CRF-based Supertagging

Timothy Baldwin

THE UNIVERSITY OF MELBOURNE

Supertagging

- Supertagging = POS tagging with a very fine-grained tagset

$$How_{[\texttt{WRB}]}\ does_{[\texttt{VBZ}]}\ that_{[\texttt{DT}]}\ sound_{[\texttt{VB}]}\ for_{[\texttt{IN}]}\ you_{[\texttt{PRP}]}\ ?_{[.]}$$

$$\Downarrow$$

$$How_{[\texttt{adj\_wh\_le}]}\quad does_{[\texttt{va\_does\_le}]}\quad that_{[\texttt{n\_-\_pr-dei-sg\_le}]}$$
$$sound_{[\texttt{v\_pp-pp\_seq\_le}]}\ for_{[\texttt{p\_le}]}\ you_{[\texttt{n\_-\_pr-you\_le}]}\ ?_{[.]}$$

Applications of Supertagging

- On-the-fly deep lexical acquisition
- Means of pruning parser search space (cf. Bangalore and Joshi, 1999; Clark and Curran, 2004)

  blurring the distinction between in/out of vocabulary
- Source of linguistic features (e.g. for word alignment)

Supertagging: A Shopping List

- We desire a method that:
  - ⋆ works across different languages with a minimum of fuss
  - ⋆ can be trained directly from treebank data
  - ⋆ scales to a large tagset (1000s of tags)
  - ⋆ achieves state-of-the-art accuracy
  - ⋆ is probabilistic

Proposed Supertagger Model

- Pseudo-likelihood CRF model, where $p_\Lambda(\mathbf{a}|\mathbf{s})$ is approximated by p_Λ^{PL}:

$$p_\Lambda^{PL}(\mathbf{a}|\mathbf{s}) = \prod_t \frac{\exp(U_\Lambda^{PL}(\mathbf{a}_t, \mathbf{s}, t))}{\sum_l (U_\Lambda^{PL}(l, \mathbf{s}, t))}$$

$$U_\Lambda^{PL}(i, \mathbf{s}, t) = \sum_k \lambda_k (h_k(t, \hat{a}_{t-1}, i, \mathbf{s}) + h_k(t, i, \hat{a}_{t+1}, \mathbf{s}))$$

- Smooth with a zero-mean Gaussian prior

- Calculate the most probable labelling $\mathbf{a}^*$ for a test sentence via Viterbi as:

  $$\mathbf{a}^* = \arg\max_{\mathbf{a}} p_\Lambda(\mathbf{a}|\mathbf{s})$$

Implementation Details

- Coded in C++, with Fortran libraries and Python bindings; MPI enabled
- Primary development under Linux; DEB packageable
- Licencing details still up in air (to be finalised in coming days/weeks)
- Expect to make code available via Google Code or Sourceforge (linked in from LOGON SVN?)

Getting it Running

- Scripts used to extract out CoNLL-style data from the gold files, which the supertagger is trained over (words and lemmas):

```
Please  adv_disc_please_le    please  adv_disc_please_le
send    v_np-np_le            send    v_np-np_le
me      n_-_pr-me_le          me      n_-_pr-me_le
status  n_pp_mc-of_le         status  n_pp_mc-of_le
of      p_prtcl_of_le         of      p_prtcl_of_le
all     det_part_pl_mass_le   all     det_part_pl_mass_le
items   n_pp_c-ns-of_le       item    n_pp_c-ns-of_le
```

EXPERIMENTS

Relevant Statistics of Target Grammars

	ERG	JACY
GRAMMAR		
Language	English	Japanese
Lexemes	16,498	41,559
Lexical items	26,297	47,997
Lexical types	915	484
Strictly continuous MWEs	2,581	422
Optionally discontinuous MWEs	699	0
Average lexical items per lexeme	1.59	1.16
TREEBANK		
Training sentences	20,000	40,000
Training words	215,015	393,668
Test sentences	1,013	1,095
Test words	10,781	10,669

Features

- Features currently based on a combination of **word context** and (existence-based) **lexical** features

