Bayesian Knowledge Corroboration with Logical Rules and User Feedback

Gjergji Kasneci, Jurgen Van Gael, Ralf Herbrich, and Thore Graepel

Microsoft Research Cambridge,
7 J J Thomson Avenue, Cambridge CB3 0FB, UK
{gjergjik,v-juvang,rherb,thoreg}@microsoft.com

Technical Report (MSR-TR-2010-45)

Abstract. Current knowledge bases suffer from either low coverage or low accuracy. The underlying hypothesis of this work is that user feedback can greatly improve the quality of automatically extracted knowledge bases. The feedback could help quantify the uncertainty associated with the stored statements and would enable mechanisms for searching, ranking and reasoning at entity-relationship level. Most importantly, a principled model for exploiting user feedback to learn the truth values of statements in the knowledge base would be a major step forward in addressing the issue of knowledge base curation.
We present a family of probabilistic graphical models that builds on user feedback and logical inference rules derived from the popular Semantic-Web formalism of RDFS [1]. Through internal inference and belief propagation, these models can learn both, the truth values of the statements in the knowledge base and the reliabilities of the users who give feedback. We demonstrate the viability of our approach in extensive experiments on real-world datasets, with feedback collected from Amazon Mechanical Turk.

Key words: Knowledge Base, RDFS, User Feedback, Reasoning, Probability, Graphical Model

1 Introduction

1.1 Motivation

Recent efforts in the area of Semantic Web have given rise to rich triple stores [6, 11, 14], which are being exploited by the research community [12, 13, 15–18]. Appropriately combined with probabilistic reasoning capabilities, they could highly influence the next wave of Web technology. In fact, Semantic-Web-style knowledge bases (KBs) about entities and relationships are already being leveraged by prominent industrial projects [7–9].

A widely used Semantic-Web formalism for knowledge representation is the Resource Description Framework Schema (RDFS) [1]. The popularity of this formalism is based on the fact that it provides an extensible, common syntax for data transfer and allows the explicit and intuitive representation of knowledge in form of entity-relationship (ER) graphs. Each edge of an ER graph can be thought of as an RDF *triple*, and each node as an RDFS *resource*. Furthermore, RDFS provides light-weight reasoning capabilities for inferring new knowledge from the one represented explicitly in the KB.

The triples contained in RDFS KBs are often subject to uncertainty, which may come from different sources:

Extraction & Integration Uncertainty Usually, the triples are the result of information extraction processes applied to different Web sources. After the extraction, integration processes are responsible for organizing and storing the triples into the KB. The mentioned processes build on uncertain techniques such as natural language processing, pattern matching, statistical learning, etc.

Information Source Uncertainty There is also uncertainty related to the Web pages from which the knowledge was extracted. Many Web pages may be unauthoritative on specific topics and contain unreliable information. For example, contrary to Michael Jackson's Wikipedia page, the Web site `michaeljacksonsightings.com` claims that Michael Jackson is still alive.

Inherent Knowledge Uncertainty Another type of uncertainty is the one that is inherent to the knowledge itself. For example, it is difficult to say when the great philosophers Plato or Pythagoras were exactly born. For Plato, Wikipedia offers two possible birth dates 428 BC and 427 BC. These dates are usually estimated by investigating the historical context, which naturally leads to uncertain information.

Leveraging user feedback to deal with the uncertainty and curation of data in knowledge bases is acknowledged as one of the major challenges by the community of probabilistic databases [32]. A principled method for quantifying the uncertainty of knowledge triples would not only build the basis for knowledge curation but would also enable many inference, search and recommendation tasks. Such tasks could aim at retrieving relations between companies, people, prices, product types, etc. For example, the query that asks how Coca Cola, Pepsi and Christina Aguilera are related might yield the result that Christina Aguilera performed in Pepsi as well as in Coca Cola commercials. Since the triples composing the results might have been extracted from blog pages, one has to make sure that they convey reliable information. In full generality, there might be many important (indirect) relations between the query entities, which could be inferred from the underlying data. Quantifying the uncertainty of such associations would help ranking the results in a useful and principled way.

Unfortunately, Semantic-Web formalisms for knowledge representation do not consider uncertainty. As a matter of fact, knowledge representation formalisms and formalisms that can deal with uncertainty are evolving as separate fields of AI. While knowledge representation formalisms (e.g., Description Logics [5], frames [3], KL-ONE [4], RDFS, OWL [2], etc.) focus on expressiveness and borrow from subsets of first-order logics, techniques for representing uncertainty focus on modeling possible world states, and usually represent these by probability distributions. We believe that these two fields belong together and that a targeted effort has to be made to evoke the desired synergy.

1.2 Related Work

Most prior work that has dealt with user feedback, has done so from the viewpoint of user preferences, expertise, or authority (e.g., [34–36]). We are mainly interested in the truth values of the statements contained in a knowledge base and in the reliability of users who give feedback. Our goal is to learn these values jointly, that is, we aim to learn from the feedback of multiple users at once.

There are two research areas of AI which provide models for dealing with reasoning over KBs: (1) logical reasoning and (2) probabilistic reasoning. Logical reasoning builds mainly on first-order logic and is best at dealing with relational data. Probabilistic reasoning emphasizes the uncertainty inherent in data.

There have been several proposals for combining techniques from these two areas. In the following, we discuss the strengths and weaknesses of the main approaches.

Probabilistic Database Model **(PDM)** The PDM [31–33] can be viewed as a generalization of the relational model which captures uncertainty with respect to the existence of database tuples (also known as tuple semantics) or to the values of database attributes (also known as attribute semantics).

In the tuple semantics, the main assumption is that the existence of a tuple is independent of the existence of other tuples. Given a database consisting of a single table, the number of possible worlds (i.e. possible databases) is 2^n, where n is the maximum number of the tuples in the table. Each possible world is associated with a probability which can be derived from the existence probabilities of the single tuples and from the independence assumption.

In the attribute semantics, the existence of tuples is certain, whereas the values of attributes are uncertain. Again, the main assumption in this semantics is that the values attributes take are independent of each other. Each attribute is associated with a discrete probability distribution over the possible values it can take. Consequently, the attribute semantics is more expressive than the tuple-level semantics, since in general tuple-level uncertainty can be converted into attribute-level uncertainty by adding one more (Boolean) attribute. Both semantics could also be used in combination, however, the number of possible worlds would be much larger, and deriving complete probabilistic representations would be very costly. So far, there exists no formal semantics for continuous attribute values [32]. Another major disadvantage of PDMs is that they build on rigid and restrictive independence assumptions which cannot easily model correlations among tuples or attributes [26].

Statistical Relational Learning **(SRL)** SRL models [28] are concerned with domains that exhibit uncertainty and relational structure. They combine a subset of relational calculus (first-order logic) with probabilistic graphical models, such as Bayesian or Markov networks to model uncertainty. These models can capture both, the tuple and the attribute semantics from the PDM and can represent correlations between relational tuples or attributes in a natural way [26].

