Dynamic Documents Indexing in Evolving Contexts

Rolf Nossum[1] and Vladimir Oleshchuk[2]

[1] Department of Mathematics, Agder University College
Serviceboks 422, N-4604 Kristiansand
[2] Department of Information and Communication Technology
Agder University College, Grooseveien 36, N-4876 Grimstad
NORWAY
{rolf.nossum,vladimir.oleshchuk}@hia.no

Abstract. In this paper we consider similarity between documents in contexts defined by ontologies. Two documents have different similarity degree depending on the context in which they are considered. We consider a scenario where context evolves incrementally, necessitating continual revision of similarity measures between documents. We develop algorithms that revise similarity between documents concurrently with operations updating the ontology.

Key words – evolving ontologies, similarity degree, graph-based representation.

1 Introduction

In this work we consider ontologies as knowledge structures that identify concepts, properties of concepts and relations among them to enable share and reuse of knowledge. Ontologies collect and organize terms of references. In that sense we consider ontologies as context descriptions. Ontologies can be presented as graphs that reflect structural and semantic relationships between concepts.

Documents articulation over ontologies means finding the linkage between ontology concepts and document contents. When ontology concepts are related to specific terms, then articulation can be seen as syntactical or structural matching between ontologies and the contents of documents. Such articulations of documents can be used to index documents in the context defined by a given ontology. Since ontology defines many concepts on different abstraction levels, we can specify abstraction levels at which documents are similar. Knowing the connection between documents on different abstraction levels we can implement highly efficient search algorithms.

In real applications however, context will often change dynamically. The degree of similarity between documents will change when context changes. Considering a large number of documents, it can become prohibitively expensive to recalculate similarity every time context is modified. Therefore we seek to develop an incremental solution, where recalculations affect only relations that have been changed. The purpose of our work is to improve efficiency of information retrieval and reduce the workload imposed by dynamically changing contexts.

In this paper we describe a solution to the problem described above. Starting with an algorithm which calculates similarity degrees in context (we shall use the one described in [5]), we give a method for keeping track of document articulations incrementally, defined inductively on an input load of ontology manipulation primitives.

2 Graphical Representation of Contexts

We assume that contexts are given in the form of ontologies. In this work we assume that an ontology O is presented as a directed labeled graph $G = (V, E)$ where vertices from V are labeled by elements from T and edges from E are labeled by elements from R. We denote such an ontology as $O = G(T, R)$.

The mappings of vertices V and edges E are defined by surjective functions $\tau_O : V \to T$ and $\rho_O : E \to R$ respectively, that is, $\tau_O(V) = T$ and $\rho_O(E) = R$.

Let T be a set of terms or concepts (for example, it can be a noun-phrase in some language), and R a set of pre-defined semantic relations among the terms such as `instance-of`, `subset-of`, `attribute-of`, `member-of-group` etc. The mappings τ_O and ρ_O describe how terms from T is associated with vertices of G and how edges from G are labeled by relations from R.

By choosing a set of concepts or terms T and a set of semantic relations R, every $G\,(T, R)$ can be interpreted as an ontology or context description. For example, one of the most common relations in the specification of ontologies is hyponymy, also called the `is_a` relation. This is a transitive and asymmetric relation that defines a hierarchical structure between more specific and more general concepts, where terms inherit all characteristics from their ancestor terms. We take R to represent hyponymy-like relations, and then from $(a, b) \in R$ it follows that b represents a more general concept then a.

We define a sub-ontology relation $\sqsubseteq$ similar to the subgraph relation. Let $O_i = G_i\,(T_i, R_i)$ where $G_i = (V_i, E_i)$, $i = 1, 2$ be two ontologies.

We say that O_1 is a sub-ontology of O_2 (denoted $O_1 \sqsubseteq O_2$) if the following properties are satisfied:

- Graph G_1 is a subgraph of graph G_2;
- $T_1 \subseteq T_2$ and $R_1 \subseteq R_2$;
- $\tau_{O_1} \subseteq \tau_{O_2}$ and $\rho_{O_1} \subseteq \rho_{O_2}$

3 Semantic Similarity between Documents

In this section we introduce some new notation, give a formal definition of semantic similarity, and explain how we can calculate similarity between two documents at different abstraction levels, based on context defined in the form of ontology.

