SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

POOLSIDE: An Online Probabilistic Knowledge Base for Shopping Decision Support

Ping Zhong, ZhanHuai Li, Qun Chen, Yanyan Wang, Lianping Wang, JiaMing Li

December 29, 2016

1 ABSTRACT

We present POOLSIDE, an online PrObabilistic knOwLedge base for ShoppIng DEcision support. POOLSIDE provides with a natural language interface and answers questions in real-time. We present how to construct the knowledge base and also how to enable real-time response in POOLSIDE. Finally, we demo that POOLSIDE can provide with high-quality product recommendation with high efficiency.

2 INTRODUCTION

Even though a wide variety of shopping decision support systems [1] have been developed, none of them provides with the on-target service that can recommend products based on explicit user requirements. The challenge of providing with such service results from the observation that the user-specified requirement 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 user comments. To this end, we propose an online knowledge base (KB) POOLSIDE that can support real-time decision making. POOLSIDE has a natural language interface and uses Deepdive [2], the state-of-the-art KB tool, to facilitate reasoning about obscure concepts. Our major contributions can be summarized as follows:

- We develop the demo system POOLSIDE that can recommend products based on explicit user requirements in real-time . We outline the major challenges of building POOLSIDE and present corresponding solutions; (Section 3 - Section 6)
- We demo how POOLSIDE interacts with users and provides with high-quality product recommendations with high efficiency; (Section 7)
- We identify two directions for future research on probabilistic knowledge base. (Section 8)

3 System Overview

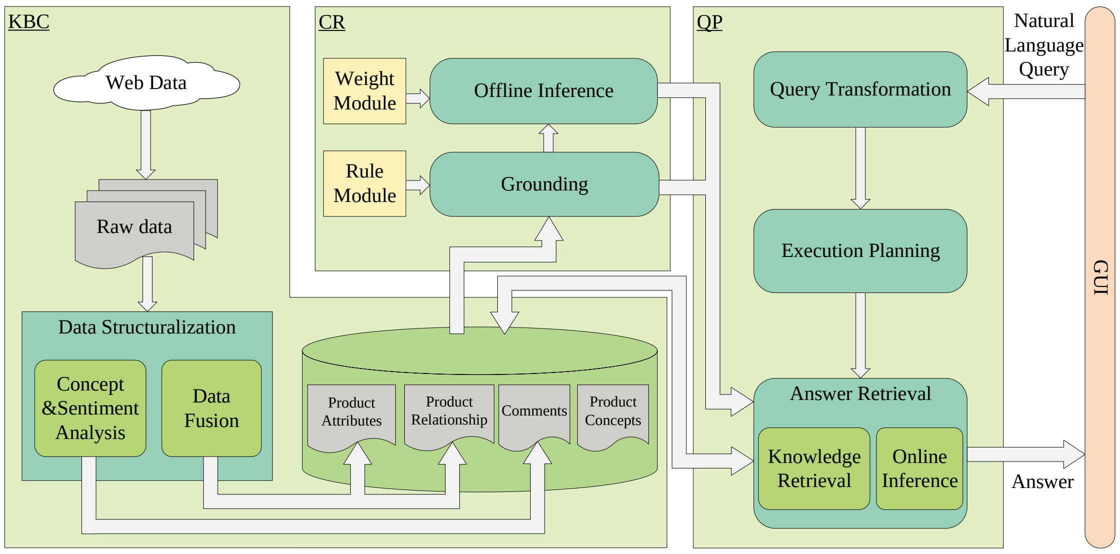


Figure 3.1: System Overview

Figure 3.1 gives an overview of the POOLSIDE system. It represents all knowledges by first-order relations, which are stored in relational databases (e.g. Greenplum). It consists of three components, Knowledge Base Construction (KBC), Concept Reasoning (CR) and Query Processing (QP). KBC extracts data from Web and transform them into structured data. CR is responsible for reasoning about obscure product concepts. This off-line procession runs periodically to enrich knowledge in Poolside. QP transforms a natural language question into a SQL query and retrieves answers from the knowledge base. When a query is not inferred, QP need to run a online inference algorithm to give a approximation result in real-time.

4 KNOWLEDGE BASE CONSTRUCTION

KBC module extracts data from web and then make them into structured data. The extracted data are stored as three kinds of first-order relations in knowledge base: product attributes, comments and product relationship.

