Übertagging

Rebecca Dridan

University of Oslo

DELPH-IN Summit
St Wendel, July 2013

First there was supertagging

- Assigning fine-grained lexical categories to tokens for:
  - lexical acquition, unknown word handling
  - parser efficiency, via lexical pruning
- Supertagging/lexical prediction for the ERG:
  - Zhang, 2007
  - Blunsom, 2007
  - Dridan, 2009
  - Ytrestøl, 2012
  - Fares, 2013

All showed potential, none are actually being used in parsing now.

Which *tokens*?

Raw 'Sun-filled', well-kept Mountain View.

REPP

initial tokens ⟨'⟩, ⟨Sun-filled⟩, ⟨'⟩, ⟨,⟩, ⟨well-kept⟩, ⟨Mountain⟩, ⟨View⟩, ⟨.⟩

chart-mapping

internal tokens ⟨'Sun-⟩, ⟨filled',⟩, ⟨well-⟩, ⟨kept⟩, ⟨Mountain⟩, ⟨View.⟩,

lexicon lookup

lexical tokens ⟨'sun-⟩, ⟨filled',⟩, ⟨well- kept⟩, ⟨Mountain View.⟩, ⟨well-⟩, ⟨kept⟩, ⟨Mountain⟩, ⟨View.⟩,

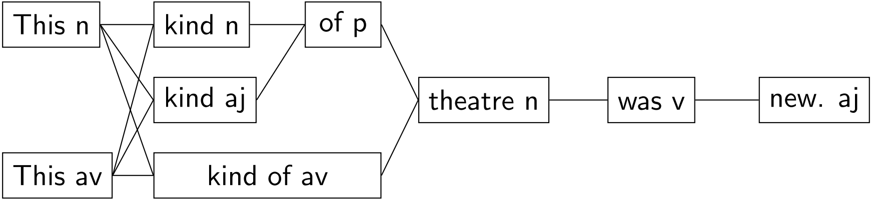
Which *tokens*?

Raw	‘Sun-filled’, well-kept Mountain View.
	REPP
initial tokens	⟨‘⟩, ⟨Sun-filled⟩, ⟨’⟩, ⟨,⟩, ⟨well-kept⟩, ⟨Mountain⟩, ⟨View⟩, ⟨.⟩
	chart-mapping
internal tokens	⟨‘Sun-⟩, ⟨filled’,⟩, ⟨well-⟩, ⟨kept⟩, ⟨Mountain⟩, ⟨View.⟩,
	lexicon lookup
lexical tokens	⟨‘sun-⟩, ⟨filled’,⟩, ⟨well- kept⟩, ⟨Mountain View.⟩, ⟨well-⟩, ⟨kept⟩, ⟨Mountain⟩, ⟨View.⟩,

Lexical tokenisation is *ambiguous*.

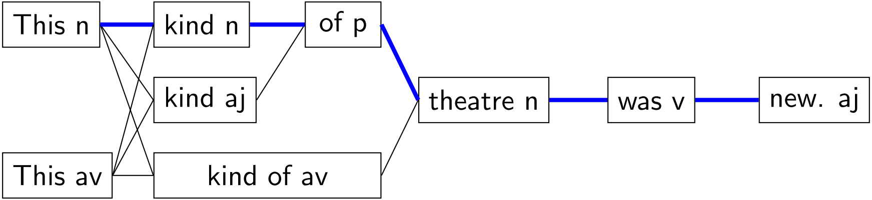
Übertagging

Übertagging predicts the correct path through the lattice, tokenising and supertagging at the same time.



Übertagging

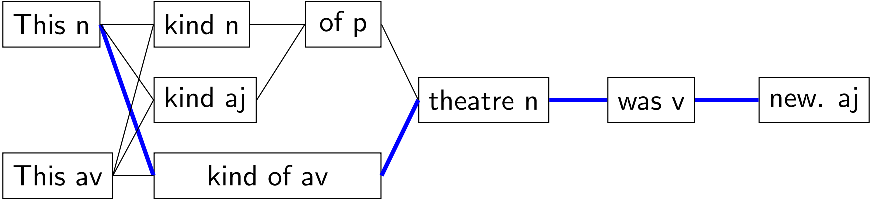
Übertagging predicts the correct path through the lattice, tokenising and supertagging at the same time.



⟨This⟩	⟨kind⟩	⟨of⟩	⟨theatre⟩	⟨was⟩	⟨new. ⟩
n	n	p	n	v	aj

Übertagging

Übertagging predicts the correct path through the lattice, tokenising and supertagging at the same time.



