Language Modelling: Advanced Smoothing Models

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 1

Advanced smoothing algorithms

Some Examples

- Good-Turing
- Kneser-Ney

Advanced smoothing algorithms

Some Examples

- Good-Turing
- Kneser-Ney

Good-Turing: Basic Intuition

Use the count of things we have see once

- to help estimate the count of things we have never seen

N_c: *Frequency of frequency c*

Example Sentences

<s>I am here </s>

<s>who am I </s>

<s>I would like </s>

N_c: Frequency of frequency c

Example Sentences

<s>I am here </s>

<s>who am I </s>

<s>I would like </s>

Computing N_c

I	3
am	2
here	1
who	1
would	1
like	1

$N_1 = 4$

$N_2 = 1$

$N_3 = 1$

Good Turing Estimation

Idea

- Reallocate the probability mass of $n-$grams that occur $r+1$ times in the training data to the $n-$grams that occur r times
- In particular, reallocate the probability mass of $n-$grams that were seen once to the $n-$grams that were never seen

Adjusted count

For each count c, an adjusted count c^* is computed as:

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

where N_c is the number of $n-$grams seen exactly c times

Good Turing Estimation

Good Turing Smoothing

P^*_{GT}(things with frequency c) = $\frac{c^*}{N}$

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

Good Turing Estimation

Good Turing Smoothing

P^*_{GT}(things with frequency c) = $\frac{c^*}{N}$

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

What if $c = 0$

P^*_{GT}(things with frequency c) = $\frac{N_1}{N}$ where N denotes the total number of bigrams that actually occur in training

Complications

What about words with high frequency?

- For small c, $N_c > N_{c+1}$
- For large c, too jumpy

Complications

What about words with high frequency?

- For small c, $N_c > N_{c+1}$
- For large c, too jumpy

Simple Good-Turing

Replace empirical N_k with a best-fit power law once counts get unreliable

Good-Turing numbers: Example

22 million words of AP Neswire

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

Count c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Good-Turing numbers: Example

22 million words of AP Neswire

$$c^* = \frac{(c+1)N_{c+1}}{N_c}$$

It looks like $c^* = c - 0.75$

Count c	Good Turing c*
0	.0000270
1	0.446
2	1.26
3	2.24
4	3.24
5	4.22
6	5.19
7	6.21
8	7.24
9	8.25

Absolute Discounting Interpolation

Why don't we just substract 0.75 (or some d)?

Absolute Discounting Interpolation

Why don't we just substract 0.75 (or some d)?

$$P_{AbsoluteDiscounting}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w_i)$$

Absolute Discounting Interpolation

Why don't we just substract 0.75 (or some d)?

$$P_{AbsoluteDiscounting}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w_i)$$

We may keep some more values of d for counts 1 and 2

Absolute Discounting Interpolation

Why don't we just substract 0.75 (or some d)?

$$P_{AbsoluteDiscounting}(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i) - d}{c(w_{i-1})} + \lambda(w_{i-1})P(w_i)$$

We may keep some more values of d for counts 1 and 2

But can we do better than using the regular unigram correct?

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?
- "Francisco" more common that "glasses"

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?
- "Francisco" more common that "glasses"
- *But "Francisco" mostly follows "San"*

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?
- "Francisco" more common that "glasses"
- *But "Francisco" mostly follows "San"*

$P(w)$: *"How likely is w?"*

Instead, $P_{continuation}(w)$: "How likely is w to appear as a novel continuation?"

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?
- "Francisco" more common that "glasses"
- *But "Francisco" mostly follows "San"*

$P(w)$: *"How likely is w?"*

Instead, $P_{continuation}(w)$: "How likely is w to appear as a novel continuation?"

- For each word, count the number of bigram types it completes
- Every bigram type was a novel continuation the first time it was seen

Kneser-Ney Smoothing

Intuition

- Shannon game: *I can't see without my reading ...*: glasses/Francisco?
- "Francisco" more common that "glasses"
- *But "Francisco" mostly follows "San"*

$P(w)$: "How likely is w?"

Instead, $P_{continuation}(w)$: "How likely is w to appear as a novel continuation?"

