POS Tagging

Topic 7

Outline

- Word Classes- Recap
- POS Tags
- POS Tagging
  - Rule Bases
  - HMM based

Parts of Speech

You may be familiar with Parts of speech in English

Eight parts of speech: **noun, verb, pronoun, preposition, adverb, conjunction, participle, and article**

Proper names are another important and anciently studied linguistic category → **Named Entity**

NLTK-Example

POS and Named Entities

Parts of speech (also known as POS) and named entities are useful clues to sentence structure and meaning

Knowing whether a word is a noun or a verb tells us about likely neighboring words (nouns in English are preceded by determiners and adjectives, verbs by nouns) and syntactic structure (verbs have dependency links to nouns), making part-of-speech tagging a key aspect of parsing.

Knowing if a named entity like *Washington* is a name of a person, a place, or a university is important to many natural language understanding tasks like question answering, instance detection, or information extraction.

Pronunciation depends on the POS category (CONtent as noun and conTENT as adjective)

The Task

- POS Tagging is the task of taking a sequence of words and assigning each word a part of speech like NOUN, VERB
- The task of **named entity recognition(NER)**, assigning words or phrases tags like PERSON, LOCATION, or ORGANIZATION.
- **Sequence Labelling Task**
  - **X_1…...x_n assign a sequence of label y_1….y_n**

POS Categories

	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 tagset

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coord. conj.	*and, but, or*	NNP	proper noun, sing.	*IBM*	TO	"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, one's*	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*

Figure 8.2 Penn Treebank part-of-speech tags.

Example

- There/PRO/EX are/VERB/VBP 70/NUM/CD children/NOUN/NNS there/ADV/RB ./PUNC/.
- Preliminary/ADJ/JJ findings/NOUN/NNS were/AUX/VBD reported/VERB/VBN in/ADP/IN today/NOUN/NN ’s/PART/POS New/PROPN/NNP England/PROPN/NNP Journal/PROPN/NNP of/ADP/IN Medicine/PROPN/NNP

POS Tagging

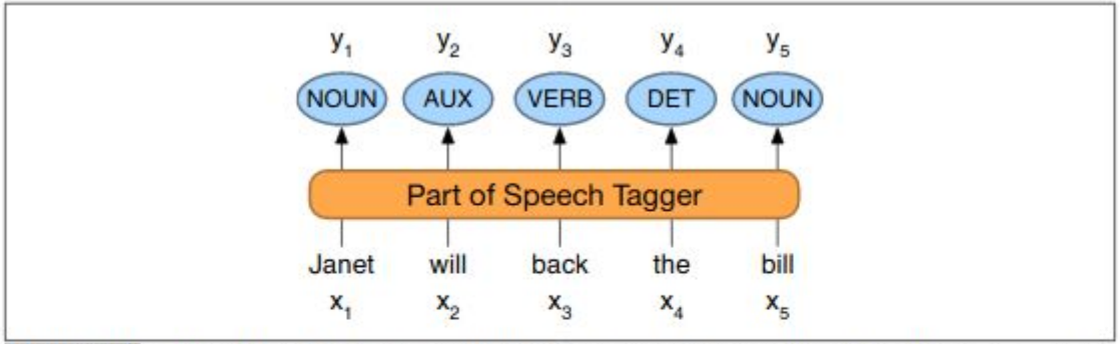


Figure 8.3 The task of part-of-speech tagging: mapping from input words $x_1, x_2, \ldots, x_n$ to output POS tags $y_1, y_2, \ldots, y_n$.

POS Tag provide disambiguation

earnings growth took a **back/JJ** seat
a small building in the **back/NN**
a clear majority of senators **back/VBP** the bill
Dave began to **back/VB** toward the door
enable the country to buy **back/RP** debt
I was twenty-one **back/RB** then

What is POS Tagging Problem?

- Input: a sequence of string of words and a specified tagset
- Output: a sequence of single best tag for each word
- Example:
  - Book that flight. → Book\VB that\DT flight\NN .\.

What is the challenge?

