Transformation-Based error-driven Learning (TBL)

LING 572
Advanced Statistical Methods for NLP
February 14, 2019

Based on F. Xia '18

Outline

- Basic concept and properties
- Case study

Basic concepts and properties

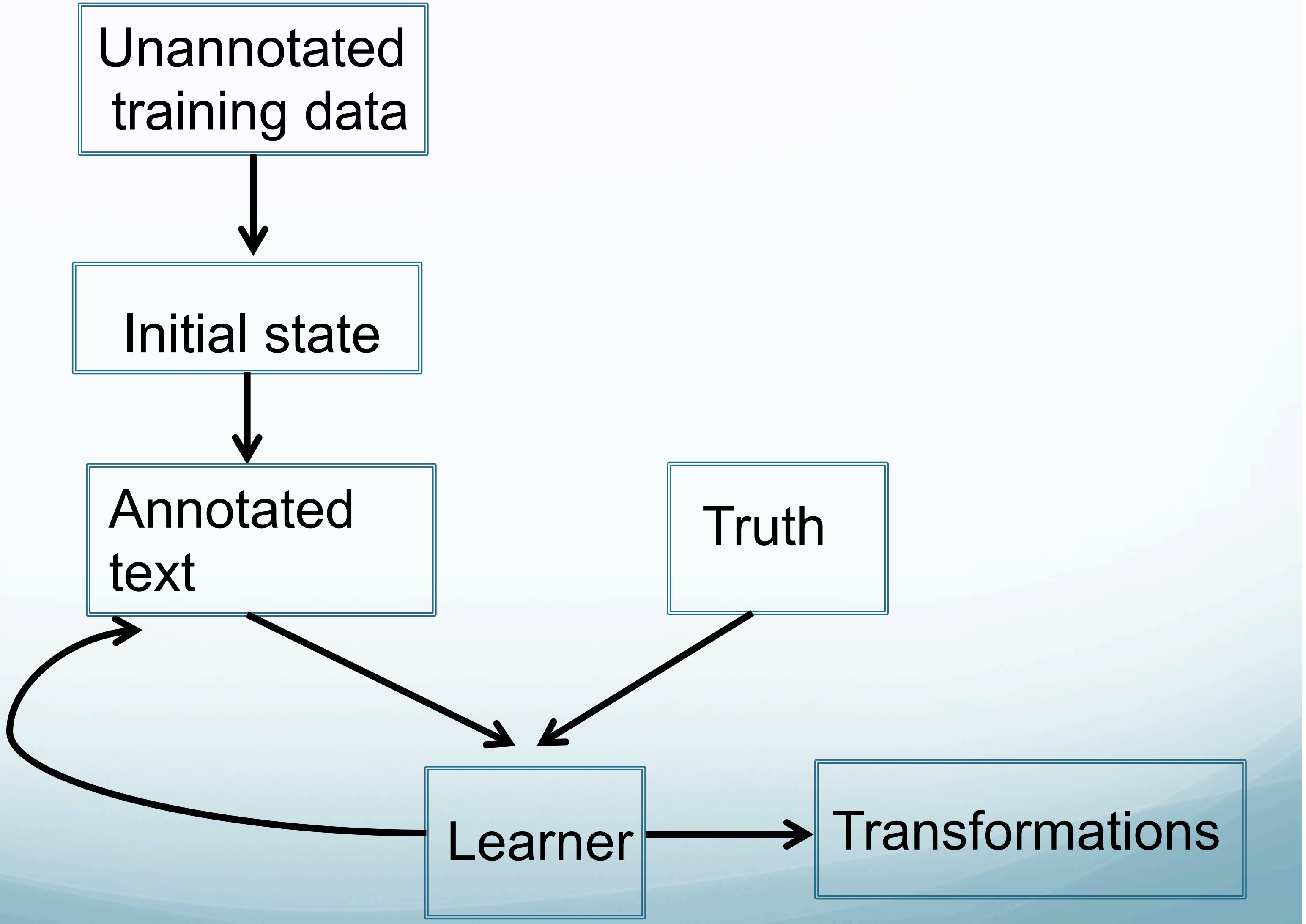
TBL overview

- Introduced by Eric Brill (1992)

