A Context Sensitive Experience Feeder for Computer Aided Engineering

Bo Song and Zuhua Jiang
Department of Industrial Engineering and Logistics Management, Shanghai Jiao Tong University, Shanghai, P.R. China

Keywords: Context, Experience, Computer Aided Engineering.

Abstract: Computer aided engineering (CAE) tries to map properties and interactions of real world entities with symbols and values readable to the machine. Modern CAE software packages are powerful in function, but users usually need a lot of knowledge and experience to manipulate them. As a kind of tacit knowledge, experiences require gauged context in order to be fully understood and applied. To better exploit the freely written, hard-to-encode experiences on the web, we propose in this paper a context sensitive experience feeding mechanism which is able to recommend experiences matching the context of a given CAE task. Our method makes use of information extraction and natural language processing techniques to find experience valuable to engineer's trouble shooting. Empirical evaluation of a prototypical feeder suggests that our method is effective.

1 INTRODUCTION

Knowledge employed in so called knowledge-intensive tasks incorporates two aspects: the explicit aspect and the tacit aspect. While explicit knowledge is relatively easy to acquire, codify and reuse, such operations for tacit knowledge are much harder for that tacit knowledge is rooted in an individual's experience and values and is difficult to reduce to formal representation (Nonaka and Konno, 1998; Chen, 2010). Computer aided engineering is a typical knowledge intensive practice which has aroused continuous interests among researchers of knowledge engineering (Colombo, Mosca and Sartori, 2007). Modern CAE software packages are powerful in function, but users have to be equipped with a lot of knowledge and experience to manipulate them. One proven is that there exist tens of thousands of questions asking for guidance or experiences in some online CAE forums such as XANSYS and iMechanica. In our work, we try to channel existing experiences to engineers who might need them in their current working context, hope doing so could save time that otherwise would be spent on query formulating, searching, and waiting for response.

2 CONTEXT SENSITIVE EXPERIENCE FEED

Knowledge management efforts, to be successful, need to be sensitive to features of the context of generation, location, and application of knowledge (Nidumolu, Subramani and Aldrich, 2001). To make recommended experience more useful to an engineer, we should understand what he is doing and what problem he will face. The information collected from an ongoing CAE task facilitating such understanding is called context.

2.1 CAE Task Context

Computer aided engineering maps the property and interaction of real world entities with symbols and values readable to machine. To use CAE software packages to solve engineering problems, people have to know the terms and concepts denoting such abstract entities. This makes the name of these entities a good indicator of what a CAE task is about. As CAE operations are composed of actions taken by people on target entity, the verb-object structure can assume an informative role in describing a CAE task. Based on these observations, we propose our context model as following.

Song B. and Jiang Z..
A Context Sensitive Experience Feeder for Computer Aided Engineering.
DOI: 10.5220/0004133103430346
In *Proceedings of the International Conference on Knowledge Engineering and Ontology Development* (KEOD-2012), pages 343-346
ISBN: 978-989-8565-30-3

The *concept list*:

$$\mathbf{C} = \{c_i \mid i=1, 2, \dots, m\}$$

where c_i denotes a concept (noun phrase) captured from a CAE task, i=1, 2, … , m defines the sequence of concepts

The *potential trouble pool*:

$$\mathbf{P} = \{(v, c) \mid c \in \mathbf{C}, v \text{ is a verb}\}$$

The number of concepts in **C** is confined to m — when a new concept is added, the oldest concept in **C** is deleted and all subscripts are adjusted. For the construction of **P**, other than manually building a static lexicon, we choose to extract verb-object pairs from the ever-evolving Internet corpus. To do this, for noun phrase c_i in **C**, we construct four quoted queries:

*"how to * <c_i's singular form> "*
*"how to * <c_i's plural form> "*
*"cannot * <c_i's singular form> "*
*"cannot * <c_i's plural form> "*

Every query is to be submitted to a search engine that supports wildcard and exact string match. After the search engine has returned results for the constructed queries, we use a part-of-speech tagger to assign POS tags to matched texts, and any contained verb-object structures using c_i as argument are extracted according to the following rule (in BNF):

("how to"|"cannot") [adverb]<verb>[pronoun] [adjective]<noun phrase c_i>

We adopt above query patterns for three reasons: first, we are mainly concerned with know-how experiences; second, though there exists other syntaxes such as "how can/do I/you do a thing" for people to query know-how knowledge, a few search trials can tell that with the same semantic meaning, petitions beginning with "how to" and "I cannot" surpass others in quantity; third, under the redundancy surmise of Internet content, a single syntax should have questioned most aspects of a concept.

