Reconciling Factor Graph with User Feedback

Nguyen Thanh Tam
Distribute Information Systems Laboratory
École Polytechnique Fédérale de Lausanne
tam.nguyenthanh@epfl.ch

Abstract— **Factor graph is a representative graphical model to handle uncertainty of random variables. Factor graph has been used in various application domains such as named entity recognition, social network analysis, and credibility evaluation. In this paper, we study the problem of reducing uncertainty in factor graph towards reaching a common truth or deterministic information. We propose a pay-as-you-go approach that leverages user feedback for uncertainty reduction. As the availability of human input is often limited, we develop techiniques to identify the most uncertain spots in factor graph for maximizing the benefits of a given user feedback. We demonstrate the efficiency of our techniques on real-world applications.**

I. Introduction

Factor graph is a highly representative probabilistic graphical model that can express dependency relationships between random variables. A factor graph is a bipartite graph that expresses how a global function of many variables factors into a product of local functions [14]. Factor graphs subsume many other graphical models including Bayesian networks and Markov random fields. Factor graph has been applied in many application domains. With the rapid development of Web-based social applications and media, researchers have started using factor graphs for discovering and analyze social network structures (e.g. finding influential users [44] and finding community properties [47]). Moreover, factor graphs are also used in the area of Web credibility evaluation due to their ability to model the mutual reinforcing relationships between Web information and its providers [20].

In principle, a factor graph always has a degree of uncertainty due to the presence of latent variables. To reduce uncertainty, information needed to provided the true values of these variables. In this paper, we go beyond the common practice of using factor graph to model domain applications. Instead, we study the reduction of uncertainty in factor graph for the purpose of finding ground truth. This is because we can expect that in real-world settings, the ability of instantiating deterministic information with high confidence or guarantee is important for reaching concrete conclusions. More precisely, we use human input as the justification of random variables to reduce the uncertainty of a factor graph. The rationale behind our approach is that if the justification comes from uncertain sources (e.g. heuristic-base features or statistical evidences), the uncertainty might not be reduced at all. An uncertain justification might end up adding more uncertainty in the final result. In other words, it is necessary to to use a source with no uncertainty to reduce the uncertain information.

One of the primary problems facing the human-involved processes is the scalability issues. This is because eliciting inputs from humans can incur high cost and long interaction time. Moreover, in the era of Big Data, domain applications often contain a large number of datasets that need to be modeled by factor graph. As such, there are far too many variables that could benefit from user feedback. Since we cannot possibly ask user justification for all of variables, the pay-as-you-go principle applies: the factor graph model incrementally integrates new information to reduce uncertainty by asking user to justify uncertain variables. In such an iterative process, the optimization goal then becomes how to determine which variables for user feedback, in order to provide the most reduction of uncertainty in factor graph.

Achieving this goal is challenging due to the complex relationships between random variables in the factor graph. Knowing the true value of one variable can affect the likelihood state of other variables. Moreover, in some cases, the relationships can be directed. And thus, the order of user feedback matters: knowing the true value of one variable can be more beneficial than the others. Therefore, the optimization usually involves case analysis for different combinations of variables and different sequences of user inputs. This analysis grows exponentially with the factor graph size and the number of variables user can handle at once.

To overcome this challenge, we propose a heuristic-based ranking algorithm that assigns a goodness score to each variable. We formalize the requirements for designing the goodness score in terms of three dimensions. (1) *Information gain* – quantifies the gaining amount of potential information of knowing the true value of a variable. (2) *Minimal redundancy* – due to the complex relationships, user assertion on one variable can be propagated to the others. The selection of a variable should not be redundant in relative to the others. (3) *Potential impact* – Seeking user input on variables with most uncertainty may not have the largest impact on the factor graph. For example, an "isolated variable" (which do not have much connection with other variables) has very high uncertainty, but the information of knowing its true value cannot be reused throughout the factor graph.

To summarize, this paper brings the following main contributions:

- *Model:* It provides a generic model for incorporating user feedback into factor graphs in Section II. On top of this model, we develop a pay-as-you-go approach that benefits from user feedback to reduce the uncertainty in factor graph.

- *Process:* It formulates the process of reducing overall uncertainty in factor graph under limited budget constraint in Section III. This is motivated by the fact that user feedback is always limited; and hence, we aim to identify the “weakest spots” (most uncertain parts) in the factor graph in order to maximize the benefits of user input at any time.
- *Method:* It proposes, in Section IV, the design criteria of goodness function to select the most beneficial variables in factor graph for user assertion. Our selection maximizes the information gain, maximizes the potential impact, and minimizes the redundancy when incorporate user feedback into factor graph.
- *Termination:* It studies different halting conditions of the uncertainty reduction process in Section V. These conditions can help to further reduce user effort by early terminating the process when the factor graph is “good enough”.
- *Application:* It shows the applicability of the proposed techniques in the domains of entity recognition and credibility evaluation, in Section VI.

The remaining sections are structured as follows. Section VII summarizes related work, before Section VIII concludes the paper.

II. Model and Approach

A. Model

A factor graph is a tuple $G = \langle V, F \rangle$, where V is a set of random variables and F is a set of factors. In that, $V = V_u \cup V_c$, where V_u contains justifiable variables (i.e. variables which we can seek human input on) and V_c contains non-justifiable (e.g. constants). The human input sought as part of uncertainty reduction is modeled by a tuple $U = \langle U^+, U^- \rangle$, where U^+ and U^- are respectively a set of variables justified as $true$ and $false$.

Each variable is associated with a probability p that indicates the uncertainty degree of that variable. We maintain a set of probabilities P that contains the probability of each variable. Our probabilistic model acts a black-box, meaning that it contains all the information given by user feedback. As such, the user feedback U is integrated directly in the set of probabilities P: user inputs are assumed to be always right, so the probabilities of justified variables are either one or zero. Combining the introduced notions, we define a factor-graph based probabilistic model as a tuple $\langle G, P \rangle$ that represents a single state assumed during the uncertainty reduction.

B. Factor Graph with User Feedback

For incorporating user feedback, each variable v is associated with an extra factor node u_v. Figure 1 illustrates an example factor graph and Figure 2 depicts the resulting factor graph when incorporating user feedback. In our system, a user asserts for a given variable whether it shall be $true$ or $false$. We exploit user input by concluding that the variable must respect the validation:

$$uf(v) = \begin{cases} 1 & \text{If } v = true \\ 0 & \text{Otherwise} \end{cases} \tag{1}$$

Given a user input u^+ (u^-) on a correspondence v, we start the message passing on the variable node of v: $u_{v \to f(X_v=1)}$ (or $u_{v \to f(X_v=0)}$) to all of its neighbors. Since one correspondence is dependent on a limited number of other correspondences, only a small region of the network is updated. In case of batch mode (user gives feedback on multiple variables at the same time), we start the message passing concurrently from all the validated correspondences. The difference is that a factor node needs to wait all the incoming messages from neighboring asserted variables before sending out the outgoing messages.

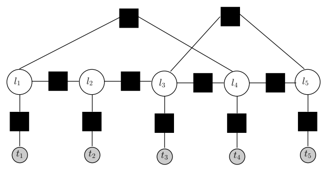

Fig. 1. An example factor graph

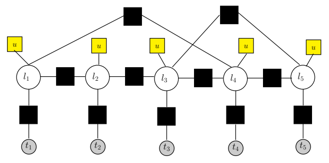

Fig. 2. Incorporate user feedback into FG

It is worth noting that this formulation is flexible for potential extensions such as user might make some errors and multiple users do the assertions at the same time. However, these situations are out of scope and are considered for future work.

C. The overall approach to uncertainty reduction

A general uncertainty reduction process is iterative, such that in each step, the user asserts the true value for a variable. This process comes to a halt when reaching a validation goal. Starting with a factor-graph based probabilistic model $\langle G, P \rangle$, the process continuously updates the state of variables. Each iteration of the process comprises the following steps:

(1) select k variables for which user feedback shall be sought
(2) elicit user input on the true value of each selected variable and update the corresponding probability.
(3) recompute the overall uncertainty.