- Intuition:
  - Start with some simple solution to the problem
  - Then apply a sequence of transformations to improve the results

- Applications:
  - Classification problem
  - Sequence labeling problem: e.g., POS tagging
  - …

TBL training stage



Unannotated training data

Initial state

Annotated text

Truth

Learner

Transformations

Training data: attribute-value table (Input to the training stage)

	f_1	f_2	…	f_K	Target
x_1	0	1	2.5	-13	c_2
x_2	2.5	0	0	20	c_1
x_3					
…					
x_n					

Training in TBL

	f_1	f_2	…	f_K	Target	CurTag
x_1	0	1	2.5	-13	c_2	c_2
x_2	2.5	0	0	20	c_1	c_3
x_3						
…						
x_n						

Transformations

- A transformation has two components:
  - A trigger environment: e.g., the previous tag is DT
  - A rewrite rule: **change** the current tag from MD to N

If (prev_tag == DT) then MD ➔ N

- Similar to a rule in decision tree, but the rewrite rule can be complicated (e.g., change a parse tree)

➔ TBL can be more powerful than a classifier

Training time: learn transformations

1. Initialize each instance in the training data with an initial annotator

2. Consider all the possible transformations, and choose the one with the highest score.

3. Append it to the transformation list and apply it to the training corpus to obtain a “new” corpus.

4. Repeat steps 2-3.

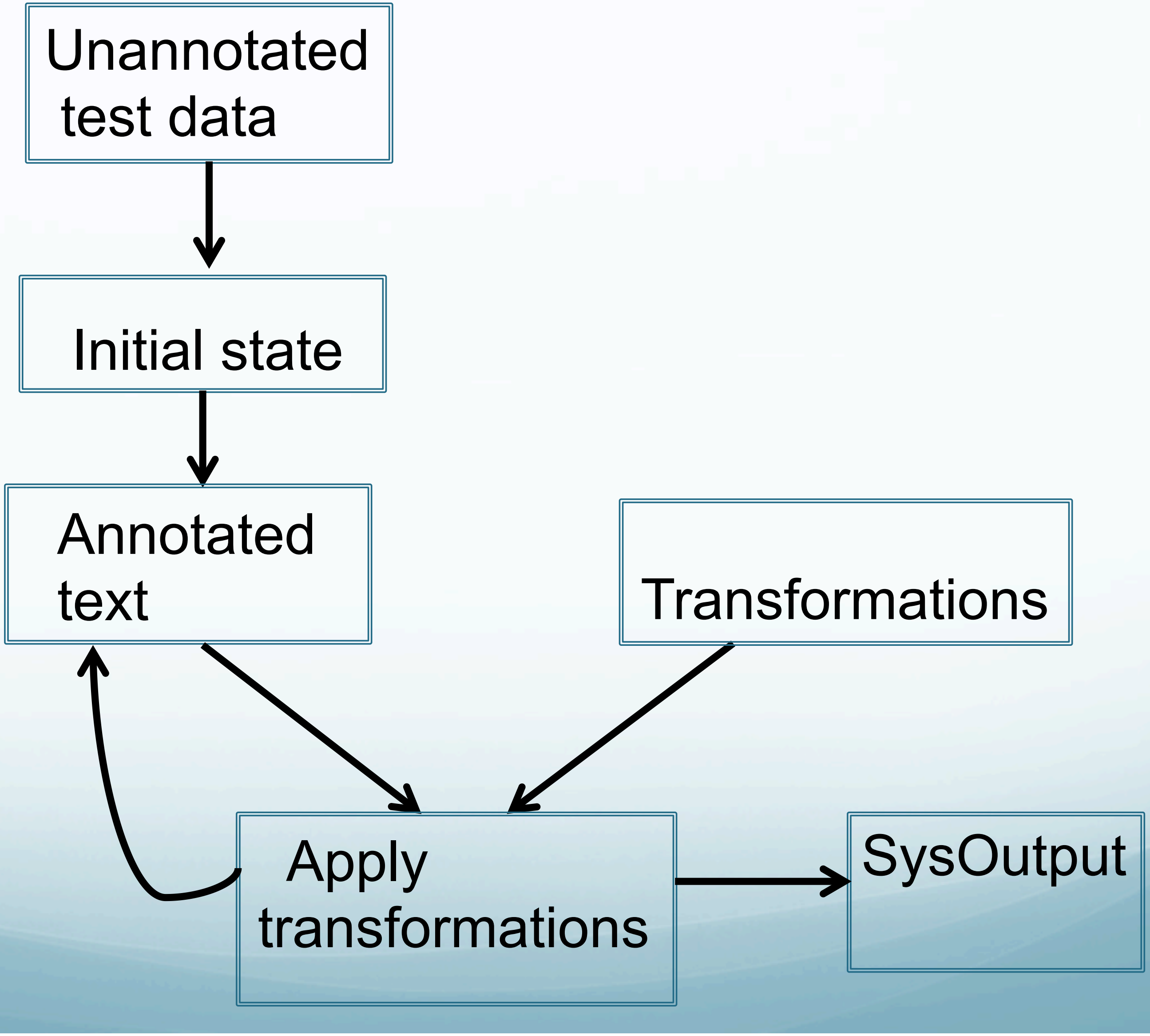
➔ Steps 2-3 can be expensive. Various ways to address the problem.