- For each word, count the number of bigram types it completes
- Every bigram type was a novel continuation the first time it was seen

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Normalized by the total number of word bigram types

$$|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|$$

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Normalized by the total number of word bigram types

$$|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|$$

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

Kneser-Ney Smoothing

How many times does w appear as a novel continuation?

$$P_{continuation}(w) \propto |\{w_{i-1} : c(w_{i-1}, w) > 0\}|$$

Normalized by the total number of word bigram types

$$|\{(w_{j-1}, w_j) : c(w_{j-1}, w_j) > 0\}|$$

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

A frequent word (Francisco) occurring in only one context (San) will have a low continuation probability

Kneser-Ney Smoothing

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

Kneser-Ney Smoothing

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

λ is a normalizing constant

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

Model Combination

As N increases

- The power (expressiveness) of an N-gram model increases

Model Combination

As N increases

- The power (expressiveness) of an N-gram model increases
- But the ability to estimate accurate parameters from sparse data decreases (i.e. the smoothing problem gets worse).

Model Combination

As N increases

- The power (expressiveness) of an N-gram model increases
- But the ability to estimate accurate parameters from sparse data decreases (i.e. the smoothing problem gets worse).

A general approach is to combine the results of multiple N-gram models.

Backoff and Interpolation

It might help to use less context

Backoff and Interpolation

It might help to use less context

when you haven't learned much about larger contexts

Backoff and Interpolation

It might help to use less context

when you haven't learned much about larger contexts

Backoff

- use trigram if you have good evidence
- otherwise bigram, otherwise unigram

Backoff and Interpolation

It might help to use less context

when you haven't learned much about larger contexts

Backoff

- use trigram if you have good evidence
- otherwise bigram, otherwise unigram

Interpolation

mix unigram, bigram, trigram

Backoff

Estimating $P(w_i|w_{i-2}w_{i-1})$

- If we do not have counts to compute $P(w_i|w_{i-2}w_{i-1})$ estimate this using the bigram probbaility $P(w_i|w_{i-1})$

Backoff

Estimating $P(w_i|w_{i-2}w_{i-1})$

- If we do not have counts to compute $P(w_i|w_{i-2}w_{i-1})$ estimate this using the bigram probbaility $P(w_i|w_{i-1})$
- If we do not have counts to compute $P(w_i|w_{i-1})$, estimate this using the unigram probability $P(w_i)$

Backoff

Estimating $P(w_i|w_{i-2}w_{i-1})$

- If we do not have counts to compute $P(w_i|w_{i-2}w_{i-1})$ estimate this using the bigram probbaility $P(w_i|w_{i-1})$
- If we do not have counts to compute $P(w_i|w_{i-1})$, estimate this using the unigram probability $P(w_i)$

$P_{bo}(w_i|w_{i-2}w_{i-1})$=

- $\hat{P}(w_i|w_{i-2}w_{i-1})$, if $c(w_{i-2}w_{i-1}w_i) > 0$
- $\lambda(w_{i-1}w_{i-2})P_{bo}(w_i|w_{i-1})$, otherwise

where $P_{bo}(w_i|w_{i-1})$ =

- $\hat{P}(w_i|w_{i-1})$ if $c(w_{i-1}w_i) > 0$
- $\lambda(w_{n-1})\hat{P}(w_n)$, otherwise

Example Problem

In a corpus, suppose there are 4 words, a, b, c, and d. You are provided with the following counts.

n-gram	count	n-gram	count	n-gram	count
aba	4	ba	5	a	8
abb	0	bb	3	b	9
abc	0	bc	0	c	8
abd	0	bd	0	d	7

Use the recursive definition of backoff smoothing to obtain the probability distribution, $P_{backoff}(w_n|w_{n-2}w_{n-1})$, where $w_{n-1} = b$ and $w_{n-2} = a$.
Also assume that $\hat{P}(x) = P(x) - 1/8$.

