Semi-supervised learning in natural language processing

Anders Søgaard

Course outline

1. Supervised learning
2. Unsupervised learning and semi-supervised learning
3. Learning from weighted data and transfer learning
4. **Applications to dependency parsing**
5. Transfer learning in the blind

Parsing

Semi-supervised dependency parsing

Cross-domain dependency parsing

Cross-language dependency parsing

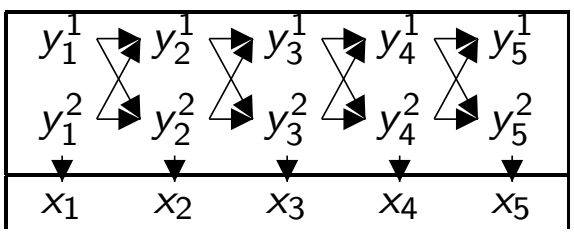
Sequential labeling

(1) Time flies like arrows.

Figure: Naive Bayes and hidden Markov models as Bayesian networks

$$P(\mathbf{x}) = \prod_{v \in \mathcal{V}} P(x_v \mid x_{\text{pa}(v)})$$

1. Consecutive classification
2. Generic structured learning: score all edges and search for minimum spanning sequence/tree/dag/...



Dependency annotation

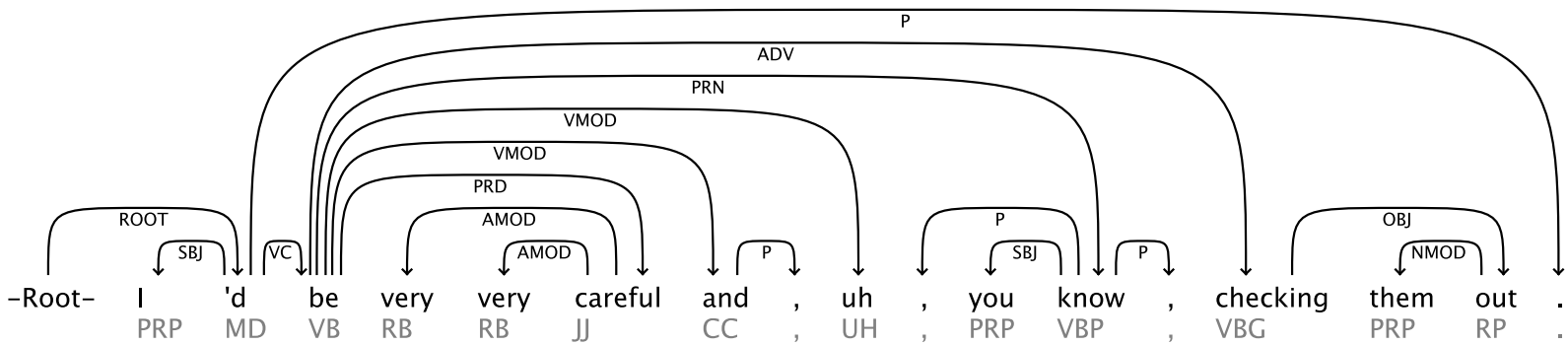


Figure: Syntactic analysis (Switchboard Corpus)

Exercise: What (do you think) is the structure for this sentence?

(2) No broadband and mobile phones banned.

Dependency annotation

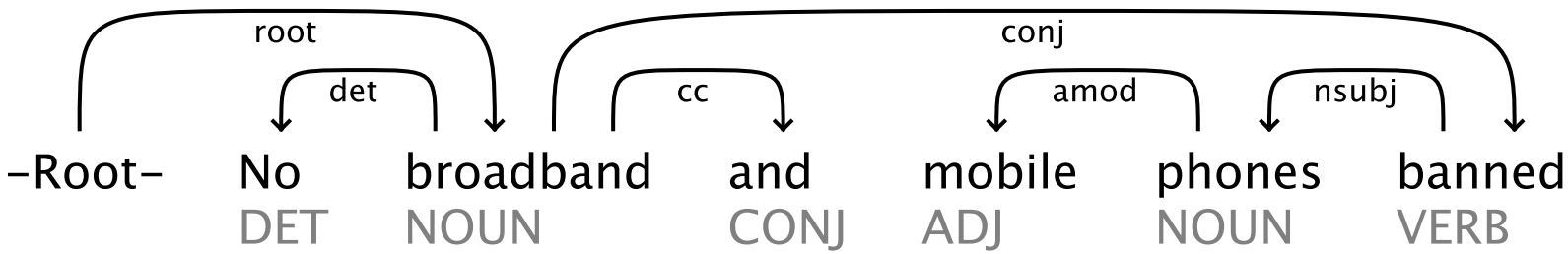


Figure: Syntactic analysis (Web 2.0 Collection)

Dependency annotation

Clear cases		**Difficult cases**	
Head	Dependent	Head	Dependent
Verb	Subject	Auxiliary	Main verb
Verb	Object	Complementizer	Verb
Noun	Attribute	Coordinator	Conjuncts
Verb	Adverbial	Preposition	Nominal
			Punctuation

(3) I can see that they rely on this and that.

