IEICE TRANS. INF. & SYST., VOL.E100–D, NO.1 JANUARY 2017

LETTER

Efficient Algorithm for Sentence Information Content Computing in Semantic Hierarchical Network

Hao WU[†a), *Member and* **Heyan HUANG**[†b), *Nonmember*

SUMMARY We previously proposed an unsupervised model using the inclusion-exclusion principle to compute sentence information content. Though it can achieve desirable experimental results in sentence semantic similarity, the computational complexity is more than $O(2^n)$. In this paper, we propose an efficient method to calculate sentence information content, which employs the thinking of the difference set in hierarchical network. Impressively, experimental results show that the computational complexity decreases to $O(n)$. We prove the algorithm in the form of theorems. Performance analysis and experiments are also provided.

key words: *information content, sentence IC, inclusion-exclusion principle, difference set, hierarchical network*

1. Introduction

Nowadays semantic textual sentence similarity becomes a research hotspot [1], [2] in short text related area of natural language processing (NLP). From the view point of information theory, the essence of natural language is the carrier of information. The amount of information can be calculated by information content (IC) [3]. IC has been successfully applied in word similarity computation [3]–[5]. In sentence similarity computation reaserch, Wu and Huang [6] proposed a sentence IC computational model utilizes the inclusion-exclusion principle from combinatorics. To the best of our knowledge, it's the first model that can compute non-overlapping sum IC for a sentence [1], [2], [6]. It is a fully unsupervised computational model and obtains desirable experimental results on the test set. But the computational complexity is over $O(2^n)$ [6] which becomes the bottleneck for its further applications.

To address the above-mentioned efficient issue, we propose a new model to compute sentence IC which employs the thinking of the difference set and makes use of the features of hierarchical network. Actually, many combinations of nodes share the same subsumer (the node subsumes the other nodes) which respects common IC of nodes. In the inclusion-exclusion principle model, the same common IC has been continuously added and subtracted, which causes the unreasonable waste of computation, but it is difficult to decide which combinations should be abolished. In order to avoid double counting, we add the words into the information space one by one and add information gain of the newly input one each time, which is the idea of the difference set rather than the inclusion-exclusion principle. The proof and experimental results demonstrate the consistency of IC values computing between the two models. The computational complexity decreases dramatically by employing the new mothod.

The contributions of this work are summarized as follows: 1) it presents a high-efficiency computational model by exploiting the thinking of the difference set for computing sentences IC, 2) it establishes a theoretical system with lemmas and theorems for sentence IC computing, and 3) the elaborated algorithms, comparative analysis and experiments about computational complexity are given.

2. Preliminaries

Following the standard argumentation of information theory, Resnik [3] defines information content (IC):

$$IC\,(c) = -\log P(c), \tag{1}$$

where $P(c)$ refers to statistical frequency of concept c. The implementation of $P(c)$ is

$$P\,(c) = \frac{\sum_{w \in words(c)} count(w)}{N}, \tag{2}$$

where $words\,(c)$ is the set of the words contained in concept c and sub-concepts of c in the hierarchy of semantic net, N is the sum of frequencies all the words contained in semantic hierarchical net.

Let $c_1, \cdots, c_n$ be the collection of concepts, we defined the quantity of common information of n-concepts*:

$$commonIC\,(c_1,\cdots,c_n) = IC\left(\bigcap_{i=1}^{n} c_i\right) = IC\left(\bigcup_{j=1}^{m} c_j\right), \tag{4}$$

where $c_j \in subsum\,(c_1, \cdots, c_n)$, m is the total number of c_j. For physical meaning of ∩ and ∪, see Sect. 3.1 for details.

*In previous work [6], we define common IC of n-concepts is

$$commonIC\,(c_1,\cdots,c_n) = IC\left(\bigcap_{i=1}^{n} c_i\right) = \max_{c \in subsum(c_1,\cdots,c_n)} [-\log P(c)], \tag{3}$$

where, $subsum\,(c_1, \cdots, c_n)$ is the set of concepts that subsume all the concepts of $c_1, \cdots, c_n$. In consideration of accurate calculation for multiple subsumers of $c_1, \cdots, c_n$, here, we change it to Eq. (4).