Testing time: applying transformations

1. Initialize each example in the test data with the same initial annotator

2. Apply the transformations in the same order as they were learned.

TBL test stage



Using TBL

- Pick the initial state-annotator
- Decide the space of allowable transformations
  - Triggering environments
  - Rewrite rules
- Choose an objective function: (e.g., minimize error rate).
  - for comparing the corpus to the truth
  - for choosing a transformation

Using TBL (cont)

- Two more parameters:
  - Whether the effect of a transformation is visible to following transformations
  - If so, what's the order in which transformations are applied to a corpus?
    - left-to-right
    - right-to-left

Order Matters

- Transformation:

  If prevLabel=A

  then change the curLabel from A to B.

- Input: A A A A

- Output:
  - "Not immediate" results: A B B B
  - Immediate results, left-to-right: A B A B
  - Immediate results, right-to-left: A B B B

Case study

TBL for POS tagging

- The initial state-annotator: most common tag for a word according to the training data

- The space of allowable transformations
  - Rewrite rules: change cur_tag from X to Y.
  - Triggering environments (feature types): unlexicalized or lexicalized

Unlexicalized features

- t_{-1} is z
- t_{-1} or t_{-2} is z
- t_{-1} or t_{-2} or t_{-3} is z
- t_{-1} is z and t_{+1} is w
- …

