Semantic Similarity Based on Compact Concept Ontology

Ce Zhang[1] Yu-Jing Wang[1] Bin Cui[1] Gao Cong[2]
[1] Key Laboratory of High Confidence Software Technologies (Peking University), Ministry of Education, CHINA & School of EECS, Peking University
{zhangce,wangyujing,bin.cui}@pku.edu.cn
[2] Microsoft Research Asia
gaocong@microsoft.com

ABSTRACT

This paper presents a new method of calculating the sematic similarity between articles based on WordNet. To further improve the performance of the proposed method, we build a new Compact Concept Ontology (CCO) from WordNet by combining the words with similar sematic meanings. The experimental results show that our approach significantly outperforms a recent proposal of computing semantic similarity, and demonstrate the superiority of the proposed CCO method.

Categories and Subject Descriptors

H.3.m [**Information Systems**]: Miscellaneous

General Terms

Algorithm, Experimentation, Performance

Keywords

Concept, Ontology, Similarity

1. INTRODUCTION

Semantic similarity plays important roles in a wide range of applications, e.g. information retrievals and clustering. Existing work [3] on semantic similarity mainly focuses on the semantic similarity between two words, while the semantic similarity between two articles has attracted much less attention. At first glance, it seems to be trivial to extend the semantic similarity between words to the semantic similarity between articles. However, we found that a straightforward extension, e.g. [1], did not work well in our project on context-based video retrieval, where the videos are typically accompanied by text descriptions such as google video. To achieve better performance, we propose a new method of computing semantic similarity between two articles based on WordNet [2], which is a concept ontology having a tree structure as shown in Figure 1(a) (See Section 2). We further improve our method by constructing a Compact Concept Ontology from WordNet by merging the nodes with similar sematic meanings (See Section 3).

2. SIMILARITY CALCULATING

A straightforward method is given in [1] to combine the semantic similarity between two words to compute the semantic similarity between two articles. Given two articles P and P', their semantic similarity is computed as follows in [1]:

$$Sim(P, P') = \frac{1}{|P|} \sum_{1 \leqslant j \leqslant |P|} \max_{1 \leqslant i \leqslant |P'|} WordSim(w_j, w'_i),$$

where $WordSim(.,.)$ is the semantic similarity between two words computed by the method [4] that is based on WordNet, w_j and w'_i are words in articles P and P', respectively.

The algorithm in [1] treats each word pair independently, and thus cannot capture the semantic of the article as a whole. For example, a word may have multiple senses and the context will help to identify the real sense in an article. Therefore, we propose a new ontology-based algorithm to calculate the similarity between articles globally. It first builds a weighted tree for each article based on WordNet, and then computes the semantic similarity between two articles using their weighted trees.

Given an article P, we build its weighted tree T_P as follows. For every word w_i in P, we find the corresponding node in WordNet for each sense of w_i to get a set $SC_i = \{C1, ..., Cr\}$ of nodes, where r is the number of senses of w_i. Each node C_j in SC_i and the corresponding path from C_j to the root of WordNet tree will be added to T_P (if the path is not in T_P); for each node from C_j to the root, we increase its weight by one. In this way, we will get the complete weighted tree T_P for P after we process all words in P. The weight tree T_P represents the semantic characteristics of the whole article P. The similarity of two articles can be calculated by comparing the similarity of their weighted trees. The similarity between two weighted trees T_1 and T_2 is defined as follows:

$$Sim(T1, T2) = -\log \sum_{node_i} \mid \frac{W_1(node_i)}{W_1(root)} - \frac{W_2(node_i)}{W_2(root)} \mid,$$

where $W_1(.)$ and $W_2(.)$ are weights for nodes in tree T_1 and T_2 respectively, and $node_i$ represents every node appearing in either $T1$ or $T2$. If a certain node only appears in one tree, its weight in another tree is set as 0.

The above similarity definition is a natural extension of the similarity in [3] that computes the similarity between two words using WordNet. Compared with the method in [1], we compute similarity as a whole. In addition, our method is more efficient. Suppose the depth of ontology tree and number of tree nodes are constant. The time complexity of the algorithm in [1] is $O(m * n)$, where n and m are the number of words in two articles; The time complexity of our algorithm is $O(m + n)$, either for calculating the similarity or building the weighted tree.

3. COMPACT CONCEPT ONTOLOGY

The performance of our algorithm is highly dependent on the underlying concept ontology from which we generate a weighted tree

WWW 2008, April 21–25, 2008, Beijing, China.
ACM 978-1-60558-085-2/08/04.

