Maximum Entropy and Context Free Grammars

Recall that using a probabilistic CFG:

- The probability of a parse is the product of all rule probabilities in that parse:

$$P(\text{parse}) = \prod_A P(A \rightarrow \alpha)$$

- The probability of a sentence is the sum of all parse probabilities for that sentence:

$$P(\text{sentence}) = \sum P(\text{parse})$$

- Rule probabilities are usually estimated by counting:

$$P(A \rightarrow \alpha) = \frac{\text{freq}(A \rightarrow \alpha)}{\sum_\beta \text{freq}(A \rightarrow \beta)}$$

Probabilistic Context Free Grammars

When we want to pick the best parse:

- We work-out the overall parse probability for each tree.
- We sort the trees by the parse probability.
- The best parse is then the one with the highest parse probability.
- (This can be efficiently done using dynamic programming).

Problems with PCFGs

There are problems with this approach:

- We are biased towards trees with *fewer* rule applications.
- It is hard to model long-range dependencies.

What we want is a modelling approach which does not *generate* a tree one step at a time.

An Alternative

Suppose we have two parse trees for some sentence:

- Parse one: using CFG rules R_1, R_2 and R_3.
- Parse two: using CFG rules $R_1, and R_4$.

And parse one is the preferred analysis of some sentence:

$$P(\text{parse one}) >> P(\text{parse two})$$

An Alternative

In general:

- Associate a non-negative number with each parse P_i.
- We can turn these numbers into probabilities:

Parse	Total Weight	Probability
one	9	9/(9 + 1)
two	1	1/(9 + 1)

Now parse one has a higher probability than parse two.

An Alternative

How do we compute these total weights?

- Model each parse as a set of *features* f_i.
- Associate with each feature an individual *weight* λ_i.
- Gather together all weighted features:

$$\text{total weight} = (f_1 \cdot \lambda_1) + (f_2 \cdot \lambda_2) + \ldots$$

- To make everything positive we *exponentiate*:

$$\text{total weight} = \exp((f_1 \cdot \lambda_1) + (f_2 \cdot \lambda_2) + \ldots)$$

Maximum Entropy

- The probability of a parse is now:

$$\text{weight(parse one)} = \exp(\sum_i f_i \lambda_i)$$

$$P(\text{parse one}) = \frac{\text{weight(parse one)}}{\text{weight(parse one) + weight(parse two)}}$$

- This is a Maximum Entropy (log-linear, maxent) model.

Features

- Features are ‘questions’ about a data point.
- Questions are mapped to numbers.
- Example:
  - How many times is *threw* a head word?
  - How many left-branching trees are there?
  - How many R_4 rules does a parse contain ?
  - Etc.
- Features are usually selected by the system designer.

Features

Another example:

$$f_{\mathrm{j}} = \begin{cases} 1 & \text{Parse contains rule } VP \rightarrow V\ NP\ NP \\ 0 & \text{Otherwise} \end{cases}$$

Features

Another example:

$$f_{\mathrm{k}} = \begin{cases} n & \text{The number of rule applications in a parse} \end{cases}$$

Features

An example:

$$f_{\mathrm{i}} = \begin{cases} 1 & \text{Parse contains rule } S \rightarrow NP\ VP \\ 0 & \text{Otherwise} \end{cases}$$

Features

Another example:

$$f_{\mathrm{k}} = \begin{cases} 1 & \text{Parse contains rule } VP \rightarrow V\ NP\ NP \\ & \text{and the head word of the verb is } \textit{gave} \\ 0 & \text{Otherwise} \end{cases}$$

A little example

Returning to our original problem:

- Parse one has rules: R_1, R_2 and R_3.
- Parse two has rules: R_1, and R_4.
- We wish to model $P(\text{parse one}) >> P(\text{parse two})$.
- Have a feature which counts the number of times a rule is seen in a parse:

Parse	Features	Overall Probability
One	f_1, f_2, f_3	9/10
Two	$f_1 f_4$	1/10

A little example

And the parse probabilities:

Parse	Total Weight	Exp	Prob
One	(1 * 1.25e-16) + (1 * 0.73) + (1 * 0.73)	4.32	0.9
Two	(1 * 1.25e-16) + (1 * -0.73)	0.48	0.1

Weights

Each feature has an associated *weight*.

- Weights are real-valued numbers (plus or minus).
- Weights informally balance the contribution each feature makes:
  - A weight of zero means the feature has no effect.
  - A positive weight makes the overall probability higher.
  - A negative weight makes the probability lower.
- There are no independence assumptions between features.

A little example

The feature weights are:

Feature	Weight
f_1	1.25153e-16
f_2	0.732133
f_3	0.732133
f_4	-0.732133

Training and Modelling

Weights are set using numerical optimisation:

- Select weights which satisfy all the constraints.
- The general problem is convex and so efficient hill-climbing methods can be used.

The best parse usually has a probability of one and the other competing parses a zero probability:

- The model now *discriminates* between the best parse and the competing parses.
- Discriminative parse selection models ignore the sentence probability.

Comments

Log-linear models are widely used in Computational Linguistics:

- For parsing, maxent models produce state-of-the-art performance.
- They also form the basis for the best sequencing models (*Conditional Random Fields*).
- . . . and also for Statistical Machine Translation.

Further Reading

An excellent introduction to log-linear models for parsing is:

- Steven P. Abney. *Stochastic Attribute-Value Grammars*. http://citeseer.ist.psu.edu/490897.html