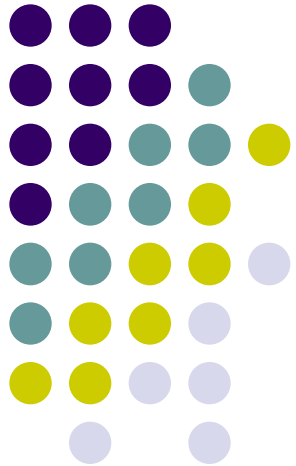
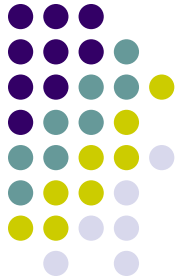
A Unified Framework for Disambiguating Proper Names

Ying Wang

Xiangrui Meng



Why Do We Need to Disambiguate?

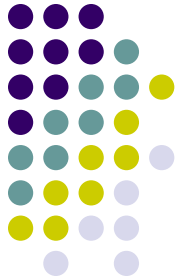


In enhancing metadata, one key step is match the extracted proper names to a knowledge base.

For each proper name (person, event, or location), there can be several matches in the knowledge base.

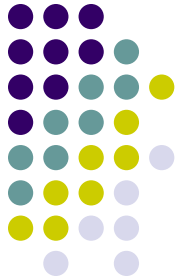
For example, if we happen to extract a person name “John Smith”…

Why is Disambiguation so Hard?



- The actual reference of the word is usually context-dependent.
  - Let's go to *Oxford* together!
- The context is ignored when matching to the knowledge base.
- It is exceedingly hard to "learn" the context by a computer program.

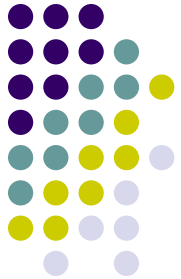
What is the Goal?



We want an algorithm that is:

- Highly accurate.
- Resistant to noise.
- Efficient in time and memory.
- If all matching candidates in the knowledge base are false matches, the algorithm should detect and report this fact.

Our Approach

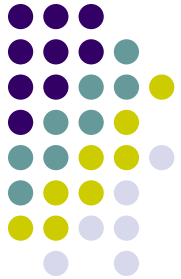


We avoid learning the context explicitly.

We take advantage of the implicit correlation (or compatibility) among named entities.

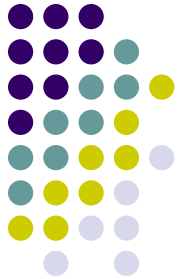
Example: Napoleon lost his last battle in Waterloo.

Big Picture



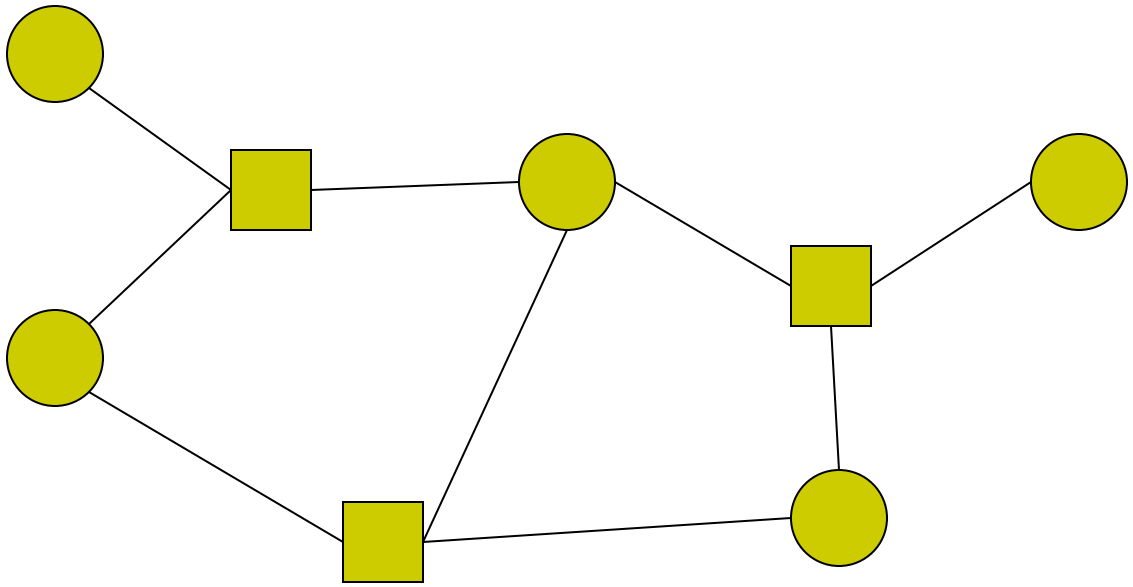
We quantify “compatibility” between groups of proper names, and then we choose a “realization” that maximizes this compatibility.

