Review

Notes

- The due date for the **first** assignment Sep 20th
- Will upload the **second** assignment on Sep 20th.
- By the end of September, you must already have groups to work with (you can also work individually)

Language Modeling

- The Maximum Likelihood Estimate (MLE) $$P(w_i \mid w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$$

- Smoothing (discounting) techniques:
  - Laplace smoothing – steal too much probabilities for zero counts
  - Backoff and interpolation – usually integrated with other smoothing techniques
  - Good Turing smoothing – use the counts for words that we have seen once
  - Extended Kneser-Ney– use $P_{continuation}$ to estimate low or zero frequent words
  - Stupid backoff – helpful for very large web-scale dataset

Kneser-Ney Smoothing – Example (Correction)

- <s> they put on their clothes <\s>
- <s> the paper is put onto the porch <\s>
- <s> Adam put the car in the garage <\s>
- <s> the girl put the keys on the garage <\s>

Find P_{KN}(the|put).

d=0.75

$$P_{KN}(w_i \mid w_{i-1}) = \frac{\max(c(w_{i-1}, w_i) - d, 0)}{c(w_{i-1})} + \lambda(w_{i-1}) P_{CONTINUATION}(w_i)$$

First term=max(2-0.75,0)/4=1.25/4=0.3125
Lambda = (0.75/4)*3=0.5625
$P_{Continuation}$= 5/27=0.185

$$\lambda(w_{i-1}) = \frac{d}{c(w_{i-1})} \left| \{ w : c(w_{i-1}, w) > 0 \} \right|$$

P_{KN}(the|put)=0.3125+(0.5625*0.185)

$$P_{CONTINUATION}(w) = \frac{\left| \{ w_{i-1} : c(w_{i-1}, w) > 0 \} \right|}{\left| \{ (w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0 \} \right|}$$

CSCI 3907/6907

Fall 2019

Lecture 4

Parts-of-Speech Tagging and Text Classification

Parts-of-Speech Tagging

- Identifying word classes (e.g. Verb, Noun, Adjective)

- The process of assigning a part of speech to each word in a corpus
  - **Input:** sentence (sequence of words) **output:** a tag for each word in the corpus

- Used as features in some applications
  - Information extraction → Identifying named entities
  - Speech synthesis → *CON*tent (noun) vs. con*TENT* (adj)
  - Parsing, co-reference resolution, word sense disambiguation, sentiment analysis, question answering, etc.

Word Classes

- Words that share:
  - Syntactic and distributional properties
    - similar neighboring words or classes
  - Morphological properties
    - take similar affixes

- ~9 traditional word classes of parts of speech
  - Noun, verb, adjective, preposition, adverb, article, interjection, pronoun, conjunction

Closed vs. Open Classes

- Closed class: relatively fixed set
  - Prepositions: of, in, by, …
  - Auxiliaries: may, can, will, had, been, …
  - Pronouns: I, you, she, mine, his, them, …
  - Usually function words (short common words which play a role in grammar)

- Open class: productive
  - English has 4: Nouns, Verbs, Adjectives, Adverbs
  - New open class words are coined or borrowed from other languages
    - Example words added to OED in 2018: Facebook (v.), whataboutery(n.), teenagery(adj.), microagression(n.), off-script (adv).

Nouns

- Semantic definitions of word classes don't hold up
  - For example: "nouns refer to people, places or things"
  - Nouns include concrete terms (pencil, ship), abstractions (bandwidth, relation), and verb-like terms (pacing, run, work).
- Syntactic and morphological definitions are more consistent
  - Ability to occur with determiners (the ship, its relation, a run)
    - Exceptions: *The Mary
  - To take possessive 's (Mary's work)
  - Can occur in a plural form
    - Exceptions: *waters, *snows

Noun Categories

- Proper nouns vs. common nouns
  - Proper nouns → Names of specific persons or entities
    - Mary, IBM, Colorado, United Nations
  - In English, Typically capitalized and can't follow articles
  - Common nouns:
    - Count nouns: can occur in plural, and can be counted
      - a book, two books, *3 salts
    - Mass nouns:
      - snow, salt, sugar, water, etc.

Verbs

- Refer to actions and processes
- In English, most verbs have morphological inflections:
  - Non-third-person-singular (eat)
  - Third-person-singular (eats)
  - Progressive (eating)
  - Past (ate)
  - Past-participle (eaten)

Adjectives

- Adjectives: identify properties or qualities of nouns
  - Color (black, while), size (small, big), age (young, old), value (good, bad) …

- Not all languages have adjectives
  - In Korean, adjectives are expressed as verbs (“to be good”, instead of “good”)

Adverbs

- Also modify things (verbs, adjectives, other adverbs)
  - The very happy man walked home extremely slowly yesterday.

- Types of Adverbs:
  - Directional/locative adverbs (here, there, home)
  - Degree adverbs (extremely, very, somewhat)
  - Manner adverbs (slowly, delicately)
  - Temporal adverbs (Monday, tomorrow)
    - Sometimes tagged as nouns

Closed classes

- Relatively fixed membership (new prepositions and pronouns are rarely coined).
- Differ more by language than open classes.
- English has the following main closed classes:

prepositions: on, under, over, near, by, at, from, to, with
particles: up, down, on, off, in, out, at, by
determiners: a, an, the
conjunctions: and, but, or, as, if, when
pronouns: she, who, I, others
auxiliary verbs: can, may, should, are
numerals: one, two, three, first, second, third

Closed Classes in English

- **Prepositions**: occur before noun phrases, and often denote spatial or temporal relations (literal or metaphorical).
  - Literal: before then, by the house, on the table
  - Metaphorical: on time, beside herself

- **Particles**: resemble prepositions or adverbs, and are used in combination with verbs to make phrasal verbs:
  - Move on, show up, turn down, rule out

Closed Classes in English

- **Determiners**: occur before nouns
  - Articles: a, an, the
  - This, that
- **Conjunctions**: join two phrases, clauses, or sentences.
  - Coordinating conjunctions: and, or, but, etc.
  - Subordinating conjunctions: that (I thought that …)
- **Pronouns**: a shorthand for some noun phrase
  - Personal pronouns: I, you, him, her, etc.
  - Possessive pronouns: hers, mine, his, your, yours, etc.
  - Wh-pronouns: who, what, whom, when, etc.

