Part-of-speech tagging

Read Chapter 8 - Speech and Language Processing

Definition

- Part of Speech (pos) tagging is the problem of assigning each word in a sentence the part of speech that it assumes in that sentence.
  - Input: a string of words + a tagset
  - Output: a single best tag for each word

  - Example 1
  - Example 2
  - Example 3
  - Example 4
  - Example 5

➢Tagging makes parsing easier

The task of POS tagging

- A simple task (usually linear processing time), which can be used in many other applications:
  - Text-to-speech: record - N: [‘reko:d], V: [ri’ko:d]; lead – N [led], V: [li:d]
  - Can be a preprocessor for a parser (speeds up parser). The parser can do it better but more expensive
  - Speech recognition, parsing, information retrieval, etc.
  - Can be done by many different methods
- Easy to evaluate (how many tags are correct?)
- Canonical finite-state task
  - Can be done well with methods that look at local context
  - Though should “really” do it by parsing!

English word classes

- Closed class (function words): fixed membership
  - Prepositions: on, under, over,…
  - Particles: abroad, about, around, before, in, instead, since, without,…
  - Articles: a, an, the
  - Conjunctions: and, or, but, that,…
  - Pronouns: you, me, I, your, what, who,…
  - Auxiliary verbs: can, will, may, should,…

English word classes

- Open class
  - nouns
    - Proper nouns: IBM, Colorado
    - common nouns
      - count nouns: book, ticket
      - mass nouns: snow, salt
  - verbs
    - auxiliaries
    - . . .
  - adjectives
    - Color: red, white
    - Age: old, young
    - Value: good, bad
  - adverbs
    - Locatives adverbs: home, here, downhill
    - Degree adverbs: extremely, very, somewhat
    - Manner adverbs: slowly, delicately
    - Temporal adverbs: yesterday, Monday

Tagsets for English

- 87 tags - Brown corpus
- Three most commonly used:
  - Small: 45 Tags - Penn treebank (next slide)
  - Medium size: 61 tags, British national corpus
  - Large: 146 tags

Brown/Penn Treebank tags

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	$
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Example from Penn Treebank

- *The grand jury commented on a number of other topics .*

⇨ The/DT grand/JJ jury/NN commented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.

Problem of POS tagging

- Example 2
- Example 3

- Problem of POS tagging is to resolve ambiguities, choosing the proper tag for the context.

Main types of taggers

- **Stochastic tagging**: Maximum likelihood, Hidden Markov model tagging

  Pr (Det-N) > Pr (Det-Det)

- **Rule based tagging**

  If <some pattern>

  Then … <some part of speech> (allows for several passes)

Approaches to Tagging

- **HMM tagging** = The bold approach: 'Use all the information you have and guess'
- **Constrain Grammar (CG) tagging** = The cautious approach: 'Don't guess, just eliminate the impossible!'
- **Transmation-based (TB) tagging** = The whimsical approach: 'Guess first, then change your mind if nessessary!'

Stochastic POS tagging

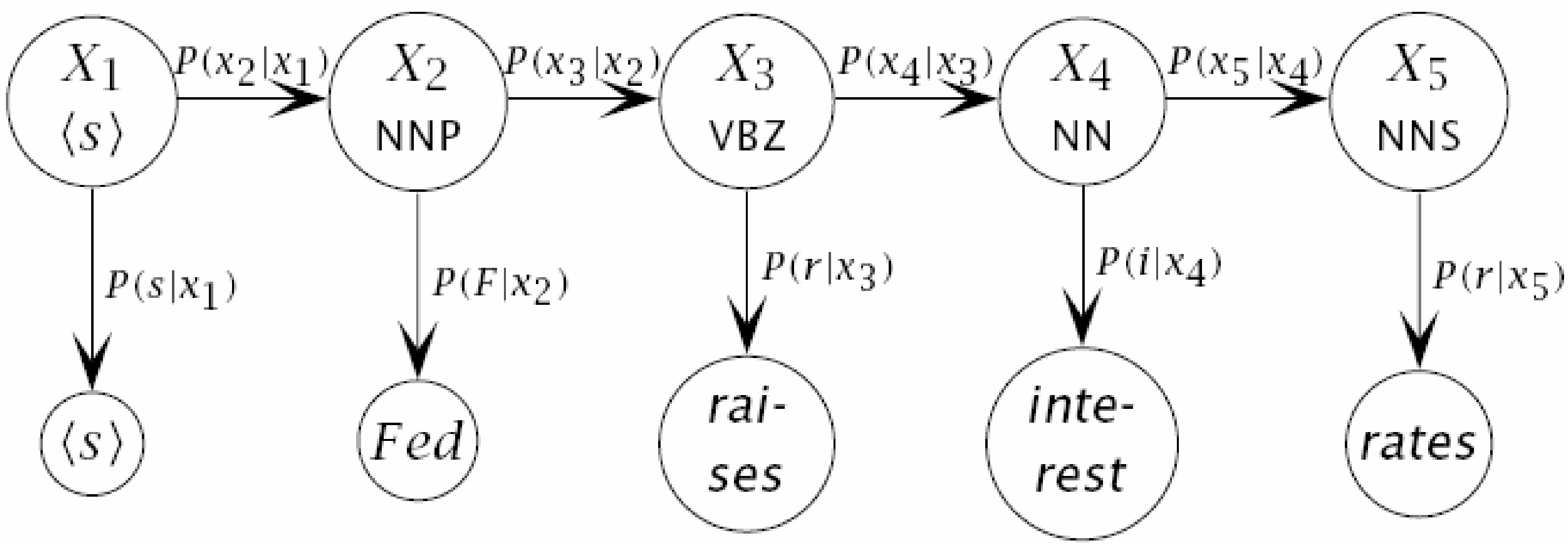
For a given sentence or word sequence, pick the most likely tag for each word.

How?

- A Hidden Markov model (HMM) tagger:
  - Choose the tag sequence that maximizes:

    $P(\text{word}|\text{tag}) \bullet P(\text{tag}|\text{previous } n \text{ tags})$

HMMs – POS example



- Top row is unobserved states, interpreted as POS tags
- Bottom row is observed output observations
- We normally do supervised training, and then inference to decide POS tags (Bayesian network style)

HMM tagging

- Bigram HMM Equation: choose t_i for w_i that is most probably given the t_{i-1} and w_i :

  $t_i = \text{argmax}_j\ P(t_j \mid t_{i-1}, w_i)$ (1)

- A HMM simplifying assumption: the tagging problem can be solved by looking at nearby words and tags.

- $t_i = \text{argmax}_j\ P(t_j \mid t_{j-1})P(w_i \mid t_j)$ (2)

pr tag sequence
(tag co-occurrence)

word (lexical) likelihood

Example

1. Secretariat/NNP is/VBZ expected/VBN to/TO race/VB tomorrow/NN

2. People/NNS continue/VBP to/TO inquire/VB the/DT reason/NN for/IN the/DT race/NN for/IN outer/JJ space/NN

Suppose we have tagged all but race

- Look at just preceding word (bigram):
  to/TO race/??? NN or VB?
  the/DT race/???

