UNIVERSITY
OF TRENTO

DEPARTMENT OF INFORMATION AND COMMUNICATION TECHNOLOGY

38050 Povo – Trento (Italy), Via Sommarive 14
http://www.dit.unitn.it

IWTRUST:
IMPROVING USER TRUST IN ANSWERS FROM THE WEB

Ilya Zaihrayeu, Paulo Pinheiro da Silva
and Deborah L. McGuinness

December 2004

Technical Report # DIT-04-086

Note: in the proceedings of the itrust conference
(http://www.rocq.inria.fr/arles/events/iTrust2005/).

IWTrust: Improving User Trust in Answers from the Web

Ilya Zaihrayeu[1†] Paulo Pinheiro da Silva[2] Deborah L. McGuinness[2]

[1]ITC-IRST, Trento, Italy
[2]Stanford University, Stanford, USA

Abstract. Question answering systems users may find answers without any supporting information insufficient for determining trust levels. Once those question answering systems begin to rely on source information that varies greatly in quality and depth, such as is typical in web settings, users may trust answers even less. We address this problem by augmenting answers with optional information about the sources that were used in the answer generation process. In addition, we introduce a trust infrastructure, IWTrust, which enables computations of trust values for answers from the Web. Users of IWTrust have access to sources used in answer computation along with trust values for those source, thus they are better able to judge answer trustworthiness.

1 Introduction

The number of information sources available to web applications is growing. As information source breadth increases, so does the diversity in quality. Users may find growing challenges in evaluating the quality of web answers, particularly in settings where answers are provided without any kind of justification. One way our work improves a user's ability to judge answers is by including *knowledge provenance information* along with answers. Knowledge provenance includes information about the origin of knowledge and a description of the reasoning processes used to produce the answers [8]. This paper describes our new work, which supports justifications that may include *trust values* with answers. The computation process takes into account the user's (stated or inferred) degree of belief in the sources, question answering engines, and in other users who provide sources and/or answering engines. Our framework allows users to define and (locally) maintain individual trust values, and use those values in their evaluation of answers from their own trust "viewpoint".

In recent work, we have addressed the problem of improving user's trust in answers by providing information about how an answer was calculated [5]. That work provides an infrastructure, called Inference Web (IW), which allows proofs and proof fragments to be stored in a portable, distributed way on the

[†] A part of the work had been done when the author was affiliated with the University of Trento, Italy

web by using the proof Interlingua called the Proof Markup Language (PML) [7]. Proofs describe information manipulation processes (i.e., information extraction, reasoning, etc.) used to answer questions providing full support for tracking knowledge provenance related to answers. In addition to PML, which serves as a proof interlingua, IW provides IWBase [4], which is a distributed repository of *knowledge provenance elements*. Knowledge provenance elements contain proof-related meta-information such as information about *question answering engine*s, *inference rule*s, *representation language*s, and *source*s such as ontologies or documents. PML documents can represent answers varying from a single database lookup operation to a derivation of complex answers in a distributed fashion, involving multiple sources, and using distinct answering engines.

Knowledge provenance alone may not provide enough information for users to determine trust levels of answers. For example, a user may be unfamiliar with a question answering component that was used to find (a part of) the answer. A user may not know much about the question answering system (e.g. reasoning method, correctness and completeness of the reasoner, reasoning assumptions, etc.). Alternatively, a user may know something about the reasoner and would normally expect to trust answers from the reasoner, however when answers appear to be incomplete or conflict with a user's expectations, a user may need more information before trusting the answer. Also, users may trust a question answering system completely however if the system relies on information from sources that the user does not trust, then the user may not trust the answer. Additional considerations include situations where a source is used that is unknown to the user but the source is known to be trusted by another user (human or agent) who is trusted by the user.

In this paper we introduce an extension of IW, called *IWTrust*, which can quantify users' degree of trust in answers obtained from web applications and services. IWTrust uses trust values between users as well as trust values between users and provenance elements. Trust values for answers are computed relative to a particular user's perspective. The final trust value for the answer uses both a user's trust value of the sources used in the answer as well as a user's trust in other user's trust of sources used to obtain the answer.

The paper is organized as follows. Section 2 provides an abstract view of how trust components interact with question answering components. Section 3 provides the details of our trust model for IW. Section 4 provides an example use of IWTrust and the final section summarizes our work.

