Deutsche Forschungsgemeinschaft DFG

Unexpected Productions May Well Be Errors

Sonderforschungsbereich 441
Linguistische Datenstrukturen

Tylman Ule and Kiril Simov
ule@sfs.uni-tuebingen.de, kivs@bultreebank.org

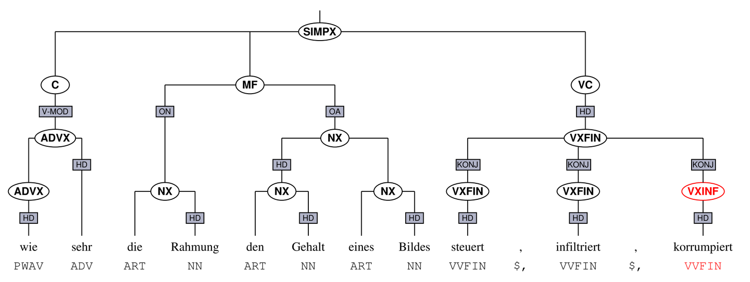
Bulgarian Academy of Sciences
Linguistic Modelling Laboratory

Overview

We present a method that looks for unexpected tree fragments in treebanks, that often turn out to be annotation errors. The method is based on the assumption that nodes should behave regularly over the whole treebank, especially when wider context is considered. We present the algorithm and its evaluation via artificial errors.

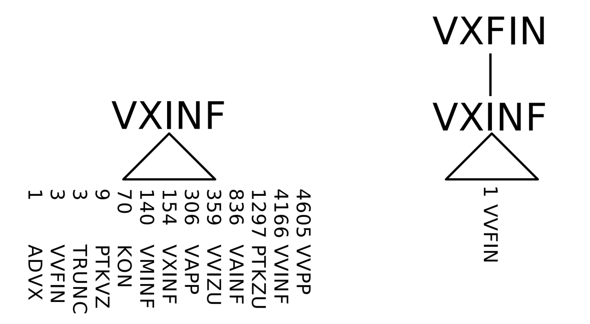
Productions of a Focus Node in Context

generally:	context-free:	e.g.:	
context	parent node	VXFIN	finite verb coordination
focus	focus node	VXINF	non-finite verb phrase
production	daughter nodes	VVFIN	finite full verb (POS)



We are looking for unexpected productions of a focus node in context

- Node types have a characteristic distribution of productions
- The distribution depends on further context
- Errors tend to be low-frequency events



Unexpected Productions

$$\chi^2_{ik} = \sum_m \frac{(expfreq(c_{ik}, p_{im}) - obsfreq(c_{ik}, p_{im}))^2}{expfreq(c_{ik}, p_{im})}$$

c_{ik} context type k of focus type i
p_{im} production type m of focus type i

Resulting statistics do not regularly point to errors directly:

m	$obsfreq(c_{ik}, p_{im})$	$expfreq(c_{ik}, p_{im})$	p_{im}
1	3793	1466.77	PPER
2	1845	506.02	PRF
3	1361	3483.56	NN
4	951	454.22	PIS
5	852	1737.48	NE
...	...	...	...
139	1	0.26	SIMPX NN
...	...	...	...
254	5	4.95	KOKOM NE

m	$obsfreq(c_{ik}, p_{im})$	$expfreq(c_{ik}, p_{im})$	p_{im}
1	1	0.00	VVFIN

- Example above:
  iteration: 36, $obsfreq(c_{ik}) = 1$,
  $\chi^2_{ik} = 3677$, c_{ik} =VXFIN, i =VXINF
- Example to the left:
  iteration: 9, $obsfreq(c_{ik}) = 23819$,
  $\chi^2_{ik} = 11324.12$, c_{ik} =MF, i =NX

Ranking Error Candidates

- The χ^2-test is sensitive to errors: each line in $\sum_m$ is an *error candidate*
- Not all candidates are equally likely errors
- Minimise human effort: give most likely candidates first
- The items of the above list are therefore first classified using ML and then sorted

Classifying Error Candidates

- Features presented to the Machine Learner [Daelemans et al., 2003] are as follows

