reich and Widmayer, 1990). Considering that the group Steiner problem is NP-Complete (Reich et al., 1991), we use an approximation algorithm in (Ihler, 1991, Section 3) to deal with it.

**Algorithm 2.** The algorithm GroupSteiner for computing a compact sub-snippet in a weighted graph

<table>
<thead>
<tr>
<th>Line</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>for each $v_1 \in V_1$ do</td>
</tr>
<tr>
<td>2</td>
<td>foreach $V_i \in {V_2, \ldots, V_g}$ do</td>
</tr>
<tr>
<td>3</td>
<td>for each $v_i \in V_i$ do</td>
</tr>
<tr>
<td>4</td>
<td>$p_{v_i, v_j} := \text{find_a_shortest_path}(G_i, v_i, v_j)$;</td>
</tr>
<tr>
<td>5</td>
<td>end</td>
</tr>
<tr>
<td>6</td>
<td>$p_{v_i, v_j} := \min_{v_i \in V_i} p_{v_i, v_j}$;</td>
</tr>
<tr>
<td>7</td>
<td>end</td>
</tr>
<tr>
<td>8</td>
<td>$G_{v_i} := \bigcup_{j=2}^{g} p_{v_i, v_j}$;</td>
</tr>
<tr>
<td>9</td>
<td>end</td>
</tr>
<tr>
<td>10</td>
<td>$G_{G_i} := \min_{v_i \in V_i} [Wt_c(G_{v_i})]$;</td>
</tr>
<tr>
<td>11</td>
<td>$G_{sub} := \text{find_a_spanning_tree}(G_{v_i})$;</td>
</tr>
<tr>
<td>12</td>
<td>return $G_{sub}$;</td>
</tr>
</tbody>
</table>

Algorithm 2 computes a compact sub-snippet in a weighted graph. In lines 1–9, for each node $v_1$ in $V_1$, a shortest path (Dijkstra’s algorithm) from $v_1$ to each other node set is found, and let $G_{v_i}$ be the union of these paths, i.e. merge them into a graph. Among all these graphs, line 10 finds the one with the lowest weight. Finally in line 11, the spanning tree of this graph is computed as the sub-snippet to be outputted.

Ihler (1991) proves that the generated sub-snippet is sub-optimal, i.e., the graph weight of this sub-snippet is less than $g – 1$ times the weight of the optimal sub-snippet, where $g$ is the number of required node sets. Next we will show that its time complexity is polynomial. Suppose that the graph has $n$ nodes and $m$ edges. The function find_a_shortest_path can be computed in $O(n \log n + m)$ steps. It is executed $n_1 \times (n_2 + \ldots + n_g)$ times in the algorithm and $n_1 = |V_i| < n$. So the overall time complexity of lines 1–9 is $O(gn^2(n \log n + m))$, which is evidently the worst-case time complexity of the entire algorithm.

For example, when the parameter $\alpha$ is set to 0.8, for each maximal 1-radius subgraph in Fig. B.3, the sub-snippet generated by our algorithm is shown in the dashed ellipse.

6. Snippet generation

So far, we have generated all sub-snippets for an ontology, each one is a connected graph connecting query-relevant terms and with a small graph weight. Next, we consider how to select some of them to form the final snippet.

6.1. Requirements for snippet

As pointed out in the Introduction, a good snippet for ontology search should be with term-association view. That is, a snippet is a subgraph of a TAG. Besides, for every pair of keywords in the query, we hope to find a close relation ($\leq 2r$, if any) between two terms corresponding to the pair of keywords.

Furthermore, to be efficient for reading and judgment, we hope each snippet is compact. That is, the snippet should be with a small graph weight. We introduce a normalized metric named compactness (i.e. the reduction ratio of its original TAG). Given a TAG $G$, the compactness of the snippet $G_s$ is

$$\text{Compactness}(G_s, G) = 1 - \frac{Wt_c(G_s)}{Wt_c(G)}$$

It should be stressed here that for a given TAG, $Wt_c(G)$ is a constant. Thus a snippet with a smaller graph weight must be more compact.