Transition-based parsing (arc-standard)

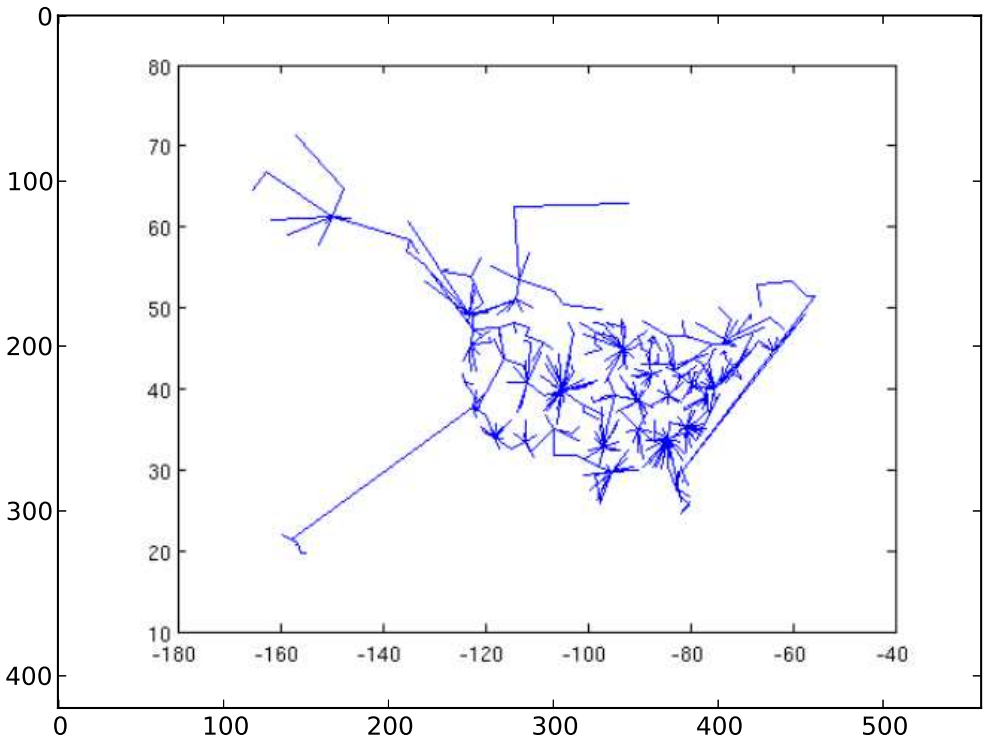
- A configuration $c = \langle \sigma, \beta, A \rangle$ consists of a stack of words σ, a buffer of words β, and a set of dependency arcs A, i.e. triples of a head word, a dependency label, and dependent word.
- The initial configuration is then $\langle [w_0]_\sigma, [w_1, \ldots, w_n]_\beta, \emptyset \rangle$. A final state is any configuration of the form $\langle \sigma, []_\beta, A \rangle$, i.e. any state in which the buffer is empty.
- The SHIFT transition takes a word from the buffer β and moves it to the top of the stack σ. The LEFT-ARC transition removes the top word w_i from the stack and adds a dependency from w_i to the first word on the buffer to A. The RIGHT-ARC transition removes the first word w_j on the buffer and takes the top word w_i on the stack, places w_i back on the buffer and adds a dependency from w_i to w_j to A.