Manuscript received August 29, 2016.
Manuscript publicized October 18, 2016.

[†]The authors are with Beijing Engineering Research Center of High Volume Language Information Processing and Cloud Computing Applications, School of Computer Science, Beijing Institute of Technology, Beijing, 100081, China.
a) E-mail: wuhao123@bit.edu.cn
b) E-mail: hhy63@bit.edu.cn (Corresponding author)

DOI: 10.1587/transinf.2016EDL8177

Specially, when n is 1, Eq. (4) becomes IC of one single concept: $commonIC\,(c_1) = IC\,(c_1)$.

Through the inclusion-exclusion principle [6], the quantity of total information of n-concepts is

$$\begin{aligned} &totalIC\,(c_1, \cdots, c_n) = IC\left(\bigcup_{i=1}^{n} c_i\right) \\ &= \sum_{k=1}^{n} (-1)^{k-1} \sum_{1 \le i_1 < \cdots < i_k \le n} commonIC\,(c_{i_1}, \cdots, c_{i_k}). \end{aligned} \tag{5}$$

For sentence $S = \{c_i | i = 1, 2, \ldots, n; n = |S|\}$, where c_i is the concept of the i-th concept in S, $|S|$ is concept count of S, the quantity of the information in S is

$$IC\,(S) = totalIC\,(c_1, \cdots, c_n)\,. \tag{6}$$

We can see Eqs. (4) and (5) are indirect recursion. Therefore, computational complexity of precise Eq. (6) must be higher than previous work using Eqs. (3) and (5).

For more details about Preliminaries section, see paper [3] and [6] for reference.

3. Sentence IC Computing

By employing the idea of the difference set, let $ICG(c_n)$ be IC gained by introducing concept c_n to the set of $n-1$ concepts and $intersectIC\,(n\,|n-1\,)$ be the common IC shared between concept c_n and previous $n-1$ concepts. Formally,

$$ICG(c_n) = IC(c_n) - intersectIC\,(n\,|n-1\,) \tag{7}$$

Specially, define $intersectIC\,(1\,|0\,) = 0$, and $ICG(c_1) = IC(c_1)$. Thus, sentence IC can be defined as

$$IC(S\,) = \sum_{i=1}^{n} ICG(c_i) \tag{8}$$

The following sections will show how to compute $intersectIC(i|i-1)$.

3.1 Basic Concepts and Functions

For convenience in the discussion, we name some concepts and define some functions:

1) HSN: Hierarchical semantic network is a semantic knowledge base with hierarchical structure such as WordNet [7]. In WordNet, content words are grouped into sets of cognitive synonyms (synsets), each expressing a distinct concept (a node in HSN). The most frequently encoded relation among synsets is the super-subordinate relation (is-a relation). All noun synsets ultimately go up to the root synset (the concept of *entity*).

2) SIS: Semantic information space is the space mapping of HSN through Eqs. (1) and (4). Concepts (Nodes) with the super-subordinate relation in HSN are the space with inclusion relation in SIS. The space of super concept is subsumed by that of subordinate one. SIS isn't a traditional space which uses orthogonality multidimensional to construct, while it utilizes the inclusion relationship of the information to represent.

Physical meaning of IC is the space size of concepts in SIS: the space size of concept c is $IC(c)$, the common space size of n-concepts is $commonIC\,(c_1, \cdots, c_n)$, the total space size of n-concepts is $totalIC\,(c_1, \cdots, c_n)$ and the intersection space size between concept c_{n+1} and n-concepts is $intersectIC(n+1|n)$.

3) $Root(c_i)$ indicates the set of paths, each path consists of sequence of nodes from c_i to the root in HSN. $Root(n)$ is the short form of $Root\,(c_1, \cdots, c_n)$. Formally, let $Set(p)$ be the set of nodes in path p, $Root\,(n) = \{p_k | \forall Root(c_i), p_k \in Root(c_i), p_t \in Root(c_i), Set(p_k) \not\subset Set(p_t), i = 1, 2, \ldots, n\}$. $|Root(c_i)|$ means the number of paths in $Root(c_i)$.

