Sequence Labeling: POS Tagging and NER

PREPARED BY: AHMAD ALAA ALDINE

POS Tagging

Parts Of Speech – History

The idea that words can be classified into grammatical categories return back to 100 B.C. (**long time ago**)

Eight parts of speech attributed to Dionysius Thrax of Alexandria:

- noun, verb, pronoun, preposition, adverb, conjunction, participle, article
- These categories are still relevant for NLP today.

Two classes of words: Open vs. Closed

Closed class words

- Relatively fixed membership
- Usually **function** words: short, frequent words with grammatical function
  - determiners: ***a, an, the***
  - pronouns: ***she, he, I***
  - prepositions: ***on, under, over, near, by, …***

Open class words

- Usually **content** words: Nouns, Verbs, Adjectives, Adverbs
  - Plus interjections: **oh, ouch, uh-huh, yes, hello**
- New nouns and verbs like *iPhone* or *to fax*

Open class ("content") words

- Nouns
  - Proper: *Janet*, *Italy*
  - Common: *cat, cats*, *mango*
- Verbs
  - Main: *eat*, *went*
  - Auxiliary: *can*, *had*
- Adjectives *old green tasty*
- Adverbs *slowly yesterday*
- Numbers *122,312* *one*
- Interjections *Ow hello*
- … more

Closed class ("function")

- Determiners *the some*
- Conjunctions *and or*
- Pronouns *they its*
- Prepositions *to with*
- Particles *off up*
- … more

What is POS Tagging?

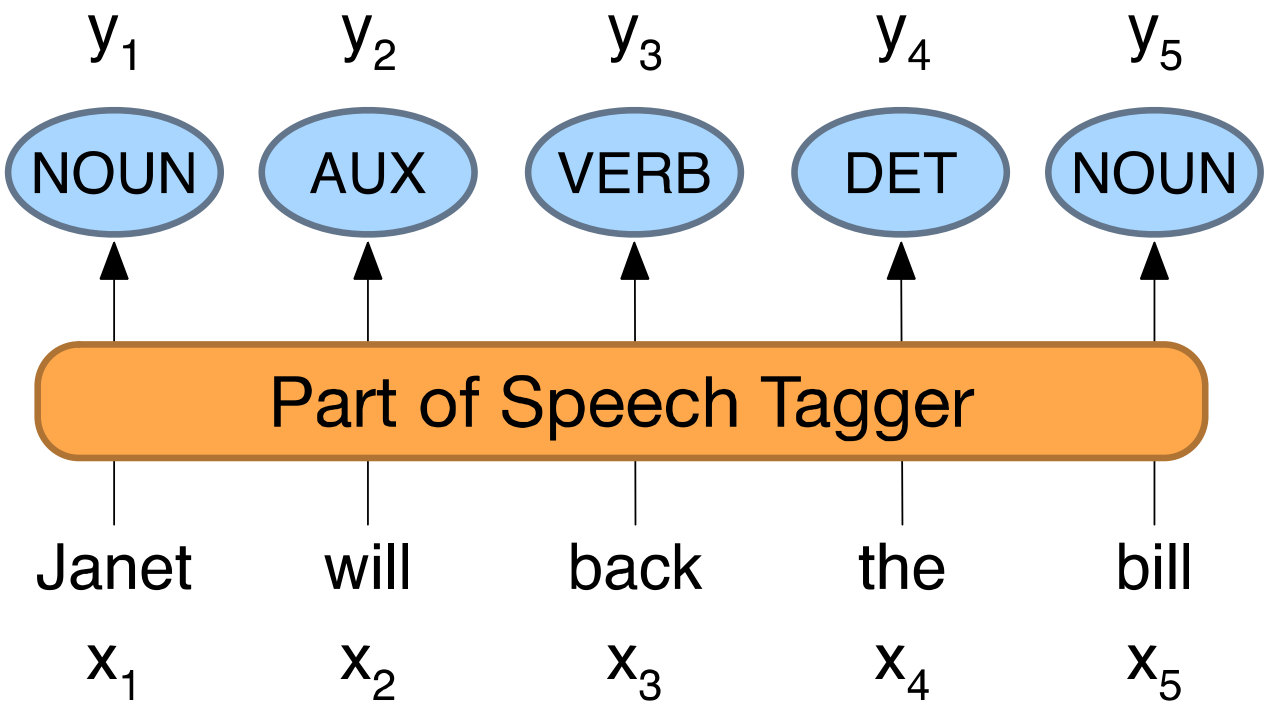
Part-of-Speech (POS) tagging is the process of **assigning grammatical categories** (like noun, verb, adjective) to each word in a sentence based on both its **definition and context**.

Example

The	quick	brown	fox	jumps
DET	ADJ	ADJ	NOUN	VERB

POS tagging – Sequence Labeling

Map from sequence $x_1,\ldots,x_n$ of words to $y_1,\ldots,y_n$ of POS tags



Common POS Tag Sets

Penn Treebank

45 tags

Most widely used (NN, VBZ, JJ, DT, etc.)