Transition-based parsing (arc-standard)

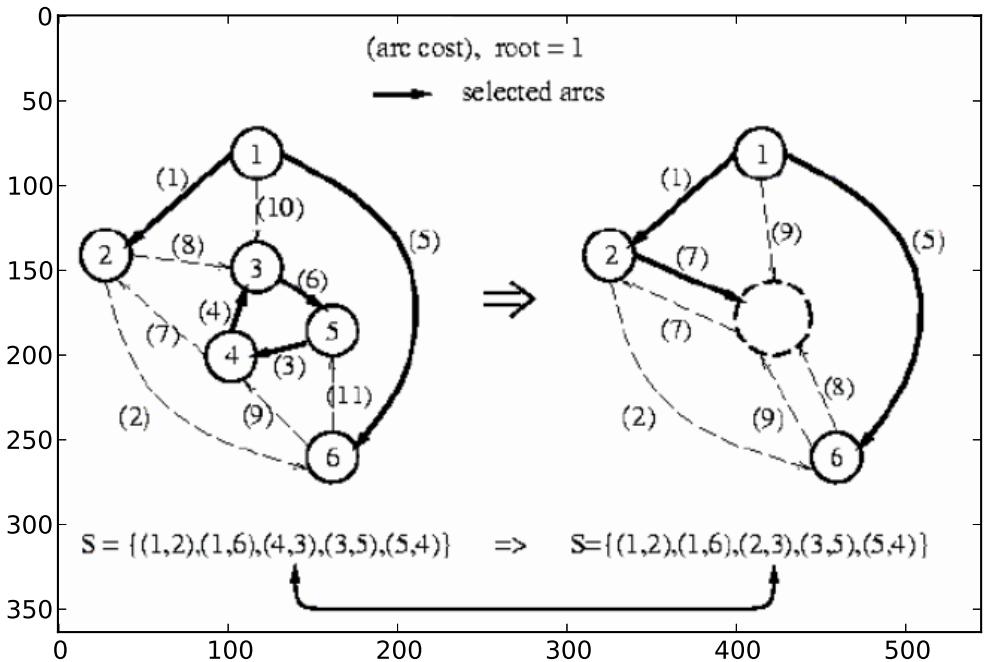
SHIFT	...	John smokes Lebanese	
LEFT-ARC	..., John	smokes Lebanese	
SHIFT	...	smokes Lebanese	John $\leftarrow$ smokes
RIGHT-ARC	..., smokes	Lebanese	
SHIFT	...	... smokes	smokes $\rightarrow$ Lebanese
LEFT-ARC	...	...	ROOT $\rightarrow$ smokes

Figure: Example derivation in transition-based dependency parsing

Minimum spanning tree parsing (in pictures)



Minimum spanning tree parsing (in pictures)



Why semi-supervised and cross-domain parsing?

- Annotation rate $\sim$ 40 words per hour. Entire treebank: 5 years.
- How many treebanks?
- Well ...

$$\text{languages} \times \text{domains} \times \text{language change} \times \text{within-population variation}$$

- Supervised parsing, total cost: $6,909 \times \infty \times \kappa \times \nu_L \times 5$ years $\sim$ Expensive

Why semi-supervised and cross-domain parsing?

HIT-Baseline	LAS	POS
Wall Street Journal	91.88	97.76
Yahoo Answers!	80.75	90.99
BBC Newsgroups	85.26	92.32
Amazon (reviews)	81.60	90.46

Charniak (from McClosky et al., 2010)	F-score
Wall Street Journal	89.7
Brown (WSJ)	84.1
Genia (med.)	76.2
Switchboard (spoken)	76.7

What type of bias can we assume?

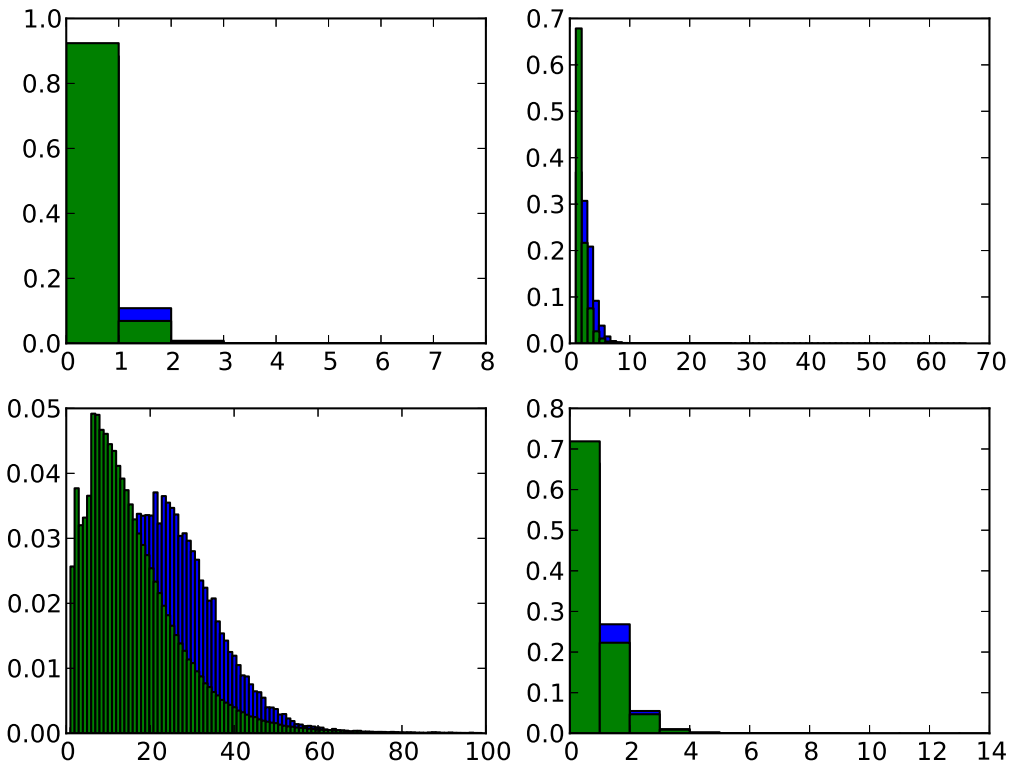
WSJ $\rightarrow$ emails:

- $P(X), P(Y)$ or $P(Y|X)$?

What type of bias can we assume?

WSJ $\rightarrow$ emails:

- $P(X), P(Y)$ or $P(Y|X)$?



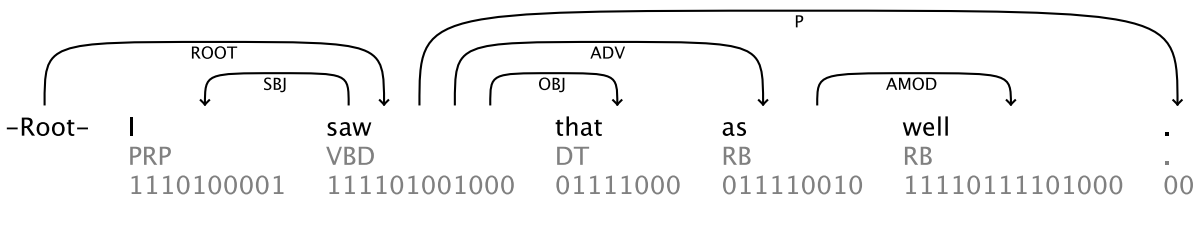
What's in our toolbox?

- Clusters-as-features
- Semi-supervised wrapper methods
- Importance weighting
- Feature-based transfer learning

Note: Are sentences or words our data points?

Clusters-as-features

- Koo et al. (2008). Simple semi-supervised dependency parsing.
- Standard in the SANCL 2012 shared task.
- Used also in POS, NER (Turian et al., 2010), SMT (Rishøj and Søgaard, 2010) and SRL.