4.1 PRODUCT ATTRIBUTES AND PRODUCT RELATIONSHIP

Product attributes(e.g. price and memory size of mobile phones) and relationship(like similar between two products) are directly extracted from e-commerce Web pages.These data are well-structured, such as XML format, which can be directly downloaded into relational data tables. The extraction module first retrieves data tables from different web sources, then data fusion module merges these data into a consistent table. In this procession, we first manually appoint the most credible source. For each tuples in rest data sources, data fusion module computes the similarities between the tuple to the tuple in credible source for matching the same product. After that, the tuples been considered to represent the same product are merged into the credible data table. At last, these data are converted into first-order relations.

Product attributes are fundamental data in building our knowledge-based system. These data are field knowledge which can be directly used for user's query. Moreover, the attributes values are important part in evaluation of products. For example, the evaluation of a mobile phone's performance is decided by its CPU performance and memory size. Product relationship information would be similarly used for reasoning about obscure concepts like product attributes. For instance, two products are considered to be similar if they are listed as competing products on a shopping site. This relationship enables the evaluation result of a product can influent on others.

4.2 USER COMMENTS

To enable reasoning about obscure concepts (e.g. high performance and performance price ratio), it also extracts user comments for conceptual and sentimental analysis. Conceptual analysis identifies the product concept a user comments on. Our solution based on the NLP tool of Word2Vec [3] 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 concept commented by user, W the string of comment keywords, and S the result of sentimental analysis (positive or negative).

We propose an method based on Word2vec to extract products' concepts in comments: For each comment, the method first uses NLP tool to obtain word segments, then transforms the segments into a word vector by using word2vec. The word vector usually can not be directly used since the dimension of vector is too high such that we apply principal component analysis(PCA) to reduce the dimensionality of the vector. To find the concept in comments, we use the concepts defined in knowledge base as class label and apply the same procession above to transform the word of concepts into word vectors. For each vector of a word segment, we compute the cosine similarity between word segment and class labels. After that, each comment contains a group of word vectors representing word segments in that comment. We choose top-k(usually $k = 3$) word segments by cosine similarity for each concept class label and compute their average values to represent the similarity between a comment to a concept. We choose top-k words because many words in comments are meaningless to concept and need to be ignored. Finally, the concept of comment is the class which have highest average value. We use method to analyze sentiment in comments. We manually construct a lexicon to store sentiment words and its weight. Then we uses word segments as feature to compute the sentiment of comment.

Poolside makes use of these comment in two ways: first, they can be used directly as knowledge base facts like product attributes. Second, few comments with high confidence can be used as machine learning labels to train knowledge-rule weights.

5 Concept reasoning

Poolside use knowledge base construction tool Deepdive to infer the concepts defined by knowledge rules. In our probabilistic knowledge base, each obscure concept is represented as an uncertain first-order relation. For each concept, we design knowledge rules to infer the obscure concepts based on knowledge base facts, which extracted from web in section 4

5.1 Probabilistic Inference Engine

Probabilistic knowledge base is a knowledge base which supports uncertain rules and facts. State-of-art probabilistic knowledge base reasoning is based on Markov logic network [4] (MLN). Deepdive extended MLN by transferring MLN into a factor graph and reason the probability by using Gibbs sampling. In Deepdive, a knowledge rule r formulates as $r = (F, W), F$ denotes a deterministic rule and W is the weight indicate how likely the rule is true. To compute the marginal distribution for uncertain relations over a set of rules, MLN inference contains two procession: grounding and inference. Grounding is the construction of factor graph. Factor graph consists of two kinds of nodes:binary random variables and factors, each kind of nodes are connected by the others. Each random variable in the graph represents an uncertain first-order relation or an evidence fact. The value of random variable means whether the relation holds. Factors are functions over variables indicating the casual relationship between these variables. Each knowledge rule creates a factor connecting the relevant variables after grounding. All factors in the graph determine the joint distribution of random variable:

$$p(X = x) = \frac{1}{Z}\prod_i f_i(X_i) = \frac{1}{Z}exp(\sum_i W_i n_i(x)) \tag{5.1}$$

where X is vector of random variables, $n_i(x)$ denotes the number of true groundings of rule in x, Z denotes the normalization constant, $f_i(X_i)$ denotes factor function where $X_i \subseteq X$ contains the variable connected by f_i. The value of $f_i(X_i)$ is decided the knowledge rule it respects to. For example, supposing there is a knowledge rule $r : relationA(x_1, x_2), relationB(x_2, x_3) \rightarrow relationC(x_1, x_3)$ with weight w, then we have $X_i = (x_1, x_2, x_3)$. The function can be computed