Linear Interpolation

Simple Interpolation

$$\tilde{P}(w_n|w_{n-1}w_{n-2}) = \lambda_1 P(w_n|w_{n-1}w_{n-2}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

Linear Interpolation

Simple Interpolation

$$\tilde{P}(w_n|w_{n-1}w_{n-2}) = \lambda_1 P(w_n|w_{n-1}w_{n-2}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

$$\sum_i \lambda_i = 1$$

Linear Interpolation

Simple Interpolation

$$\tilde{P}(w_n|w_{n-1}w_{n-2}) = \lambda_1 P(w_n|w_{n-1}w_{n-2}) + \lambda_2 P(w_n|w_{n-1}) + \lambda_3 P(w_n)$$

$$\sum_i \lambda_i = 1$$

Lambdas conditional on context

$$\tilde{P}(w_n|w_{n-1}w_{n-2}) = \lambda_1(w_{n-2}, w_{n-1}) P(w_n|w_{n-1}w_{n-2})$$
$$+ \lambda_2(w_{n-2}, w_{n-1}) P(w_n|w_{n-1}) + \lambda_3(w_{n-2}, w_{n-1}) P(w_n)$$

Setting the lambda values

Use a held-out corpus

Choose λs to maximize the probability of held-out data:

- Find the N-gram probabilities on the training data
- Search for λs that give the largest probability to held-out data

Computational Morphology

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 2

Morphology

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

Morphology

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

dogs

- 2 morphemes, ‘dog’ and ‘s’
- ‘s’ is a plural marker on nouns

Morphology

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

dogs

- 2 morphemes, ‘dog’ and ‘s’
- ‘s’ is a plural marker on nouns

unladylike

3 morphemes

- un- ‘not’
- lady ‘well-behaved woman’
- -like ‘having the characteristic of’

Allomorphs

Variants of the same morpheme, but cannot be replaced by one another

Example

- opposite: un-happy, in-comprehensible, im-possible, ir-rational

Bound and Free Morphemes

Bound

Cannot appear as a word by itself.
-s (dog-s), -ly (quick-ly), -ed (walk-ed)

Bound and Free Morphemes

Bound

Cannot appear as a word by itself.

-s (dog-s), -ly (quick-ly), -ed (walk-ed)

Free

Can appear as a word by itself; often can combine with other morphemes too.

house (house-s), walk (walk-ed), of, the, or

Stems and Affixes

Stems and Affixes

- Stems (roots): The core meaning bearing units
- Affixes: Bits and pieces adhering to stems to change their meanings and grammatical functions

Stems and Affixes

- Stems (roots): The core meaning bearing units
- Affixes: Bits and pieces adhering to stems to change their meanings and grammatical functions

Mostly, stems are free morphemes and affixes are bound morphemes

Types of affixes

Types of affixes

- Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
  un-happy, pre-existing

Types of affixes

- Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
  un-happy, pre-existing
- Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)
  talk-ing, quick-ly

Types of affixes

- Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
  un-happy, pre-existing
- Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)
  talk-ing, quick-ly
- Infix: '*n*' in '*vindati*' (he knows), as contrasted with *vid* (to know).
  Philippines: basa 'read' → *b-um-asa 'read'*
  English:

Types of affixes

- Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
  un-happy, pre-existing
- Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)
  talk-ing, quick-ly
- Infix: '*n*' in '*vindati*' (he knows), as contrasted with *vid* (to know).
  Philippines: basa 'read' → *b-um-asa 'read'*
  English: abso-bloody-lutely (emphasis)

Types of affixes

- Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
  un-happy, pre-existing
- Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)
  talk-ing, quick-ly
- Infix: '*n*' in '*vindati*' (he knows), as contrasted with *vid* (to know).
  Philippines: basa 'read' → *b-um-asa 'read'*
  English: abso-bloody-lutely (emphasis)
- Circumfixes - precedes and follows the stem
  Dutch: berg 'mountain', ge-berg-te 'mountains'

Content and functional morphemes

Content morphemes

Carry some semantic content

car, -able, un-

Content and functional morphemes

Content morphemes

Carry some semantic content

car, -able, un-

Functional morphemes

Provide grammatical information

-s (plural), -s (3^{rd} singular)

