PageRank, ProPPR, and Stochastic Logic Programs

Dries Van Daele, Angelika Kimmig, and Luc De Raedt

Department of Computer Science, KU Leuven
{dries.vandaele,angelika.kimmig,luc.deraedt}@cs.kuleuven.be

Abstract. ProPPR is the first probabilistic logic language using personalized PageRank for inference. We consider personalized PageRank on the SLD tree of a stochastic logic program (SLP), show that the resulting probability distribution over answer substitutions can be represented by an incomplete SLP, and relate this result to ProPPR.

1 Introduction

The PageRank algorithm [5], originally proposed for ranking web pages, has recently been used as the basis for efficient inference in the probabilistic Prolog ProPPR [7], a language similar in spirit to stochastic logic programs (SLPs) [2, 4], but biased towards short derivations. In this paper, we consider inference by PageRank for SLPs, show that the resulting probability distribution over answer substitutions can be represented by an incomplete SLP, and discuss differences with the ProPPR case.

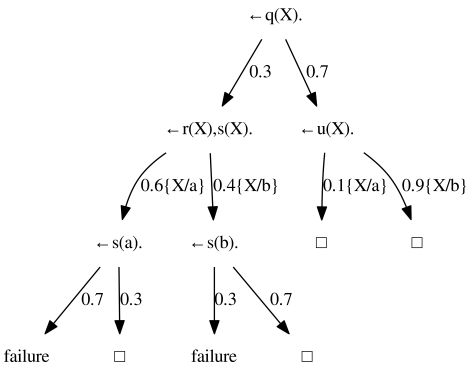
2 Background

2.1 PageRank

The PageRank algorithm [5] assigns a weight to individual web pages, expressing their relative importance on the World Wide Web. Intuitively, this PageRank distribution can be regarded as the likelihood that a 'random surfer' [1] arrives at each respective web page. By regarding the World Wide Web as a graph where web pages are nodes and hyperlinks are edges, a 'random surfer' can be simulated by executing a modified random walk over it. In each step, an edge is followed by choosing uniformly between the outgoing edges. To ensure that the random walk can reach every part of the graph, PageRank introduces a fixed probability jump to any node in the graph. When this probability is not uniformly distributed over all the nodes, the algorithm is referred to as personalized PageRank.

2.2 Stochastic Logic Program

A stochastic logic program (SLP) [2, 4] is a set of definite clauses with a probabilistic interpretation. In a definite clause $a \leftarrow b$, the head a consists of a single




```
0.3: q(X) ← r(X), s(X).
0.7: q(X) ← u(X).

0.6: r(a).
0.4: r(b).

0.3: s(a).
0.7: s(b).

0.1: u(a).
0.9: u(b).
```

Fig. 1. SSLD tree with probabilities annotating its edges and underlying SLP

positive literal, and the body $b = a_1, \ldots, a_n$ is a conjunction of positive literals. A pure SLP associates a probability label with each of its clauses (i.e. $p : a \leftarrow b$ where $p \in [0, 1]$). In a complete SLP, the probability labels of the clauses that share the predicate symbol in their head sum up to one. We use the pure and complete SLP on the right of Figure 1 as running example throughout the text.

The root of the SLD tree for query q and SLP $\mathcal{S}$ is labeled $\leftarrow q$, and the tree is constructed by adding for every node N and every clause C in $\mathcal{S}$ whose head has the same predicate as N's first subgoal a child node labeled with the subgoal obtained by SLD resolution (for more details, see [3]). Nodes with empty goals are labeled $\square$ (and called success nodes), nodes for which the resolution fails are labeled failure.

Definition 1 (SSLD tree). *The stochastic SLD tree (SSLD tree) $\mathcal{T}_\mathcal{S}$ for a pure SLP $\mathcal{S}$ and query q is the SLD tree for q and $\mathcal{S}$, where each edge is labeled with the probability of the clause used in the corresponding resolution step. $\mathcal{N}$ is the set of success nodes (labeled $\square$) in $\mathcal{T}_\mathcal{S}$, and $\mathcal{N}_\theta$ the subset of $\mathcal{N}$ resulting in a given answer substitution θ. For each node $N \in \mathcal{N}$, we have*

$$P_\mathcal{S}(N) = \prod_{p_i : C_i \in d(N)} p_i$$

where $d(N) = p_1 : C_1, \ldots, p_n : C_n$ is the derivation ending in N.

