NPFL147

Statistical Methods in Natural Language Processing

10. Tagging

Pavel Pecina, Jindřich Helcl

6 January, 2026

Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics

Course Segments

1. Introduction, probability, essential information theory
2. Statistical language modelling (n-gram)
3. Statistical properties of words
4. Word representations
5. Hidden Markov models, Tagging

Recap from Last Week

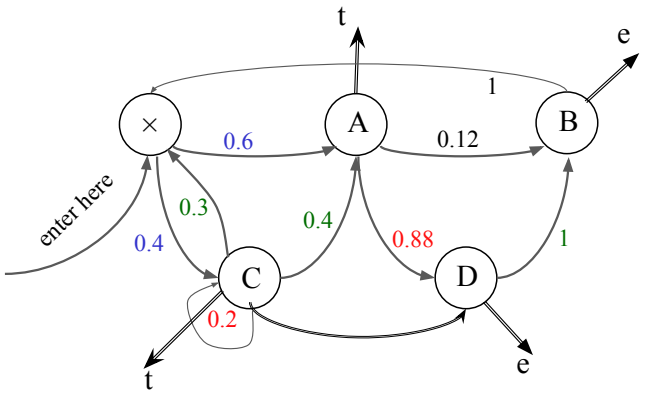
HMM: The Two Tasks (Plus One)

- HMM (the general case): five-tuple (S, S_0, Y, P_S, P_Y), where:
  - $S = \{s_1, s_2, \ldots, s_T\}$ is the set of states, S_0 is the initial state,
  - $Y = \{y_1, y_2, \ldots, y_V\}$ is the output alphabet,
  - $P_S(s_j|s_i)$ is the set of prob. distributions of transitions,
  - $P_Y(y_k|s_i, s_j)$ is the set of output (emission) probability distributions

- Given an HMM & an output sequence $Y = \{y_1, y_2, \ldots, y_k\}$:
  - (Task 1) compute the probability of Y;
  - (Task 2) compute the most likely sequence of states which has generated Y.
  - (Task 3) Estimating the parameters (transition/output distributions)

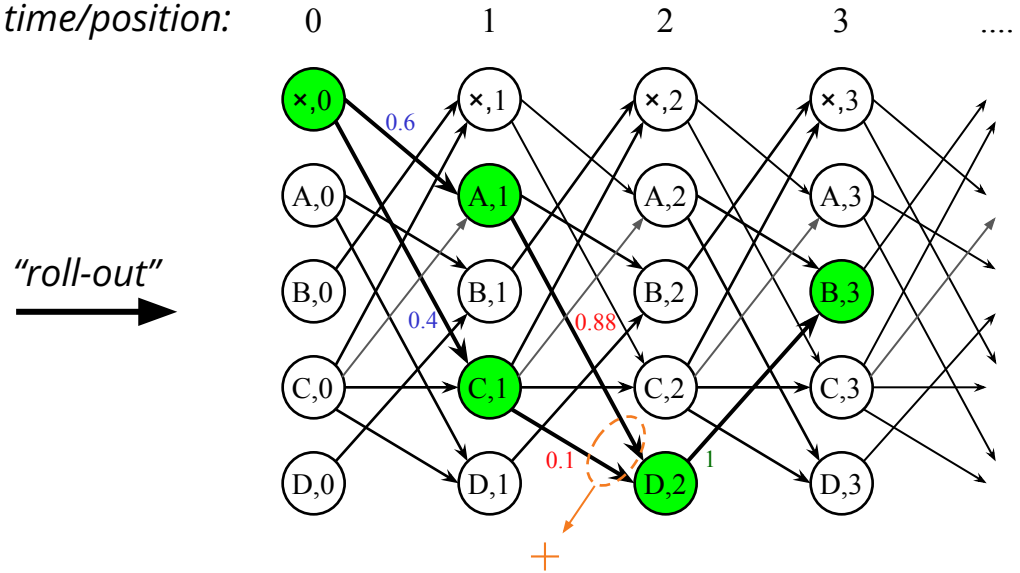
Trellis: HMM “roll-out”

HMM:



$p(\text{toe}) = 0.6 \times 0.88 \times 1 + 0.4 \times 1 \times 1 \cong 0.568$

Trellis:



- Trellis state: (HMM state, position)
- each state: holds ***one*** number (prob): **α**
- probability of Y: Σα in the last state

Y: t o e

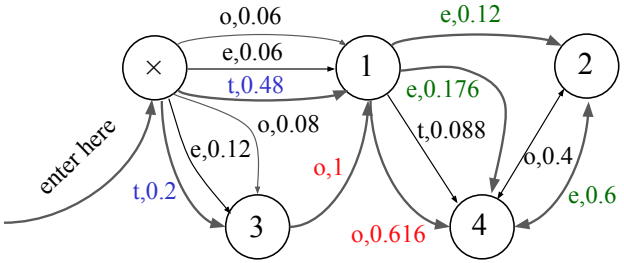
$\alpha(\times,0) = 1$ $\alpha(A,1) = 0.6$ $\alpha(C,1) = 0.4$ $\alpha(D,2) = 0.568$ $\alpha(B,3) = 0.568$

Trellis: The General Case (still, bigrams)

- Start as usual:
  - start state (×), set its $\alpha(\times,0)$ to 1.

position/stage: 0

$\alpha(\times,0) = 1$



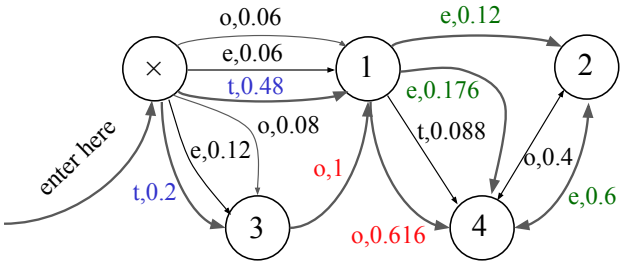
p(toe) = 0.48×0.616×0.6+
0.2×1×0.176 +
0.2×1×0.12 ≅ 0.237

Y:

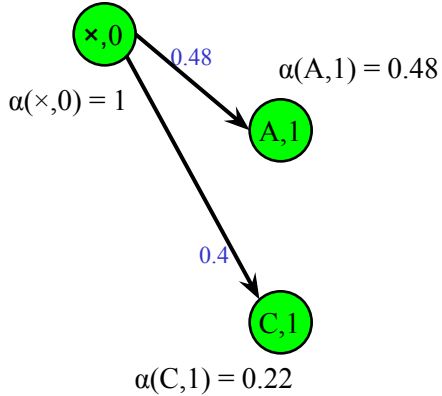
General Trellis: The Next Step

We are in stage i :

- Generate the next stage i+1 as before (except now arcs generate output, thus use only those arcs marked by the output symbol y_{i+1})
- For each generated *state*, compute $\alpha(state,i+1) = \Sigma_{\text{inc. arcs}} P_Y(y_{i+1}|state, prev.state) \times \alpha(prev.state, i)$