Inflectional and Derivational Morphology

Two different kind of relationship among words

Inflectional and Derivational Morphology

Two different kind of relationship among words

Inflectional morphology

Grammatical: number, tense, case, gender

Creates new forms of the same word: *bring, brought, brings, bringing*

Inflectional and Derivational Morphology

Two different kind of relationship among words

Inflectional morphology

Grammatical: number, tense, case, gender
Creates new forms of the same word: *bring, brought, brings, bringing*

Derivational morphology

Creates new words by changing part-of-speech: *logic, logical, illogical, illogicality, logician*

Inflectional and Derivational Morphology

Two different kind of relationship among words

Inflectional morphology

Grammatical: number, tense, case, gender

Creates new forms of the same word: *bring, brought, brings, bringing*

Derivational morphology

Creates new words by changing part-of-speech: *logic, logical, illogical, illogicality, logician*

Fairly systematic but some derivations missing: *sincere - sincerity, scarce - scarcity, curious - curiosity, fierce - fiercity?*

Morphological processes

Concatenation

Adding continuous affixes - the most common process:

- hope+less, un+happy, anti+capital+ist+s

Morphological processes

Concatenation

Adding continuous affixes - the most common process:

- hope+less, un+happy, anti+capital+ist+s

Often, there are phonological/graphemic changes on morpheme boundaries:

- book + s [s], shoe + s [z]
- happy +er → happier

Morphological processes

Reduplication: part of the word or the entire word is doubled

Morphological processes

Reduplication: part of the word or the entire word is doubled

- Nama: ‘go’ (look), ‘go-go’ (examine with attention)

Morphological processes

Reduplication: part of the word or the entire word is doubled

- Nama: ‘go’ (look), ‘go-go’ (examine with attention)
- Tagalog: ‘basa’ (read), ‘ba-basa’(will read)

Reduplication: part of the word or the entire word is doubled

- Nama: ‘go’ (look), ‘go-go’ (examine with attention)
- Tagalog: ‘basa’ (read), ‘ba-basa’(will read)
- Sanskrit: ‘pac’ (cook), ‘papāca’ (perfect form, cooked)

Morphological processes

Reduplication: part of the word or the entire word is doubled

- Nama: ‘go’ (look), ‘go-go’ (examine with attention)
- Tagalog: ‘basa’ (read), ‘ba-basa’(will read)
- Sanskrit: ‘pac’ (cook), ‘papāca’ (perfect form, cooked)
- Phrasal reduplication (Telugu): *pillavāḍu naḍustū naḍustū paḍi pōyāḍu* (The child fell down while walking)

Morphological processes

Suppletion

'irregular' relation between the words

go - went, good - better

Morphological processes

Suppletion

'irregular' relation between the words

go - went, good - better

Morpheme internal changes

The word changes internally

sing - sang - sung, man - men, goose - geese

Word Formation

Compounding

Words formed by combining two or more words

Example in English:

- Adj + Adj $\rightarrow$ Adj: bitter-sweet
- N + N $\rightarrow$ N: rain-bow
- V + N $\rightarrow$ V: pick-pocket
- P + V $\rightarrow$ V: over-do

Word Formation

Compounding

Words formed by combining two or more words

Example in English:

- Adj + Adj $\rightarrow$ Adj: bitter-sweet
- N + N $\rightarrow$ N: rain-bow
- V + N $\rightarrow$ V: pick-pocket
- P + V $\rightarrow$ V: over-do

Particular to languages

room-temperature: Hindi translation?

Word Formation

Compounding

Words formed by combining two or more words

Example in English:

- Adj + Adj → Adj: bitter-sweet
- N + N → N: rain-bow
- V + N → V: pick-pocket
- P + V → V: over-do

Particular to languages

room-temperature: Hindi translation?