More ambitious models in this realm are *Markov Logic Networks* [23, 24], *Multi-Entity Bayesian Networks* [29] and *Probabilistic Relational Models* [27]. Some of these models (e.g., [23, 24, 29]) aim at exploiting the whole expressive power of first-order logic. While [23, 24] represent the formalism of first-order logic by factor graph models, [27] and [29] deal with Bayesian networks applied to first-order logic. Usually, inference in such models is performed using standard techniques such as belief propagation or Gibbs sampling. In order to avoid complex computations, [22, 23, 26] propose the technique of *lifted inference*, which avoids materializing all objects in the domain by creating all possible groundings of the logical clauses. Although lifted inference can be more efficient than standard inference on these kinds of models, it is not clear whether they can be trivially lifted (see [25]). Hence, very often these models fall prey to high complexity when applied to practical cases.

More related to our approach is the work by Galland et al. [38], which presents three probabilistic fix-point algorithms for aggregating disagreeing views about knowledge fragments and learning their truth values as well as the trust in the views. However, as admitted by the authors, their algorithms cannot be used in an online fashion, while our approach builds on a Bayesian framework and is inherently flexible to online updates. Furthermore, [38] does not deal with the problem of logical inference, which is a core ingredient of our approach. In our experiments, we show that our approach outperforms all algorithms from [38] on a real-world dataset (provided by the authors of [38]).

Finally, a very recent article [39] proposes a supervised learning approach to the mentioned problem. In contrast to our approach, the solution proposed in [39] is not fully Bayesian and does not deal with logical deduction rules.

1.3 Contributions and Outline

We argue that in many practical cases the full expressiveness of first-order logic is not required. Rather, reasoning models for knowledge bases need to make a trade-off between expressiveness and simplicity. Expressiveness is needed to reflect the domain complexity and allow inference; simplicity is crucial in anticipation of the future scale of Semantic-Web-style data sources [6].

In this paper, we present a Bayesian reasoning framework for inference in triple stores through logical rules and user feedback. The main contributions of this paper are:

– A family of probabilistic graphical models that exploits user feedback to learn the truth values of statements in a KB. As users may often be inconsistent or unreliable and give inaccurate feedback across knowledge domains, our probabilistic graphical models jointly estimate the truth values of statements and the reliabilities of users.
– The proposed model uses logical inference rules based on the proven RDFS formalism to propagate beliefs about truth values from and to derived statements. Consequently, the model can be applied to any RDF triple store.
– We present the superiority of our approach in comparison to prior work on real-world datasets with user feedback from Amazon Mechanical Turk.

In Section 2, we describe an extension of the RDFS formalism, which we refer to as RDFS#. In Section 3, we introduce the mentioned family of probabilistic graphical models on top of the RDFS# formalism. Section 4 is devoted to experimental evaluation and we conclude in Section 5.

2 Knowledge Representation with RDFS

Semantic-Web formalisms for knowledge representation build on the entity-relationship (ER) graph model. ER graphs can be used to describe the knowledge from a domain of discourse in a structured way. Once the elements of discourse (i.e., entities or so-called *resources* in RDFS) are determined, an ER graph can be built. In the following, we give a general definition of ER graphs.

Definition 1 (Entity-Relationship Graph) *Let Ent and $Rel \subseteq Ent$ be finite sets of entity and relationship labels respectively. An entity-relationship* *graph over Ent and Rel is a multigraph* $G = (V, l_{Ent}, E_{Rel})$ *where* V *is a finite set of nodes,* $l_{Ent} : V \to Ent$ *is an injective vertex labeling function, and* $E_{Rel} \subseteq l_{Ent}(V) \times Rel \times l_{Ent}(V)$ *is a set of labeled edges.*

The labeled nodes of an ER graph represent entities (e.g., people, locations, products, dates, etc.). The labeled edges represent relationship instances, which we refer to as *statements* about entities (e.g., <*AlbertEinstein*, *hasWon*, *NobelPrize*>). Figure 1 depicts a sample ER subgraph from the YAGO knowledge base.

Element of discourse	ER term	RDFS term
$c \in Ent$	entity	resource
$r \in Rel$	relationship (type)	property
$f \in E_{Rel}$	relationship instance / fact	statement / RDF triple / fact

Table 1. Correspondence of ER and RDFS terminology.

One of the most prominent Semantic-Web languages for knowledge representation that builds on the concept of ER graphs is the Resource Description Framework Schema (RDFS) [1]. Table 1 shows the correspondence between ER and RDFS terminology.

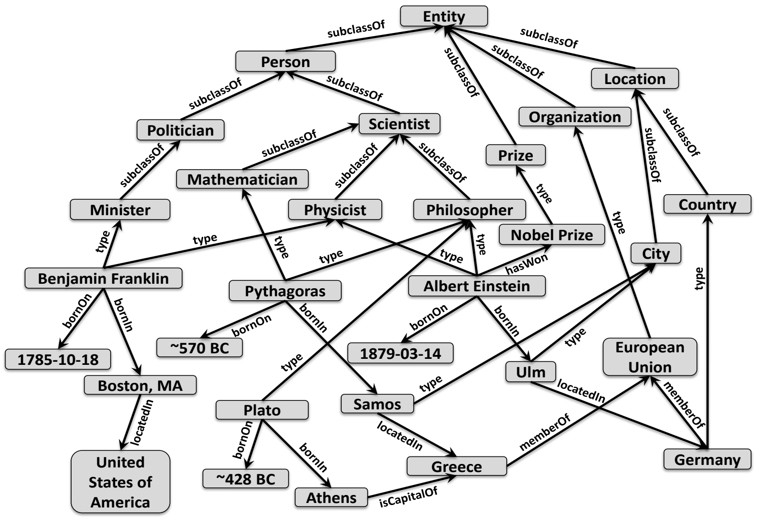


Fig. 1. Sample ER subgraph from the YAGO knowledge base

RDFS is an extensible knowledge representation language recommended by the World Wide Web Consortium (W3C) for the description of a domain of discourse (such as the Web). It enables the definition of domain resources, such as individuals (e.g. *AlbertEinstein*, *NobelPrize*, *Germany*, etc.), classes (e.g. *Physicist*, *Prize*, *Location*, etc.) and relationships (or so-called properties, e.g. *type*, *hasWon*, *locatedIn*, etc.). The basis of RDFS is RDF which comes with three basic symbols: URIs (Uniform Resource Identifiers) for uniquely addressing resources, literals for representing values such as strings, numbers, dates, etc., and blank nodes for representing unknown or unimportant resources.

Another important RDF construct for expressing that two entities stand in a binary relationship is a statement. A statement is a triple of URIs and has the form <*Subject*, *Predicate*, *Object*>, for example <*AlbertEinstein*, *bornIn*, *Ulm*>. An RDF statement can be thought of as an edge from an ER graph, where the *Subject* and the *Object* represent entity nodes and the *Predicate* represents the relationship label of the corresponding edge. Consequently, a set of RDF statements can be viewed as an ER graph. RDFS extends the set of RDF symbols by new URIs for predefined class and relation types such as *rdfs:Resource* (the class of all resources), *rdfs:subClassOf* (for representing the subclass-class relationship), etc.