It should be emphasized that our approach allows the selection of multiple variables (i.e. top-k) for user feedback at the same time. This is motivated by the two observations. First, if one variable is resolved at a time, even if quick, it requires human time scales, and comes with some overhead during human interaction. Second, the variable in the factor graph are not independent (via the factors), using this relationship information between variables might help user give better justification of their true values. As a result, the top-k selection can identify “white-spots”, i.e. parts of the factor graph where there has been little input so far. User input on a variable will be isolated if we do not consider the transitive links between variables.

III. Reducing Uncertainty in Factor Graph

A. Probability Computation

The probabilities of variables in factor graph can be computed or updated efficiently by Belief Propagation algorithm, which utilizes message passing to propagate the probabilities. More precisely, for each iteration, message passing is performed according to the following update rules:

variable node v to factor node f:

$$u_{v \to f}(v) = \prod_{f' \in n(v) \setminus \{f\}} u_{f' \to v}(v)$$

factor node f to variable node v:

$$u_{f \to v}(v) = \sum_{\neg\{v\}} \left(f(V) \prod_{v' \in n(f) \setminus \{v\}} u_{v' \to f}(v') \right)$$

where $n(.)$ stands for the neighbors of a variable / function node in the graph. $f(.)$ denotes the factor function associated with particular factor node f. $\sum_{\neg\{v\}}$ means to sum up the factor function over all variables except v.

These computations are known to be exact for cycle-free factor graphs; i.e. the algorithm terminates after sending two messages – one in each direction – for every edge in the factor graph. When the graph contains cycles, the algorithm only results in approximate solutions with multiple iterations. After the algorithm converges to a fixed point, the probability of each correspondence can be obtained as the product of all messages passing toward it:

$$Pr(v) = \frac{1}{Z} \prod_{f \in n(v)} u_{f \to v} v \tag{2}$$

where Z is a normalization constant factor ensuring that the probabilities of all possible values sum to one. The message passing algorithm allows to compute all marginal functions of a factor-graph in a concurrent and efficient manner. Moreover, it enables to collect new information (e.g. user input) incrementally without recomputing the whole graph. Note that our factor graph model acts as a black-box, meaning that it contains all the information given by user assertions. As such, the user input is integrated directly in the variable probabilities: user assertions are assumed to be always right, so the probabilities of asserted variables are either one or zero as follows.

Large-scale computation. The belief propagation takes $\mathcal{O}(n)$ time and $\mathcal{O}(n)$ memory [14]. In case of large-scale factor graphs, the number of variable nodes and factor nodes could be large than the computer's memory to handle. To circumvent this problem, one well-known approach is to partition the factor-graph by regrouping nodes (clustering or stretching transformations) [14]. However, one disadvantage of this approach is that the dependency between correspondences is broken down.

Another approach is resorting to sampling. Arguably the most popular of these approaches, and the one on which we focus, is Gibbs sampling. Achieving high throughput for Gibbs sampling is well studied for factor graphs that fit in main memory. However, applying sampling approach to large amounts of data faces several challenges, including parameter configuration and data storage. In this paper, we employ the state-of-the-art Elementary framework [33] that allows probability computations on large-scale factor graphs that are larger than main memory. Elementary has some important features: the customizable data model that supports sophisticated statistical learning and inference, an end-to-end system.

B. A Measure of Uncertainty

As mentioned above, each variable in the factor graph is associated with a probability that represents how like the variable is assigned as $true$. Hence, we measure the overall uncertainty of a factor graph as the Shannon entropy [39] over its set of random variables. Formally, the entropy for a factor graph is defined as follows:

$$H(G, P) = \sum_{v \in V} -P(v) \log P(v) \tag{3}$$

where V is the set of random variables of the factor graph G. An overall uncertainty $H(G, P) = 0$ means all probabilities are equal to one or zero; or in other words, the true values of all random variables are known. Hence, our goal is to reduce the overall uncertainty to zero.

A major reason for utilizing Shannon entropy as the measurement of uncertainty is its non-parametric nature. The probability distribution of random variables is very dynamic, depending on the domain applications. Entropy does not require any assumptions about the distribution of variables. Besides, entropy permits non-linear models, such as categorical variables [36].

C. The Process of Uncertainty Reduction

Reducing uncertainty in a pay-as-you-go fashion means that the probabilistic model is continuously updated by: (1) selecting a set of variables $D \subseteq V$, (2) eliciting user assertion on the variables, and (3) updating the probabilities in the probabilistic model $\langle G, P \rangle$. That is, by seeking user input for variables, the state of the probabilistic model $\langle G, P \rangle$ is changed, leading to the probabilistic model $\langle G, P' \rangle$, where P' is recomputed from user input as describe in the previous section. We denote this step of reducing the uncertainty with feedback on a set of variables $D \in V$ by $\langle G, P \rangle \xrightarrow{D} \langle G, P' \rangle$. The process of reducing uncertainty may come to a halt once the reconciliation goal is reached. Such a reconciliation goal may be given, for instance, in terms of an effort budget (i.e., the number of user assertions is limited) or a pre-defined threshold for the desired overall uncertainty.

A generic procedure of uncertainty reduction is illustrated in Algorithm 1. It takes the factor-graph based probabilistic model $\langle G, P \rangle$ and a reconciliation goal δ as input and returns a reconciled probabilistic model $\langle G, P' \rangle$. We proceed as follows: First, a set of top-k variables is selected from the candidate variables (i.e. justifiable variables) based on the information of factor graph. Second, we elicit the assertion for these variables. Third, we integrate user feedback by recomputing the probabilities P' and the overall uncertainty $H(V, P')$.

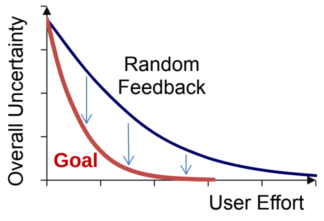

Fig. 3. Uncertainty minimization

Clearly, there is a trade-off between user effort and overall uncertainty: the greater the user input, the less overall uncertainty in the factor graph. Yet, instantiations of Algorithm 1 lead to a different realization of this trade-off. That is, the $select$ routine that chooses the variables for which feedback shall be sought has to be implemented and affects the degree of uncertainty reduction that is achieved by a certain amount of user input. As a baseline, we consider a human user working without any support tools. This scenario corresponds to the higher curve (random feedback) in Figure 3, in which the $select$ routine selects candidate variables for assertion in a random order. A more effective implementation of the $select$ routine would lower this curve, leading to a higher reduction in uncertainty for the same amount of user effort compared to the baseline.

Algorithm 1: Generic procedure for reducing uncertainty

```
  input : a factor-graph based probabilistic model $\langle G, P \rangle$,
          a reconciliation goal $\delta$
  output: a reconciled probabilistic model $\langle G, P' \rangle$
  // Initialization
1 $P' \leftarrow P$;
2 while not $\delta$ do
      // (1) Select a set of top-$k$ correspondences
3     $D \leftarrow select(G, P)$;
      // (2) Elicit and update user input
4     $\forall v \in D$, compute $uf(v)$ based on user input on $v$ ;
      // (3) Integrate the feedback
5     Recompute probabilities$N'$ $P'$ in the factor graph $G$ by the message
      passing algorithm ;
6     Recompute network uncertainty $H(G, P')$;
7 return $\langle G, P' \rangle$;
```

D. Problem Statement

To approach an effective implementation of the routine for selecting variables for assertion, we address a concrete reconciliation goal. Since reasonable thresholds for the overall uncertainty are hard to estimate and user feedback is commonly the bottleneck for reconciliation, we focus on limited budget of user effort. In that case, we would like to minimize overall uncertainty under a fixed number of feedback steps. Formally, our objective is defined as follows.