$obsfreq(c_{ik}, p_{im})$	occurrences observed in the corpus
$expfreq(c_{ik}, p_{im})$	the expected number of occurrences
$\frac{(expfreq(c_{ik},p_{im})-obsfreq(c_{ik},p_{im}))^2}{expfreq(c_{ik},p_{im})}$	its contribution to χ^2_{ik}
χ^2_{ik}	the overall χ^2_{ik} over all m
termratio	fraction of overall χ^2_{ik} contributed by this triple
rank	triple is *rank* highest contributor to χ^2_{ik}
rankratio	the relative position as contributor: *rank*/m
alt	the number of other contributors to χ^2_{ik}, i.e. $m-1$
$obsfreq(c_{ik} > f_i)$	the number of times c_{ik} dominates f_i
iter	the iteration detecting (c_{ik}, f_i, p_{im})
iterratio	fraction of *iter* from all iterations

- ML output classes are: error in context | focus | production | no error

Sorting Error Candidates

- The ML stage can only predict *focus* errors reliably
- Error Candidates should be sorted by decreasing chance that they are true errors
- Sorting is performed according to the following intellectually defined sort keys:

1	$obsfreq(c_{ik}, p_{im}) \leq 3$	4	$rankratio = termratio$	7	higher *rankratio*
2	ML says focus node error	5	smaller $obsfreq(c_{ik}, p_{im})$	8	higher *termratio*
3	ML predicts some error	6	smaller $expfreq(c_{ik}, p_{im})$	9	lower *iterratio*

1 low-frequency events first
2–3 the classification via ML
4–6 smaller number of productions first
7–8 more relevant contributors to χ^2_{ik} first
9 prefer early iterations

Spotting Errors

➤determine all χ^2_{ik} → ➤rank error candidates → ➤remove error(s)

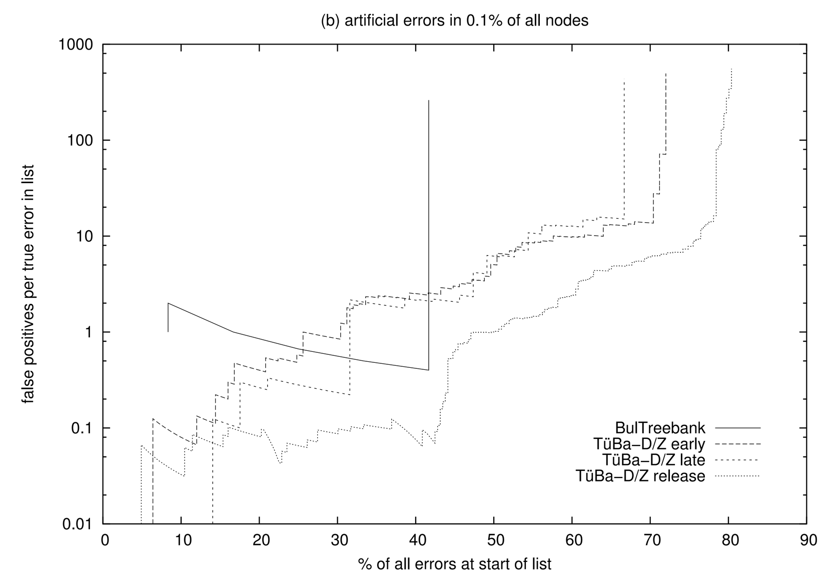
repeat until top of list is error-free

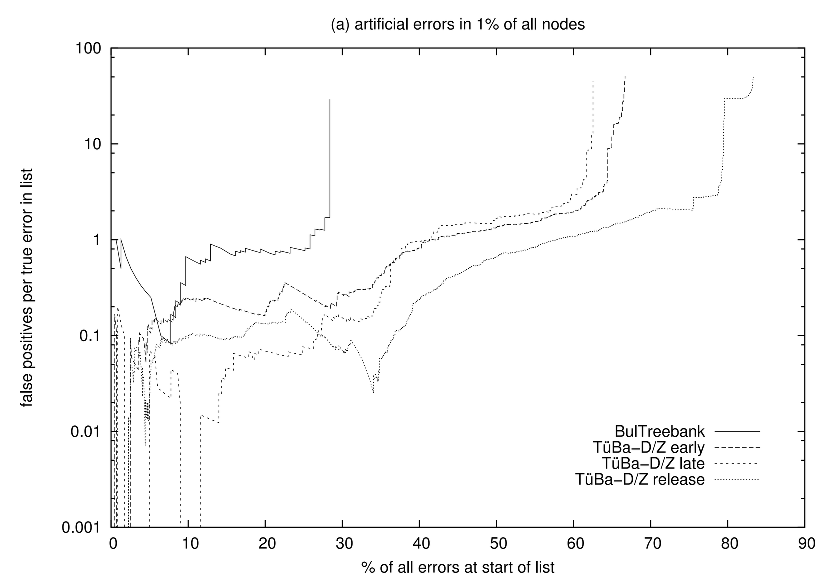
How many Error Candidates need to be inspected to find a True Error?