Table 5.1: Concept Reasoning by Rules: An Example

Groups	Knowledge Rules	Weight
Group 1	$r_1 : bigMemory(p) \rightarrow HighPerformance(p)$	2.0
	$r_2 : hasCPU(p,c), fastCPU(c) \rightarrow HighPerformance(p)$	3.0
	$r_3 : positivePerformanceComments(p,b) \rightarrow HighPerfomance(p)$	$f(b)$
Group 2	$r_4 : hasMemory(p,m) \rightarrow bigMemory(p)$	$f(m)$
	$r_5 : hasFrequecy(c,h) \rightarrow fastCPU(c)$	$f(h)$
	$r_6 : hasCore(c,n) \rightarrow fastCPU(c)$	$f(n)$
Group 3	$r_7 : HighPerformance(p_1), Similar(p_1,p_2) \rightarrow HighPerformance(p_2)$	1.0

by follow formula:

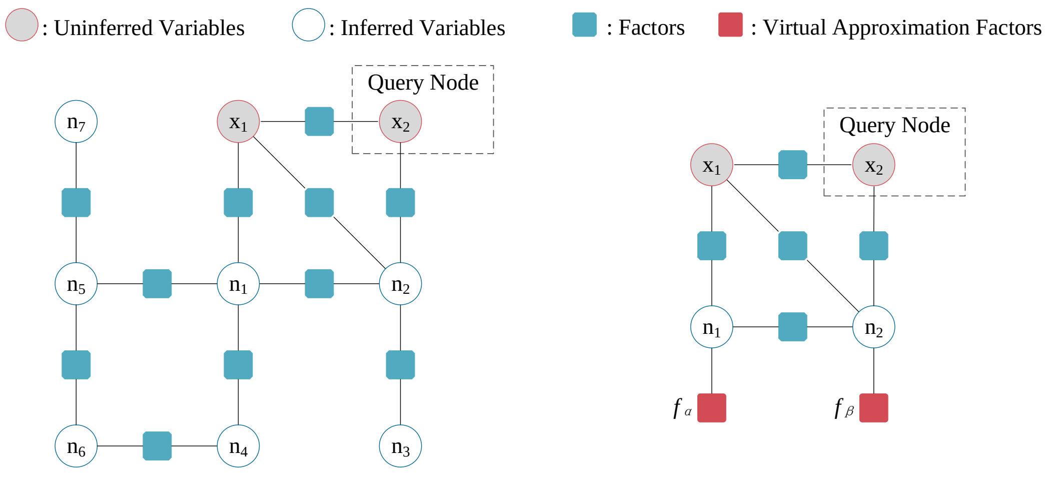
$$f_i(x_1,x_2,x_3) = \begin{cases} 1 & if(x_1,x_2,x_3) = (1,1,0) \\ e^w & otherwise \end{cases} \tag{5.2}$$

In inference procession, current Deepdive uses MCMC sampling on the factor graph to reason the marginal distribution. This procession requires plenty of time to compute all variables in factor graph each execution so that it can only be used in off-line scenarios.

5.2 Knowledge Rule Module

POOLSIDE uses Deepdive to reason about obscure product concepts by rules. An example of reasoning about high performance of mobile phones is showed in Table 5.1. The first group of rules (r_1 to r_3) specifies that the aspects of memory, CPU and percentage of positive comments should be considered for evaluation of high performance. The second group of rules (r_4 to r_6) specifies how the basic attribute values of a phone influence the evaluation aspects of high performance. Finally, the rule r_7, which is built on product relationship information, specifies that a phone can be probably considered to be of high performance if it has a competing product with high performance.