Universal Dependencies

17 tags

Language Independent (NOUN, VERB, ADJ, DET)

Example Penn Treebank Tags

NN (noun, singular)
VBZ (verb, 3rd person)
JJ (adjective)
DT (determiner)
RB (adverb)
PRP (pronoun)

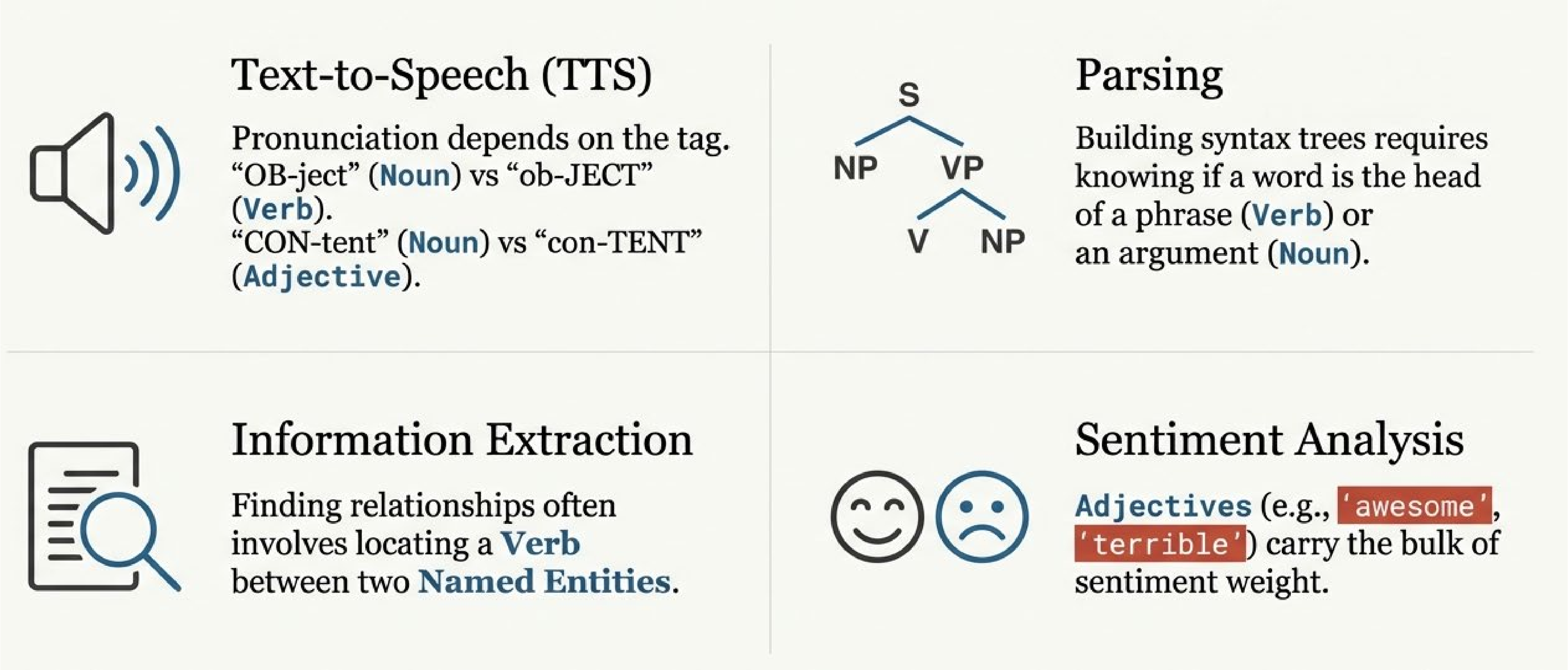
Universal Dependencies Tagset

	Tag	Description	Example
Open Class	**ADJ**	Adjective: noun modifiers describing properties	*red*, *young*, *awesome*
	ADV	Adverb: verb modifiers of time, place, manner	*very*, *slowly*, *home*, *yesterday*
	NOUN	words for persons, places, things, etc.	*algorithm*, *cat*, *mango*, *beauty*
	VERB	words for actions and processes	*draw*, *provide*, *go*
	PROPN	Proper noun: name of a person, organization, place, etc..	*Regina*, *IBM*, *Colorado*
	INTJ	Interjection: exclamation, greeting, yes/no response, etc.	*oh*, *um*, *yes*, *hello*
Closed Class Words	**ADP**	Adposition (Preposition/Postposition): marks a noun's spacial, temporal, or other relation	*in, on, by under*
	AUX	Auxiliary: helping verb marking tense, aspect, mood, etc.,	*can, may, should, are*
	CCONJ	Coordinating Conjunction: joins two phrases/clauses	*and*, *or*, *but*
	DET	Determiner: marks noun phrase properties	*a, an, the, this*
	NUM	Numeral	*one, two, first, second*
	PART	Particle: a preposition-like form used together with a verb	*up, down, on, off, in, out, at, by*
	PRON	Pronoun: a shorthand for referring to an entity or event	*she, who, I, others*
	SCONJ	Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement	*that*, *which*
Other	**PUNCT**	Punctuation	; , ()
	SYM	Symbols like $ or emoji	$, %
	X	Other	asdf, qwfg