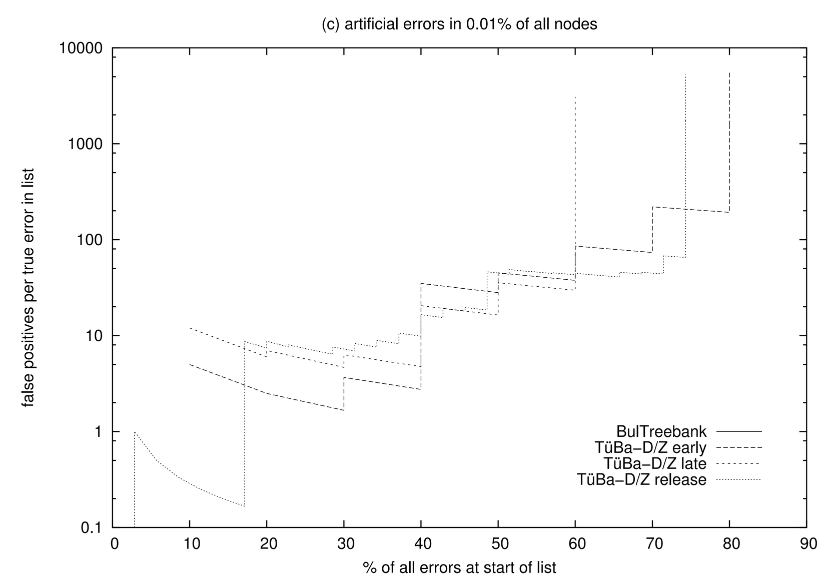
- Attempting an evaluation

- Inject errors into a fraction of all nodes
- Apply ML via ten-fold training/classification
- Sort test data according to sort keys

- Plot the percentage of true injected errors included from the beginning of the list
- Against the number of falsely presented errors per true injected error found







Spotting Errors in Corpora of different Language, Size, and Quality

- Corpora usually vary in size and quality during development
- Annotation schemes can usually be coerced into *context, focus, production*

- Larger/Cleaner Corpora ease error spotting
- Small Minimal Threshold for Detectable Errors
- Applicable early in corpora development: Spot errors in small corpora with many errors

Sizes of the Data Sets

	BTB	TB *late*	TB *early*	TB *release*
sent.	580	3074	7398	15260
nodes	15013	56601	132640	318596

Artificial Errors Introduced / Detected

1.00%	155/44	579/362	1279/856	3168/2641
0.10%	12/5	57/38	125/90	306/246
0.01%	2/0	10/6	10/8	35/26

ML Precision/Recall for *focus* Errors

1.00%	.42/.35	.72/.72	.60/.65	.68/.69
0.10%	0.0/0.0	.49/.59	.44/.61	.73/.69
0.01%	0.0/0.0	0.0/0.0	0.0/0.0	.30/.30

Artificial Errors vs. Genuine Errors

- True errors occur among artificial errors
- Many errors have been accidentally found during application of χ^2_{ik} for a different purpose

- Objective evaluation is otherwise very hard on several corpora
- Alternative will be to compare an early version of a corpus with a more revised version

Another approach will be to evaluate against the errors found by other methods. In contrast to the presented method, other approaches to automatic detection of errors usually target the layer of POS tags [Kveton and Oliva, 2002], or they rely heavily on lexical information [Dickinson and Meurers, 2003], requiring larger corpora. They report performance in terms of true errors on one corpus each.

References

W. Daelemans, J. Zavrel, K. van der Sloot, and A. van den Bosch. TiMBL: Tilburg Memory Based Learner, version 5.0, Ref. Guide. Technical Report 03-10, ILK, 2003. URL http://ilk.uvt.nl/downloads/pub/papers/ilk.0310.pdf.

M. Dickinson and W. D. Meurers. Detecting inconsistencies in treebanks. In *Proceedings of TLT 2003*, Växjö, Sweden, 2003. URL http://ling.osu.edu/~dm/papers/dickinson-meurers-tlt03.html.

P. Kveton and K. Oliva. (Semi-)automatic detection of errors in PoS-tagged corpora. In *Proceedings of COLING 2002*, Taipei, Taiwan, August 2002.