⟨This⟩	⟨kind of⟩	⟨theatre⟩	⟨was⟩	⟨new. ⟩
n	av	n	v	aj

Hidden semi-Markov Models

Standard HMM:

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} Pr(s_i|s_{i-1}) Pr(o_i|s_i) \cdot Pr(\langle E \rangle |s_n)$$

with $s_0 = \langle S \rangle$

Hidden semi-Markov Models

Standard HMM:

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} Pr(s_i|s_{i-1})Pr(o_i|s_i) \cdot Pr(\langle E \rangle \,|s_n)$$

with $s_0 = \langle S \rangle$

In a segmental HMM, we have:

- frames, equivalent to the time slices in a standard HMM; and
- segments, of one or more frames in length

Hidden semi-Markov Models

Standard HMM:

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} Pr(s_i|s_{i-1})Pr(o_i|s_i) \cdot Pr(\langle E \rangle |s_n)$$

$$\text{with } s_0 = \langle S \rangle$$

In a segmental HMM, we have:

- frames, equivalent to the time slices in a standard HMM; and
- segments, of one or more frames in length

States in a segmental HMM have a tag, t and a length, l.

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} Pr(t_i|t_{i-l})Pr(l|t)Pr(o_{i-l+1:i}|t, l)$$

Hidden semi-Markov Models

Standard HMM:

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} \boxed{Pr(s_i|s_{i-1})} Pr(o_i|s_i) \cdot Pr(\langle E \rangle |s_n)$$

$$\text{with } s_0 = \langle S \rangle$$

In a segmental HMM, we have:

- frames, equivalent to the time slices in a standard HMM; and
- segments, of one or more frames in length

States in a segmental HMM have a tag, t and a length, l.

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} \boxed{Pr(t_i|t_{i-l})} Pr(l|t) Pr(o_{i-l+1:i}|t, l)$$

Hidden semi-Markov Models

Standard HMM:

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} Pr(s_i|s_{i-1}) Pr(o_i|s_i) \cdot Pr(\langle E \rangle |s_n)$$

with $s_0 = \langle S \rangle$

In a segmental HMM, we have:

- frames, equivalent to the time slices in a standard HMM; and
- segments, of one or more frames in length

States in a segmental HMM have a tag, t and a length, l.

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} Pr(t_i|t_{i-l}) Pr(l|t) Pr(o_{i-l+1:i}|t, l)$$

Hidden semi-Markov Models

Standard HMM:

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} Pr(s_i|s_{i-1})\ Pr(o_i|s_i) \cdot Pr(\langle E\rangle|s_n)$$

with $s_0 = \langle S\rangle$

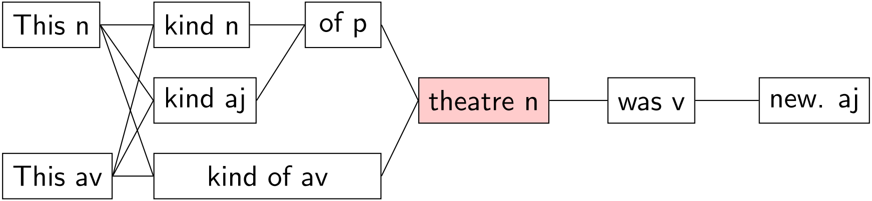
In a segmental HMM, we have:

- frames, equivalent to the time slices in a standard HMM; and
- segments, of one or more frames in length

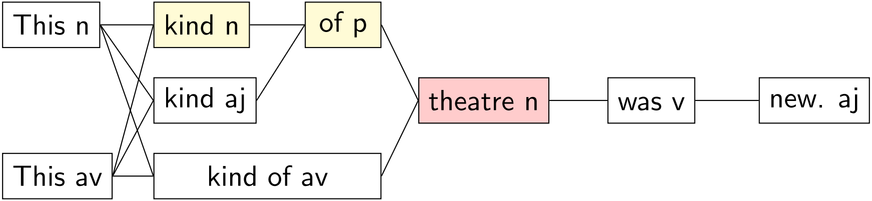
States in a segmental HMM have a tag, t and a length, l.