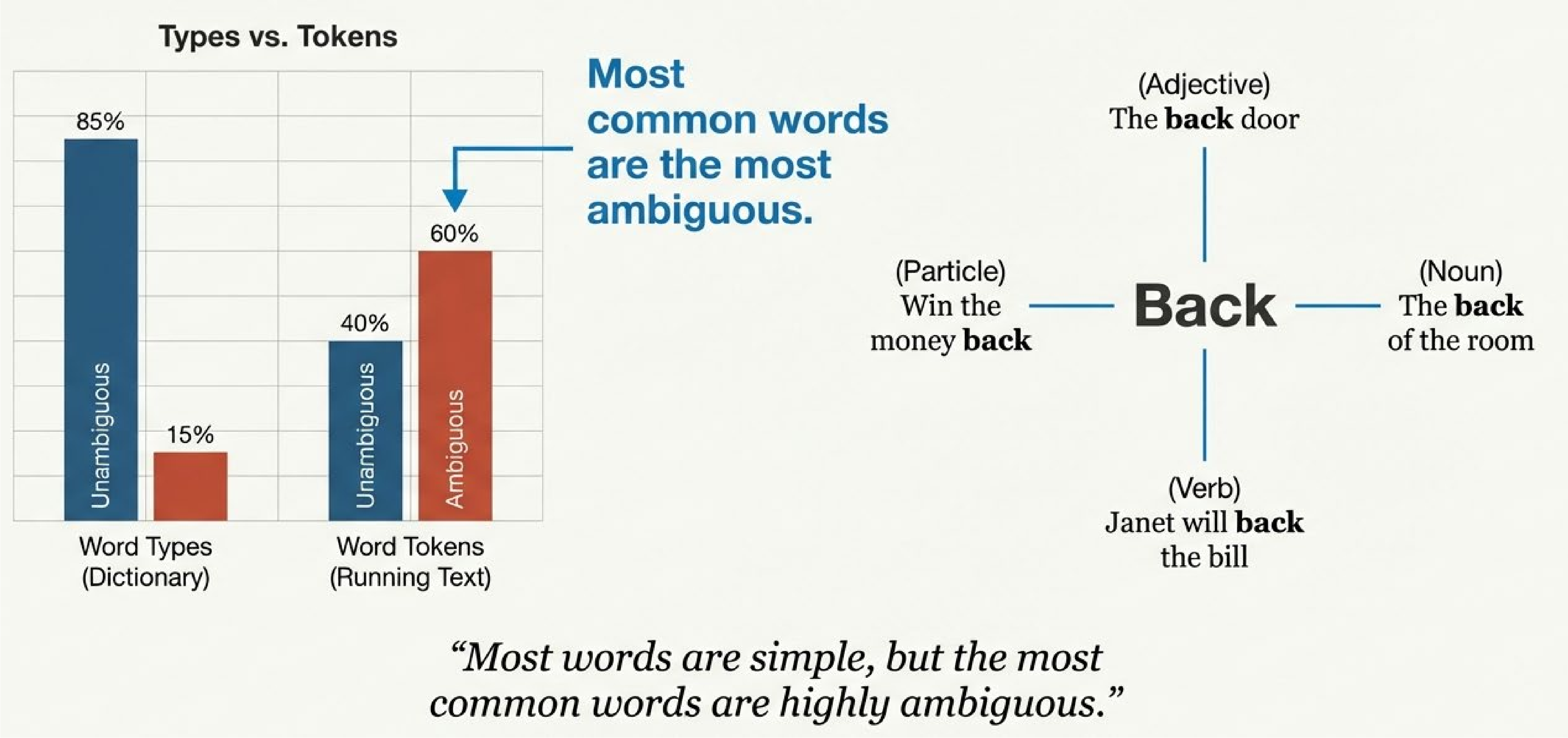
Penn Treebank core 36 part-of-speech tags

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	infinitive to	*to*
CD	cardinal number	*one, two*	NNPS	proper noun, plu.	*Carolinas*	UH	interjection	*ah, oops*
DT	determiner	*a, the*	NNS	noun, plural	*llamas*	VB	verb base	*eat*
EX	existential 'there'	*there*	PDT	predeterminer	*all, both*	VBD	verb past tense	*ate*
FW	foreign word	*mea culpa*	POS	possessive ending	*'s*	VBG	verb gerund	*eating*
IN	preposition/ subordin-conj	*of, in, by*	PRP	personal pronoun	*I, you, he*	VBN	verb past partici- ple	*eaten*
JJ	adjective	*yellow*	PRP$	possess. pronoun	*your*	VBP	verb non-3sg-pr	*eat*
JJR	comparative adj	*bigger*	RB	adverb	*quickly*	VBZ	verb 3sg pres	*eats*
JJS	superlative adj	*wildest*	RBR	comparative adv	*faster*	WDT	wh-determ.	*which, that*
LS	list item marker	*1, 2, One*	RBS	superlatv. adv	*fastest*	WP	wh-pronoun	*what, who*
MD	modal	*can, should*	RP	particle	*up, off*	WP$	wh-possess.	*whose*
NN	sing or mass noun	*llama*	SYM	symbol	*+, %, &*	WRB	wh-adverb	*how, where*

Why Part of Speech Tagging Matters



How difficult is POS tagging in English?



Sources of information for POS tagging

`Janet will back the bill`

AUX/NOUN/VERB? NOUN/VERB?

Prior probabilities of word/tag

- "will" is usually an AUX

Identity of neighboring words

- "the" means the next word is probably not a verb

Morphology and wordshape:

- Prefixes unable: un- → ADJ
- Suffixes importantly: -ly → ADJ
- Capitalization Janet: CAP → PROPN

Named Entity Recognition (NER)

Named Entities

> Part of speech tagging can tell us that words like Janet, Stanford University, and Colorado are all proper nouns; being a proper noun is a grammatical property of these words.

From a semantic perspective, these proper nouns refer to different kinds of **Named Entities**:

- Janet / Person
- Stanford University / Organization
- Colorado / Location

Often multi-word phrases

Type	Tag	Sample Categories	Example sentences
People	PER	people, characters	**Turing** is a giant of computer science.
Organization	ORG	companies, sports teams	The **IPCC** warned about the cyclone.
Location	LOC	regions, mountains, seas	**Mt. Sanitas** is in **Sunshine Canyon**.
Geo-Political Entity	GPE	countries, states	**Palo Alto** is raising the fees for parking.

Extended Named Entities

Named Entities are also extended to things that aren't entities:

- dates, times, money, events, products

Example

Citing high fuel prices, [ORG United Airlines] said [TIME Friday] it has increased fares by [MONEY $6] per round trip on flights to some cities also served by lower-cost carriers. [ORG American Airlines], a unit of [ORG AMR Corp.], immediately matched the move, spokesman [PER Tim Wagner] said. [ORG United], a unit of [ORG UAL Corp.], said the increase took effect [TIME Thursday] and applies to most routes where it competes against discount carriers, such as [LOC Chicago] to [LOC Dallas] and [LOC Denver] to [LOC San Francisco].