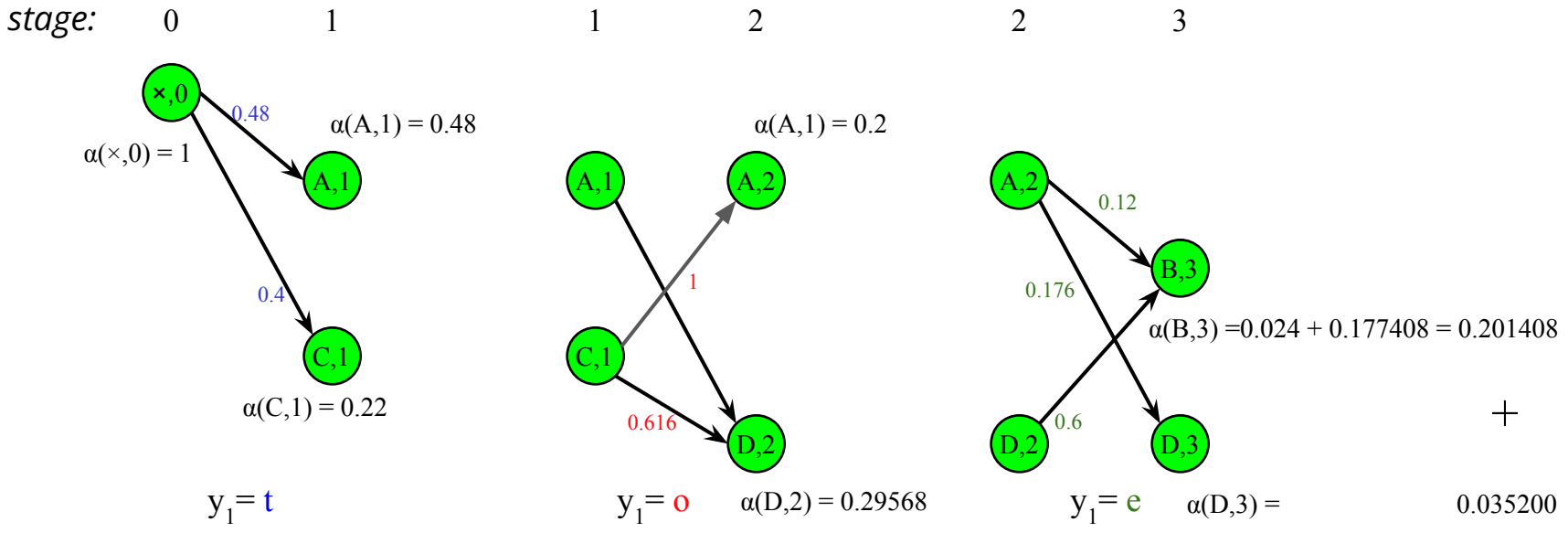
position/stage: 0 1

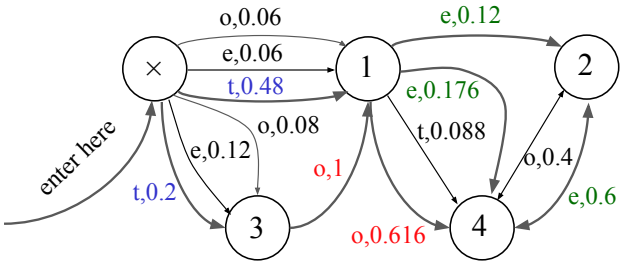


Y: t

And forget about the previous state ...

Trellis: The Complete Example

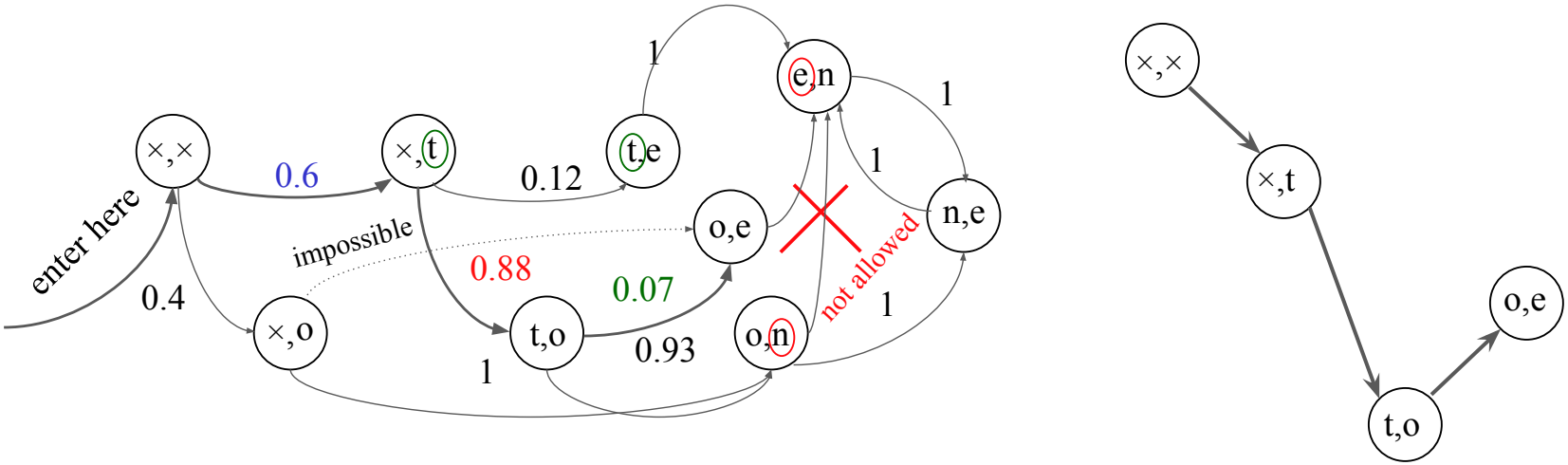




p(toe) = 0.48×0.616×0.6+
0.2×1×0.176 +
0.2×1×0.12 ≅ 0.237

The Case of Trigrams

- Like before, but:
  - states correspond to bigrams
  - output function always emits the second output symbol of the pair (state) to which the arc goes:



- Multiple paths not possible → trellis not really needed

The Viterbi Algorithm

- Solving the task of finding the most likely sequence of states which generated the observed data
- i.e., finding

$$S_{best} = \text{argmax}_S P(S|Y)$$

which is equal to (Y is constant and thus P(Y) is fixed):

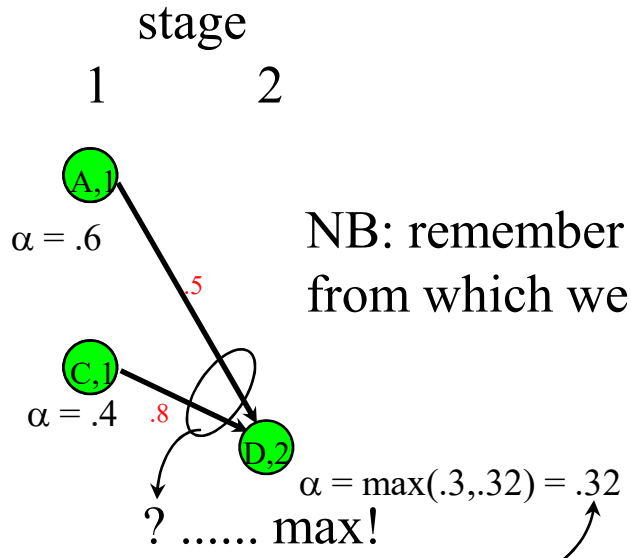
$$S_{best} = \text{argmax}_S P(S,Y) =$$

$$= \text{argmax}_S P(s_0,s_1,s_2,...,s_k,y_1,y_2,...,y_k) =$$

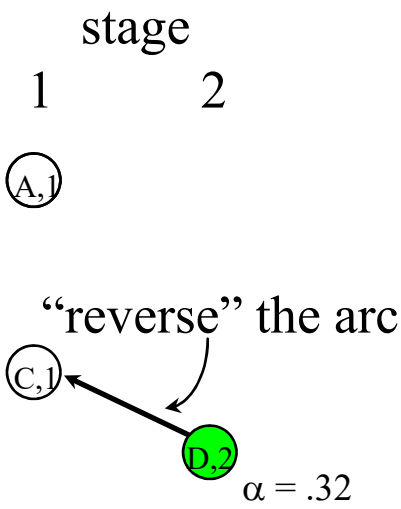
$$= \text{argmax}_S \Pi_{i=1..k}\, p(y_i|s_i,s_{i-1})p(s_i|s_{i-1})$$

The Crucial Observation

- Imagine the trellis build as before (but do not compute the αs yet; assume they are o.k.); stage *i*:

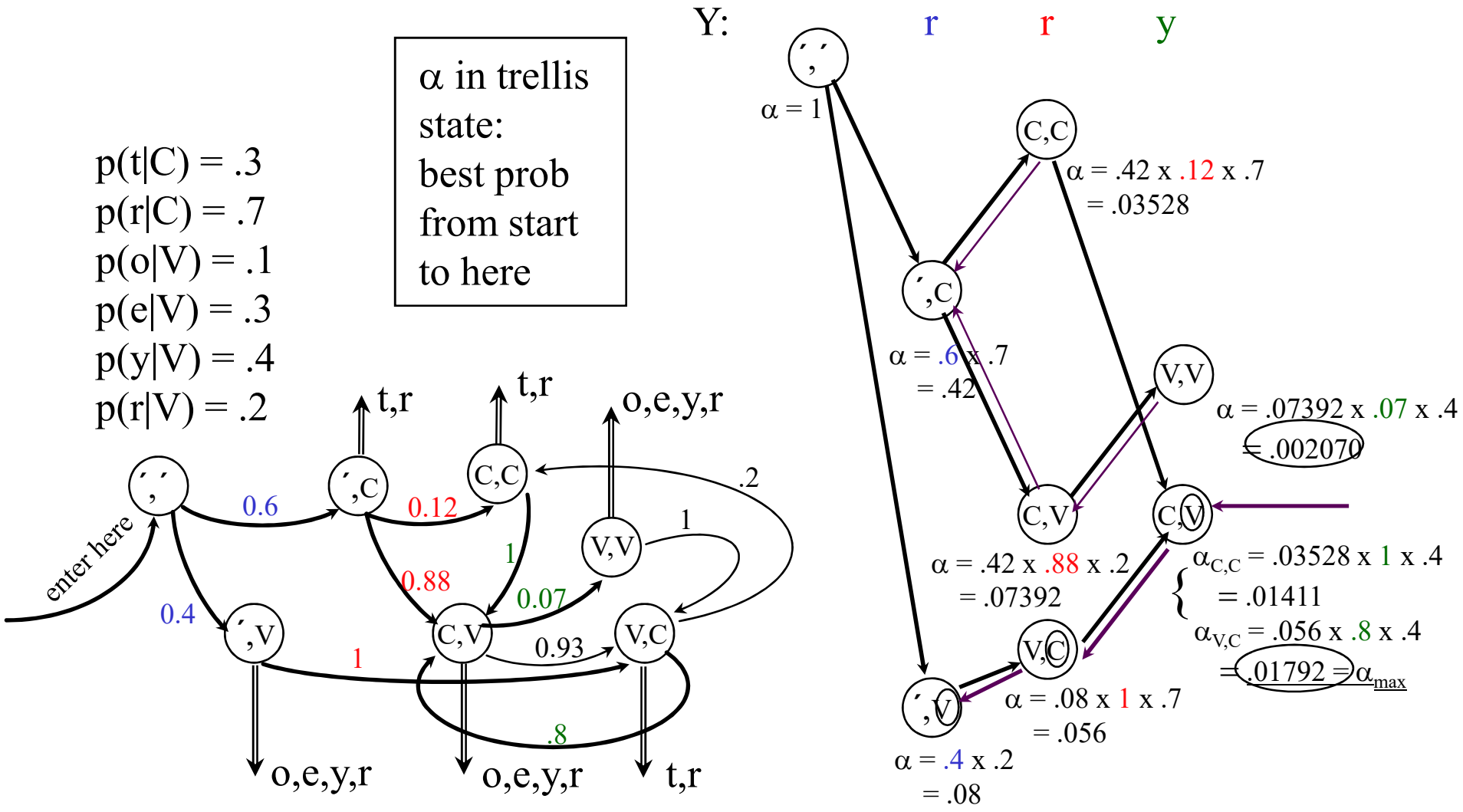


NB: remember previous state from which we got the maximum:



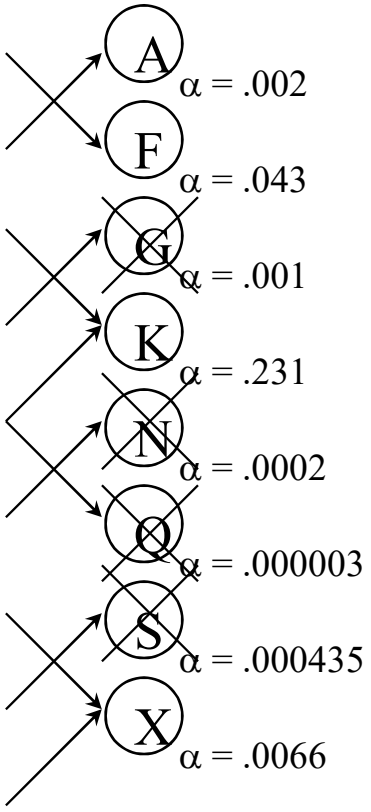
this is certainly the "backwards" maximum to (D,2)... but
it cannot change even whenever we go forward (M. Property: Limited History)

Viterbi Computation



Pruning

- Sometimes, too many trellis states in a stage:



criteria: (a) $\alpha <$ threshold
(b) $\Sigma\pi <$ threshold
(c) # of states > threshold
(get rid of smallest α)

HMM Parameter Estimation: the Baum-Welch Algorithm

Setting

- HMM (without P_S, P_Y) (S, S_0, Y), and data T = $\{y^i \in Y\}_{i=1..|T|}$
  - **will use T ~ |T|**
  - HMM structure is given: (S, S_0)
  - P_S:Typically, one wants to allow "fully connected" graph
    - **(i.e. no transitions forbidden ~ no transitions set to hard 0)**
    - **why? → we better leave it on the learning phase, based on the data!**
    - **sometimes possible to remove some transitions ahead of time**
  - P_Y: should be restricted (if not, we will not get anywhere!)
    - **restricted ~ hard 0 probabilities of p(y|s,s')**
    - **"Dictionary": states ↔ words, "m:n" mapping on S × Y (in general)**

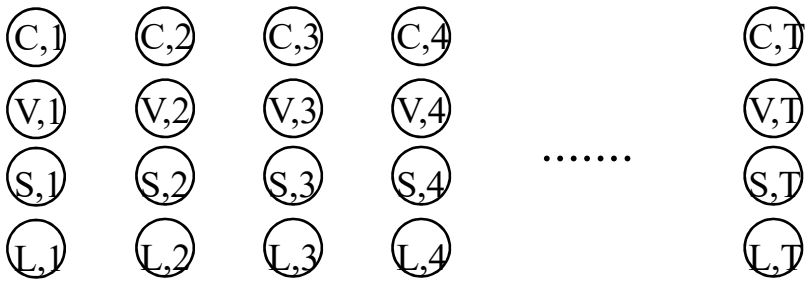
Initialization

- For computing the initial expected "counts"
- Important part
  - EM guaranteed to find a *local* maximum only (albeit a good one in most cases)
- P_Y initialization more important
  - fortunately, often easy to determine
    - **together with dictionary ↔ vocabulary mapping, get counts, then MLE**
- P_S initialization less important
  - e.g. uniform distribution for each p(.|s)

Data Structures

- Will need storage for:
  - The predetermined structure of the HMM
    (unless fully connected → need not to keep it!)
  - The parameters to be estimated (P_S, P_Y)
  - The expected counts (same size as P_S, P_Y)
  - The training data $T = \{y^i \in Y\}_{i=1..T}$
  - The trellis (if f.c.): ↑ T Size: T ´ S (Precisely, |T|´|S|)

Each trellis state:
two [float] numbers
(forward/backward)



(...and then some)

The Algorithm Part I

1. Initialize P_S, P_Y
2. Compute "forward" probabilities:
   - **follow the procedure for trellis (summing), compute $\alpha(s,i)$**
   - **use the current values of P_S, P_Y ($p(s'|s)$, $p(y|s,s')$):**

$$\alpha(s',i) = \sum_{s \to s'} \alpha(s,i-1) \times p(s'|s) \times p(y_i|s,s')$$

   - **NB: do not throw away the previous stage!**
3. Compute "backward" probabilities
   - **start at all nodes of the last stage, proceed backwards, $\beta(s,i)$**
   - **i.e., probability of the "tail" of data from stage *i* to the end of data**

$$\beta(s',i) = \sum_{s \leftarrow s'} \beta(s,i+1) \times p(s|s') \times p(y_{i+1}|s',s)$$

   - **also, keep the $\beta(s,i)$ at all trellis states**

The Algorithm Part II

4. Collect counts:
   - for each output/transition pair compute

$$c(y,s,s') = \Sigma_{i=0..k-1, y=y_{i+1}} \alpha(s,i)\, p(s'|s)\, p(y_{i+1}|s,s')\, \beta(s',i+1)$$

one pass through data, only stop at (output) y

prefix prob.

this transition prob ´ output prob

tail prob

$$c(s,s') = \Sigma_{y \in Y} c(y,s,s') \text{ (assuming all observed } y_i \text{ in Y)}$$

$$c(s) = \Sigma_{s' \in S} c(s,s')$$

5. Reestimate: $p'(s'|s) = c(s,s')/c(s)$ $p'(y|s,s') = c(y,s,s')/c(s,s')$
6. Repeat 2-5 until desired convergence limit is reached.

