Conceptual aggregation graphs CAG for natural language representation and reasoning*

Luo MinCong
China Institute of Atomic Energy
Peking
luomincong@foxmail.com

ABSTRACT

In this paper we propose a method that can be used for machine understanding of natural language logic by building conceptual aggregation graph model called **CAG**, where each node represents a specific object, action, or concept, and each edge represents specific relationship between specific nodes, so the logical connotation of the text can be represented and deduced on this basis. First of all, we represent the concepts and logic of the text as the conceptual aggregation graph model CAG. In order to measure the logical relevance between sentences, we propose a multi-scale logical relevance measurement algorithm based on the CAG. New meaningful relationships between concepts and patterns can be discovered and inferred based on the CAG. We use experiment to show that such concepts representation and inference method are feasible.

KEYWORDS

conceptual aggregation graph ,logic representation ,logical relevance measurement

ACM Reference Format:
Luo MinCong. 2018. Conceptual aggregation graphs CAG for natural language representation and reasoning. In *Proceedings of ACM ICCES (Conference ICCES).* ACM, Kuala Lumpur,Malaysia, Article 4, 4 pages. http://www.ijist.net/ICCES2018/10.475/123_4

1 INTRODUCTION

In this paper we propose the conceptual aggregation graph model CAG that focuses on excavating the logic correlation between sentences. After converting a large number of natural language statements into CAG, the machine can reason based on existing knowledge. In the early research of AI, the first-order predicate logic [3] and the language comprehender network [8] was used for the representation and reasoing of knowledge, the defect was a lot of artificial facts must be constructed, and the first-order logic also has limitations for representation of complex facts . The conceptual dependency [8] [9] is more expressive than first-order logic and is closer to the expression of natural language, but it's still difficult to convert natural language into concept dependency graph using algorithms, and rule-based reasoning can only dealing with "certain reasoning" and is difficult to deal with "uncertain reasoning". Markov logic network [4] softens the hard constraints in the first-order predicate logic rules and can cope with the difficulties of "uncertain reasoning". We derive inspiration from this combination of numerical reasoning and symbolic reasoning [6] [10] [2] [1], and propose a CAG model makes natural language statements can be expressed both symbolically and computationally.

First, we propose an algorithm for converting natural language statements into CAG and give an example. Based on CAG, we give the algorithm $L_s(G_b|G_a)$ to measure the logical association between the sentence b and the sentence a. Then how to use the existing knowledge for logical reasoning based on the above algorithms is showed.

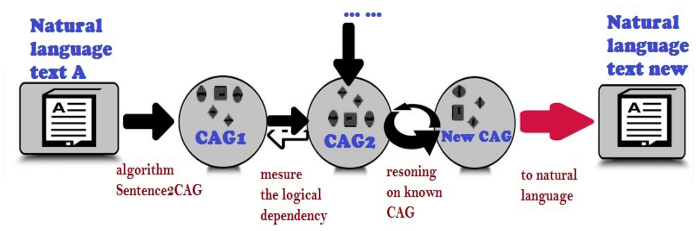

Figure 1: Steps of CAG for natural language representation and reasoning.

2 CONVERT A SENTENCE INTO CAG

Conceptual aggregation refers to the aggregation of words $word \in C_{tag}$ in the sentence as a special node "concept node" in the CAG by defining an interval vocabulary set C_{tag}. For example, for the statement "we will implement deep learning algorithms with very simple codes", if we define the interval vocabulary set $C_{tag} = \{'MD', 'VB', 'IN'\}$, then the extracted concept set will be
$C_{concept} = \{'we', 'deeplearningalgorithms', 'verysimplecodes'\}$.
The advantage of conceptual aggregation is complete phrases of the concepts extracted by no longer segmenting all the vocabularies, which helps logical reasoning.

To translate a natural language sentence into CAG, we can construct the algorithm $G = f(S)$ in sentence units

*The Source Code:https://github.com/Luomin1993/SciBot to obtain the CAG G. CAG is different from traditional conceptual dependency graphs [4]. Except for the special node of "concept", verbs, and adverbs are also nodes in the CAG. For example, "Active people often send the friend-circle photos because of small things." and "Lv is an active man in the friend-circle." The specific conversion algorithm Sentence2Graph is showed in algorithm.1.