What is Named Entity Recognition?

NER is the task of identifying and classifying named entities in text into predefined categories such as persons, organizations, locations, dates, and more.

At its core, NLP is just a two-step process:

- Span Detection (Identify Entity Spans): The system scans text to locate potential entities, such as proper nouns, dates, or numerical values, and determines where they start and end (e.g., recognizing that "New" and "York" form the entity "New York").

- Entity Classification (Tagging): The system assigns a category to the identified span based on pre-defined types, such as Person (PER), Organization (ORG), or Location (LOC).

Why NER matters?

NER is a prerequisite for many sophisticated NLP applications

The Bridge to Structured Information

"Microsoft acquired GitHub for $7.5 billion. The deal was announced by CEO Satya Nadella in San Francisco."

{**Organizations:** Microsoft, GitHub ; **Person:** Satya Nadella **;** **Location:** San Francisco **;** **Money:** $7.5 billion}

Question Answering

Input: *Who founded Microsoft?*
NER Helps: Extract PERSON entity associated with ORG 'Microsoft'

Relation Extraction

Input: *"Elon Musk is the CEO of Tesla"*
NER Helps: Extract: (Elon Musk, CEO_OF, Tesla)

Document Summarization

Input: *Long article about Apple's earnings*
NER Helps: Focus on key entities: Apple, revenue, products

Why NER is hard

1. Segmentation
   - In POS tagging, no segmentation problem since each word gets one tag.
   - In NER, we have to find and segment the entities!

2. Type ambiguity

[$_{\text{PER}}$ Washington] was born into slavery on the farm of James Burroughs.
[$_{\text{ORG}}$ Washington] went up 2 games to 1 in the four-game series.
Blair arrived in [$_{\text{LOC}}$ Washington] for what may well be his last state visit.
In June, [$_{\text{GPE}}$ Washington] passed a primary seatbelt law.

BIO Tagging

How can we turn this structured problem into a sequence problem like POS tagging, with one label per word?

[PER Jane Villanueva] of [ORG United Airlines Holding] discussed the [LOC Chicago] route.

B: token that *begins* a span

I: tokens *inside* a span

O: tokens outside of any span

# of tags (where n is #entity types):

- 1 O tag,
- *n* B tags,
- *n* I tags
- total of *2n+1*

Words	BIO Label
Jane	B-PER
Villanueva	I-PER
of	O
United	B-ORG
Airlines	I-ORG
Holding	I-ORG
discussed	O
the	O
Chicago	B-LOC
route	O
.	O

BIO Tagging variants: IO and BIOES

IO tagging, which loses some information by eliminating the B tag.

BIOES tagging, which adds an end tag E for the end of a span, and a span tag S for a span consisting of only one word.

Words	**IO Label**	**BIO Label**	**BIOES Label**
Jane	I-PER	B-PER	B-PER
Villanueva	I-PER	I-PER	E-PER
of	O	O	O
United	I-ORG	B-ORG	B-ORG
Airlines	I-ORG	I-ORG	I-ORG
Holding	I-ORG	I-ORG	E-ORG
discussed	O	O	O
the	O	O	O
Chicago	I-LOC	B-LOC	S-LOC
route	O	O	O
.	O	O	O

Standard algorithms for POS tagging and NER

Supervised Machine Learning given a human-labeled training set of text annotated with tags

- Hidden Markov Models
- Conditional Random Fields (CRF)
- Maximum Entropy Markov Models (MEMM)
- Neural sequence models (RNNs or Transformers)

All required a hand-labeled training set

Approach	Training Data	Accuracy	Complexity
Rule-Based	None	~85-90%	Low
HMM	Annotated corpus	~96-97%	Medium
CRF	Annotated corpus	~97%	Medium
RNNs	Large annotated corpus	~97-98%	High
Transformer	Massive pre-training	~98-99%	Very High

Hidden Markov Model for POS Tagging

Markov Chains: The Foundation

Markov Assumption

To predict the future, only the present matters. The past is irrelevant except through the current state.

Formula:

$$P(q_i = a \mid q_1 \ldots q_{i-1}) = P(q_i = a \mid q_{i-1})$$

Think of it like predicting tomorrow's weather based ONLY on today's weather!

Markov Chain Components

Q - States

$Q = \{q1, q2, ..., qN\}$

The set of N possible states (e.g., HOT, COLD, WARM)

A - Transition Matrix

$aij = P(qj \mid qi)$

Probability of moving from state i to state j. Each row sums to 1.0

π - Initial Distribution

$\pi = [\pi1, \pi2, ..., \piN]$

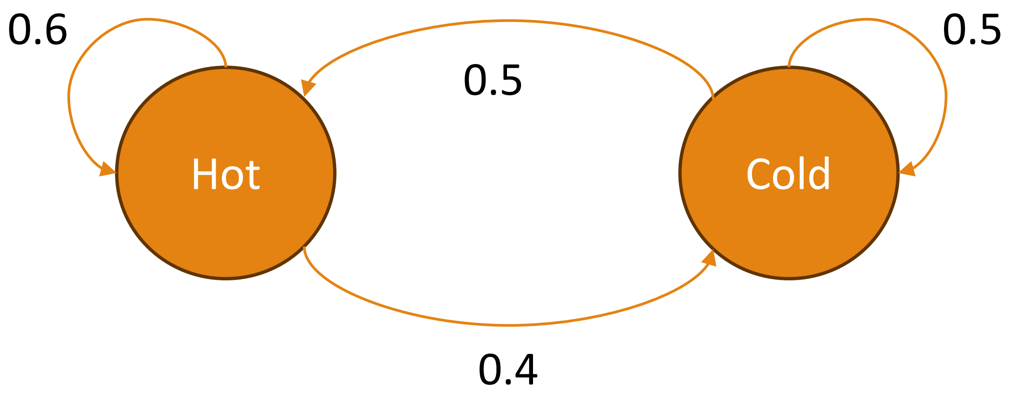
Probability of starting in each state. Sum equals 1.0

Weather Example

States: HOT, COLD

Transitions:

- HOT → HOT: 60%
- HOT → COLD: 40%
- COLD → COLD: 50%
- COLD → HOT: 50%



Transition Matrix

	HOT	COLD
HOT	**0.6**	**0.4**
COLD	**0.5**	**0.5**

If today is HOT, tomorrow has 60% chance of being HOT and 40% chance of being COLD.