RDFS is popular because it is a light-weight modeling language with practical logical reasoning capabilities, including reasoning over properties of relationships (e.g., reflexivity, transitivity, domain, and range). However, in the current specification of RDFS, reflexivity and transitivity are defined only for *rdfs:subClassOf, rdfs:subPropertyOf,* and the combination of the relationships *rdf:type+rdfs:subClassOf.* The more expressive Web Ontology Language (OWL) [2], which builds on RDFS, allows the above properties to be defined for arbitrary relationships, but its expressive power makes consistency checking undecidable. The recently introduced YAGO model [14] permits the definition of arbitrary acyclic transitive relationships but has the advantage that it still remains decidable. Being able to define transitivity for arbitrary relationships can be a very useful feature for ontological models, since many practically relevant relationships, such as *isA, locatedIn, containedIn, partOf, ancestorOf, siblingOf,* etc., are transitive. Hence, in the following, we will consider a slightly different variant of RDFS.

Let RDFS#[1] denote the RDFS model, in which blank nodes are forbidden and the reasoning capabilities are derived from the following rules. For all $X, Y, Z \in Ent, R, R' \in Rel$ with $X \neq Y, Y \neq Z, X \neq Z, R \neq R'$:

1. $<X,\ type,\ Y> \wedge <Y,\ subClassOf,\ Z> \rightarrow <X,\ type,\ Z>$
2. $<X,\ R,\ Y> \wedge <Y,\ R,\ Z> \wedge <R,\ type,\ TransitiveRelation> \rightarrow <X,\ R,\ Z>$
3. $<R,\ subPropertyOf,\ R'> \wedge <X,\ R,\ Y> \rightarrow <X,\ R',\ Y>$
4. $<R,\ hasDomain,\ Dom> \wedge <X,\ R,\ Y> \rightarrow <X,\ type,\ Dom>$
5. $<R,\ hasRange,\ Ran> \wedge <X,\ R,\ Y> \rightarrow <Y,\ type,\ Ran>$

Theorem 1 (Tractability of Inference) *For any* RDFS# *knowledge base $\mathcal{K}$, the set of all statements that can be inferred by applying the inference rules can be computed in polynomial time in the size of $\mathcal{K}$ (i.e., number of statements in $\mathcal{K}$). Furthermore, consistency can be checked in polynomial time.*

The proof of the theorem is a straight-forward extension of the proof of tractability for RDFS entailment, when blank nodes are forbidden [37].

We conclude this section by sketching an algorithm to compute the deductive closure of an RDFS# knowledge base $\mathcal{K}$ with respect to the above rules. Let $\mathcal{F}_{\mathcal{K}}$ be the set of all statements in $\mathcal{K}$. We recursively identify and index all pairs of statements that can lead to a new statement (according to the above rules) as shown in Algorithm 1.

For each pair of statements (f, f') that imply another statement $\tilde{f}$ according to the RDFS# rules, Algorithm 1 indexes $(f, f', \tilde{f})$. In case $\tilde{f}$ is not present in $\mathcal{F}_{\mathcal{K}}$ it is added and the algorithm is ran recursively on the updated set $\mathcal{F}_{\mathcal{K}}$.

[1] Read: RDFS sharp.

Algorithm 1 InferFacts($\mathcal{F}_\mathcal{K}$)

```
for all pairs $(f, f') \in \mathcal{F}_\mathcal{K} \times \mathcal{F}_\mathcal{K}$ do
  if $f \wedge f' \rightarrow \tilde{f}$ and $(f, f', \tilde{f})$ is not indexed then
    index $(f, f', \tilde{f})$
    $\mathcal{F}_\mathcal{K} = \mathcal{F}_\mathcal{K} \cup \{\tilde{f}\}$
    InferFacts($\mathcal{F}_\mathcal{K}$)
  end if
end for
```

3 A Family of Probabilistic Models

Using the language of graphical models, more specifically directed graphical models or Bayesian networks [40], we develop a family of Bayesian models each of which jointly models the truth value for each statement and the reliability for each user. The Bayesian graphical model formalism offers the following advantages:

- Models can be built from existing and tested modules and can be extended in a flexible way.
- The conditional independence assumptions reflected in the model structure enable efficient inference through message passing.
- The hierarchical Bayesian approach integrates data sparsity and traces uncertainty through the model.

We explore four different probabilistic models each incorporating a different body of domain knowledge. Assume we are given an RDFS# KB $\mathcal{K}$. Let $\mathcal{F}_\mathcal{K} = \{f_1, ..., f_n\}$ be the set of all statements contained in and deducible from $\mathcal{K}$. For each statement $f_i \in \mathcal{F}_\mathcal{K}$ we introduce a random variable $t_i \in \{T, F\}$ to denote its (unknown) truth value. We denote by $y_{ik} \in \{T, F\}$ the random variable that captures the feedback from user k for statement f_i. Let us now explore two different priors on the truth values t_i and two user feedback models connecting for y_{ik}.

3.1 Fact Prior Distributions

Independent Statements Prior. A simple baseline prior assumes independence between the truth values of statements, $t_i \sim \text{Bernoulli}(\alpha_t)$. Thus, for $\mathbf{t} \in \{T, F\}^n$, the conditional probability distribution for the independent statements prior is

$$p(\mathbf{t}|\alpha_t) = \prod_{i=1}^{n} p(t_i|\alpha_t) = \prod_{i=1}^{n} \text{Bernoulli}(t_i; \alpha_t). \tag{1}$$

This strong independence assumption discards existing knowledge about the relationships between statements from RDFS#. This problem is addressed by the *Deduced Statements Prior.*

Deduced Statements Prior. A more complex prior will incorporate the deductions from RDFS# into a probabilistic graphical model. First, we describe a general mechanism to turn a logical deduction into a probabilistic graphical model. Then, we show how this can be used in the context of RDFS#.

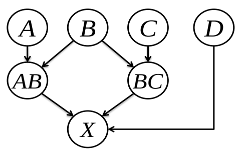


Fig. 2. A graphical model illustrating the logical derivation for the formula $X = (A \wedge B) \vee (B \wedge C) \vee D$.

Let X denote a variable that can be derived from $A \wedge B$ or $B \wedge C$, where the premises A, B, and C are known. Let D denote all unknown derivations of X. The truth of X can be expressed in disjunctive normal form: $X = (A \wedge B) \vee (B \wedge C) \vee D$. This can automatically be turned into the graphical model shown in Figure 2. For each conjuctive clause, a new variable with corresponding conditional probability distribution is introduced, e.g.,

$$p(AB|A, B) = \begin{cases} 1 & \text{if } A \wedge B \\ 0 & \text{otherwise} \end{cases} \tag{2}$$

This simplifies our disjunctive normal form to the expression $X = AB \vee BC \vee D$. Finally, we connect X with all the variables in the disjunctive normal form by a conditional probability:

$$p(X|AB, BC, D) = \begin{cases} 1 & \text{if } AB \vee BC \vee D \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

This construction can be applied to all the deductions implied by RDFS#. After computing the deductive closure of the KB (see Algorithm 1), for each statement $f_i \in \mathcal{F}_{\mathcal{K}}$, all pairs of statements that imply f_i can be found; we denote this set by $\mathcal{D}_i$. An additional binary variable $\tilde{t}_i \sim \text{Bernoulli}(\alpha_t)$ is introduced to account for the possibility that our knowledge base does not contain *all* possible deductions of statement f_i. The variable $\tilde{t}_i$ is added to the probabilistic graphical model similar to the variable D in the example above. Hence, we derive the following conditional probability distribution for the prior on statements

$$p(\mathbf{t}|\alpha_t) = \prod_{i=1}^{n} \sum_{\tilde{t}_i \in \{T,F\}} p(t_i|\tilde{t}_i, \mathcal{D}_i, \alpha_t) p(\tilde{t}_i|\alpha_t), \tag{4}$$

where Equations (2) and (3) specify the conditional distribution $p(t_i|\tilde{t}_i, \mathcal{D}_i, \alpha_t)$.