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} Pr(t_i|t_{i-l})Pr(l|t)\ Pr(o_{i-l+1:i}|t, l)$$

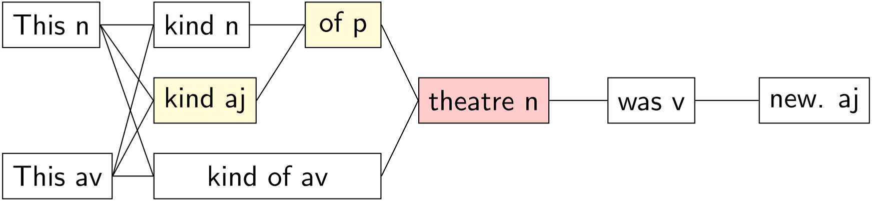
Hidden semi-Markov Models



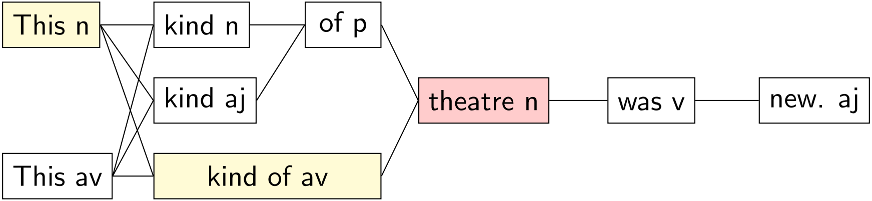
Hidden semi-Markov Models



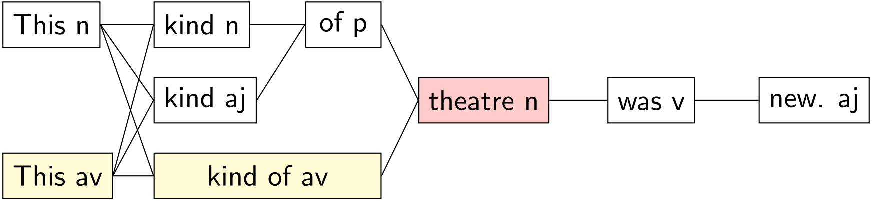
Hidden semi-Markov Models



Hidden semi-Markov Models



Hidden semi-Markov Models



Training

Supervised training using relative frequency counts

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} Pr(t_i|t_{i-l}) \;\; Pr(l|t) \;\; Pr(o_{i-l+1:i}|t, l)$$

Training

Supervised training using relative frequency counts

$$Pr(S_{0:n}, O_{0:n}) = \prod_{i=1}^{n} Pr(t_i|t_{i-l})\ Pr(l|t)\ Pr(o_{i-l+1:i}|t, l)$$

$$Pr(t_i|t_{pp}t_p) = \frac{C(t_{pp}\ t_p\ t_i)}{C(t_{pp}\ t_p)} \text{ (trigram)}$$

$$Pr(l|t)Pr(o_{i-l+1:i}|l, t) = \frac{C(l, t)}{C(t)} \cdot \frac{C(o_{i-l+1:i}, l, t)}{C(l, t)} = \frac{C(o_{i-l+1:i}, l, t)}{C(t)}$$

Training

			Lexitems	
Data Set	**Gold?**	**Trees**	**All**	**M-T**
DeepBank 1.0 §00–19	yes	33783	661451	6309
Redwoods Treebank	yes	39478	432873	6568
NANC	no	2185323	42376523	399936

Tag types:

- FULL: lexical type plus all lexical rules
  v_np_le:v_prp_olr:v_nger-tr_dlr:w_comma-nf_plr
- INFL: lexical type plus non-punctuation lexical rules
  v_np_le:v_prp_olr:v_nger-tr_dlr
- LTYPE: lexical type
  v_np_le

Observation Complication

Surface form capitalisation is a important clue in assigning tags, *but* this has been ‘normalised’ within the parser.

Current solution: observation consists of tLexItem’s orth() plus information stored at +CLASS.+CASE in the token input feature structure, e.g.

agency:capitalized+lower	n_-_mc_le:n_ms-cnt_ilr
Agency	n_-_pn-gen_le:n_sg_ilr

Not ideal, but at least they are conceptually the same observation.

Tagging: Best Path

Using Viterbi, we pick the best path:

	Segmentation		Tagging	
Tag Type	**F1**	**Sent.**	**F1**	**Sent.**
FULL	99.55	94.48	93.92	42.13
INFL	99.45	93.55	93.74	41.49
LTYPE	99.40	93.03	93.27	38.12

Tagging: Best Path

Using Viterbi, we pick the best path:

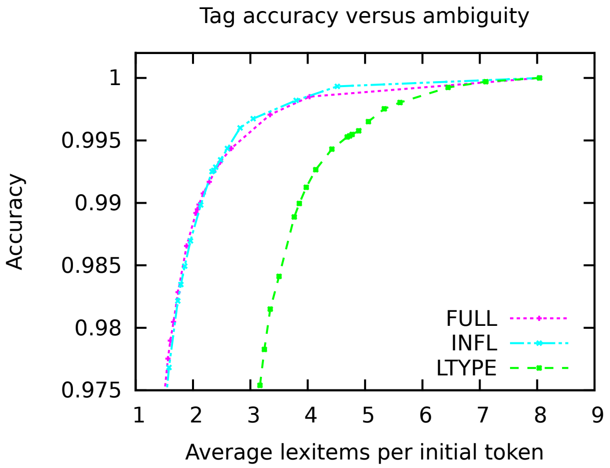
	Segmentation		**Tagging**	
Tag Type	**F1**	**Sent.**	**F1**	**Sent.**
FULL	99.55	94.48	93.92	42.13
INFL	99.45	93.55	93.74	41.49
LTYPE	99.40	93.03	93.27	38.12

Quite good, but not good enough for parsing. Instead we calculate posterior probabilities of each lexitem, and prune those lower than threshold *rho*.