4) $HSN(c_i)$ expresses the set of nodes in any of path in $Root(c_i)$. $HSN(n)$ is the short form of $HSN\,(c_1, \cdots, c_n)$. Formally, $HSN\,(n) = \{c_k | \forall HSN(c_i), c_k \in HSN(c_i), i = 1, 2, \ldots, n\}$. From is-a relationship among concepts, we have

$$totalIC(HSN(c_i)) = IC(c_i); \tag{9}$$

$$totalIC(HSN(n)) = totalIC(c_1, \cdots, c_n). \tag{10}$$

5) $SIS\,(c_i)/SIS\,(n)$ denotes the space occupied by the nodes of $HSN(c_i)/HSN(n)$. $SIS\,(n)$ is also the shortened form of $SIS\,(c_1, \cdots, c_n)$. $|SIS\,(c_i)|$ and $|SIS\,(n)|$ is the size of the space $SIS\,(c_i)$ and $SIS\,(n)$ respectively. From the physical meaning of $totalIC$, we have

$$|SIS\,(c_i)| = totalIC(HSN(c_i)) > 0; \tag{11}$$

$$|SIS\,(n)| = totalIC(HSN(n)) > 0. \tag{12}$$

3.2 Method Proving

Suppose Ω is the universal set of all the nodes in HSN, define $Outer(c_i) = \{c_k | \forall c_k \in \Omega, c_i \notin HSN(c_i)\}$, $Outer(n) = \{c_i | \forall c_i \in \Omega, c_i \notin HSN(n)\}$. For $Outer(n)$, we have

Lemma 1 (Only Outer Node Expands Space)**.** *If* $n \in \mathbb{N}+$, *then* $c_{n+1} \in Outer(n) \Leftrightarrow |SIS\,(n+1)| > |SIS\,(n)|$.

Proof. From the relationship between HSN and SIS, we know each node in HSN holds a space in SIS. Equations (9) and (10) show the space owned by subordinate nodes embody the space possessed by super nodes. From the definition of $Outer(c_i)/Outer(n)$, only nodes in $Outer(c_i)/Outer(n)$ are not the super nodes of any node in $HSN(c_i)/HSN(n)$, so only $Outer(c_i)/Outer(n)$ provide additional space for $SIS\,(c_i)/SIS\,(n)$ and vice versa. According Eqs. (11) and (12) we can have Lemma 1. □

For the space of $SIS(c_i)/SIS\,(n)$ is already held by nodes of $HSN(c_i)/HSN(n)$, we can easily infer the following corollary from Lemma 1:

Corollary. *If* $n \in \mathbb{N}+$, *then* $c_{n+1} \in HSN(n) \Leftrightarrow |SIS\,(n+1)| = |SIS\,(n)|$.

Let $Deepest(ps)$ be the deepest node in HSN from path set ps. Define $Intersect(n+1|n) = \{Deepest(\{Set(p_t) \wedge$

$HSN(n)\}), p_t \in Root(c_{n+1}), t{=}1, \cdots, |Root(c_{n+1})|\}$. The node number of $Intersect(n+1|n)$ is $|Intersect(n+1|n)|$. We have

Theorem 1 (Intersected Margin Nodes)**.** *If $c_{n+1} \in Outer(n)$, let $|Root(c_{n+1})| = m$, then $|Intersect(n+1|n)| \le m$ and $intersectIC(n+1|n) = totalIC(Intersect(n+1|n))$.*

Proof. Prove $|Intersect(n+1|n)| \le m$: For $c_{n+1} \in Outer(n)$, geometrically, we know that m non-overlapping paths of c_{n+1} have m intersected points at marginal nodes of $HSN(n)$. One case that overlaps of paths which begin from c_{n+1} to root happen in $Outer(n)$, and they keep overlapping till paths reached $HSN(n)$, the number of intersected nodes must be less than m.

Prove $intersectIC(n+1|n) = totalIC(Intersect(n+1|n))$: Define $HSN(n+1|n) = HSN(c_{n+1}) \wedge HSN(n)$. Let $SIS(n+1|n)$ be the space held by nodes in $HSN(n+1|n)$, in other words, the space intersected between $SIS(c_{n+1})$ and $SIS(n)$; $|SIS(n+1|n)|$ denotes the space size of $SIS(n+1|n)$. From Eq. (12), $|SIS(n+1|n)| = totalIC(HSN(n+1|n))$. From physical meaning of $intersectIC$, $intersectIC(n+1|n) = |SIS(n+1|n)|$.