2.2 Trouble Detection

As a CAE task proceeds, the concept list **C** and potential trouble pool **P** are dynamically changing. At any time the task owner encounters difficulty and stops to check the experience feeder, we must assess the task context and come up with remedies for the trouble that the task owner is most probably facing. This is done by approximating the probability $P(t|\mathbf{C})$ for every potential trouble t in **P**:

$$\begin{aligned} P(t_{ij} \mid \mathbf{C}) &= \frac{P(t_{ij}, \mathbf{C})}{P(\mathbf{C})} \\ &\approx \frac{1}{\mathrm{p}} P(t_{ij}, \mathbf{C}) \\ &\approx \mathrm{q}\max\{w(x) P(t_{ij}, c_x)\} \\ &\approx \mathrm{z}\max\{w(x) \left| \{e(v_{ij}, c_i, c_x)\} \right|\} \end{aligned} \quad (1)$$

where

subscript i ranges over the m concepts in **C**, and subscript ij denotes the jth verb that has c_i as its direct object ;
$x \neq i$;
$w(x)$ is weight function ;
$e(v_{ij}, c_i, c_x)$ denotes any experience piece that contains the three keywords: v_{ij}, c_i and c_x ;
z is normalizer.

The three approximate equalities each have its meaning:

- The first approximate equality means the chance of any specific concept series is treated as equal.
- The second approximate equality significantly reduces the scale of joint distribution of concepts out of computational complexity concern. Besides, we add a weight function here to gain some control over the choice of concepts. An instinctive idea is to use more recent concepts to infer possible troubles.
- The third approximate equality relaxes the verb-object constraint between v_{ij} and c_i when using them to retrieve evidence. This is because it is impractical to parse every sentence while searching a gigantic corpus.

Whenever a new concept is captured and changes **C**, we recalculate $P(t|\mathbf{C})$ for every potential problem in **P**. For some most probable troubles, we would search for relevant experiences and recommend them to the task owner.

2.3 Experience Retrieval

To retrieve a piece of experience as candidate remedy for sensed trouble (suppose the most troublesome is t_{ij}), we use three features of textual experience for match making: trouble mentioning, context overlap, and procedural marks.

Trouble mentioning: an empirical remedy should explicitly mention the target trouble, t_{ij}, which is described by a verb-object pair. To do this, a natural language parser is needed.

Context overlap: for a piece of textual experience,

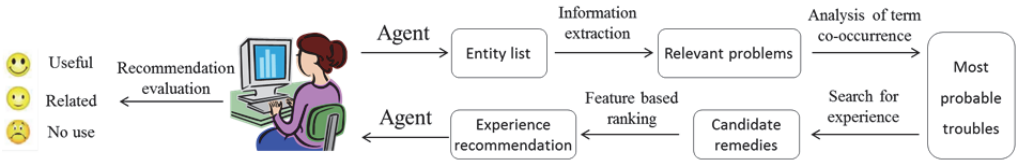

Figure 1: Context sensitive experience feeding.

e, we count the times that each concept in **C** appears in e and denote it with f_{ei} , i=1, 2, … , m. Then context overlap index for e can be calculated by:

$$O_e = \Sigma_i (f_{ei} / L_e) \quad (2)$$

where L_e denotes the words count of e

Procedural marks: since useful experience usually takes the form of procedural guidance, we studied literature investigating the characteristics of procedural text and use the linguistic marks proposed in (Aouladomar, 2005) and (Fontan and Saint-Dizier, 2008) to assess the quality of a candidate experience.

Retrieved experience pieces are firstly grouped based on what trouble they address. These groups are then ranked in descending order according to the product of trouble probability and averaged context overlap. Experience-carrying texts within each group are sorted descendently according to the number of procedural marks they contain. The information process flow for experience feeding is shown in Figure 1.

3 METHOD EVALUATION

3.1 Method Implementation

In our research, we choose ANSYS based finite element analysis as an exemplary CAE task. ANSYS is a wildly used computer program for doing finite element analysis, and it is operated in such way that a series of text-rich dialog windows are gone through one by one as the task stage advances. Thus an agent able to record texts on ANSYS dialog window can be used for context collecting. The whole procedure of experience feeding is shown in Figure 1.

