POOLSIDE: An Online Probabilistic Knowledge Base for Shopping Decision Support

Ping Zhong, Zhanhuai Li, Qun Chen, Yanyan Wang, Lianping Wang,
Murtadha HM Ahmed, Fengfeng Fan
School of Computer Science Northwestern Polytechnical University
127 West Youyi Road, Xian,P.R.China, 710072
zhongping@mail,lizhh@,chenbenben@,wangyanyan@mail,wanglp@mail,a.murtadha@mail,fanfengfeng@mail.nwpu.edu.cn

ABSTRACT

We present POOLSIDE, an online PrObabilistic knOwLedge base for ShoppIng DEcision support, that provides with the on-target recommendation service based on explicit user requirement. With a natural language interface, POOLSIDE can answer question in real-time. We present how to construct the knowledge base and how to enable real-time response in POOLSIDE. Finally, we demonstrate that Poolside can give high-quality product recommendations with high efficiency.(The demo video can be accessed via the link:https://www.youtube.com/watch?v=D8ALi11CUcc)

CCS CONCEPTS

•**Mathematics of computing** →*Probabilistic reasoning algorithms;* •**Information systems** →*Decision support systems;* Online shopping;

KEYWORDS

knowledge base, decision support system, markov logic network

1 INTRODUCTION

The existing shopping decision support systems [2] focus on product price comparison or product recommendation based on user's past shopping behaviors. Unfortunately, none of them provides with the on-target service that can recommend products based on explicit user requirements. The challenge of providing such service results from the observation that the user-specified requirements may involve not only the basic attributes of products but their multi-aspect and more obscure concepts. For instance, a user may ask the system to recommend a mobile phone priced around 500$ and with high performance. The concept of high performance is composite and obscure. It should be evaluated on various factors including memory size, CPU frequency and number, and most importantly, the user comments. To this end, we propose an online knowledge base, denoted by POOLSIDE, that can support real-time decision making. POOLSIDE provides a natural language interface and uses Deepdive [3], the state-of-the-art KB tool, to facilitate reasoning about obscure concepts. POOLSIDE is an ongoing project. Our major contributions can be summarized as follows:

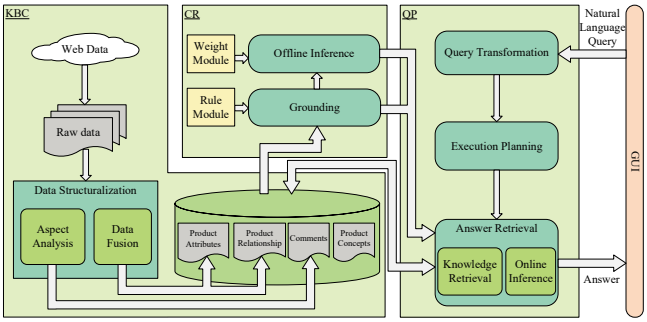


Figure 1: System Overview

- We develop the demo system POOLSIDE that can recommend products based on explicit user requirement in real-time. We outline the major challenges of building POOLSIDE, concept reasoning and real-time response, and present the corresponding solutions in Section 4 and 5;
- We demo how POOLSIDE interacts with users and provides high-quality product recommendations with high efficiency; (Section 6)
- Based on our experience with POOLSIDE, we identify two future directions for the research on probabilistic knowledge base. (Section 7)

2 SYSTEM OVERVIEW

Figure 1 gives an overview of the POOLSIDE system. It represents all knowledge facts by first-order relations, which are stored in relational databases (Postgresql in our demo). It consists of three components, Knowledge Base Construction (KBC), Concept Reasoning (CR) and Query Processing (QP).

KBC extracts data from Web and transforms them into structured data. It has a data fusion module that can merge product informations from different sources. It also contains an aspect analysis module that performs aspect and sentimental analysis on natural language comments. CR is responsible for reasoning about obscure product concepts. It is built on the KB tool Deepdive and provides a rule module and a weight module to facilitate rule definition and rule weight setting respectively. QP transforms a natural language question into a SQL query and retrieves answers from the knowledge base in real-time.