Problem 1 (Uncertainty Minimization): Let $\langle G, P \rangle$ be a factor-graph based probabilistic model, m be a budget of user effort (i.e. the number of user interaction steps), and k be the number of selected variables in each user step ($k \ll m$). The uncertainty minimization problem is the identification of variables $V' \subseteq V$ with $|V'| \leq m \times k$, such that $\langle G, P \rangle \xrightarrow{V'} \langle G, P' \rangle$ and $H(G, P')$ is minimal.

Finding a good identification strategy to solve the uncertainty minimization is challenging. Solving the problem optimally requires investigating all the permutations of all subsets (with size $\leq m \times k$) of candidate correspondences; this is computationally intractable.

IV. Top-k Selection

To address this problem, we focus on a greedy approach to ordering candidate variables for user feedback. The key concept used in our approach is the value of perfection information [38]. The value of perfect information is a means of quantifying the potential benefit of determining the true value for some unknown variables. Formally, in each user interaction step, we select a set of variables D with maximal value of perfect information:

$$\operatorname*{arg\,max}_{D \subseteq V, |D| \leq k} Q(D) \tag{4}$$

with $Q : 2^V \rightarrow R$ is the utility/benefit function indicating the value of perfect information of a variable set. Here, the selection of the top-k variables is of particular practical relevance for user feedback and active learning [19]. In general, increasing k avoids frequently asking the user, but might also exceeds the cognitive load of human user. An appropriate value for k depends on the user and the application context [32].

A. Goodness function

Now the task is to well define the function $Q(D)$ and design an efficient algorithm to maximize $Q(D)$. Without prior knowledge of domain applications, we argue that in general a good selection should satisfy the following requirements:

(R1) *Potential information:* Each variable in the factor graph has a distinguished amount of potential information. This information must be quantified in the presence of complex relationships between correspondences (i.e. some variables are connected via pre-defined factors while some others never go along with each other). Suggesting the variables with higher potential information would provide more chances of minimizing the number of user feedback steps for reducing the overall uncertainty.

(R2) *Avoidance of information redundancy:* Selection of correlated variables leads to high redundancy due to the fact that the user assertion on one variable can be propagated to the others. Therefore, the selection of a variable should be done related to other variables that have been selected and those already asserted.

(R3) *Consideration of variable importance:* Large groups of correlated variables hint that the information can be propagated widely, maximizing the information benefit of user feedback. This implies that a variable can be more important than another due to its connection degree in the factor graph.

(R4) *Avoidance of local optimal:* poor selection of variables can lead to local effect of user input when it is propagated within a small region of factor graph. To maximize the effects of user input on a wide scale, we need to aim for different variables across multiple regions of factor graph.

To combine the above requirements into a unified utility function, we implement by the following measurements.

Information gain. The first criterion is to select one by one the variables with highest information gain (i.e. selection of variables at the border of probabilities closed to 0.5):

$$IG(D) = \sum_{v \in D} IG(v) \tag{5}$$

The information gain of a variable v is then computed as the difference of the overall uncertainty resulting from the user justification of v, defined as follows:

$$IG(v) = H(G, P) - H(G|v, P) \tag{6}$$

where $H(G|v, P) = p_v H(G, P^+) + (1 - p_v) H(G, P^-)$ is the overall uncertainty conditioned by the assertions for a particular variable. In that P^+ are the variable probabilities of the factor graph after v is justified as $true$, and P^- are the variable probabilities of the factor graph after v is justified as $false$.

Redundancy penalty. Simply summation over information gain of each variable ignores the correlation between the variables in the factor graph. As a result, user justification efforts on these variables can be redundant. Therefore, there is a need to minimize the information overlap between the selected variables. Formally, the redundancy of the selected set can be measured by the following function:

$$R(D) = \sum_{v, v' \in D} s(v) M(v, v') s(v') \tag{7}$$

where the correlation function $M(v, v') = \frac{1}{Z} |\{f \in F : v, v' \in f\}|$ measures the number of factors that v and v' connect to, and Z is a constrant normalization factor ensuring that $M(.,.) \in [0, 1]$ (e.g. the largest number of factors between any two variables). Intuitively, we avoid selecting the variables that are correlated to each other and to those already asserted by user in previous iterations. $s(.)$ serves as a score factor when counting the correlation over variables. For example, we can use the information gain of the variable itself: $s(v) = IG(v)$ to reflect the correlation on the separating the possible values of v.

Put it altogether. The utility measure should incorporate given information scores of variables in a fine-grained level, by weighting the importance of variables unequally. The idea behind is that variables stemming from a large group of correlated variables often have a high chance to propagate information. Specifically, we define $q(v) = \sum_{v' \in V} M(v, v') IG(v')$ as the importance of variable v due to its ability to propagate information. Put it altogether, we formally design the utility function as a bi-criteria combination of information gain and redundancy:

$$Q(D) = w \sum_{v \in D} q(v) IG(v) - \sum_{v, v' \in D} IG(v) M(v, v') IG(v') \tag{8}$$

where $w \in R^+$ is a positive weight parameter to balance the information gain and redundancy terms.

While our notion of utility is motivated by the information propagation and correlation of variables, it also shows several intuitive properties that are detailed below.

Property 1 (Monotonicity): $Q(D)$ is monotonic. That is, $\forall D_1, D_2 \subseteq V$, $D_1 \cap D_2 = \emptyset$, we have:

$$Q(D_1 \cup D_2) \leq Q(D_1)$$

Proof: [Sketch] With $w \geq 2$, we have:

$$\begin{aligned}
Q(D_1 \cup D_2) - g(D_1) &= w \sum_{v \in D_2} q(v) IG(v) - (\sum_{v \in D_2, v' \in D_1} IG(v) M(v, v') IG(v') \\
&+ \sum_{v \in D_1, v' \in D_2} IG(v) M(v, v') IG(v') + \sum_{v, v' \in D_2} IG(v) M(v, v') IG(v')) \\
&= w \sum_{v \in D_2} IG(v) \sum_{v' \in V} M(v, v') IG(v') - (2 \sum_{v \in D_1, v' \in D_2} IG(v) M(v, v') IG(v') \\
&+ \sum_{v, v' \in D_2} IG(v) M(v, v') IG(v')) \geq 2 \sum_{v \in D_2} IG(v) \sum_{v' \in V} M(v, v') IG(v') \\
&- (2 \sum_{v \in D_1, v' \in D_2} IG(v) M(v, v') IG(v') + \sum_{v, v' \in D_2} IG(v) M(v, v') IG(v')) \\
&= 2 \sum_{v \in D_2} (\sum_{v' \in V} M(v, v') IG(v') - \sum_{v' \in D_1 \cup D_2} M(v, v') IG(v')) \\
&= 2 \sum_{v \in D_2} \sum_{v' \notin D_1 \cup D_2} M(v, v') IG(v') \geq 0
\end{aligned}$$

which completes the proof of monotonicity. ■

Property 2 (Submodularity): $Q(D)$ is submodular. That is, $\forall D \subseteq V$, $\forall v_1, v_2 \in V \setminus D$, we have:

$$Q(D \cup \{v_1\}) + Q(D \cup \{v_2\}) \geq Q(D \cup \{v_1, v_2\}) + Q(D)$$

Proof: [Sketch] We have:

$$Q(D \cup \{x\}) - Q(D) = wq(x) IG(x) - 2IG(x) \sum_{v \in D} M(x, v) IG(v) + IG^2(x) \tag{9}$$

Then with $w > 0$, we have:

$$\begin{aligned}
&Q(D \cup \{v_1\}) + Q(D \cup \{v_2\}) \geq Q(D \cup \{v_1, v_2\}) + Q(D) \\
&\Leftrightarrow Q(D \cup \{v_1\}) - Q(D) \geq Q(D \cup \{v_2\} \cup \{v_1\}) - Q(D \cup \{v_2\}) \\
&\Leftrightarrow wq(v_1) IG(v_1) - 2IG(v_1) \sum_{v \in D} IG(v) M(v, v_1) + IG^2(v_1) \\
&\geq wq(v_1) IG(v_1) - 2IG(v_1) \sum_{v \in D \cup \{v_2\}} IG(v) M(v, v_1) + IG^2(v_1) \\
&\Leftrightarrow 2IG(v_1) IG(v_2) M(v_1, v_2) \geq 0
\end{aligned}$$

which completes the proof of submodularity. ■

B. Maximizing the Goodness

Solving the utility function $Q(D)$ turns out to be intractable.

