Transformation-Based Tagging

PA154 Language Modeling (6.1)

Pavel Rychlý

pary@fi.muni.cz

March 23, 2023

Source: Introduction to Natural Language Processing (600.465)
Jan Hajič, CS Dept., Johns Hopkins Univ.
www.cs.jhu.edu/˜hajic

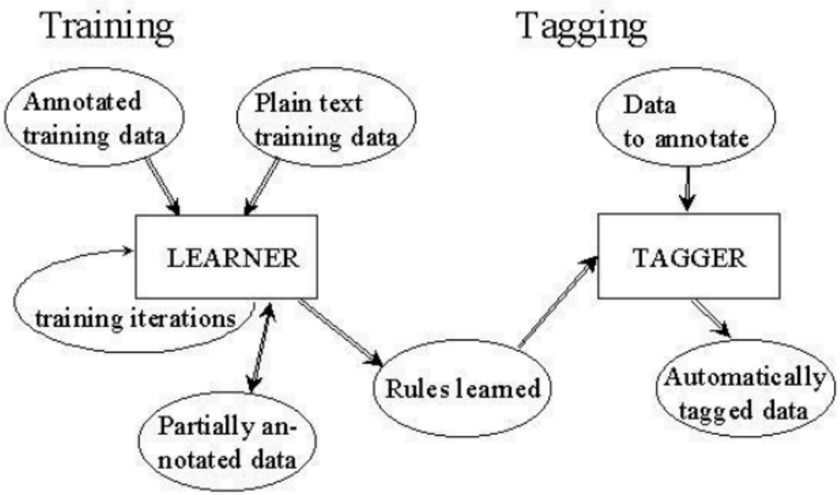
The Task, Again

- Recall:
  - tagging $\sim$ morphological disambiguation
  - tagset $V_T \in (C_1, C_2, \ldots, C_n)$
    - C_i - moprhological categories, such as POS, NUMBER, CASE, PERSON, TENSE, GENDER,....
  - mapping $w \rightarrow \{t \in V_T\}$ exists
    - restriction of Morphological Analysis: $A^+ \rightarrow 2^{(L, C1, C2, \ldots, Cn)}$, where A is the language alphabet, L is the set of lemmas
  - extension to punctuation, sentence boundaries (treated as word)

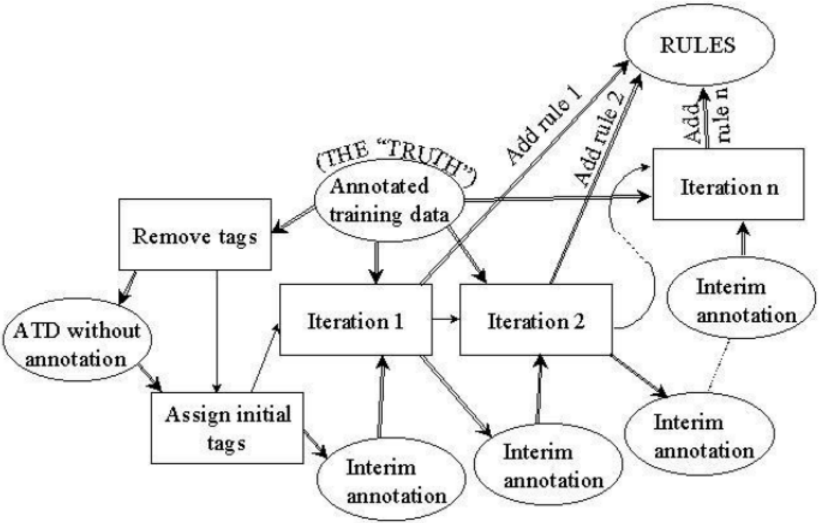
Setting

- **Not** a source channel view
- **Not** even a probabilistic model (no *numbers* used when tagging a text after a model is developed)
- Statistical, yes:
    - uses training data (combination of supervised [manually annotated data available] and unsupervised [plain text, large volume] training)
    - learning [rules]
    - criterion: accuracy (that’s what we are interested in in the end after all!)

The General Scheme



The Learner


RULES
Add rule 1
Add rule 2
Add rule n
(THE "TRUTH")
Annotated training data
Iteration n
Remove tags
Iteration 1
Iteration 2
Interim annotation
ATD without annotation
Assign initial tags
Interim annotation
Interim annotation
Interim annotation

The I/O of an Iteration

- In (iteration i):
  - Intermediate data (initial or the result of previous iteration)
  - The TRUTH (the annotated training data)
  - *pool of possible rules*
- Out:
  - One rule $r_{selected(i)}$ to enhance the set of rules learned so far
  - Intermediate data (input data transformed by the rule learned in this iteration, $r_{selected(i)}$)

The Initial Assignment of Tags

- One possibilty:
  - NN
- Another:
  - the most frequent tag for a given word form
- Even:
  - use an HMM tagger for the initial assignment
- Not particulary sensitive