2 Trust for Question Answering

In the Inference Web context, a question *answering engine* is any kind of software system providing services able to produce answers in response to queries. From this general definition, answering engines may vary from retrieval-based services, such as database management systems and search engines, to reasoning-based services such as automated theorem provers.

We assume an environment where a user interacts with a *query front-end* component to formulate and ask a query q. The query front-end responds with a set of answers A. The query front-end is also responsible for forwarding the query q to the answering engine, grouping answers from the answering engine into a set of answers A, and forwarding A to the user. Optionally, the user may provide a set S of information sources to be used by the answering engine when retrieving information. On demand, answering engines may also provide $N(A)$, a set of justifications for answers in A.

Query languages for q vary accordingly to the kinds of answering engines used in the environment. For example, q may be a SQL query if the answering engine is a relational database management system or q may be an OWL-QL [2] query if the answering engine is a hybrid automated reasoner such as JTP [3]. The set of justifications $N(A)$ is represented in PML [7].

In general, *trust components* are responsible for computing trust values for resources such as users. In a question answering environment, we believe *trust components* should be also able to compute trust values for trust graph resources such as sources, told information, and derived information.

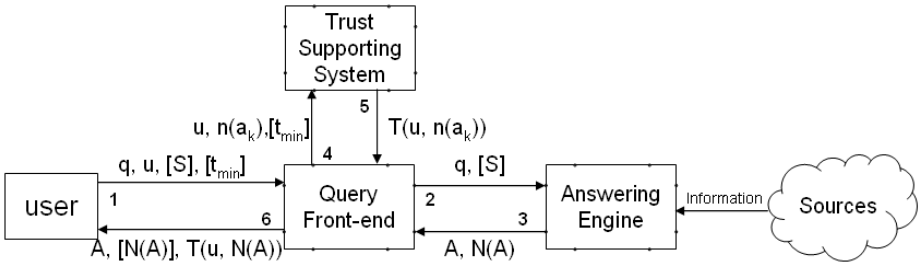

Fig. 1. Trust component in question answering systems

The computation of trust values for answers presented in Figure 1 provides a scenario where the trust component tries to assign trust values to every (intermediate) conclusion c_m during the question answering process. Trust is user-specific information; and, different users may trust other users and sources differently. Trust values for answers are computed on demand in the same way that justifications are computed on demand. Thus, if positive answer trust values are required, the answering engine should return answer justifications (step 3 in Figure 1), even if justifications are not asked by the user. From an answer and its justifications, the trust component computes (using the underlying trust network) user's trust values $T(u_i, n(a_k))$ for answer a_k , which are returned to the query front-end (step 5). The query front-end consolidates available trust values for justifications of answers in A into a single set $T(u_i, N(A))$[1]. Finally, the query front-end returns $T(u_i, N(A))$ to the user (step 6).

[1] $T(u_i, N(A))$ is defined as $\{X \mid \forall a_k \in A, T(u_i, n(a_k)) \in X\}$.

3 IWTrust Framework

In this section we introduce the IWTrust framework, referred to in the rest of the paper as IWTrust. Figure 2 shows IWTrust in action where a user u_1, submits a query q and obtains a set of answers $\{a_1, \ldots, a_n\}$ with their associated trust values $\{t_{11}, t_{12}, \ldots, t_{1m}, t_{21}, \ldots t_{n1}, \ldots\}$. The user is connected by trust relations to provenance elements (sources and answering engines in the diagram) that are used in answering the query. The user, u_1, is directly connected to some of them such as e_2. The user is connected to other provenance elements through other users (e.g., u_1 is connected to s_2, s_3 through u_3 and u_4). Provenance elements are connected to told assertions in proofs by provenance relations. Finally, Figure 2 identifies proof fragments, queries, IWBase and TrustNet as IW components supporting the trust graph as discussed in this section.