Theorem 1: Maximizing $Q(D)$ in Equation 8 is NP-complete.

Algorithm 2: Heuristic algorithm for top-k selection

```
input : A factor graph $G = \langle V, F \rangle$,
        The current validated variables $U \ll V$,
        a weight factor $w \geq 2$, and a threshold for the number of variables $k$.
output: A selection of variables $V^* \subseteq V \setminus U$ with $|V^*| = k$.
1 $V^* \leftarrow \emptyset$ ;
  // Compute ranking score for each variable
2 Let $r : V \rightarrow \mathbb{R}$, $r(v) \mapsto w \cdot IG(v) \cdot \sum_{v' \in V} M(v, v') IG(v')$;
3 while $|V^*| < k$ do
4     $v_m \leftarrow \arg\max_{v \in V \setminus U, v \notin V^*} r(v)$ ;
5     $V^* \leftarrow V^* \cap \{v_m\}$ ;
      // Update ranking score for the remaining variables
6     $r' \leftarrow r$;
7     Let $r' : V \rightarrow \mathbb{R}$, $r(v) \mapsto r'(v) - 2 \cdot IG(v_m) \cdot M(v, v_m) \cdot IG(v)$;
8 return $V^*$
```

Proof: [Sketch] As mentioned above, $Q(D)$ is a submodular set function. Maximization of submodular set functions is known to be NP-complete [23]. The proof is done. ■

Given the complexity of the set function maximization problem, we now present a heuristic algorithm to approximate its optimal solution. The main idea of our algorithm is to start from the null set and add one element at a time, taking at each step the element which increases the goodness of the selection most. To achieve a provably near-optimal solution, our algorithm exploits the two aforementioned properties of the goodness function g, i.e., monotonicity ad submodularity. In essence, the algorithm iteratively expands the selection of variables by adding the variable that maximizes the goodness value, thus it can be bounded. Solving the problem requires k iterations.

The details of our heuristic are given in Algorithm 2. It takes a factor graph $G = \langle V, F \rangle$, a set of validated variables $U \subset V$, a weight factor w, and a threshold for the number of variables k as input and returns a selection V^* of k variables. We begin by computing a ranking score for each variable $v \in V$ that is based on the weight factor, the information gain, the redundancy and the variable importance (line 2). In the actual greedy selection step, we select k variables. In each iteration, we add the variable with the highest ranking score (lines 4 and 5), before the ranking score is updated for the remaining variables (line 7). The latter avoids re-computation of the ranking scores from scratch in each iteration.

Algorithm Analysis. The proposed algorithm shows several desirable properties. First, the approximation error is bounded.

Guarantee 1 (Near-Optimality): Algorithm 2 is a (1- 1/e)-approximation for the variable selection problem.

Proof: For any monotone, submodular function f with $f(\emptyset) = 0$, it is known that an iterative algorithm selecting the element e with maximal value of $f(I \cup \{e\}) - f(I)$ with I as the set of elements selected so far has a performance guarantee of $(1 - 1/e) \approx 0.63$ [24]. This result is applicable to algorithm 2, since our goodness function Q is monotonic (property 1) and submodular (property 2), it holds $Q(\emptyset) = 0$, and the ranking score is defined as $r(v) = Q(V^* \cup \{v\}) - Q(V^*)$ (lines 2 and 7). ■

Next, we consider the complexity of our heuristic.

Guarantee 2 (Complexity): The time complexity and the space complexity of Algorithm 2 are $\mathcal{O}(|V|^2 + k|V|)$ and $\mathcal{O}(|V|)$, respectively.

Proof: Time complexity: The quadratic term $|V|^2$ stems from the computation of the ranking score. The linear term $k|V|$ is explained by k iterations, in each of which we iterate over all remaining variables, for selection of v_{max} and for updating the ranking score. Space complexity: Storing variable correlations requires $\frac{|V||V-1|}{2}$ space since M is symmetric and $M(v, v) = 1$. ■

C. Learning the trade-off parameter

There is a trade-off between information gain and redundancy criteria. If we only maximize the summation of individual information gain of selected variables in eq. (6), the actual information benefit of user feedback on these variables (i.e. the "joint" information gain) might be sub-optimal. This is because random variables in factor graph are correlated (via factor functions), the propagation of user input might be duplicated (i.e. redundancy). For example, let us have two variables v_1 and v_2 connected via a factor function $f(v_1, v_2) = 1$ iff $v_1 = v_2$. It is not necessary to ask user input on v_1 and v_2 at the same time since the true value of one variable can be inferred as a consequence of user input on the other.

The regularization parameter w in eq. (8) captures this trade-off between the information gain and redundancy of selecting random variables. The higher value of w, the more we favor information gain over redundancy; and vice-versa. Without prior knowledge, it is often difficult for user to set an appropriate value of w. We propose a guided searching method [1] to automatically determine the best-suited value. Informally, we would like w to be as large as possible to maximize the potential information of each variable while controlling a reasonable redundancy.

The searching method is technically defined as follows. First, we define the control criterion over the selection being that there does not exist any two variables being directly correlated. Formally, w is valid iff $\forall v \in D^*$, $\nexists v' \in D^*$ such that $\exists f \in F$ and $v, v' \in F$ where D^* is the output of top-k selection. Second, we find the upper bound w_{ub} that still satisfies the control criterion (e.g. starting with $w = 1$ and continuously double it until the control criterion is violated). Finally, we iterate over $[1, w_{ub}]$ to find the largest valid value of w. To speed up the search, we can leverage bisection method (i.e. binary search) to bisect the interval and select a subinterval recursively. Note that since w is real number, we set the minimum interval as 0.1.

V. Optimizing for Early Termination

The previous section described our algorithm for selecting the top validating candidates to minimize user effort. In practice, other optimization questions arise. For example, when the current factor graph is good enough, we can terminate the user feedback process early to further reduce cost. The major challenge of early termination is how to measure the quality of the current results. In Section V-A, we address how to decide when our factor graph is "good enough". In Section V-B, we study different halting conditions to stop eliciting validations from user. Given that a user can justify multiple variables in parallel, in Section V-C, we discuss the effect of selection size (number of selected variables in a validation round) on the uncertainty reduction performance.

A. Quality Assessment

Assessing the quality of current results is important for several reasons. First, it provides a guarantee on the quality that are dynamically improved by user feedback. In other words, by knowing the up-to-date quality, user would have a better understanding and strong feelings of trust on the efforts he spends for the validation. Moreover, it can provides a guideline on how much benefit is gained given a fixed budget. Based on this, user can decide whether to spend more budget or accept the current satisfactory results.

Extract computation. The true quality of a factor graph model is typically measured by the precision metric [17] on the entire variable set. Measuring the precision requires to compare the true value (i.e. ground truth) of each variable with its most probable value, which can be computed by the decision function as follows:

$$decide(v) = \begin{cases} 1 & \text{If } Pr(v) \geq 0.5 \\ 0 & \text{Otherwise} \end{cases} \tag{10}$$

Then, the precision is the ratio of random variables that are matched against ground truth:

$$A(G,P) = \frac{|v \in V \mid decide(v) = gold(v)|}{|V|} \tag{11}$$

where $gold(v)$ is the true value of v that can be obtained via humans. However, in practice we only have access to the ground truth of a (small) finite set of random variables. This is because the factor graph is large while the effort budget of seeking ground truth is limited (if we can validate all random variables, there is no need of probabilistic model in the first place).