From the above case, Poolside system rules can be divide into three kind of groups, respecting to the types of facts , for reasoning the obscure concepts: a) Direct rules: We use this type of rules to specify the knowledge having impact on the target concept(e.g high Performance in Table 5.1). Some relations in the left-hand-side(LHS) of the rule are knowledge which reflect the evaluation to the attributes. For instance, r_1 the relation $bigMemory(p)$ is decided by a single attribute specified in r_4:the memory size of the phone. These relations are also obscure concepts can be queried by users. Some LHS relations are comments to the concepts. The user comments are less reliable, incomplete and inconsistent, therefore, the weight of comments should be evaluated by their positive or negative comments rate. b) Indirect rules: These rules specifies attributes values to their evaluation. For example, memory size to a big-memory phone. There is a problem that the evaluation of a attribute is subjective. Different people have different answers on the question like "For a mobile phone, how large the memory-size can be considered as big". To solve this problem, we use the comments we trust(with high positive or negative comments rate and numbers) as the labels, and use machine-learning in Deepdive and weight module to compute the weights for evaluation. c) production relationship rules: These rules represent the relationship between products and connect different products in factor graph, as a result, products can have impact on the others in probabilistic reasoning.



(a) original factor graph (b) approximation factor graph

Figure 6.1: Online Inference

5.3 Weight Module

In probabilistic knowledge bases, each rule has its own weight, which can be set beforehand or learned. This means that every factor has the same weight after grounding. However, Poolside aims to recommend the best candidate for the query. To address this issue, we should expect that a phone with 4GB memory has a higher probability to be considered as high-performance than another one with only 3GB memory. It provides with a weight module, which extends Deepdive's weight learning mechanism, to achieve this purpose. The weight module makes each factor in factor graph has its own weight. Weight module also implements the logic of numeric data such as the greater number of attribute values should have better results. The user only needs to simply specify this logic in Poolside configuration file.

6 Query Processing

The QP component receives a natural language query, transforms it into a SQL query and then retrieves answers from the knowledge base. The answer retrieval module first query the database for knowledge in need. If query concept are uninferred, system runs online inference algorithm for a approximation answer in real-time.

6.1 Query Understanding

Poolside uses the method in section 4.2 and a predefined dictionary to extracted 5-tuple(*product type,brand,product series, knowledge,Knowledge description*) in the query statement. The first three types of data are used to specify the unique product in our knowledge base. The knowledge means product attributes, comments, relationship and obscure concepts stored in knowledge base. The description of knowledge adjective words for a concept evaluation such as "high or fast". In a real-world query, some data int this 5-tuple may absent. We design different database operations for answer retrieval and result representation according to the composition of data presenting in 5-tuples. For example, a query specifies product type, brand and series but contains no knowledge and description possibly want to retrieve information for a specific product, while a query contains only concepts may ask system to recommend products.

6.2 Online Inference Algorithm

The whole factor graph constructed by Deepdive can be very large because of the large number of different products. Therefore, probabilistic inference over all the variables in the factor graph is usually very time-consuming, and unnecessary as well because not all the products are interesting. Even though the technique of k-hop approximation [5] can be used to speed up the inference, it remains very challenging to simultaneously achieve good-quality results and real-time response.

To address this challenge, we propose a pay-as-you-go mechanism of incremental inference. It achieves real-time response by reusing the inferred probabilities of variable nodes(concepts) in the factor graph. It labels the nodes whose probabilities have been inferred as evidence nodes. When a variable node N_i has to be inferred as quested by user, it identifies N_i's neighbors of evidence nodes and then create virtual factor nodes to approximate the influence of the factor graph on these evidence nodes and N_i. An example of the incremental inference has been shown in Figure 6.1.

6.2.1 Virtual Approximation Factor Computing

Supposing there is a inferred variable node n in factor graph G with its marginal distribution $P(n)$, we can create a factor node f_α with weight α connect to n. After that We delete all nodes in factor graph except n and f_α and recompute the distribution $\hat{P}(n)$ for n. If the distribution $\hat{P}(n)$ is same to $P(n)$, we consider f_α and its weight α represents influence of G/n to n. We denote the factor f_α as virtual approximation factor. Moreover, we can create a virual approximation factor to represent any influence from other nodes if

$$\hat{P}(n) = P(n) \tag{6.1}$$

In a knowledge base for real-world products, its factor graph would be extramely large and not every nodes in graph are interesting. Therefore, our system only infers part of nodes which are hot for queries or having many edges. The other nodes use a query-driven method to return the result in real-time. Online inference algorithm uses virtual approximation factor to acheve better accuracy on small subgraph.