In the SSLD tree for our example, cf. Figure 1, the probability of the second derivation $0.3 : q(X) \leftarrow r(X), s(X), 0.6 : r(a), 0.3 : s(a)$ is $0.3 \cdot 0.6 \cdot 0.3 = 0.054$, and similar for all others.

Definition 2 (SLP probability $P_\mathcal{S}$). *Given a pure SLP $\mathcal{S}$, the probability of a query q with answer substitution θ is given by*

$$P_\mathcal{S}(q\theta) = \frac{\sum_{N \in \mathcal{N}_\theta} P_\mathcal{S}(N)}{\sum_{M \in \mathcal{N}} P_\mathcal{S}(M)}$$

In our example, we get

$$P_{\mathcal{S}}(\mathtt{q(a)}) = (0.054 + 0.07)/(0.054 + 0.07 + 0.084 + 0.63) = 0.148$$
$$P_{\mathcal{S}}(\mathtt{q(b)}) = (0.084 + 0.63)/(0.054 + 0.07 + 0.084 + 0.63) = 0.852.$$

2.3 ProPPR

The probabilistic logic programming language ProPPR [7] uses personalized PageRank on a graph similar to the SSLD tree for inference in a logic program with probabilistic interpretation. Though the focus of ProPPR is on efficient approximate inference by further modification of that graph, we here consider the case of exact inference and its relation to inference for SLPs.

The differences between the SSLD tree and ProPPR's graph are:

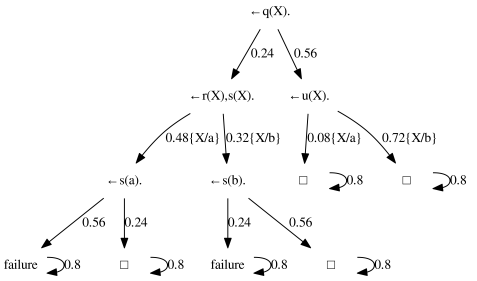
1. The ProPPR graph is limited to those nodes and transitions used in successful SSLD resolution steps. This corresponds to the modified SSLD tree where all failure nodes and their ingoing edges are omitted.
2. Each success node in the ProPPR graph has an edge leading back to itself.

3 Personalized PageRank over the SSLD Tree

We now consider personalized PageRank over the full SSLD tree, linking back to ProPPR in Section 4. We distinguish two separate cases, with and without looping edges for the success and failure nodes. As in ProPPR, a jump in our personalized PageRank always leads to the root of the SSLD tree, thus simply restarting the walk for that query.

Definition 3 (PageRank SSLD graphs). *For SLP $\mathcal{S}$, query q and restart probability $\alpha \in (0, 1]$, the non-loopy PageRank SSLD graph $\mathcal{G}_{\mathcal{S},\alpha}$ consists of the nodes and edges of the SSLD tree $\mathcal{T}_{\mathcal{S}}$ and an additional edge from every node to the root. An edge labeled p_i in $\mathcal{T}_{\mathcal{S}}$ is labeled $p_i \cdot (1 - \alpha)$ in $\mathcal{G}_{\mathcal{S},\alpha}$, edges from leaves to the root are labeled 1, and all other restart edges are labeled α. The loopy PageRank SSLD graph $\mathcal{G}^{loop}_{\mathcal{S},\alpha}$ is $\mathcal{G}_{\mathcal{S},\alpha}$ extended with a direct self-loop for every leaf ($\square$ or failure) of $\mathcal{T}_{\mathcal{S}}$. An edge labeled p_i in $\mathcal{T}_{\mathcal{S}}$ is labeled $p_i \cdot (1 - \alpha)$ in $\mathcal{G}^{loop}_{\mathcal{S},\alpha}$, self-loops are labeled $1 - \alpha$, and all restart edges are labeled α.*

Figure 2 shows the loopy PageRank SSLD graph for the SLP in Figure 1 with query $q(X)$ and $\alpha = 0.2$, where we omit restart edges to avoid clutter. An alternative description is given by the logic program labeled with PageRank SSLD probabilities in the middle of Figure 2. The special symbol $\mathcal{R}$ for the jump to the query node highlights the extra-logical nature of the restart step, which is similar to Prolog's cut operator, but has a far greater effect. Resolution on a goal such as (`r(X)`, `s(X)`) using the restart clause for the first subgoal `r(X)` results in the initial query with fresh variables, i.e., all subsequent subgoals (here just `s(X)`) are dropped, and all substitutions obtained so far forgotten.