3.1 Document articulation

In order to compare documents with respect to a given ontology, we have to find their articulations with respect to that ontology. In this section we present a modified version of the algorithm from [5]. A document t is to be articulated with respect to an ontology $O = G(T, R)$ where $G = (V, E)$. The articulation of t with respect to O is a sub-ontology denoted O_t such that $O_t \sqsubseteq O$. Let $term(t, O)$ denote the set of terms from T that occur in t, that is, $term(t, O) \subseteq T$.

Input: A document t, ontology $O = G(T, R)$ where $G = (V, E)$.
Output: Document articulation O_t.
Algorithm: $TextArticuation$
Step 1: Select $V_t \subseteq V$ such that $\tau_O(V_t) = term(t, O)$.
Step 2: Let $G_t = (V_t, E_t)$ be the subgraph of $G = (V, E)$ spanned by V_t.
Step 3: Expand G_t upwards as follows:
  $V'' \leftarrow \{v \mid (u, v) \in E \wedge v \notin V_t\}$
  while $V'' \neq \emptyset$ **do**
    $E \leftarrow \emptyset$; $Buf \leftarrow \emptyset$
    for all $u \in V''$ and $v \in V$ such that $(u, v) \in E$ **do**
      $E'' \leftarrow E'' \cup \{(u, v)\}\,; Buf \leftarrow Buf \cup \{v\}$
    $V_t \leftarrow V_t \cup V''$; $E_t \leftarrow E_t \cup E''$; $V'' \leftarrow Buf$
Step 4: Define O_t as $G_t(T_t, R_t)$ where
  $G_t = (V_t, E_t)$, $T_t = \tau_O(V_t)$, $R_t = \rho_O(E_t)$
  $\tau_{O_t} = \tau_O|_{V_t}$ is a restriction τ_O to V_t
  $\rho_{O_t} = \rho_O|_{E_t}$ is a restriction ρ_O to E_t

3.2 Assignation of abstraction levels

The articulation of a document t relative to an ontology $O = G(T, R)$ with $G = (V, E)$ is in general a forest of trees, i.e. a sub-ontology of O. Parent vertices represent more abstract concepts than their children, and root vertices represent the most abstract concepts. Abstraction levels are assigned inductively as follows:

- All root vertices have abstraction level 0
- Every non-root vertice has abstraction level 1 more than its parent.

We denote the abstraction level of a vertex $v \in V$ as $level(v)$, that is

$$level : V \rightarrow \{0, 1, 2, ...\}$$

3.3 Computation of similarity

We are now ready to define the similarity of two documents t_1, and t_2 relative to an ontology $O = G(T, R)$ where $G = (V, E)$. Let $O_{t_i} = G_{t_i}(T_{t_i}, R_{t_i})$, $i = 1, 2$ denote articulations of t_1 and t_2 with respect to O. Similarity is measured as a number between 0 and 1, at each level of abstraction. Each number is calculated as the ratio of the number of common terms to the total number of terms, at the relevant abstraction level.

Assuming that $G_{t_i} = (V_{t_i}, E_{t_i})$, let V_i^j be the set of vertices at abstraction level j in the articulation of t_i relative to O, let m_i be the highest value of j such that $V_i^j \neq \emptyset$, $i = 1, 2$, and let $m = \min(m_1, m_2), M = \max(m_1, m_2)$.

The similarity of t_1 and t_2 relative to O is a vector $Sim(t_1, t_2, O) = \langle s_0, \cdots, s_M \rangle$ where

$$s_j = \frac{\left|V_1^j \cap V_2^j\right|}{\left|V_1^j \cup V_2^j\right|} \quad \text{for } 0 \leq j \leq m$$

$$s_j = 0 \qquad \text{for } m < j \leq M$$

4 Evolution of Ontologies

In this section we describe operations that can be applied to modify context (ontology) such as: add concept, remove concept, add relation between existing concepts, remove relation between existing concepts. We

also describe how these changes of context influence document articulations. In order to efficiently track changes of document articulations (caused by ontology modification) that may affect existing similarity relations between documents, we introduce a set P_t that used to collect abstraction levels where changes occur.