3.2 User Feedback Models

The proposed user feedback model jointly models the truth values t_i, the feedback signals y_{ik} and the user reliabilities. In this section we discuss both a one-parameter and a two-parameter per user model for the user feedback component. Note that not all users rate all statements: this means that only a subset of the y_{ik} will be observed.

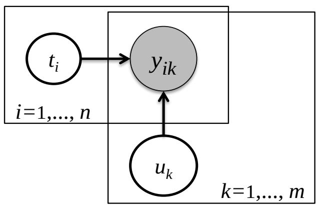
1-Parameter Model. This model represents the following user behavior. When user k evaluates a statement f_i, with probability u_k he will report the real truth value of f_i and with probability $1 - u_k$ he will report the opposite truth value.

Figure 4 represents the conditional probability table for $p(y_{ik}|u_k, t_i)$. Consider the set $\{y_{ik}\}$ of observed true/false feedback labels for the statement-user pairs. The conditional probability distribution for $\mathbf{u} \in [0,1]^m$, $\mathbf{t} \in \{T, F\}^n$ and $\{y_{ik}\}$ in the 1-parameter model is

$$p(\{y_{ik}\}, \mathbf{u}|\mathbf{t}, \alpha_u, \beta_u) = \prod_{\text{users } k} p(u_k|\alpha_u, \beta_u) \prod_{\text{statements } i \text{ by } k} p(y_{ik}|t_i, u_k). \quad (5)$$

2-Parameter Model. This model represents a similar user behavior as above, but this time we model the reliability of each user k with two parameters $u_k \in [0,1]$ and $\bar{u}_k \in [0,1]$, one for true statements and one for false statements. Figure 5 represents the conditional probability table for $p(y_{ik}|u_k, \bar{u}_k, t_i)$. The conditional probability distribution for the 2-parameter model is

$$p(\{y_{ik}\}, \mathbf{u}, \bar{\mathbf{u}}|\mathbf{t}, \alpha_u, \beta_u, \alpha_{\bar{u}}, \beta_{\bar{u}}) = \prod_{\text{users } k} p(u_k|\alpha_u, \beta_u) p(\bar{u}_k|\alpha_{\bar{u}}, \beta_{\bar{u}}) \prod_{\text{statements } i \text{ by } k} p(y_{ik}|t_i, u_k, \bar{u}_k). \quad (6)$$



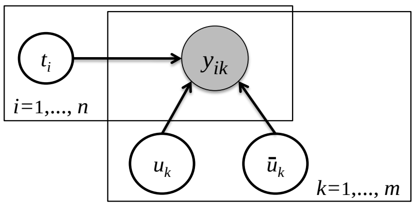


Fig. 3. The graphical models for the user feedback components. Left, the 1-parameter feedback model and right, the 2-parameter feedback model.

y_{ik} \ t_i	T	F
T	u_k	$1-u_k$
F	$1-u_k$	u_k

Fig. 4. The conditional probability distribution for feedback signal y_{ik} given reliability u_k and truth t_i.

y_{ik} \ t_i	T	F
T	u_k	$1-\bar{u}_k$
F	$1-u_k$	$\bar{u}_k$

Fig. 5. The conditional probability distribution for feedback signal y_{ik} given reliabilities $u_k, \bar{u}_k$ and truth t_i.

In both models, the prior belief about u_k (and $\bar{u}_k$ in the 2-parameter model) is modeled by a Beta(α_u, β_u) (and Beta$(\alpha_{\bar{u}}, \beta_{\bar{u}})$) distribution, which is a conjugate prior for the Bernoulli distribution.

Table 2 depicts four different models, composed using all four combinations of statement priors and user feedback models. We can write down the full joint probability distribution for the I1 model as

$$p(\mathbf{t}, \{y_{ik}\}, \mathbf{u}|\alpha_t, \alpha_u, \beta_u) = \left(\prod_i^n \text{Bernoulli}(t_i; \alpha_t)\right) \left(\prod_{\text{users } k} p(u_k|\alpha_u, \beta_u) \prod_{\text{statements } i \text{ by } k} p(y_{ik}|t_i, u_k)\right). \quad (7)$$

The joint distribution for I2, D1 and D2 can be written down similarly by combining the appropriate equations above.

Model Name	Composition
I1	independent priors & 1-parameter feedback model
I2	independent priors & 2-parameter feedback model
D1	deduced statements priors & 1-parameter feedback model
D2	deduced statements priors & 2-parameter feedback model

Table 2. The four different models

3.3 Discussion

Figure 6 illustrates how the 1-parameter feedback model, D1, can jointly learn the reliability of two users and the truth values of two statements, f_i and f_j, on which they provide feedback. Additionally, it can also learn the truth value of the statement f_l, which can be derived from $f_i \wedge f_j$. An additional variable $\tilde{t}_l$ is added to account for any deductions which might not be captured by the KB. Note that the model in Figure 6 is loopy but still satisfies the acyclicity required by a directed graphical model.

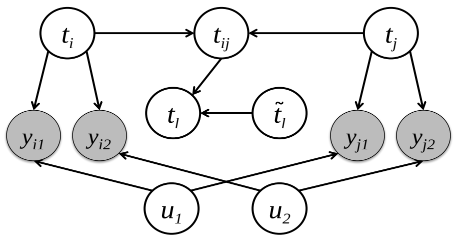


Fig. 6. Illustration of a small instance of the D1 model. Note how user feedback is propagated through the logical relations among the statements.

Given a probabilistic model we are interested in computing the posterior distribution for the statement truth variables and user reliabilities: $p(\mathbf{t}|\{y_{ik}\}, \alpha_t, \alpha_u, \beta_u)$ and $p(\mathbf{u}|\{y_{ik}\}, \alpha_t, \alpha_u, \beta_u)$. Both computations involve summing (or integrating) over all possible assignments for the unobserved variables

$$p(\mathbf{u}|\{y_{ik}\}, \alpha_t, \alpha_u, \beta_u) \propto \sum_{t_1 \in \{T,F\}} \cdots \sum_{t_n \in \{T,F\}} p(\mathbf{t}, \{y_{ik}\}, \mathbf{u}|\alpha_t, \alpha_u, \beta_u). \quad (8)$$

As illustrated in Figure 6, the resulting graphical models are loopy. Moreover deep deduction paths may lead to high treewidth graphical models making exact computation intractable. We chose to use an approximate inference scheme based on message passing known as expectation propagation [30, 21].