- Lexical features based on n-gram prefixes & suffixes, and basic character sets in the given language
  - ⋆ English = 5 character sets (upper case, lower case, numbers, punctuation and hyphens)
  - ⋆ Japanese = 6 character sets (Roman letters, hiragana, katakana, kanji, (Arabic) numerals and punctuation)

Feature Types

FEATURE	DESCRIPTION
WORD CONTEXT FEATURES	
$\mathit{lexeme}(\mathbf{s}_t) = x \;\&\; \mathbf{a}_t = l$	lexeme + label
$\mathbf{s}_t = w \;\&\; \mathbf{a}_t = l$	word unigram + label
$\mathbf{s}_{t-1} = w \;\&\; \mathbf{a}_t = l$	previous word unigram + label
$\mathbf{s}_{t+1} = w \;\&\; \mathbf{a}_t = l$	next word unigram + label
$\mathbf{s}_t = w \;\&\; \mathbf{s}_{t-1} = y \;\&\; \mathbf{a}_t = l$	previous word bigram + label
$\mathbf{s}_t = w \;\&\; \mathbf{s}_{t+1} = y \;\&\; \mathbf{a}_t = l$	next word bigram + label
$\mathbf{a}_{t-1} = l \;\&\; \mathbf{a}_t = m$	clique label pair
LEXICAL FEATURES	
$\mathit{prefix}_n(\mathbf{s}_t) \;\&\; \mathbf{a}_t = l$	n-gram prefix + label
$\mathit{suffix}_n(\mathbf{s}_t) = x \;\&\; \mathbf{a}_t = l$	n-gram suffix + label
$\mathit{contains}(\mathbf{s}_t, C_i) \;\&\; \mathbf{a}_t = l$	word contains element of character set C_i + label

Experimental Setup

- Train supertagger over Redwoods (EN) and Hinoki (JP) treebank data
- Evaluate relative to a held-out set of $\sim$1000 sentences
- Baseline $=$ unigram supertagger

Experiments

- **Experiment 1:** how effectively are the models able to learn novel lexical items (token vs. type)?
- **Experiment 2:** how effective are the models at reducing the parse search space?

Experiments

- **Experiment 1:** how effectively are the models able to learn novel lexical items (token vs. type)?

- **Experiment 2:** how effective are the models at reducing the parse search space?