- Applying (2): $t_i = \text{argmax}_j \, P(t_j \mid t_{j-1}) P(w_i \mid t_j)$

- Choose tag with greater of the two probabilities:
  P(VB|TO)P(race|VB) or P(NN|TO)P(race|NN)

Calculate Pr

Let's consider *P*(VB|TO) and *P*(NN|TO)

- Can find these pr estimates by *counting* in a corpus (and normalizing)
- Expect that a verb is more likely to follow TO than a Noun is, since infinitives *to race, to walk,* are common in English. A noun *can* follow TO, *run to school*
- From the Brown corpus
  - *P*(NN|TO)= .021
  - *P*(VB|TO)= .340

Calculate Pr

- *Now P*(race|VB) and *P*(race|NN): the *lexical likelihood* of the noun *races* given each tag, *P*(race|VB) and *P*(race|NN), e.g., "if we were expecting a verb, would it be *race?*"
- From the Brown corpus
  - *P*(race|NN)= 0.00041
  - *P*(race|VB)= 0.00003

1. *P*(VB|TO)*P*(race|VB)= 0.00001
2. *P*(NN|TO)*P* (race|NN)= 0.000007

➢ *race should be a* VB *after* "TO"

The full model

- Now we want the best *sequence* of tags for the whole sentence
- Given the sequence of words, *W*, we want to compute the most probably tag sequence, $T=t_1, t_2, \ldots, t_n$ or,

$$\hat{T} = \underset{T \in \tau}{\arg\max}\, P(T \mid W)$$

$$= \underset{T \in \tau}{\arg\max} \frac{P(T)P(W \mid T)}{P(W)} \qquad \text{(Bayes' Theorem)}$$

$$= \underset{T \in \tau}{\arg\max}\, P(T)P(W \mid T)$$

Expand this using chain rule

- From chain rule for probabilities:

P(A,B) = P(A|B)P(B) = P(B|A)P(A)

P(A,B,C) = P(B,C|A)P(A) = P(C|A,B)P(B|A)P(A)

= P(A)P(B|A)P(C|A,B)

P(A,B,C,D…) = P(A)P(B|A)P(C|A,B)P(D|A,B,C..)

$$P(T)P(W \mid T) = \prod_{i=1}^{n} \underbrace{P(w_i \mid w_1 t_1 \ldots w_{i-1} t_{i-1} t_i)}_{\text{pr word}} \underbrace{P(t_i \mid w_1 t_1 \ldots w_{i-1} t_{i-1})}_{\text{tag history}}$$

$$\Updownarrow$$

$P(w_1)P(w_2|w_1)P(w_3|w_2w_1)\ldots.$

Make simplifying trigram assumption to approximate these 2 factors:

- Probability of a word depends only on its tag

$$P(w_i \mid w_1 t_1 \ldots t_{i-1} t_i) = P(w_i \mid t_i)$$

- Tag history approximated by two most recent tags (trigram: two most recent + current state)

$$P(t_i \mid w_1 t_1 \ldots t_{i-1}) = P(t_i \mid t_{i-2} t_{i-1})$$

Replacing to the equation

P(T)P(W|T) =

$$P(t_1)P(t_2 \mid t_1)\prod_{i=3}^{n} P(t_i \mid t_{i-2}t_{i-1})[\prod_{i=1}^{n} P(w_i \mid t_i)]$$

We estimate these from *counts on corpus*

- We can do a *maximum likelihood estimate* by using relative frequencies from corpus to estimate these probabilities:

$$P(t_i \mid t_{i-1} t_{i-2}) = \frac{c(t_{i-2} t_{i-1} t_i)}{c(t_{i-2} t_{i-1})}$$

$$P(w_i \mid t_i) = \frac{c(w_i, t_i)}{c(t_i)}$$

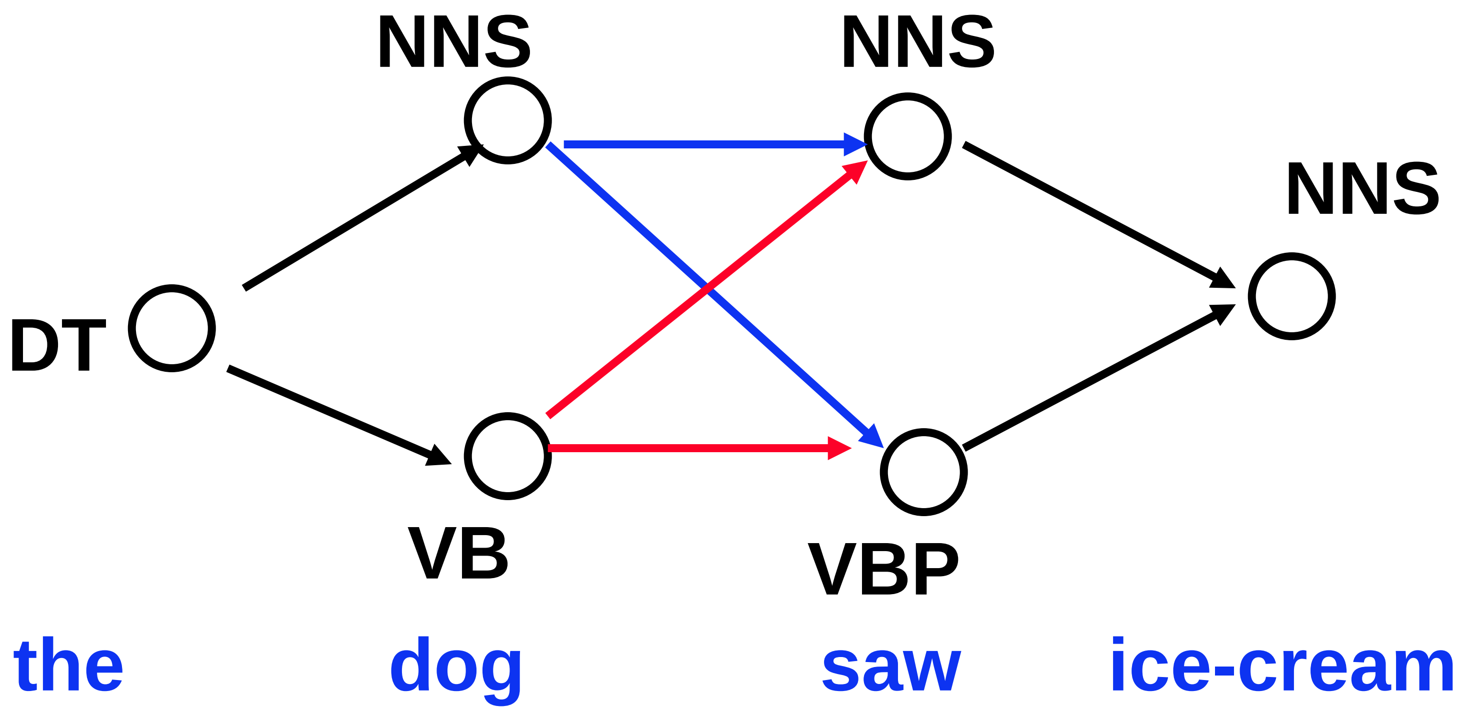
Problem

- The problem to solve:

$$\hat{T} = \arg\max_{T \in \tau} P(T)P(W \mid T)$$

- All P(T)P(W|T) can now be computed

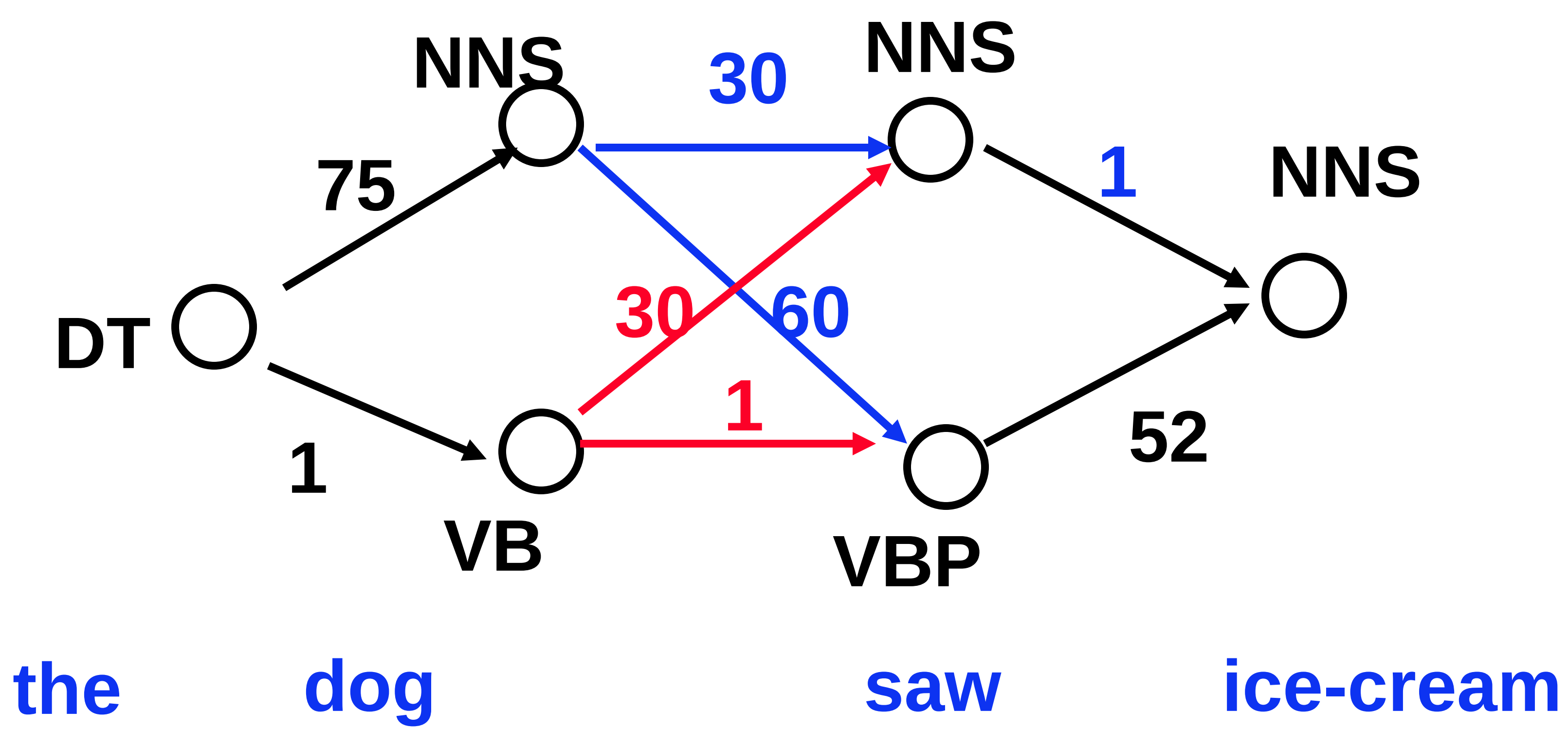
Example



How do we find maximum (best) path?

- we want efficient way to go through this

The counts add scores - we want to find the maximum scoring path



How do we find maximum (best) path?

- We use best-first (A*) search, as in AI…
  1. At each step, k best values ($\hat{T}$) are chosen. Each of the k values corresponds to one possible tagging combination of the visited words.
  2. When tagging the next word, recompute probabilities. Go to step 1.
- Advantage: fast (do not need to check all possible combinations, but only k potential ones).
- Disadvantage: may not return the best solution, but only acceptable results.

Accuracy

- Accuracy of this method > 96%
- Baseline? **90%**
  - Baseline is performance of stupidest possible method
  - Tag every word with its most frequent tag
  - Tag unknown words as nouns
- Human: **97%+/- 3%**; if discuss together: **100%**

How do we find maximum (best) path?

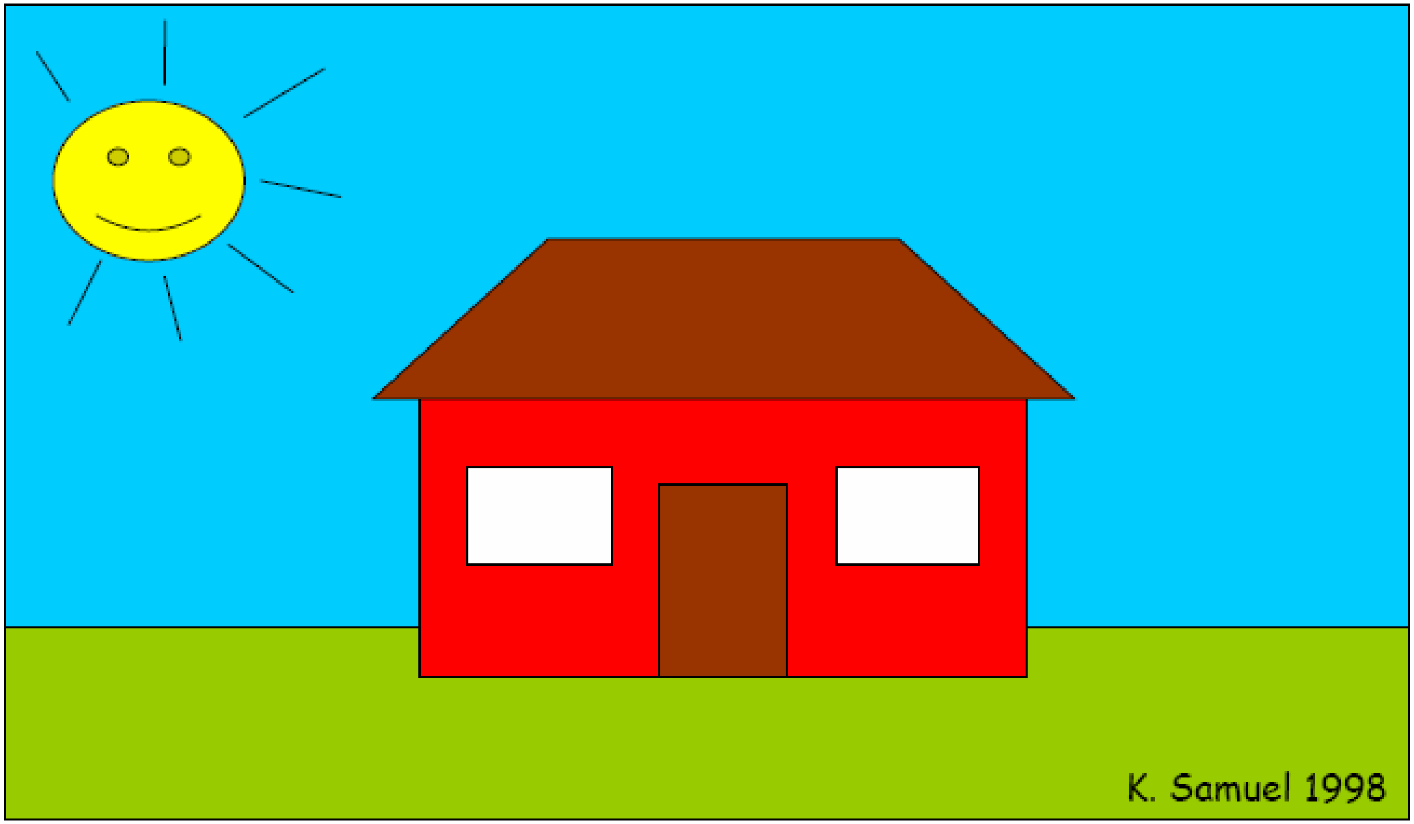
- Suppose we don’t have training data?
- Can estimate roughly:
  - start with uniform probabilities,
  - use EM algorithm to re-estimate from counts - try labeling with current estimate,
  - use this to correct estimate

➢ Not work well, a small amount of hand-tagged training data improves the accuracy