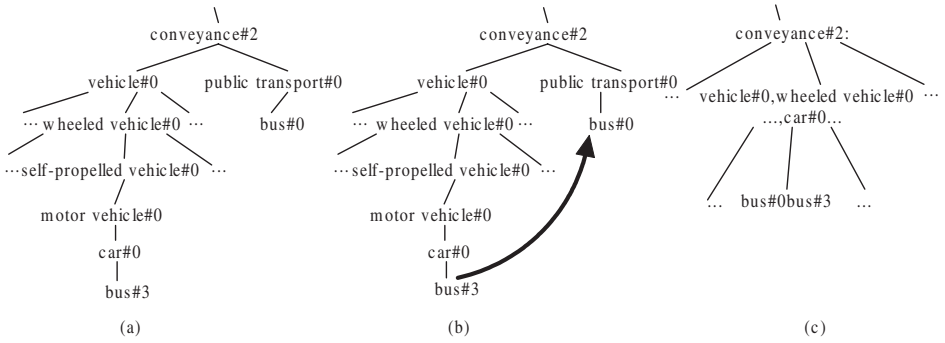

Figure 1: Structure of WordNet and Process of Union

for each article. Although the algorithm presented in Sec. 2 adopts WordNet as the concept ontology, we find that WordNet does not fit very well with our algorithm. For instance, for two different words, their corresponding nodes in the tree structure are probably not the same even if they have similar semantic meaning. Consider three words, e.g. $rabbit$, $bunny$ and bus, $rabbit$ and $bunny$ are semantically very similar with each other while bus is not relevant to them. In WordNet, the three words will be matched to three different nodes and thus the semantic relationship among them will be ignored in our algorithm. If words $rabbit$ and $bunny$ can be matched to the same node in the ontology tree, the performance of our algorithm can be improved. To cope with the closely similar words, we build a more compact concept ontology from WordNet by combining nodes that are semantically similar. The Compact Concept Ontology(CCO) has the following properties:

1. CCO inherits the hierarchical relationship of WordNet structure.
2. Different senses for the same word are combined to a single concept if they are very similar.
3. CCO highlights semantic factors and considers combination when two concepts have similar semantic meaning.
4. Compared with WordNet, it is much more compact and yields better time efficiency.

We next present the WordNet Union Algorithm to build CCO from WordNet by combining semantically similar nodes. More specifically, we find semantically similar node pairs (i, j) and then the nearest common ancestor node a in WordNet. Then, we combine all the nodes on the paths between a and i,j to form a new node k, which is child node of a, and also combine i and j to form a new node, which is the child node of node k. Figure 1(b) and (c) shows an example. In Figure 1(b), nodes bus#0 and bus#3 are semantically similar nodes and the nodes between them and their common ancestor will be combined into one node in Figure 1(c) and nodes bus#0 and bus#3 are also combined into one node.

The open problem is to find the semantically similar node pairs (i, j). They need to satisfy the following two conditions: 1) they represents two senses of the same word, and 2) they are highly semantically similar i.e. $Sim(i, j) > threshold$, where $Sim(.,.)$ is computed using the WUP method in [1]. To determine a reasonable threshold, we choose 300 words and find their synonyms, and then compute the similarity with their synonyms using WUP. The average of the similarity will be used as threshold. The generated CCO contains 2561 nodes, and each of them presents a concept which is semantically different from others. Note that, we can generate different CCOs by setting different thresholds.

4. EXPERIMENT

To evaluate the algorithm and the usefulness of CCO, we download 100 articles from Google Video and each article represents the content description of a video. For each article P_i, we randomly select 4 articles P_{i1}, P_{i2}, P_{i3}, P_{i4} from the left 99 articles. An annotator is asked to label the 4 articles as *relevant* or *non-relevant* to P_i. We then compare our method of computing semantic similarity with the method of computing semantic similarity in [1] and cosine similarity without considering similarity. For each P_i, we compute its similarity with the other 4 articles P_{i1},... P_{i4} respectively and rank them in terms of similarity. If the similarity of the top 1 is larger than a threshold, the article is *relevant* to P_i; otherwise *non-relevant*. If it is consistent with the human annotation, it is correct; otherwise, it is wrong.

Table 1: Result of Experiment

	CCO	WordNet	[1]	Cosine
Accuracy	73.6%	68.18%	49.09%	40%

The results are given in Table 1. The results for all methods are obtained through 10-fold cross validation (to determine the threshold of *relevant* and *non-relevant*). The results show that our method outperforms the method of computing semantic similarity in [1]. Moreover, the CCO performs better than WordNet in our algorithm. In addition, the use of CCO can decrease the execution time by 2 times than WordNet, as CCO contains much fewer tree nodes than WordNet and the depth of the tree is much smaller. Our algorithm is almost 28 times faster than the algorithm in [1] when using CCO as ontology.

Our future work includes evaluating on larger datasets, and applying the semantic similarity for context-based video retrieval.

5. REFERENCES

[1] M. D. Boni and S. Manandhar. Implementing clarification dialogues in open domain question answering. In *Natural Language Engineering*, 2005.
[2] C. D. Fellbaum. *WordNet: An Electronic Lexical Database*. MIT Press, 1998.
[3] D. Lin. Using syntactic dependency as local context to resolve word sense ambiguity. In *35th ACL*, 1997.
[4] Z. Wu and M. Palmer. Verb semantics and lexical selection. In *32nd ACL*, 1994.