Some tagging decisions are ambiguous

A word can take more than one tags

Example: book

Book \VB that flight

Book\NN available in flight

The Task

POS Tagging is to ***resolve these ambiguities,*** choosing the proper tag for the context

Approaches:

- Rule based
- Stochastic/statistical

Rule Based POST

- Large database with hand-written rules to disambiguate tags
- Example EngCG tagger
- Two stage approach:
  - First stage assign all possible tags to the word (using dictionary)
  - Second stage use hand-written rules to disambiguate
- Each entry consider morphological and syntactic features
- Then apply constraints to select the correct tag

Word	POS	Additional POS features
smaller	ADJ	COMPARATIVE
entire	ADJ	ABSOLUTE ATTRIBUTIVE
fast	ADV	SUPERLATIVE
that	DET	CENTRAL DEMONSTRATIVE SG
all	DET	PREDETERMINER SG/PL QUANTIFIER
dog's	N	GENITIVE SG
furniture	N	NOMINATIVE SG NOINDEFDETERMINER
one-third	NUM	SG
she	PRON	PERSONAL FEMININE NOMINATIVE SG3
show	V	PRESENT -SG3 VFIN
show	N	NOMINATIVE SG
shown	PCP2	SVOO SVO SV
occurred	PCP2	SV
occurred	V	PAST VFIN SV

ADVERBIAL-THAT RULE

Given input: "that"

if

(+1 A/ADV/QUANT); /∗ *if next word is adj, adverb, or quantifier* ∗/

(+2 SENT-LIM); /∗ *and following which is a sentence boundary,* ∗/

(NOT -1 SVOC/A); /∗ *and the previous word is not a verb like* ∗/

/∗ *'consider' which allows adjs as object complements* ∗/

then eliminate non-ADV tags

else eliminate ADV tag

HMM- POS Tagger

- Probability for tagging
- Hidden Markov Model
- POS tagging is considered as a sequence classification task
  - For sequence of words assign sequence of tags
- Find most probable tag sequence
- Given a sequence of words w1...wn, out of all possible sequences of n tags, we need to find tag sequence which maximize the probability P(t1...tn|w1..wn)
- $$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n | w_1^n)$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$$

We can conveniently simplify 5.26 by dropping the denominator $P(w_1^n)$. Why is that? Since we are choosing a tag sequence out of all tag sequences, we will be computing $\frac{P(w_1^n|t_1^n)P(t_1^n)}{P(w_1^n)}$ for each tag sequence. But $P(w_1^n)$ doesn't change for each tag sequence; we are always asking about the most likely tag sequence for the same observation w_1^n, which must have the same probability $P(w_1^n)$. Thus we can choose the tag sequence which maximizes this simpler formula:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(w_1^n|t_1^n)P(t_1^n)$$

To summarize, the most probable tag sequence $\hat{t}_1^n$ given some word string w_1^n can be computed by taking the product of two probabilities for each tag sequence, and choosing the tag sequence for which this product is greatest. The two terms are the **prior probability** of the tag sequence $P(t_1^n)$), and the **likelihood** of the word string $P(w_1^n|t_1^n)$:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} \overbrace{P(w_1^n|t_1^n)}^{\text{likelihood}} \overbrace{P(t_1^n)}^{\text{prior}}$$

Unfortunately, the above eqn is hard to compute directly. HMM taggers therefore make two simplifying assumptions. The first assumption is that the probability of a word appearing is dependent only on its own part-of-speech tag; that it is independent of other words around it, and of the other tags around it:

$$P(w_1^n|t_1^n) \approx \prod_{i=1}^{n} P(w_i|t_i)$$

The second assumption is that the probability of a tag appearing is dependent only on the previous tag, the **bigram** assumption

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i|t_{i-1})$$

Plugging the simplifying assumptions (5.29) and (5.30) into (5.28) results in the following equation by which a bigram tagger estimates the most probable tag sequence:

$$\hat{t}_1^n = \underset{t_1^n}{\operatorname{argmax}} P(t_1^n|w_1^n) \approx \underset{t_1^n}{\operatorname{argmax}} \prod_{i=1}^{n} P(w_i|t_i)P(t_i|t_{i-1})$$

Markov Chain

- A weighted finite-state automaton is a simple augmentation of the finite automaton in which each arc is associated with a probability, indicating how likely that path is to be taken.
- The probability on all the arcs leaving a node must sum to 1.
- A Markov chain is a special case of a weighted automaton in which the input sequence uniquely determines which states the automaton will go through.