Hidden Markov Model (HMM)

States are hidden, but observations are visible.

We only see OBSERVATIONS generated by hidden states.

Goal: infer the hidden states!

For example, we don’t normally observe part-of-speech tags in a text.

Rather, we see words, and must infer the tags from the word sequence.

HMM Components

B - Emission Probabilities

$b_i(o_t) = P(o_t \mid q_i)$

Probability of observing ot given state qi

P(3 ice creams | HOT) = 0.4

O - Observation Sequence

$O = \{o_1, o_2, \ldots, o_T\}$

The sequence we actually see

O = {3, 1, 3} (ice cream counts)

Plus all Markov Chain components: Q (states), A (transitions), π (initial distribution)

Weather HMM Example

Transition Probabilities (Hidden States)

	HOT	COLD
HOT	0.7	0.3
COLD	0.4	0.6

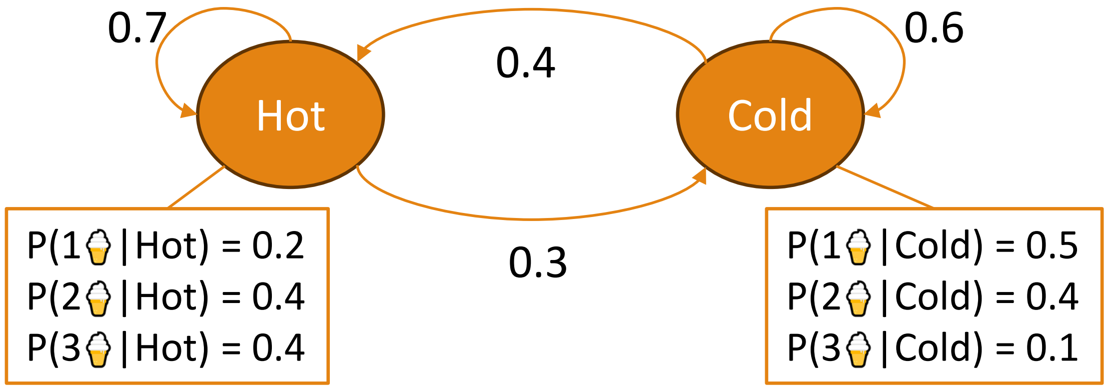
Emission Probabilities (Observations)

	1 🍦	2 🍦	3 🍦
HOT	0.2	0.4	0.4
COLD	0.5	0.4	0.1

Initial Probabilities: P(HOT) = 0.8 P(COLD) = 0.2

Interpretation

- If it's HOT today, 70% chance it stays HOT tomorrow
- If it's HOT, there's 40% chance we observe 3 ice creams sold
- We start with 80% probability that day 1 is HOT



HMM: Two Simplifying Assumptions

1 **Markov Assumption**

$P(q_i \mid q_1 \ldots q_{i-1}) = P(q_i \mid q_{i-1})$

Current state depends only on previous state, not entire history

2 **Output Independence**

$P(o_i \mid q_1 \ldots q_T, o_1 \ldots o_T) = P(o_i \mid q_i)$

Observation depends only on current state, not others

The components of an HMM tagger

An HMM tagger has two components, the A (Transition) and B (Emission) probabilities.

Transition Matrix

The A matrix contains the tag transition probabilities:

$$P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$$

C(ti-1, ti): how often (count) ti-1 is followed by ti in the labeled corpus.
C(ti-1): how often (count) ti-1 found in the labeled corpus.

Emission Matrix

The B matrix represent the probability, given a tag, that it will be associated with a given word

$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

C(ti, wi): how often (count) ti is associated with wi in the labeled corpus.
C(ti): how often (count) ti found in the labeled corpus.

HMM tagging as decoding

Decoding

Given as input an HMM λ = (A, B) and a sequence of observations O = o1,o2, :::,oT , find the most probable sequence of states Q = q1q2q3 : : :qT

HMM Decoding for POS Tagging

$$\hat{t}_{1:n} = \underset{t_1 \ldots t_n}{\operatorname{argmax}} P(t_1 \ldots t_n | w_1 \ldots w_n)$$

choose the tag sequence t1 : : : tn that is most probable given the observation sequence of n words w1 : : :wn

Simplifying Decoding

$$\hat{t}_{1:n} = \underset{t_1 \dots t_n}{\operatorname{argmax}} P(t_1 \dots t_n | w_1 \dots w_n)$$

Bayes Rule

$$\hat{t}_{1:n} = \underset{t_1 \dots t_n}{\operatorname{argmax}} \frac{P(w_1 \dots w_n | t_1 \dots t_n) P(t_1 \dots t_n)}{P(w_1 \dots w_n)}$$

Dropping the denominator P(w1…wn)

$$\hat{t}_{1:n} = \underset{t_1 \dots t_n}{\operatorname{argmax}} P(w_1 \dots w_n | t_1 \dots t_n) P(t_1 \dots t_n)$$

Output Independence

Markov Assumption

$$\hat{t}_{1:n} = \underset{t_1 \dots t_n}{\operatorname{argmax}} P(t_1 \dots t_n | w_1 \dots w_n) \approx \underset{t_1 \dots t_n}{\operatorname{argmax}} \prod_{i=1}^{n} \overbrace{P(w_i | t_i)}^{\text{emission}} \overbrace{P(t_i | t_{i-1})}^{\text{transition}}$$

Decoding Algorithms

Brute Force, $O(N^T)$

N states (POS tags) and T observations (word sequence length)

Viterbi Algorithm, $O(T * N^2)$

Dynamic programming algorithm to find the most likely sequence of hidden states

Viterbi Algorithm

Key Idea

Build up the best path incrementally by storing the maximum probability of reaching each state at each time step, along with the path that led there.