From the definition of $Intersect(n+1|n)$, each node in $HSN(n+1|n)$ either has subordinate nodes in $Intersect(n+1|n)$, or is the node in $Intersect(n+1|n)$. From Eq. (10), $totalIC(HSN(n+1|n)) = totalIC(Intersect(n+1|n))$. Thus, $intersectIC(n+1|n) = totalIC(Intersect(n+1|n))$. □

From Theorem 1 and Eq. (8), sentence IC is

$$IC(S) = \sum_{i=1}^{n} [IC(c_i) - totalIC(Intersect(i|i-1))]. \tag{13}$$

Specially, when network degenerates to tree structure, $\forall c_{n+1} \in Outer(n), |Root(c_{n+1})| \equiv 1$. Thus, $|Intersect(n+1|n)| \equiv 1$.

Let $Subordinate(c_i)$ denotes the set of subordinate nodes of c_i in HSN. Define leaf nodes of $HSN(n)$: $Leaf(n) = \{c_i | \forall c_i \in HSN(n), Subordinate(c_i) \wedge HSN(n) = \varnothing\}$:

Lemma 2 (Leaf Nodes Represent Space)**.** *If $n \in \mathbb{N}+$, then $|SIS(n)| = |SIS(Leaf(n))|$.*

Proof. From the definition of $Leaf(n)$: Any leaf concept can't be subsumed by any other concepts in $HSN(n)$, including any other leaf concepts. On the contrary, any non-leaf concept can be subsumed by at least one other concept in $HSN(n)$. In other words, only nodes in $Leaf(n)$ don't have any subordinate node in $HSN(n)$. The space of subordinate nodes subsumes the space of their super nodes. From Eqs. (10) and (12), the space size of $SIS(n)$ can be represented by all leaf nodes of $HSN(n)$. □

Let $Leaf(n+1|n)$ be leaf nodes of $HSN(n+1|n)$. Formally, $Leaf(n+1|n) = \{c_i | \forall c_i \in HSN(n+1|n), HSN(n+1|n) \wedge Subordinate(c_i) = \varnothing\}$. Then we have

Theorem 2. *If $n \in \mathbb{N}+$, then $intersectIC(n+1|n) = totalIC(Leaf(n+1|n))$.*

Proof. From Lemma 2, $|SIS(n+1|n)| = |SIS(Leaf(n+1|n))|$. According to the physical meaning of $totalIC$, $totalIC(Leaf(n+1|n)) = |SIS(Leaf(n+1|n))|$. Thus, $intersectIC(n+1|n) = totalIC(Leaf(n+1|n))$. □

Algorithm 1: $getTotalIC(S)$

Input: $S : \forall\{c_1, c_2, \cdots\}$
Output: tIC: Total IC of input S
if S *is empty* **then**
  return 0
$S = \{c_i | i = 1, 2, \ldots, n; n = |S|\} \leftarrow Leaf(S)$
Initialize: $tIC = 0$, $LeafRoot(0) \leftarrow$ empty root path set
for $i = 1; i \le n; i++$ **do**
  $Intersect(i|i-1), LeafRoot(i) \leftarrow getIntersectNode(LeafRoot(i-1), c_i)$
  $ICG = IC(c_i) - getTotalIC(Intersect(i|i-1))$
  $tIC += ICG$
return tIC