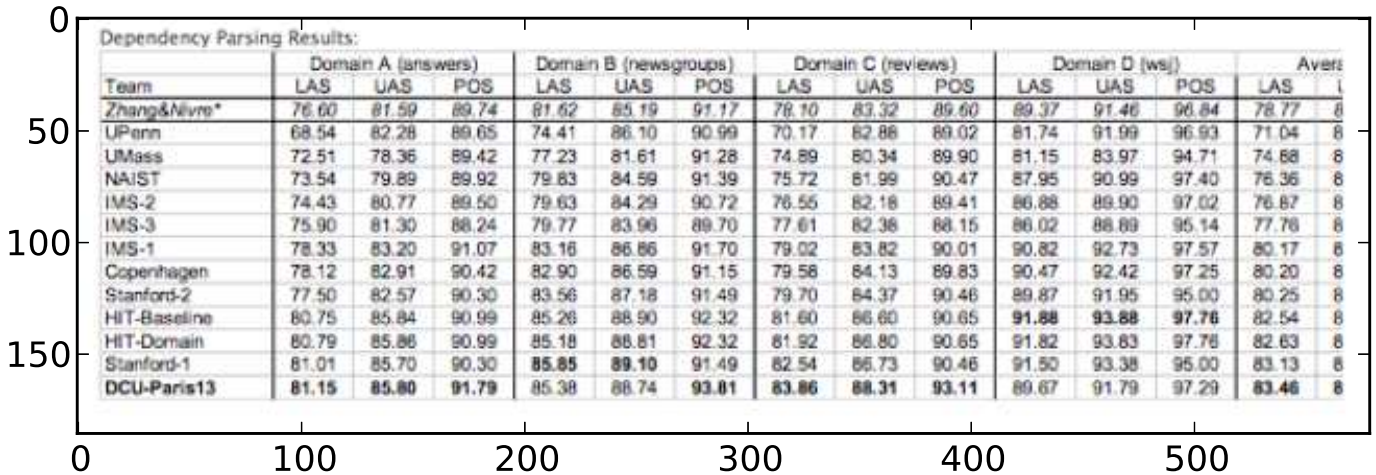


Semi-supervised wrapper methods

- Self-training is often reported not to work (e.g. Spreyer and Kuhn, 2009).
- McClosky et al. (2006) obtained good results with self-training and re-ranking.
- Sagae and Tsujii (2007) won the CoNLL 2007 shared task on domain adaptation using co-training.
- Søgaard and Rishøj (2010) get some of the best results in the literature using (generalized) tri-training, but on unbiased multilingual data. In their experiments, tri-training performs significantly better than S3VM and co-training.
- Combining self-trained parsers in an ensemble also has been successful (Huang et al., 2010; Roux et al., 2012).
- *Exercise:* Why do you think (a) self-training and re-ranking, (b) tri-training, and (c) ensembles of self-trained parsers have lead to good results (while self-training hasn't)?
- Structural correspondence learning has been unsuccessful (Shimizu and Nakagawa, 2007; Plank, 2009).

Danish	LAS(%)	UAS(%)	LA(%)	EM(%)	Δ LAS	err.red(%)	*p*-value
malt/mst2	86.36	90.50	92.09	27.64			
gen-tri	**87.76**	**92.11**	**92.87**	**27.95**	1.40	10.26	<0.0001
gen-tri (-p)	87.54	92.61	91.68				
CONLL-X best (-p)	84.79	90.58	89.22				
Martins et al. (-p)	86.79	-	-				
Dutch							
malt/mst2	81.00	84.58	85.46	24.35			
gen-tri	**83.42**	**88.18**	**87.82**	**28.00**	2.42	12.74	<0.0001
gen-tri (-p)	81.73	86.97	86.61				
CONLL-X best (-p)	79.19	83.57	83.89				
Martins et al. (-p)	81.61	-	-				
German							
malt/mst2	88.06	90.53	93.52	40.06			
gen-tri	**90.41**	**93.22**	**94.61**	**43.14**	2.35	19.68	<0.0001
gen-tri (-p)	90.30	93.49	93.87				
CONLL-X best (-p)	87.34	90.38	92.11				
Martins et al. (-p)	88.66	-	-				
Portuguese							
malt/mst2	85.39	88.80	92.59	28.13			
gen-tri	**88.03**	**91.89**	**93.54**	**29.86**	2.64	18.07	<0.0001
gen-tri (-p)	89.18	93.69	92.43				
CONLL-X best (-p)	87.60	91.36	91.54				
Martins et al. (-p)	88.46	-	-				
Swedish							
malt/mst1	84.74	89.83	89.07	31.11			
gen-tri	**86.83**	**92.04**	**90.65**	**32.65**	2.09	13.70	<0.0001
gen-tri (-p)	86.66	92.45	89.58				
CONLL-X best (-p)	84.58	89.50	87.39				
Martins et al. (-p)	85.16	-	-				
AV					2.18	14.89	

SANCL 2012 shared task (Parsing the Web)



Dependency Parsing Results:

	Domain A (answers)			Domain B (newsgroups)			Domain C (reviews)			Domain D (wsj)			Aver	
Team	LAS	UAS	POS	LAS	UAS	POS	LAS	UAS	POS	LAS	UAS	POS	LAS	
*Zhang&Nivre**	*76.60*	*81.59*	*89.74*	*81.62*	*85.19*	*91.17*	*78.10*	*83.32*	*89.60*	*89.37*	*91.46*	*96.84*	*78.77*	*8*
UPenn	68.54	82.28	89.65	74.41	86.10	90.99	70.17	82.86	89.02	81.74	91.99	96.93	71.04	8
UMass	72.51	78.36	89.42	77.23	81.61	91.28	74.89	80.34	89.90	81.15	83.97	94.71	74.68	8
NAIST	73.54	79.89	89.92	79.83	84.59	91.39	75.72	81.99	90.47	87.95	90.99	97.40	76.36	8
IMS-2	74.43	80.77	89.50	79.63	84.29	90.72	76.55	82.18	89.41	86.88	89.90	97.02	76.87	8
IMS-3	75.90	81.30	88.24	79.77	83.96	89.70	77.61	82.38	88.15	86.02	88.89	95.14	77.76	8
IMS-1	78.33	83.20	91.07	83.16	86.86	91.70	79.02	83.82	90.01	90.82	92.73	97.57	80.17	8
Copenhagen	78.12	82.91	90.42	82.90	86.59	91.15	79.58	84.13	89.83	90.47	92.42	97.25	80.20	8
Stanford-2	77.50	82.57	90.30	83.56	87.18	91.49	79.70	84.37	90.46	89.87	91.95	95.00	80.25	8
HIT-Baseline	80.75	85.84	90.99	85.26	88.90	92.32	81.60	86.60	90.65	**91.88**	**93.88**	**97.76**	82.54	8
HIT-Domain	80.79	85.86	90.99	85.18	88.81	92.32	81.92	86.80	90.65	91.82	93.83	97.76	82.63	8
Stanford-1	81.01	85.70	90.30	**85.85**	**89.10**	91.49	82.54	86.73	90.46	91.50	93.38	95.00	83.13	8
DCU-Paris13	**81.15**	**85.80**	**91.79**	85.38	88.74	**93.01**	**83.86**	**88.31**	**93.11**	89.67	91.79	97.29	**83.46**	**8**

SANCL 2012 shared task (Parsing the Web)

- IMS used self-training and re-ranking.
- Copenhagen used importance weighting.
- Stanford used self-training and re-ranking (and stacking).
- HIT used tri-training.
- DCU-Paris13 used ensembles of self-trained parsers.

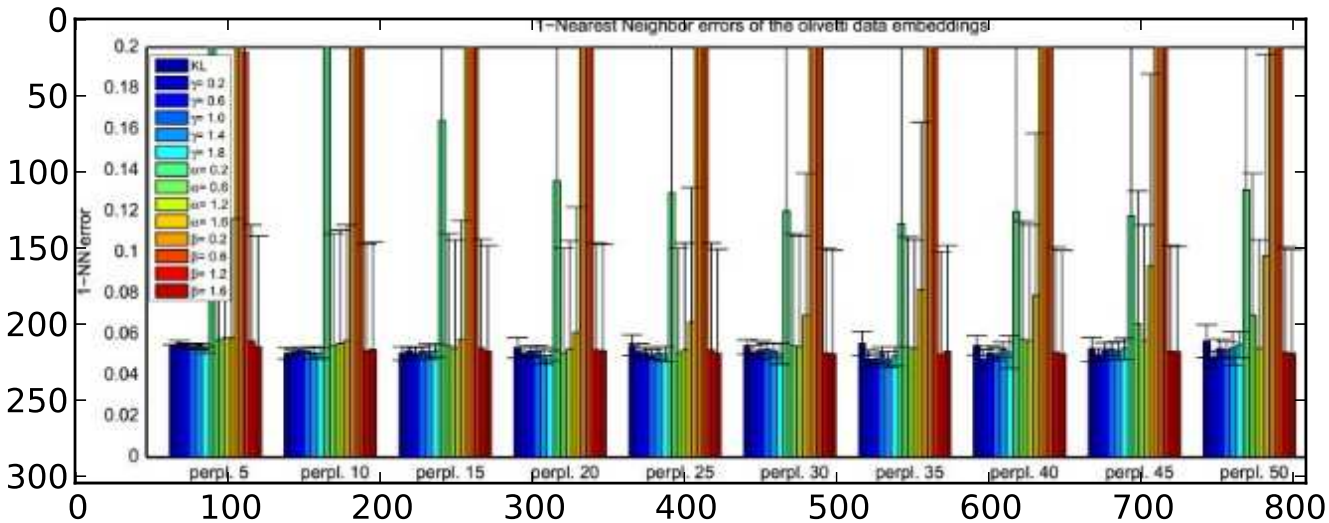
Our insights:

- Combining several weight functions was important.
- Standard ways to obtain robust weight functions did not help us.
  - Standard methods require held-out data to estimate parameters.
- We did observe that clipping or squeezing weights into $(t \rightarrow)$ with $0.1 \leq t \leq 0.5$.

From yesterday: Robust importance weighting?

