# Can Web-Based Recommendation Systems Afford Deep Models: a context-based approach for efficient model-based reasoning

Leiguang Gong
T. J. Watson Research Center, IBM, Hawthorne, NY 10532, USA
leiguang@us.ibm.com

## ABSTRACT
Web-based product and service recommendation systems have become ever popular on-line business practice with increasing emphasis on modeling customer needs and providing them with targeted or personalized service solutions in real-time interaction. Almost all the commercial web service systems adopt some kind of simple customer segmentation models and shallow pattern matching or rule-based techniques for high performance. The models built based on these techniques though very efficient have a fundamental limitation in their ability to capture and explain the reasoning in the process of determining and selecting appropriate services or products. However, using deep models (e.g. semantic networks), though desirable for their expressive power, may require significantly more computational resources (e.g. time) for reasoning. This can compromise the system performance. This paper reports on a new approach that represents and uses contextual information in semantic net-based models to constrain and prune potentially very large search space, which results in more efficient reasoning and much improved performance in terms of speed and selectivity as evidenced by the evaluation results.

## Categories and Subject Descriptors
I.6.5 [Simulation and Modeling]: **Model Development** - modeling methodologies. I.2.1 [Artificial Intelligence]: applications and expert systems – industrial automation

## General Term
Algorithm, experimentation, performance.

## Keywords
Model, context, semantic network, reasoning, recommendation systems.

## 1. INTRODUCTION
The objective of all web-based recommendation systems is to provide online customers with services, products, and information targeted or customized for their personal need(s), and does it in real-time interaction. To achieve high degree of selectivity or specificity of recommended services and products and speed (of responsiveness) is of paramount importance for attracting new customers and for attaining existing customers. Almost all the commercial web service systems adopt some kind of simple customer segmentation models, and shallow pattern matching or rule-based techniques for high performance [2, 9]. The models built based on these techniques though very efficient have a fundamental limitation in their ability to capture and explain the customer's reasoning in the process of determining and selecting the appropriate services or products to fulfill customers' needs. This fundamental limitation severely undermines these systems' ability to incrementally improve their performance in terms of the selectivity of services and products. On one occasion, for instance, I was offered by a well-known online retailer to buy a digital camera *because* (according to the system) I bought a book on the subject of neuroscience previously. However, using deep models (e.g. semantic networks) to represent such a business process, though desirable for its expressive power, may require much more computational resources (e.g. time) for reasoning, which may compromise the system performance. We report, in this article, on a new methodology and technique that represents and associates (customer) contextual information or constraints (CC) to relations (links) in semantic net-based models. The contextual constraints define the conditions for traversing the sub-graphs following the links. Inferences can then be performed within contexts. Systematic evaluation experiments were carried out to test the effectiveness of the method. The results suggest that with our approach the reasoning performance can be improved significantly, and real-time application of deep models in the context of online service and product recommendation is certainly feasible.

Strictly speaking, all human problem solving is done within a context, which constrains the solution space [3, 4]. Even universally applicable solutions are conditioned by the only universe we know of. However, serious interest and research effort in the scientific community of machine intelligence have only been demonstrated in the last ten to fifteen years. Most research in machine intelligence still assumes context is defined a-priori by the investigator and is external to the computational problem solving systems, though the importance of representing and using contextual knowledge or information in a explicit and systematic way has been recognized by more and more researchers and demonstrated by increasing number of theoretical works and application systems [1, 4, 5, 6].


*WWW 2004*, May 17–22, 2004, New York, New York, USA.
ACM 1-58113-912-8/04/0005.

## 2. MODELING BUSINESS IN SEMANTIC NETWORK

Semantic networks [8] can be viewed as a directed graph in which nodes or vertices represent concepts, while the links or arcs between these nodes represent relations among the corresponding concepts. Representing business models and ontology in semantic network has become increasingly popular in recent years. Figure 1 is a snapshot of the graph view of an experimental model for customer financial need analysis and product/service recommendation. (Note: This model and the recommendation system were jointly developed by a group of researchers at IBM Watson Research Center.) The modeling framework consists of set of node types representing high-level business and customer concepts (e.g. behavior, situation, need, products, etc.), and a set of link types representing membership and plausible relationships (e.g. include, trigger, served by, etc.). Figure 2 shows in diagram a small portion of this model. Business concepts or objects (e.g. "need to fund education" and "college savings") are represented as nodes and relations (e.g. "subsumes") between these objects are expressed in the form of links.