Finally, a good snippet should be biased towards the query, i.e. a snippet matches more keywords are preferred. This is achieved by defining a normalized metric named snippet relevance. Specifically, given a snippet $G_s$ and a set of keywords $Q$, the snippet relevance is

$$\text{Relevance}(G_s, Q) = \frac{|\{q \in Q \land (\exists v \in V(G_s), q \in \text{Lbl}_v(v))\}|}{|Q|}$$
We incorporate both snippet relevance and snippet compactness into a single quality function:

\[
\text{Quality}(G_s, G, Q) = \beta \ast \text{Compactness}(G_s, G) + (1 - \beta) \ast \text{Relevance}(G_s, Q)
\]  

Since sub-snippets generated in Section 5 are with term-association view and compact, we consider how to select some of them into the final snippet without violating the length constraint (the number of terms).

6.2. Sub-snippet assembling

Suppose that all sub-snippets are with disjoint node sets (and hence disjoint edge sets), and we only consider the compactness as the quality metric. Then the task becomes to select some sub-snippets so that the number of distinct terms is within a given restriction and the total quality is as large as possible. This is a 0–1 knapsack problem, which is NP-complete. Obviously, our problem is at least as hard as it. Therefore, we use a greedy approach. At each loop, the algorithm selects a sub-snippet whose addition will have a maximal marginal increase in the snippet quality, while the total number of terms in the snippet is not larger than the limit. If there is a sub-snippet, it is merged into the final snippet. If no, the loop is terminated.

For example, we set parameters \( \alpha = 0.8 \) and \( \beta = 0.5 \), and the number of terms is not larger than 5. For the query “conference, staff, person”, the snippet quality of five sub-snippets in Fig. B.3 are 0.758, 0.667, 0.758, 0.733, 0.667 respectively. Thus, the sub-snippet in Fig. B.3a is selected in the first loop. Next, sub-snippets in (e and c) are selected, successively. The final result is shown in Fig. B.4.

7. Evaluation

This section gives an empirical study on the implementation of our approach to demonstrate its feasibility. Then a questionnaire-based usability evaluation is performed to compare our approach with state-of-the-art snippet generation methods.

7.1. Feasibility

To demonstrate the feasibility of the proposed approach in Web-scale ontology search, we performed extensive experiments based on 4522 ontologies, each including at least one class or property as its constituent term. On average, each ontology consists of 554 terms described by 2179 RDF triples, or 1917 RDF sentences.

In our implementation, for each ontology, we generate a TAG. Next we extract all maximal 1-radius subgraphs from each TAG. Parameters \( \alpha \) and \( \beta \) in Eqs. 1 and 6 are set to 0.8 and 0.5, respectively. Maximal 1-radius subgraphs are computed offline and stored in a database. Moreover, two kinds of inverted indices are used to support fast retrieval: one maps keywords to subgraphs and the other maps keywords and subgraphs to terms. Sub-snippet generation and assembling are computed online. All experiments are performed on a IBM server with two 4-core Xeon E7400 (2.4G) and 24GB memory.

7.1.1. Usability of maximal 1-radius subgraphs

All TAGs are split into 1,199,841 maximal 1-radius subgraphs, that is, each TAG is extracted into 265 maximal 1-radius subgraphs on average. Obviously, this number is smaller than the average number of nodes (554) in a TAG. With a multithreading (10 threads) JAVA program in the server, computing all maximal 1-radius subgraphs cost about three days.
Next, we show some comparison on the scale of TAGs and their maximal 1-radius subgraphs to demonstrate the usability of subgraph extraction for reducing the graph scales.

On average, a TAG contains 554 nodes and 657 edges. 797 TAGs have less than 10 nodes; 2543 TAGs have [10, 100) nodes; 1044 TAGs have [100, 1000) nodes; 98 TAGs have [1000, 10,000) nodes; and only 40 TAGs have more than 10,000 nodes. In contrast, each maximal 1-radius subgraphs have 4.75 nodes and 38 edges on average. 422,971 (35%) subgraphs have only one node. 1,198,486 (99.89%) maximal 1-radius subgraphs have no more than 100 nodes. Besides, 1355, 182 and 11 subgraphs have more than 100, 1000 and 10,000 nodes, respectively.