MARKOV CHAIN

A **Markov chain** is a special case of a weighted automaton in which the input sequence uniquely determines which states the automaton will go through. Because it can't represent inherently ambiguous problems, a Markov chain is only useful for assigning probabilities to unambiguous sequences.

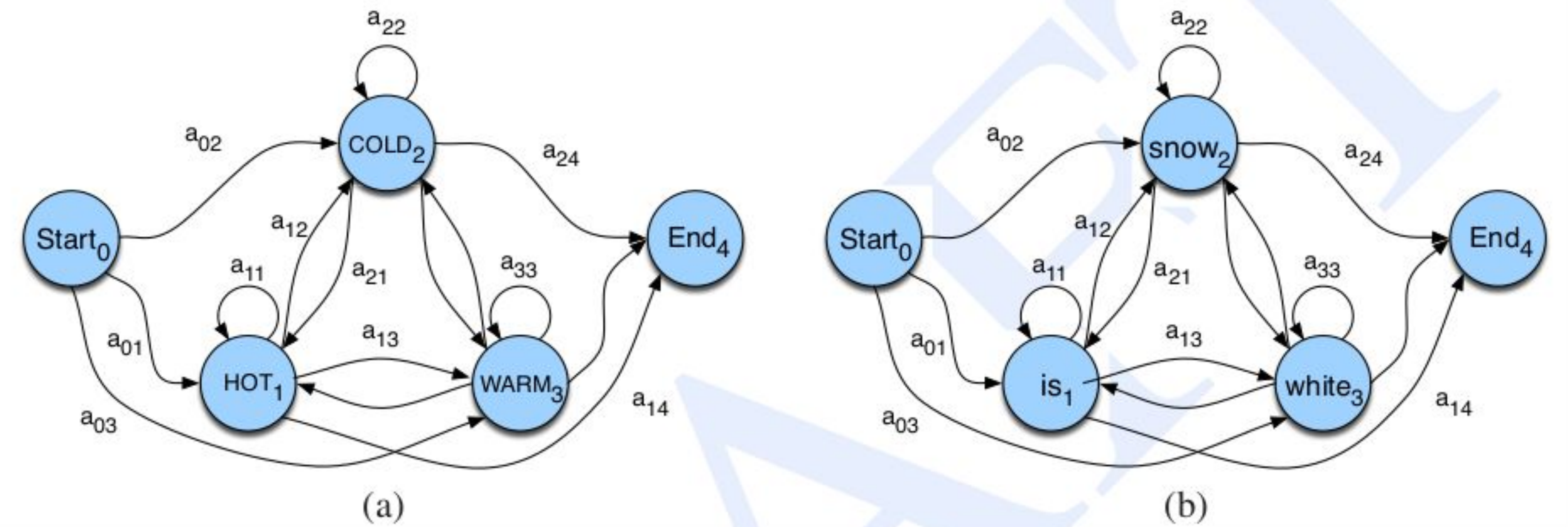


Figure 6.1 A Markov chain for weather (a) and one for words (b). A Markov chain is specified by the structure, the transition between states, and the start and end states.

Markov Chain

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{01} a_{02} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
q_0, q_F	a special **start state** and **end (final) state** which are not associated with observations.