CIKM'17, November 6–10, 2017, Singapore.
 ISBN 978-1-4503-4918-5/17/11…$15.00
DOI: http://dx.doi.org/10.1145/3132847.3133168

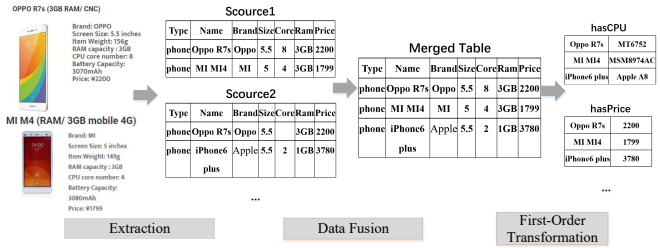


Figure 2: Basic Product Information Extraction

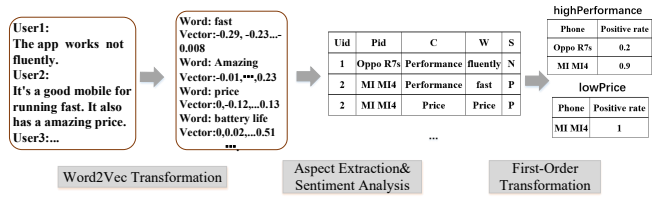


Figure 3: Aspect Analysis on Comments

3 KNOWLEDGE BASE CONSTRUCTION

Knowledge base construction involves processing both basic product attribute information and user comments. The workflow of processing basic product attribute information is presented in Figure 2.We extract productsfi data from business website by a crawler tool BaZhuaYu[1]. . For each product type, it first directly extracts basic attribute values of products (e.g. price and memory size of a mobile phone) from different e-commerce Web pages and then merges them into a unified relational data. It also extracts product relationship information (if it exists) for concept reasoning. For instance, two products are considered to be similar if they are listed as competing products on a shopping site.

The workflow of processing user comments is shown in Figure 3. Aspect analysis identifies the product aspect a user comments on and the sentiment of the comment (positive or negative). Our solution for aspect identification and sentimental analysis are based on the NLP tool Word2Vec [1] and [4]. It transforms a comment into a 6-tuple, (*Uid*,*Pid*,*T*,*C*,*W*,*S*), in which *Uid* and *Pid* denotes the identities of user and product respectively, *T* the time of comment being submitted, *C* the product aspect commented by user, *W* the string of comment keywords, and *S* the result of sentiment analysis (positive or negative). For example, in Figure 3, the comment of user2 contains the terms of "running" and "fast". KBC would classify the comment into the aspect class of *performance* and also understand that the sentiment of the comment is positive.

4 CONCEPT REASONING

To answer the query containing obscure concepts, POOLSIDE treats these concepts as uncertain first-order relations stored in probabilistic knowledge base. It uses Deepdive, a state-of-the-art probabilistic knowledge base construction tool based on Markov logic network (MLN), to reason about obscure concepts. Deepdive represents a MLN by a factor graph and reasons the probabilities of uncertain knowledge using Gibbs sampling. In Deepdive, factor graph is constructed by rules. In the rest of this section, we describe how to specify the rules and set their weights in POOLSIDE.