Algorithm 2: $getIntersectNode(c_i, LeafRoot(i-1))$

Input: c_i, $LeafRoot(i-1)$
Output: $Intersect(i|i-1)$, $LeafRoot(i)$
Initialize: $Intersect(i|i-1) \leftarrow$ empty concept set; $LeafRoot(i) \leftarrow LeafRoot(i-1)$; get $Root(c_i)$ from HSN
if $LeafRoot(i-1)$ *is empty* **then**
  $LeafRoot(i) \leftarrow Root(c_i)$
  return $Intersect(i|i-1)$
foreach $r_{new} \in LeafRoot(c_i)$ **do**
  $iPos \leftarrow$ Root position of r_{new}
  foreach $r_{orig} \in LeafRoot(i-1)$ **do**
    $(p, q) \leftarrow$ intersected position between (r_{new}, r_{orig})
    if $p == 0$ **then** /* $c_i \notin Outer(i-1)$ */
      $Intersect(i|i-1) \leftarrow$ add c_i
      break outer ForEach loop
    if $q == 0$ **then**
      $LeafRoot(i) \leftarrow$ remove r_{orig}
    if $p < iPos$ **then**
      $iPos = p$
  $LeafRoot(i) \leftarrow$ add r_{new}
  $Intersect(i|i-1) \leftarrow$ add the $iPos^{th}$ concept in r_{new}
return $Intersect(i|i-1), LeafRoot(i)$

From Theorem 2 and Eq. (8), sentence IC is

$$IC(S) = \sum_{i=1}^{n} [IC(c_i) - totalIC(Leaf(i|i-1))]. \tag{14}$$

3.3 Algorithms

The elaborate algorithms to compute sentence IC are showed in Algorithm 1 and 2. Algorithm 1 describes how to get sentence IC with Eq. (14), where $LeafRoot(n) = \{p_k | \forall c_i \in Leaf(n), p_k \in Root(c_i)\}$. Algorithm 2 represents the way to obtain $Intersect(i|i-1)$ from $HSN(i-1)$ for $\forall c_i \in$ HSN.

From the definitions of $Intersect(n+1|n)$ and $Leaf(n+1|n)$, $\forall c_i \in Leaf(n+1|n), c_i \in Intersect(n+1|n)$, but it's possible: $\exists c_i \in Intersect(n+1|n), Subordinate(c_i) \wedge HSN(n+1|n) \neq \varnothing$, that is, $c_i \notin Leaf(n+1|n)$. We can acquire $Leaf(i|i-1) \subseteq Intersect(i|i-1)$. Thus, Eq. (14) is the most

Table 1 The efficiency contrasted between the methods. The first two columns are the number of noun concept in a sentence pairs and the amount of this kind of sentence pairs. The 3-5 and 6-8 columns are the contrast of time consuming (unit: ms) for the two methods with designed complexity.

n	pairs	$O(n*2^n)$	In-Ex	$O(2^n)$	$O(n^2)$	Fast	$O(n)$
2	909	0.02	0.02	0.02	0.01	0.01	0.01
3	1368	0.08	0.07	0.04	0.04	0.03	0.02
4	1413	0.24	0.17	0.08	0.09	0.04	0.03
5	1486	0.64	0.40	0.16	0.16	0.06	0.04
6	1122	1.60	1.02	0.32	0.25	0.08	0.05
7	866	3.84	2.45	0.64	0.36	0.09	0.06
8	563	8.96	5.77	1.28	0.49	0.11	0.07
9	385	20.48	12.54	2.56	0.64	0.13	0.08
10	194	46.08	28.51	5.12	0.81	0.15	0.09
11	147	102.40	63.46	10.24	1.00	0.18	0.10
12	107	225.28	140.78	20.48	1.21	0.20	0.11
13	63	491.52	312.25	40.96	1.44	0.22	0.12
14	54	1064.96	689.81	81.92	1.69	0.25	0.13
15	34	2293.76	1453.12	163.84	1.96	0.26	0.14
16	29	4915.20	3128.24	327.68	2.25	0.28	0.15
17	16	10485.76	6804.69	655.36	2.56	0.30	0.16
18	9	22282.24	14685.67	1310.72	2.89	0.31	0.17

efficient form of Eq. (13). However, when concepts have specific features and cannot be removed at will for the further modification of algorithms, we should use full set of $Intersect(n+1|n)$ instead of $Leaf(n+1|n)$. In this case, Step 3 in Algorithm 1 should be deleted and the algorithm becomes the computation of sentence IC using Eq. (13).

3.4 Complextiy Analysis and Experiments

Searching subsumers between concepts, which consists of deepest intersected nodes between paths of two nodes in HSN, is the most time-consuming computing. Let one time of comparing between two nodes be the minimum computational unit ($O(1)$).