4.1 Adding a concept

When a new concept a is added into the ontology $O = G(T, R)$, it is initially added as a new vertex unconnected to any other vertices. That gives rise a modified ontology $O' = G'(T \cup \{a\}, R)$ where $G' = (V', E)$, $V' = V \cup \{v'\}$ such that $v' \notin V$ and $\tau_{O'} = \tau_O \cup (v', a)$. Suppose the newly added concept a represents a term that occurs in a document t, that is, $a \in term(t, O')$. Then the sub-ontology $O_t = G_t(T_t, R_t)$ which articulates t with respect to O' must be augmented by the concept a at abstraction level 0 as described in the following algorithm *AddConcept*.

Input: A document t, an ontology O, an articulation O_t, a new concept a
Output: A revised articulation O_t and P_t
Algorithm: *AddConcept*
Let $O = G(T, R)$ where $G = (V, E)$, and $O_t = G_t(T_t, R_t)$ where $G_t = (V_t, E_t)$.
if $a \in term(t, O)$ and $a \notin term(t, O_t)$ **then**
$V_t \leftarrow V_t \cup \{v\}$ where $v \notin V_t$
$T_t \leftarrow T_t \cup \{a\}$
$\tau_{O_t} \leftarrow \tau_{O_t} \cup (v, a)$; $P_t \leftarrow \{0\}$

4.2 Adding a relation between concepts

When a new semantic relation r between concepts a and b is to be inserted into an ontology $O = G(T, R)$ where $G = (V, E)$, it is assumed that $r \in R$, $a, b \in T$, and there are vertices $u, v \in V$ such that $\tau_O(u) = a$ and $\tau_O(v) = b$. This gives rise to a modified ontology $O' = G'(T, R \cup \{r\})$ where $G' = (V, E \cup \{(u, v)\})$, $\rho_{O'} = \rho_O \cup \{((u, v), r)\}$, and necessitates revision of document articulations and similarity vectors.

Let us first consider revising an existing articulation O_t of a document t with respect to O, so that it becomes the articulation of t with respect to O'. Given a new edge (u, v), we assume that $level(u) \geq level(v)$.

Input: A document t, an ontology O' augmented with a new concept r that connects concepts a and b, O_t is an articulation t with respect to O.
Output: A modified articulation O_t and P_t
Algorithm: *AddRelation*
Let $O' = G(T, R)$ where $G = (V, E)$ and $O_t = G_t(T_t, R_t)$ where $G_t = (V_t, E_t)$
Let u and v be such that $\tau_{O'}(u) = a$ and $\tau_{O'}(v) = b$ and $\rho_{O'}(u, v) = r$
$Buf1 \leftarrow \{(u, v)\}$; $Buf2 \leftarrow \emptyset$; $P_t \leftarrow \emptyset$
while $Buf1 \neq \emptyset$ **do**
  for all $(\alpha, \beta) \in Buf1$ **do**
    $E_t \leftarrow E_t \cup \{(\alpha, \beta)\}$;
    $\rho_{O_t} \leftarrow \rho_{O_t} \cup \{((\alpha, \beta), \rho_O(\alpha, \beta))\}$
    if $\beta \notin V_t$ **then**
      $V_t \leftarrow V_t \cup \{\beta\}; Buf2 \leftarrow Buf2 \cup \{(\beta, \delta) \mid (\beta, \delta) \in E\}$;
      $P_t \leftarrow P_t \cup \{level(\beta)\}$
  $Buf1 \leftarrow Buf2; Buf2 \leftarrow \emptyset$

4.3 Removing a relation between two concepts

Removing a relation r connecting concepts a and b from an ontology $O = G(T, R)$ means removing an edge (u, v) from the graph $G(V, E)$ where $\tau_O(u) = a$, $\tau_O(v) = b$ and $r \in \rho_O(u, v)$. It gives rise to $O' = G'(T, R')$ where $G' = (V, E \setminus \{(u, v)\})$, and if $|\rho_O(E)| = 1$ then $R' = R' \setminus \{r\}$ otherwise $R' = R$. It necessitates revising an existing articulation $O_t = G_t(T_t, R_t)$ of a document t with respect to O'. We have to take into consideration all edges from E_t of $G_t = (V_t, E_t)$ that have been included into O_t as the result of expansion caused by inclusion of the relation r connecting a and b into O. More generally, we should exclude those edges those occurrence in O_t caused by occurrence an of edge (a, b). To simplify presentation, we introduce two sets $BUF_t(\alpha) = \{\beta | (\alpha, \beta) \in E_t$ and $level(\alpha) \geq level(\beta)\}$ and $BUF^t(\beta) = \{\alpha | (\alpha, \beta) \in E_t$ and $level(\alpha) \geq level(\beta)\}$.

