A Comparison of Bottom-Up Approaches to Grounding for Templated Markov Random Fields

Eriq Augustine  
UC Santa Cruz  
eaugusti@ucsc.edu

Lise Getoor  
UC Santa Cruz  
geetoor@soe.ucsc.edu

1 Introduction

Markov Random Fields (MRFs) have been shown to be a flexible and powerful tool in modeling complex problems [5]. Templating languages like Markov Logic Networks (MLN) [2] and Probabilistic Soft Logic (PSL) [1] have emerged to help construct MRFs. They are particularly well-suited to constructing MRFs over richly structured domains, because they provide a logical formalism for describing entities and their relationships, and a compact mechanism for describing the parameters of the Markov Random Field. These languages define weighted rules using a first order logic-like syntax. The rules act as templates for real-valued feature functions called potentials. These templates are then combined with data to form ground rules in a process called grounding. Together, the ground rules define the probability distribution of an MRF. More formally:

Definition 1.1. Let \( \mathbf{x} = (x_1, \ldots, x_n) \) be a vector of known variables, \( \mathbf{y} = (y_1, \ldots, y_n) \) be a vector of unknown random variables, \( R = (R_1, \ldots, R_l) \) be a set of rules, \( w = (w_1, \ldots, w_l) \) be a set of real-valued weights each corresponding to a rule, and \( \phi = (\phi_1, \ldots, \phi_l) \) be a set of potentials where each potential corresponds to a rule. Then \( \phi_{R}(x_r, y_r) \) assigns the variables of this ground instance \( r \) of rule \( R \) a real-valued score. Then, a templated Markov Random Field is a probability distribution of the form:

\[
P(y|x) \propto \exp \left( \sum_{R \in R} \sum_{r \in R} w_{R} \phi_{R}(x_r, y_r) \right)
\]

2 Grounding Templated MRFs

Grounding, the process of instantiating each rule in the templated MRF, is a key step in performing inference. Grounding is a difficult problem and can be the limiting factor of inference in templated MRFs [6, 9–11].

Consider a simple link prediction rule: \( \text{Link}(R1, R2) \land \text{Sim}(R2, R3) \rightarrow \text{Link}(R1, R3) \) or a simple transitive equality rule from the entity resolution domain: \( \text{SameRef}(R1, R2) \land \text{SameRef}(R2, R3) \rightarrow \text{SameRef}(R1, R3) \). These rules seem simple, but the number of groundings is potentially cubic in number of entity references in the input data. Rules like these allow our MRF to be expressive, but present a significant systems challenge to efficiently ground.

Grounding is generally approached in two ways: top-down and bottom-up. Top-down grounding starts with the rules and employs nested loops to perform replacements over all the variables. It is simple and easy to implement, but slow. Alternatively, bottom-up grounding expresses the grounding for each rule as a database query. As a result, grounding leverages the huge amount of query optimization work done by the RDBMS community. Bottom-up grounding has shown significant improvements over top-down grounding, but comes at the cost of more complex systems [9].

There are several key design decisions that need to be made when building a bottom-up grounding system.
the language. If blocking information is explicitly known, then grounding queries can be written to reflect the blocking structure. This has the potential to greatly reduce the size of intermittent operations performed by the database as well as the number of joins required by the database for grounding queries.

3 Experiments\(^1\)

3.1 Defining Inference Targets & Scoping

The work of defining explicit targets is done outside of the grounding system. Therefore, it would be unfair to compare the time to generate targets between implicit and explicit target definition system. Instead, we look at how scoping can effect grounding time in Tuffy, a system that implicitly defines targets. Scoping is a cost-efficient way for methods that implicitly define targets to limit the target set. Figure 1 shows the grounding time for the same program with and without scoping. Using scoping reduced the grounding time by on average more than 30%.

3.2 Trivial Potential Removal

Next we investigate the impact of where trivial potentials are removed. We ground a MRF with a single rule:

\[
\text{SIMILAR}(P1, P2) \rightarrow \text{FRIENDS}(P1, P2)
\]

on datasets which vary the percentage of trivial potentials from 0% to 100%.