Factor Graph Model



- There are 2 types of nodes: variable nodes and function nodes.
- Each variable node can take a state among several choices.
- Each function node is connected to several variable node, and it computes the interaction of these nodes.

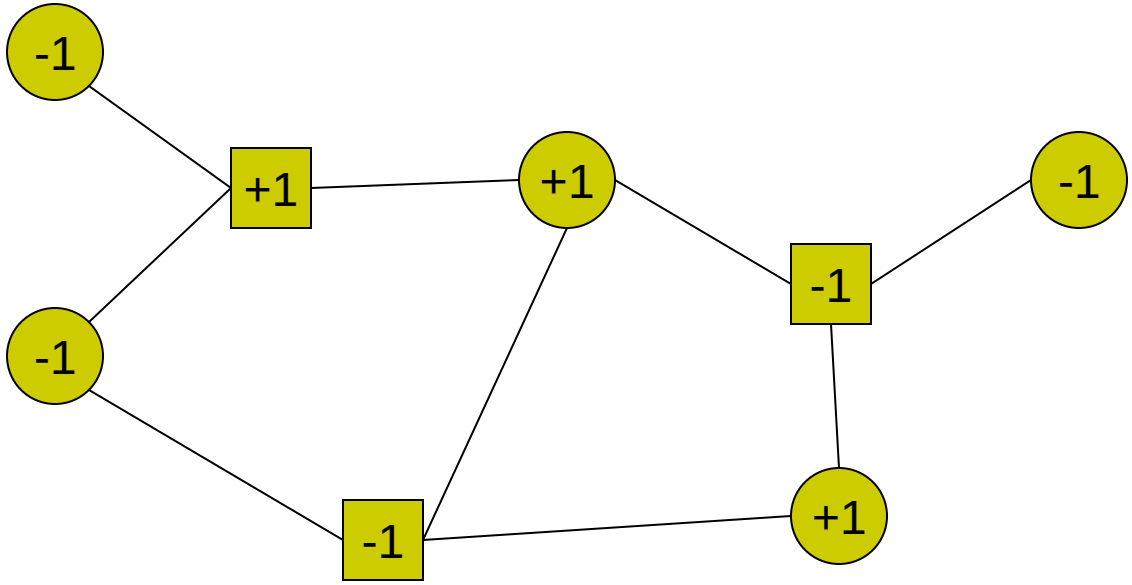
An Example of Factor Graph



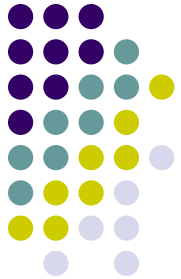
Each circle represents a variable node, which can take value either +1 or -1.

Each square represents a function node, which computes the product of its neighbors.