Closed Classes in English

- **Auxiliary Verbs:**
  - Typically mark some semantic features like tense, aspect, polarity, or mood.
  - In English, the verbs *be*, *do*, *have*, and modal verbs (can, may, must).

- Other classes in English:
  - Interjections (oh, hey, alas)
  - negatives (no, not)
  - politeness markers (please, thanks)
  - greetings (hello, bye)
  - the existential there (*there* is a …)

Tagsets

- Vary in number of tags: a dozen to over 200
- Size of tag sets depends on language and purpose
  - Some tagging approaches (e.g., constraint grammar based) make fewer distinctions
    - e.g., conflating prepositions, conjunctions, particles in one class

Choosing a Tagset

- To do POS tagging, first need to choose a set of tags
  - Could pick coarse-grained (small) tagsets
    - Universal tagset: 17 tags
  - Or a fine-grained tagset: Brown Corpus (Francis & Kucera '82), 1M words, 87 tags
    - more informative but more difficult to tag

- Most commonly used: Penn Treebank (PTB):
  - hand-annotated corpus of Wall Street Journal, 1M words, 45 tags (subset of the Brown corpus tags)

PTB Tagset

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*’s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential ‘there’	*there*	PRP$	possess. pronoun	*your, one’s*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	*$*
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	*#*
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

Example

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

- **There**/EX **are**/VBP **70**/CD **children**/NNS **there**/RB

- -- findings/NNS were/VBD **reported/VBN**---

Ambiguity in Tagging

- Ambiguous POS contexts
  - E.g., Time flies like an arrow.
    Time/[V,N] **flies**/[V,N] **like**/[V,Prep] **an**/Det **arrow**/N
  - Possible POS assignments
    - Time/**N** flies/**V** like/Prep an/Det arrow/N
    - Time/**V** flies/**N** like/Prep an/Det arrow/N
    - Time/**N** flies/**N** like/**V** an/Det arrow/N
    - …..

Ambiguity in Tagging

- Degree of ambiguity in English (based on Brown corpus)
  - 11.5% of word types are ambiguous.
  - 40% of word tokens are ambiguous.

- Average POS tagging disagreement amongst expert human judges for the Penn treebank was 3.5%

Resolving Ambiguity

- Many words have only one POS tag
  - e.g. is, Mary, very, smallest

- Tags also tend to co-occur regularly with other tags (e.g. DET, N)
  - We can use hand-crafted rules to disambiguate.
  - We can use probabilities:
    - conditional probabilities of word bigrams $P(w_n|w_{n-1})$
    - look at tag bigram likelihoods $P(t_n|t_{n-1})$

Baseline

- Picking the most frequent tag for each specific word type (~90% accuracy)

E.g., Time flies like an arrow.
Time/[V,N] **flies**/[V,N] **like**/[V,Prep] **an**/Det **arrow**/N

Time (V) → 40

Time (N) → 20

Flies (V) → 15

Flies (N) → 5

- We tag *time* as a verb and *flies* as a verb following the most frequent approach

Approaches

- **Rule-Based**: Human crafted rules based on lexical and other linguistic knowledge

- **Learning-Based**: Trained on human annotated corpora like the Penn Treebank
  - Statistical models: Hidden Markov Model (HMM), Maximum Entropy Markov Model(MEMM), Conditional Random Field (CRF)
  - Rule learning: Transformation Based Learning(TBL)

- learning-based approaches → more effective overall

Evaluation

- Compare predicted tags against human annotated labels
  - Accuracy: the percentage of tags correctly labeled (matching human labels) **on the test set**
  - State-of-the-art taggers achieve ~97% tag accuracy in English

Evaluation - Confusion Matrix

- Confusion matrix (contingency table):
  - E.g. which tags did we most often confuse with which other tags?
  - How much of the overall error does each confusion account for?

- Used to analyze performance and possibly improve the model.

	VB	TO	NN
VB			
TO			
NN			

Evaluation - Confusion Matrix

	IN	JJ	NN	NNP	RB	VBD	VBN
IN	-	0.2			0.7		
JJ	0.2	-	3.3	2.1	1.7	0.2	2.7
NN		8.7	-				0.2
NNP	0.2	3.3	4.1	-	0.2		
RB	2.2	2.0	0.5		-		
VBD		0.3	0.5			-	
VBN		2.8				2.6	-

Rule Based Tagging

- Start with a dictionary of words and possible tags
- Assign all possible tags to words using the dictionary
- Write rules by hand to selectively remove tags
  - Typically >1k rules
- Stop when each word has exactly one (presumably correct) tag

Example: Tag Dictionary

- **she:** PRP
- **promised**: VBN,VBD
- **to**: TO
- **back**: VB, JJ, RB, NN
- **the**: DT
- **bill**: NN, VB