Baum-Welch: Tips & Tricks

- Normalization badly needed
  - long training data → extremely small probabilities
- Normalize α,β using the same norm. factor:

$$N(i) = \Sigma_{s \in S} \alpha(s,i)$$

as follows:

- **compute $\alpha(s,i)$ as usual (Step 2 of the algorithm), computing the sum N(i) at the given stage *i* as you go.**
- **at the end of each stage, recompute all αs (for each state s):**

$$\alpha^*(s,i) = \alpha(s,i) / N(i)$$

- **use the same N(i) for βs at the end of each backward (Step 3) stage:**

$$\beta^*(s,i) = \beta(s,i) / N(i)$$

(A Short Intro to) Morphology

Levels of (Formal) Description of Language

- 6 basic levels (more or less explicitly present in most theories):
  - and beyond (pragmatics/logic/...)
  - meaning (semantics)
  - (surface) syntax
  - morphology
  - phonology
  - phonetics/orthography
- Each level has an input and output representation
  - output from one level is the input to the next (upper) level
  - sometimes levels might be skipped (merged) or split

Morphology: Morphemes & Order

- Handles what is an ***isolated form*** in written text
- A morpheme is the smallest meaningful unit of language
- Grouping of phonemes (graphemes/letters) into morphemes
  - **deliverables** → **deliver**, **able** and **s** (3 ***morphemes***)
- Morpheme Combination
  - certain combinations/sequencing possible, other not:
    - **deliver+able+s, but not able+derive+s; noun+s, but not noun+ing**
    - **typically fixed (in any given language)**

Morphology: From Morphemes to Lemmas & Categories

- Lemma: lexical unit, “pointer” to lexicon
  - might as well be a number, but typically is represented as the “base form”, or “dictionary headword”
    - **possibly indexed when ambiguous/polysemous:**
      - state[1] (verb), state[2] (state-of-the-art), state[3] (government)
  - from one or more morphemes (“root”, “stem”, “root+derivation”, ...)
- Categories: non-lexical
  - small number of possible values (< 100, often < 5-10)

Morphology Level: The Mapping

- Formally: $A^+ \rightarrow 2^{(L,C_1,C_2,...,Cn)}$
  - A is the alphabet of phonemes (A^+ denotes any non-empty sequence of phonemes)
  - L is the set of possible lemmas, uniquely identified
  - C_i are morphological categories, such as:
    - **grammatical number, gender, case**
    - **person, tense, negation, degree of comparison, voice, aspect, ...**
    - **tone, politeness, ...**
    - **part of speech (not quite morphological category, but...)**
  - $2^{(L,C1,C2,...,Cn)}$ denotes the power set of $(L,C_1,C_2,...,C_n)$
  - A, L and C_i are obviously language-dependent

The Dictionary (or Lexicon)

- Repository of information about words:
  - Morphological:
    - **description of morphological "behavior": inflection patterns/classes**
  - Syntactic:
    - **Part of Speech**
    - **relations to other words:**
      - subcategorization (or "surface valency frames")
  - Semantic:
    - **semantic features**
    - **valency frames**
  - ...and any other! (e.g., translation)

The Categories: Part of Speech: Open and Closed Categories

- Part of Speech - POS (pretty much stable set across languages)
  - not so much morphological (can be looked up in a dictionary), but:
  - morphological "behavior" is typically consistent within a POS category
  - Open categories: ("open" to additions)
    - **verb, noun, pronoun, adjective, numeral, adverb**
      - subject to inflection (in general); subject to cross-category derivations
      - newly coined words always belong to open POS categories
      - potentially unlimited number of words
  - Closed categories:
    - **preposition, conjunction, article, interjection, clitic, particle**
      - not a base for derivation (possibly only by compounding)
      - finite and (very) small number of words

The Categories: Number and Gender

- Grammatical Number: Singular, Plural
  - nouns, pronouns, verbs, adjectives, numerals
    - **computer / computers; (he) goes / (they) go**
  - In some languages (Czech): Dual (nouns, pronouns, adjectives)
    - **(Pl.) nohami / (Dl.) nohama (Cz.; (by) legs (of sth)/(by) legs (of sb))**
- Grammatical Gender: Masculine, Feminine, Neuter
  - nouns, pronouns, verbs, adjectives, numerals
    - **he/she/it; читал, читала, читало (Ru.; (he/she/it) was-reading)**
    - **nouns: (mostly) do not change gender for a single lexical unit**
  - Also: animate/inanimate (gram., some genders), etc.
    - **Mädchen (Ge.; girl, neuter); děti (Cz.; children, masc. inanim.)**

The Categories: Case

- Case
  - English: only personal pronouns/possessives, 2 forms
  - other languages: 4 (German), 6 (Russian), 7 (Czech,Slovak,...)
    - **nouns, pronouns, adjectives, numerals**
  - most common cases (forms in singular/plural)
    - **nominative I/we (work) třída/třídy (Cz.; class)**
    - **genitive (picture of) me/us třídy/tříd**
    - **dative (give to) me/us třídě/třídám**
    - **accusative (see) me/us třídu/třídy**
    - **vocative -/- třído/třídy**
    - **locative (about) me/us třídě/třídách**
    - **instrumental (by) me/us třídou/třídami**

The Categories: Person, Tense

- Person
  - verbs, personal pronouns
    - **1st, 2nd, 3rd: (I) go, (you) go, (he) goes; (we) go, (you) go, (they) go**
    - **jdu, jdeš, jde, jdeme, jdete, jdou (Cz.)**
- Tense

		(Cz.: go)	(Pol.: go)
– past:	(you) went	-	szliście
– present:	(you pl.) go	jdete	idziecie
– future (!if not “analytical”)	-	půjdete	-
– concurrent (gerund)	going	jda	idąc
– preceding	-	-	szedłszy

The Categories: Voice & Aspect

- Voice
  - active vs. passive
    - **(I) drive / (I am being) driven**
    - **(Ich) setzte (mich) / (Ich bin) gesetzt (Ge.: to sit down)**
- Aspect
  - imperfective vs. perfective:
    - **покупал / купил (Ru.: I used to buy, I was buying) / I (have) bought)**
  - imperfective continuous vs. iterative (repeating)
    - **spal / spával (Cz.: I was sleeping / I used to sleep (every ...))**

The Categories: Negation, Degree of Comparison

- Negation:
  - even in English: impossible (~ not possible)
    - **Cz: every verb, adjective, adverb, some nouns; prefix *ne-***
- Degree of Comparison (non-analytical):
  - adjectives, adverbs:
    - **positive (big), comparative (bigger), superlative (biggest)**
    - **Pol.: (new) nowy, nowszy, najnowszy**
- Combination (by prefixing):
  - order? both possible: (neg.: Cz./Pol.: *ne-/nie-*, sup.: nej-/naj-)
    - **Cz.: nej*ne*možnější (the most impossible)**
    - **Pol.: *nie*najwierniejszy (the most unfaithful)**

Typology of Languages

- By morphological features
  - Analytical: using (function) words to express categories
    - **English, also French, Italian, ..., Japanese, Chinese**
      - I would have been going ~ (Pol.) szłabym
  - Inflective: using prefix/suffix/infix, combines several categ.
    - **Slavic: Czech, Russian, Polish,... (not Bulgarian); also French, German; Arabic**
      - (Cz. new(acc.)) nov***ou*** (Adj, Fem., Sg., Acc., Non-neg., Pos.)
  - Agglutinative: one category per (non-lexical) morpheme
    - **Finnish, Turkish, Hungarian**
      - (Fin. plural): -i-