From this point of view, when user queries for a uninferred variable x, algorithm searches every edges from x until the inferred nodes, this operation creates a subgraph denoted as SG, the inferred nodes in SG denoted as $v(x)$. For each variable $n_i \in v(x)$, algorithm uses a virtual approximation factor f_i connecting to n_i. Every virtual factor represents the influence passing through n_i to x. If we know the weights of each virtual factor, we can run inference algorithm on the subgraph SG with virtual approximation factors added on $v(x)$.

To compute the weight of virtual approximation factor, the algorithm can be illustrated from a simple example. Supposing there is a uninferred note x connecting to an inferred note y with rule $r:(x \to y, w)$. The distribution of y denotes as $P(y)$. The weight of virtual approximation factor f_y, denoted as w_y, can be calculated by solving the equation build by the formula 6.1 , 5.1 and 5.2 such that

$$\hat{P}(y) = \frac{1+e^{w_y}}{1+3e^{w_y}} = P(y) \tag{6.2}$$

$$w_y = ln\left|\frac{P(y=1)-1}{1-3P(y=1)}\right| \tag{6.3}$$

For more nodes connecting to x, our solution builds an equation group for computing the weight of each factor. Supposing W is the set of all virtual approximation factors' weights such that $w_i \in W$. For each $n_i \in v(x)$, its marginal distribution $p(n_i)$ can be formulated by all the weights in W. Since $|v(x)| = |W|$, there is an equation group consists of $|v(x)|$ equations and the same number of unknown weights. Solving this equation group we can have all weights of virtual approximation factors.

For example in figure 6.3(a), x_1 and x_2 are uninferred variables connecting to n_1 and n_2 with virtual approximation factors f_α and f_β added like figure6.3(b). The joint distribution of (n_1, n_2, x_1, x_2) can be computed by the product of all factors between the (n_1, n_2, x_1, x_2), in this case

$$P(n_1,n_2,x_1,x_2) = \frac{1}{Z} f_\alpha \cdot f_\beta \cdot f(n_1,n_2) \cdot f(n_1,x_1) \cdot f(n_2,x_2) \cdot f(n_1,n_2) \cdot f(x_1,x_2) \tag{6.4}$$

After marginalizing the formula6.4 can get two equation for $P(n_1)$ and $P(n_2)$.

$$\begin{cases} P(n_1=1) & = \frac{1}{Z}\sum_{n_2 x_1 x_2} f_\alpha(n_1=1)\cdot f_\beta(n_2)\cdot f(n_1=1,n_2)\cdot f(n_1,x_1)\cdot f(n_2,x_2)\cdot f(n_1,n_2)\cdot f(x_1,x_2) \\ \\ P(n_2=1) & = \frac{1}{Z}\sum_{n_1 x_1 x_2} f_\alpha(n_1)\cdot f_\beta(n_2=1)\cdot f(n_1,n_2=1)\cdot f(n_1,x_1)\cdot f(n_2,x_2)\cdot f(n_1,n_2)\cdot f(x_1,x_2) \end{cases} \tag{6.5}$$

Then we solve the equation group and get the weight of the virtual factors.

6.2.2 Inference Algorithm

The uninferred query nodes are not always connect to inferred nodes such that algorithm use BFS on factor graph until meeting inferred nodes or the subgraph reach the max hop. The latter case our algorithm degenerates to k-hop algorithm. In most case, our algorithm can stop early and achieve higher speed in inference and higher accuracy. We illustrate our online inference in algorithm 1.

This Algorithm first extracts a subgraph for query node x (line 2 to 24), and then add virtual approximation factors and computes their weights by using method in section 6.2.1(line 25 to 28). Finally, it runs inference algorithm on this subgraph (line 29). For simplicity, we ignore the factors nodes in the graph in algorithm 1(All nodes in algorithm 1 means variable nodes).