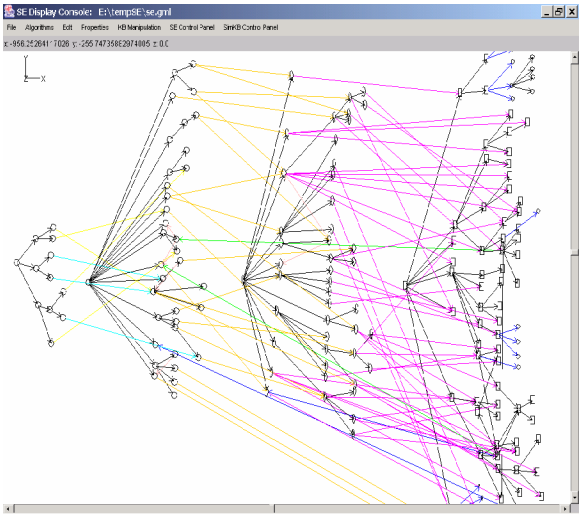

Figure 1: A model for customer financial need analysis

The inference mechanism for this type of semantic net-based model performs two basic functions: evaluation and traversing. If a node is defined by, for instance, a logical expression, the evaluation of the expression (against input data) will result in either "true" or "false", which in turn decides if the inference should continue from the node. From a node evaluated to be true, the system continues the inference by traversing the links originating from it. Not all nodes are defined by logical expression. The semantics of such type of nodes is defined by our common sense or business domain knowledge and manifested by the English sentences or labels associated with the nodes. For instance, "need to fund child's education" is such type of node. Inference reaching such type of nodes does nothing but traversing. The existence of such type of nodes serves the important purpose of the system being articulate and explainable in its reasoning. The reasoning can be regulated or controlled by inference patterns (IP) specific to particular classes of tasks. An inference pattern is an inference template specifying a type of inference path or a set of paths that are considered by the system to be valid. Inference patterns can be defined in the form of designer's choice. We use regular expression to represent inference patterns. For example, the regular expression:

"SituationTriggersNeedServedByProduct"

defines an inference pattern that specifies that *situation* (node) *triggers* (link) *need* (node), and *need* is *served* by (link) *product* (node). Any other possible types of paths will be treated by the inference engine as being invalid, and thus will not be traversed. A more complex inference pattern is defined by the following regular expression:

Situation(ImpliesSituation)*
TriggersNeed(SubsumesNeed)*ServedByProduct
(SubsumesProduct|HasFeatureFeature
(SubfeatureFeature)*)*

Using inference patterns to represent task-specific reasoning strategies prunes the reasoning space and thus improves the inference performance. However, in cases in which multiple tasks are involved, many inference patterns will be defined and used. For very large real world models, potential complexity in traversing the net may still prove to be impractical in real-time environments. What is the problem? The next section tries to answer this question with the given simple example.

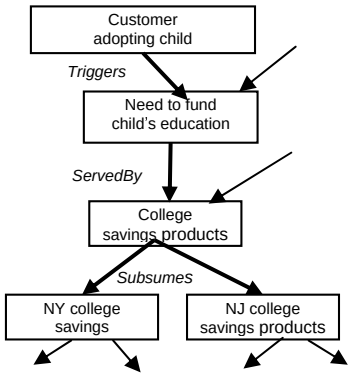


Figure 2: A segment from the model in Figure 1

## 3. PROBLEM

The issue we try to address here is the potential performance breakdown using deep models (e.g. semantic networks) to represent and reason about business processes and customer needs in on-line recommendation systems. Using the example from Figure 2, reasoning starts from an entry node "customer is adopting a child", this type of event or behavior of the customer *triggers* a *need* for planning the education fund for the child.

The *need* can be fulfilled (*ServedBy*) by the college saving *product*. The college saving product contains (*subsumes*) two specific types of savings: NY college savings and NJ college savings. Each type of state-specific college saving may further contain more specific saving products. With this model, the final recommendation will be a list of all college saving products. Since the customer usually only lives in one state, all the college saving products provided for the residents in the other state are

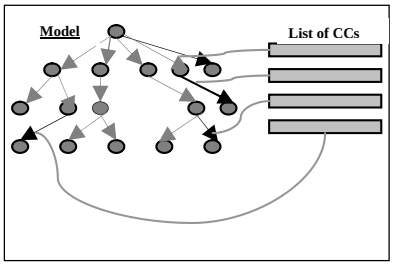


**Figure 3:** Associate contextual constraints (CCs) with relations in the model.

not just redundant, but also incorrect. Imagine reasoning with a full-blown model of this type (e.g. a graph with hundreds of thousands nodes and links illustrated in Figure 2. Not only will the system generate too many incorrect recommendations, but also take long time to derive due to the scale and complexity of the model. This raises a serious question about whether or not such a deep model should be used for web-based product and service recommendation.