Input: Document t, a revised ontology O' where relation r connecting concepts a and b is removed, O_t is an articulation of t with respect to O
Output: Revised articulation O_t with respect to O' and P_t
Algorithm: $RemoveRelation((a, b), O_t)$
Let $O_t = G_t(T_t, R_t)$ where $G_t = (V_t, E_t)$
Let u and v be such that $\tau(u) = a$ and $\tau(v) = b$

$E_t \leftarrow E_t \setminus \{(u, v)\}$
$\alpha \leftarrow v$; $P_t \leftarrow \emptyset$
while $BUF^t(\alpha) \neq \emptyset$ **do**
  for all $\beta \in BUF^t(\alpha)$ **do**
    if $|BUF_t(\beta)| = 1$ **then**
      $E_t \leftarrow E_t \setminus (\alpha, \beta)$
      $\rho_{O_t} \leftarrow \rho_{O_t} \setminus \{(\alpha, \beta), \rho_{O_t}(\alpha, \beta)\}$
      if $\alpha \notin term(t, O')$ **then** $V_t \leftarrow V_t \setminus \{\alpha\}$
      $P_t \leftarrow P_t \cup \{level(\alpha)\}$
  $\alpha \leftarrow \beta$

4.4 Removing a concept

Removing a concept a from $O = G(T, R)$ gives rise to $O' = G'(T', R')$ and $G' = (V', E')$ where $V' = V \setminus \{v\}$, $E' = E \setminus \{(\alpha, \beta) \mid \alpha = v \text{ or } \beta = v\})$, $\tau_O(v) = a$, $T' = \tau_O(V')$ and $R' = \rho_O(E)$. It necessitates revising an existing articulation $O_t = G_t(T_t, R_t)$ of a document t with respect to O.

Input: A document t, an ontology O' with removed concept a, an articulation O_t with respect to O, a concept a
Output: A revised articulation O_t with respect to O' and P_t
Algorithm: *RemoveConcept*
Let $O_t = G_t(T_t, R_t)$ with $G_t = (V_t, E_t)$
Let u be such that $\tau(u) = a$
$V_t \leftarrow V_t \setminus \{u\}$;
$\tau_{O_t} \leftarrow \tau_{O_t} \setminus \{(u, a)\}$
$P_t \leftarrow \{level(u)\}$
for all $(\alpha, \beta) \in E_t$ such that $\alpha = u$ or $\beta = u\}$) **do**
  $(O_t, P'_t) \leftarrow RemoveRelation((\alpha, \beta), O_t)$
  $P_t \leftarrow P_t \cup P'_t$

5 Maintaining similarity vectors incrementally

Similarity vectors represent relations between documents. We consider two documents t_1 and t_2 where O_{t_1} and O_{t_2} are articulations of these documents relative to an ontology O. The ontology O can be modified by applying the operations described in previous subsections. Not every modification of O necessarily causes changes in O_{t_1} and O_{t_2}. However, only modifications that change the number of vertices will cause modification of the similarity vector. We use a set P_t to represent the abstraction levels at which the number of vertices were changed in the articulation O_t.

Generally, when an ontology O is modified as a result of using any operation described in the former section, the set P_t contains those abstraction levels of articulation O_t that were modified. It means that similarity vector $Sim(t_1, t_2, O)$ should be revised. Given two documents t_1 and t_2, the following algorithm describes similarity vector revision in the case of applying any of the operations of adding or removing a vertex or an edge of ontology O.