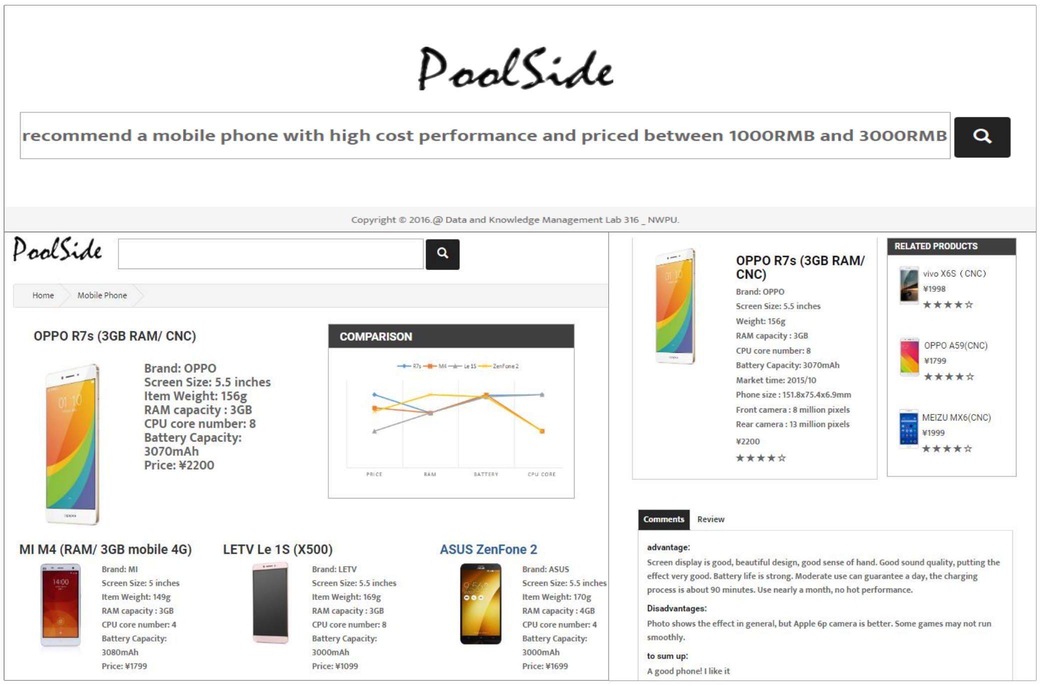
For reasoning the probability on the subgraph, We choose belief propagation(BP) algorithm

Algorithm 1 Online Inference Algorithm

```
Require: Query Node $x$, Factor graph $g$, Inferred Nodes $v_g$, Envidence Nodes $e_g$ Max-hop $k$,
  Max-distance $s$
  Initialize $sg \leftarrow \emptyset$, $sList \leftarrow [x]$, $v_{sg}, e_{sg} \leftarrow \emptyset$
  $//v_{sg}$ and $e_{sg}$ represent the inferred and evidence nodes in subgraph
  while $sList \neq \emptyset$ do
    $n \leftarrow sList.dequeue(0)$
    if $distance(x, n) \leqslant k$ then
      $neighbors(n) \leftarrow neighbors \in g - sg$
      if $n \in v_{sg}$ then
        $sg.append(n), v_{sg}.append(n)$
      else if $n \in e_{sg}$ then
        $sg.append(n), e_{sg}.append(n)$
      else
        $sList.append(neighbors(n))$
      end if
    end if
  end while
  for $i \in v_{sg}$ do
    for $j \in v_{sg}$ do
      if $i \neq janddistance(i, j) \leqslant s$ then
        $edge(i, j) \leftarrow$ shortest path between but not include $i$ and $j$
        if $edge(i, j) \cap e_g = \emptyset$ then
          $sg.append(edge(i, j))$
        end if
      end if
    end for
  end for
  for $i \in v_{sg}$ do
    add approximation factors $f_i$ to sg with weight $w_i$
  end for
  solve equation group for $w_i$ using method in section 6.2.1
  compute $p(x)$ on $sg$
  return $p(x)$
```

to compute the $p(x)$. The MCMC methods, such as Gibbs sampling used in Deepdive, scales well on large graph. However, the size of subgraph extracted in our online algorithm is very small, which can be computed by exact inference algorithm to achieve better accuracy in real-time.

7 System Demostration and Experimental Study



recommend a mobile phone with high cost performance and priced between 1000RMB and 3000RMB

Copyright © 2016.@ Data and Knowledge Management Lab 316 _ NWPU.