Algo Steps

1. Initialization $v1(s) = \pi s \times bs(o1)$, *for all s*

Start probability × emission for first observation

2. Recursion $vt(j) = MAXi\ vt\text{-}1(i) \times aij \times bj(ot)$

Take MAX, store backpointer

3. Termination $P* = MAXj\ vT(j)$

Highest final probability

4. Backtrace *Follow backpointers*

Reconstruct best sequence

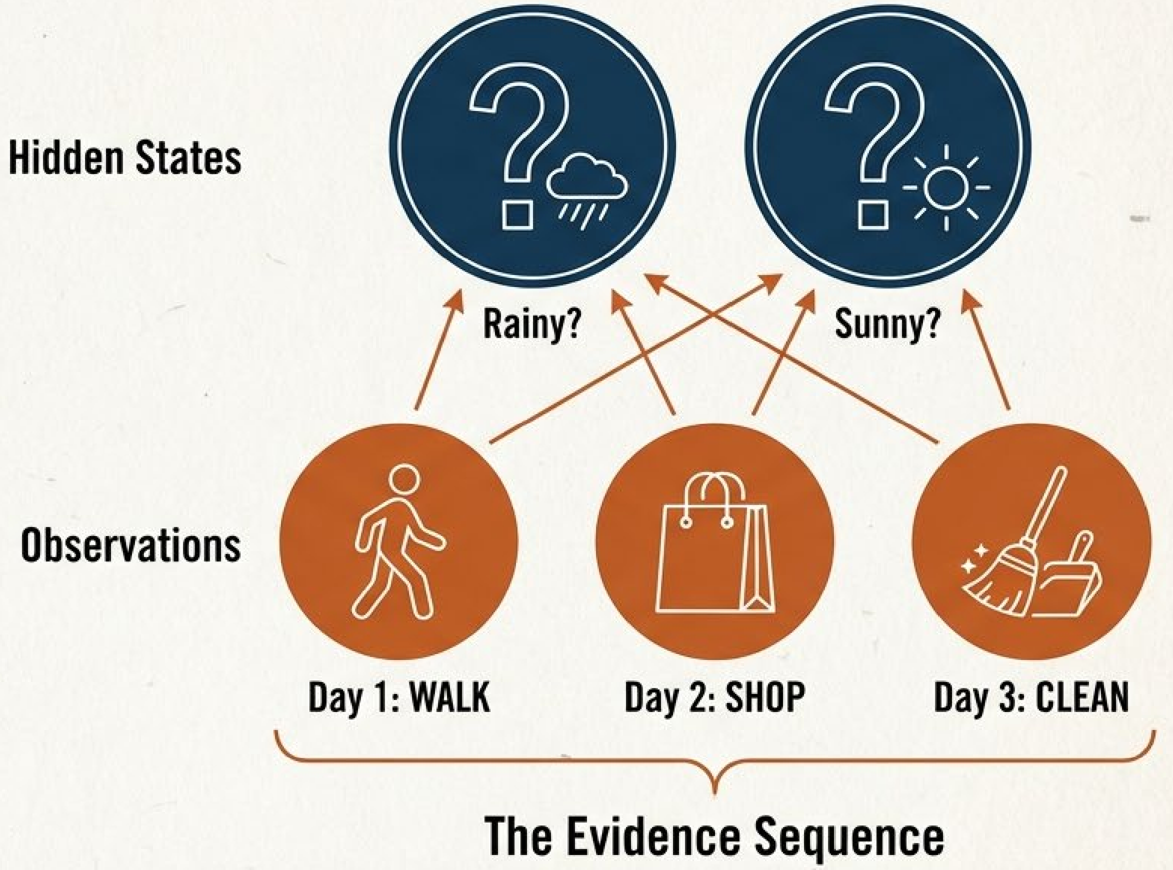
Viterbi Algorithm

function VITERBI(*observations* of len T,*state-graph* of len N) **returns** *best-path*, *path-prob*

create a path probability matrix *viterbi[N,T]*
create a backpointer matrix *backpointer[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
    $viterbi[\mathrm{s},1] \leftarrow \pi_s * b_s(o_1)$
    $backpointer[\mathrm{s},1] \leftarrow 0$
for each time step t **from** 2 **to** T **do** ; recursion step
  for each state s **from** 1 **to** N **do**
    $viterbi[\mathrm{s,t}] \leftarrow \max_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
    $backpointer[\mathrm{s,t}] \leftarrow \operatorname{argmax}_{s'=1}^{N} viterbi[s',t-1] * a_{s',s} * b_s(o_t)$
$bestpathprob \leftarrow \max_{s=1}^{N} viterbi[s,T]$; termination step
$bestpathpointer \leftarrow \operatorname{argmax}_{s=1}^{N} viterbi[s,T]$; termination step
bestpath $\leftarrow$ the path starting at state *bestpathpointer*, that follows backpointer[] to states back in time
return *bestpath*, *bestpathprob*

Viterbi Algorithm in Action: Scenario

The Scenario:
We have a friend, Bob. We never see the weather where he lives (Hidden States), but we see his daily blog posts about his activities (Observations).