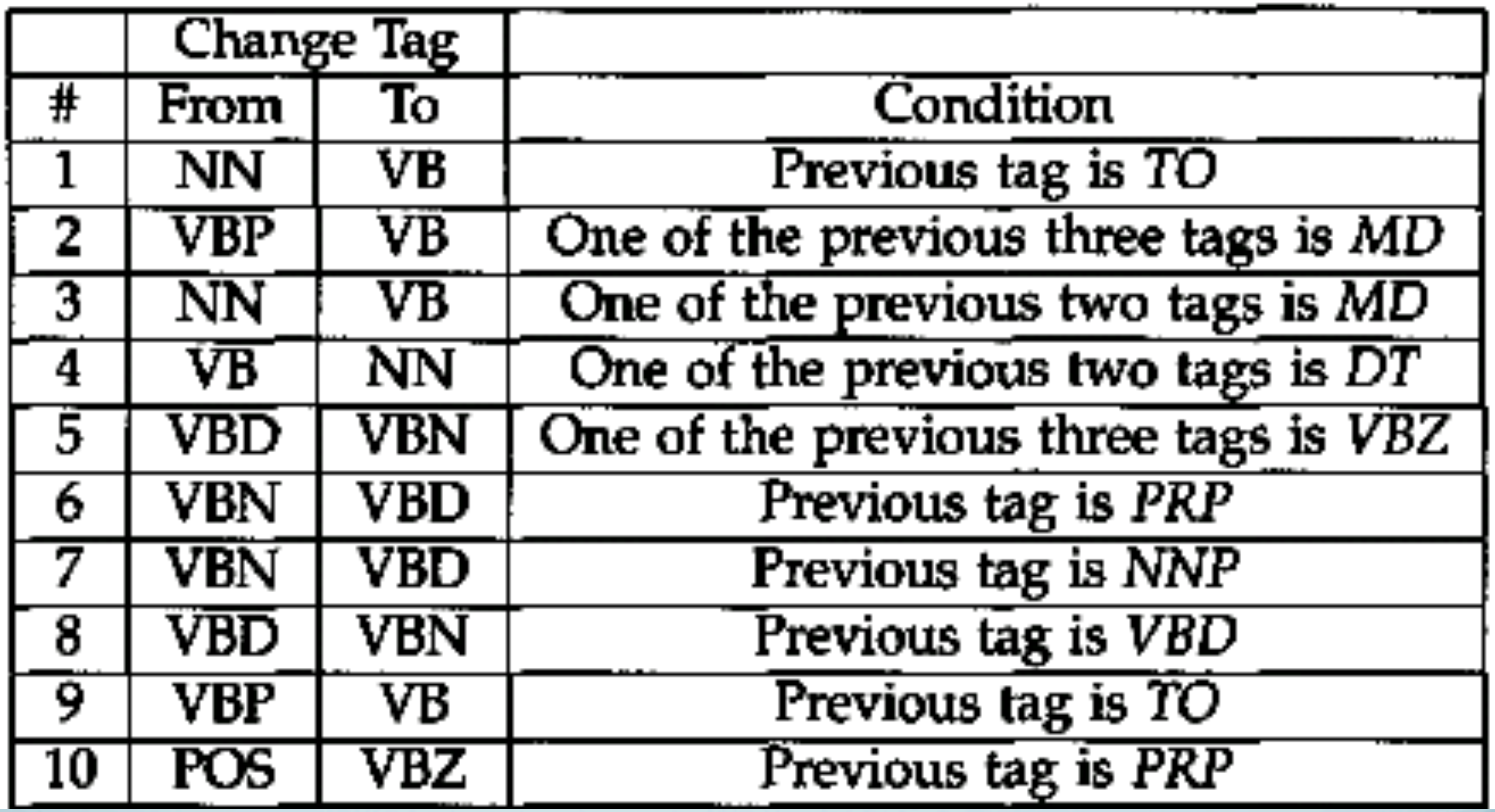
Lexicalized features

- w_0 is w.
- w_{-1} is w
- w_{-1} or w_{-2} is w
- t_{-1} is z and w_0 is w.
- …

TBL for POS tagging (cont)