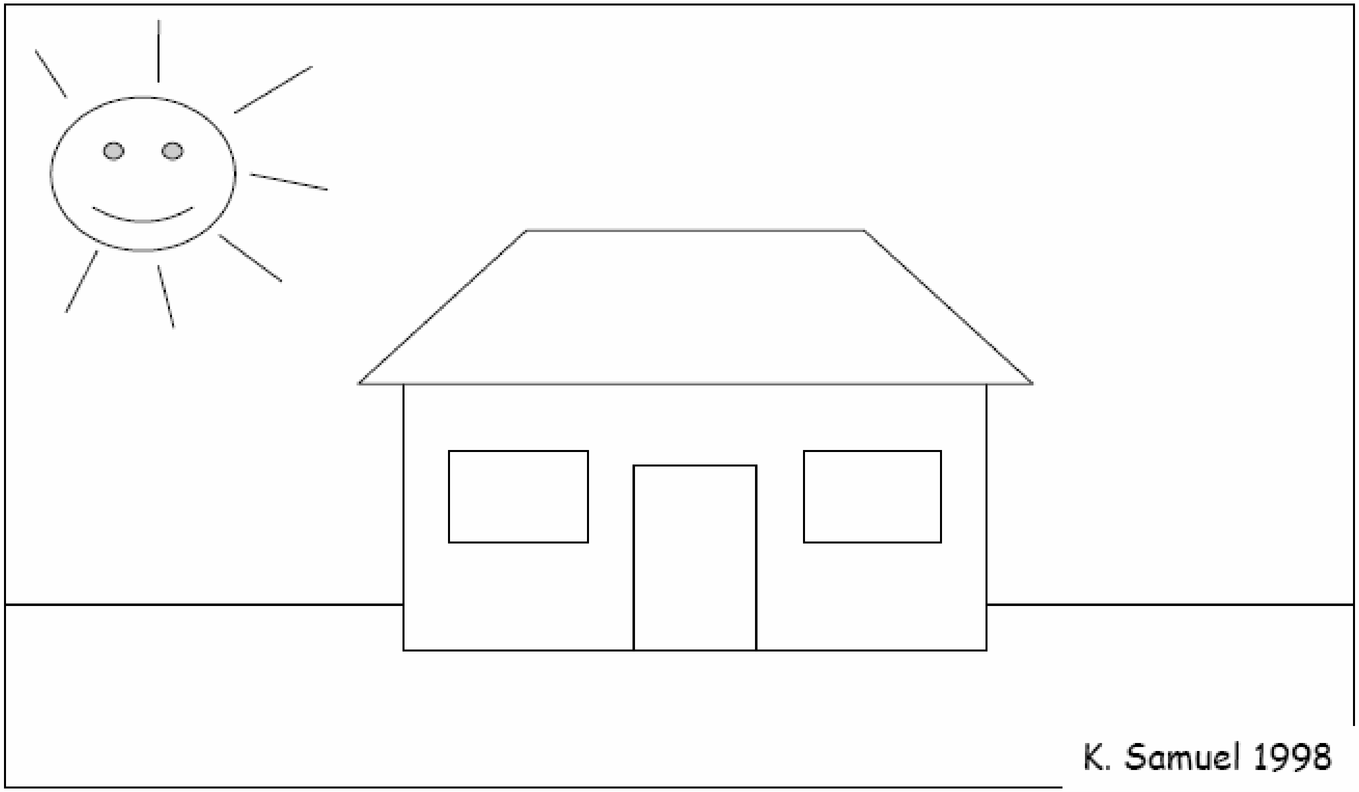
Second approach: transformation-based tagging (TBL)

- Combines symbolic and stochastic approaches: uses machine learning to refine its tags, via several passes
- Tag using a broadest (most general) rule; then an narrower rule, that changes a smaller number of tags, and so on.