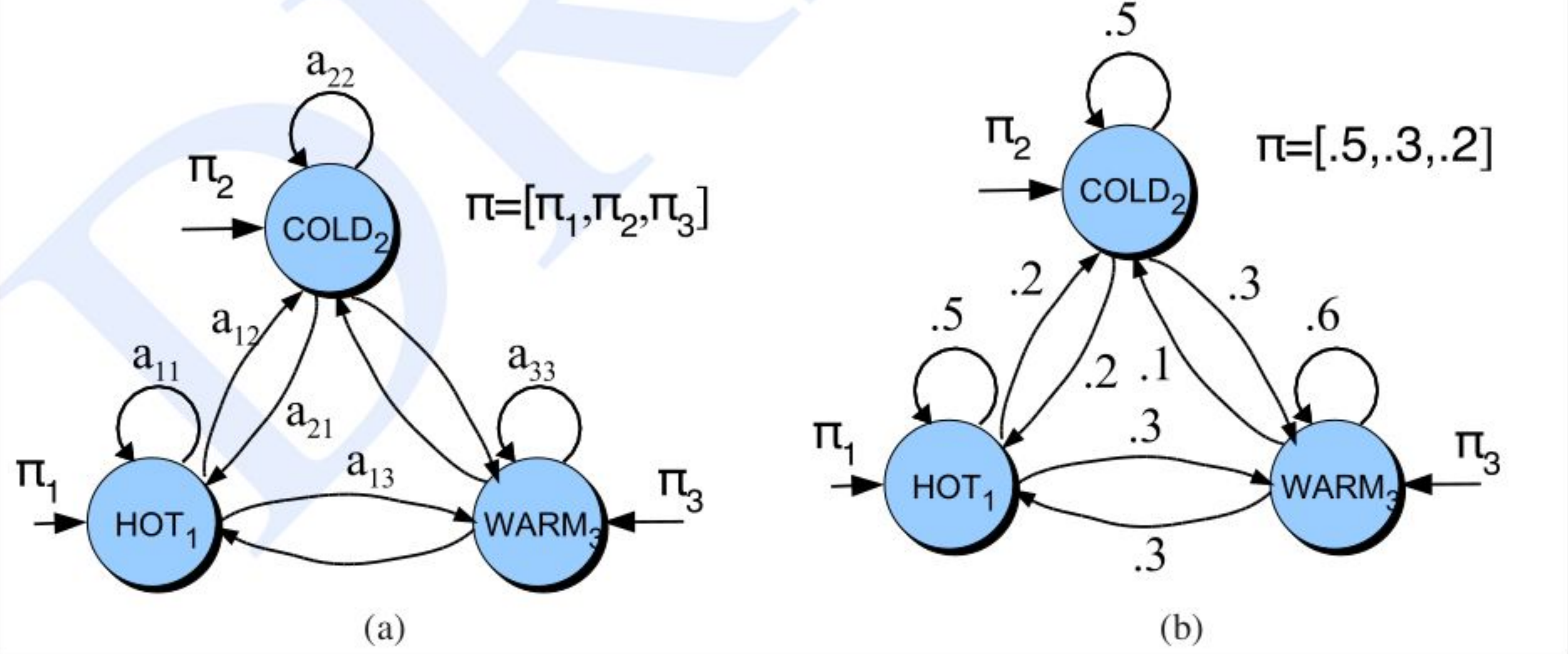
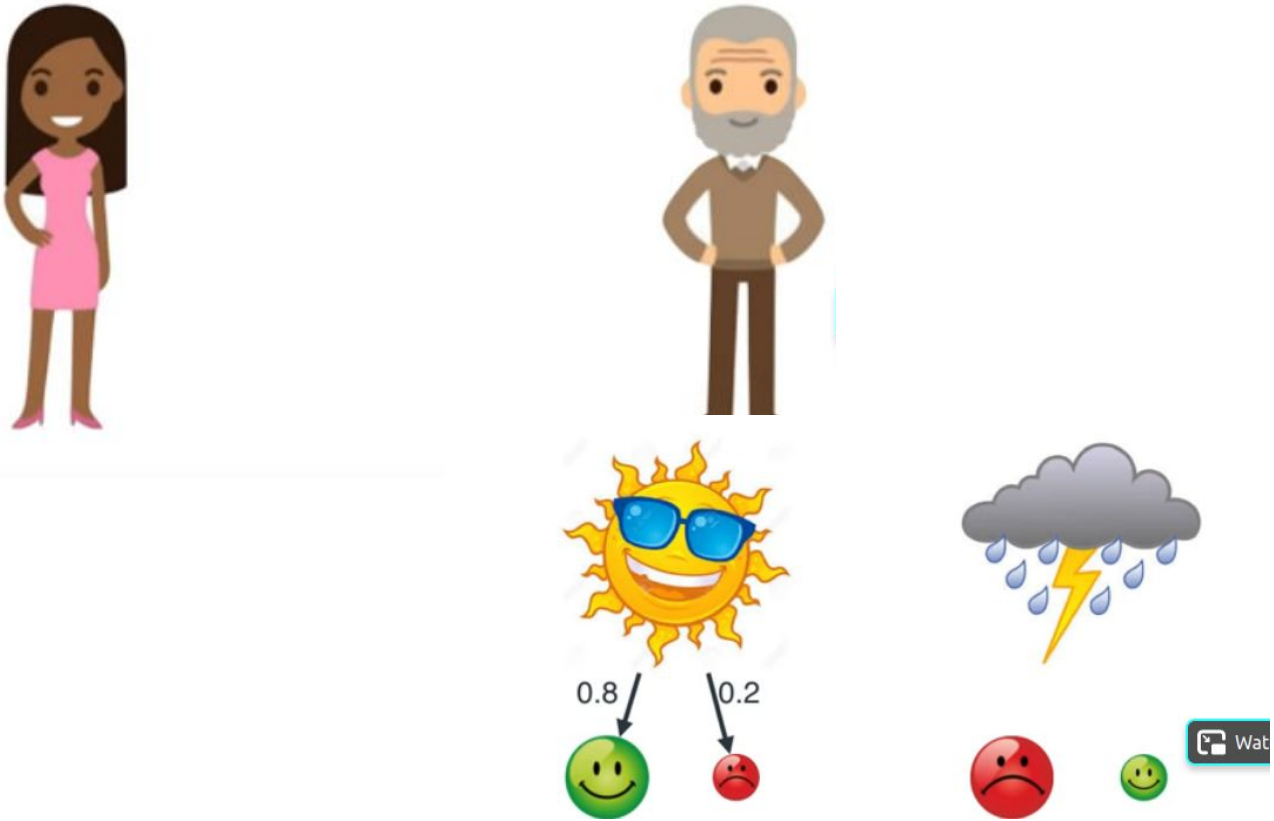