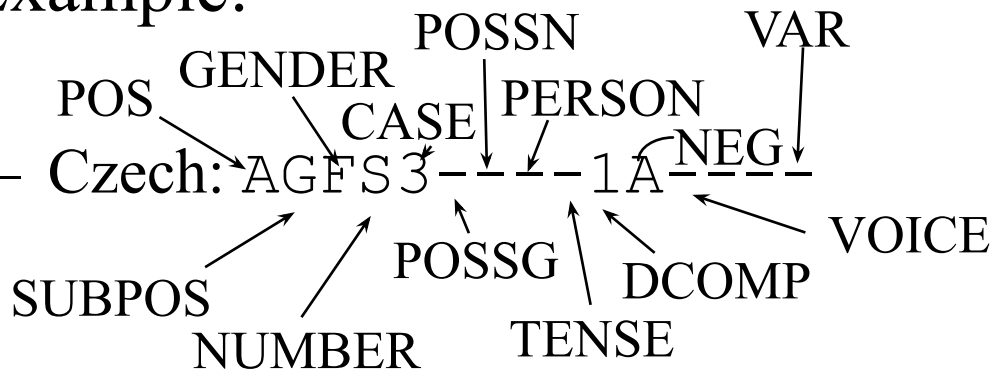
Tags, Tagsets, Tagging

Categories & Tags

- Tagset:
  - list of all possible combinations of category values for a given language
  - $T \subset C_1 \times C_2 \times \ldots \times C_n$
  - typically string of letters & digits:
    - **compact system: short idiosyncratic abbreviations:**
      - NNS (gen. noun, plural)
    - **positional system: each position i corresponds to C_i:**
      - AAMP3----2A---- (gen. Adj., Masc., Pl., 3rd case (dative), comparative (2nd degree of comparison), Affirmative (no neg.))
      - tense, person, variant, etc.: N/A (marked by empty position, or '-')
- Famous tagsets: Brown, Penn, Multext[-East], ...

Other Language Tagsets

- Differences:
  - size (10..10k)
  - categories covered (POS, Number, Case, Negation,...)
  - level of detail
  - presentation (short names vs. structured ("positional"))
- Example:
  - Czech: AGFS3---1A----



The task of (Morphological) Tagging

- Formally: $A^+ \rightarrow T$
  - A is the alphabet of phonemes (A^+ denotes any non-empty sequence of phonemes)
    - often: phonemes ~ letters
  - T is the set of tags (the "tagset")
- Tagging among the levels of language description:
  - phonetics ... phonology ... morphology ... syntax ... meaning ...
- Tasks: $A^+ \rightarrow 2^{(L,C_1,C_2,...,Cn)} \rightarrow T$

  - morphology
  - tagging: ***disambiguation*** (~ "select")

  - a step aside: tagging

Tagging Examples

- Word form: $A^{+} \rightarrow 2^{(L,C_1,C_2,...,Cn)} \rightarrow T$
  - He always books the violin concert tickets early.
    - MA: books → {(book-1,Noun,Pl,-,-),(book-2,Verb,Sg,Pres,3)}
    - tagging (disambiguation): ... → (Verb,Sg,Pres,3)
  - ...was pretty good. However, she did not realize...
    - MA: However → {(however-1,Conj/coord,-,-,-),(however-2,Adv,-,-,-)}
    - tagging: ... → (Conj/coord,-,-,-)
  - [æ n d] [g i v] [i t] [t u:] [j u:] (“and give it to you”)
    - MA: [t u:] → {(to-1,Prep),(two,Num),(to-2,Part/inf),(too,Adv)}
    - tagging: ... → (Prep)

Tagging Inside Morphology

- Do tagging first, then morphology:
- Formally: $A^+ \rightarrow T \rightarrow (L,C_1,C_2,...,C_n)$
- Rationale:
  - have $|T| < |(L,C_1,C_2,...,C_n)|$ (thus, less work for the tagger) and keep the mapping $A^+ \times T \rightarrow (L,C_1,C_2,...,C_n)$ unique.
- Possible for some languages only ("English-like")
- Same effect within "regular" $A^+ \rightarrow 2^{(L,C1,C2,...,Cn)} \rightarrow T$:
  - mapping $R : (C_1,C_2,...,C_n) \rightarrow T_{reduced}$, then (new) unique mapping $U: A^+ \times T_{reduced} \rightarrow (L,T)$

Lemmatization

- Full morphological analysis:

  MA: $A^+ \rightarrow 2^{(L,C1,C2,...,Cn)}$

  (recall: a lemma $l \in L$ is a lexical unit (~ dictionary entry ref)
- Lemmatization: reduced MA:
  - $L: A^+ \rightarrow 2^L: w \rightarrow \{l; (l,t_1,t_2,...,t_n) \in MA(w)\}$
  - again, need to disambiguate (want: $A^+ \rightarrow L$)

    (special case of word sense disambiguation, WSD)
  - "classic" tagging does not deal with lemmatization

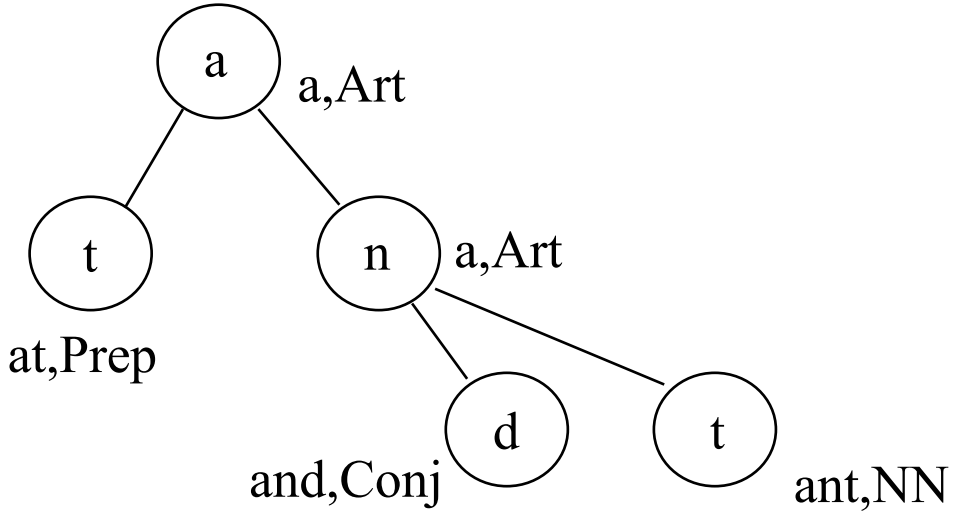
    (assumes lemmatization done afterwards somehow)

Morphological Analysis: Methods

- Word form list
  - books: book-2/VBZ, book-1/NNS
- Direct coding
  - endings: verbreg:s/VBZ, nounreg:s/NNS, adje:er/JJR, ...
  - (main) dictionary: book/verbreg, book/nounreg,nic/adje:nice
- Finite state machinery (FSM)
  - many “lexicons”, with continuation links: reg-root-lex → reg-end-lex
  - phonology included but (often) clearly separated
  - Two-level morphology (Kimmo Koskenniemi, 1983)

Word Lists

- Works e.g. for English
  - simpler morphology
- Implementation issues:
  - search trees
  - hash tables
  - (letter) trie:
- Minimization?



Word-internal[1] Segmentation (Direct)

- Strip prefixes: (un-, dis-, ...)
- Repeat for all plausible endings:
  - Split rest: root + ending (for every possible ending)
  - Find root in a dictionary, keep dictionary information
    - in particular, keep inflection class (such as reg, noun-irreg-e, ...)
  - Find ending, check inflection+prefix class match
  - If match found:
    - Output root-related info (typically, the lemma(s))
    - Output ending-related information (typically, the tag(s)).