- Dasgupta and Long (2003) and Cortes et al. (2010) show that importance weighting can hurt in finite sample cases.
- The reason is that distributions must be *estimated*.
- Cortes et al. (2010) suggest to implement a bias-variance trade-off by binning weights in q quantiles.
- It is shown that for some q this gives robust importance weighting, but Cortes et al. (2010) suggest to estimate q on held-out data (which is not available in our case).
- **Technical note:** The problem is related to why KL-divergence is sometimes suboptimal. Hero et al. (2002) show that Renyi divergence with $\alpha = 0.5$ is optimal for relatively similar distributions: $D_\alpha(P, Q) = \frac{1}{\alpha - 1} \log(\sum_i \frac{P(x_i)^\alpha}{Q(x_i)^{\alpha-1}})$
- Thresholding (Jiang and Zhai, 2007) is an alternative.
- We report on dependency parsing experiments with binning and thresholding tomorrow.

From yesterday: Olivetti faces with $\alpha = 1.2$ optimal (Bunte et al., 2012)



What is implemented?

The book code provides wrappers for using your classifiers with:

- MaltParser (Nivre et al., 2007)
- Easyfirst (Goldberg and Elhadad, 2010)

In addition we have implemented sentence-based instance weighting for MSTParser (McDonald et al., 2005) and Mate (Bohnet, 2010). Both use weighted MIRA (Søgaard and Haulrich, 2011).

Importance weighting in MATE

```
1: echo train: $1
2: echo test: $2
3: for TOPICS in 100 200 500 10 do
4:    python lda.py $1.txt gweb-all.txt $TOPICS > $2.lsi.$TOPICS.weights
5:    for DECODETYPE in "proj" do
6:       for ORDER in 2 do
7:          for ITERS in 10 do
8:             echo weights $DECODETYPE $ORDER $TOPICS
9:             if  ! -s $2.$DECODETYPE.$ORDER.$TOPICS.lsi.weights.out  then
10:               java -Xmx32g -cp dist/anna-dt.jar:lib/mstparser-dt.jar is2.parser.Pa
                  rser -model $2.$DECODETYPE.$ORDER.$TOPICS.lsi.weights.model
                  -train $1.conll09 -test $2.c onll09 -weights $2.lsi.$TOPICS.weights
                  -out $2.$DECODETYPE.$ORDER.$TOPICS.lsi.weights.co nll09.out
11:               java -Xmx32g -cp dist/anna-dt.jar:lib/mstparser-dt.jar is2.parser.Pa
                  rser -model $2.$DECODETYPE.$ORDER.$TOPICS.lsi.weights.model
                  -test $3.conll09 -weights $2 .lsi.$TOPICS.weights -out
                  $3.$DECODETYPE.$ORDER.$TOPICS.lsi.weights.conll09.out
12:            end if
13:         end for
14:      end for
15:   end for
16: end for
```

Cross-language dependency parsing

- Delexicalized transfer (Zeman and Resnik, 2008) refers to (a) delexicalizing the source treebank and (b) mapping POS tags. You then simply pretend the source language *is* the target language.
- The idea was revised in 2011:
  - Søgaard (2011) added importance weighting.
  - McDonald et al. (2011) combined delexicalized transfer with structure projection (Smith and Eisner, 2009; Spreyer and Kuhn, 2009).
  - Cohen et al. (2011) used delexicalized transfer models to initialize unsupervised parameter estimation on unlabeled target data.
- Naseem et al. (2012) subsequently explored a more complex transfer model where only hierarchical information is transferred directly to reflect that languages have very different word orders.
- Täckström et al. (2012a) augment delexicalized transfer with bilingual clusters, while Durrett et al. (2012) use a bilingual dictionary to project lexical features.
- Täckström et al. (2012b) used self-training to supply the bilingual word clusters with monolingual clusters, but evaluated the idea on named entity recognition rather than cross-language parser adaptation.

Language-level weighting

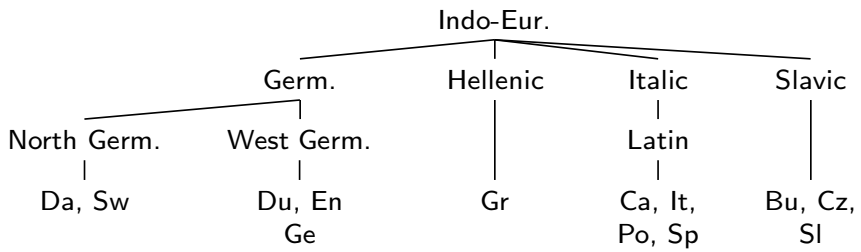


Figure: Language genealogy.