[1]http://www.bazhuayu.com/download

4.1 Knowledge Rule Generation

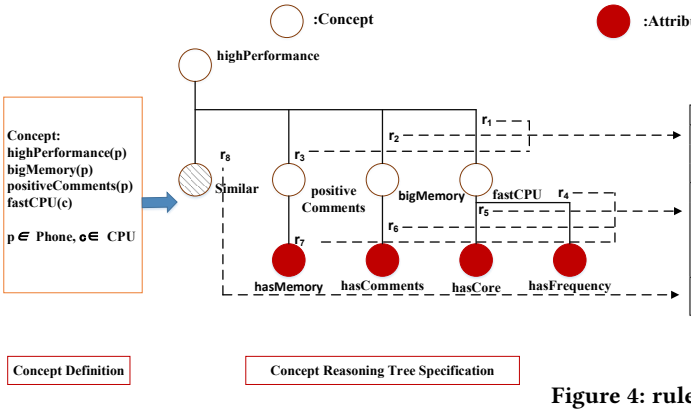
Given a target concept, the rule module generates the rules according to a concept reasoning tree, which needs to be specified by users. A concept reasoning tree is directed, and consists of three types of nodes, including concept node, attribute node and product relationship node. A concept node corresponds to a product-relevant concept. An attribute node corresponds to a product attribute or user comments. A product relationship node has the target to specifiy the relationship between two products. An edge from a parent to a child means that the evaluation on the child node would have an impact on the evaluation on the parent node.

An example of concept reasoning tree is shown in Figure 4 (a). Its root specifies the target concept that needs to be reasoned, in this example it is labeled "highPerformance". The concept nodes also include "bigMemory", "positiveComments" and "fastCPU". Their evaluation would influence the evaluation of their parent node "highPerformance". The node of "Similar" is a product relationship node. The edge between "highPerformance" and "Similar" dictates that if two products are similar, high performance on one of them would mean that another one is also of high performance. The nodes of "Memory", "Core", "Frequency" and "Comments" are attribute nodes. The edge between "bigMemory" and "Memory" dictates that a product's attribute value at memory has an impact on the evaluation of the concept "bigMemory". It can be observed that in a concept reasoning tree, an internal node should be a concept node, and the leaf nodes can be either attribute nodes or product relationship nodes.

The rule model automatically translates a concept reasoning tree into a set of rules. An example of the translation is shown in Figure 4. Each edge in the tree corresponds to a generated rule. Formally, given an edge between two concept nodes, $C_i \rightarrow C_j$, in which C_i and C_j denote two concepts, its corresponding rule can be specified by $C_j(p) \rightarrow C_i(p)$, in which p denotes a product ID. An edge between a concept node and an attribute node, $C_i \rightarrow A_j$, in which A_j denotes the label of the attribute node, can be translated into the rule: $A_j(p, a_k) \rightarrow C_i(p)$, in which a_k denotes the attribute value of p at A_j. An edge between a concept node and a product relationship node, $C_i \rightarrow R_j$, in which R_i denotes the relationship label, is translated into the rule: $C_i(p_l), R_i(p_l, p_k) \rightarrow C_i(p_k)$. For instance, in Figure 4, the edge between "highPerformance" and "bigMemory" corresponds to the rule r_1: $bigMemory(p) \rightarrow highPerformance(p)$. The edge between "bigMemory" and "Memory" corresponds to the rule r_2: $Memory(p, m) \rightarrow bigMemory(p)$.

4.2 Rule Weight Tuning

POOLSIDE labels some nodes in a factor graph and then uses the training mechanism provided by Deepdive to learn the rule weights in the factor graph. For instance, in the factor graph for reasoning about performance of mobile phone, it can label some phones as high performance beforehand according to user comments. Unfortunately, it can be observed that Deepdive was primarily designed to support reasoning about the knowledges defined over string values, but not numerical values. As a result, the learned weight assignment may not be consistent with the plain knowledge involving numerical value comparison. For instance, in Figure 5, the rule r_3, $Memory(p, 2GB) \rightarrow bigMemory(p)$, has even a lower weight



