


UNIVERSITEIT VAN TILBURG

Levenshtein-distance-based post-processing

shared task spotlight

Antal van den Bosch
ILK / CL and AI, Tilburg University



Ninth Conference on Computational Natural Language Learning (CoNLL-2005)
Ann Arbor, MI, June 29-30, 2005

UNIVERSITEIT VAN TILBURG

Global models \approx Post-processing

- Many systems:
 - local model vs. global model
- Goal of global model:
 - correcting mislabelings caused by “blind” decisions of a more local model
- Methods:
 - Probabilistic language models over *argument labeling language*
 - Distance-based error correction over *argument labeling language*

UNIVERSITEIT VAN TILBURG

Hand-crafted post-processing

- “eraser” script by Erik Tjong Kim Sang (2004):
 - For each verb, if there is any double A0 - A5 in the sentence, delete the one furthest from the verb.
- Can only improve precision, not recall.
- Will typically lower recall, because it will delete the incorrect one of a double occasionally.

UNIVERSITEIT VAN TILBURG

Data-driven post-processing

- Idea: argument labeling correction as spelling correction.
 - Classical solution: Levenshtein or string-edit distance (Levenshtein, 1965). Sum over:
 - Deletion: distance++
 - Insertion: distance++
 - Substitution: distance++
 - Closest found argument patterns contains corrections that need to be applied.
 - Predicted: *emphasize A0 V AI A0*
 - Nearest in training data and PropBank at distance 1: *emphasize A0 V AI*
 - Correction: *delete final A0 in predicted string*

UNIVERSITEIT VAN TILBURG

Data-driven post-processing (2)

- Implements deletions and replacements
- Does not perform insertions
 - Does not know where to “insert”
- Levenshtein-based correction
 - Should be able to improve precision, like “eraser”
 - Might improve recall, due to correct replacements
- Can be applied to all systems!
 - >500 deletions, >250 replacements for some systems on WSJ dev & test
 - <100 deletions, <75 replacements for systems that already have a global model

UNIVERSITEIT VAN TILBURG

Levenshtein post-processing: Example 1

- System **marquez**, Brown: 40 deletions, 30 replacements. E.g.,
 - bend *AI V AI*
 - bend *A0 V AI*
 - love *AI A0 V AI*
 - love *A0 V AI*
 - unite *AI V A2*
 - unite *AI V*

Levenshtein postprocessing: Example 2

- System **punayakanok**, Brown: 20 deletions, 17 replacements. E.g.,

- search V A0
- search V A I

because in data:

- search A0 V
- search A0 V A I

Levenshtein postprocessing: Example 3

- System **pradhan**, WSJ dev: 111 deletions, 72 replacements. E.g.,

[[That dividend]_{A1} is almost double the 35% currently **taken** out of Farmers by B.A.T]_{A1} , the spokesman **added** .

[That dividend is almost double the 35% currently **taken** out of Farmers by B.A.T]_{A1} , the spokesman **added** .

Levenshtein post-processing: effect

system	post	glob	WSJ test	WSJ test LPP	Brown test	Brown test LPP
punayakanok		✗	79.44	79.16	67.75	67.59
haghighi		✗	78.45	78.26	67.71	67.18
marquez			77.97	77.98	67.42	67.88
pradhan			77.34	77.10	67.07	66.90
surdasnu		✗	76.46	76.16	65.42	65.20
che			76.44	76.29	65.09	64.97
moschitti			75.89	76.84	63.81	65.46
yi			75.17	75.05	63.14	63.03
ozgencl			74.44	74.32	64.20	64.02
johansson			74.30	75.06	62.79	63.82
cohn		✗	73.10	73.25	63.63	63.85
park			72.68	73.40	62.38	63.36
sutton			66.73	67.33	58.60	59.73

Blue: increase; red: decrease; bold number and cell color: effect > 0.2