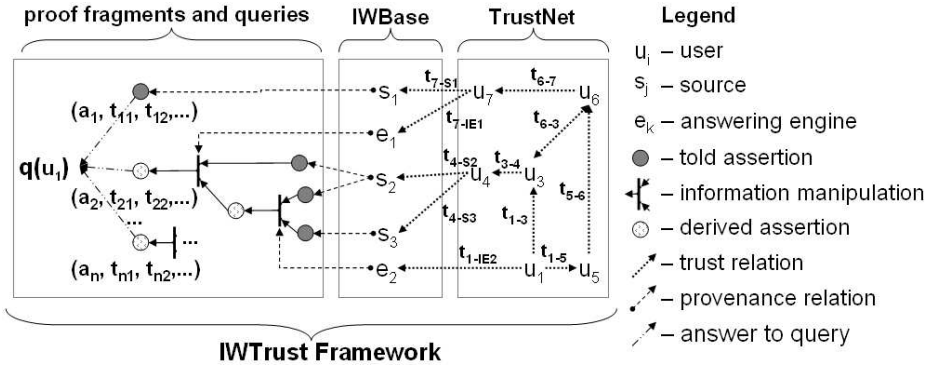

Fig. 2. IWTrust Framework

3.1 Proof Fragments and Queries

PML is used to build OWL documents representing proof fragments and queries. A PML *NodeSet* is the primary building block of proofs and is used to represent information manipulation from queries to answers and from answers to sources. A PML *Query* is used to represent user queries; it identifies the node sets containing conclusions used in answering the query. PML is an ontology written in W3C's OWL Semantic Web representation language [6] allowing justifications to be exchanged between Semantic Web services and clients using XML/RDF/OWL.

A node set $n(c)$ represents a step in a proof whose conclusion c is justified by a set of inference steps associated with the node set. The conclusion c represents the expression concluded by the proof step. Every node set has one conclusion which is the element in the trust network that requires a trust value. Each inference step of a node set represents an application of an inference rule that justifies the node set's conclusion. The antecedents of an inference step form a sequence of node sets each of whose conclusion is a *premise* of the application of the inference step's rule. Each source of an inference step refers to an entity representing original statements from which the conclusion was obtained. An

inference step's source supports the justification of the node set conclusion when the step's rule is a *DirectAssertion*.

3.2 TrustNet

TrustNet is a trust network, where users may define trust values w.r.t. other users, answering engines, and sources. In addition to these trust relations, the TrustNet trust graph represents provenance relations between sources. An edge in the graph may connect a source node to a created source node, provided that the source is the author (publisher or submitter) of the created source. Using the information in these edges, we can compute trust for a created source based on the trust of the source that created it. All edges in the graph are associated with two values: *length* and *trust value*. Intuitively, the length of an edge represents the trust "distance" between the origin (i.e., users or sources) and destination nodes.

Trust values are defined in the range [0,1], and are given a probabilistic interpretation. A trust value means the probability that: (1) a source contains relevant and correct information; (2) an answering engine correctly applies rules to derive statements (as conclusions of node sets); (3) a user provides a reference to a source that meets the requirements from (1), and/or to an answering engine that meets the requirements from (2); (4) a user recommends other user(s) who can provide trustworthy references to source(s) and/or to answering engine(s).

Edges connecting sources have length values equal to 0 and trust values equal to 1. These values represent the connection between created sources and their associated sources. All other edges have length 1 and may have an arbitrary trust value, which is computed as follows: each statement, used in query answering, and originated from some source, may be evaluated by the user either as correct or as incorrect. Users aggregate this information, and define their trust value of a source as the ratio of correct statements w.r.t. all evaluated statements from the source. The general formula for computing trust values follows: $t = \frac{t_p * n_p + n_t}{n_p + n_t + n_u}$, where n_t is the number of interactions evaluated as trustworthy; n_u is the number of interactions evaluated as untrustworthy; t_p is the level of prejudice; and n_p is a hypothetical number of interactions. t_p predetermines the starting trust level; and n_p defines the level of confidence of the user that t_p is correct – the higher n_p, the slower t changes its value, while "recording" actual interactions of the user, from the value of t_p. This approach is not absolutely new, and used in a similar form, for instance in [9].

4 Trusting Answers: An Example

IWTrust's typical use of trust values is for comparison and ordering of answers. We do not expect that typical users will be interested in looking at raw trust values such as 0.234 but we do expect that they will be interested in knowing that some answers were trusted more than others. In this section, we present an

example showing how trust values can be used to rank both answers and justifications, as well as briefly discussing the algorithms used for trust computation[2].