The Criterion

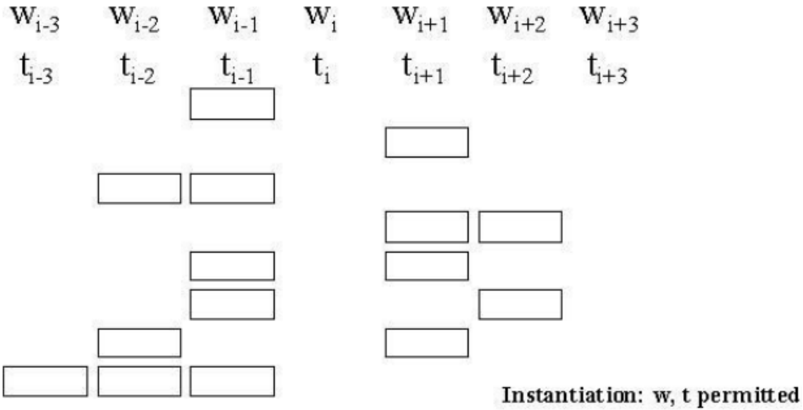
- Error rate (or Accuracy):
  - beginning of an iteration: some error rate E_{in}
  - each possible rule r, when applied at every data position:
    - makes an improvement somewhere in the data ($c_{improved}(r)$)
    - makes it worse at some places ($c_{worsened}(r)$)
    - and, of course, does not touch the remaining data
- Rule contribution to the improvement of the error rate:
  - $contrib(r) = c_{improved(r)} - c_{worsened}(r)$
- Rule selection at iteration i:
  - $r_{selected(i)} = argmax_r contrib(r)$
- New error rate: $E_{out} = E_{in} - contrib(r_{selected(i)})$

The Stopping Criterion

- Obvious:
    - no improvement can be made
        - contrib(r) ≤ 0
    - or improvement too small
        - contrib(r) $\leq$ Threshold
- NB: prone to overtraining!
    - therefore, setting a reasonable threshold advisable
- Heldout?
    - maybe: remove rules which degrade performance on H

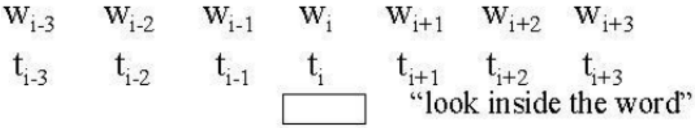
The Pool of Rules(Templates)

- Format: *change tag at position i from a to b / condition*
- Context rules (condition definition - ”template”):


w_{i-3} w_{i-2} w_{i-1} w_i w_{i+1} w_{i+2} w_{i+3}
t_{i-3} t_{i-2} t_{i-1} t_i t_{i+1} t_{i+2} t_{i+3}
Instantiation: w, t permitted

Lexical Rules

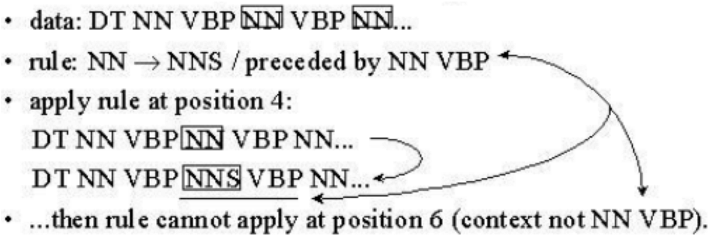
- Other type: lexical rules


w_{i-3} w_{i-2} w_{i-1} w_i w_{i+1} w_{i+2} w_{i+3}
t_{i-3} t_{i-2} t_{i-1} t_i t_{i+1} t_{i+2} t_{i+3}
“look inside the word”

- Example:
  - w_i has suffix -ied
  - w_i has prefix ge-

Rule Application

- Two possibilities:
  - immediate consequences (left-to-right):



- data: DT NN VBP NN VBP NN...
- rule: NN → NNS / preceded by NN VBP
- apply rule at position 4:

  DT NN VBP NN VBP NN...

  DT NN VBP NNS VBP NN...
- ...then rule cannot apply at position 6 (context not NN VBP).

  - delayed (”fixed input”):
    - use original input for context
    - the above rule then applies twice

In Other Words...

1. Strip the tags off the truth, keep the original truth
2. Initialize the stripped data by some simple method
3. Start with an empty set of selected rules S.
4. Repeat until the stopping criterion applies:
   - compute the contribution of the rule r, for each r:
     $contrib(r) = c_{improved}(r) - c_{worsened}(r)$
   - select r which has the biggest contribution contrib(r), add it to the final set of selected rules S.
5. Output the set S

The Tagger

- Input:
    - untagged data
    - rules (S) learned by the learner
- Tagging:
    - use the same initialization as the learner did
    - for i = 1..n (n - the number of rules learnt)
        - apply the rule i to the whole intermediate data, changing (some) tags
    - the last intermediate data is the output

N-best & Unsupervised Modifications

- N-best modification
  - allow adding tags by rules
  - criterion: optimal combination of accuracy and the number of tags per word (we want: close to $\downarrow$ 1)
- Unsupervised modification
  - use only unambiguous words for evaluation criterion
  - work extremely well for English
  - does not work for languages with few unambiguous words