Viterbi Algorithm in Action: Given

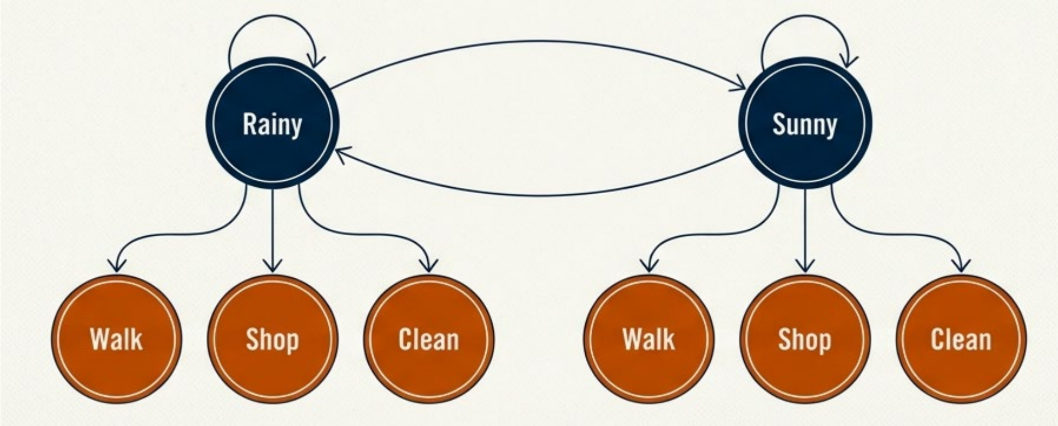
Transition Probabilities (Hidden States)

	Rainy	Sunny
Rainy	0.7	0.3
Sunny	0.4	0.6

Emission Probabilities (Observations)

	Walk	Shop	Clean
Rainy	0.1	0.4	0.5
Sunny	0.6	0.3	0.1

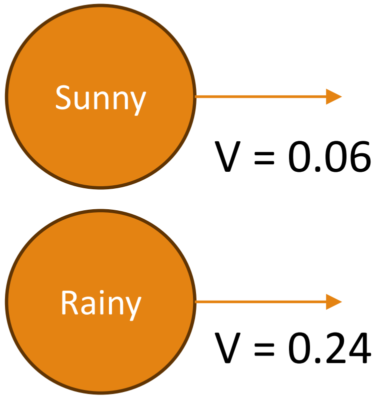
Initial Probabilities: P(Rainy) = 0.6 P(Sunny) = 0.4



Viterbi Algorithm in Action: Initialization

DAY 1: Observation Walk

- v1(Rainy) = P(Rainy) * P(Rainy | Walk) = 0.6 * 0.1 = 0.06
- V1(Sunny) = P(Sunny) * P(Sunny | Walk) = 0.4 * 0.6 = 0.24



Day 1: Walk

Viterbi Algorithm in Action: Recursion

DAY 2: Observation Shop

- *v2(Rainy) = Max (v1(Rainy) * P(Rainy|Shop) * P (Rainy|Rainy) , v1(Sunny) * P (Rainy|Shop) * P(Rainy|Sunny))*

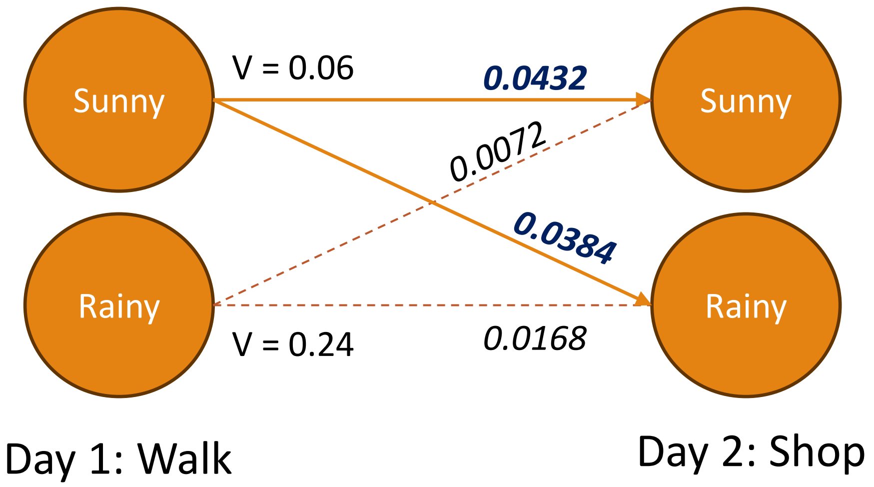
  *v2(Rainy) = Max (0.06 * 0.4 * 0.7 = 0.0168, 0.24 * 0.4 * 0.4 = **0.0384**)*

  Backpointer[Rainy, Day 1] = Sunny

- *v2(Sunny) = Max (v1(Rainy) * P(Sunny|Shop) * P (Sunny|Rainy) , v1(Sunny) * P (Sunny|Shop) * P(Sunny|Sunny))*

  *v2(Sunny) = Max (0.06 * 0.3 * 0.4 = 0.0072, 0.24 * 0.3 * 0.6 = **0.0432**)*

  Backpointer[Sunny, Day 1] = Sunny



Viterbi Algorithm in Action: Recursion

DAY 3: Observation Clean

- *v3(Rainy) = Max (v2(Rainy) * P(Rainy|Clean) * P (Rainy|Rainy) , v2(Sunny) * P (Rainy|Clean) * P(Rainy|Sunny))*