A Realization of the Factor Graph



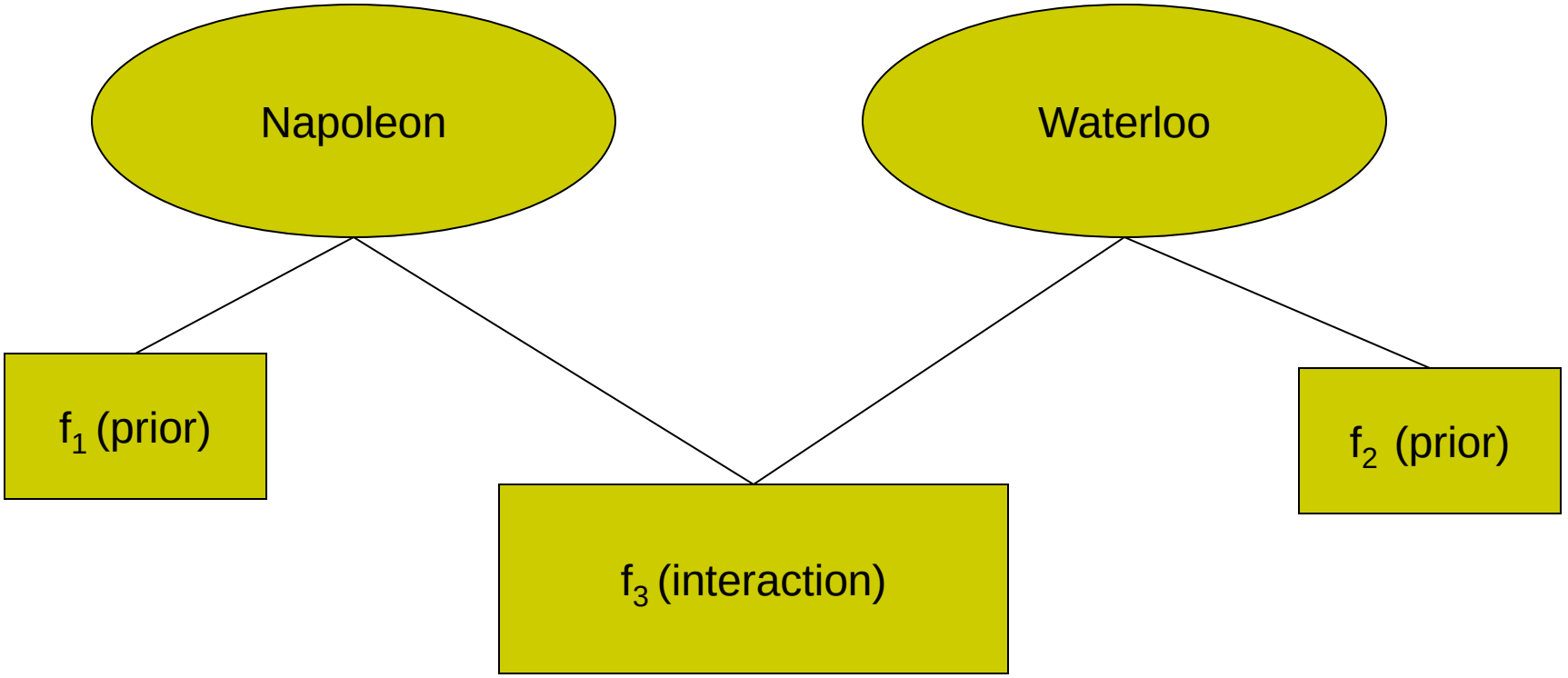
Applying the Factor Graph Model



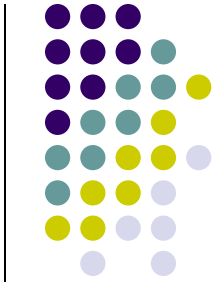
Each identified proper name will be a variable node, whose choice of states will be its possible matches in the knowledge base.

Function nodes will be added to compute the interactions between proper names.

An Example of Factor Graph



Prior Distribution

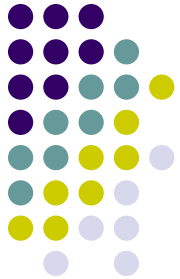


Each variable node has a prior distribution over its possible states. For example, when no other information presents, “San Francisco” may refer to the city in CA with 99% probability, and 1% being something else.

The prior distribution is ignoring the interaction.

Essentially, the prior distribution can be represented by a function node connected only to one variable node.

MAP Estimate

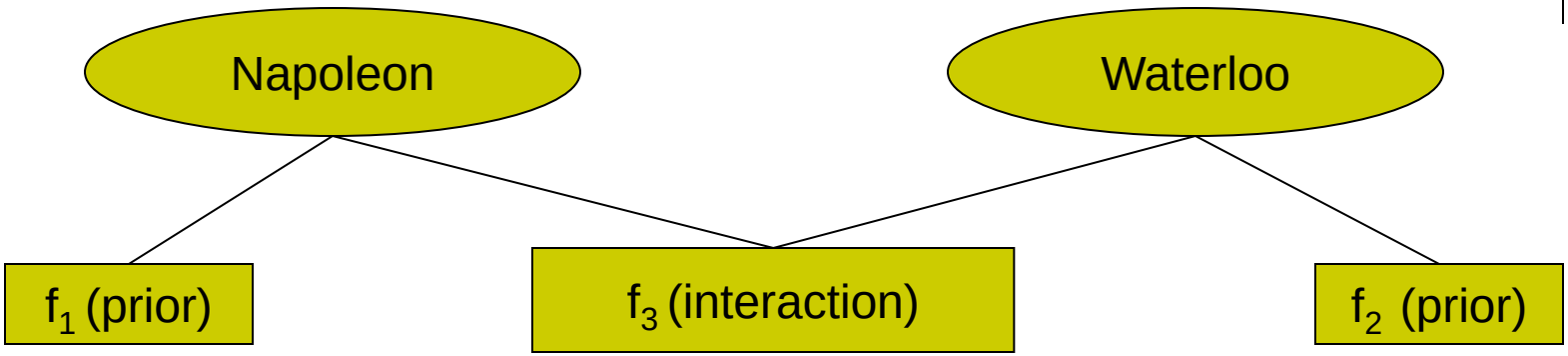


Given a realization of variable nodes, each function node compute a score based on the states of variable nodes.

The total score (energy) of this realization will be the product of the scores of all function nodes.

We want to find the realization with maximum score, which is called the maximum a posteriori (MAP) estimate.