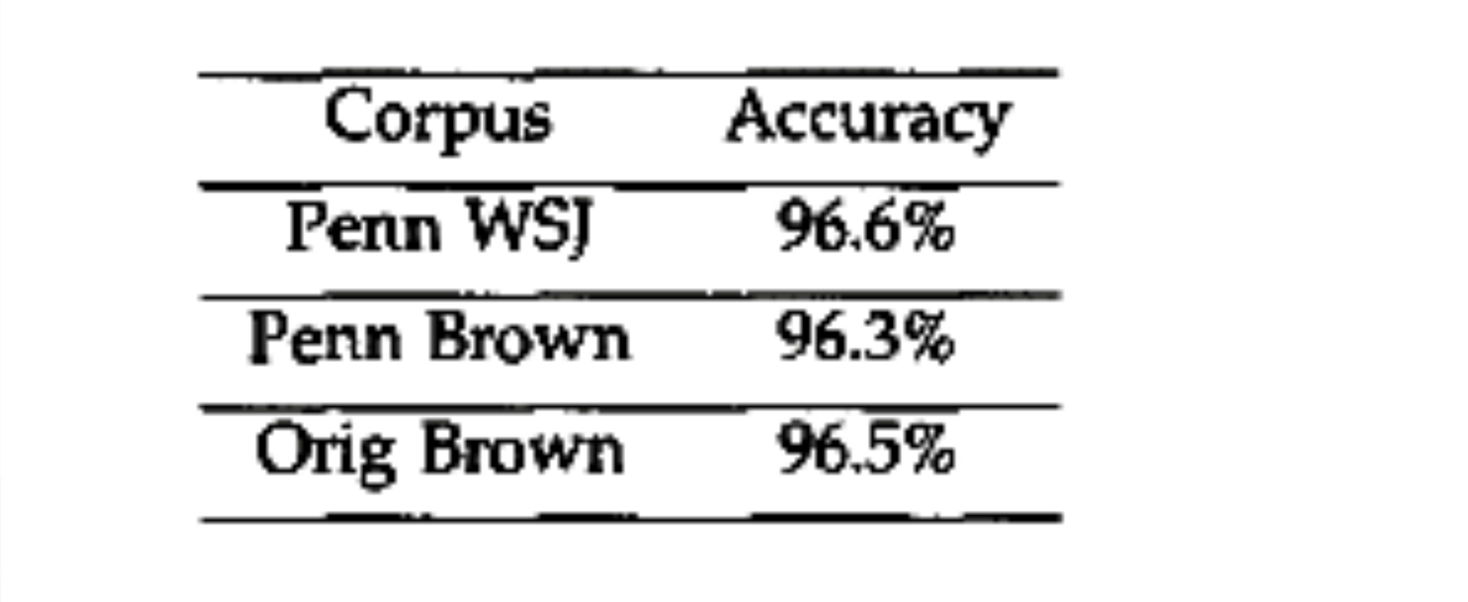
- The objective function: tagging accuracy
  - for comparing the corpus to the truth:
  - For choosing a transformation: choose the one that results in the greatest error reduction.
- The order of applying transformations: left-to-right.
- The results of applying transformations are not visible to other transformations.

Learned transformations



	Change Tag		
#	From	To	Condition
1	NN	VB	Previous tag is *TO*
2	VBP	VB	One of the previous three tags is *MD*
3	NN	VB	One of the previous two tags is *MD*
4	VB	NN	One of the previous two tags is *DT*
5	VBD	VBN	One of the previous three tags is *VBZ*
6	VBN	VBD	Previous tag is *PRP*
7	VBN	VBD	Previous tag is *NNP*
8	VBD	VBN	Previous tag is *VBD*
9	VBP	VB	Previous tag is *TO*
10	POS	VBZ	Previous tag is *PRP*

Experiments



Corpus	Accuracy
Penn WSJ	96.6%
Penn Brown	96.3%
Orig Brown	96.5%

Summary

Properties

- Existence of initial annotator
- Existence of current label: those labels are updated in each iteration.
- Can apply to sequence labeling and other problems
- Features can refer to the current label of **any** token in the sequence.

Strengths of TBL

- TBL is very different from other learners covered so far:
  - Existence of initial annotator.
  - Transformations are applied in sequence
  - Results of previous transformations are visible to following transformations.
  - Existance of current label ➔ It can handle dynamic problems well.
- TBL is more than a classifier
  - Classification problems: POS tagging
  - Other problems: e.g., parsing
- TBL performs well because it minimizes (training) errors directly.

Weaknesses of TBL

- Learning can be expensive → various methods

- TBL is not probabilistic, and it cannot produce topN hypotheses or confidence scores.