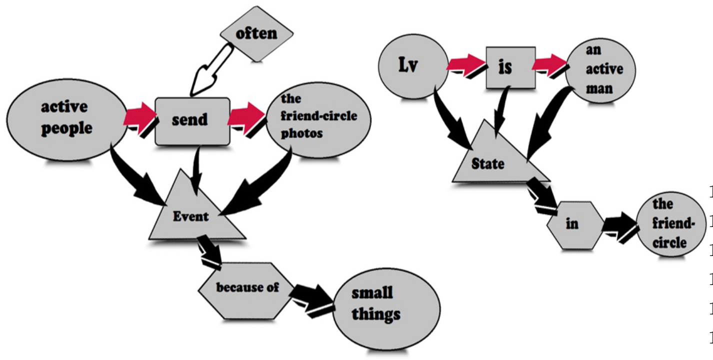

Figure 2: The CAG of "Active people often send the friend-circle photos because of small things." and "Lv is an active man in the friend-circle."

3 LOGICAL ASSOCIATION BETWEEN THE SENTENCES

Now we need an algorithm to measure the logical association and dependency between different sentences, which can be used to measure the logical difference between different statements. Human can easily judge "He will send friend-circle photos on marrige of SunYiKai." is reasonable based on the sentences "Active people often send the friend-circle photos because of small things." and "Lv is an active man in the friend-circle.". For computers, the links between the statements of this logical inference process need to be calculated with logical dependencies, which can be measured by Algorithm.2.

The algorithm of logical dependency is multi-scale:

- **Similarity of the nodes:** first we calculate the metric from the similarity of the nodes: $l(b_i, a_i) = ||V_{b_i} - V_{a_i}||_2$.
- **Structural similarity:** then calculate from the structural similarity between the nodes: $d(b_i, b_j, a_i, a_j) = (l(b_i, a_i)+l(b_j, a_j)) \sum_{b_B \in B_b, a_B \in B_a} max \quad l(b_B, a_B)$ (where B_a are the set of the nodes between a_i and a_j in G_a).
- **Similarity between groups:** finally measure from the structure of similar groups: $d(N_i, N_j) = \sum_{b_B \in B_b, a_B \in B_a} max \quad l(b_B, a_B)$, such small to large metric calculation helps to calculate the dependency from the details and macroscopically.

Algorithm 1 Sentence2Graph

```
Input: sentence $Sen$;  Output: graph $G$;
Init:$G \leftarrow \{\}$;concept set $C_c$;VBZ set $C_b$;VB set $C_v$;temporary concept $c$;temporary event node $E$;temporary state node $S$;
1: $map \leftarrow givetag(Sen)$;
2: for word $w$ in $Sen$ do
3:     if $map[w] \in C_c$ then $c.push_back(w)$;
4:     end if
5:     if $map[w] \notin C_c$ and $c \neq \varnothing$ then $G.append(c)$; clear c: $c.clear()$;
6:     end if$G.append(node.new(w, map[w]))$;
7: end for
8: for node $n$ in $G$ do
9:     if $n.class \in C_v$ then $E.link(n)$;
10:    end if
11:    if $n.class \in C_b$ then $S.link(n)$;
12:    end if
13:    if $n.class ==$ "$concept$" then
14:        if $S \neq \varnothing$ then
15:            $n_first = n$;
16:            while $n_f.class \neq$ "$concept$" do
17:                $n_f \leftarrow n_f.prev$
18:            end while$S.link(n_f)$; $S.link(n)$;
19:        end if
20:        if $E \neq \varnothing$ then
21:            $n_f = n$;
22:            while $n_f.class \neq$ "$concept$" do
23:                $n_f \leftarrow n_f.prev$
24:            end while$E.link(n_f)$; $E.link(n)$;
25:        end if
26:    end if
27: end for
28: $G.linkall()$;
29: return $G$;
```

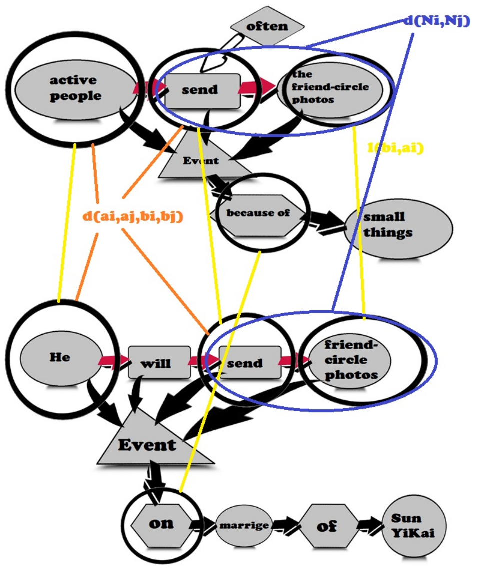

Figure 3: multi-scale measure of logical association and dependency between 2 sentences.