Example

- Assign all possible tags to each word in a sentence

She	Promised	To	Back	The	Bill
PRP	VBN VBD	TO	VB NN JJ RB	DT	NN VB

Example

- Apply rules to eliminate some tags:
  - Eliminate VBN if VBD is an option when VBN|VBD follows “PRP”

She	Promised	To	Back	The	Bill
PRP	~~VBN~~ VBD	TO	VB NN JJ RB	DT	NN VB

Example

- Apply rules to eliminate some tags:
  - Eliminate NN/RB/JJ if VB is an option following a “TO”

She	Promised	To	Back	The	Bill
PRP	~~VBN~~ VBD	TO	VB ~~NN~~ ~~JJ~~ ~~RB~~	DT	NN VB

Example

- Apply rules to eliminate some tags:
  - Eliminate VB if NN is an option following a “DT”

She	Promised	To	Back	The	Bill
PRP	~~VBN~~ VBD	TO	VB ~~NN~~ ~~JJ~~ ~~RB~~	DT	NN ~~VB~~

Transformation Based Learning (TBL)

- Combines Rule-based and Stochastic Tagging
  - Like rule-based → specify tags in a certain environment
  - Like stochastic-based → use a tagged corpus to find the best performing rules

- Rules are learned from data
  - Input:
    - Tagged corpus
    - Dictionary (with most frequent tags for each word)

TBL Algorithm

- Step 1: Label every word with most likely tag (from a dictionary)
- Step 2: Check every possible transformation & select the one that results in the most improved tagging accuracy
- Step 3: Re-tag corpus applying this rule, and add rule to end of rule set
- Repeat 2-3 until some stopping criterion is reached,
  - e.g., X% correct with respect to training corpus

OUTPUT → an ordered set of transformation rules

Rule Learning

- There are infinitely many possible transformation rules:
  - The algorithm needs a way to limit the set of possible transformations
  - This is done using a small set of templates
    - Every allowable transformation is an instantiation of one of the templates
  - Additional constraints can be added for efficiency

Rule Templates for TBL

Rule: Change current tag from **x** to **y** IF:

The preceding (following) word is tagged **z**.
The word two before (after) is tagged **z**.
One of the two preceding (following) words is tagged **z**.
One of the three preceding (following) words is tagged **z**.
The preceding word is tagged **z** and the following word is tagged **w**.
The preceding (following) word is tagged **z** and the word two before (after) is tagged **w**.

TBL Learning Algorithm Sketch

```
function TBL(corpus) returns transforms-queue
  INTIALIZE-WITH-MOST-LIKELY-TAGS(corpus)
  until end condition is met do
    templates ← GENERATE-POTENTIAL-RELEVANT-TEMPLATES
    best-transform ← GET-BEST-TRANSFORM(corpus, templates)
    APPLY-TRANSFORM(best-transform, corpus)
    ENQUEUE(best-transform-rule, transforms-queue)
  end
  return(transforms-queue)
```

Examples of Learned Rules

#	Change tags From	To	Condition	Example
1	NN	VB	Previous tag is TO	to/TO race/NN → VB
2	VBP	VB	One of the previous 3 tags is MD	might/MD vanish/VBP → VB
3	NN	VB	One of the previous 2 tags is MD	might/MD not reply/NN → VB
4	VB	NN	One of the previous 2 tags is DT	
5	VBD	VBN	One of the previous 3 tags is VBZ	

PTB Tagset

Tag	Description	Example	Tag	Description	Example	Tag	Description	Example
CC	coordinating conjunction	*and, but, or*	PDT	predeterminer	*all, both*	VBP	verb non-3sg present	*eat*
CD	cardinal number	*one, two*	POS	possessive ending	*’s*	VBZ	verb 3sg pres	*eats*
DT	determiner	*a, the*	PRP	personal pronoun	*I, you, he*	WDT	wh-determ.	*which, that*
EX	existential ‘there’	*there*	PRP$	possess. pronoun	*your, one’s*	WP	wh-pronoun	*what, who*
FW	foreign word	*mea culpa*	RB	adverb	*quickly*	WP$	wh-possess.	*whose*
IN	preposition/ subordin-conj	*of, in, by*	RBR	comparative adverb	*faster*	WRB	wh-adverb	*how, where*
JJ	adjective	*yellow*	RBS	superlatv. adverb	*fastest*	$	dollar sign	*$*
JJR	comparative adj	*bigger*	RP	particle	*up, off*	#	pound sign	*#*
JJS	superlative adj	*wildest*	SYM	symbol	+,%, &	“	left quote	‘ or “
LS	list item marker	*1, 2, One*	TO	“to”	*to*	”	right quote	’ or ”
MD	modal	*can, should*	UH	interjection	*ah, oops*	(	left paren	[, (, {, <
NN	sing or mass noun	*llama*	VB	verb base form	*eat*	)	right paren	],), }, >
NNS	noun, plural	*llamas*	VBD	verb past tense	*ate*	,	comma	,
NNP	proper noun, sing.	*IBM*	VBG	verb gerund	*eating*	.	sent-end punc	. ! ?
NNPS	proper noun, plu.	*Carolinas*	VBN	verb past part.	*eaten*	:	sent-mid punc	: ; ... – -

Example

- … is expected to race tomorrow
- Labels every word with its most-likely tag
  - E.g. race occurrences in the Brown corpus:
    - $P(NN|race) = .98$
    - $P(VB|race) = .02$
  - is/VBZ expected/VBN to/TO race/NN tomorrow/NN
- Then TBL applies the following rule
  - "Change NN to VB when previous tag is TO"