Aside from the description presented in section 2, we use following concrete configurations when realizing an ANSYS experience feeder:

- m, the number of memorized concepts, equals 12 ;
- recorded window titles are assigned POS tags for noun phrase extraction;
- for each noun serving as context, we retain the 4 most appeared verb (counted from each query's first Google 100 results);
- when using formula (1) to find the most probable potential problem, $e(v_{ij},c_i,c_x)$ is any Google result in which the three keywords co-occur, and the weight function is set as below:

$$w(x) = \begin{cases} 1 & if\ x \geq m/2 \\ 0 & else \end{cases} \quad (3)$$

- for a perceived trouble, t_{ij}, relevant experience is picked from the Google results used to assess the probability $P(t_{ij}|\mathbf{C})$ — if parsed $e(v_{ij},c_i,c_x)$ gets the *trouble mentioning* feature, then the paragraph it resides in is treated as one relevant experience ;
- Stanford Parser (Klein and Manning, 2003) is used to parse Google results and assign POS tags.

3.2 Empirical Evaluation

Totally 24 college students in their junior year taking a finite element analysis course were materially rewarded to assess the effectiveness of experience feeder. We chose a static contact problem which requires about 20 steps to complete as background task, and recorded the ANSYS window titles each student had gone through when solving the task. Each time an ANSYS window is activated, the recommendation list is refreshed and the 10 most highly ranked experiences are shown. Table 1 is a snapshot of experiences recommended when the 7th window is activated.

Each student is asked to read through the experience lists and rate each experience with three grades: 1) useful to current problem solving; 2) good to know but not directly useful to current task; 3) totally irrelevant. The evaluation result is shown below (proportion=0.2 means).

Number of instantly useful ratings among 24 participants							
rating count	0	1	2	3	4	5	>5
person count	1	5	6	6	4	1	1
Proportion of relevant ratings among 24 participants							
proportion	0.1	0.2	0.3	0.4	0.5	0.6	0.7
person count	1	3	6	6	4	2	2

Table 1: Recommended experiences.

Window title history	Relevant experiences
Global Element Sizes Meshing Attributes Volume Sweepings Mesh Volumes Contact Manager Contact Wizard	For trouble: *generate mesh* concerning: *area* Typically, you will generate a mesh for the source area yourself, before you sweep the volume. For trouble: *generate element* concerning: *contact* 1. Sets of nodes that are likely to come into contact must be defined and used to generate the necessary elements. 2. Generate contact elements whose contact node is within a radius of RADC measured from the centroid of each target element face. For trouble: *find area* concerning: *mesh* While meshing the model in ANSYS we are facing difficulty in finding missing common areas.
Select Areas for Target	…

4 CONCLUSIONS

Experiences are not formal, validated knowledge, but they are highly thought-provoking and worth sharing. In this paper we have proposed an experience feeder for engineers who process finite element analysis, a typical CAE task. Experiment shows that by our method engineers can gain knowledge about what problems they may encounter at different task stage and what the possible solutions are. Other study has achieved higher performance in knowledge recommending (Shen, Geyer, Muller, Dugan, Brownholtz and Millen, 2008), but they require training on manually annotated activities and resources, which we do not. In future work, to enhance our method, geometry feature recognition can be used to capture more informative concepts form CAE tasks, and keyword extraction technique can be used for extracting concepts from task defining documents.

ACKNOWLEDGEMENTS

The author is most grateful to National Nature Science Foundation of China (No. 70971085) and the Research Fund for the Doctoral Program of Higher Education of China (No. 20100073110035), for financial support that made this research possible.

REFERENCES

Aouladomar, F. (2005). Towards answering procedural questions. In *Knowledge and Reasoning for Answering Questions*, *Workshop associated with IJCAI05*, 21-31.

Chen, Y. (2010). Development of a method for ontology-based empirical knowledge representation and reasoning. *Decision Support Systems*,50, 1–20.

Colombo, G., Mosca, A. and Sartori, F. (2007). Towards the design of intelligent CAD systems: an ontological approach. *Advanced Engineering Informatics*, 21, 153–168.

Fontan, L. and Saint-Dizier, P. (2008). Constructing a know-how repository of advices and warnings from procedural texts. In *Doceng'08: Proceedings of the Eighth Acm Symposium on Document Engineering*, 249-252.

Klein, D. and Manning, C. D. (2003). Accurate unlexicalized parsing. In *Proceedings of the 41st Meetingof the Association for Computational Linguistics*, 423-430.

Nidumolu, S. R., Subramani, M. and Aldrich, A. (2001). Situated learning and the situated knowledge web: exploring the ground beneath knowledge management. *Journal of Management Information Systems*, 18(1), 115-151.

Nonaka, I. and Konno, N. (1998). The concept of 'ba': building a foundation for knowledge creation. *California Management Review*, 40(3), 40-54.

Shen, J., Geyer, W., Muller, M., Dugan, C., Brownholtz, B. and Millen, D. (2008) Automatically finding and recommending resources to support knowledge workers' activities, In *ACM 2008*, 207-216.