Tagging: Multi-tagging

Tag Type	ρ	Acc.	Lexitems Kept	Ave.
FULL	0.00001	99.71	41.6	3.34
FULL	0.0001	99.44	33.1	2.66
FULL	0.001	98.92	25.5	2.05
FULL	0.01	97.75	19.4	1.56
INFL	0.0001	99.67	37.9	3.04
INFL	0.001	99.25	29.0	2.33
INFL	0.01	98.21	21.6	1.73
INFL	0.02	97.68	19.7	1.58
LTYPE	0.0002	99.75	66.3	5.33
LTYPE	0.002	99.43	55.0	4.42
LTYPE	0.02	98.41	43.5	3.50
LTYPE	0.05	97.54	39.4	3.17

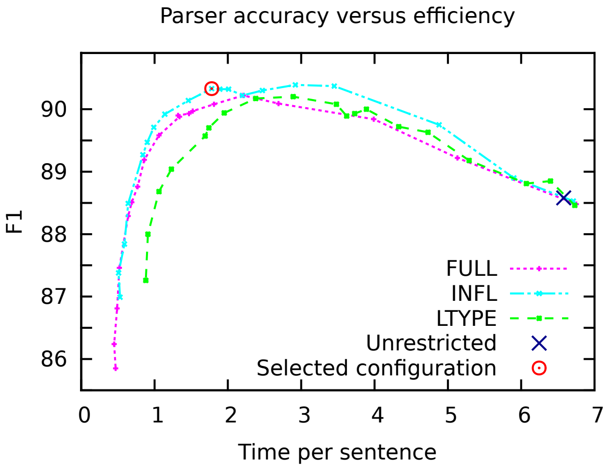
Tagging


Tag accuracy versus ambiguity
Accuracy
Average lexitems per initial token
1
0.995
0.99
0.985
0.98
0.975
1 2 3 4 5 6 7 8 9
FULL
INFL
LTYPE

Parsing

Tag Type	ρ	Lexitem	Bracket	Time
No Pruning		94.06	88.58	6.58
FULL	0.00001	95.62	89.84	3.99
FULL	0.0001	95.95	90.09	2.69
FULL	0.001	95.81	89.88	1.34
FULL	0.01	94.19	88.29	0.64
INFL	0.0001	96.10	90.37	3.45
INFL	0.001	96.14	90.33	1.78
INFL	0.01	95.07	89.27	0.84
INFL	0.02	94.32	88.49	0.64
LTYPE	0.0002	95.37	89.63	4.73
LTYPE	0.002	96.03	90.20	2.89
LTYPE	0.02	95.04	89.04	1.23
LTYPE	0.05	93.36	87.26	0.88

Parsing


Parser accuracy versus efficiency
F1
Time per sentence
FULL
INFL
LTYPE
Unrestricted
Selected configuration
86
87
88
89
90
0
1
2
3
4
5
6
7

Conclusions

	Baseline		Pruned	
Data Set	**F_1**	**Time**	**F_1**	**Time**
WSJ_{21}	88.12	6.06	89.93	1.77
$WeScience_{13}$	86.25	4.09	87.14	1.48
CatB	86.31	5.00	87.11	1.78

- We can select a configuration that gives at least 2-3 times speed up with an increase in F_1 across a variety of data sets
- The speed versus accuracy trade-off can be easily tuned to an application's requirements
- Many of the errors arise from the proper noun vs common noun choice in noun compounds which:
  - may not be important for many applications
  - could probably be more consistent/standardised in the grammar and treebanks

Thank You!

Download

PET branch with übertagging

https://pet.opendfki.de/repos/pet/branches/uebertagger

Only needs the trigram models to be in the grammar.

Training code

http://svn.dridan.com/sandpit/uebertagger

But you need training data in the right form:

$\langle$observation - possibly including case class$\rangle$ $\langle$tag$\rangle$

These links will change, once I work out integration details with both grammar developers and other PET developers, but it is available to test now.