Groups	Knowledge Rules	Weight
$C_i \rightarrow C_j$	$r_1 : hasCPU(p, c), fastCPU(c) \rightarrow HighPerformance(p)$	3.0
	$r_2 : bigMemory(p) \rightarrow HighPerformance(p)$	2.0
	$r_3 : positiveComments(p) \rightarrow HighPerfomance(p)$	2.0
$C_i \rightarrow A_j$	$r_4 : hasFrequecy(c, h) \rightarrow fastCPU(c)$	$f(h)$
	$r_5 : hasCore(c, n) \rightarrow fastCPU(c)$	$f(n)$
	$r_6 : hasMemory(p, m) \rightarrow bigMemory(p)$	$f(m)$
	$r_7 : PerformanceComments(p, v) \rightarrow positiveComments(p)$	$f(v)$
$C_i \rightarrow R_j$	$r_8 : HighPerformance(p_1), Similar(p_1, p_2) \rightarrow HighPerformance(p_2)$	1.0

Figure 4: rule module

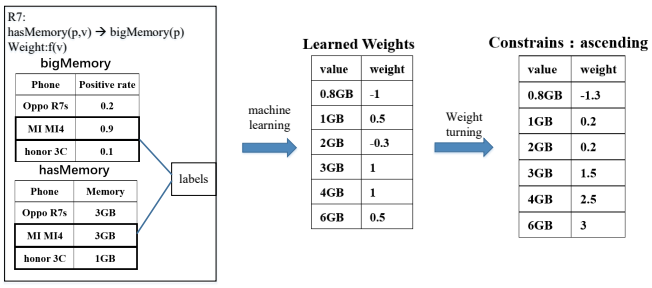
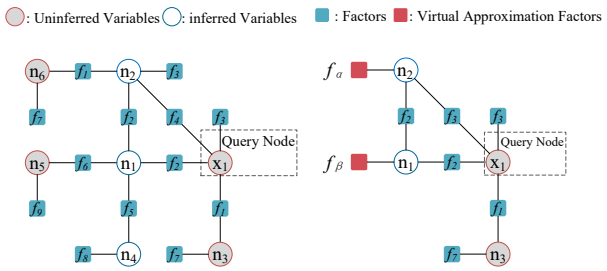


Figure 5: weight module



(a) original factor graph (b) approximation subgraph

Figure 6: Online Inference: An Example

than the rule r_4, $Memory(p, 1GB) \rightarrow bigMemory(p)$. However, r_3 should have a larger weight than r_4 because the memory of 2 GB is indeed bigger than the memory of 1GB. To overcome the shortcoming of Deepdive's weigh learning mechanism, we tune the learned weights such that they satisfy the monotonacity relationship: a higher attribute value would consistently result in better (or worse) evaluation on its corresponding concept. Formally speaking, let $X = \{x_1 \cdots x_n\}$ denotes the weight vector sorted by its attributes values(better or worse), it tunes each $x_i \in X$ to $\hat{x}_i$ by following fomula:

$$\hat{x}_i = x_i + \sum_{j=1}^{i-1} s(x_j)x_j + \sum_{j=i+1}^{n} s(-x_j)x_j \quad (1)$$

where $s(x)$ is a signal function where $s(x) = 1$ if $x > 0$ and $s(x) = 0$ otherwise. In Figure 5, after tuning, the weight of the rule, $Memory(p, m) \rightarrow bigMemory(p)$, increases with memory size.

5 ENABLING REAL-TIME RESPONSE

The whole factor graph constructed by Deepdive can be very large due to the large number of different products. Therefore, probabilistic inference over all the variables in the resulting factor graph is usually very time-consuming, and unnecessary as well because not all the products are interesting to users. Even though the technique of k-hop approximation [5] can be used to speed up inference, it remains very challenging to simultaneously achieve good-quality results and real-time response. To address this challenge, we propose a novel query-driven online inference technique, which can achieve both good-quality inference results and real-time response by reusing the inferred values of the variable nodes.

The online technique labels the nodes whose values have been inferred as inferred nodes. When a variable node v_i has to be inferred as quested by user, it first identifies v_i's neighbors of inferred nodes, then creates virtual factor nodes to approximate the influence of the factor graph on the inferred nodes, and finally infers the probability of v_i based on the constructed small graph. An example of visual factor construction is shown in Figure 6. The size of subgraph(Figure 6(b)) is small that can be inferred in real-time.