Home Mobile Phone

OPPO R7s (3GB RAM/ CNC)

Brand: OPPO
Screen Size: 5.5 inches
Item Weight: 156g
RAM capacity : 3GB
CPU core number: 8
Battery Capacity: 3070mAh
Price: ¥2200

COMPARISON

MI M4 (RAM/ 3GB mobile 4G)

Brand: MI
Screen Size: 5 inches
Item Weight: 149g
RAM capacity : 3GB
CPU core number: 4
Battery Capacity: 3080mAh
Price: ¥1799

LETV Le 1S (X500)

Brand: LETV
Screen Size: 5.5 inches
Item Weight: 169g
RAM capacity : 3GB
CPU core number: 8
Battery Capacity: 3000mAh
Price: ¥1099

ASUS ZenFone 2

Brand: ASUS
Screen Size: 5.5 inches
Item Weight: 170g
RAM capacity : 4GB
CPU core number: 4
Battery Capacity: 3000mAh
Price: ¥1699

OPPO R7s (3GB RAM/ CNC)

Brand: OPPO
Screen Size: 5.5 inches
Weight: 156g
RAM capacity : 3GB
CPU core number: 8
Battery Capacity: 3070mAh
Market time: 2015/10
Phone size : 151.8x75.4x6.9mm
Front camera : 8 million pixels
Rear camera : 13 million pixels
¥2200
★★★★☆

RELATED PRODUCTS

vivo X6S（CNC）
¥1998
★★★★☆

OPPO A59(CNC)
¥1799
★★★★☆

MEIZU MX6(CNC)
¥1999
★★★★☆

Comments Review

advantage:
Screen display is good, beautiful design, good sense of hand. Good sound quality, putting the effect very good. Battery life is strong. Moderate use can guarantee a day, the charging process is about 90 minutes. Use nearly a month, no hot performance.

Disadvantages:
Photo shows the effect in general, but Apple 6p camera is better. Some games may not run smoothly.

to sum up:
A good phone! I like it

Figure 7.1: Poolside query screenshot

we will demonstrate the result of Poolside system by a web GUI interface for customer's query, as shown in Figure 3. The demonstration plan consists of three parts: query interface, recommendation listing and product detail presentation. The demo will use the KB system we have built for the mobile phone products. We choose mobile phone as the product shows in our demonstration because many people are familiar with it such that the results are easy to understand. The query interface accepts a user query. The interface of recommendation listing lists the products satisfying a user query and orders them by user-specified attributes/concepts. Finally, clicking a product hyperlink on the page of recommendation listing would open 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[1].

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

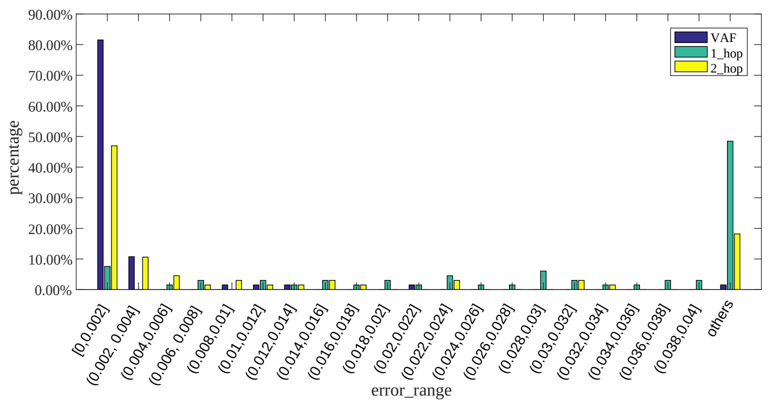


Figure 7.2: Query Accuracy Comparison

Table 7.1: Comparion On Average Time And Accuracy

	VAF	1-hop	2-hop
mean error	0.0057	0.0710	0.0241
mean time	1.0482	0.3293	27.988

We experiment our online inference algorithm by using the mobile phone knowledge base in poolside, which contains 2429 variable nodes. We compare K-hop algorithm with $k = 1$ and $k = 2$. Our online inference algorithm (Virtual Approximation Factor, VAF) chooses Max-hop $k = 2$, Max-distance $s = 1$. Figure 7.2 illustrates the comparison results on accuracy with a histogram of statistical distribution. In this figure, horizontal axis represents errors range while vertical axis is the percentage of queries. This figure shows that our dominates 1-hop and 2-hop algorithm. Our algorithm is more accurate in errors less than 0.002. and both 1-hop and 2-hop have more bad queries (errors more than 0.04) than our algorithm. This result also shows in table 7.1, the our algorithm is better than 1-hop and 2-hop in average accuracy.