    … is/VBZ expected/VBN to/TO race/NN tomorrow/NN

    becomes

    … is/VBZ expected/VBN to/TO race/VB tomorrow/NN

Sequence Modeling

- A sequence classifier → a model whose job is to assign a label or class to each unit in a sequence
  - mapping a sequence of observations to a sequence of labels
  - Compute a probability distribution over possible labels and choose the best label sequence

The girl in the → class1 class2 class3 class4 **vs.** *the girl in the → one class*

- Examples (e.g. Hidden Markov Model (HMM), Maximum Entropy Markov Model (MEMM))

Markov Chains

- A Markov chain is a probabilistic automaton specified by the following components:

$Q = q_1 q_2 \ldots q_N$	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. $\sum_{j=1}^{n} a_{ij} = 1 \quad \forall i$
$\pi = \pi_1, \pi_2, ..., \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Markov Assumption: $P(q_i = a | q_1 ... q_{i-1}) = P(q_i = a | q_{i-1})$

Example

Compute the Probability of each of the following state sequences:

1. hot hot hot hot → 0.1*0.6*0.6*0.6
2. cold hot cold hot → 0.7*0.3*0.1*0.3

$$\pi = [.1, .7., 2]$$

Hidden Markov Model

- Markov Chains are useful for computing probabilities of **observable** events

- Sometimes, the events we are interested in are unobserved directly (**hidden**)
  - For example, we don't normally observe POS tags in text. We only observe the words
  - We need to infer tags from word sequences

- HMMs allow us to model both observed and hidden events.

HMM Specifications

$Q = q_1 q_2 \ldots q_N$	a set of N **states**
$A = a_{11} \ldots a_{ij} \ldots a_{NN}$	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 $V = v_1, v_2, \ldots, v_V$
$B = b_i(o_t)$	a sequence of **observation likelihoods**, also called **emission probabilities**, each expressing the probability of an observation o_t being generated from a state i
$\pi = \pi_1, \pi_2, \ldots, \pi_N$	an **initial probability distribution** over states. π_i is the probability that the Markov chain will start in state i. Some states j may have $\pi_j = 0$, meaning that they cannot be initial states. Also, $\sum_{i=1}^{n} \pi_i = 1$

Assumptions

Assumptions of a first-order Hidden Markov Model:

Markov Assumption: $P(q_i|q_1 \ldots q_{i-1}) = P(q_i|q_{i-1})$

- Similar to Markov Chains:
  - each state depends only on the previous state

Output Independence: $P(o_i|q_1 \ldots q_i, \ldots, q_T, o_1, \ldots, o_i, \ldots, o_T) = P(o_i|q_i)$

- The output depends only on the current state
  - does not depend on previous states or outputs.

Example (Jason Eisner, 2002)

- A climatologist in the future studies the history of global warming, but has no records for the weather in Baltimore. She finds Jason's diary which lists how many ice creams he had each day.
  - Use the observations in the diary (# of ice creams) to estimate what we're interested in (the weather : hot vs. cold)

HMM Problems

Problem 1 (Likelihood):	Given an HMM $\lambda = (A, B)$ and an observation sequence O, determine the likelihood $P(O\|\lambda)$.
Problem 2 (Decoding):	Given an observation sequence O and an HMM $\lambda = (A, B)$, discover the best hidden state sequence Q.
Problem 3 (Learning):	Given an observation sequence O and the set of states in the HMM, learn the HMM parameters A and B.

Problem 1: Likelihood

Likelihood

- Given the ice cream HMM, what is the probability of observing the following sequence (# of ice creams eaten that day)
  - 3 1 3 ?

- But we don't know the hidden states!

- Assume we know the hidden state to be hot cold hot, what is the probability of the observed sequence?

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)$$

Conditional Likelihood

$$P(O|Q) = \prod_{i=1}^{T} P(o_i|q_i)$$

$$P(3\ 1\ 3|\text{hot hot cold}) = P(3|\text{hot}) \times P(1|\text{hot}) \times P(3|\text{cold})$$

Likelihood

- To calculate the probability of an observed sequence, we should sum up the probability over all possible state sequences (Total Probability Theorem):

$$P(O) = \sum_{Q} P(O,Q) = \sum_{Q} P(O|Q)P(Q)$$

$$P(O,Q) = P(O|Q) \times P(Q) = \prod_{i=1}^{T} P(o_i|q_i) \times \prod_{i=1}^{T} P(q_i|q_{i-1})$$

P(3 1 3, hot hot cold) = P(hot|start)*P(hot|hot)*p(cold|hot)*P(3|hot)*P(1|hot)*P(3|cold)
P(3 1 3) = P(3 1 3, hot hot cold)+P(3 1 3, cold hot cold)+P(3 1 3, cold cold cold) …+….. **(8 possibilities)**

Complexity

- For an HMM with N hidden states, and observation of length K, there are N^K possible hidden sequences.
  - For real tasks, this is very large.

- An efficient algorithm called the **forward algorithm** uses dynamic programming to calculate the likelihood in $O(N^2T)$
  - Similar to minimum edit distance

The Forward Algorithm

- Build a table (aka. **trellis**) of forward probabilities, α_t (j), which is the probability in state j after seeing the first t observations, given the automaton λ.

$$\alpha_t(j) = P(o_1, o_2 \ldots o_t, q_t = j|\lambda)$$

- This is calculated by summing up all the paths that lead to the current cell.

The Forward Algorithm

- Forward path probability:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

$\alpha_{t-1}(i)$	the **previous forward path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

The Forward Algorithm

- Back to the ice cream example: 3 1 3

The Forward Algorithm – Cont.