The process of online inference consists of three steps: subgraph extraction, visual factor construction and sugraph inference. Suppose that the original factor graph is denoted by G. The step of subgraph extraction searches for the limited-sized subgraph for reasoning about v_i in G. It searches for the k-hop subgraph of n_i in G in a breadth-first manner. However, the search process stops at any inferred node. Suppose that the resulting factor subgraph is denoted by G_k. The second step of visual factor construction constructs a visual approximation factor for each inferred node in G_k to simulate the inference influence of G/G_k on G_k. We denote the resulting factor subgraph with visual factors as $\hat{G}_k$. In $\hat{G}_k$, any inferred node v_j should satisfy

$$\hat{P}(v_j) = P(v_j) \quad (2)$$

in which $P(v_j)$ denotes v_j's inferred probability on G and $\hat{P}(v_j)$ denotes v_j's inferred probability on $\hat{G}_k$.

Now we describe how to estimate the weights of inserted visual factors in $\hat{G}_k$. Suppose that V denotes the variable set in G_k, m denotes the number of factors in G_k, and $\hat{m}$ denotes the number of inserted visual factors in $\hat{G}_k$. f_i $(1 \le i \le m)$ denotes the factor function of a factor in G_k, and $\hat{f}_j$ $(1 \le j \le \hat{m})$ denotes the factor

function of a visual factor in $\hat{G}_k$. The factor function of $\hat{f}_j$ corresponds to the variable (node) v_j in $\hat{G}_k$. Note that we have $\hat{f}_j$=1 if v_j=0, and $\hat{f}_j$=e_j^w if v_j=0, where w_j denotes the visual factor weight. Therefore, for each inferred node v_p, the condition specified in Equation 2 corresponds to the following equation

$$p(v_p) = \frac{1}{Z} \sum_{V \setminus v_p} (\prod_{i,j} f_i \cdot f_j^*), 1 \le i \le m, 1 \le j \le \hat{m}, \qquad (3)$$

where Z is a normalization constant.

Since there are totally $\hat{m}$ inferred nodes (visual factors), we have to solve the equation group consisting of $\hat{m}$ equations of $\hat{m}$ order. It can be observed that the equation group can be easily solved if the value of $\hat{m}$ is small. In case that the value of $\hat{m}$ is large, we also propose a divide-and-conquer approach to speed up the process of weight estimation. It first splits $\hat{G}_k$ into multiple subgraphs and then solves their corresponding equation groups independently.

Since the resulting factor subgraph $\hat{G}_k$ is usually small, we can use exact inference algorithms (belief propagation algorithm in our demo) in the final step of probability inference over $\hat{G}_k$.

