ACS Syntax and Semantics of Natural Language

Lecture 8: Statistical Parsing Models for CCG

Stephen Clark

Natural Language and Information Processing (NLIP) Group

sc609@cam.ac.uk

- The parsing models that have been developed for CCG are not CCG-specific, but general models for structural prediction problems
- Generative models can be applied to CCG derivations, generating the tree top-down using a similar process to the Collins model (Hockenmaier, 2003)
- Feature-based models have been successfully applied to CCG, including log-linear models and the generalized perceptron

Feature-Based Models for CCG

$$d\_\mathsf{max}_{\Phi,\Lambda}(S) = \arg\max_d \mathsf{Score}(\Phi(d,S),\Lambda)$$

- $d_$max is the highest scoring derivation for sentence S, according to a feature representation function Φ and a model (set of weights) Λ
- The algorithm which implements the $\arg\max$ is the decoder - let's worry about that later, but bearing in mind that how we choose to define Φ is likely to impact on the efficiency of the decoder
- $\Phi(d,S)$ takes a derivation d for sentence S and returns the features for that derivation - how we choose to break d into features will be crucial for accuracy
- Λ is the set of weights corresponding to the complete set of features - Λ will be learned from annotated data (CCGbank)
- The Score function relates Φ and Λ and assigns a real-valued score to a derivation - we'll be using log-linear and linear formulations

The Feature Representation

- Features will be defined *locally* in terms of rule instantiations
  - by *rule instantiation* I just mean the subtree consisting of a parent and children (one or two children in CCG's case)
- We could extend the feature range, and this may increase the discriminatory power of the model, but the efficiency of the decoding and estimation algorithms will decrease
- Mathematically, features are functions from derivations onto integers (i.e. counts)
  - so extensions of the binary indicator functions we saw for tagging

Feature type	Example
LexCat + Word	$(S/S)/NP$ + Before
LexCat + POS	$(S/S)/NP$ + IN
RootCat	$S[dcl]$
RootCat + Word	$S[dcl]$ + was
RootCat + POS	$S[dcl]$ + VBD
Rule	$S[dcl]\rightarrow NP\ S[dcl]\backslash NP$
Rule + Word	$S[dcl]\rightarrow NP\ S[dcl]\backslash NP$ + bought
Rule + POS	$S[dcl]\rightarrow NP\ S[dcl]\backslash NP$ + VBD
Word-Word	$\langle$*company*, $S[dcl]\rightarrow NP\ S[dcl]\backslash NP$, *bought*$\rangle$
Word-POS	$\langle$*company*, $S[dcl]\rightarrow NP\ S[dcl]\backslash NP$, VBD$\rangle$
POS-Word	$\langle$NN, $S[dcl]\rightarrow NP\ S[dcl]\backslash NP$, *bought*$\rangle$
POS-POS	$\langle$NN, $S[dcl]\rightarrow NP\ S[dcl]\backslash NP$, VBD$\rangle$
Word + Distance(words)	$\langle$*bought*, $S[dcl]\rightarrow NP\ S[dcl]\backslash NP\rangle$ + >2
Word + Distance(punct)	$\langle$*bought*, $S[dcl]\rightarrow NP\ S[dcl]\backslash NP\rangle$ + 2
Word + Distance(verbs)	$\langle$*bought*, $S[dcl]\rightarrow NP\ S[dcl]\backslash NP\rangle$ + 0
POS + Distance(words)	$\langle$VBD, $S[dcl]\rightarrow NP\ S[dcl]\backslash NP\rangle$ + >2
POS + Distance(punct)	$\langle$VBD, $S[dcl]\rightarrow NP\ S[dcl]\backslash NP\rangle$ + 2
POS + Distance(verbs)	$\langle$VBD, $S[dcl]\rightarrow NP\ S[dcl]\backslash NP\rangle$ + 0

- This basic feature set results in a few hundred thousand features and leads to good parsing performance

Log-Linear Model

$$P(d|S) = \text{exp}(\sum_i \lambda_i f_i(d, S))/Z(S)$$

where d is a derivation for sentence S and $Z(S)$ is a normalisation constant

- Note that this is a *global* model, assigning a probability to the complete parse tree
- Can also be thought of as a maximum entropy model, as for the local maxent models we used for tagging
- To esimate the model parameters (weights, λ's), we need to calculate feature expectations, as before
- Calculating feature expectations requires a sum over *all* derivations for each sentence in the training data
  - sum can be performed efficiently using a dynamic programming algorithm (the inside-outside algorithm) over a packed chart

Decoding

- We can build a *packed chart* for CCG as we did for a PCFG in the Intro to NLP module
- Categories with the same CCG type, same span in the sentence, and same headword, are grouped together into an equivalence class
  - definition of equivalence depends on the feature range
- The Viterbi algorithm runs recursively top-down over the chart, choosing the highest scoring category in each equivalence class
- The Viterbi algorithm is optimal and operates in polynomial time (with respect to sentence length)

Linear Model with Perceptron Training

$$\text{Score}(d, S) = \sum_i \lambda_i f_i(d, S)$$

- An alternative to the log-linear model which is trained using a simple parameter update which aims to maximise accruacy on the training data
- Performs surprisingly well given simple nature of the update
- Also has some theoretical guarantees
- See Collins (2002) for application to tagging

Perceptron Training

$$\text{Score}(d, s) = \sum_i \lambda_i . f_i(d, s) = \overline{\lambda} \cdot \overline{f}(d)$$

Inputs: training examples (s_j, d_j)
Initialisation: set $\overline{\lambda} = 0$
Algorithm:
for $t = 1..T$, $j = 1..N$
calculate $d_{\text{max}} = \arg\max_d \overline{\lambda} \cdot \overline{f}(d)$
if $d_{\text{max}} \neq d_j$
$\overline{\lambda} = \overline{\lambda} + \overline{f}(d_j) - \overline{f}(d_{\text{max}})$
Outputs: $\overline{\lambda}$

[see JHU tutorial slides for animated description of online update]

References

- Stephen Clark and James R. Curran. Perceptron Training for a Wide-Coverage Lexicalized-Grammar Parser. Proceedings of the ACL-07 Workshop on Deep Linguistic Processing, Prague, Czech Republic, 2007
- Julia Hockenmaier. Data and models for statistical parsing with Combinatory Categorial Grammar. Edinburgh PhD thesis. 2003
- Michael Collins. Discriminative Training Methods for Hidden Markov Models: Theory and Experiments with Perceptron Algorithms. EMNLP 2002.