Experiment 1: Evaluation Metrics

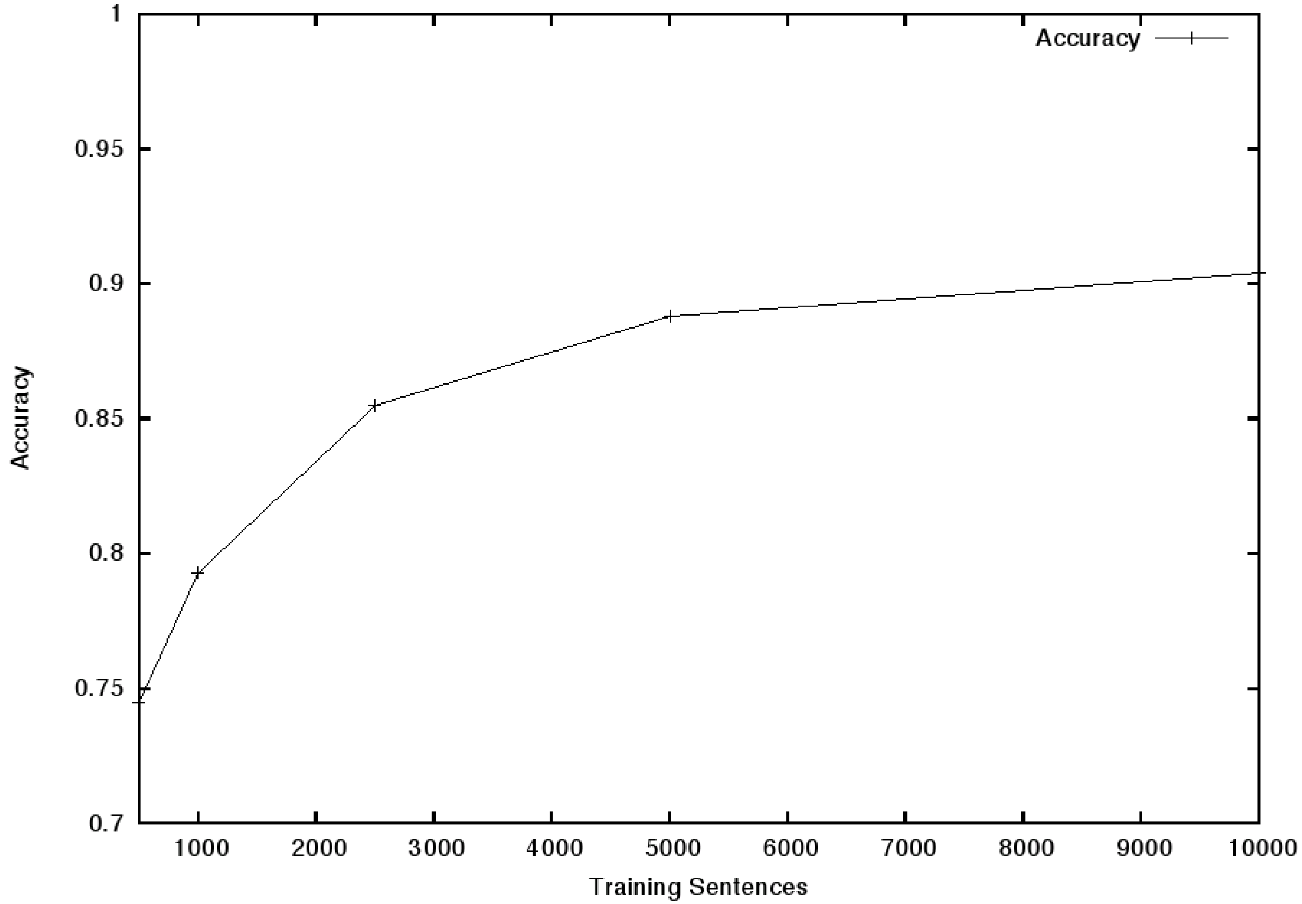
- **Token accuracy** [$\textsc{Acc}$] = overall token-level accuracy
- **Unknown token accuracy** [$\textsc{Acc}_U$] = token-level accuracy over unknown lexemes
- **Type precision** [$\textsc{Prec}$] = % correct unknown LEs
- **Type recall** [$\textsc{Rec}$] = % unknown gold-standard LEs correctly predicted
- **Type F-score** [$\textsc{F-score}$]

Results for ERG

	ACC	ACC$_U$	PREC	REC	F-SCORE
Baseline	0.806	0.227	0.190	0.219	0.203
CRF$_{-\mathrm{LEX}}$	0.908	0.338	0.226	0.340	0.271
CRF$_{+\mathrm{LEX}}$	**0.904**	**0.447**	**0.302**	**0.448**	**0.361**