Since the computation of exact precision is impractical, we present two heuristic quality estimations. Then, we elaborate on the pros and cons of each approach. [CANNOT: Finally, we show how to combine both heuristics in an efficient manner.]

k-fold cross estimation. User feedback is an iterative process in which we incrementally obtain a set of validated variables. An efficent approach is using these validated variables themselves as representatives of the entire graph. To this end, we tailor the standard k-fold cross validation technique [35] for quality estimation of factor graph. Informally, we randomly partition the user-validated data into "test" and "training" sets, and measuring the ability of the factor graph model learned on training variables to decide the test variables. Formally, given the set of validated variables U, we divide into k equal size partitions $U = U_1 \cup \ldots \cup U_k$. We repeat the following procedure k times: (i) leave the variables of the i-th partition U_i out as non-validated variables, (ii) recompute the variable probabilities of factor graph model $\langle G, P\rangle$ by the message-passing algorithm, (iii) compare the decision function of each variable $v \in U_i$ with the true value already given before by user to compute the "partial" precision:

$$A_{U_i}(G,P) = \frac{|v \in U_i \mid decide(v) = gold(v)|}{|U_i|} \tag{12}$$

For an accurate estimation, we take the average of k runs as an overall estimation of the model precision:

$$A_U(G,P) = \frac{\sum_{i=1}^{k} A_{U_i}(G,P)}{k} \tag{13}$$

The advantage of k-fold cross estimation is that all the validated variables are eventually used for both training and testing. As such, we will get an unbiased assessment of model quality since cross-validation estimate in an "out-of-sample" estimate [35]. However, this method has some drawbacks. First, using only the validatabled variables might have a "local-optimal" effect due to the complex relationships between variables in the entire factor graph. The selected variables might not be well-connected with the remaining ones, making the overall estimation has a high error rate. Second, the error rate of overall estimation depends on the number of folds k [35]. In practice, the choice of the number of folds depends on the dataset characteristics and the application domain.

Sampling-based estimation. To overcome the limitations of the k-fold cross estimation, we approach by not using the validated variables. The idea is to ask an oracle for the precision of the non-validate variables. Since the factor graph is large, we cannot compute the precision exactly (i.e. the same reason why the exact computation of precision is impractical). Rather we implement the oracle using sampling techniques. Technically, we take a sampling set of non-validated variables $\Omega \subset V \setminus U$ and ask human input for the true values of these variables. Thus, the sampling-based precision is:

$$A_\Omega(G,P) = \frac{|v \in \Omega \mid decide(v) = gold(v)|}{|\Omega|} \tag{14}$$

To obtain an accurate estimation, it is critical to draw good samples that well capture the random variables in the factor graph. Because of the complex joint relationships of these variables, uniform sampling methods like Monte Carlo are insufficient [5] for selection. We develop a non-uniform sampling that overcomes this limitation by making use of a random-walk strategy and simulated annealing. The idea is using the neighborhood factor of simulated annealing to guide the random-walk to "jump" over representative variables in the factor graph. The neighborhood factor can be measured by the connection degree of the variable node (i.e. the number of factors it connects to) which the random-walk is currently at. For details, refer to our technical report [31].

The main drawback of the sampling-based estimation is that we have to maintain a considerable cost for implementing the oracle (i.e. asking human input for ground truth). This cost is proportional to the sampling size, which indeed depends on the number of random variables if we want to obtain a highly accurate estimation.

B. When to Stop Asking

In practice, user feedback processes are often limited by a fixed budget B for the cost of human work and interactions. This is the baseline halting condition that was discussed in Problem 1. As our original problem is to minimize the effort given a pre-defined budget, we can further exploit the effort wisely by terminating the process early when it converges. At the beginning, the gain of incorporating user feedback is significant as overall uncertainty of the factor graph model is high. Once the process converges (e.g. variable probabilities are slightly or not changed), the gain of new user input becomes neglible. As such, the benefit of getting more feedback might be insignificant compared to the additional effort. Early termination helps to reduce the user effort further while the quality is unchanged. With the purpose of helping user decide on the termination, this section studies some practical indicators of the process convergence. By observing these indicators, user would gain more insights to make final decisions.

Uncertainty Reduction Rate. The purpose of this convergence indicator is to show the current effect of user feedback on the uncertainty reduction. At the beginning, the overall uncertainty of factor graph is high since there are still many non-validated variables. New user input thus has a high amount of benefit as its information can be propagated widely on many variables. As long as more user feedback is received, the overall uncertainty is reduced (the number of uncertain variables is reduced). Therefore, the information propagation is narrowed down, leading to the insignificant benefit of getting new human input.

Formally, after each validation step, the factor graph model $\langle G, P\rangle$ becomes $\langle G, P'\rangle$. The reduction rate of uncertainty can be measured by the ratio of the uncertainty difference before and after the validation:

$$\frac{H(G,P) - H(G,P')}{H(G,P)} \tag{15}$$

When the process converges, the uncertainty reduction rate reaches to zero. User can decide to terminate the process when the rate levels off (e.g. $\leq 1\%$).

The number of changes. While the previous indicator is based on the variable probabilitties, this indicator only concerns the most probable value of each variable. This is motivated by the fact that users might be interested in the instantiation of the probabilistic model (i.e. returning the most probable world) rather than the probability values themselves. The purpose of this metric is to measure the change of the instantiation in two consecutive feedback iterations. In some cases, the overall uncertainty is reduced but the most probable values are unchanged. For example, before user feedback, we have $Pr(v = 1) = 0.8$ and $Pr(v = 0) = 0.2$; and after user feedback, we have $Pr(v = 1) = 0.9$ and $Pr(v = 0) = 0.1$. The uncertainty of variable v is indeed reduced but its most probable value (i.e. 1) is unchanged. After a considerable number of iterations, if the number of changes is zero or insignificant, it implies that the factor graph model is likely to be correct. In other words, the factor graph is stable if the current decision of variables will not change by the new user feedback we choose to forgo. This is because an incorrect instantiation (improbable world) would be strongly changed by user input.

Formally, given that $\langle G, P\rangle \xrightarrow{V'} \langle G, P'\rangle$, the unstability (or the number of changes) of the factor graph can be measured by:

$$\sum_{v \in G} \mathbb{1}(decide(v,P) \neq decide(v,P')) \tag{16}$$

Note that, this metric might be a local indicator if user feedback only affects a small region of factor graph (which, consequently, makes the change insignificant). However, as our selection strategy favors picking variables across multiple regions, this indicator should be global. User can decide an early termination if the unstability is, for example, less than 10% after 3 consecutive iterations.

The number of good predictions. Another useful indicator that helps user have a strong feeling of a "good" model is the ability to instantiate the variables matched with user input. Intuitively, if the decision of a variable is matched with user feedback, it indicates that the factor graph model is in good state. The more of these consecutive correct predictions, the stronger feelings of trust on the model quality.

Formally, in a user feedback iteration, we have $\langle G, P\rangle \xrightarrow{V'} \langle G, P'\rangle$. The number of good predictions over one iteration can be measured by:

$$\sum_{v \in V'} \mathbb{1}(uf(v) = decide(v,P)) \tag{17}$$

Here, $\mathbb{1}(c)$ is the decision function which is 1 when condition c holds and is zero otherwise. User can decide to terminate the process when, for example, there are more than 90% good predictions after 3 consecutive iterations.

Precision Improvement Rate. The above metrics are, in fact, "indirect" indicators of the process convergence. These indicators might not be accurate at some points since they are based on heuristic obserations. The direct way to measure the convergence is using the precision of model itself, which is computed as in the previous section. Formally, let us have the transition $\langle G, P\rangle \rightarrow \langle G, P'\rangle$ after a validation step. The improvement precision rate of the factor graph model can be measured by:

$$\frac{A(G,P') - A(G,P)}{A(G,P)} \tag{18}$$

The benefit of using the precision improvement rate is the independence of the domain characteristics of factor graph as it is based on user input. However, this indicator is costly to compute as it involves seeking more ground truth (i.e. sampling-based approach) or computing multiple times (i.e. k-fold cross validation approach). As such, we approach by only showing this indicator periodically (e.g after each 3-5 iterations).