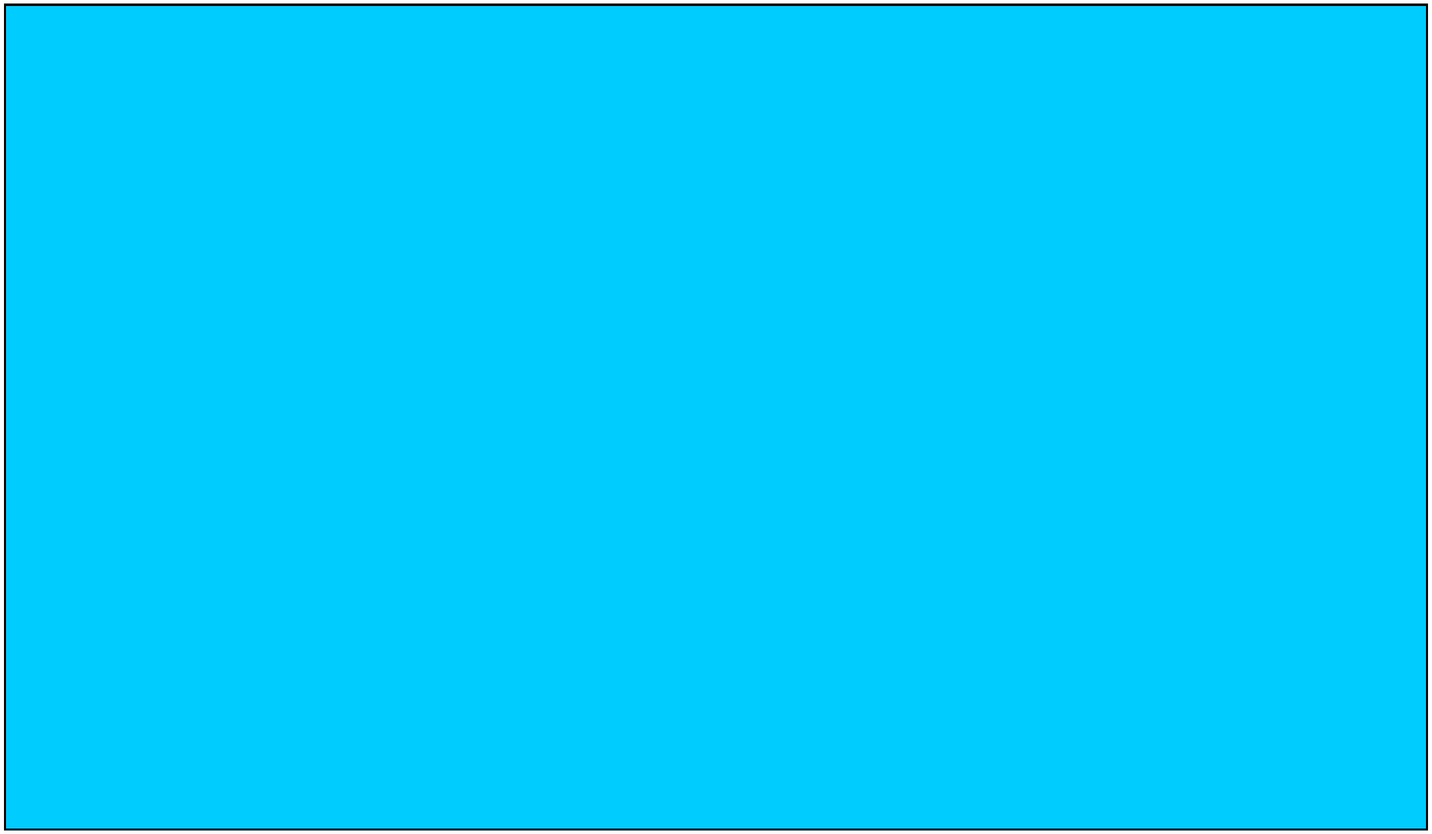
Transformation-based painting



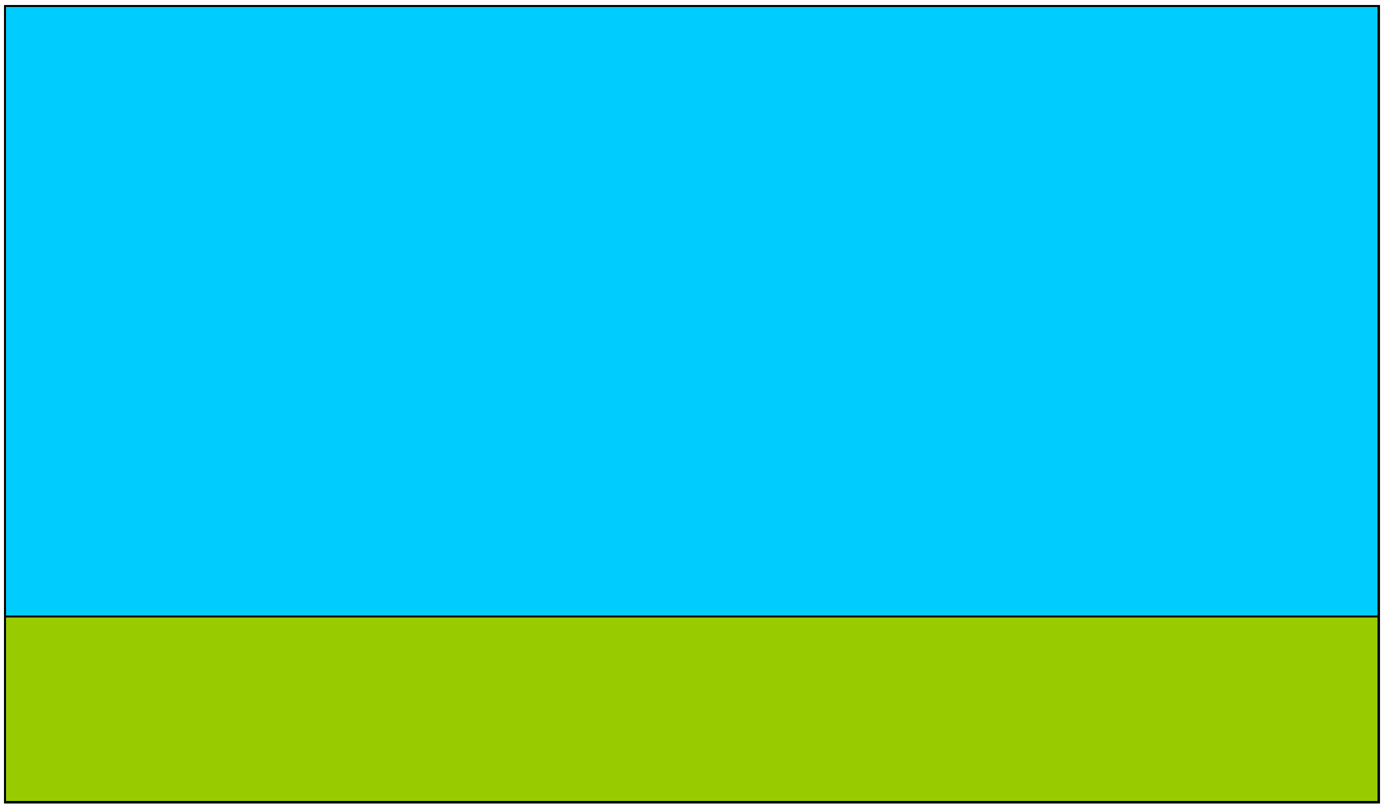
Transformation-based painting



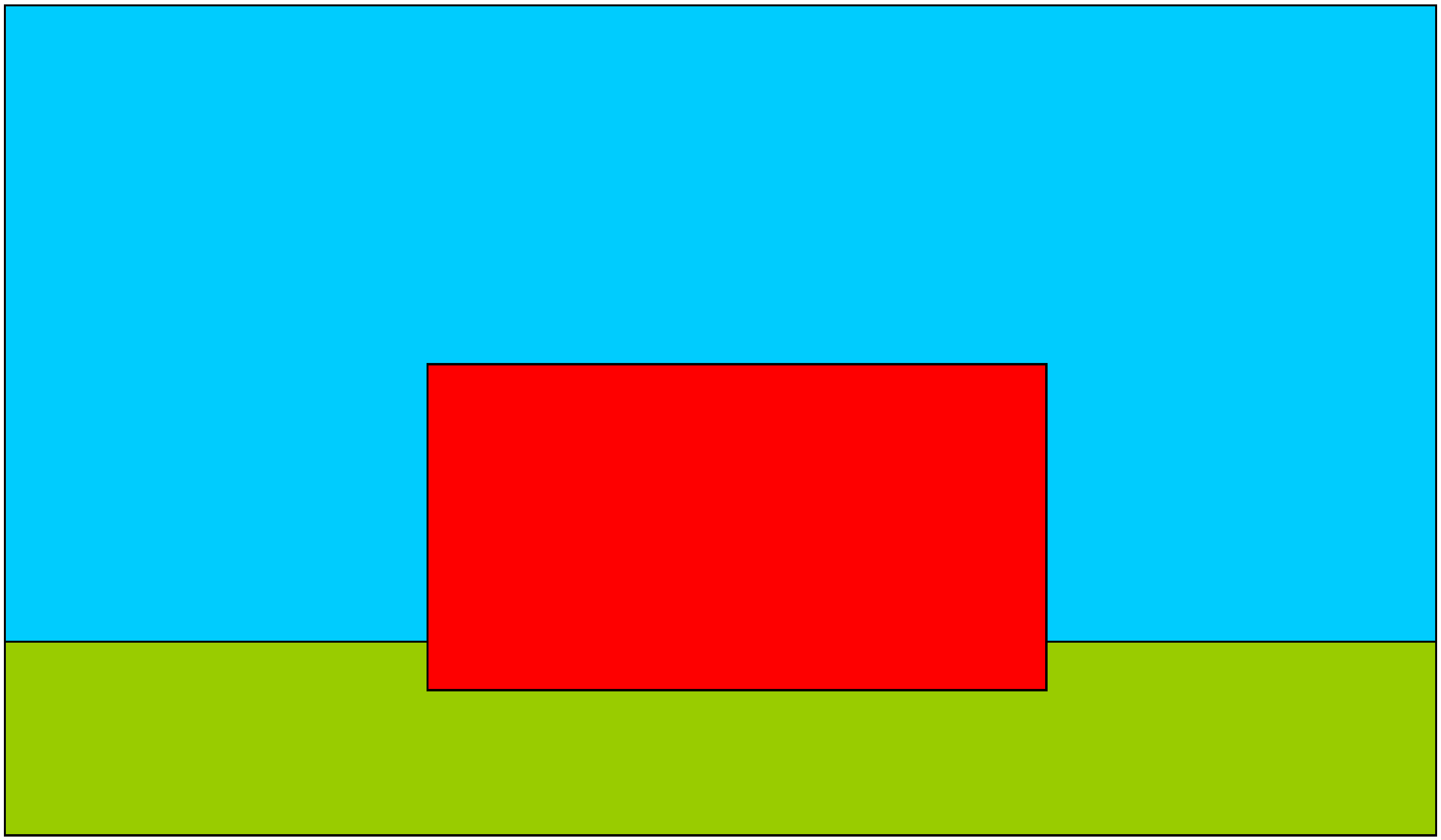
Transformation-based painting



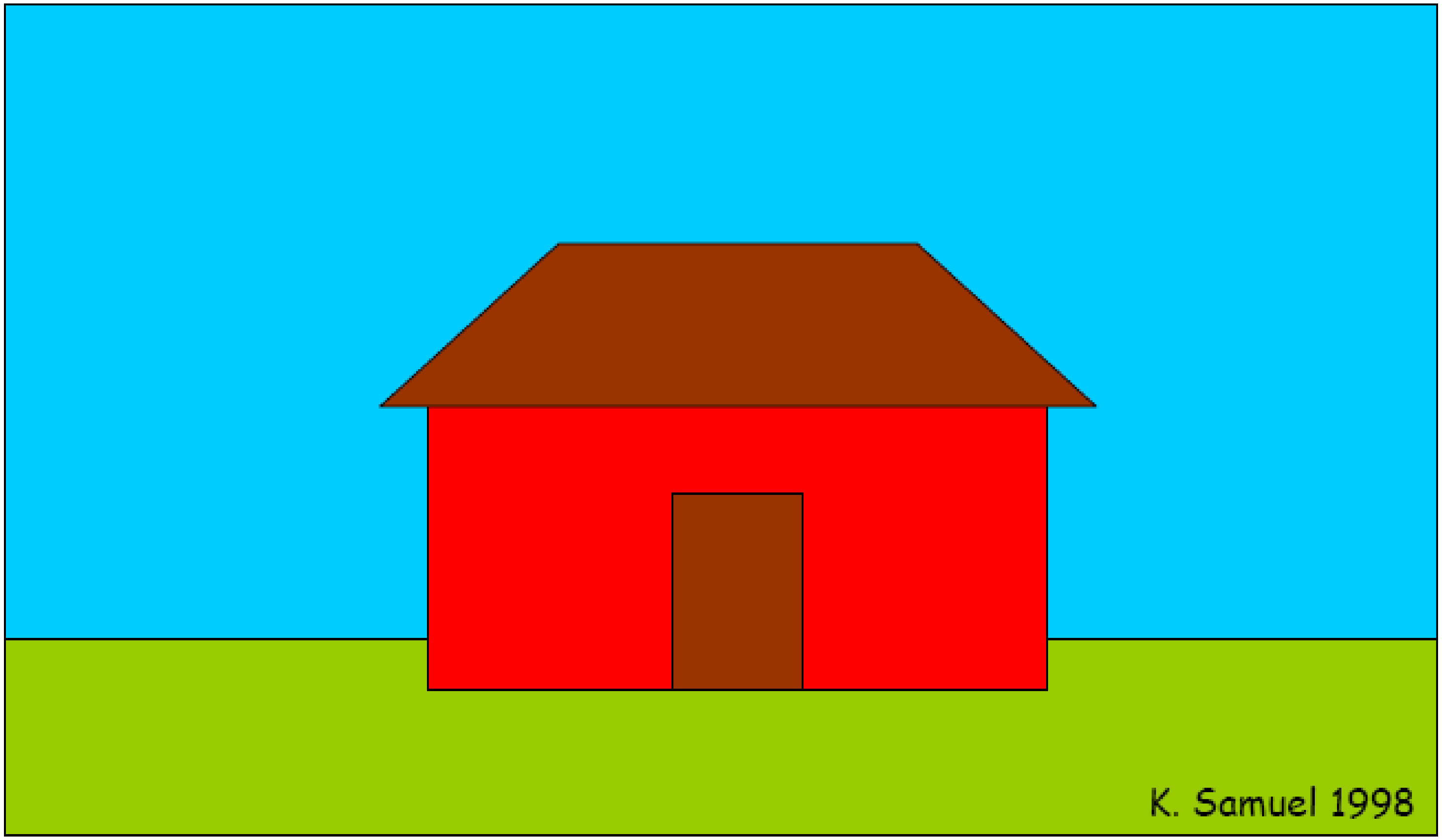
Transformation-based painting



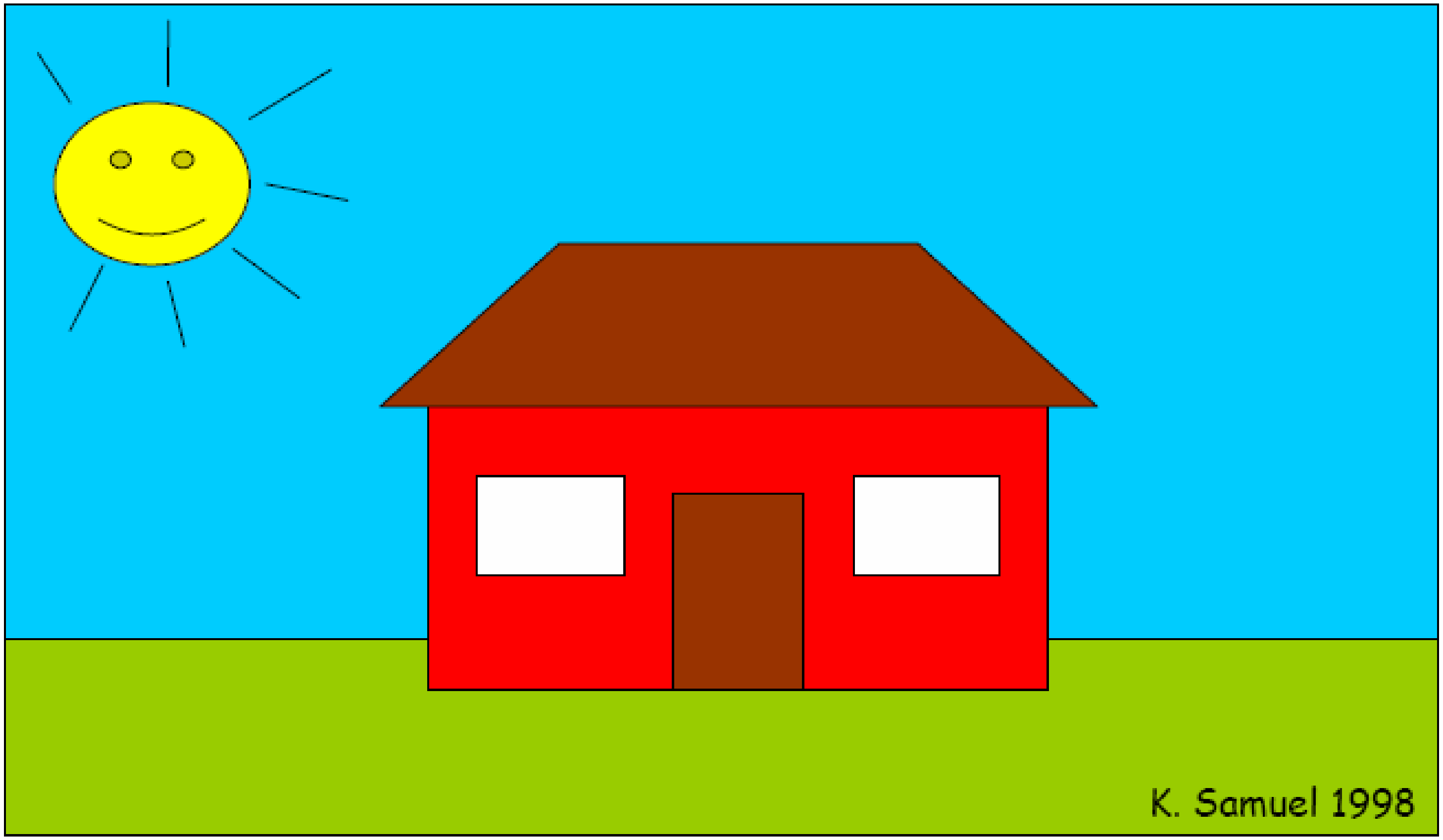
Transformation-based painting



Transformation-based painting



Transformation-based painting



How does the TBL system work?

lexicon

```
data:NN
decided:VB
her:PN
she:PN N
table:NN VB
to:TO
```

rules

```
pos:NN>VB <- pos:TO@[-1] o
pos:VB>NN <- pos:DT@[-1] o
....
```

input

```
She decided to  table her data
NP  VB      TO  NN/VB PN  NN
```

How does the TBL system work?

1. First label every word with its most-likely tag (as we saw, this gets 90% right…!) for example, in Brown corpus, *race* is most likely to be a Noun:

   P(NN|race)= 0.98

   P(VB|race)= 0.02

2. …expected/VBZ to/TO race/VB tomorrow/NN

   …the/DT race/NN for/IN outer/JJ space/NN

3. Use transformational (learned) rules to change tags:

   Change **NN** *to* **VB** *when the previous tag is* **TO**

   pos: ‘NN’>’VB’ ← pos: ‘TO’ @[-1] o

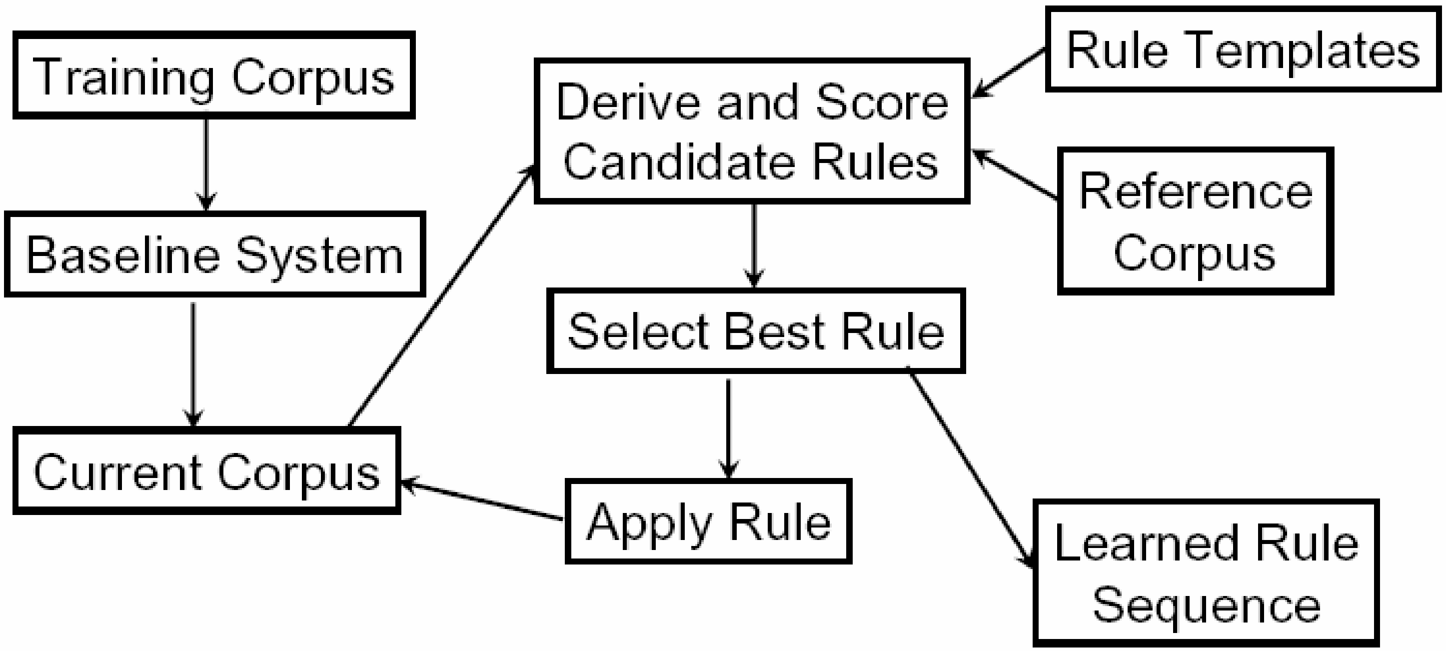
Rules for POS tagging

```
pos:'NN'>'VB' <- pos:'TO'@[-1] o
pos:'VBP'>'VB' <- pos:'MD'@[-1,-2,-3] o
pos:'NN'>'VB' <- pos:'MD'@[-1,-2] o
pos:'VB'>'NN' <- pos:'DT'@[-1,-2] o
pos:'VBD'>'VBN' <- pos:'VBZ'@[-1,-2,-3] o
pos:'VBN'>'VBD' <- pos:'PRP'@[-1] o
pos:'POS'>'VBZ' <- pos:'PRP'@[-1] o
pos:'VB'>'VBP' <- pos:'NNS'@[-1] o
pos:'IN'>'RB' <- wd:as@[0] & wd:as@[2] o
pos:'IN'>'WDT' <- pos:'VB'@[1,2] o
pos:'VB'>'VBP' <- pos:'PRP'@[-1] o
pos:'IN'>'WDT' <- pos:'VBZ'@[1] o
...
```

Rules for POS tagging

```
NN VB PREVTAG TO
VB VBP PREVTAG PRP
VBD VBN PREV1OR2TAG VBD
VBN VBD PREVTAG PRP
NN VB PREV1OR2TAG MD
VB VBP PREVTAG NNS
VB NN PREV1OR2TAG DT
VBN VBD PREVTAG NNP
VBD VBN PREV1OR2OR3TAG VBZ
IN DT PREVTAG IN
VBP VB PREV1OR2OR3TAG MD
IN RB WDAND2AFT as as
VBD VBN PREV1OR2TAG VB
RB JJ NEXTTAG NN
VBP VB PREV1OR2OR3TAG TO
POS VBZ PREVTAG PRP
NN VBP PREVTAG PRP
DT PDT NEXTTAG DT
…
```

Learning TB rules in TBL system



Stop when score of best rule falls below threshold.

Various Corpora

- Training corpus

  w0 w1 w2 w3 w4 w5 w6 w7 w8 w9 w10
- Current corpus (CC 1)

  dt vb nn dt vb kn dt vb ab dt vb
- Reference corpus

  dt nn vb dt nn kn dt jj kn dt nn

Rule Templates

- In TBL, only rules that are instances of *templates* can be learned.
- For example, the rules

  tag:'VB'>'NN' ← tag:'DT'@[-1].

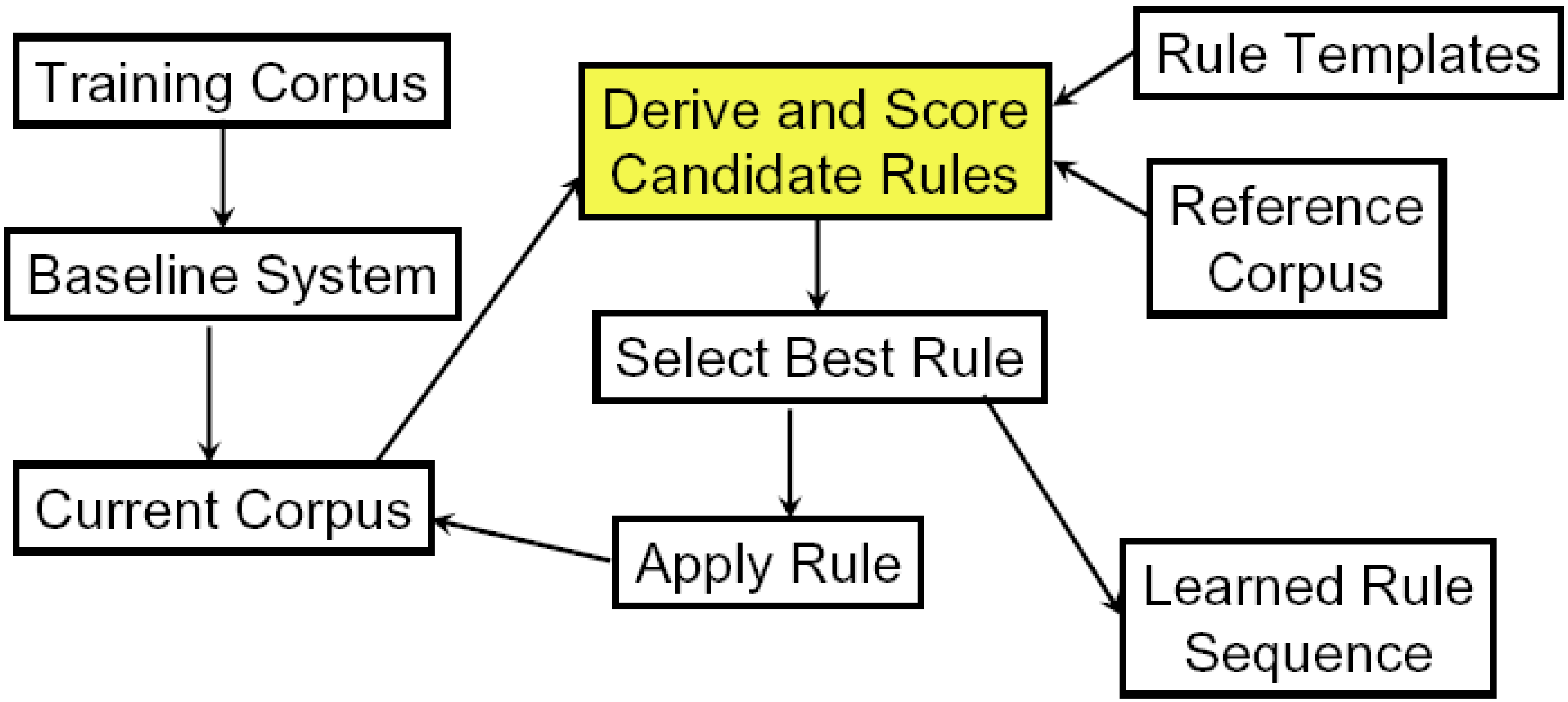
  tag:’NN’>’VB' ← tag:'DT'@[-1].

  are instances of the template

  tag:A>B ← tag:C@[-1].
- Alternative syntax using anonymous variables

  tag:_>_ ← tag:_@[-1].