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

The Forward Algorithm

1. Initialization:

$$\alpha_1(j) \;=\; \pi_j b_j(o_1) \;\; 1 \leq j \leq N$$

2. Recursion:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \leq j \leq N, 1 < t \leq T$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

The Forward Algorithm

function FORWARD(*observations* of len T, *state-graph* of len N) **returns** *forward-prob*

create a probability matrix *forward[N,T]*
for each state s **from** 1 **to** N **do** ; initialization step
$forward[s,1] \leftarrow \pi_s * b_s(o_1)$
for each time step t **from** 2 **to** T **do** ; recursion step
for each state s **from** 1 **to** N **do**

$$forward[s,t] \leftarrow \sum_{s'=1}^{N} forward[s',t-1] * a_{s',s} * b_s(o_t)$$

$$forwardprob \leftarrow \sum_{s=1}^{N} forward[s,T]$$; termination step

return *forwardprob*

Problem 2: Decoding

Decoding

- Given an observation sequence, find the most likely hidden state sequence that generated it.
  - What is the mostly likely weather pattern that generated the ice cream observation 3 1 3 ?
    - For each possible state sequence, calculate the likelihood of observing 3 1 3
    - Find the maximum over all state sequences.
    - But there are too many possible state sequences …

The Viterbi Algorithm

- A dynamic programming algorithm for efficient decoding
  - Similar to the forward algorithm, but calculates a **max** instead of a sum and keeps **backpointers**.
  - Each cell $v_t(j)$ represents the probability of being in state *j* after seeing the first *t* observations, passing through the most probable state sequence $q_1,\ldots,q_{t-1}$

$$v_t(j) = \max_{q_1,\ldots,q_{t-1}} P(q_1 \ldots q_{t-1}, o_1, o_2 \ldots o_t, q_t = j|\lambda)$$

The Viterbi Algorithm

- Viterbi path probability:

$$v_t(j) = \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t)$$

$v_{t-1}(i)$	the **previous Viterbi path probability** from the previous time step
a_{ij}	the **transition probability** from previous state q_i to current state q_j
$b_j(o_t)$	the **state observation likelihood** of the observation symbol o_t given the current state j

The Viterbi Algorithm

- Back to the ice cream example: what is the most likely weather sequence that produced the observation 3 1 3 ?

The Viterbi Algorithm

1. **Initialization:**

$$\begin{aligned} v_1(j) &= \pi_j b_j(o_1) \qquad 1 \le j \le N \\ bt_1(j) &= 0 \qquad\qquad\quad 1 \le j \le N \end{aligned}$$

2. **Recursion**

$$\begin{aligned} v_t(j) &= \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T \\ bt_t(j) &= \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T \end{aligned}$$

3. **Termination:**

$$\text{The best score:} \quad P* = \max_{i=1}^{N} v_T(i)$$

$$\text{The start of backtrace:} \quad q_T* = \operatorname*{argmax}_{i=1}^{N} v_T(i)$$

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

Problem 3: Learning

Supervised HMM POS Tagger

- Estimate parameters from labeled training data : Maximum likelihood estimation
  - Estimate tag transition probabilities $P(t_i|t_{i-1}) = \frac{C(t_{i-1}, t_i)}{C(t_{i-1})}$
  - Estimate emission probabilities for each tag $P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$
  - Apply smoothing where applicable

Example

$$P(VB|MD) = \frac{C(MD,VB)}{C(MD)} = \frac{10471}{13124} = .80$$

$$P(will|MD) = \frac{C(MD,will)}{C(MD)} = \frac{4046}{13124} = .31$$

Example

Janet will back the bill → NNP MD VB DT NN

Example

- Transition probabilities calculated from the WSJ

	NNP	**MD**	**VB**	**JJ**	**NN**	**RB**	**DT**
$<s>$	0.2767	0.0006	0.0031	0.0453	0.0449	0.0510	0.2026
NNP	0.3777	0.0110	0.0009	0.0084	0.0584	0.0090	0.0025
MD	0.0008	0.0002	0.7968	0.0005	0.0008	0.1698	0.0041
VB	0.0322	0.0005	0.0050	0.0837	0.0615	0.0514	0.2231
JJ	0.0366	0.0004	0.0001	0.0733	0.4509	0.0036	0.0036
NN	0.0096	0.0176	0.0014	0.0086	0.1216	0.0177	0.0068
RB	0.0068	0.0102	0.1011	0.1012	0.0120	0.0728	0.0479
DT	0.1147	0.0021	0.0002	0.2157	0.4744	0.0102	0.0017

Example

- Emission probabilities calculated from the WSJ

	Janet	will	back	the	bill
NNP	0.000032	0	0	0.000048	0
MD	0	0.308431	0	0	0
VB	0	0.000028	0.000672	0	0.000028
JJ	0	0	0.000340	0	0
NN	0	0.000200	0.000223	0	0.002337
RB	0	0	0.010446	0	0
DT	0	0	0	0.506099	0

q7 DT
q6 RB
q5 NN
q4 JJ
q3 VB
q2 MD
q1 NNP
q0 π
v1(7)
v1(6)
v1(5)
v1(4)= .045*0=0
v1(3)= .0031 x 0 = 0
v1(2)= .0006 x 0 = 0
v1(1) = .28* .000032 = .000009
v2(7)
v2(6)
v2(5)= max * .0002 = .0000000001
v2(4)
v2(3)= max * .000028 = 2.5e-11
v2(2) = max * .308 = .0000002772
v2(1)
v3(6)= max * .0104
v3(5)= max * . 000223
v3(4)= max * .00034
v3(3)= max * .00067
P(JJ |start) = .045
P(VB|start) = .0031
P(MD|start) = .0006
P(NNP|start) = .28
* P(MD|JJ) = 0
* P(MD|VB) = 0
* P(MD|MD) = 0
* P(MD|NNP) = .000009*.01 = .0000009
* P(RB|NN)
* P(NN|NN)
backtrace
backtrace
backtrace
start
start
start
start
start
Janet
will
back
the
bill
o1
o2
o3
o4
o5