0.24: q(X) ← r(X), s(X).
0.56: q(X) ← u(X).
0.20: q(X) ←$\mathcal{R}$.

0.48: r(a).
0.32: r(b).
0.20: r(X) ←$\mathcal{R}$.

0.24: s(a).
0.56: s(b).
0.20: s(X) ←$\mathcal{R}$.

0.08: u(a).
0.72: u(b).
0.20: u(X) ←$\mathcal{R}$.

0.24: q(X) ← r(X), s(X).
0.56: q(X) ← u(X).

0.48: r(a).
0.32: r(b).

0.24: s(a).
0.56: s(b).

0.08: u(a).
0.72: u(b).

Fig. 2. The loopy PageRank SSLD graph (restart edges omitted) for the SLP in Fig. 1 and an explicit notation for that SLP under the PageRank interpretation with restart probability 0.2 (middle) and the corresponding incomplete SLP (right).

Definition 4 (PageRank SSLD distribution). *A (loopy or non-loopy) PageRank SSLD graph $\mathcal{G}$ defines a PageRank vector π with an entry π_N for every node N in the graph. For a query q with answer substitution θ, we obtain the PageRank SSLD probability*

$$P_{\mathcal{G}}(q\theta) = \frac{\sum_{N \in \mathcal{N}_\theta} \pi_N}{\sum_{M \in \mathcal{N}} \pi_M}$$

where $\mathcal{N}_\theta$ is the set of success nodes with answer substitution θ, and $\mathcal{N}$ the set of all success nodes.

The PageRank vector can be computed iteratively, starting from the vector placing all the probability mass on the query node. The resulting distribution is is equivalent to the limiting distribution [6] of the Markov chain that is described by the graph where the jumps to the root query are explicitly included (the nodes of the graph become the states of the Markov chains and labeled edges become transitions). In our example, we obtain

$$P_{\mathcal{G}}(\mathtt{q(a)}) = \frac{0.027648 + 0.0448}{0.027648 + 0.0448 + 0.4032 + 0.043008} = 0.1397$$
$$P_{\mathcal{G}}(\mathtt{q(b)}) = \frac{0.4032 + 0.043008}{0.027648 + 0.0448 + 0.4032 + 0.043008} = 0.8603$$

These differ from the ones for the original SLP, but can be obtained from a modified, incomplete SLP (shown on the right of Figure 2 for our example).

Theorem 1. *For a given SLP $\mathcal{S}$, query q and restart probability α, let $\mathcal{S}_\alpha$ be the incomplete SLP obtained by multiplying all labels in $\mathcal{S}$ with $1-\alpha$. For every answer substitution θ, we have*

$$P_{\mathcal{G}_{\mathcal{S},\alpha}}(q\theta) = P_{\mathcal{G}^{loop}_{\mathcal{S},\alpha}}(q\theta) = P_{\mathcal{S}_\alpha}(q\theta)$$

We prove the theorem by showing that PageRank converges to a vector that defines the same normalized distribution over success nodes as $\mathcal{S}_\alpha$.

Convergence It suffices to show that the PageRank SSLD Markov chain is irreducible, aperiodic, and positive-recurrent [6]. Since every state can be reached from and has a restart transition back to the query state, the chain is irreducible. Since the restart transition in the query node creates a loop, any state can be visited at an irregular time, and the chain is thus aperiodic. Given that the chain is irreducible, positive recurrency directly follows for finite state spaces (as given by finite SSLD trees), and can be shown for infinite state spaces by showing that the mean recurrence time of one state (i.e. the expected number of steps before that state is first revisited) is finite. The mean recurrence time of the query state in the loopy PageRank SSLD graph $\mathcal{G}^{loop}_{\mathcal{S},\alpha}$, where every state has a transition with probability α to the query state, is given by the finite quantity

$$\mathrm{MRT}(q) = \sum_{n=1}^{\infty} n \cdot \alpha \cdot (1-\alpha)^{n-1} = \frac{1}{\alpha}$$

For the case of $\mathcal{G}_{\mathcal{S},\alpha}$, we have $\mathrm{MRT}^{\mathrm{loop}}(q) \leq \int_\alpha \mathrm{MRT}(q)$. Let a be the infimum over the set of all restart transition probabilities in the Markov chain. Since $\int_{\alpha=a}^{1} \frac{1}{\alpha}$ for fixed $a > 0$ is finite, the query node is positive recurrent.