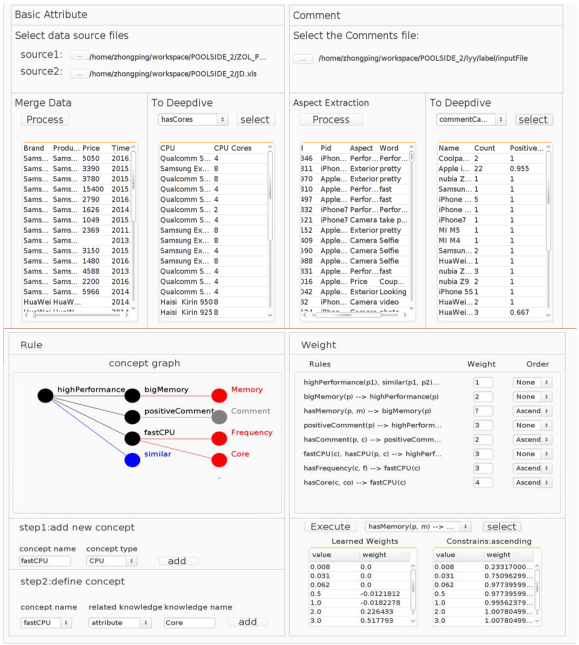


Figure 7: GUI Screenshots

6 DEMONSTRATION PLAN

To construct a mobile phone knowledge base, we use POOLSIDE GUI to construct the product KB in an interactive way. It first runs KBC module to transform data into first order knowledge, and then let the user to define the concepts and choose its relating knowledge from existing knowledge files. After that, GUI runs Rule module to generate knowledge rules according to user's input and create factor graph by Deepdive. Finally, GUI demands user to specify the directory of label data file and then infers the concepts.

The query demonstration consists of three parts: query interface, recommendation and product detail presentation. The demo will use the KB system that we have built for the mobile phone products.

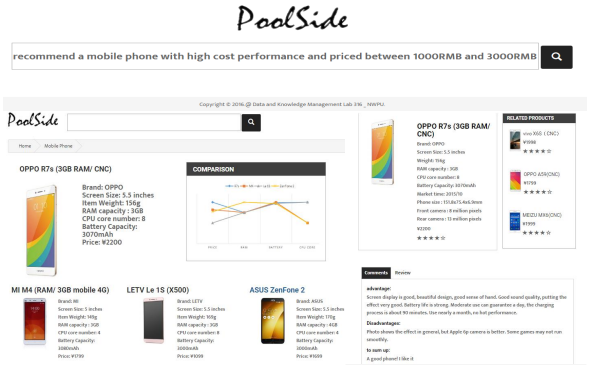


Figure 8: Interface Screenshots

The query interface accepts the user query. The interface of recommendation lists the products satisfying a user query and orders them by user-specified attributes/concepts. Finally, clicking on products hyperlinks on the page of recommendation would take you to a new page detailing its major properties and strengths/weaknesses compared with other popular products. The results recommended by POOLSIDE are very similar to what are reported on the professional mobile phone testing website Zealer[2].

7 THOUGHTS ON FUTURE WORK

Our work on POOLSIDE points out two interesting directions for future research on probabilistic knowledge base. Firstly, the inference engines of the existing probabilistic KBs are optimized for text values. They are usually clumsy in handling the knowledges defined on numerical value comparisons, which can however be richly found in real applications. For instance, a phone with a faster CPU should be considered to have correspondingly higher performance. Secondly, the reasoning rules in a knowledge base currently have to be specified by experts beforehand. Automatic rule detection can greatly reduce human workload and significantly improve the intelligence of KB systems as well.

ACKNOWLEDGMENTS

This work is supported by the Ministry of Science and Technology of China, National Key Research and Development Program (Project Number:2016YFB1000703), the Natural Science Foundation of China under Grant No.61332006, No.61672432, No.61472321 and No.61502390.

REFERENCES

[1] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. *Computer Science* (2013).
[2] Bhavik Pathak. 2010. A Survey of The Comparison Shopping Agent-based Decision Support Systems. *Journal of Electronic Commerce Research* 11, 3 (2010), 178–192.
[3] Christopher De Sa, Alex Ratner, Christopher R, Jaeho Shin, Feiran Wang, Sen Wu, and Ce Zhang. 2016. Incremental knowledge base construction using DeepDive. *Vldb Journal* (2016), 1–25.
[4] Dongwen Zhang, Hua Xu, Zengcai Su, and Yunfeng Xu. 2015. Chinese comments sentiment classification based on word2vec and SVM perf. *Expert Systems with Applications* 42, 4 (2015), 1857–1863.
[5] Xiaofeng Zhou, Yang Chen, and Daisy Zhe Wang. 2016. ArchimedesOne: Query Processing over Probabilistic Knowledge Bases. *Proceedings of the VLDB Endowment* 9, 13 (2016).

[2]http://tool.zealer.com/