DT
RB
NN
JJ
VB
MD
NNP
Janet
will
back
the
bill

Extending HMMs to Trigrams

- More beneficial to include more history
- Modify Markov assumption for tag transitions:

$$P(t_1^n) \approx \prod_{i=1}^{n} P(t_i | t_{i-1}, t_{i-2})$$

- Requires significant change to Viterbi decoding and increases complexity
  - Small boost in performance compared with bigrams.

Beam Search

- Viterbi decoding can be very slow for trigram taggers.

- Beam search: keep the best few hypotheses (state sequences) instead of keep the entire column.
  - Beam width b is the number of hypotheses kept
  - The values are sorted to identify the max b values in each time step, then pruned.

Example

- Beam width =2

Unsupervised HMM Training

- Input → an unlabeled set of observation sequences O, and a set of possible HMM states

- Learn the HMM parameters A and B

Example:

- O = 1 3 1 (# of ice creams)
- Vocabulary of hidden states → {Hot, Cold}

We don't know the hidden states that correspond to 1 3 1 and we don't know have labeled dataset that allows us to compute MLE

Forward-Backward Algorithm

- Learning A and B parameters → using the the **forward-Backward** algorithm (aka. **Baum-Welch** algorithm)
  - A special case of Expectation Maximization (EM)
- Start with an estimate of transition and emission probabilities
- Use the estimates to derive better probabilities
  - Compute the forward probability for an observed sequence, then divide the probability mass among all the paths that contributed to the forward probability
- Repeat until convergence

Backward Probabilities

- The **backward probability** is defined as the probability of seeing the observation from t+1 to the end, given that we are in state i at time t:

$$\beta_t(i) = P(o_{t+1}, o_{t+2} \ldots o_T | q_t = i, \lambda)$$

- Computed inductively, similar to the forward probability:

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j)$$

Backward Probabilities - Visualization

Forward vs. Backward Probabilities

Forward Probabilities

1. Initialization:

$$\alpha_1(j) \;=\; \pi_j b_j(o_1) \;\; 1 \le j \le N$$

2. Recursion:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Backward Probabilities

1. **Initialization:**

$$\beta_T(i) \;=\; 1, \quad 1 \le i \le N$$

2. **Recursion**

$$\beta_t(i) = \sum_{j=1}^{N} a_{ij}\, b_j(o_{t+1})\, \beta_{t+1}(j), \quad 1 \le i \le N, 1 \le t < T$$

3. **Termination:**

$$P(O|\lambda) = \sum_{j=1}^{N} \pi_j\, b_j(o_1)\, \beta_1(j)$$

Estimating Transition Probabilities

- A variant of maximum likelihood estimation (expectations instead of actual counts) :

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

- Define ξ_t as the probability of being in state i at time t and state j at time $t + 1$, given the observation sequence:

$$\xi_t(i,j) = P(q_t = i, q_{t+1} = j | O, \lambda)$$

- Use the conditional probability law:

$$\xi_t(i,j) = \frac{P(q_t = i, q_{t+1} = j, O|\lambda)}{P(O|\lambda)}$$

Estimating Transition Probabilities

$$\xi_t(i,j) = \frac{P(q_t = i, q_{t+1} = j, O|\lambda)}{P(O|\lambda)}$$

$$P(q_t = i, q_{t+1} = j, O|\lambda) = \alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$ **and** $$P(O|\lambda) = \sum_{j=1}^{N} \alpha_t(j)\beta_t(j)$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

Joint Probability Computation

$$P(q_t = i, q_{t+1} = j, O|\lambda) = \alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)$$

Estimating Transition Probabilities

$$\hat{a}_{ij} = \frac{\text{expected number of transitions from state } i \text{ to state } j}{\text{expected number of transitions from state } i}$$

- The expected number of transitions from state i to state j is the sum over t for ξ_t

- The expected number of transitions from state i is the sum of all transitions out of i

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

Estimating Output Probabilities

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

- We'll need the probability of being in state j at time t, given the observed sequence O:

$$\gamma_t(j) = P(q_t = j | O, \lambda) = \frac{P(q_t = j, O | \lambda)}{P(O|\lambda)}$$

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{P(O|\lambda)}$$

Joint Probability Computation

$$P(q_t = j, O|\lambda) = \alpha_t(j)\beta_t(j)$$

Estimating Output Probabilities

$$\hat{b}_j(v_k) = \frac{\text{expected number of times in state } j \text{ and observing symbol } v_k}{\text{expected number of times in state } j}$$

- The expected number of times in state j and observing v_k is the sum $\gamma_t(j)$ over all time steps t in which the observation o_t is v_k
- The expected number of times in state j is the sum of $\gamma_t(j)$ over all time steps

$$\hat{b}_j(v_k) = \frac{\sum_{t=1\ s.t. O_t = v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

Forward-Backward Algorithm

- Now we have ways to re-estimate transition and output probabilities from previous estimates

- Forward-Backward Algorithm:
  - Initialize A and B
  - The **E-step** (expectation): Compute expected state occupancy counts γ and the expected state transition counts ξ from previous A and B estimates.
  - The **M-step** (maximization): Use γ and ξ to re-compute the model parameters A and B.

function FORWARD-BACKWARD(*observations* of len T, *output vocabulary* V, *hidden state set* Q) **returns** *HMM=(A,B)*

initialize A and B
iterate until convergence
E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

Forward-Backward - Example

	$\alpha_t(j)$ for observation 1		
H	1/4	1/18	1/54
C	1/12	1/18	1/54
	o_1	o_2	o_3

	$\beta_t(j)$ for observation 1		
H	1/9	1/3	1
C	1/9	1/3	1
	o_1	o_2	o_3

Observation: 3 2 2

Calculate the updated HMM transition probabilities when i=hot and j=cold.