CRF$_{\pm\mathrm{LEX}}$= CRF with/without lexical features

Scalability of Results: ERG

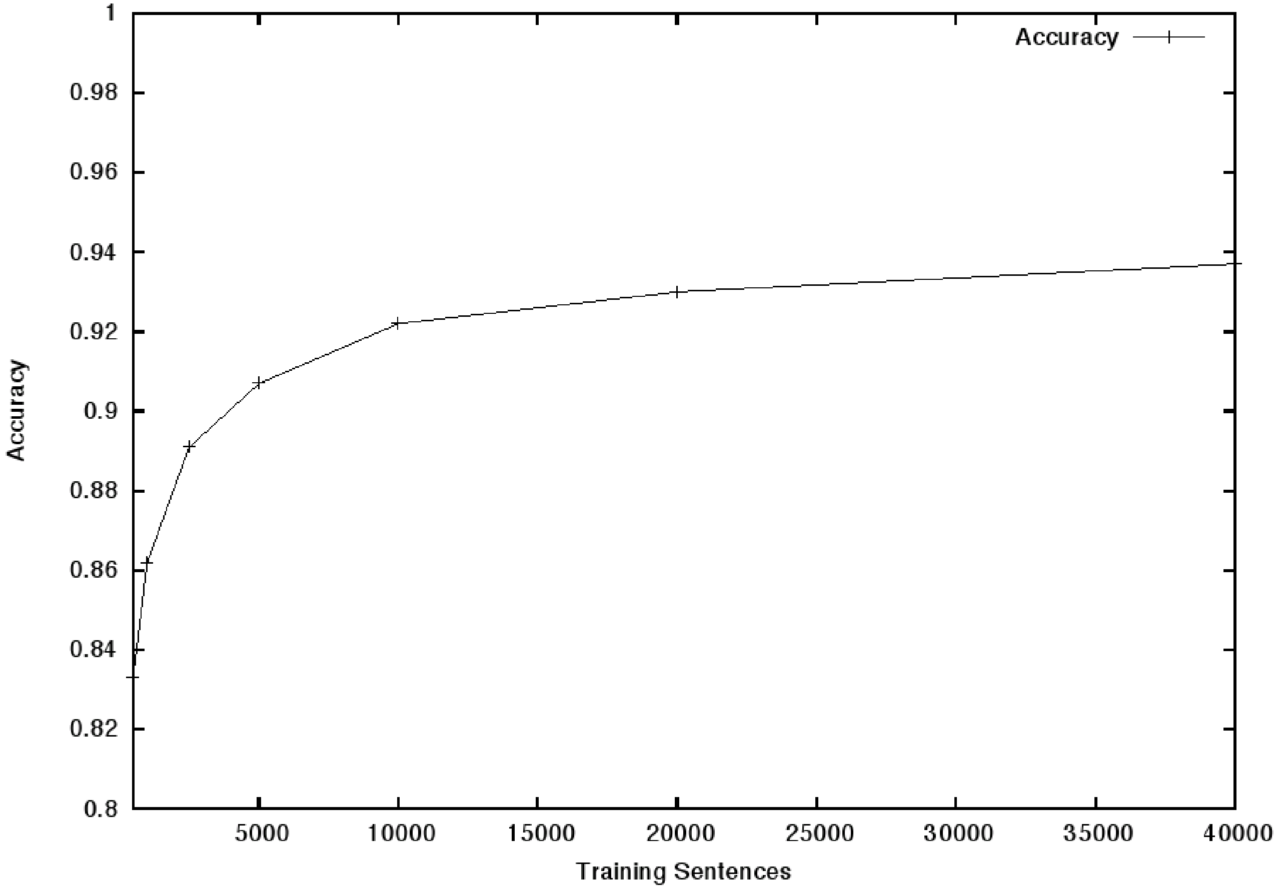


Results for JACY

	ACC	ACC$_U$	PREC	REC	F-SCORE
Baseline	0.865	0.646	0.559	0.643	0.598
CRF$_{-\mathrm{LEX}}$	0.922	0.857	0.515	0.857	0.643
CRF$_{+\mathrm{LEX}}$	**0.937**	**0.874**	**0.712**	**0.874**	**0.785**

CRF$_{\pm\mathrm{LEX}}$= CRF with/without lexical features

Scalability of Results: JACY



Experiment 1: Reflections

- Token accuracy very high (incl. MWEs)
- Type precision of unknown LEs also respectably high (> 0.50), suggesting possibilities of (semi-)automating DLA
- Type recall highly variable (esp. low for EN)
- Remarkably good results given lack of feature engineering

Experiments

- **Experiment 1:** how effectively are the models able to learn novel lexical items (token vs. type)?

- **Experiment 2:** how effective are the models at reducing the parse search space?