## 4. METHOD

Our approach to the problem is to capture and represent a certain type of contextual knowledge (about customers) in the system as a list of contextual constraints (CC). Associate these constraints to the relations (links) in the model, so that contexts for the sub-graph following the links are clearly and properly defined (as illustrated by Figure 3). Inference can be thus performed within contexts, and so constrained the overall inference space defined by the model.

In our modeling framework, a customer's contextual information is classified into two categories: eventual information and demographic information. The eventual information basically refers to important changes or transitions in the customer's career or life which have somewhat significant financial implications. For instance, a person may need an education loan if he or she is graduating from high school and going to college. A person will need auto insurance if he or she is buying a car. The demographic information refers to the status of the customer in many personal aspects such as education (e.g. is a college student), marriage (e.g. single), property ownership (e.g. has a car; has a house), etc. Such kind of information may or may not have direct financial implications of need for financial services or products, but can be very valuable in constraining the reasoning for determining the needed services and products, and thus improving the system selectivity and performance. The eventual information can be naturally used as triggers to initiate the reasoning process to provide customers with the selected service or product recommendations or suggestions that very likely will serve their needs. These two types of contextual information can be conveniently represented in the form of predicate and functions.

As we described previously, the reasoning is carried out following links between nodes. The starting nodes are nodes defined primarily by the eventual contexts (e.g. customer adopting a baby or AdoptingBaby(Customer)). If a particular customer's profile data matches this condition, the node is triggered and a reasoning path starts from that node. All the links originating from the node are traversed except those that are invalid according to the given inference pattern(s). If all the non-starting nodes do not have any associated conditions, the reasoning space can be expanded exponentially and becomes unmanageable if the model is very large. In the simplified scenario shown in Figure 2, the system will reach a list of two recommendations: NY college savings and NJ savings. In the real world model, this list can be very long. If we can represent the customer's residential status as a contextual constraint and associate it to the two links originating from node "college savings", then the system will not embarrassing itself by generating irrelevant recommendations as it did previously without using contextual constraints. Our experiments have shown that associating contextual constraints with relations (links) can be very effective in constraining the system's reasoning.

## 5. EXPERIMENTS AND RESULTS

To systematically evaluate the effectiveness of the approach, we developed an integrated experiment testing environment and a set of tools, which consists of three major components: a GUI, a model generator (simulator), and an inference engine. The experiment was first conducted with a relatively small experimental model in the domain of banking and finance, and then with a very large simulated generic model automatically generated by the system's model generator. The evaluation experiments are designed as follows:

- Same models are used in two comparative scenarios: with and without contextual constraints.
- Measure the number of solution node (SN). A solution node is a node representing a valid final result (e.g. a recommendation of a product), and CPU consumptions (in million second).
- A test case consists of input data specified as a set of contextual facts, an inference strategy, and a given model. Each case is executed 20 times and time consumption is measured statistically. A new test case is created by just replacing the old set of contextual facts with a new set of contextual facts randomly selected from a pool of contextual data.
- An IBM ThinkPad T20 (700 MHz, 512MB memory) is used for running the experiments. The testing system and tools is implemented in Java SDK 1.3.

## 5.1. Using a Small Real Domain Model

An experimental model of 190 nodes and 290 links in the domain of finance and banking is used. The model has 20 entry nodes. Two different inference patterns (IP1 and IP2) were used. The initial experiment has tested 10 cases. All results showed very similar improvements of using contextual constrains comparing to the model without using contextual constraints. The measurement for one test case is shown in Table 1. Noticeable, though not very significant improvement is shown using contextual constraints (CC) in comparison to the results

Table 1: Result with real domain model

| | IP1 | | IP2 | |
|---|---|---|---|---|
| | CC | NC | CC | NC |
| SN | 17 | 12 | 27 | 20 |
| CPU | 376 | 360 | 371 | 358 |

without contextual constraints (NC). The insignificance in the system's performance improvement is due the model being small. The number of solution nodes (SN) is reduced (more selective) due to the use of contextual constraints.