Word Formation

Acronyms

laser: Light Amplification by Simulated Emission of Radiation

Word Formation

Acronyms

laser: Light Amplification by Simulated Emission of Radiation

Blending

Parts of two different words are combined

- breakfast + lunch → brunch
- smoke + fog → smog
- motor + hotel → motel

Word Formation

Acronyms

laser: Light Amplification by Simulated Emission of Radiation

Blending

Parts of two different words are combined

- breakfast + lunch → brunch
- smoke + fog → smog
- motor + hotel → motel

Clipping

Longer words are shortened

Word Formation

Acronyms

laser: Light Amplification by Simulated Emission of Radiation

Blending

Parts of two different words are combined

- breakfast + lunch → brunch
- smoke + fog → smog
- motor + hotel → motel

Clipping

Longer words are shortened

doctor, laboratory, advertisement, dormitory, examination, bicycle, refrigerator

Processing morphology

- Lemmatization: word → lemma
  saw → {see, saw}

Processing morphology

- Lemmatization: word → lemma
  saw → {see, saw}
- Morphological analysis : word → setOf(lemma +tag)
  saw → { <see, verb.past>, < saw, noun.sg>}

Processing morphology

- Lemmatization: word → lemma
  saw → {see, saw}
- Morphological analysis : word → setOf(lemma +tag)
  saw → { <see, verb.past>, < saw, noun.sg>}
- Tagging: word → tag, considers context
  Peter *saw* her → { <see, verb.past>}

Processing morphology

- Lemmatization: word → lemma
  saw → {see, saw}
- Morphological analysis : word → setOf(lemma +tag)
  saw → { <see, verb.past>, < saw, noun.sg>}
- Tagging: word → tag, considers context
  Peter *saw* her → { <see, verb.past>}
- Morpheme segmentation: de-nation-al-iz-ation

Processing morphology

- Lemmatization: word → lemma
  saw → {see, saw}
- Morphological analysis : word → setOf(lemma +tag)
  saw → { <see, verb.past>, < saw, noun.sg>}
- Tagging: word → tag, considers context
  Peter *saw* her → { <see, verb.past>}
- Morpheme segmentation: de-nation-al-iz-ation
- Generation: see + verb.past → saw

What are the applications?

- Text-to-speech synthesis:
  lead:

What are the applications?

- Text-to-speech synthesis:
  lead: verb or noun?
  read:

What are the applications?

- Text-to-speech synthesis:
  lead: verb or noun?
  read: present or past?
- Search and information retrieval
- Machine translation, grammar correction

Morphological Analysis

Input	Morphological Parsed Output
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +PL
geese	goose +N +PL
goose	(goose +N +SG) or (goose +V)
gooses	goose +V +3SG
merging	merge +V +PRES-PART
caught	(catch +V +PAST-PART) or (catch +V +PAST)

Morphological Analysis

Input	Morphological Parsed Output
cats	cat +N +PL
cat	cat +N +SG
cities	city +N +PL
geese	goose +N +PL
goose	(goose +N +SG) or (goose +V)
gooses	goose +V +3SG
merging	merge +V +PRES-PART
caught	(catch +V +PAST-PART) or (catch +V +PAST)

Goal

To take input forms like those in the first column and produce output forms like those in the second column.
Output contains stem and additional information; +N for noun, +SG for singular, +PL for plural, +V for verb etc.

Issues involved

boy $\rightarrow$ boys

Issues involved

boy $\longrightarrow$ boys
fly $\longrightarrow$ flys $\longrightarrow$ flies (y$\longrightarrow$ i rule)

Issues involved

boy $\longrightarrow$ boys
fly $\longrightarrow$ flys $\longrightarrow$ flies (y$\longrightarrow$ i rule)

Toiling $\longrightarrow$ toil

Issues involved

boy $\longrightarrow$ boys
fly $\longrightarrow$ flys $\longrightarrow$ flies (y$\longrightarrow$ i rule)

Toiling $\longrightarrow$ toil
Duckling $\longrightarrow$ duckl?

Issues involved

boy $\longrightarrow$ boys
fly $\longrightarrow$ flys $\longrightarrow$ flies (y$\longrightarrow$ i rule)

Toiling $\longrightarrow$ toil
Duckling $\longrightarrow$ duckl?

- Getter $\longrightarrow$ get + er
- Doer $\longrightarrow$ do + er

Issues involved

boy $\longrightarrow$ boys
fly $\longrightarrow$ flys $\longrightarrow$ flies (y$\longrightarrow$ i rule)

Toiling $\longrightarrow$ toil
Duckling $\longrightarrow$ duckl?