In all cases, the full size of the MRF (including trivial potentials) is around 160K potentials. We compare Tuffy, which removes trivial potentials at the database level, with PSL, which removes trivial potentials at the application level. Figure 2 shows the grounding time of Tuffy and PSL with a varying number of threads. When there are fewer trivially satisfied rules, the database removal performance is very poor. It becomes more competitive as more potentials require removal. However, it is only at the 80% trivial mark that we see a crossover where database removal outperforms single-threaded application removal. At the 100% trivial mark, database removal outperforms all shown methods\(^2\). Because application removal requires testing all materialized potentials, it runs in constant time regardless of the percentage of trivial potentials.

3.3 Blocking

Next, we investigate the impact of blocking for three rules:

- Similarity – \(\text{SIMILAR}(P1, P2) \rightarrow \text{FRIENDS}(P1, P2)\)
- Symmetry – \(\text{FRIENDS}(P1, P2) \rightarrow \text{FRIENDS}(P2, P1)\)
- Transitivity – \(\text{FRIENDS}(P1, P2) \land \text{FRIENDS}(P2, P3) \rightarrow \text{FRIENDS}(P1, P3)\)

We compare three different blocking methods:

- No Blocking – Results in a larger result set, but there is no blocking overhead.
- Implicit Blocking – Blocking structures are defined as data. This generates a reduced result set, but suffers overhead from a larger grounding query.
- Explicit Blocking – System is given explicit knowledge about the desired blocking structures. An optimal query is constructed that minimizes the result set and number of joins.

The results are summarized in Table 1.

<table>
<thead>
<tr>
<th>Query</th>
<th>Result Size</th>
<th>Max Node Size</th>
<th>Query Time (ms)</th>
<th>Result Size</th>
<th>Max Node Size</th>
<th>Query Time (ms)</th>
<th>Result Size</th>
<th>Max Node Size</th>
<th>Query Time (ms)</th>
</tr>
</thead>
<tbody>
<tr>
<td>No Blocking</td>
<td>999,000</td>
<td>999,000</td>
<td>211</td>
<td>999,000</td>
<td>999,000</td>
<td>385</td>
<td>997,002,000</td>
<td>997,002,000</td>
<td>266,313</td>
</tr>
<tr>
<td>Implicit Blocking</td>
<td>100,204</td>
<td>100,204</td>
<td>599</td>
<td>100,204</td>
<td>100,204</td>
<td>700</td>
<td>10,064,832</td>
<td>11,063,313</td>
<td>8,385</td>
</tr>
<tr>
<td>Explicit Blocking</td>
<td>100,204</td>
<td>100,204</td>
<td>49</td>
<td>100,204</td>
<td>100,204</td>
<td>50</td>
<td>10,064,832</td>
<td>10,064,832</td>
<td>2,809</td>
</tr>
</tbody>
</table>

3.4 End-to-End Performance

So far, we have focused on the decisions made when grounding and the impact of those decisions. However, it is important to also understand grounding in the context of the end-to-end inference over an MRF. Figure 3 shows the full run times of Tuffy and PSL for varying problem sizes. Figures 4 and 5 further show the breakdown of how much time is spent for each major task. In PSL, no single task dominates the run time. In Tuffy, however, we see that inference tends to dominate and as the result sets becomes large, the time spent grounding also jumps up.

4 Conclusion

From our experiments, we see several takeaways to consider when designing bottom-up grounding for templated MRFs. If using implicit target definition, be sure to include scoping. Removing trivial groundings at the database level is typically not worth the query overhead. In contrast, removing groundings in the application layer allows systems to exploit parallelism. Explicitly knowing the blocking structure can provide a huge boost to performance both in terms of memory usage and run time. However, there is still much more to be understood, when optimizing overall system performance.

Acknowledgements: This work was supported by NSF CCF-1740850, NSF IIS-1703331, AFRL, and DARPA.
References