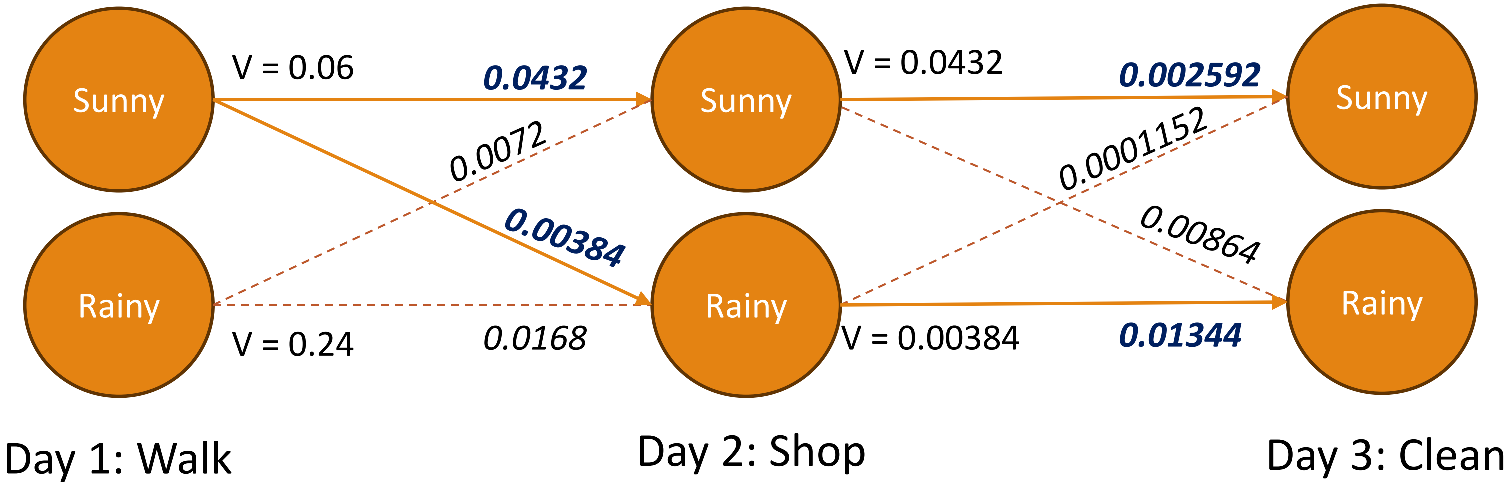
  *v3(Rainy) = Max (0.0384 * 0.5 * 0.7 = **0.01344**, 0.0432 * 0.5 * 0.4 = 0.00864)*

  Backpointer[Rainy, Day 2] = Rainy

- *v3(Sunny) = Max (v2(Rainy) * P(Sunny|Clean) * P (Sunny|Rainy) , v2(Sunny) * P (Sunny|Clean) * P(Sunny|Sunny))*

  *v3(Sunny) = Max (0.00384 * 0.1 * 0.3 = 0.0001152, 0.0432 * 0.1 * 0.6 = **0.002592**)*

  Backpointer[Sunny, Day 2] = Sunny

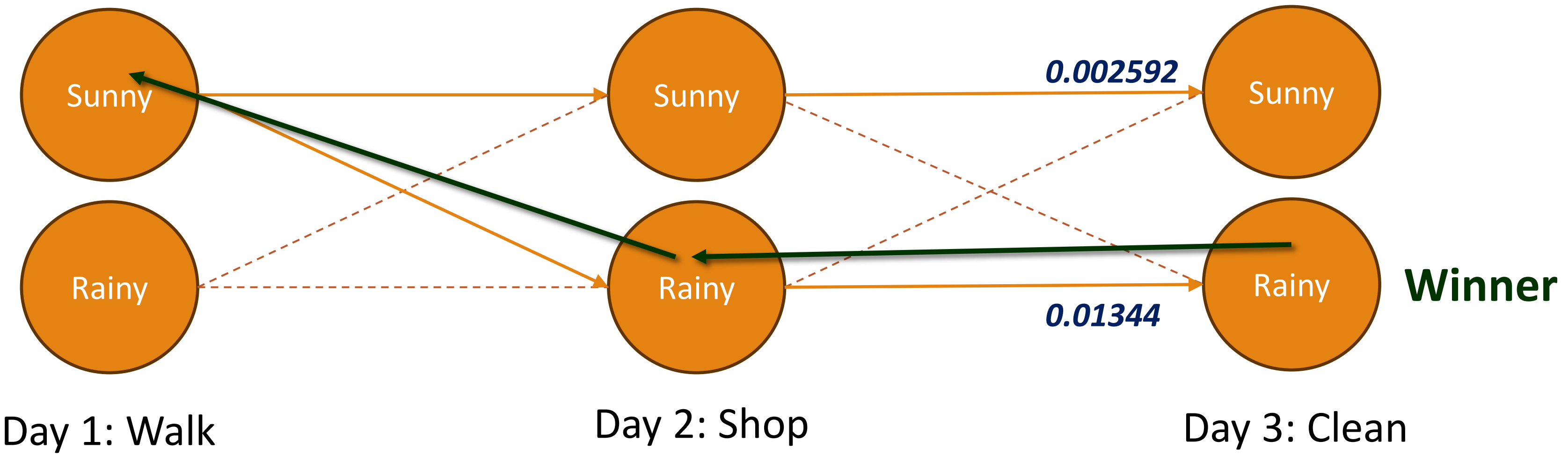


Viterbi Algorithm in Action: Termination and Backtrace

Termination

*P = Max(v3(Sunny), v3(Rainy)) = Max(0.002592**, 0.01344**) = **0.01344** → **Last Hidden State is Rainy***

Backtrace



Final Sequence: Sunny → Rainy → Rainy