Look-ahead. In general, the above indicators are computed only after the user feedback is truly elicited. The drawback of these "history-based" indicators is that the effects can only be known when user effort actually happens. This might lead to some redundant user inputs when the selection of candidate variables for validation are not optimal as expected. To avoid this limitation, the purpose of this indicator is to foreseen the expected effects of user input before-hand.

We can implement the look-ahead of each aforementioned indicator by estimating how much information benefit can be gained in the next iteration as follows. After a validation step, we have the transition $\langle G, P\rangle \rightarrow \langle G, P'\rangle$. We will use the instantation of the current factor graph model $\langle G, P'\rangle$ as user feedback. Specifically, given a set of selected variables V'' of $\langle G, P'\rangle$ to be validated, we assume $uf(v) = decide(v, P')$ for each $v \in V''$. Then we recompute the new state of factor graph $\langle G, P''\rangle$ by the messaging-passing algorithm. Finally, we measure the relative difference between the "current" value and the "look-ahead" value of a given indicator $I(G, P')$:

$$\frac{|I(G, P') - I(G, P)|}{I(G, P)} \tag{19}$$

One advantage of look-ahead mechanism is the ability to dynamically deal with the convergence speed of the validation process. For example, the number of changes can reduce quickly at the beginning but slowly later on. By looking ahead, we can estimate at some point the change would be small and eligible to terminate the user feedback early. The early termination can be decided when, for example, the relative comparison between the current computation and the look-ahead computation of a given indicator is less than 1%.

C. *Effect of Selection Sizes*

At each iteration of the validation process, we must choose a subset of the non-validated variables accoording to their selection scores, and send it to user for justification. We call this subset a batch (denoted as V' in $\langle G, P\rangle \xrightarrow{V'} \langle G, P'\rangle$). An interesting question is how to set this batch size, say β.

Intuitively, a smaller β increases opportunities to improve the factor graph effectiveness in a fine-grain manner. Previously requested variables will be truly incorporated before deciding which variables to request next. For instance, best results are achieve when $\beta = 1$. However, larger batch sizes reduce the overall run-time substantially by (i) allowing user to justify multiple variables at the same time due to their dependence relationships in factor graph, and (ii) reducing the number of iterations. This is confirmed by our experiments, which show that the impact of increasing β on the effectiveness of factor graph model is less dramatic than its impact on the overall run-time. Thus, to find the optimal β, a reasonable choice is to start from a smaller batch size and continuously increase it (say, double it) until the run-time becomes reasonable, or the user cognitive load is reached.

VI. Deployment on Real Applications

Factor graph has been used as a graphical model for a wide variety of applications, such as signal processing [16], sensor networks [7, 8, 18], and video decoding [22]. In the following, we illustrate the two highlighted applications as case studies:

A. *Named Entity Recognition*

The problem of named entity recognition (NER) is to label each token in the text document with an entity type. NER is a subtask of information extraction that has been thriving for more than 20 years [21]. It aims at extracting and classifying mentions of rigid designators, from text, such as proper names, biological species, and temporal expressions. The uncertainty of NER problem comes from the inherent ambiguity of natural texts. Therefore, a probabilistic graphical model like factor graph can be applied to capture this uncertainty.

1) Model: In [43], the authors have a dataset with ten-million tokens from 1788 New York Times articles from the year 2004. They define the relational factor graph over the TOKEN relation as in Figure 4. Formally, given an unannotated block of text (e.g.), we can model this text as a factor graph $G = \langle V, F\rangle$, where V is the set of random variables and F is the set of factors.

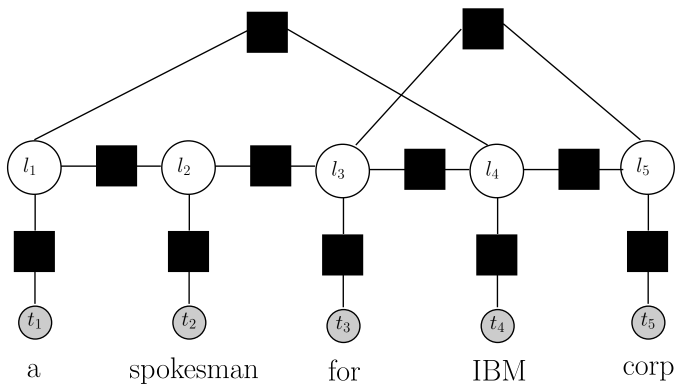

Fig. 4. Named Entity Resolution (NER)

Variables. The set of random variables $V = \{t_1, \ldots, t_n\}$. Each variable t_i in a token in the text. A possible value of a variable is the entity type of the associated token. There are many different entity types, for example, "PER" (person entity such as Bill), "ORG" (organization such as IBM), "LOC" (location such as New York City), "MISC" (miscellaneous entity – none of the above), and "O" (not a named entity).

Factors. In [43], the authors defined three factor templates: (1) factors between observed strings and corresponding labels, (2) transition factors between consecutive labels, and (3) bias factors over labels. The underlying models to implement these factors include traditional linear chain model for NER and skip-chain models [43].

2) User Feedback: Figure 5 illustrates how to incorporate user feedback in the NER factor graph. Each label variabe l_i (of each token t_i) is associated with an additional user feedback factor u.

$$u(l_i) =' PER' |' ORG' |' LOC' |' MISC' |' O' \tag{20}$$

When user gives feedback (i.e. assign the true label/entity type for a given token), the current state of the factor graph is recomputed by the message-passing algorithm with the user feedback factor defined in Equation 1.

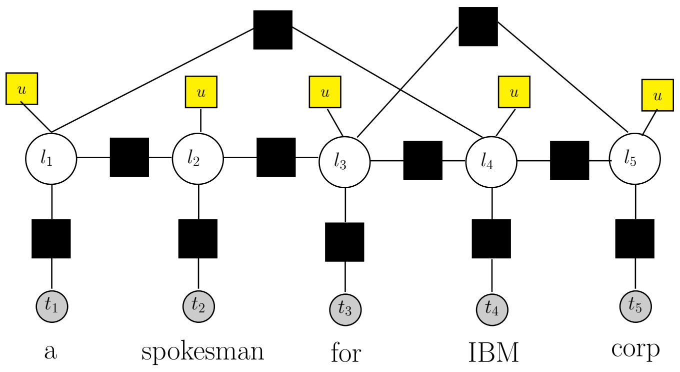

Fig. 5. User Feedback Factor Graph for NER

B. Credibility Evaluation

One of the major issues in online communities is the widespread concern regarding the quality and credibility of user-generated content [34]. To address this issue, the literature often formulates the problem of assessing the credibility of statements made by users in their online posts (e.g. blogs, forums). The problem output will help to identify trustworthy users and discover important facts on the Web.

To solve this problem, the authors of [20] approach by leveraging the intuition that there is an important interaction between statement credibility, linguistic objectivity, and user trustworthiness. We therefore model these elements jointly through a probabilistic graphical model, more specifically a factor graph, where each statement, post and user is associated with a binary random variable. Figure 6 provides an overview of our model. For a given statement, the corresponding variable should have value 1 if the statement is credible, and 0 otherwise. Likewise, the values of post and user variables reflect the objectivity and trustworthiness of posts and users.

1) Model: The primary goal of the proposed factor graph is to retreive the credibility label of unobserved statements given some expert labeled statements and the observed features by leveraging the mutual influence between the model's variables. The factor graph will capture the following relationships: (i) each user is conencted to all hist posts, (ii) each statement is connected to all posts from which it can be extracted (by state

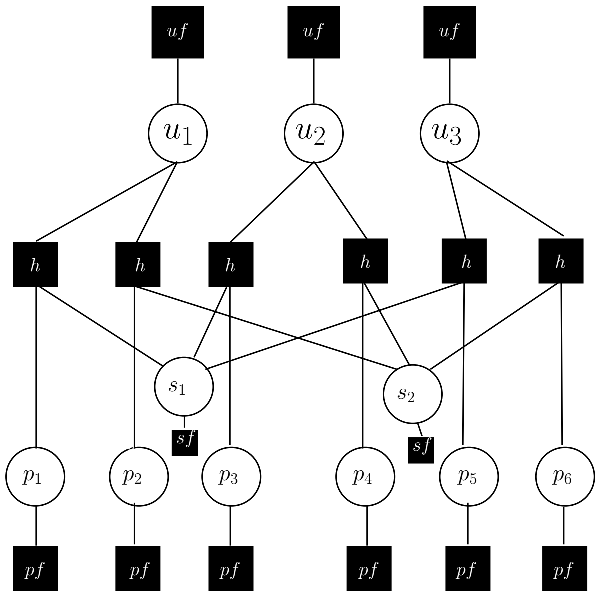

Fig. 6. Credibility Evaluation

of the art information extraction methods), (iii) each user is connected to statements that appear in at least one of his posts.