Figure 3 shows a proof tree supporting the answer to a question concerning the type of Tony's specialty (`TonysSpeciality` in Figure 3). This particular proof tree encodes information justifying that Tony's specialty is a shellfish dish. In this example, Source 2 states that Tony's specialty is a CRAB dish, and Source 1 states that the type is transitive (thus if a dish is of type crab and crab is a kind of shellfish, then the dish is of type shellfish). Thus, using generalized modus ponens (GMP), the proof concludes that Tony's specialty has the type of all of the superclasses of CRAB. Further, Sources 2 and 3 state that SHELLFISH is a superclass of CRAB. Thus, using GMP again, the proof concludes that Tony's specialty is a SHELLFISH dish.

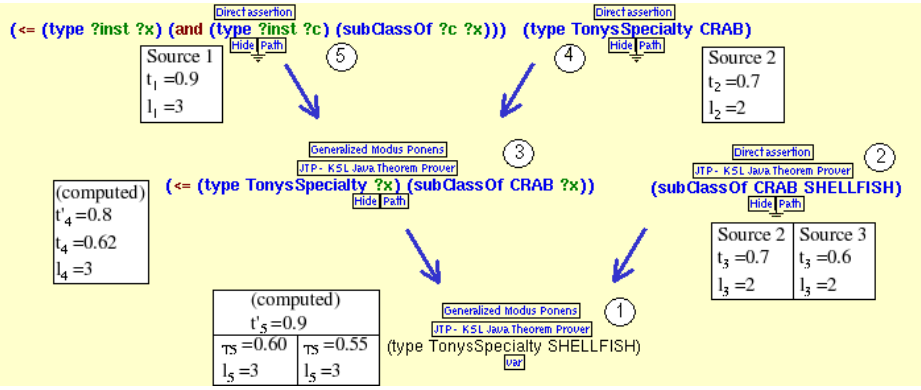

Fig. 3. Trust Composition Algorithm: an example

The proof shows that the question has at least two answers: one that Tony's specialty is a SHELLFISH dish, and also that it is a CRAB dish. The proof also shows that the SHELLFISH answer has two justifications: one based on statements from Sources 1, 2 and 3; and the other based on statements from Sources 1 and 2 only.

Trust rankings may be valuable for ranking of justifications and their components. For instance, if the user asks for a list of sources for the SHELLFISH answer, we may not include Source 3 since the justification based on Source 3 has a trust value (t=0.55) lower than the justification without it (t=0.60). Source 4, also stating that SHELLFISH is a superclass of CRAB, could be listed if its trust value was high enough to make a justification (based on its statement) with trust value higher than 0.60. The trust composition algorithm (see Algorithm 1) is used to compute these values. Thus, starting from the last step of the proof, i.e., node 1 in Figure 3, and proceeding through the set of antecedents of

[2] See the full version of the paper for comprehensive details of the algorithms. Link: http://www.ksl.stanford.edu/people/pp/papers/Zaihrayeu_iTrust_2005.pdf

node 1, i.e., nodes 2 and 3, the algorithm computes t_c and l_c recursively (lines 6, 12 and 13 in Algorithm 1). The algorithm terminates when conclusions inside a proof have no antecedents ($C = \emptyset$), i.e., when it reaches nodes 2, 4 and 5 in our example. In this case, we assume an implicit trust relation with value 1.0 and length 1 between the conclusion and its source s; whereas t_c and l_c for the resulting tuple are inferred as the trust and length values for s (line 3). We adopt Agrawal's generalized version of the *semi-naive* algorithm to compute trust values between users and from users to sources [1]. In our case, the algorithm computes the transitive closure of a (trust) relation in an iterative manner, producing at each iteration a set of new transitively held relations.

