1 Dependency Parsing using MTT

Definition

Construct dependency tree (DT) with syntactic relations, e.g., noun-modifier, determiner, etc. The additional root node should only have degree 1.

Relation to context free grammar

CRF has no information about syntactic relation, DT has no information about constituency structure.

Types

(1) Projective DT: no crossing arcs. Algorithms are generally dynamic programming. (2) Non-Projective DT: crossing arcs. Algorithms use matrix-tree theorem (MTT).

Edge-Factored Assumption targeted the difficulty

Assume $\mathrm{score}(t, w)$ is the product of edge scores (including the root edge).
Define $A_{ij} = \exp(\mathrm{score}(i, j, w))$ be the edge score, $\rho_j = \exp(\mathrm{score}(j, w))$ be the root score. Then MTT says $Z = |L|$ for following L which is $O(n^3)$:

$$L_{ij} = \begin{cases} \rho_j & \text{if } i = 1 \\ \sum_{i'=1, i' \neq j}^{n} A_{i'j} & \text{if } i = j \\ -A_{ij} & \text{otherwise} \end{cases}$$

Decoding the best DT

Equivalent to find the best directed spanning tree starting from root and the degree of root is 1. Greedy algorithms that work in undirected graph do not work.
We apply Chu-Liu-Edmonds Algorithm which is $O(n^3)$: (1) Find the best *incoming* edge for each vertex. (2) Contract cycles to be a single node c and reweight the *incoming* edge to c by adding the valid weights in c if the edge is chosen. (3) The graph now has a spanning tree. If the root constraint is not satisfied, reweight each *outcoming* edge from root to v by subtracting the weight of next best *incoming* edge to v. Remove the lowest *outcoming* edge from root and repeat step 3. (4) Expand contract nodes by breaking cycles accordingly.

2 Semantic Parsing with CCG

Definition

Parse of meaning. Represented by logical form composed of variables, predicates, quantifiers and boolean. Example: $\forall p.(Person(p) \rightarrow \exists q.(Person(q) \wedge p \neq q \wedge \mathrm{Loves}(p, q)))$.

Principle of Compositionality

The meaning of a complex expression is a function of the meanings of that expression's constituent parts.

Enriched lambda calculus to represent meanings

(1) Logical constants such as entities (e.g., ALEX) and relations (e.g., likes). (2) Variables, which are undetermined objects, similar to free variables in lambda calculus. (3) Literals such as LIKES(ALEX, x), formed by applying relations to objects. (4) Logical terms built using literals with logical connectives (e.g., $\wedge$) and quantifiers (e.g., $\exists$). (5) lambda terms built using lambda operator.