Comparing the scales of TAGs and maximal 1-radius subgraphs, we find that the graph scale declines markedly no matter in the average or the maximal scale. This means that time-consuming graph algorithms (e.g. group Steiner algorithm) can become efficient after extracting subgraphs. Moreover, to speed up the sub-snippet generation, we try to cache shortest paths between all pairs of nodes in maximal 1-radius subgraphs. If we suppose that reading this cache takes constant time (\texttt{find shortest path in Algorithm 2} is constant time), the time complexity for sub-snippet generation becomes \(O(gn^2)\). Because there may be tremendous pairs of nodes (\(n^2\) at most) for large graphs, we only cache graphs with not more than 1000 nodes.\(^{14}\) And for these large graphs, we just list terms matched by keywords, and ignore finding term relations. Furthermore, for small graphs (less than 100 nodes), generating sub-snippets without cache is not very time-consuming. Therefore, we cache only subgraphs whose node number is between 100 and 1000 by trading off between time and space.

7.1.2. Response time

To evaluate the time cost of generating snippets, for each ontology we randomly select 1–10 keywords from local names and labels of its terms. Fig. B.5 presents the average execution time of generating a snippet for different scale ontologies with varying number of keywords. Obviously, the runtime is proportional to the ontology scale and the number of keywords. Most ontologies (97%, less than 1000 terms) can be computed within about 1 second on average, even if 10 keywords are fed. As snippets can be computed in parallel (multi-threading in Java), we can generate snippets for all matched ontologies in a page effectively. For few large ontologies (138, more than 1000 terms), generating a snippet needs 4–10 s on average when more than three keywords are inputted. The long average response time is caused by some queries on several large ontologies, where queries contain digits and special characters, and thus many maximal 1-radius subgraphs and terms are hit. As these queries are not likely to be issued by users, the actual response time for large ontologies and long queries is not necessarily unacceptable.

7.2. Effectiveness

To evaluate the effectiveness of the proposed approach, we implement our approach as well as several state-of-the-art snippet generating methods. For a query-relevant ontology, several snippets are generated by applying all these methods. We invited people to evaluate these snippets. Their opinions were collected in the way of completing questionnaires.

7.2.1. Competitors

All snippet generation methods in ontology search systems and literatures can be divided into three categories.

\(^{14}\) Only 182 (0.015%) subgraphs are with more than 1000 nodes, and they come from 45 ontologies, which are specific knowledge-intensive domains such as medicine, biology, and knowledge management.
Term set. This line of work views ontologies as a set of terms, and selects some important and query-relevant ones to form the snippet, such as Swoogle (Ding et al., 2005) and Watson (d’Aquin et al., 2007).

Sentence set. This line of work views ontologies as a set of RDF sentences (or triples), and selects important and query-relevant ones to form the snippet, (e.g., Zhang et al., 2007; Penin et al., 2008).

Term association. Our work is the first to propose the term-association view snippet, i.e. not only query-relevant terms but also their relations.

We have implemented the methods of (Ding et al., 2005) (Term.S), (d’Aquin et al., 2007) (Term.W), (Zhang et al., 2007) (Sent) and (Penin et al. (2008)) (Sent + Q). Moreover, we have implemented our proposed approach (TA + C, i.e. term association view and compactness). Besides, to evaluate the importance of compactness, we directly select and assemble some query-relevant maximal r-radius subgraphs into the snippet, that is, ignore the procedure to generate compact sub-snippets using a group Steiner algorithm. We call this approach TA. Finally, to evaluate the validity of the preference of term associations (edge weight) in the compactness, we set the weights of all edges in TAGs to 1, and call this approach TA + C0.

7.2.2. Participants
A total of 30 people having experience of ontology-based research or development were invited to participate in the evaluation. All these participants come from two Chinese universities and two Chinese companies.