From a computational perspective, it is easiest to translate the graphical models into factor graphs, and describe the message passing rules over them. Table 3 summarizes how to translate each component of the above graphical models into a factor graph. We rely on Infer.NET [10] to compute a schedule for the message passing algorithms and to execute them. The message passing algorithms run until convergence. The complexity of every iteration is linear in the number of nodes in the underlying factor graph.

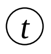
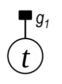
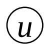
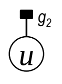
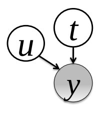
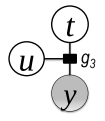
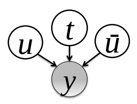
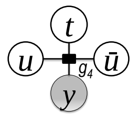
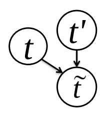
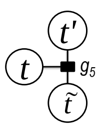
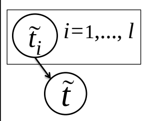
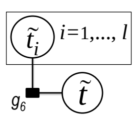
Bayes Net	Factor Graph	Factor Semantics
		$g_1(t) = \pi^{[\![t]\!]}(1-\pi)^{(1-[\![t]\!])}$
		$g_2(u) = \dfrac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha+\beta)} u^{\alpha-1}(1-u)^{\beta-1}$
		$g_3(y,t,u) = (u^{[\![y]\!]}(1-u)^{(1-[\![y]\!])})^{[\![t]\!]}$ $\cdot(u^{(1-[\![y]\!])}(1-u)^{[\![y]\!]})^{(1-[\![t]\!])}$
		$g_4(y,t,u,\bar{u}) = (u^{[\![y]\!]}(1-u)^{(1-[\![y]\!])})^{[\![t]\!]}$ $\cdot(\bar{u}^{(1-[\![y]\!])}(1-\bar{u})^{[\![y]\!]})^{(1-[\![t]\!])}$
		$g_5(t,t',\tilde{t}) = [\![t]\!][\![t']\!][\![\tilde{t}]\!] + (1-[\![t]\!])[\![t']\!](1-[\![\tilde{t}]\!])$ $+[\![t]\!](1-[\![t']\!])(1-[\![\tilde{t}]\!]) + (1-[\![t]\!])(1-[\![t']\!])(1-[\![\tilde{t}]\!])$
		$l = 2 : g_6(\tilde{t}_1,\tilde{t}_2,\tilde{t}) = [\![\tilde{t}_1]\!][\![\tilde{t}_2]\!][\![\tilde{t}]\!] + (1-[\![\tilde{t}_1]\!])[\![\tilde{t}_2]\!][\![\tilde{t}]\!]$ $+[\![\tilde{t}_1]\!](1-[\![\tilde{t}_2]\!])[\![\tilde{t}]\!] + (1-[\![\tilde{t}_1]\!])(1-[\![\tilde{t}_2]\!])(1-[\![\tilde{t}]\!])$

Table 3. Detailed semantics for the graphical models. The first column depicts the Bayesian network dependencies for a component in the graphical model, the second column illustrates the corresponding factor graph, and the third column gives the exact semantics of the factor. The function $[\![t]\!]$ maps T and F to 1 and 0, respectively.

4 Experimental Evaluation

For the empirical evaluation we constructed a dataset by choosing a subset of 833 statements about prominent scientists from the YAGO knowledge base [14]. Since the majority of statements in YAGO are correct, we extended the extracted subset by 271 false but semantically meaningful statements[2] that were randomly generated from YAGO entities and relationships, resulting in a final set of 1,104 statements. The statements from this dataset were manually labeled as true or false, resulting in a total of 803 true statements and 301 false statements.

YAGO provides transitive relationships, such as *locatedIn*, *isA*, *influences*, etc. Hence, we are in the RDFS# setting. We ran Algorithm 1 to compute the

[2] E.g., the statement <*AlbertEinstein*, *bornIn*, *Berlin*> is meaningful although false, whereas <*Berlin*, *bornIn*, *AlbertEinstein*> is not semantically meaningful.

closure of our dataset with respect to the transitive relationships. This resulted in 329 pairs of statements from which another statement in the dataset could be derived.

For the above statements we collected feedback from Amazon Mechanical Turk (AMTurk). The users were presented with tasks of at most 5 statements each and asked to label each statement in a task with either true or false. This setup resulted in 221 AMTurk tasks to cover the 1,104 statements in our dataset. Additionally, the users were offered the option to use any external Web sources when assessing a statement. 111 AMTurk users completed between 1 and 186 tasks. For each task we payed 10 US cents. At the end we collected a total number of 11,031 feedback labels.

4.1 Quality Analysis

First we analyze the quality of the four models, I1, I2, D1, D2. As a baseline method we use a "voting" scheme, which computes the probability of a statement f being true as

$$p(f) = \frac{1 + \text{\# of true votes for } f}{2 + \text{\# of votes for } f}.$$

We choose the negative log score (in bits) as our accuracy measure. For a statement f_i with posterior p_i the negative log score is defined as

$$\text{nls}(p_i, t_i) := \begin{cases} -\log_2(p_i) & \text{if ground truth for } f_i \text{ is true} \\ -\log_2(1-p_i) & \text{if ground truth for } f_i \text{ is false} \end{cases} \qquad (9)$$

The negative log score represents how much information in the ground truth is captured by the posterior; when $p_i = [\![t_i]\!]$ the log score is zero. To illustrate the learning rate of each model, in Figure 7 we show aggregate negative log scores for nested subsets of the feedback labels. For each of the subsets, we use all 1,104 statements of the dataset.

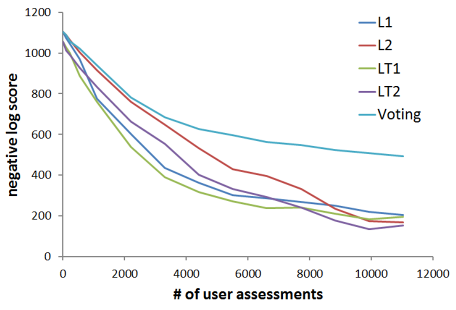


Fig. 7. The negative log score for the different models as a function of the number of user assessments.

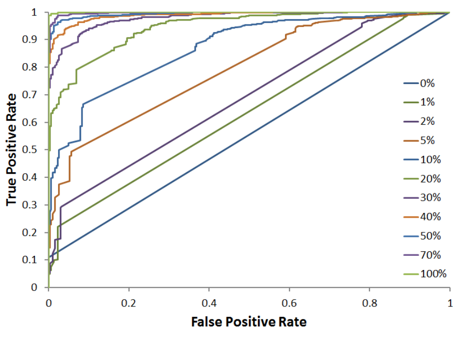


Fig. 8. The ROC curves for the D1 model, for varying numbers of user assessments.