Learning TB rules in TBL system



Score, Accuracy and Thresholds

- The *score* of a rule is the number of its positive matches minus the number of its negative instances:

$$\text{score}(R) = |\text{pos}(R)| - |\text{neg}(R)|$$

- The *accuracy* of a rule is its number of positive matches divided by the total number of matches of the rule:

$$\text{accuracy}(R) = \frac{|\text{pos}(R)|}{|\text{pos}(R)| + |\text{neg}(R)|}$$

- The *score threshold* and the *accuracy threshold* are the lowest score and the lowest accuracy, respectively, that the highest scoring rule must have in order to be considered.
- In *ordinary* TBL, we work with an accuracy threshold < 0.5.

Derive and Score Candidate Rule 1

- Template = tag:_>_ ← tag:_@[-1]
- R1 = tag:vb>nn ← tag:dt@[-1]

CC i	**dt**	**vb**	nn	**dt**	**vb**	kn	**dt**	**vb**	ab	**dt**	**vb**
CC i+1	dt	**nn**	nn	dt	**nn**	kn	dt	**nn**	ab	dt	**nn**

Ref. C	dt	nn	vb	dt	nn	kn	dt	jj	kn	dt	nn

- pos(R1) = 3
- neg(R1) = 1
- score(R1) = pos(R1) - neg(R1) = 3-1 = 2

Derive and Score Candidate Rule 2

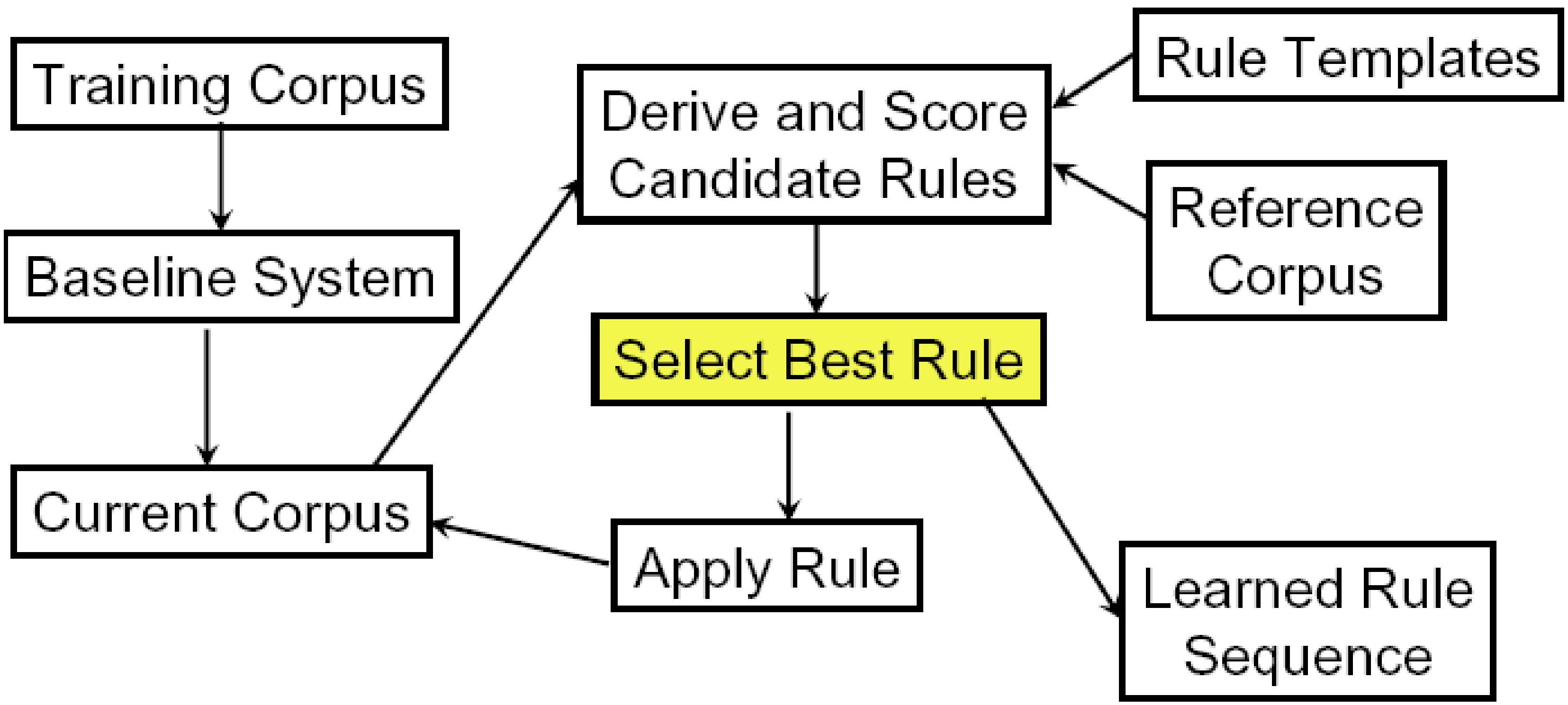
- Template = tag:_>_ ← tag:_@[-1]
- R2 = tag:nn>vb ← tag:vb@[-1]

CC i	dt	**vb**	**nn**	dt	vb	kn	dt	vb	ab	dt	vb
CC i+1	dt	vb	**vb**	dt	vb	kn	dt	vb	ab	dt	vb

Ref. C	dt	nn	vb	dt	nn	kn	dt	nn	kn	dt	nn

- pos(R2) = 1
- neg(R2) = 0
- score(R2) = pos(R2) - neg(R2) = 1-0 = 1

Learning TB rules in TBL system



Stop when score of best rule falls below threshold.

Select Best Rule

- Current ranking of rule candidates

  R1 = tag:vb>nn $\leftarrow$ tag:dt@[-1] Score = 2

  R2 = tag:nn>vb $\leftarrow$ tag:vb@[-1] Score = 1

  …
- If score threshold =< 2 then select R1, else if score threshold > 2, terminate.

Select Best Rule Optimizations

- **Reduce some of the naïve generate-and-test behaviour**: We only need to generate candidate rules that have at least one match in the training data.
- **Incremental evaluation**: Keep track of the leading rule candidate. If the number of positive matches of a rule is less than the score for the leading rule, we don't need to count the negative matches.

Greedy Best-First Search

- h(n) = estimated cost of the cheapest path from the state represented by the node n to a goal state
- Best-first search with h as its evaluation function
- NB: Greedy best-first search is not necessarily optimal

Advantages of TB Tagging

- Transformation rules can be created/edited manually
- Sequences of transformation rules have a declarative, logical semantics
- TB taggers are simple to implement
- Transformation-based taggers can be extremely fast (but then implementation is more complex)

Error analysis: what's hard for taggers

- Common errors (> 4%)
  - NN (common noun) vs .NNP (proper noun) vs. JJ (adjective): hard to distinguish; important to distinguish especially for information extraction
  - RP vs. RB vs IN: all can appear in sequences immediate after verb
  - VBD vs. VBN vs. JJ: distinguish past tense, past participles (*raced* vs. *was raced* vs. *the out raced horse*)

Most powerful unknown word detectors

- 3 inflectional endings (*-ed, -s, -ing*); 32 derivational endings (*-ion,* etc.); capitalization; hyphenation
- More generally: should use morphological analysis! (and some kind of machine learning approach)