7.2.3. Experimental design
From all 15 topics listed in ODP, the evaluation system randomly selects 10 of them. For a given topic, each participant is supposed to search ontologies about this topic for reasoning, mapping, extending or reusing. To broaden the views of participants, the system also provides 10 popular sub-topics under this topic. Each participant is free to arbitrarily feed query keywords about this topic to our evaluation system. For each query, the system finds a query-relevant ontology and generates seven snippets (one snippet for each method). To eliminate the impact of the display order, the system ranks these snippets in random order. Besides, to be fair, all snippets are shown in graphic formats and with no more than five nodes.

A questionnaire is provided for them to collect their satisfactions on these methods. The questionnaire, which is derived from two existing usability questionnaires (Brooke, 1996; Chin et al., 1988), consists of four Likert items (in Table B.2a). Each Likert item is a statement with a degree of agreement to be specified. The participant would provide the degree of her satisfaction from the snippets presented in this search in the range [1,5]. Statement 1 (Organization) tries to know their satisfaction in different snippet views, i.e. term set, sentence set, and term association. Statement 2 (Reading) tends to know the ease for reading and understanding of the snippet. Statement 3 (Judgement) asks the instructiveness for the relevance judgement. Statement 4 (Overall) asks their overall experience.

7.2.4. Experimental results
We received a total of 295 searches from all the participants, and the correspondence between the number of keywords in a query and the number of queries is listed in Table B.2b. That is, most queries are with two or three keywords, and only nine queries has more than five keywords. The longest query contains 11 keywords.

Fig. B.6a summarizes the average satisfaction in four aspects for different methods. According to the Organization, users prefer the sentence-set view and our term-association view. Both of them are much better than the term-set view. This means that snippets with isolated terms are not preferred by users. Furthermore, all snippets have good experiences in

---

**Table B.2**
(a) Statements in the usability questionnaire. (b) Number of keywords in a query vs. the number of queries.

<table>
<thead>
<tr>
<th>(a) Statements</th>
<th>Abbr.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. The information organization of the snippet is pleasant</td>
<td>Organization</td>
</tr>
<tr>
<td>2. I spend a little time on reading the snippet</td>
<td>Reading</td>
</tr>
<tr>
<td>3. I can confidently judge whether the ontology satisfies my requirement by using the snippet</td>
<td>Judgement</td>
</tr>
<tr>
<td>4. Overall, I am satisfied with the snippets</td>
<td>Overall</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>(b) #Keywords in a query</th>
<th>#Queries</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>47</td>
</tr>
<tr>
<td>2</td>
<td>104</td>
</tr>
<tr>
<td>3</td>
<td>113</td>
</tr>
<tr>
<td>4</td>
<td>18</td>
</tr>
<tr>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>≥ 6</td>
<td>9</td>
</tr>
</tbody>
</table>

---

16 http://ws.nju.edu.cn/ssos/eval/.
17 In Sent or Sent+Q, top ranking sentences are added if a snippet has more than five nodes.
18 One participant only finished 5 searches.
the Reading. This means that displaying term relations will not bring burdens for the reading. Besides our method has a better judgement, as it shows compact and to-the-point snippets. Therefore, the overall satisfaction of ours is the highest. Moreover, for two methods in the term-set view, Term.W is better than Term.S. This means that listing all query-relevant terms has a better experience than listing only one query-relevant term and some query-independent terms. Finally, the query-independent summarization (Sent) is almost with the lowest satisfaction for all aspects, which demonstrates that the query relevance is the essential factor for the snippet.

Fig. B.6b presents the average overall degrees of users’ satisfaction from the snippets by submitting queries consisting of varying number of keywords. For single-keyword queries, our approach (TA + C) has the worst performance. This is because only several (usually only one) terms that are matched with the query keywords and in different sub-snippets are presented in the snippet. On the contrary, TA provides one or more maximal 1-radius subgraphs containing query-relevant terms, which carries more information than only several query-relevant terms. As there are 47 (15.9%) single-keyword queries, how to improve our method for this case is our further work. But, when the number of query keywords is increasing, the average degrees of users’ satisfaction from the snippets provided by TA + C is always the highest (except the number is 5). It is because, with more and more query keywords submitted, the snippet generated by our approach is compact for reading, and it also has more space to contain more query-relevant terms and their relations. This demonstrates that the query relevance is the essential factor for the snippet.