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{j=1}^{N} \alpha_t(j)\beta_t(j)}$$

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1} \sum_{k=1}^{N} \xi_t(i,k)}$$

POS Tagging – Summary

- Coarse or fine grained tagsets
- Rule based or learning based approaches
- Statistical based models perform better in pos tagging
- Recently neural based models are used
- Accuracy are typically used for evaluation against human annotated data

II : Text Classification

Is this spam?

Subject: **Important notice!**
From: Stanford University <newsforum@stanford.edu>
Date: October 28, 2011 12:34:16 PM PDT
To: undisclosed-recipients:;

Greats News!

You can now access the latest news by using the link below to login to Stanford University News Forum.

http://www.123contactform.com/contact-form-StanfordNew1-236335.html

Click on the above link to login for more information about this new exciting forum. You can also copy the above link to your browser bar and login for more information
about the new services.

© Stanford University. All Rights Reserved.

Positive or negative movie review?

- unbelievably disappointing
- Full of zany characters and richly applied satire, and some great plot twists
- this is the greatest screwball comedy ever filmed
- It was pathetic. The worst part about it was the boxing scenes.

Classification

- Classification *"non-sequential"* tasks take a single observation, extract some features, then classify the observation into one of a set of discrete classes

- **Probabilistic classifiers** calculate a probability distribution over the set of classes, then pick the class with the highest probability given the observation.
  - The probabilities can also be used as features when combining multiple classifiers.

Text Classification: definition

- *Input*:
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$

- *Output*: a predicted class $c \in C$

Text Classification

- We can re-define almost any problem as a classification problem!
  - Language modeling: predict the next word

    Classes: all the words in the vocabulary **Input:** past context
  - Segmentation: identify word boundaries

    Classes: word-boundary, not-a-word-boundary **Input**: characters in the context.
  - End-of-Sentence period identification
  - Parts-of-speech tagging (?)

    Classes: All possible label sequences **Input**: a sequence of words (sentence)

Supervised Machine Learning

- *Input:*
  - a document d
  - a fixed set of classes $C = \{c_1, c_2, \ldots, c_J\}$
  - A training set of m hand-labeled documents $(d_1,c_1),\ldots,(d_m,c_m)$

- *Output:*
  - a learned classifier $\gamma:d \rightarrow c$

Supervised Machine Learning

- Any kind of classifier:
  - Naïve Bayes
  - Logistic regression
  - Support-vector machines
  - k-Nearest Neighbors
  - …

Feature Representation

- In order to use these machine learning methods, we need to convert text to a suitable representation
  - Feature extraction.

- For each observation (e.g. a sentence), we need to extract a vector of features $[x_1, x_2, \ldots, x_n]$

- A classification function to compute the label y

The Bag-of-Words Representation

I love this movie! It's sweet, but with satirical humor. The dialogue is great and the adventure scenes are fun... It manages to be whimsical and romantic while laughing at the conventions of the fairy tale genre. I would recommend it to just about anyone. I've seen it several times, and I'm always happy to see it again whenever I have a friend who hasn't seen it yet!

it	6
I	5
the	4
to	3
and	3
seen	2
yet	1
would	1
whimsical	1
times	1
sweet	1
satirical	1
adventure	1
genre	1
fairy	1
humor	1
have	1
great	1
…	…

BOW Representation

- A fixed vocabulary representation, where each word is represented as an element in a high-dimensional vector
- Can be either a count or binary
- Other features: bag-of-bigrams

BOW Representation

- Example “I love this movie”

the	a	I	you	to	this	that	……	love	hate	walk	…	movie	book	…….]
[0	0	1	0	0	1	0	……	1	0	0	……	1	0	…….]

Task-Specific Features

- Task: EOS period classification:

$$x_1 = \begin{cases} 1 & \text{if } \text{“}Case(w_i) = \text{Lower”} \\ 0 & \text{otherwise} \end{cases}$$

$$x_2 = \begin{cases} 1 & \text{if } \text{“}w_i \in \text{AcronymDict”} \\ 0 & \text{otherwise} \end{cases}$$

$$x_3 = \begin{cases} 1 & \text{if } \text{“}w_i = \text{St. } \& \ Case(w_{i-1}) = \text{Cap”} \\ 0 & \text{otherwise} \end{cases}$$

Task-Specific Features

- Task: sentiment classification

$$
\begin{array}{ll}
x_1 & \text{count(positive lexicon)} \in \text{doc}) \\
x_2 & \text{count(negative lexicon)} \in \text{doc}) \\
x_3 & \begin{cases} 1 & \text{if } \text{“no”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases} \\
x_4 & \text{count}(\text{1st and 2nd pronouns} \in \text{doc}) \\
x_5 & \begin{cases} 1 & \text{if } \text{“!”} \in \text{doc} \\ 0 & \text{otherwise} \end{cases} \\
x_6 & \log(\text{word count of doc})
\end{array}
$$