Variables. Formally we have $u_i \in \{0,1\}$ is the random variable that represents the user trustworthiness. $u_i = 0$ means that the user u_i must not be trusted while $u_i = 1$ means user u_i is trusted. Second, $s_i \in \{0,1\}$ is the random variable that represents the statement credibility. $s_i = 0$ means that the statement is not credible and $s_i = 1$ means that the statement is credible. Third, $p_i \in \{0,1\}$ is the random variable that represents the post objectivity. $p_i = 0$ means that the post is subjective whereas $p_i = 1$ means that the post is objective.

Factors. Nodes associated with users and posts have observable features, which can be extracted from the online community.

User factors/features. For users, we derive engagement features (number of questions and answers posted), interaction features (e.g. replies, giving thanks), and demographic information (e.g. age, gender). The features are presented in details in [20].

$$uf(u_i) = \exp(F_1(u_i)F_2(u_i)\ldots F_n(u_i) \tag{21}$$

where $F_j(u_i)$ is the j-th feature of user u_i.

Post factors. For posts, we extract linguistic features in the form of discourse markers and affective phrases. The features are presented in detail in [20].

$$pf(p_i) = \exp(F_1(p_i)F_2(p_i)\ldots F_n(p_i)) \tag{22}$$

where $F_j(p_i)$ is the j-th feature of post p_i.

Statement factors. While for statements there are no observable features, we can use distant supervision technique on top of external knowledge bases (e.g. in the domain of medical there are expert databases, like the Mayo Clinic, which list typical as well as rare side-efects of widely used drugs). This information is contained in the factor $sf()$ in the factor graph.

Connection factors. Each factor $h()$ depicts the mutual reinforcing relationship between user, post, and statement.

$$h(u_i, p_j, s_k) = exp(u_i \cdot p_j \cdot s_k) \tag{23}$$

2) User Feedback: Figure 7 illustrates how to incorporate user feedback into the credibility evaluation factor graph. We connect each justifiable variable (user trustworthiness, statement credibility, and post objectivity) with a user feedback factor u. There are three different types of user feedback: (i) feedback on user trustworthiness, (ii) feedback on statement credibility, and (iii) feedback on post objectivity. In general, the user feedback can be modeled as a generic function:

$$u(x) = \begin{cases} 1 & \text{If user justifies that } x = 1 \\ 0 & \text{If user justifies that } x = 0 \end{cases} \tag{24}$$

When a variable is asserted by user, the probabilities are recomputed by the message-passing algorithm.

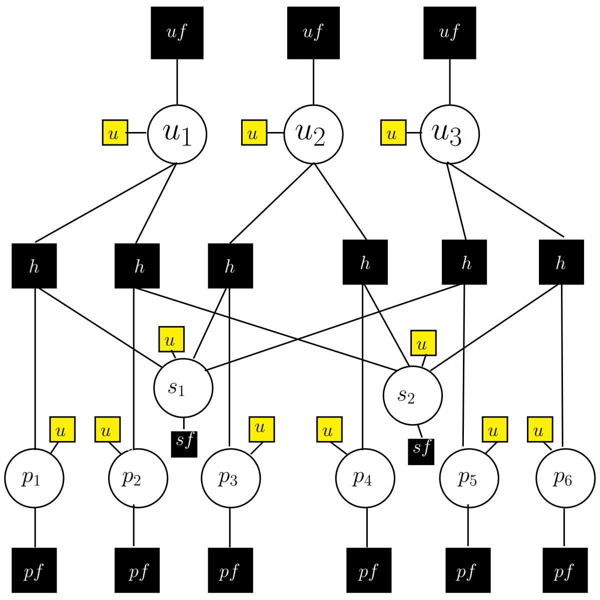

Fig. 7. Credibility Evaluation with User Feedback

VII. Related Work

Factor graph and applications. Factor graph has been used to for various applications in the areas of database, information retrieval, and trust management. In [43], the authors use factor graph to solve the named entiry recognition task by modeling tokens as variables and linguistic features as factors. In [41], the authors leverage factor graph to model the entity resolution problem by capturing the mentions of entities as variables and the semantic relationships between them as factors. In [20], the authors evaluate the trustworthiness of online users and the credibility of their statements on the Web by encoding the mutual reinforcing relationship between users and statements into factor graph. Our proposed techniques on reducing factor graph uncertainty is generic ans thus can be tailored in these applications.

Learning with user feedback. Exploiting human intelligence to improve the automatic models has been studied in various contexts such as training classifiers [42], entity extraction [2], sentiment analysis [15], credibility evaluation [6], data integration [3, 4, 11, 25, 26, 28, 30], and crowdsourcing [9, 10, 27, 29]. For example, in the knowledge acquisision task to build knowledge bases, human input is used for assessing the validity of facts and for gathering additional knowledge [13]. In machine learning, user feedback is used as ground truth to train classifiers [19] and probabilistic models including factor graph [40, 46]. To the best of our knowledge, there is no work on comprehensively and systematically harnessing user feedback to assess the validity of random variables in factor graph. Since the proposed techniques are generic, our work can be tailored to other probabilistic formulations such as Markov models and Bayesian models as well.

Guiding user feedback. Guiding user or expert feedback has been studied in different contexts. In the field of data integration, Jeffery et al. [12] proposed a decision theoretic framework to rank candidate matches for answer validation in order to improve the quality of a dataspace. Focusing on matching of data schemas in a network setting, Nguyen et al. [31] presented a reconciliation algorithm that leverages expert input. Yakout et al. [45], in turn, proposed an active-learning based process that requests expert input to help training classifiers in order to detect and repair erroneous data. Similar to these works, we rely on models from the fields of Decision Theory and Active Learning [37]. Despite the similarities in the applied models, there are a number of differences between the aforementioned approaches to user guidance and the method presented here. First, our method is independent of application domains. The selection of feedback candidates is purely based on the information measure and the network structure of a generic factor graph itself. Second, we also proposal additional guiding indicators on the current quality of the factor graph model for further reducing user effort.

VIII. Conclusions

This paper proposes a novel approach that enables pay-as-you-go reconciliation in factor graph. We formulate the need of reducing uncertainty in factor graph due to the heuristic nature of the factor functions that constitute the probable values of random variables in the graph. To this end, we design an incremental process that leverages user feedback for uncertainty reduction. The problem we tackle is to fine a set of candidate variables with maximal information gain. Since computing the joint information gain is intractable, we propose heuristic criteria to estimate information gain, including *individual information*, *redundancy*, and *potential* *impact*. The problem turns out to be NP-hard; and thus, we propose a bounded greedy algorithm to find the approximate solution. We further optimize user efforts by proposing user-guiding indicators for the convergence status of the uncertainty reduction process. Finally, we demonstrate the applicability of involving human input for various factor graph applications.

References

[1] D. T. Anh, V. H. Tam, and N. Q. V. Hung. "Generating complete university course timetables by using local search methods." In: *RIVF*. 2006, pp. 67–74.

[2] G. Demartini, D. E. Difallah, and P. Cudré-Mauroux. "ZenCrowd: Leveraging Probabilistic Reasoning and Crowdsourcing Techniques for Large-scale Entity Linking". In: *WWW*. 2012, pp. 469–478.