[1] Word segmentation is a different problem (Japanese, speech in general)

Tagging: An Overview

Rule-based Disambiguation

- Example after-morphology data (using Penn tagset):

I	watch	a	fly	.
NN	NN	DT	NN	.
PRP	VB	NN	VB	
	VBP		VBP	

- Rules using
  - word forms, from context & current position
  - tags, from context and current position
  - tag sets, from context and current position
  - combinations thereof

Example Rules

I	watch	a	
fly	.		
NN	NN	DT	NN
.			
PRP	VB	NN	VB
	VBP		VBP

- If-then style:
  - $DT_{eq,-1,Tag} \Rightarrow NN$
    (implies $NN_{in,0,Set}$ as a condition)
  - $PRP_{eq,-1,Tag}$ *and* $DT_{eq,+1,Tag} \Rightarrow VBP$
  - $\{DT,NN\}_{sub,0,Set} \Rightarrow DT$
  - $\{VB,VBZ,VBP,VBD,VBG\}_{inc,+1,Tag} \Rightarrow$ *not* DT
- Regular expressions:
  - *not*(<*,*,DT> <*,*,*not*NN>))
  - *not*(<*,*,PRP>,<*,*,*not*VBP>,<*,*,DT>)
  - *not*(<*,$\{DT,NN\}_{sub,}$ *not*DT>)
  - *not*(<*,*,DT>,<*,*,{VB,VBZ,VBP,VBD,VBG}>)

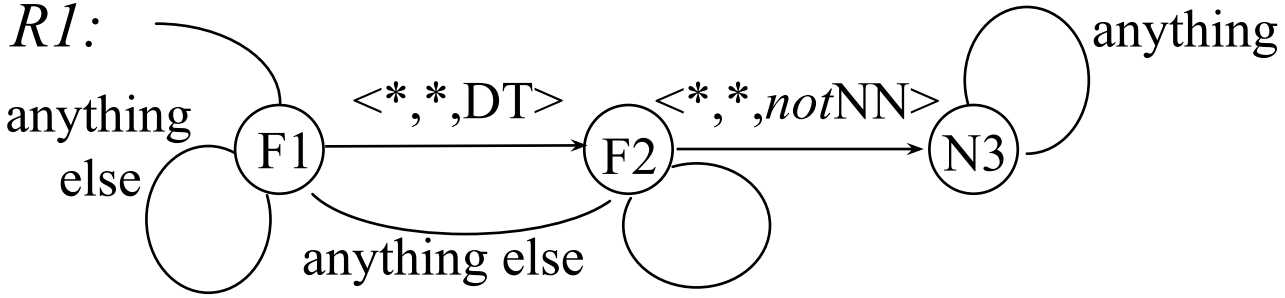
Implementation

- Finite State Automata
  - parallel (each rule ~ automaton);
    - algorithm: keep all paths which cause all automata say *yes*
  - compile into single FSA (intersection)
- Algorithm:
  - a version of Viterbi search, but:
    - no probabilities (“categorical” rules)
    - multiple input:
      - keep track of all possible paths

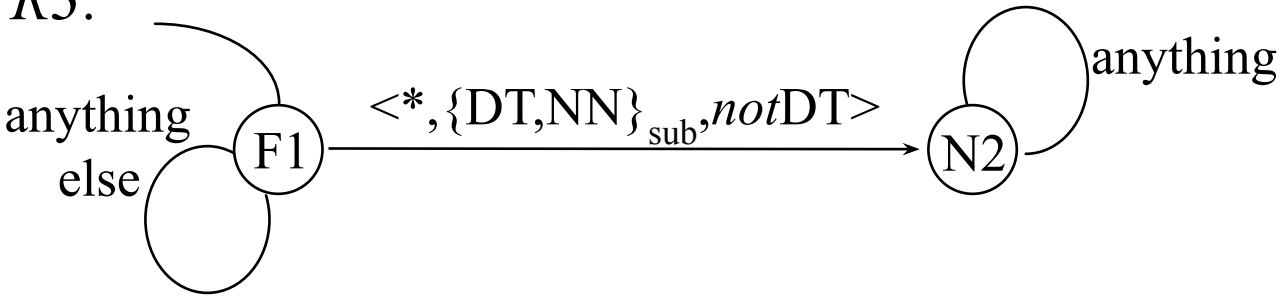
Example: the FSA

- *R1: not*(<*,*,DT> <*,*,*not*NN>))
- *R2: not*(<*,*,PRP>,<*,*,*not*VBP>,<*,*,DT>)
- *R3: not*(<*,{DT,NN}$_{sub,}$ DT>)
- *R4: not*(<*,*,DT>,<*,*,{VB,VBZ,VBP,VBD,VBG}>)

- *R1:*



- *R3:*



Applying the FSA

I	watch	a	fly
NN	NN	DT	NN
PRP	VB	NN	VB
	VBP		VBP

- *R1: not*(<*,*,DT> <*,*,*not*NN>))
- *R2: not*(<*,*,PRP>,<*,*,*not*VBP>,<*,*,DT>)
- *R3: not*(<*,{DT,NN}$_{sub,}$ DT>)
- *R4: not*(<*,*,DT>,<*,*,{VB,VBZ,VBP,VBD,VBG}>)

- R1 blocks:

a	fly
DT	
	VB
	VBP

remains:

a	fly
DT	NN

or

a	fly
	NN
NN	VB
	VBP

- R2 blocks:

I	watch	a
	NN	DT
PRP	VB	

remains e.g.:

I	watch	a
		DT
PRP		
	VBP	

and more

- R3 blocks:

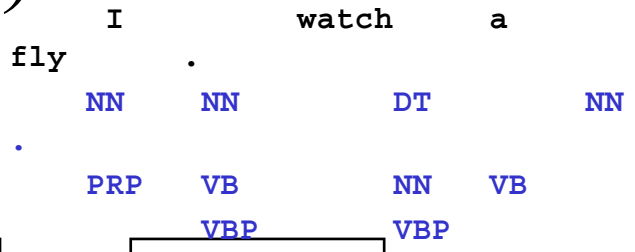
a
NN

remains only:

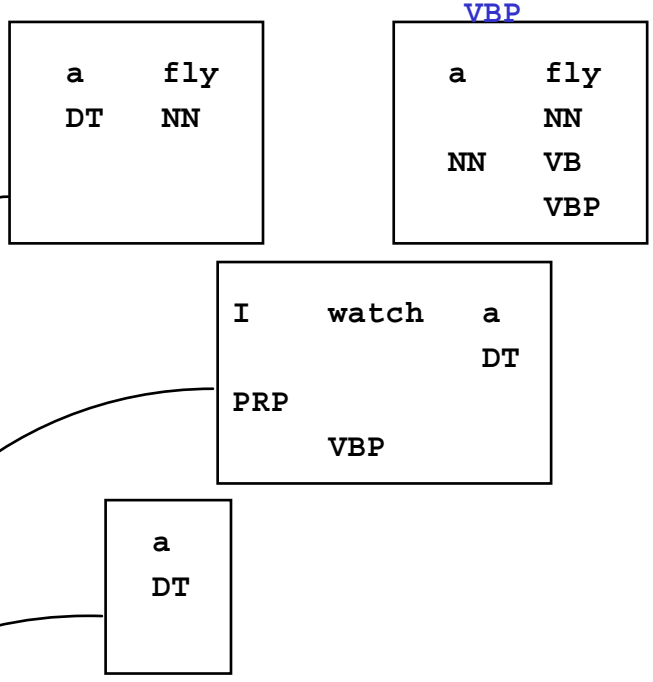
a
DT

- R4 $\subset$ R1!

Applying the FSA (Cont.)



- Combine:

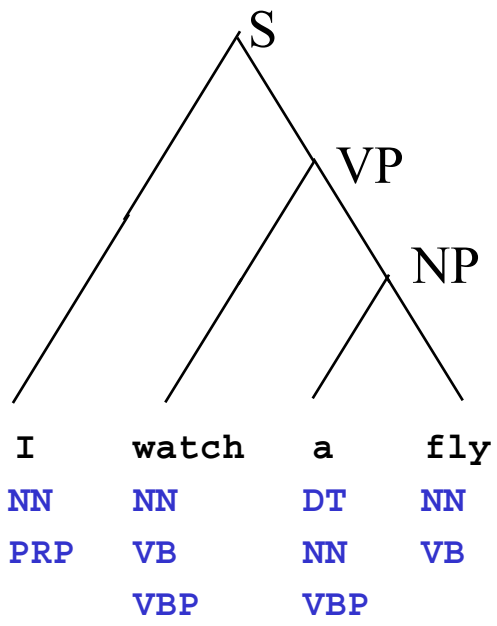


- Result:

I watch a fly .
PRP VBP DT NN .