Table 7.1 also compares the query time to k-hop algorithm. The 1-hop algorithm computes the result very fast with great errors introduced. 2-hop algorithm need a lot of time to computed the result and its average accuracy are less than online inference algorithm. Our solution simultaneously achieves good-quality results and real-time response.

8 Some Thoughts on Future Work

We have built a demo system POOLSIDE to enable real-time on-target decision support for online shoppers. Our demo points out two interesting directions for future research on probabilistic knowledge base. Firstly, the existing mainstream knowledge bases were primarily designed to support reasoning on text values; However, the knowledges based on numerical value comparison can be amply found in real applications. For instance, if a phone has a bigger memory size than another one, it can be probably considered to have higher performance. The inference engine based on factor graph is usually clumsy in handling the knowledges involving numerical value comparison. Secondly, the inference engines of existing knowledge bases were optimized for offline analytical tasks, but not online tasks requiring real-time response. Large-scale deployment of online analytics applications requires KB to better support real-time response.

9 Related works

Shopping decision support systems are widely learned by difference fields. The major applications are comparison shopping-agent (CSA) and recommender system. There are hundreds of such systems have been built, [1] gives a survey on current research and systems. However, most systems focus on price comparison and neglected the knowledge in products. [6] and [7] proposed the idea of using knowledge base in CSA research, but they just use knowledge base to help matching queries with entities in their system. State-of-art probabilistic knowledge bases are mostly based on markov logic networks (MLN). Tuffy [8] is the first system that pushes MLN into RDBMS, Tuffy implements MLN inference scale to large data-sets in real-world by taking advantages of RDBMS to process grounding. Deepdive is a knowledge base construction system which inherited the idea of Tuffy. Deepdive [2] combines statistical inference and machine learning to provide a powerful construction tool, and also proposed an incremental grounding method. ProKB [9] is a probabilistic knowledge base system which allows uncertain first-order relations and dramatically reduced the grounding time-cost in Tuffy. ArchimedesOne [5] is built for fast querying over the probabilistic knowledge base, it uses approximate inference on a subgraph to response the query in short times.

References

[1] B. Pathak, "A survey of the comparison shopping agent-based decision support systems," *Journal of Electronic Commerce Research*, vol. 11, no. 3, pp. 178–192, 2010.

[2] C. D. Sa, A. Ratner, C. RÍę, J. Shin, F. Wang, S. Wu, and C. Zhang, "Incremental knowledge base construction using deepdive," *Vldb Journal*, pp. 1–25, 2016.

[3] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," *Computer Science*, 2013.

[4] M. Richardson and P. Domingos, "Markov logic networks," *Machine Learning*, vol. 62, no. 1-2, pp. 107–136, 2006.

[5] X. Zhou, Y. Chen, and D. Z. Wang, "Archimedesone: Query processing over probabilistic knowledge bases," *Proceedings of the VLDB Endowment*, vol. 9, no. 13, 2016.

[6] S. L. Huang and Y. H. Tsai, "Designing a cross-language comparison-shopping agent." *Decision Support Systems*, vol. 50, no. 2, pp. 428–438, 2011.

[7] M. Eddahibi, A. Nejeoui, and C. Cherkaoui, "Towards an intelligent and deeply automated shopping bot," *International Journal of Computer Applications*, vol. 45, no. 24, pp. 21–27, 2012.

[8] F. Niu, C. , A. H. Doan, and J. Shavlik, “Tuffy: scaling up statistical inference in markov logic networks using an rdbms,” *Proceedings of the Vldb Endowment*, vol. 4, no. 6, pp. 373–384, 2011.

[9] Y. Chen and D. Z. Wang, “Knowledge expansion over probabilistic knowledge bases,” in *Proceedings of the 2014 ACM SIGMOD International Conference on Management of Data*, ser. SIGMOD ’14, 2014, pp. 649–660.