## 5.2. Using Very Large Simulated Models

The experiment with the small model, though useful in testing the correctness of the model and the inference engine, it does not tell the real story about the merits of the approach with very large models. Since it is extremely difficult to obtain a real domain model of very large scale to experiment with, a simulated model is used instead. A simulated model is a generic graph with the structure comparable to the real domain model and randomly generated with the scaling factors estimated according to the analysis of business ontology by the domain experts in finance industry. Three scaling factors are considered: depth, the number of root nodes and branching factors. The model generator automatically constructs a model of total 383244 nodes and 384383 links. There are 221 starting nodes. Three inference types were used respectively. The generation of the input data, though random, is controlled by a pre-specified pruning factor (probability of a link being true i.e. the test of contextual constraints associated with the link being true). In all cases, the selectivity (SN) is all significantly improved. The Table 2 shows a representative test results for one set of input data. In the case of inference pattern 1, 63 solution nodes are derived without contextual constraints. In contrast, only 3 solution nodes are produced by using contextual constraints, even though the average CPU cost does not seem to improve a lot. In the case where inference pattern 2 is used, not only the selectivity (from 1482 to 2 solution nodes), but also the average time cost (from over a second to about half second) all greatly improved. In the case of inference pattern 3, the improvement is even greater, where average time cost is apparently too high (almost 8 minutes) for the model without CC.

Table 2: Result with simulated model

| | | IP 1 | IP 2 | IP 3 |
|---|---|---|---|---|
| CPU | NC | 541 | 1121 | 462495 |
| | CC | 526 | 544 | 682 |
| SN | NC | 63 | 1482 | 61227 |
| | CC | 2 | 2 | 73 |

# 6. DISCUSSIONS

Most existing web-based product/service recommendation systems are shy away from using deep models, primarily due to the concern of reasoning being computationally expensive. We believe and demonstrate, through systematic study and evaluation, that the problem of potentially unmanageable reasoning space with models of free links (no associated constraints) can be overcome or significantly alleviated by using customers' contextual information, demographic information in particular as contextual constraints. Our preliminary experiment results have shown that not only the reasoning can be performed in a more focused fashion to eliminate irrelevant outcomes, but also greatly reduce the overall computational complexity, and thus make the use of deep models for web-based customer services more manageable. The time consumption for evaluating input data against contextual constraints is a worthwhile overhead comparing to the significant gain in overall performance. The performance of using deep models can be further improved by combining other techniques such as Bayesian network [7], or simple relation-ranking with which types of relations are ranked according to some domain-specific criteria. Possible solution paths defined by a sequence of highest ranked links (relations) can then be examined first. Determination and extraction of useful contextual information is crucial, and may require thorough understanding of the reasoning processes performed by the business professionals. The contextual data for a particular customer may be obtained from a CRM database, an online questionnaire, an online conversation, or by guessing based on known limited information if none of other ways can be used. The question of how deep models can be used with very limited or incomplete customer data remains an open issue.

# ACKNOWLEDGEMENT

The experimental real domain model, the semantic network system, and the inference engine used in the study reported here were jointly developed by a group of researchers at IBM Watson. They are Doug Riecken, Erik Mueller, Leora Morgenstern, and Moninder Singh, and the author.

# REFERENCES

[1] Brezillon, P. Context in problem solving: a survey, The Knowledge Engineering Review, 14(1):1-34, 1999.

[2] Fink, J., and Kosba, A. A review and analysis of commercial user modeling servers for personalization and world wide web, Journal of User Modeling and User-Adapted Interaction, Kluwer Academic Publishers, 10: 209-48, 2000.

[3] Gong, L., and Kulikowski, C.A. Composition of image analysis processes through object-centered hierarchical

planning, IEEE Transaction on PAMI, Vol. 17, No.10, pp. 997-1009, 1995.
[4] Gong, L., Exploring Computational Mechanism for Contexts. IEEE Computational Intelligence Bulletin, 2002: 1(1):19-25.
[5] Guha, R. Contexts: formalization and some applications, PhD thesis, Stanford Univ. 1995.
[6] McCarthy, J., Notes on formalizing contexts, Proceedings of 13th IJCAI, Vol. 1, pp. 555-60.
[7] Ji, J., Sha, Z., and Liu, C., Zhong, **N.**, Online Recommendation Based on Customer Shopping Model in E-Commerce. In Proceedings of IEEE/WIC International Conference on Web Intelligence (WI'03), pp. 68-74, 2003.
[8] Quillian, M. Semantic Memory, in M. Minsky (ed.), Semantic Information Processing, pp 227-270, MIT Press, 1968.
[9] Sinha, R., and Swearingen, K. Comparing Human Recommenders to Online Systems, In Proceedings of the Delos-NSF Workshop on "Personalization and Recommender Systems in Digital Libraries", June 01