The previous method uses Eqs. (3) and (5). According to the Binomial Theorem, the amount of combinations among concepts to find subsumers of them can be deduced [6]:

$$C(n,1) + C(n,2) + \cdots + C(n,n) = 2^n - 1. \tag{15}$$

where n is the amount of the concepts in the sentence pair, $C(n,1)$ is the number of 1-combinations from n-concepts. Actually, this method is approximate and the real computational times of precise method are more than $[0 * C(n,1) + 1 * C(n,2) + \cdots + (n-1) * C(n,n)]$. Therefore, the computational complexity of previous method is between $O(2^n)$ and $O(n * 2^n)$.

Efficient method employs Eq. (13) or (14). There are two layers of loops to find intersected nodes between nodes from Algorithm 1 and 2. The function $Leaf(S)$ at Step 3 in Algorithm 1 can be realized by no more than two layers of loops based on its definition. Generally, $|Intersect(n+1|n)| \leq 2$ and Step 2 in Algorithm 1 could be deemed as an ordinary step rather than recursive statement. Hence, the computational complexity is about $O(n^2)$.

We use all dataset of English from [2] to setup our experiments. Because total IC of two sentences are required in sentence similarity computing, we use each joint of two sentences as one computing unit which has the max computational complexity. For convenience in IC computing using WordNet, only real nouns are employed. The experimental results show the consistency of IC values from two models.

Table 1 show the efficiency contrasted between the models. To our surprise, the computational complexity of efficient algorithm is only $O(n)$ according to the polynomial index from curve fitting of experimental results utilizing Matlab toolkit. This complexity decrease may be caused by employing $LeafRoot(n)$ to efficiently represent $HSN(n)$.

4. Conclusion

This work proposes an efficient model to compute sentence IC by utilizing the thinking of the difference set in hierarchical network. It solves the waste of computation by employing the inclusion-exclusion principle. Theoretical system with lemmas and theorems has been established for supporting the correctness of sentence IC computing. Algorithms based on the theorems are elaborated. The computational complexity decreases to $O(n)$ from more than $O(2^n)$. Efficiency improvement indicates that sentence IC model could be applied to long texts such as paragraphs or even documents.

Acknowledgments

The work described in this paper was mainly supported by State Key Program of National Natural Science Foundation of China under Grant 61132009, National Programs for Fundamental Research and Development of China (973 Program) under Grant 2013CB329303 and National Natural Science Foundation of China under Grant 61502259.

References

[1] E. Agirre, C. Banea, C. Cardie, D. Cer, M. Diab, A. Gonzalez-Agirre, W. Guo, I. Lopez-Gazpio, M. Maritxalar, R. Mihalcea, G. Rigau, L. Uria, and J. Wiebe, "SemEval-2015 task 2: Semantic textual similarity, English, Spanish and pilot on interpretability," Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), June, 2015.

[2] E. Agirre, C. Banea, D. Cer, M. Diab, A. Gonzalez-Agirre, R. Mihalcea, G. Rigau, and J. Wiebe, "Semeval-2016 task 1: Semantic textual similarity, monolingual and cross-lingual evaluation," Proceedings of SemEval, pp.497–511, 2016.

[3] P. Resnik, "Using information content to evaluate semantic similarity in a taxonomy," International Joint Conference on Artificial Intelligence (IJCAI), 1995.

[4] J.J. Jiang and D.W. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," Proceedings of International Conference Research on Computational Linguistics (ROCLING X), 1997.

[5] D. Lin, "Using syntactic dependency as local context to resolve word sense ambiguity," Proceedings of the 35th Annual Meeting of the Association for Computational Linguistics and Eighth Conference of the European Chapter of the Association for Computational Linguistics, pp.64–71, ACL, 1997.

[6] H. Wu and H. Huang, "Sentence similarity computational model based on information content," IEICE Trans. Inf. & Syst., vol.E99-D, no.6, pp.1645–1652, June 2016.

[7] G.A. Miller, "WordNet: a lexical database for English," Communications of the ACM, vol.38, no.11, pp.39–41, 1995.