Figure 7 shows that for smaller subsets of feedback labels the simpler models perform better and have lower negative log scores. However, as the number of labels increases, the two-parameter models become more accurate. This is in line with the intuition that simpler (i.e., one-parameter) models learn quicker (i.e., with fewer labels). Nonetheless, observe that with more labels, the more flexible (i.e., 2-parameter) models achieve lower negative log scores. Finally, the logical inference rules reduce the negative log scores by about 50 bits when there are no labels. Nevertheless, when the amount of labels grows, the logical inference rules hardly contribute to the decrease in negative log score. All models consistently outperform the voting approach.

We computed ROC curves for model D1 for different nested subsets of the data. In Figure 8, when all labels are used, the ROC curve shows almost perfect true positive and false positive behavior. The model already performs with high accuracy for 30% of the feedback labels. Also, we get a consistent increase in AUC as we increase the number of feedback signals.

4.2 Case Studies

Our probabilistic models have another big advantage: the posterior probabilities for truths and reliabilities have clear semantics. By inspecting them we can discover different types of user behavior.

When analyzing the posterior probabilities for the D2 model, we found that the reliability of one of the users was 89% when statements were true, while it was only 8% when statements were false. When we inspected the labels that were generated by the user we found that he labelled 768 statements, out of which 693 statements were labelled as “true”. This means that he labelled 90% of all statements that were presented to him as “true”, whereas in our dataset only about 72% of all statements are true. Our model suggests that it is more likely that this user was consciously labelling almost all statements as true. Similarly we found users who almost always answered “false” to the statements that were presented. In Figure 9, the scatter plot for the mean values of u and $\bar{u}$ across all users gives evidence for the existence of such a biased behavior. The points in the lower-right and in the upper-left part of the plot represent users who report statements mainly as true and false, respectively.

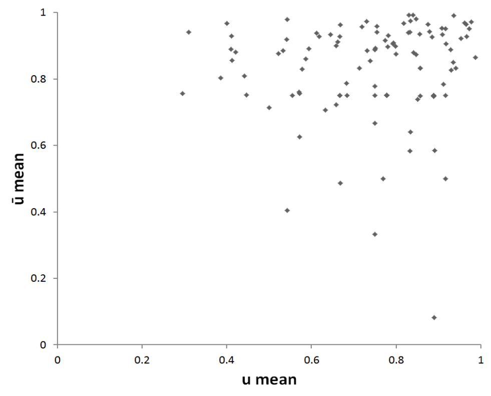


Fig. 9. Scatter plot of u versus $\bar{u}$ for the D2 model. Each dot represents a user.

Interestingly enough, we did not find any users who were consistently reporting the opposite truth values compared to their peers. We would have been able to discover this type of behavior by the D2 model. In such a case, a good indication would be reliabilities below 50%.

The previous analysis also hints at an important assumption of our model: only because most of our users are providing correct feedback, it is impossible for malicious behavior to go undetected. If enough reliable users are wrong about a statement, our model can converge on the wrong belief.

4.3 Comparison

In addition, we evaluated the D1 model on another real-world dataset that was also used by the very recent approach presented in [38]. The authors of [38] present three fixed-point algorithms for learning truth values of statements by aggregating user feedback. They report results on various datasets one of which is a sixth-grade biology test dataset. This test consists of 15 yes-no questions which can be viewed as statements in our setting. The test was taken by 86 participants who gave a total of 1,290 answers, which we interpret as feedback labels. For all algorithms presented in [38], the authors state that they perform similarly to the voting baseline. The voting baseline yields a negative log score of 8.5, whereas the D1 model yields a much better negative log score of $3.04e-5$.

5 Conclusion

We presented a Bayesian approach to the problem of knowledge corroboration with user feedback and semantic rules. The strength of our solution lies in its capability to jointly learn the truth values of statements and the reliabilities of users, based on logical rules and internal belief propagation. We are currently investigating its application to large-scale knowledge bases with hundreds of millions of statements or more. Along this path, we are looking into more complex logical rules and more advanced user and statement features to learn about the background knowledge of users and the difficulty of statements. Finally, we are exploring active learning strategies to optimally leverage user feedback in an online fashion.

In recent years, we have witnessed an increasing involvement of users in annotation, labeling, and other knowledge creation tasks. At the same time, Semantic Web technologies are giving rise to large knowledge bases that could facilitate automatic knowledge processing. The approach presented in this paper aims to transparently evoke the desired synergy from these two powerful trends, by laying the foundations for complex knowledge curation, search and recommendation tasks. We hope that this work will appeal to and further benefit from various research communities such as AI, Semantic Web, Social Web, and many more.

6 Acknowledgments

We thank the Infer.NET team, John Guiver, Tom Minka, and John Winn for their consistent support throughout this project.

References

1. W3C: RDF Vocabulary Description Language 1.0: RDF Schema. http://www.w3.org/TR/rdf-schema/
2. W3C: OWL Web Ontology Language. http://www.w3.org/TR/owl-features/
3. Minsky, M.: A Framework for Representing Knowledge. MIT-AI Laboratory Memo 306, 1974. http://web.media.mit.edu/~minsky/papers/Frames/frames.html
4. Brachman, R.J., Schmolze, J.: An Overview of the KL-ONE Knowledge Representation System. In: Cognitive Science, 9(2), 1985.
5. Baader, F., Calvanese, D., McGuinness D. L., Nardi D., Patel-Schneider, P. F.: The Description Logic Handbook. Cambridge University Press (2003)
6. W3C SweoIG: The Linking Open Data Community Project. http://esw.w3.org/topic/SweoIG/TaskForces/CommunityProjects/LinkingOpenData
7. Wolfram Alpha: A Computational Knowledge Engine. http://www.wolframalpha.com/
8. EntityCube. http://entitycube.research.microsoft.com/
9. True Knowledge. http://www.trueknowledge.com/
10. Infer.NET http://research.microsoft.com/en-us/um/cambridge/projects/infernet/
11. Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., Hellmann, S.: DBpedia: A Nucleus for a Web of Open Data. In: 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference (ISWC/ASWC 2007), pp. 722–735. Springer (2007)
12. Lehmann, J., Schüppel, J., Auer, S.: Discovering Unknown Connections - The DBpedia Relationship Finder. In: 1st Conference on Social Semantic Web (CSSW 2007) pp. 99–110. GI (2007)
13. Suchanek, F. M., Sozio, M., Weikum, G.: SOFIE: Self-Organizing Flexible Information Extraction. In: 18th International World Wide Web conference (WWW 2009), pp. 631–640. ACM Press (2009)
14. Suchanek, F. M., Kasneci, G., Weikum, G.: Yago: A Core of Semantic Knowledge. In: 16th International World Wide Web Conference (WWW 2007), pp. 697–706. ACM Press (2007)
15. Kasneci, G., Suchanek, F. M., Ifrim, G., Ramanath, M., Weikum, G.: NAGA: Searching and Ranking Knowledge. In: 24th International Conference on Data Engineering (ICDE 2008), pp. 953–962. IEEE (2008)
16. Kasneci, G., Ramanath, M., Sozio, M., Suchanek, F. M., Weikum, G.: STAR: Steiner-Tree Approximation in Relationship Graphs. In: 25th International Conference on Data Engineering (ICDE 2009), pp. 868–879. IEEE (2009)
17. Kasneci, G., Shady, E., Weikum, G.: MING: Mining Informative Entity Relationship Subgraphs. In: 18th ACM Conference on Information and Knowledge Management (CIKM 2009), pp. 1653–1656. ACM Press (2009)
18. Preda, N. Kasneci, G., Suchanek, F. M., Yuan, W., Neumann, T., Weikum, G.: Active Knowledge: Dynamically Enriching RDF Knowledge Bases by Web Services. In:30th ACM International Conference on Management Of Data (SIGMOD 2010), ACM Press (2010)
19. Wu, F., Weld, D. S.: Autonomously Semantifying Wikipedia. In: 16th ACM Conference on Information and Knowledge Management (CIKM 2007), pp. 41–50. ACM Press (2007)
20. Weld, D. S., Wu, F., Adar, E., Amershi, S., Fogarty, J., Hoffmann, R., Patel, K., Skinner, M.: Intelligence in Wikipedia. In: 23rd AAAI Conference on Artificial Intelligence (AAAI 2008), pp. 1609–1614. AAAI Press (2008)
21. Minka, T. P.: A Family of Algorithms for Approximate Bayesian Inference. Massachusetts Institute of Technology (2001)
22. Poole, D.: First-Order Probabilistic Inference. In: 8th International Joint Conference on Artificial Intelligence (IJCAI 2003), pp. 985–991, Morgan Kaufmann (2003)