- Getter $\longrightarrow$ get + er
- Doer $\longrightarrow$ do + er
- Beer $\longrightarrow$ be + er?

Knowledge Required

Knowledge of stems or roots

Duck is a possible root, not *duckl*.

We need a dictionary (lexicon)

Knowledge Required

Knowledge of stems or roots

Duck is a possible root, not *duckl*.

We need a dictionary (lexicon)

Morphotactics

Which class of morphemes follow other classes of morphemes inside the word?

Ex: plural morpheme follows the noun

Knowledge Required

Knowledge of stems or roots

Duck is a possible root, not *duckl*.
We need a dictionary (lexicon)

Morphotactics

Which class of morphemes follow other classes of morphemes inside the word?
Ex: plural morpheme follows the noun

Only some endings go on some words

- *Do+er*: ok
- *Be+er*: not so

Knowledge Required

Knowledge of stems or roots

Duck is a possible root, not *duckl*.
We need a dictionary (lexicon)

Morphotactics

Which class of morphemes follow other classes of morphemes inside the word?
Ex: plural morpheme follows the noun

Only some endings go on some words

- *Do+er*: ok
- *Be+er*: not so

Spelling change rules

Adjust the surface form using spelling change rules

- Get + er → getter

Why can't this be put in a big lexicon?

- English: just 317,477 forms from 90,196 lexical entries, a ratio of 3.5:1

Why can't this be put in a big lexicon?

- English: just 317,477 forms from 90,196 lexical entries, a ratio of 3.5:1
- Sanskrit: 11 million forms from a lexicon of 170,000 entries, a ratio of 64.7:1

Why can't this be put in a big lexicon?

- English: just 317,477 forms from 90,196 lexical entries, a ratio of 3.5:1
- Sanskrit: 11 million forms from a lexicon of 170,000 entries, a ratio of 64.7:1
- New forms can be created, compounding etc.

One of the most common methods is finite-state-machines

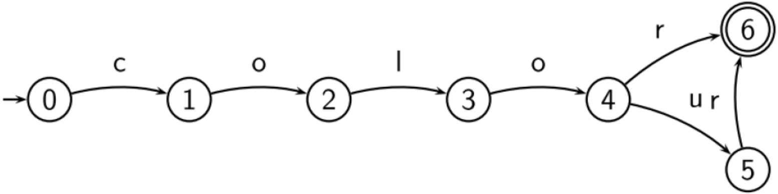
Finite-state methods for morphology

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 3

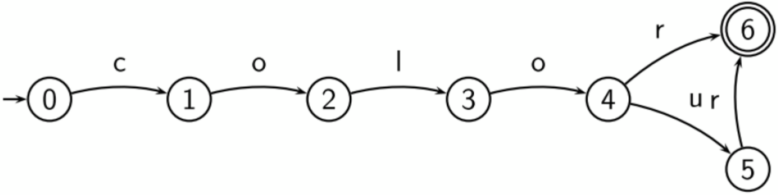
Finite State Automaton (FSA)



What is FSA?

- A kind of directed graph

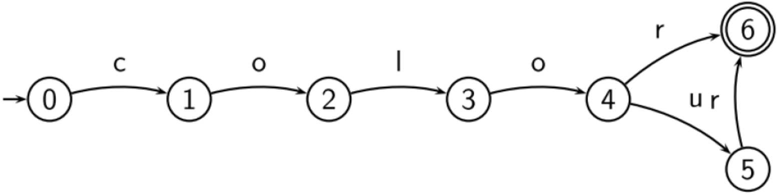
Finite State Automaton (FSA)



What is FSA?

- A kind of directed graph
- Nodes are called states, edges are labeled with symbols (possibly empty ϵ)

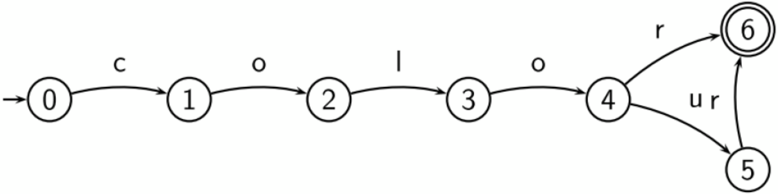
Finite State Automaton (FSA)



What is FSA?