Tagging by Parsing

- Build a parse tree from the multiple input:



- Track down rules: e.g., NP → DT NN: extract (a/DT fly/NN)
- More difficult than tagging itself; results mixed

Statistical Methods (Overview)

- Probabilistic:
  - **Hidden Markov Models (already discussed)**
    - Merialdo and many more
  - Maximum Entropy
    - DellaPietra et al., Ratnaparkhi, and others
- Rule-based:
  - **TBEDL (Transformation Based, Error Driven Learning)**
    - Brill's tagger
  - Example-based
    - Daelemans, Zavrel, others

Transformation-Based Tagging

The Task, Again

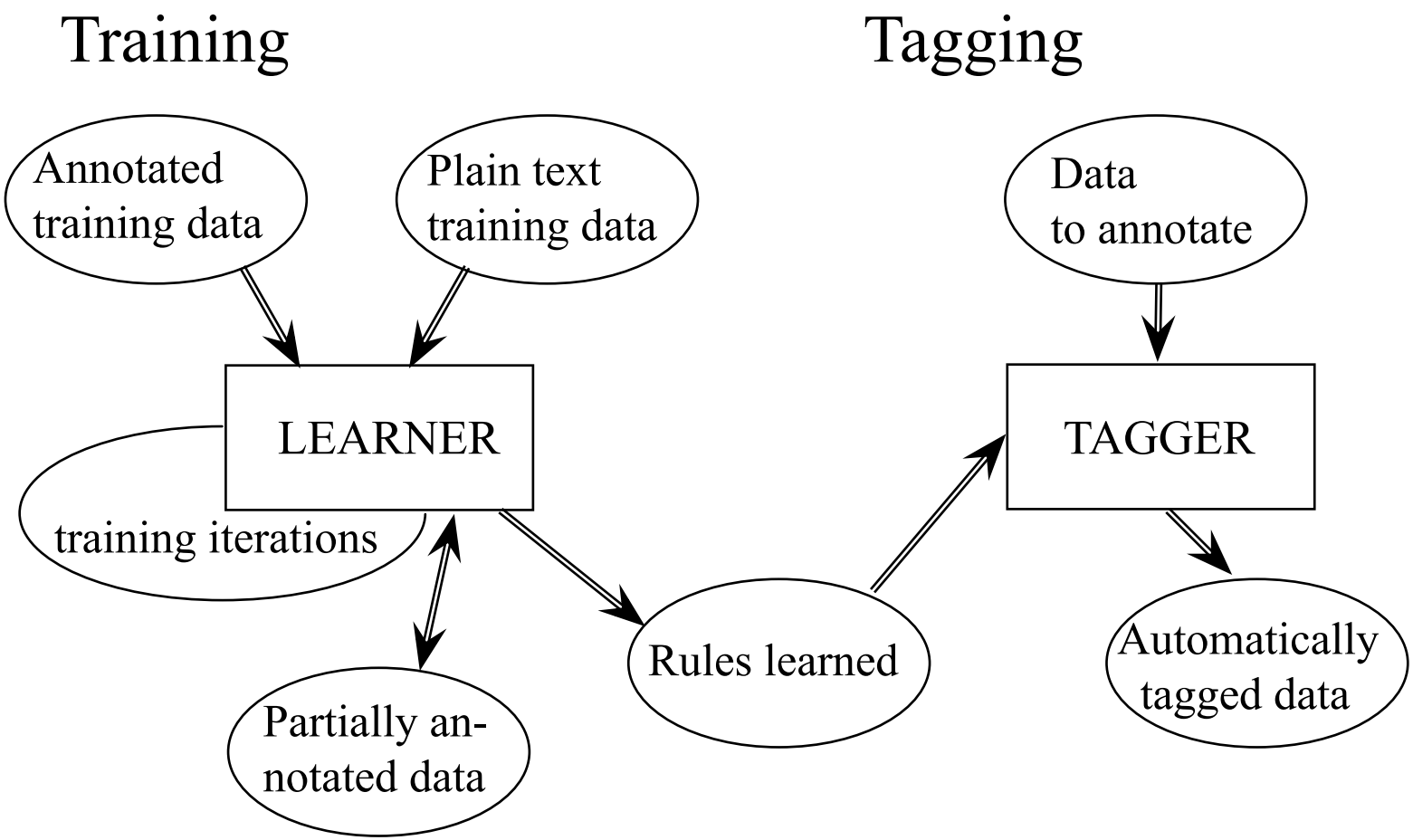
- Recall:
  - tagging ~ morphological disambiguation
  - tagset $V_T \subset (C_1,C_2,...C_n)$
    - C_i - morphological categories, such as POS, NUMBER, CASE, PERSON, TENSE, GENDER, ...
  - mapping $w \rightarrow \{t \in V_T\}$ exists
    - restriction of Morphological Analysis: $A^+ \rightarrow 2^{(L,C1,C2,...,Cn)}$

      where A is the language alphabet, L is the set of lemmas
  - extension to punctuation, sentence boundaries (treated as words)

