Graph Similarity Scoring
Applied to
Abstract Meaning Representation

Justin DeBenedetto
AMRs are a semantic formalism which models sentences.
Abstract Meaning Representation (AMR)

- AMRs are a semantic formalism which models sentences
  - Nodes represent concepts
  - Edges represent relations between concepts
    - Semantic roles
    - ARG0 = Agent
    - ARG1 = Patient
    - Example AMR for sentence: “John wants Mary to believe him.”
Properties of AMRS as Graphs

● Some properties of AMRs
  ○ Directed Acyclic Graphs (DAGs)
  ○ Single rooted (focus of sentence)
  ○ Each AMR represents a sentence
Dataset

- Set of 10,312 AMRs from various news sources
- Average number of nodes is: 17.1
- Average number of edges is: 17.1
- More than half are trees
Dataset

AMR Node Counts

AMR Edge Counts
Kernel: Graph Similarity Scoring

• Use some AMRs for training
  – Given multiple candidate AMRs, choose best one
  – Need a way to score each choice
  – Want pairwise digraph similarity score
• Typical metric used for AMRs is SMATCH
SMATCH Score

- Semantic Match score
  - Find best matching of nodes
  - Score based on node and edge labels
  - F1 score
    - Node label
    - For each edge: edge type and end points
**Algorithm 1 Basic SMATCH pseudocode**

1: procedure GETSMATCH(A,B)
2:     maxF1 ← 0
3:     for mapping in nodeMapping(a,b) do
4:         correct ← 0
5:         for alignedPair in mapping do
6:             if labels match then
7:                 correct ← correct + 1
8:         for edges in a do
9:             replace end-points with aligned nodes from b
10:            if new edge exists in b then
11:               correct ← correct + 1
12:               precisionDenominator ← number of triples in b
13:               recallDenominator ← number of triples in a
14:               precision ← correct/precisionDenominator
15:               recall ← correct/recallDenominator
16:               f1 ← (recall + precision)/2
17:               if f1 > maxF1 then
18:                   maxF1 ← f1
19: return maxF1

20: procedure NODEMAPPING(A,B)
21:     allAlignments ← empty
22:     Select node_a in a
23:     for node_b in b do
24:         newAlignments ← align node_a to node_b
25:         newA ← a - node_a
26:         newB ← b - node_b
27:         newAlignments ← nodeMapping(newA, newB)
28:         append newAlignments to allAlignments
29: return allAlignments
Complexity

• Most direct way (previous slide) has complexity $\sim O(N!/(N-M)!*|M+E|)$
  – $N =$ number of nodes in larger graph
  – $M =$ number of nodes in smaller graph
  – $E =$ number of edges in smaller graph
• In practice, heuristics are used
  – Faster, but no optimality guarantee
  – I want to avoid heuristics, and parallelize instead
Implementation

• Python using networkX
• Just under 100 new lines (including some debugging lines)
• Highly recursive
  – Match node pair, match remaining subgraphs
  – Memory problems as problem size increases
Timing Results

SMATCH Time with 4 node AMR

\[ y = 0.0051x^3 - 0.1463x^2 + 1.2341x - 2.4207 \]

\[ R^2 = 0.9982 \]
Timing Results

The graph shows the timing results for SMATCH Time with 4 node AMR. The equation for the curve is:

\[ y = 0.0088x^3 - 0.3248x^2 + 3.5285x - 8.7677 \]

with an \( R^2 = 0.9977 \).
Timing Results

![Graph showing SMATCH Time with a=b]

- Time (s) on the Y-axis
- Number of Nodes on the X-axis
- Two data points marked: Observed and Factorial
Other Results

- Memory consumption is high
  - At graph sizes of 11 nodes, 10 edges each memory consumption approaches 20GB
  - Memory scales similar to runtime
- SMATCH score returned is correct (optimal)
  - In some cases this is better than popular heuristic
  - Will compare against heuristic more with enhanced algorithm
Plans for Improvements

- Combine mapping and scoring
  - Score nodes as they are matched
  - Avoids recomputing
- Send subgraphs to worker machines for parallelism
- Score likely alignments first, use as cutoff
  - Denominator does not change (N+E)
  - Can avoid unnecessary computation
Try SNAP

• Interface looks very similar to networkX
• They claim it is an order of magnitude faster