Sentiment Lexicon

- A **sentiment lexicon** is a list of words annotated with positive and negative sentiment

- We might not have sufficient training data to estimate positive and negative sentiments of words

- We can add a feature that counts the number of positive words from the lexicon that occur in a document, and another feature that counts the number of negative words

Negation

- Negation is very important in sentiment identification. It often flips the sentiment of words (not great, didn’t like, don’t dismiss)

- Simple baseline: Add NOT_ to every word between negation and following punctuation:

```
didn’t like this movie , but I
```

didn’t **NOT_like NOT_this NOT_movie** but I

NAÏVE BAYES CLASSIFICATION

Naïve Bayes Intuition

- Simple ("naïve") classification method based on Bayes rule
  - (Bayes' Rule) For a document *d* and a class *c*

$$P(c \mid d) = \frac{P(d \mid c)P(c)}{P(d)}$$

- Relies on very simple representation of document
  - Bag of words
  - Assumes words in a document are independent

Naïve Bayes Classifier

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} P(c|d)$$

$$= \underset{c \in C}{\operatorname{argmax}} \frac{P(d \mid c)P(c)}{P(d)}$$

Bayes Rule

$$= \underset{c \in C}{\operatorname{argmax}} P(d \mid c)P(c)$$

Dropping the denominator

Naïve Bayes Classifier

$$\hat{c} = \underset{c \in C}{\operatorname{argmax}} \overbrace{P(d|c)}^{\text{likelihood}} \overbrace{P(c)}^{\text{prior}}$$

$$= \underset{c \in C}{\operatorname{argmax}} P(x_1, x_2, \ldots, x_n \mid c) P(c)$$

Document d represented as features $x_1..x_n$

Multinomial Naïve Bayes Assumptions

- **Bag of Words assumption**: Assume position doesn't matter

$$P(x_1, x_2, \ldots, x_n \mid c)$$

- **Conditional Independence**: Assume the feature probabilities $P(x_i|c_j)$ are independent given the class *c.*

$$P(x_1, \ldots, x_n \mid c) = P(x_1 \mid c) \cdot P(x_2 \mid c) \cdot P(x_3 \mid c) \cdot \ldots \cdot P(x_n \mid c)$$

Training Multinomial Naïve Bayes Models

- First attempt: maximum likelihood estimates
  - simply use the frequencies in the data

$$\hat{P}(c_j) = \frac{doccount(C = c_j)}{N_{doc}}$$

← # documents that belong to class c_j

← Total number of documets

$$\hat{P}(w_i \mid c_j) = \frac{count(w_i, c_j)}{\sum_{w \in V} count(w, c_j)}$$

← Count of word w_i in all documents classified as c_j

Problem with Maximum Likelihood

- What if we have seen no training documents with the word ***fantastic*** and classified in the topic **positive**?

$$\hat{P}(\text{"fantastic"} \mid \text{positive}) = \frac{count(\text{"fantastic"}, \text{positive})}{\sum_{w \in V} count(w, \text{positive})} = 0$$

- Zero probabilities cannot be conditioned away, no matter the other evidence!

$$\hat{c} = \text{argmax}_c \, \hat{P}(c) \prod_i \hat{P}(x_i \mid c)$$

Laplace (add-1) smoothing

$$\hat{P}(w_i \mid c) = \frac{count(w_i, c) + 1}{\sum_{w \in V} \left(count(w, c) + 1 \right)}$$

$$= \frac{count(w_i, c) + 1}{\left(\sum_{w \in V} count(w, c) \right) + |V|}$$

Multinomial Naive Bayes for Text Classification

positions ← all word positions in test document

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(x_i \mid c_j)$$

To avoid underflow, use the log:

$$c_{NB} = \underset{c \in C}{\operatorname{argmax}} \log P(c) + \sum_{i \in positions} \log P(w_i | c)$$

Handling Unknown Words

- Simply drop words that are not in the training vocabulary
  - Unlike language models, the position of the word does not matter. So there is no need to use the placeholder <UNK>

Example

- Sentiment classification

	Cat	Documents
Training	-	just plain boring
	-	entirely predictable and lacks energy
	-	no surprises and very few laughs
	+	very powerful
	+	the most fun film of the summer
Test	?	predictable with no fun

Example

- Calculate prior probabilities

$$P(-) = \frac{3}{5} \qquad P(+) = \frac{2}{5}$$

- Calculate likelihood for each word (per class)

$$P(\text{“predictable”}|-) = \frac{1+1}{14+20} \qquad P(\text{“predictable”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“no”}|-) = \frac{1+1}{14+20} \qquad P(\text{“no”}|+) = \frac{0+1}{9+20}$$

$$P(\text{“fun”}|-) = \frac{0+1}{14+20} \qquad P(\text{“fun”}|+) = \frac{1+1}{9+20}$$

Example

- "Predictable with no fun"

$$P(-)P(S|-) = \frac{3}{5} \times \frac{2 \times 2 \times 1}{34^3} \boxed{= 6.1 \times 10^{-5}}$$

$$P(+)P(S|+) = \frac{2}{5} \times \frac{1 \times 1 \times 2}{29^3} = 3.2 \times 10^{-5}$$

Summary: Naïve Bayes

- Very Fast, low storage requirements
- Robust to Irrelevant Features
  - Irrelevant Features cancel each other without affecting results
- A good dependable baseline for text classification
  - **But we will see other classifiers that give better accuracy**