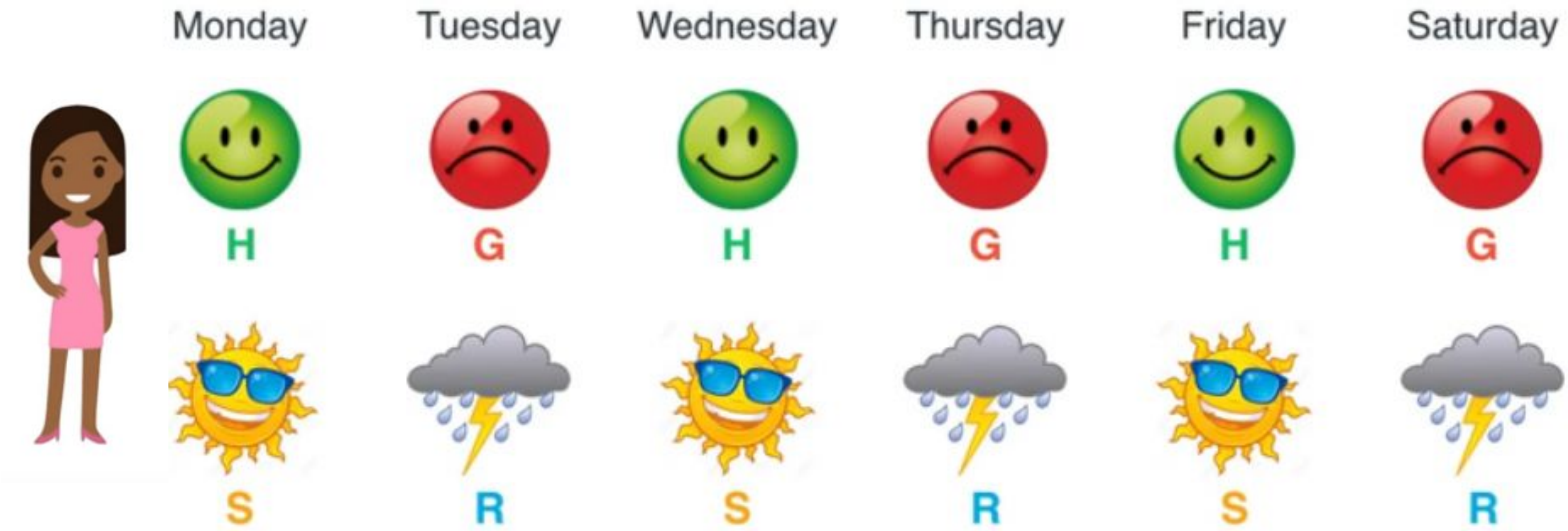
Figure 6.2 Another representation of the same Markov chain for weather shown in Fig. 6.1. Instead of using a special start state with a_{01} transition probabilities, we use the π vector, which represents the distribution over starting state probabilities. The figure in (b) shows sample probabilities.