Algorithm 2 Logical dependency degree $L_s(G_b|G_a)$

```
Input: graph of sentence A: $G_a$; graph of sentence B: $G_b$;threshold value t; Output:dependency degree $L_s \in [0,1]$ Init:match nodes pair set $C_p \leftarrow \{\}$;max neighbor nodes set $C_n \leftarrow \{\}$;max neighbor nodes $N \leftarrow \{\}$
1: $\hat{L_s}(G_a|G_b) = 0$
2: for node $b_i$ in $G_b$ do
3:     $a_i = \arg\max_{a_i} l(a_i, b_i)$ and $l(a_i, b_i) > t$;
4:     $C_p.append((b_i, a_i))$
5: end for
6: //calculate the metric from the similarity of the nodes
7: for pair $(b_i, a_i)$ in $C_p$ do
8:     $\hat{L_s}(G_a|G_b) = \hat{L_s}(G_a|G_b) + l(b_i, a_i)$;
9: end for
10: //calculate from the structural similarity between the nodes
11: for $(b_i, a_i), (b_j, a_j)$ in $C_p$ and $i \neq j$ do
12:     $\hat{L_s}(G_a|G_b) = \hat{L_s}(G_a|G_b) + d(b_i, b_j, a_i, a_j)$;
13: end for
14: for pair $(b_i, a_i)$ in $C_p$ do
15:     if $N.empty()$ then $N.append((b_i, a_i))$;
16:     end if
17:     if $linked((b_i, a_i), (b_{i+1}, a_{i+1}))$ then $N.append((b_{i+1}, a_{i+1}))$;
18:     end if
19:     if Else then $C_n.append(N)$; clear N:$N \leftarrow \{\}$;
20:     end if
21: end for
22: //measure from the structure of similar groups
23: for $N_i, N_j$ in $C_n$ and $i \neq j$ do
24:     $\hat{L_s}(G_a|G_b) = \hat{L_s}(G_a|G_b) + d(N_i, N_j)$;
25: end for
26: return $L_s(G_a|G_b) = \hat{L_s}(G_a|G_b)/\hat{L_s}(G_a|G_a)$
```

4 REASONING BASED ON KNOWN KNOWLEDGE

Assume that machine currently knows some statements, as known knowledge, based on which machine needs to infer something. By abstracting it to the question of deriving the CAG G_b with a blank-information node from the known CAG set $G_k = \{G_j\}$, which is equivalent to a fill-in problem. We define the n-order logical dependency:

Definition 4.1. n-order: n-order logical dependency refers to the logical dependence of one CAG on n CAGs.

which can be expressed as a formula:

$$\prod_{i=1}^{n-1} L_s(G_i|G_{i+1}) = L_s(G_1|G_2)L_s(G_2|G_3)...L_s(G_{n-1}|G_n)$$

Algorithm.3 (in which 2-order logical dependency is used) is proposed to solve the fill-in problem.

Algorithm 3 Blank-fill algorithm of the graph

```
Input: set of graphs known: $G_k = \{G_j\}$; graph $G_b$ with a blank node $N_b$ to fill; Output: graph $G_r$ filled; Init:similar nodes space $N_s \leftarrow \{\}$ ;
1: for node $N'_j$ in $G_k$ do
2:     if $N'_j.class = N_b.class$ then $N_s.append(N'_j)$;
3:     end if
4: end for
5: make the replaced graph optional space: $G_o = \{G_i\} = replace(G_b, N_s)$
6: $G_r, G_m, G_{m'} =$
7: $\max_{G_{m'} \in G_k/G_m} \max_{G_r \in G_o} \max_{G_m \in G_k} L_s(G_r|G_m)L_s(G_m|G_{m'})$
8: return $G_r$
```

5 USING CAG:TO ESTABLISH THE CONNECTION BETWEEN SYMBOLS AND REAL-WORLD ENTITIES

CAG can be used to map symbols(that is, natural language) as real-world entities. A typical application scenario is the language interaction function of map applications: First, the user's speech requirements can be converted into natural language texts, and then these texts are is converted into a CAG set by algorithm **Sentence2Graph**, and the conceptual nodes and other nodes in the CAG can be mapped as map information like geographic coordinates and path schemes, etc. The map application can output this information to the users. The CAG in this application scenario acts as a middle tier.

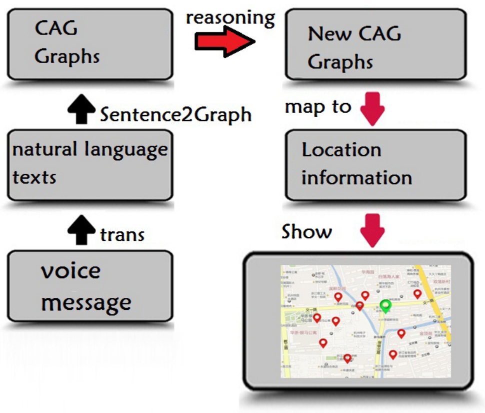

Figure 4: Using CAG for language interaction in the map application, CAG can establish the connection between language and real-world entities.