Experiment 2: Methodology

1. For each token, calculate the marginal probabilities for all lexical types

2. Determine the most probable lexical type y_i^*, and constrain the set of lexical type hypotheses by thresholding (threshold $= \beta$) over the marginal probabilities, relative to $p_\Lambda(y_i^*)$

Experiment 2: Evaluation Metrics

- **Average categories** [$\textsc{Cats}$] = average lexical types per lexeme
- **token accuracy** [$\textsc{Acc}$] = % lexemes for which the correct lexical type is predicted
- **sentence accuracy** [$\textsc{Acc}_S$] = % sentences for which the correct lexical type is predicted for all lexemes

Experiment 2: Results for ERG

	Baseline			CRF$_{+\mathrm{LEX}}$		
β	CATS	ACC	ACC$_S$	CATS	ACC	ACC$_S$
1.0	1.00	0.809	0.257	1.00	0.904	0.499
0.5	1.32	0.877	0.391	1.10	0.932	0.595
0.1	2.45	0.956	0.670	1.54	0.972	0.780
0.05	3.30	0.969	0.739	1.94	0.982	0.846
0.01	7.52	0.985	0.863	4.13	0.993	0.935
0.005	10.25	0.988	0.887	6.34	0.995	0.956
0.001	16.14	0.990	0.907	20.38	0.998	0.979