23. Domingos, P., Singla, P.: Lifted First-Order Belief Propagation. In: 23rd AAAI Conference on Artificial Intelligence (AAAI 2008), pp. 1094–1099. AAAI Press (2008)
24. Domingos, P., Richardson, M.: Markov Logic Networks. In: Machine Learning, 62(1–2), pp. 107–136. Springer (2006)
25. Jaimovich, A., Meshi, O., Friedman, N.: Template Based Inference in Symmetric Relational Markov Random Fields. In: 23rd Conference on Uncertainty in Artificial Intelligence (UAI 2007), pp. 191-199. AUAI Press (2007)
26. Sen, P., Deshpande, A., Getoor, L.: PrDB: Managing and Exploiting Rich Correlations in Probabilistic Databases. In: Journal of Very Large Databases, 18(5), pp. 1065–1090. Springer (2009)
27. Friedman, N., Getoor, L., Koller, D., Pfeffer, A. Learning Probabilistic Relational Models. In: 16th International Joint Conference on Artificial Intelligence (IJCAI 1999), pp. 1300–1309. Morgan Kaufman (1999)
28. Getoor, L.: Tutorial on Statistical Relational Learning. In: 15th International Inductive Logic Programming Conference (ILP 2005), Springer (2005)
29. Da Costa, P. C. G., Ladeira, M., Carvalho, R. N., Laskey, K. B., Santos, L. L., Matsumoto, S.: A First-Order Bayesian Tool for Probabilistic Ontologies. In: 21st International Florida Artificial Intelligence Research Society Conference (FLAIRS 2008), pp. 631–636. AAAI Press (2008)
30. Frey, B. J., Mackay, D. J. C.: A Revolution: Belief Propagation in Graphs with Cycles. In: Advances in Neural Information Processing Systems 10, pp. 479–485. MIT Press (1997)
31. Antova, L., Koch, C., Olteanu, D.: 10^{10^6} Worlds and Beyond: Efficient Representation and Processing of Incomplete Information. In: 23rd International Conference on Data Engineering (ICDE 2007), pp. 606–615. IEEE (2007)
32. Dalvi, N. N., Ré, C., Suciu, D.: Probabilistic Databases: Diamonds in the Dirt. In: Communications of ACM, 52(7), (CACM 2009), pp. 86–94. ACM Press (2009)
33. Agrawal, P., Benjelloun, O., Sarma, A. D., Hayworth, C., Nabar, S. U., Sugihara, T., Widom, J.: Trio: A System for Data, Uncertainty, and Lineage. In: 32nd International Conference on Very Large Data Bases (VLDB 2006), pp. 1151–1154. ACM Press (2006)
34. Osherson, D., Vardi, M. Y.: Aggregating Disparate Estimates of Chance. In: Games and Economic Behavior, 56(1), pp. 148–173. Elsevier (2006)
35. Jøsang, A., Marsh, S., Pope, S.: Exploring Different Types of Trust Propagation. In: 4th International Conference on Trust Management (iTrust 2006), pp: 179–192. Springer (2006)
36. Kelly, D., Teevan, J.: Implicit Feedback for Inferring User Preference: A Bibliography. In: SIGIR Forum, 37(2), pp. 18–28. ACM Press (2003)
37. Horst, H. J. T.: Completeness, Decidability and Complexity of Entailment for RDF Schema and a Semantic Extension Involving the OWL Vocabulary. In: Journal of Web Semantics: Science, Services and Agents on the World Wide Web, 3(2–3), pp. 79–115, Elsevier (2005)
38. Galland, A., and Abiteboul, S., and Marian, A., and Senellart, P.: Corroborating Information from Disagreeing Views. In: 3rd ACM International Conference on Web Search and Data Mining (WSDM 2010), pp. 1041–1064, ACM Press (2010)
39. Raykar, V. C., and Yu, S., and Zhao, L. H., and Valadez, G. H., and Florin, C., and Bogoni, L., Moy, L.: Learning From Crowds. In: Journal of Machine Learning Research, 11, pp. 1297–1322, MIT Press (2010)
40. Pearl, J.: Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference. Morgan Kaufmann (1997)

Appendix: Message Passing Operations

As we discussed in Section 3.3, we use expectation propagation for posterior inference. We implement expectation propagation in a message passing framework. For completeness, we describe the exact message passing operations for the various components in our graph.

Message from Independent Fact Prior. The Bernoulli prior on a truth value has form $g_1(t) = \pi^{[\![t]\!]}(1-\pi)^{(1-[\![t]\!])}$. The only message that needs to be sent is a message from factor g_1 to t. This message has the form $m_{g_1 \to t}(t) = \text{Bernoulli}(\pi)$.

Message for the User Reliabilities. The Beta prior on a user's reliability u has form $g_2(u) = \frac{\Gamma(\alpha)\Gamma(\beta)}{\Gamma(\alpha_u+\beta_u)} u^{\alpha_u - 1}(1-u)^{\beta_u - 1}$. The only message that needs to be sent is a message from factor g_2 to u. This message has the form $m_{g_2 \to u}(u) = \text{Beta}(\alpha_u, \beta_u)$. Analogously, one can derive the message from g_2 to $\bar{u}$.