Equivalence of distributions over success nodes We first consider the non-loopy graph $\mathcal{G}_{\mathcal{S},\alpha}$. Let π_q be the value associated with the query node in the corresponding PageRank vector π, and π_N the value associated with the success node reached by the SSLD refutation $p_1 : C_1, \ldots, p_n : C_n$, where the p_i are the labels in $\mathcal{S}$. The corresponding labels in $\mathcal{S}_\alpha$ are thus $p'_i = p_i \cdot (1-\alpha)$. We have

$$\pi_N = \pi_q \cdot \prod_{i=1}^{n} p'_i$$

which is the probability of the corresponding refutation in $\mathcal{S}_\alpha$ multiplied by the constant π_q. When normalizing over all successful derivations, this constant cancels out, and we thus obtain the same distribution as for $\mathcal{S}_\alpha$. Similarly, for the loopy graph $\mathcal{G}^{loop}_{\mathcal{S},\alpha}$, we have

$$\pi_N = \pi_q \cdot \prod_{i=1}^{n} p'_i + (1-\alpha) \cdot \pi_N = \frac{\pi_q}{\alpha} \cdot \prod_{i=1}^{n} p'_i$$

which again only differs by a multiplicative constant from the probability of the corresponding refutation in $\mathcal{S}_\alpha$.

4 Relation to ProPPR

All of the states in the Markov chain given by a PageRank SSLD graph (excluding the success (and failure) states in the non-loopy case) have a constant

restart transition α. The transition probabilities of the remaining outgoing edges in each node are scaled in order to ensure that the total outgoing probability adds up to one. ProPPR instead normalizes over all outgoing edges, including the restart edge. Furthermore, as mentioned in Section 2.3, ProPPR performs PageRank over a partial SSLD tree, omitting all failure nodes and their ingoing edges. Because of these differences, each transition probability in the Markov chain given by a ProPPR program will be dependent on the probability of its associated clause, the basic restart weight α, as well as the probabilities of the other clauses that successfully unify with its associated goal in the SSLD tree. Because of this additional dependency, it is not possible to concisely represent a general ProPPR program as an SLP.

The distribution expressed by a ProPPR program is nevertheless closely related to our PageRank SSLD distribution. The convergence proof used in Theorem 1 applies also to the graph used by ProPPR, and the values of the success nodes of that graph are computed in the same manner. As a result, any difference in their respective probability distributions is due to their different approach to deriving transition probabilities.

5 Conclusions

We have shown that the probability distribution over answer substitutions expressed by personalized PageRank over the full SSLD tree of a pure and complete SLP corresponds to the probability distribution over answer substitutions given by an incomplete SLP with labels computed from the complete program and the PageRank restart probability. Furthermore, we have outlined its relationship with ProPPR. This clarifies the link between these languages and opens up possibilities for using ProPPR's efficient inference approaches for SLPs.

References

1. Prasad Chebolu and Páll Melsted. PageRank and the random surfer model. In *Proceedings of the 19th annual ACM-SIAM symposium on Discrete algorithms*, 2008.
2. James Cussens. Stochastic logic programs: Sampling, inference and applications. In *Proceedings of the 16th Conference on Uncertainty in Artificial Intelligence*, 2000.
3. Peter Flach. *Simply Logical*. John Wiley, 1994.
4. Stephen Muggleton. Stochastic logic programs. *Advances in inductive logic programming*, 32:254–264, 1996.
5. Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.
6. Sidney I. Resnick. *Adventures in Stochastic Processes*. Birkhauser Verlag, 1992.
7. William Yang Wang, Kathryn Mazaitis, and William W Cohen. Programming with personalized PageRank: a locally groundable first-order probabilistic logic. In *Proceedings of the 22nd ACM International Conference on information & knowledge management*, 2013.