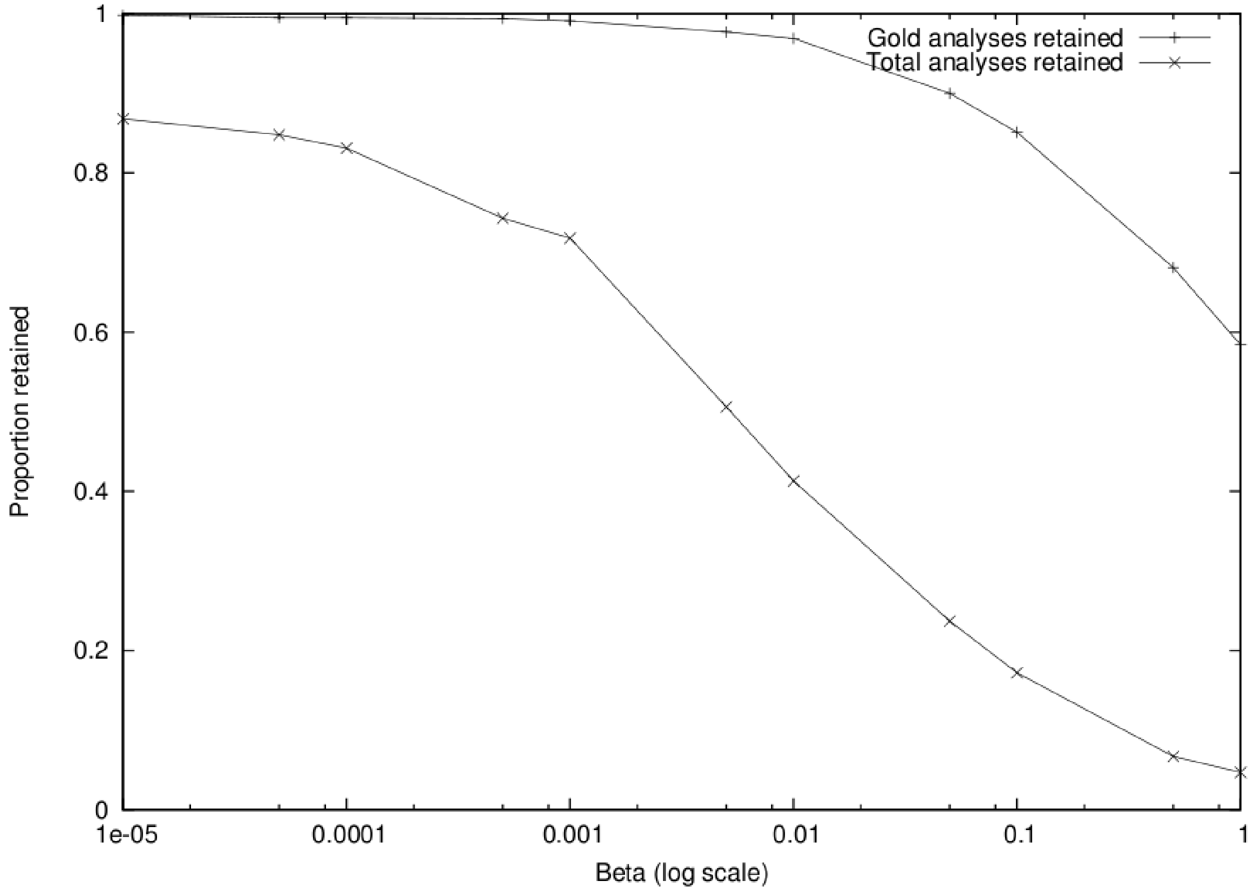
Experiment 2: Results for JACY

	Baseline			$\mathrm{CRF}_{+\mathrm{LEX}}$		
β	Cats	Acc	Acc_S	Cats	Acc	Acc_S
1.0	1.00	0.867	0.304	1.00	0.937	0.597
0.5	1.07	0.884	0.343	1.06	0.962	0.742
0.1	1.82	0.965	0.722	1.21	0.991	0.926
0.05	2.26	0.977	0.815	1.31	0.995	0.960
0.01	3.89	0.994	0.942	1.74	0.998	0.984
0.005	4.95	0.995	0.956	2.10	0.999	0.991
0.001	7.69	0.997	0.968	3.83	1.000	0.996

Experiment 2: Parse Pruning with ERG (1)

β	Sent	Analyses/sent	Coverage	Sec/sent	Passive edges/sent
1.0	729	9.48	0.73	0.02	161.7
0.5	729	10.86	0.81	0.03	181.1
0.1	729	39.66	0.91	0.04	254.2
0.05	729	114.18	0.93	0.05	311.3
0.01	729	3733.76	0.97	0.17	687.9
ALL	694	70.06	0.79	0.11	555.8

Experiment 2: Parse Pruning with ERG (2)



Experiment 2: Reflections

- Extremely high token accuracies achieved with only a small number of lexical types per word (esp. JACY)

  = possible to dramatically reduce the parse search space while preserving the gold-standard parse

- Relative loss in sentence accuracy slight compared to parse selection accuracy for JACY and ERG ($\sim$0.50 and 0.80, resp.)

CONCLUSION

Conclusion

- New method for learning lexical items for HPSG-based precision grammars through supertagging, using a pseudo-likelihood CRF
- State-of-the-art results achieved for English and Japanese with language-independent feature set
- Illustration of the ability of the proposed model to reduce the parse search space
- New toolkit to play around with

AND NOW FOR SOMETHING COMPLETELY DIFFERENT

Online Linguistic Exploration: Deeper, Faster, Broader Language Documentation

Aim: develop a real-time “language analysis” environment which:

- identifies both positive and (near-miss) negative instances that arise as a result of the analysis
- facilitates rapid resource development
- provides an active annotation interface
- presents the linguist with related analyses in other LRs

- Efficient indexing and expressive querying of treebanks ...
- Automatic corpus construction ...
- Multilingual lexical acquisition ...
- Unsupervised and semi-supervised parse (re)ranking ...
- Error mining of grammars/parse forests ...

Immediate Objectives

- Index treebanks and exhaustive parse forest for different datasets, different grammars (existing datasets, as well as random web data, etc.)

- Support various query types both monolingually and crosslingually
  - ⋆ derivation trees vs. AVMs
  - ⋆ natively vs. via GOLD etc
  - ⋆ LPATH vs. ???

Questions for This Audience

- Do you commonly query DELPH-IN resources, and if so, which and in what way?
- What sorts of queries do you most commonly perform over DELPH-IN treebanks/parse forests (e.g. via tsql)?
- Are there particular treebank query types you would like to perform which aren't (well) supported in the existing machinery?
- What (if any) sorts of things do you most commonly

search for in the grammar files?

- Are there particular **grammar** query types you would like to perform which aren't supported in the existing machinery?

- Would you be willing to help out: (a) using the service, and (b) providing results for given queries (annotation) [say yes!]

References

Bangalore, S. and Joshi, A. K. (1999). Supertagging: An approach to almost parsing. *Computational Linguistics*, 25(2):237–65.

Blunsom, P. and Baldwin, T. (2006). Multilingual deep lexical acquisition for HPSGs via supertagging. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing (EMNLP 2006)*, pages 164–71, Sydney, Australia.

Clark, S. and Curran, J. R. (2004). The importance of supertagging for wide-coverage CCG parsing. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING 2004)*, pages 282–8, Geneva, Switzerland.