- Language weights using genealogical distance (Berg-Kirkpatrick and Klein, 2010)
- Language weights using typological table distance (Naseem et al., 2012)

$$f_w(L_S) = \frac{5}{H(L_S, L_T)}$$

Constant makes weights comparable to those obtained using linguistic genealogy.

Sentence-level weighting

Each sentence in every source language treebank can be weighted by:

$$f_w(\mathbf{x}) = \frac{\sqrt{ppl_S(\mathbf{x})}}{\sqrt{ppl_T(\mathbf{x})}}$$

where $ppp(\cdot)$ is the perplexity per word given a language model trained on a target language corpus.

sour/targ	Bu	Ca	Cz	Da	Du	Ge	Gr	It	Po	Sl	Sp	Sw	AV	AV	*p*-value
Bu	68.5	55.7	45.0	44.1	50.4	49.2	47.8	55.7	67.0	35.8	51.7	61.1			
Ca	59.7	77.8	43.2	45.9	49.4	57.0	59.1	**74.4**	71.3	51.6	**70.2**	52.5			
Cz	29.6	24.8	33.8	28.1	23.6	24.0	26.2	26.0	23.0	23.3	22.7	21.9			
Da	36.9	30.0	29.5	45.4	25.5	22.9	18.7	37.7	30.5	26.0	29.5	26.6			
Du	59.6	59.4	43.4	46.6	71.2	57.6	63.1	59.5	67.4	44.1	53.1	57.4			
Ge	54.2	51.7	41.7	40.8	43.7	81.2	53.4	55.5	55.0	41.3	49.3	57.9			
Gr	56.5	67.6	45.8	42.3	60.5	52.3	70.6	66.9	67.9	**55.2**	58.0	57.5			
It	61.6	**79.5**	41.1	47.0	49.6	55.1	61.9	77.0	71.6	52.2	67.5	51.7			
Po	49.5	59.7	34.5	25.8	49.0	37.9	50.6	59.3	58.1	34.1	50.6	39.7			
Sl	20.8	16.6	19.5	33.1	19.2	20.4	23.9	18.7	19.3	24.0	17.2	22.8			
Sp	46.0	74.8	30.5	42.1	41.3	46.6	42.6	64.1	67.2	42.1	72.9	46.9			
Sw	58.6	58.3	36.5	45.0	50.8	54.2	56.7	57.6	68.3	37.5	55.1	80.6			
flat	62.1	68.9	45.5	46.0	54.0	**58.0**	63.4	67.4	76.5	44.7	64.4	60.7	59.3	61.3	
gtree	62.1	67.0	**46.9**	45.8	54.3	57.7	63.4	69.4	75.5	47.4	66.0	61.6	59.8	61.7	
typology	62.1	68.9	45.5	46.0	54.3	56.3	62.8	67.7	76.5	44.7	64.7	58.4	59.0	60.8	
pr	62.6	69.3	46.1	45.8	53.2	57.4	63.2	68.0	76.0	46.4	65.3	60.1	59.4	61.1	
vote (a)	62.7	69.3	45.8	46.3	54.0	57.7	63.4	68.8	76.4	45.8	65.6	61.2	60.5	62.6	
weighted	64.6	68.5	**46.9**	48.6	57.2	56.3	65.0	67.2	**77.6**	44.6	62.9	62.4	60.1	62.1	< 0.01
w-typ	64.6	68.5	**46.9**	48.6	58.6	55.7	**66.2**	68.3	**77.6**	44.6	62.8	61.4	60.3	62.4	< 0.01
w-gtree	64.2	69.1	46.7	48.3	57.2	56.3	65.0	68.3	76.9	46.1	63.8	63.1	60.4	62.4	< 0.01
vote (b)	64.2	68.9	46.9	48.6	57.3	57.4	65.3	69.0	77.3	45.7	63.6	62.6	-	-	< 0.01
MPH(di)	-	-	-	48.9	55.8	56.7	60.1	64.1	74.0	-	64.2	65.3		61.2	
MPH(pr)	-	-	-	**49.5**	**65.7**	56.6	65.1	65.0	75.6	-	64.5	**68.0**		**63.8**	
TMU(di)	-	-	-	36.7	52.8	48.9	-	64.6	66.8	-	60.2	55.4		55.1*	
TMU(cl)	-	-	-	38.7	54.3	50.7	-	68.8	71.0	-	62.9	56.9		57.6*	
NBO(b)	**66.8**	71.8	44.6	-	55.9	53.7	67.4	65.6	73.5	-	62.1	61.5			