Message for the AND Factor The probabilistic model with the deduced facts prior introduces conjunctions of random boolean variables. The expression that corresponds to the factor $\tilde{t} = t \wedge t'$ is

$$g_5(t, t', \tilde{t}) = [\![t]\!][\![t']\!][\![\tilde{t}]\!] + (1-[\![t]\!])[\![t']\!](1-[\![\tilde{t}]\!]) + [\![t]\!](1-[\![t']\!])(1-[\![\tilde{t}]\!]) + (1-[\![t]\!])(1-[\![t']\!])(1-[\![\tilde{t}]\!]).$$

Let us denote the incomming messages as $m_{t \to g_5}(t) = \text{Bernoulli}(\mu_t)$, $m_{t' \to g_5}(t') = \text{Bernoulli}(\mu_{t'})$ and $m_{\tilde{t} \to g_5}(\tilde{t}) = \text{Bernoulli}(\mu_{\tilde{t}})$. The outgoing messages have the form

$$m_{g_5 \to t}(t) = \text{Bernoulli}(\frac{\mu_{t'}\mu_{\tilde{t}} + (1-\mu_{t'})(1-\mu_{\tilde{t}})}{\mu_{t'}\mu_{\tilde{t}} + (2-\mu_{t'})(1-\mu_{\tilde{t}})}),$$

and

$$m_{g_5 \to t}(t) = \text{Bernoulli}(\frac{\mu_{t}\mu_{\tilde{t}} + (1-\mu_{t})(1-\mu_{\tilde{t}})}{\mu_{t}\mu_{\tilde{t}} + (2-\mu_{t})(1-\mu_{\tilde{t}})}),$$

and

$$m_{g_5 \to \tilde{t}}(\tilde{t}) = \text{Bernoulli}(\mu_t \mu_{t'}).$$

Message for the OR Factor The probabilistic model with the deduced facts prior introduces disjunctions of random boolean variables. The expression that corresponds to the factor $\tilde{t} = \tilde{t}_1 \vee \tilde{t}_2$ is

$$g_6(\tilde{t}_1, \tilde{t}_2, \tilde{t}) = [\![\tilde{t}_1]\!][\![\tilde{t}_2]\!][\![\tilde{t}]\!] + (1-[\![\tilde{t}_1]\!])[\![\tilde{t}_2]\!][\![\tilde{t}]\!] + [\![\tilde{t}_1]\!](1-[\![\tilde{t}_2]\!])[\![\tilde{t}]\!] + (1-[\![\tilde{t}_1]\!])(1-[\![\tilde{t}_2]\!])(1-[\![\tilde{t}]\!]).$$

Let us denote the incomming messages as $m_{\tilde{t}_1 \to g_6}(\tilde{t}_1) = \text{Bernoulli}(\mu_{\tilde{t}_1})$, $m_{\tilde{t}_2 \to g_6}(\tilde{t}_2) = \text{Bernoulli}(\mu_{\tilde{t}_2})$ and $m_{\tilde{t} \to g_6}(\tilde{t}) = \text{Bernoulli}(\mu_{\tilde{t}})$. The outgoing messages have the form

$$m_{g_6 \to \tilde{t}_1}(\tilde{t}_1) = \text{Bernoulli}(\frac{\mu_{\tilde{t}}}{2\mu_{\tilde{t}} + (1-2\mu_{\tilde{t}})(1-\mu_{\tilde{t}_2})}),$$

and

$$m_{g_6 \to \tilde{t_2}}(\tilde{t_2}) = \text{Bernoulli}(\frac{\mu_{\tilde{t}}}{2\mu_{\tilde{t}} + (1-2\mu_{\tilde{t}})(1-\mu_{\tilde{t_1}})}),$$

and

$$m_{g_6 \to \tilde{t}}(\tilde{t}) = \text{Bernoulli}(1-(1-\mu_{\tilde{t_1}})(1-\mu_{\tilde{t_2}})).$$

6.1 Message for the 1-Parameter Feedback Model

The factor for the 1-parameter model can be written as

$$g_3(y,t,u) = (u^{[\![y]\!]}(1-u)^{(1-[\![y]\!])})^{[\![t]\!]}(u^{(1-[\![y]\!])}(1-u)^{[\![y]\!]})^{(1-[\![t]\!])}.$$

Let us denote with $m_{t \to g_3}(t) = \text{Bernoulli}(\mu_t)$ and $m_{u \to g_3}(u) = \text{Beta}(\alpha, \beta)$ the incomming messages for factor g_3. The outgoing messages are

$$m_{g_3 \to t}(t) = \begin{cases} \text{Bernoulli}(\beta/(\alpha+\beta)) & \text{if } y = 0 \\ \text{Bernoulli}(\alpha/(\alpha+\beta)) & \text{if } y = 1 \end{cases} \tag{10}$$

$$m_{g_3 \to u}(u) = \begin{cases} \text{proj}\left[\mu_t \frac{\beta}{\alpha+\beta}\text{Beta}(1,2) + (1-\mu_t)\frac{\alpha}{\alpha+\beta}\text{Beta}(2,1)\right] & \text{if } y = 0 \\ \text{proj}\left[\mu_t \frac{\alpha}{\alpha+\beta}\text{Beta}(2,1) + (1-\mu_t)\frac{\beta}{\alpha+\beta}\text{Beta}(1,2)\right] & \text{if } y = 1 \end{cases} \tag{11}$$

In these equations the proj operator maps the mixture of Beta distributions onto the closest Beta distribution. Ideally we would like the mixture and resulting Beta distribution to be close in terms of KL divergence, however for computational reasons it is often simpler to find the Beta distribution to match the moments of the mixture.

6.2 Message for the 2-Parameter Feedback Model

The factor for the 2-parameter model can be written as

$$g_4(y,t,u,\bar{u}) = (u^{[\![y]\!]}(1-u)^{(1-[\![y]\!])})^{[\![t]\!]}(\bar{u}^{(1-[\![y]\!])}(1-\bar{u})^{[\![y]\!]})^{(1-[\![t]\!])}.$$

Let us denote with $m_{t \to g_4}(t) = \text{Bernoulli}(\mu_t)$ and $m_{u \to g_4}(u) = \text{Beta}(\alpha, \beta), m_{\bar{u} \to g_4}(\bar{u}) = \text{Beta}(\bar{\alpha}, \bar{\beta})$ the incomming messages for factor g_4. The outgoing messages are

$$m_{g_4 \to t}(t) = \begin{cases} \text{Bernoulli}(1 + \frac{\alpha(\bar{\alpha}+\bar{\beta})}{\bar{\beta}(\alpha+\beta)}) & \text{if } y = 0 \\ \text{Bernoulli}(1 + \frac{\bar{\alpha}(\alpha+\beta)}{\beta(\bar{\alpha}+\bar{\beta})}) & \text{if } y = 1 \end{cases} \tag{12}$$

$$m_{g_4 \to u}(u) = \begin{cases} \text{proj}\left[\mu_t \text{Beta}(1,2) + (1-\mu_t)\frac{\alpha}{\alpha+\beta}\text{Beta}(1,1)\right] & \text{if } y = 0 \\ \text{proj}\left[\mu_t \text{Beta}(2,1) + (1-\mu_t)\frac{\beta}{\alpha+\beta}\text{Beta}(1,1)\right] & \text{if } y = 1 \end{cases} \tag{13}$$

Again, in these equations the proj operator maps the mixture of Beta distributions onto the closest Beta distribution as described above. Using a symmetry argument one can derive that the messages for $m_{g_4 \to \bar{u}}(\bar{u})$ are the same up to a swap of μ_t and $1 - \mu_t$.