Input: Articulations O_{t_1} and O_{t_2} with corresponding sets P_{t_1} and P_{t_2} and similarity vector $Sim(t_1, t_2, O) = \langle s_0, \cdots, s_M \rangle$.
Output: Revised similarity vector $Sim(t_1, t_2, O)$
Algorithm: *SimilarityUpdate*
for all $k \in P_{t_1} \cup P_{t_2}$ **do**
$s_k \leftarrow |V_k^1 \cap V_k^2| / |V_k^1 \cup V_k^2|$

6 Comparison with some other approaches

In [7], Zhang and Lee develop a taxonomy integration method which uses support vectors to capture structural similarities between ontologies. Their approach is, however, different from ours in its reliance on implicit knowledge gained from particular sets of test examples.

In [2], Doan et.al. propose a multistrategy learning approach to automatically finding mappings between the query interface and the schemas of data sources. Their approach applies multiple learner modules, where each module exploits a different type of information, in the schemas of the sources or in their data. Learner modules employ nearest-neighbor classification to entity recognition and information retrieval. Their system exploits domain integrity constraints, user feedback, and nested structures in XML data.

In [6], Roddick et.al. develop a model of semantic distance in which a graph-based approach is used to quantify the distance between two data values. Their notion of semantic distance includes both a simple traversal distance and one based on weighted arcs. Transition costs, such as an additional expense of passing through a node, are also accommodated. Multiple distance measures are accomodated and relevant information is allowed to take precedence over less relevant information. An SQL based implementation is given.

In [3], Doan et.al. take a probabilistic approach to finding semantic mappings between ontologies. Their system, GLUE, employs machine learning techniques to find such mappings. Given two ontologies, for each

concept in one ontology GLUE finds the most similar concept in the other ontology. Probabilistic definitions of several different similarity measures are used. This is in contrast to most existing approaches, which deal with a single similarity measure. GLUE uses multiple learning strategies, each of which exploits a different type of information either in the data instances or in the taxonomic structure of the ontologies.

Finally, in [1], Benerecetti et.al. take on a significant challenge in attempting to incorporate semantical annotations when matching ontological schemata. They obtain some criteria for soundness and adequacy, but observe, correctly we think, that developing semantically competent autonomous agents seems to be out of reach of current methods. Their approach is in stark contrast to ours, which focuses exclusively on structural properties of ontological representation.

In [4], there is a comprehensive review of several more methods.

7 Conclusion

Just as people interpret the world in different ways, we can use ontologies to provide context filters for documents to gain different world views. Depending on the context, documents may or may not be similar. Our method can be used in a plug-in fashion, where different ontologies may be used to provide different world-views. Considering any set of documents and an ontology, the similarity relation studied in this paper induces a semantic interconnection between documents. It produces an implicit web of documents and can be utilized for finding related documents in a dynamically changing environment. A next step will be to integrate this method in efficient algorithms for information retrieval in evolving contexts.

References

1. M. Benerecetti, P. Bouquet, and S. Zanobini. Soundness of semantic methods for schema matching. In *Meaning Coordination and Negotiation Workshop (MCNW-04)*, 3th International Semantic Web Conference (ISWC-04). 2004.
2. Anhai Doan, Pedro Domingos, and Alon Halevy. Learning to match the schemas of data sources: A multistrategy approach. *Mach. Learn.*, 50(3), 2003.
3. AnHai Doan, Jayant Madhavan, Pedro Domingos, and Alon Halevy. Learning to map between ontologies on the semantic web. In *WWW '02: Proceedings of the eleventh international conference on World Wide Web*, 2002.
4. Yannis Kalfoglou and Marco Schorlemmer. Ontology mapping: the state of the art. *Knowl. Eng. Rev.*, 18(1), 2003.

5. Vladimir Oleshchuk and Asle Pedersen. Ontology based semantic similarity comparison of documents. In *Proceedings of the 14th International Workshop on Database and Expert Systems Applications*, DEXA'03. 2003.
6. John F. Roddick, Kathleen Hornsby, and Denise de Vries. A unifying semantic distance model for determining the similarity of attribute values. In *CRIPTS '16: Proceedings of the twenty-sixth Australasian computer science conference on Conference in research and practice in information technology*, 2003.
7. Dell Zhang and Wee Sun Lee. Web taxonomy integration using support vector machines. In *WWW '04: Proceedings of the 13th international conference on World Wide Web*, 2004.