Back to the Napoleon Example…



Suppose Napoleon can be either the French Emperor or his son, with prior

f_1(Nap I) = 0.99 f_1(Nap II) = 0.01

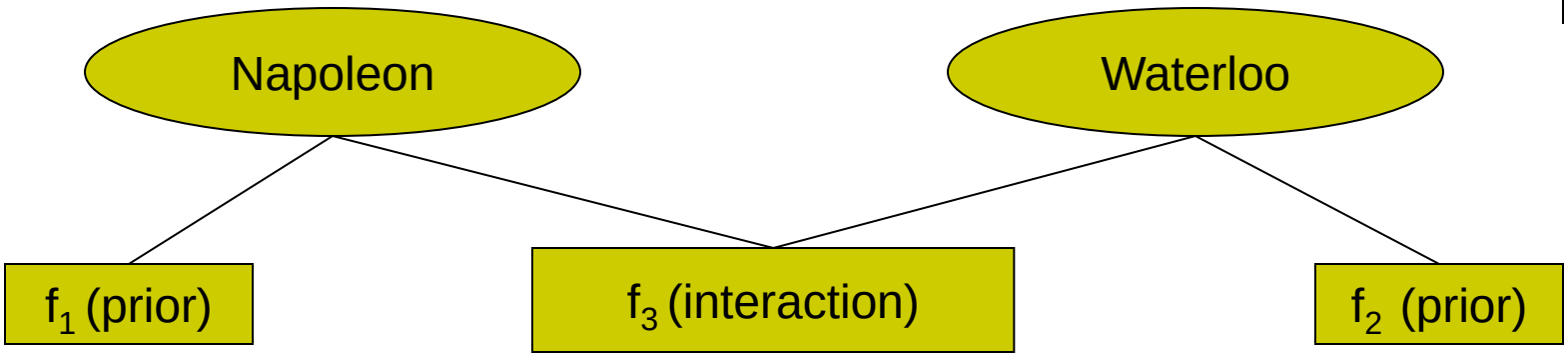
Similarly, Waterloo can either be the Canadian city or the Belgian city, with

f_2(W, Can) = 0.9 f_2(W, Bel) = 0.1

The interaction is defined as

f_3(Nap I, Waterloo Belgium) = 100 f_3(other) = 1

Napoleon Example Continued



Now calculation shows the MAP estimate is

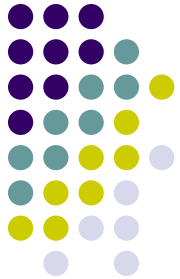
Napoleon = Napoleon I, and Waterloo = Waterloo, Belgium

Furthermore, the posterior distribution is

f_1(Nap I) = 0.999 f_1(Nap II) = 0.001

f_2(W, Can) = 0.09 f_2(W, Bel) = 0.91

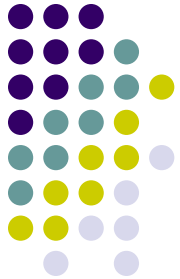
Computing MAP Estimate



- In general, computing the exact MAP estimate of a factor graph is a hard problem, unless the structure of the graph is very special.

- Recent research shows that the *belief propagation* (BP) algorithm is very efficient in finding approximate MAP estimate.

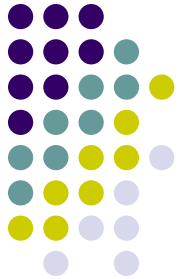
Challenge of the Model



- What interactions do we compute?
  - To get started, we can add a function node for each pair of variable node.
- How do we define the strength of an interaction?
  - We can use the Wikipedia graph. Use the graph distance as a measure of closeness.

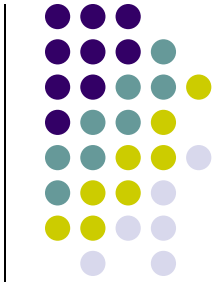
We have a large freedom when choosing these functions. Once they are chosen, the model is fixed.

Why is this Model Awesome?



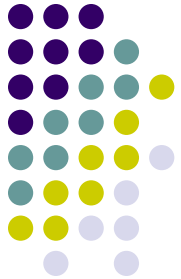
- It captures the interaction of all proper names appeared in the text. Essentially it is computing the context implicitly.
- There is a large degree of freedom in defining the compatibility.
- The BP algorithm enables fast calculation of near-optimal MAP estimate.

Conclusion



- We have proposed a flexible framework for disambiguation of proper names.
- The framework is based on the idea of choosing compatible entities for the extracted names.
- Eventually it boils down to solving an optimization problem using cutting-edge algorithms.

What to Do Next?



- Implement this framework!
- Try various compatibility measures and study their performances.
- Test it on different collections and validate the results.