HMM

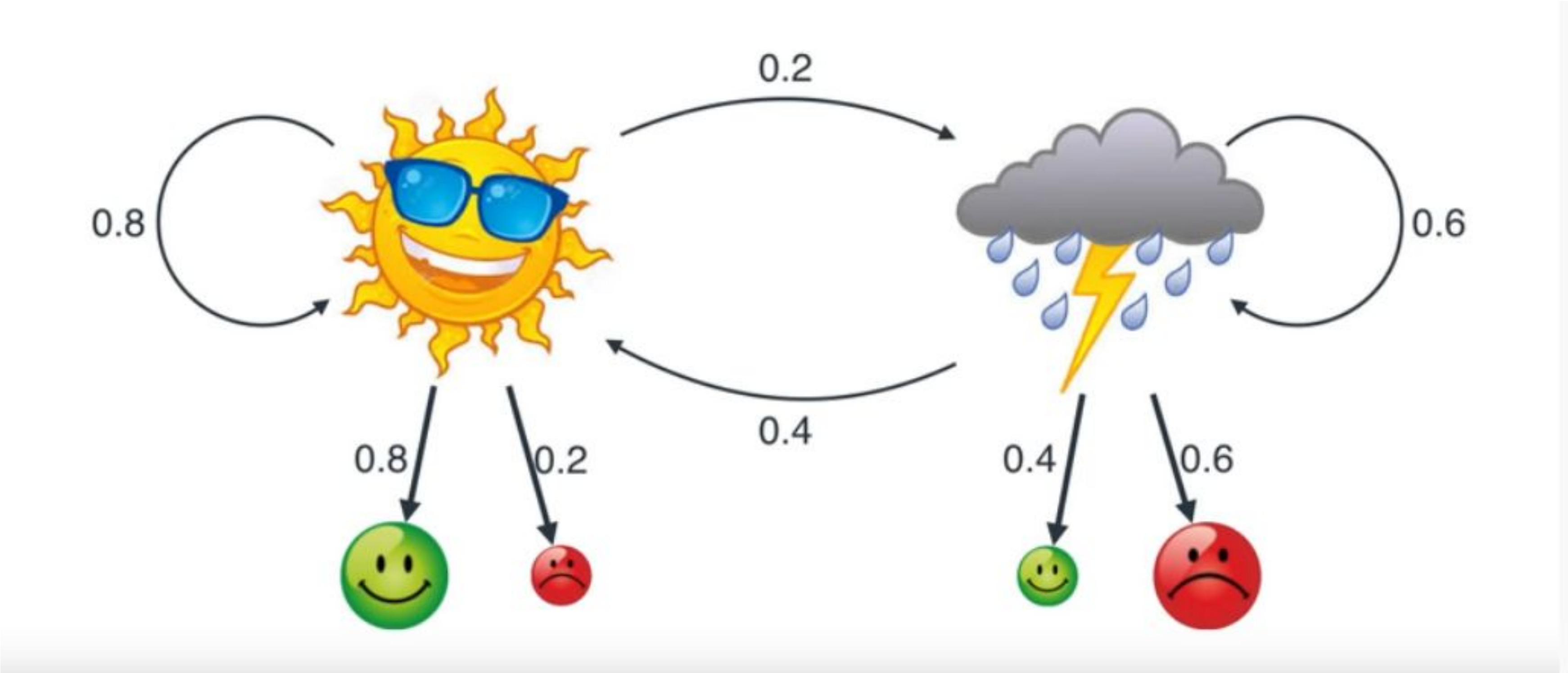
- A Markov chain is useful when we need to compute a probability for a sequence of observable events.
- In many cases, however, the events we are interested in arehidden: we don’t observe them directly.
- A simple story will be better