Algorithm 1 Trust Composition

input: u_i, R_a, j; **output**: $< u_i, c, t_c, l_c >$
note: c is the conclusion of proof j; s is the source of c, if any; e and C are the engine and set of antecedents for the last step in j deriving c

1: **if** $C = \emptyset$ **then**
2: $t_c, l_c \leftarrow R_a(u_i, s)$;
3: **else**
4: $watv, wl, length \leftarrow 0$;
5: **for all** $c_j \in C$ **do**
6: $t_j, l_j \leftarrow trustCompositionAlgorithm(u_i, c_j, R_a, t_{min})$;
7: $watv \leftarrow watv + (t_j / l_j)$;
8: $wl \leftarrow wl + (1/l_j)$;
9: $length \leftarrow total + l_j$;
10: **end for**
11: $t_e, l_e \leftarrow R_a(u_i, e)$
12: $t_c \leftarrow (watv/wl) * t_e$;
13: $l_c \leftarrow INT(length/|C|) + 1$
14: **end if**

For a conclusion c from a proof step with one or more antecedents, the algorithm works as follows: it first computes a weighted average over t_j for all $c_j \in C$, whereas the weights are inversely proportional to the path lengths of nodes in C. $watv$ and wl are used to compute the answer weighted average trust value. By computing a trust value for the answering engine input as the ratio between $watv$ and wl we are fully trusting the answering engine. However, we may have reasons for not trusting an engine that much. For instance, the engine may not be sound for some kind of questions. Thus, we "weigh" the input trust values against path lengths as shorter paths are likely to be more "credible" than longer ones. Then, we compute the trust value for t_c, l_c, by multiplying the weighted average trust value ($watv/wt$) by t_e (line 12). We compute the path length to c as the integer part of the average of all c_j, incremented by 1 (line 13).

5 Conclusions

In this paper, we have introduced IWTrust as a solution supporting trust in question answering environments. IWTrust leverages the Inference Web infrastructure to provide explanations from many question answering environments ranging from retrieval-intensive systems, such as database management systems and search engines, to reasoning-intensive systems such as theorem provers. It then enhances these explanations with user-customized trust values, which can then be used to determine trust of answers, sources, and answer justifications.

We present two primary contributions. First we provide an implemented solution infrastructure that can generate explanations for a wide range of question answering systems that has been integrated with a trust network. Second, we provide a design and prototype of a trust network with trust functionalities supporting trust computation in a distributed question answering environment such as the web. While others have provided trust algebras previously, and implemented explanation solutions exist for different types of question answering paradigms, our work is the first we know of that provides explanations with integrated trust values for a wide range of question answering systems and simultaneously provides an extensible architecture (allowing integration of other trust algorithms). Our primary contributions in the trust area include our design in the TrustNet layer presented in Section 3.2 and the answer trust computation algorithms used in Section 4.

References

1. R. Agrawal, S. Dar, and H. Jagadish. Direct transitive closure algorithms: Design and performance evaluation. *ACM Transactions on Database Systems.*, 15(3):427–458, September 1990.
2. Richard Fikes, Pat Hayes, and Ian Horrocks. DAML Query Language (DQL) Abstract Specification. Technical report, W3C, 2002.
3. Richard Fikes, Jessica Jenkins, and Gleb Frank. JTP: A System Architecture and Component Library for Hybrid Reasoning. Technical Report KSL-03-01, Knowledge Systems Laboratory, Stanford University, Stanford, CA, USA, 2003.
4. Deborah L. McGuinness and Paulo Pinheiro da Silva. Registry-based support for information integration. In S. Kambhampati and C. Knoblock, editors, *In Proceedings of IJCAI-2003's Workshop on Information Integration on the Web (IIWeb-03)*, pages 117–122, Acapulco, Mexico, August 2003.
5. Deborah L. McGuinness and Paulo Pinheiro da Silva. Explaining Answers from the Semantic Web. *Journal of Web Semantics*, 1(4):397–413, October 2004.
6. Deborah L. McGuinness and Frank van Harmelen. OWL Web Ontology Language Overview. Technical report, World Wide Web Consortium (W3C), February 10 2004. Recommendation.
7. Paulo Pinheiro da Silva, Deborah L. McGuinness, and Richard E. Fikes. A Proof Markup Language for Semantic Web Services. *Information Systems*, 2005. (to appear).
8. Paulo Pinheiro da Silva, Deborah L. McGuinness, and Rob McCool. Knowledge Provenance Infrastructure. *In Data Engineering Bulletin Vol.26 No.4*, pages 26–32, December 2003.
9. Wang Y. and Vassileva J. Trust-based community formation in peer-to-peer file sharing networks. *Proc. of IEEE/WIC/ACM International Conference on Web Intelligence (WI 2004), Beijing, China*, September 2004.