[3] A. Gal et al. "Completeness and ambiguity of schema cover". In: *CoopIS*. 2013, pp. 241–258.

[4] A. Gal et al. "Making sense of top-k matchings: A unified match graph for schema matching". In: 2012, p. 6.

[5] C. Gomes, J Hoffmann, and A Sabharwal. "From sampling to model counting". In: *IJCAI*. 2007, pp. 2293–2299.

[6] Z. Huang, A. Olteanu, and K. Aberer. "CredibleWeb: a platform for web credibility evaluation". In: *CHI*. 2013, pp. 1887–1892.

[7] N. Q. V. Hung, H. Jeung, and K. Aberer. "An evaluation of model-based approaches to sensor data compression". In: *TKDE* (2013), pp. 2434–2447.

[8] N. Q. V. Hung, S. Sathe, D. C. Thang, and K. Aberer. "Towards enabling probabilistic databases for participatory sensing". In: *CollaborateCom*. 2014, pp. 114–123.

[9] N. Q. V. Hung, D. C. Thang, M. Weidlich, and K. Aberer. "ERICA: Expert guidance in validating crowd answers". In: *SIGIR*. 2015, pp. 1037–1038.

[10] N. Q. V. Hung, D. C. Thang, M. Weidlich, and K. Aberer. "Minimizing efforts in validating crowd answers". In: *SIGMOD*. 2015, pp. 999–1014.

[11] N. Q. V. Hung et al. "SMART: A tool for analyzing and reconciling schema matching networks". In: *ICDE*. 2015, pp. 1488–1491.

[12] S. R. Jeffery, M. J. Franklin, and A. Y. Halevy. "Pay-as-you-go user feedback for dataspace systems". In: *SIGMOD*. 2008, pp. 847–860.

[13] S. Kondreddi, P. Triantafillou, and G. Weikum. "Combining information extraction and human computing for crowdsourced knowledge acquisition". In: *ICDE*. 2014, pp. 988–999.

[14] F. Kschischang, B. Frey, and H.-A. Loeliger. "Factor graphs and the sum-product algorithm". In: *TIT* (2001), pp. 498–519.

[15] X. Liu et al. "Cdas: a crowdsourcing data analytics system". In: *VLDB*. 2012, pp. 1040–1051.

[16] H.-A. Loeliger et al. "The factor graph approach to model-based signal processing". In: *Proceedings of the IEEE* (2007), pp. 1295–1322.

[17] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to information retrieval*. Vol. 1. Cambridge university press, 2008.

[18] Y. Mao, F. R. Kschischang, B. Li, and S. Pasupathy. "A factor graph approach to link loss monitoring in wireless sensor networks". In: *Selected Areas in Communications, IEEE Journal on* (2005), pp. 820–829.

[19] B. Mozafari, P. Sarkar, and M. Franklin. "Scaling Up Crowd-Sourcing to Very Large Datasets: A Case for Active Learning". In: *VLDB*. 2014.

[20] S. Mukherjee, G. Weikum, and C. Danescu-Niculescu-Mizil. "People on Drugs: Credibility of User Statements in Health Communities". In: *KDD*. 2014, pp. 65–74.

[21] D. Nadeau and S. Sekine. "A survey of named entity recognition and classification". In: *Lingvisticae Investigationes* (2007), pp. 3–26.

[22] M. R. Naphade, I. Kozintsev, T. S. Huang, and K. Ramchandran. "A factor graph framework for semantic indexing and retrieval in video". In: *Content-based Access of Image and Video Libraries, 2000. Proceedings. IEEE Workshop on*. 2000, pp. 35–39.

[23] G. L. Nemhauser and L. A. Wolsey. "Maximizing submodular set functions: formulations and analysis of algorithms". In: *North-Holland Mathematics Studies* (1981), pp. 279–301.

[24] G. Nemhauser, L. Wolsey, and M. Fisher. "An analysis of approximations for maximizing submodular set functions–I". In: *MP* (1978), pp. 265–294.

[25] H. Q. V. Nguyen et al. "Minimizing human effort in reconciling match networks". In: *ER*. 2013, pp. 212–226.

[26] Q. V. H. Nguyen, S. T. Do, T. Nguyen Thanh, and K. Aberer. "Privacy-Preserving Schema Reuse". In: *DASFAA*. 2014, pp. 234–250.

[27] Q. V. H. Nguyen, T. T. Nguyen, N. T. Lam, and K. Aberer. "BATC: a benchmark for aggregation techniques in crowdsourcing". In: *SIGIR*. 2013, pp. 1079–1080.

[28] Q. V. H. Nguyen, T. T. Nguyen, Z. Miklós, and K. Aberer. "On Leveraging Crowdsourcing Techniques for Schema Matching Networks". In: *DASFAA*. 2013, pp. 139–154.

[29] Q. V. H. Nguyen, T. Nguyen Thanh, T. Lam Ngoc, and K. Aberer. "An evaluation of aggregation techniques in crowdsourcing". In: *WISE*. 2013, pp. 1–15.

[30] Q. V. H. Nguyen et al. "Collaborative Schema Matching Reconciliation". In: *CoopIS*. 2013, pp. 222–240.

[31] Q. V. H. Nguyen et al. "Pay-as-you-go reconciliation in schema matching networks". In: *ICDE*. 2014, pp. 220–231.

[32] T. T. Nguyen, Q. V. H. Nguyen, M. Weidlich, and K. Aberer. "Result selection and summarization for Web Table search". In: *ICDE*. 2015, pp. 231–242.

[33] F. Niu, C. Zhang, C. Ré, and J. Shavlik. "Elementary: Large-scale knowledge-base construction via machine

learning and statistical inference". In: *IJSWIS* (2012), pp. 42–73.

[34] T. G. Papaioannou, J.-E. Ranvier, A. Olteanu, and K. Aberer. "A Decentralized Recommender System for Effective Web Credibility Assessment". In: *CIKM*. 2012, pp. 704–713.

[35] P. Refaeilzadeh, L. Tang, and H. Liu. "Cross-validation". In: *Encyclopedia of database systems*. 2009, pp. 532–538.

[36] A. RRNYI. "On measures of entropy and information". In: *Fourth Berkeley Symposium on Mathematical Statistics and Probability*. 1961, pp. 547–561.

[37] S. J. Russell and P. Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education, 2003.

[38] S. J. Russell et al. *Artificial intelligence: a modern approach*. Prentice hall Englewood Cliffs, 1995.

[39] C. E. Shannon and W. Weaver. *A mathematical theory of communication*. 1948.

[40] L. Shi, Y. Zhao, and J. Tang. "Batch Mode Active Learning for Networked Data". In: *ACM Trans. Intell. Syst. Technol.* (2012), 33:1–33:25.

[41] S. Singh, K. Schultz, and A. McCallum. "Bi-directional joint inference for entity resolution and segmentation using imperatively-defined factor graphs". In: *ECML PKDD*. 2009, pp. 414–429.

[42] C. Sun, N. Rampalli, F. Yang, and A. Doan. "Chimera: Large-Scale Classification using Machine Learning, Rules, and Crowdsourcing". In: *VLDB*. 2014, pp. 1529–1540.

[43] M. Wick, A. McCallum, and G. Miklau. "Scalable probabilistic databases with factor graphs and MCMC". In: *VLDB*. 2010, pp. 794–804.

[44] H. Xu, Y. Yang, L. Wang, and W. Liu. "Node classification in social network via a factor graph model". In: *PAKDD*. 2013, pp. 213–224.

[45] M. Yakout et al. "Guided data repair". In: *VLDB*. 2011, pp. 279–289.

[46] Z. Yang, J. Tang, B. Xu, and C. Xing. "Active Learning for Networked Data Based on Non-progressive Diffusion Model". In: *WSDM*. 2014, pp. 363–372.

[47] Z. Yang, J. Tang, J. Li, and W. Yang. "Social community analysis via a factor graph model". In: *IEEE Intelligent Systems* (2011), pp. 58–65.