0.8
0.2

Alice assumes the weather from Bob's mood



But weather depends on other factors- Hidden from Alice



HMM

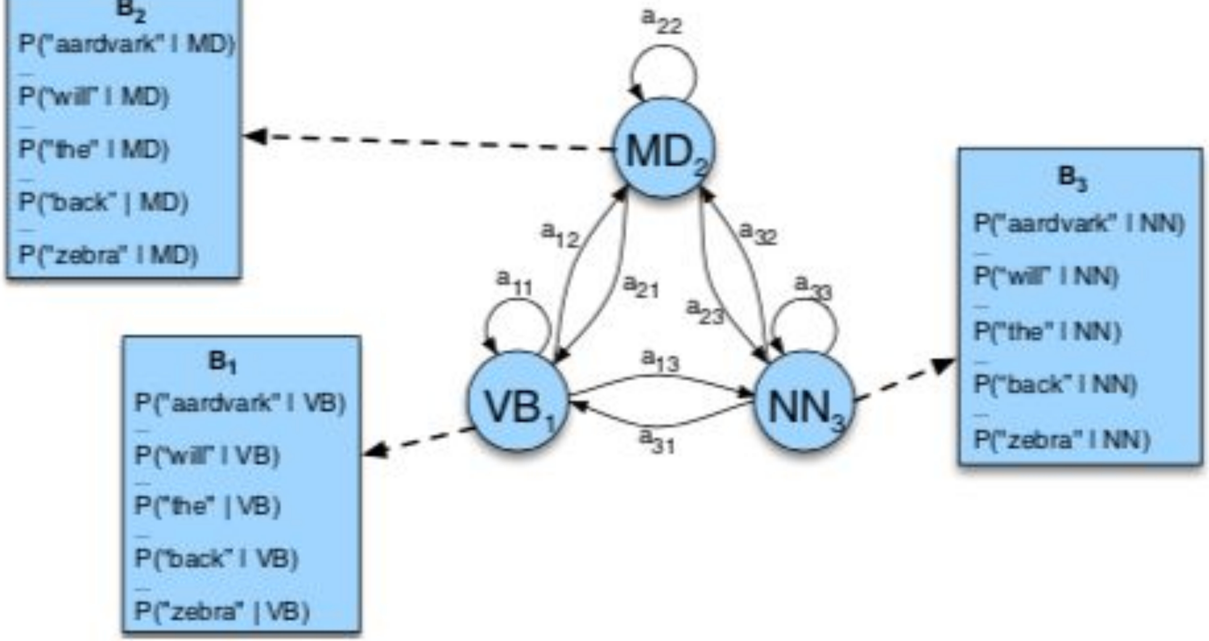
In hidden Markov model(HMM), the states (weather conditions) are not observable, but when hidden Markov model we visit a state, an observation (Bob's mood) is recorded that is a probabilistic function of the state

HMMs allow us to compute the joint probability of a set of hidden states given a set of observed states.   Once we know the joint probability of a sequence of hidden states, we determine the best possible sequence i.e. the sequence with the highest probability and choose that sequence as the best sequence of hidden states.

HMM

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} a_{12} \ldots a_{n1} \ldots a_{nn}$	a **transition probability matrix** A, each a_{ij} representing the probability of moving from state i to state j, s.t. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$O = o_1 o_2 \ldots o_T$	a sequence of T **observations**, each one drawn from a vocabulary $V = v_1, v_2, ..., v_V$.
$B = b_i(o_t)$	a sequence of **observation likelihoods:**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i.
q_0, q_F	a special **start state** and **end (final) state** which are not associated with observations, together with transition probabilities $a_{01} a_{02} .. a_{0n}$ out of the start state and $a_{1F} a_{2F} ... a_{nF}$ into the end state.

HMM


B2
P("aardvark" | MD)
...
P("will" | MD)
...
P("the" | MD)
...
P("back" | MD)
...
P("zebra" | MD)
B1
P("aardvark" | VB)
...
P("will" | VB)
...
P("the" | VB)
...
P("back" | VB)
...
P("zebra" | VB)
B3
P("aardvark" | NN)
...
P("will" | NN)
...
P("the" | NN)
...
P("back" | NN)
...
P("zebra" | NN)
MD2
VB1
NN3
a22
a11
a33
a12
a21
a32
a23
a13
a31