Significant test shows that the differences between TA + C and TA + C in overall satisfaction are statistically insignificant. That is, when the significance level is 0.05, the hypothesis that there is no difference between two approaches is accepted. Although differences between two approaches are statistically insignificant, we argue that the preferences of term associations take effect. This is because that both methods have the same effects in most situations, but if there are differences in the snippet, TA + C is almost always better than TA + C. Specifically, in all 295 searches, we find that TA + C and TA + C have the same effects in 244 searches. Manual inspections show that there is only one schema sentence (Section 3.2) connecting each pair of terms in these situations, which causes all preferences of term associations in every snippet are with the same value. While, in the left 51 searches, TA + C has a higher degree of overall satisfaction in 44 searches, and TA + C is better in only 7 searches. This means that the preferences of term associations can be used to help users to inspect snippets.

7.3. Parameter selection

In the previous experiments, we set \( \alpha = 0.8 \) and \( \beta = 0.5 \) in our approach. The reason that we set \( \alpha = 0.8 \) is to indicate that users focus more on schema sentences than data-describing sentences in ontology search. We have discussed it in Section 3.2. In this section, we will report an experiment to analyze the sensitivity of different values for the parameter \( \beta \) on the degree of users’ satisfaction.

7.3.1. Competitors

We implemented five variants of TA+C by setting the parameters in Eq. 6 to different values (\( \beta = 0, 0.25, 0.5, 0.75, 1 \)). \( \beta = 0 \) indicates that the compactness is not considered in the quality metric. Similarly, \( \beta = 1 \) indicates that the relevance is ignored in the quality metric.

---

19 As queries with 6, . . . , or 11 keywords are not statistical significant, we merge them together and refer them as “\( \geq 6 \)".
7.3.2. Participants
We invite 27 people having experience of ontology-based research or development to participate in the evaluation. More than half of them also participate in the previous experiment. All these participants come from three Chinese universities and four Chinese companies.

7.3.3. Experimental design
Five variants of TA+C with different parameters are evaluated in the same evaluation system as the previous experiment. The difference is that participants only need to provide the degree of their overall satisfaction, i.e. only Statement 4 in the previous experiment.

7.3.4. Experimental results
Fig. B.7 shows the sensitivity of $b$ on the average degree of overall satisfaction. We find that TA+C has the highest average degree of overall satisfaction when $b = 0.25$, and it is the worst when only using the compactness as the quality metric ($b = 1$). Besides, the degree of satisfaction of TA+C with $b = 0$ is close to that of TA+C with $b = 0.25$. Indeed, in all 270 searches, approaches with $b = 0$ and 0.25 have the same degree of satisfaction in 225 searches. While, in the left 45 searches, $b = 0.25$ has a higher degree of overall satisfaction in 34 searches, and $b = 0$ is better in 11 searches. Therefore, we argue that the compactness also takes effects in the quality metric, and we recommend that $b = 0.25$.

8. Conclusion and future work
In this paper, we study the snippet generation issue in keyword-based ontology search engines. We illustrate that term-association view snippets are preferred by users, and the compactness should also be considered for reading and judgement. Next, we present an approach to generate snippets for ontology search, and integrate it into a real online ontology search engine.\footnote{http://ws.nju.edu.cn/ssos/}. The technical contributions of this paper are summarized as follows:

1. A model of term association graph (TAG) is proposed to model associations between terms. The preference of term associations is incorporated into the edge weight of the graph.
2. To preserve relations between closely-related terms and downsize the scale of the TAG for subsequent treatments, an algorithm is proposed to divide each TAG into some maximal $r$-radius subgraphs.
3. For a query-relevant maximal $r$-radius subgraph, we use an approximation algorithm of the group Steiner problem to generate a compact sub-snippet. A greedy approach is used to select and assemble some sub-snippets to form a term-association view snippet considering the query relevance and the compactness.
4. An empirical study on our implementation shows that our approach is feasible. An evaluation on usability shows that the term-association view snippet is favored by users, and the compactness is useful for reading and judgment.