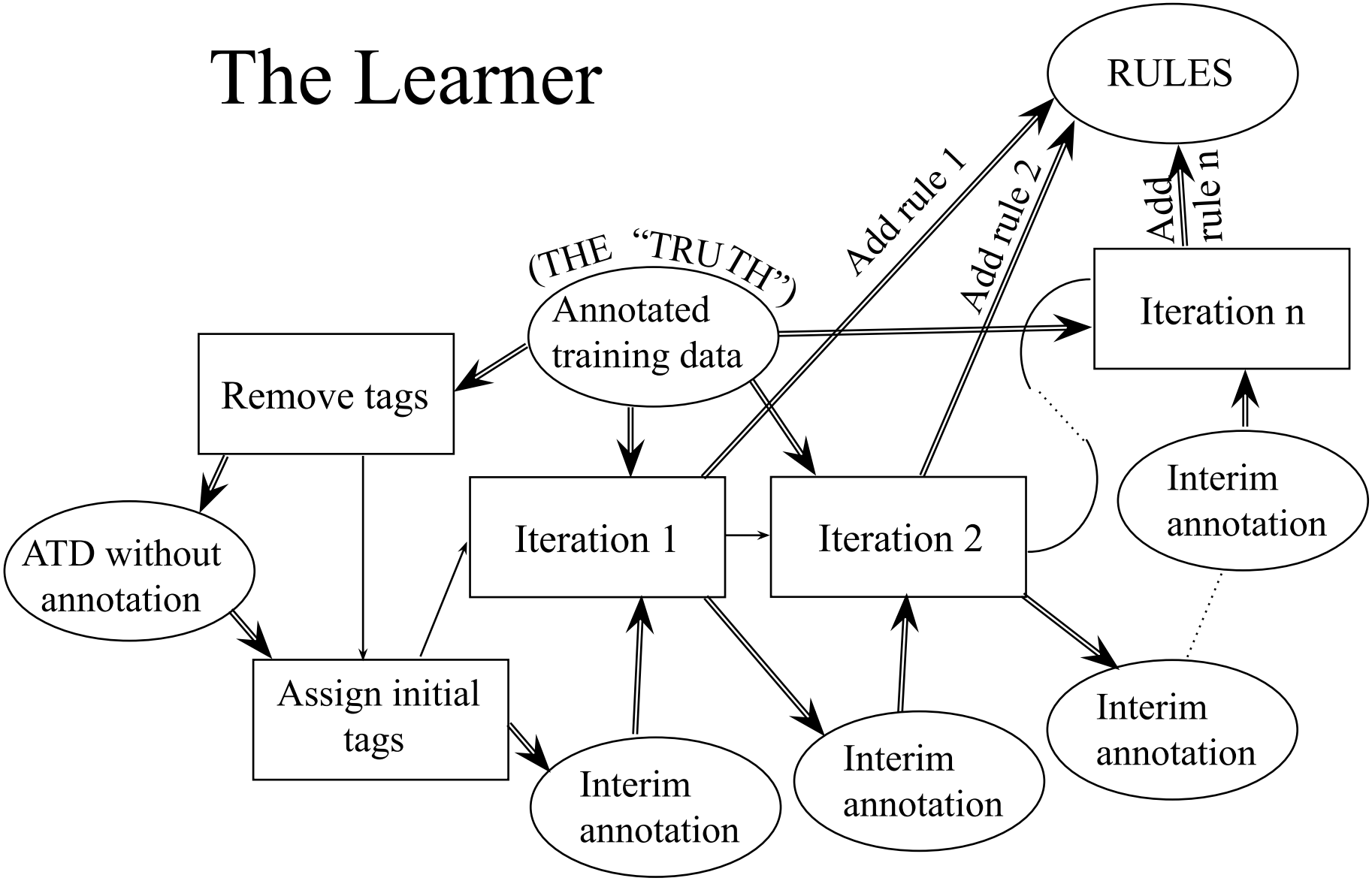
Setting

- ***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
Interim annotation
ATD without annotation
Iteration 1
Iteration 2
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 possibility:
  - NN
- Another:
  - the most frequent tag for a given word form
- Even:
  - use an HMM tagger for the initial assignment
- Not particularly sensitive

The Criterion

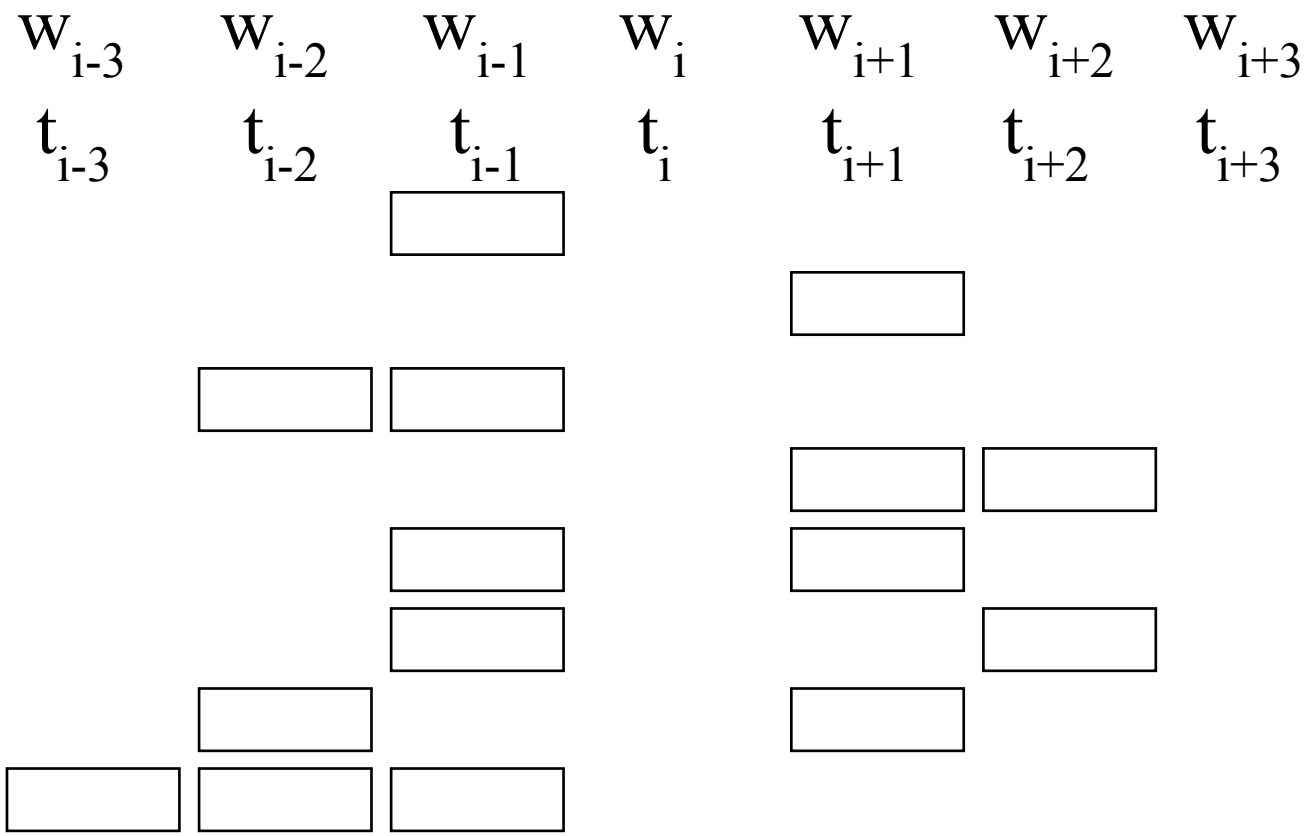
- Error rate (or Accuracy):
  - beginning of an iteration: some error rate E_{in}
  - each possible rule r, when applied at every data position:
    - makes an improvement somewhere in the data ($c_{improved}(r)$)
    - makes it worse at some places ($c_{worsened}(r)$)
    - and, of course, does not 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) ≤ Threshold
- NB: prone to overtraining!
  - therefore, setting a reasonable threshold advisable
- Heldout?
  - maybe: remove rules which degrade performance on H

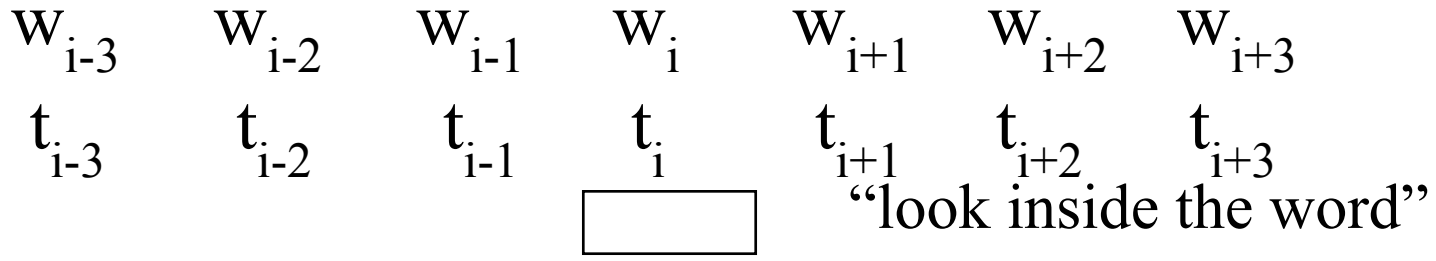
The Pool of Rules (Templates)

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



Lexical Rules

- Other type: lexical rules



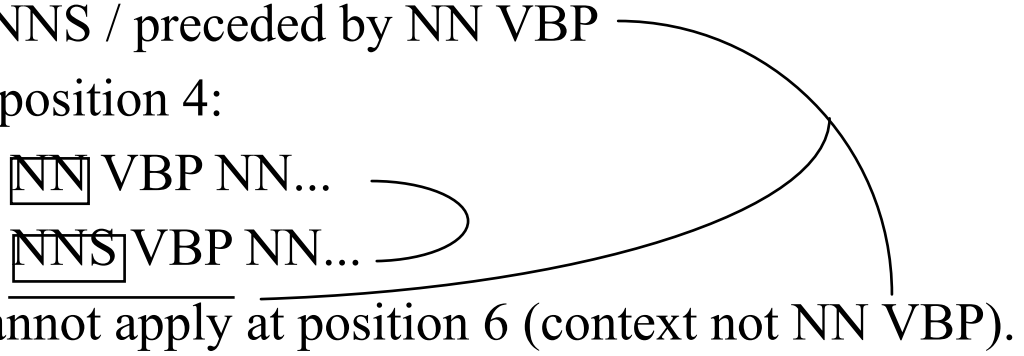
- Example:
  - w_i has suffix -ied
  - w_i has prefix ge-

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.