Using CAG to associate symbols with real-world entities can also be used in many other scenarios, such as natural language interaction scenarios for weather applications, food applications.

6 EXPERIMENT

We trained a model on the natural language text from the book **Capital Theory of Karl Marx**, includes 61398 lines and 5640044 characters. The text data converted to txt format is used as context prediction experiment data. As a comparison of this method, we trained the Word2Vec [5] network as a comparison.

Because of the differences between the CAG method in this paper and the statistical natural language processing method like Word2Vec in this paper, we cannot use the same method to train the model, but we split the text data into multiple unit sentence sample datasets with size of δm to reduce this difference.

The word prediction method of Word2Vec can be expressed as:

$$P(w_i|w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2})$$

The word prediction method of CAG in this paper can be expressed as:

$$P(w_i|S_{before}) = P(w_i|G_{before}) = P(w_i \in G_i|G_{before})$$

Table 1: The predicton accuracy of the models.

model	prediction method	Acc	average $Acc_{increment}$
Word2Vec	$P(w_i\|w_{i-2}, w_{i-1}, w_{i+1}, w_{i+2})$	0.24	0.974
CAG	$P(w_i\|S_{before})$	0.29	1.322

Incremental accuracy $Acc_{increment}$ is a measure of how well the model predicts word accuracy as the predictive text increases.The $Acc_{increment}$ is expressed as:

$$Acc_{increment} = Acc_{m+\delta m}/Accm$$

The average incremental accuracy is the measure of the average change in word prediction accuracy over the entire prediction experiment as the predictive text increases.

Results are reported in Table 1.The experimental results show that CAG method has no advantage compared with word2vec in the text prediction of unit interval, but with the accumulation of text, the change of word2vec model is not sensitive, but the prediction of CAG is obviously better. Therefore, compared with statistical methods, the CAG method can use existing accumulated text data to make logical reasoning.

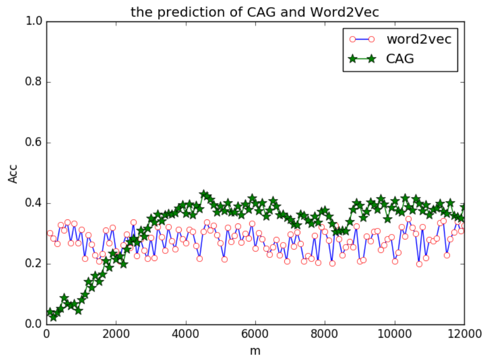

Figure 5: With the accumulation of text data, the change of word2vec model is not sensitive, but the prediction of CAG is obviously better.Compared with statistical methods, the CAG method can use existing accumulated text data to make logical reasoning.

7 CONCLUSION AND FUTURE WORK

The pure statistical machine learning method is difficult to solve the problem of structural reasoning [7], by presenting the conceptual aggregation graph model CAG, algorithms for natural language representation and reasoning are proposed, and the concrete cases are given and the feasibility is proved. At the bottom of the algorithm, for example, the semantic similarity comparison method of nodes still needs to be improved, and future work is based on the use of larger data sets for more complex reasoning (such as the extension of a single empty node to multi-nodes inferences).

REFERENCES

[1] Samuel Boutin. 1997. Using reflection to build efficient and certified decision procedures. In *International Symposium on Theoretical Aspects of Computer Software*. 515–529.
[2] Tuyen N. Huynh and Raymond J. Mooney. 2011. *Online Structure Learning for Markov Logic Networks*. Springer Berlin Heidelberg. 81–96 pages.
[3] Saul A Kripke. 1963. Semantical considerations on modal logic. *Acta Philosophica Fennica* (1963), 83–94.
[4] Lilyana Mihalkova and Raymond J. Mooney. 2007. Bottom-up learning of Markov logic network structure. (2007), 625–632.
[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Computer Science* (2013).
[6] G. Steven Olley and Ariel Pakes. 1992. The Dynamics of Productivity in the Telecommunications Equipment Industry. *Nber Working Papers* 64, 6 (1992), 1263–1297.
[7] Judea Pearl. 2018. Theoretical Impediments to Machine Learning With Seven Sparks from the Causal Revolution. (2018).
[8] M. Ross Quillian. 1969. The teachable language comprehender: a simulation program and theory of language. *Communications of the Acm* 12, 8 (1969), 459–476.
[9] Roger C. Schank. 1974. Inference and the computer understanding of natural language. *Artificial Intelligence* 5, 4 (1974), 373–412.
[10] John F. Sowa. 1976. Conceptual Graphs for a Data Base Interface. *Ibm Journal of Research and Development* 20, 4 (1976), 336–357.