For future work, we are interested in other approximation algorithms to generate sub-snippets. Besides, it is also an interesting research work to find other algorithms to assemble sub-snippets into the final result.
Acknowledgements

This work is supported by the National Natural Science Foundation of China (NSFC) under Grants 60973024, 61170068 and 61100040, and it is also supported by National Social Science Foundation of China under Grant 11AZD121. Firstly, we would like to express our sincere appreciation to the anonymous reviewers for their insightful comments, which have greatly aided us in improving the quality of the paper. Secondly, we would like to thank Dr. Yanbing Wang for his valuable suggestion on approximation algorithms of group Steiner problem. We are also grateful to Huiyao Wu and Jianfeng Chen for their effort in experimentation. Finally, we also thank all the participants in the usability evaluation for their time and effort.

Appendix A. Proof of Theorem 1

Proof. First, all subgraphs generated by MRR_extraction are r-radius subgraphs. Because line 3 ensures that for each subgraph Gi, there exists a node vi whose distance to any other node in Gi is within r, i.e., its eccentricity ε(vi) ≤ r. Then, rad(Gi) = min_{v ∈ V(Gi)} ε(v) ≤ ε(n) ≤ r.

Next, we prove that all r-radius subgraphs in the result are maximal. That is, for each subgraph Gi in the result, there is no other subgraph Gj in the result such that Gi is a subgraph of Gj. Lines 8-21 ensure that V(Gi) ⊆ V(Gj). Besides, Gj is a subgraph of G induced by Vj (line 4). That is, for any pair of nodes vj and vj of Gj, if (vj, vj) is an edge in G, then (vj, vj) is an edge in Gj. Thus E(Gi) ⊆ E(Gj). So Gi is maximal.

Later, we prove that all maximal r-radius subgraphs of the graph G are included in the result. Suppose Gi is a maximal r-radius subgraph that are not included in our result. According to the definition, rad(Gi) ≤ r, i.e., ∃ v′ ∈ V(Gi), ε(v′) ≤ r. For the node v′, lines 3-4 generate a subgraph Gi. Because line 3 computes all nodes that v′ can reach in r steps, V(Gi) ⊆ V(Gj). Moreover, as Gi is a subgraph of G induced by V(Gi), E(Gi) ⊆ E(Gj). Then, Gi is a subset of G. Although Gi may be excluded from the result in line 8-21, in this situation there is another graph Gj such that Gi is a subgraph of Gj. So, there must exist a graph in the result such that Gi is a subgraph of it. Therefore, Gi is not a maximal r-radius subgraph.

Finally, we will prove that its time complexity is polynomial. Suppose that the graph G has n nodes and m edges. The function breadth_first_search can be computed in O(n + m) steps. It is executed n times. So the time complexity of line 2-6 is O(n(n + m)). Line 13 checks whether a node is contained in a set. If the set is a hashset, this can be done in constant time. So the time complexity of lines 12-17 is O(n). Therefore, the time complexity of lines 8-28 is O(n^3), which is the worst-case time complexity of the entire algorithm. □

Appendix B. Proof of Theorem 2

Proof.

1. As Gi is a maximal r-radius subgraph, the radius of (Gi) is less than or equal to r. Therefore, there exists a node v whose eccentricity is less than or equal to r. For every two nodes vi, vj in Gi, d(v, vi) ≤ r and d(v, vj) ≤ r. So, d(v, vi) ≤ d(v, vj) + d(vj, vi) ≤ 2r.

2. If the distance between vi and vj is less than or equal to 2r, there exists a node v which has a path to vj within r steps and has a path to vi within r steps. As our algorithm generates a r-radius subgraph from v, that is, all nodes within r steps to v are included in the graph. Thus, the nodes vi and vj must be in some maximal r-radius graph. □

References