- A kind of directed graph
- Nodes are called states, edges are labeled with symbols (possibly empty ϵ)
- Start state and accepting states

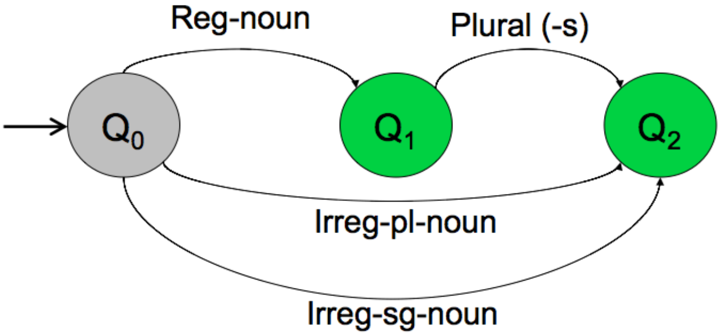
Finite State Automaton (FSA)



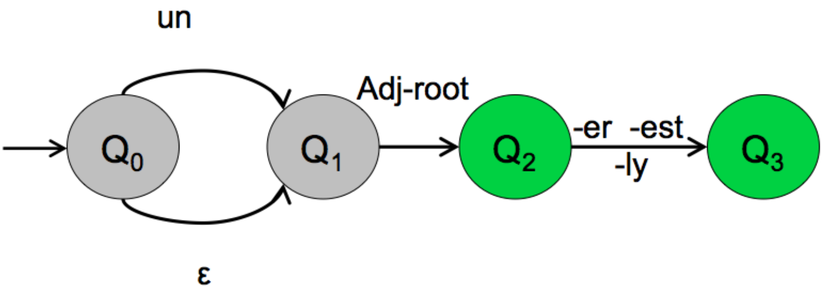
What is FSA?

- A kind of directed graph
- Nodes are called states, edges are labeled with symbols (possibly empty ϵ)
- Start state and accepting states
- Recognizes regular languages, i.e., languages specified by regular expressions

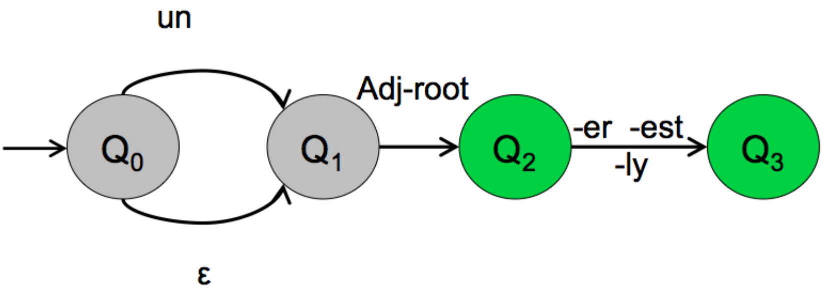
FSA for nominal inflection in English



FSA for English Adjectives



FSA for English Adjectives



Word modeled

happy, happier, happiest, real, unreal, cool, coolly, clear, clearly, unclear, unclearly, ...

Morphotactics

- The last two examples model some parts of the English morphotactics
- But what about the information about regular and irregular roots?

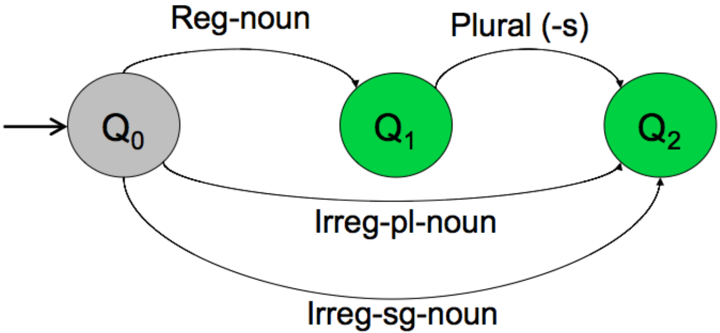
Morphotactics

- The last two examples model some parts of the English morphotactics
- But what about the information about regular and irregular roots?

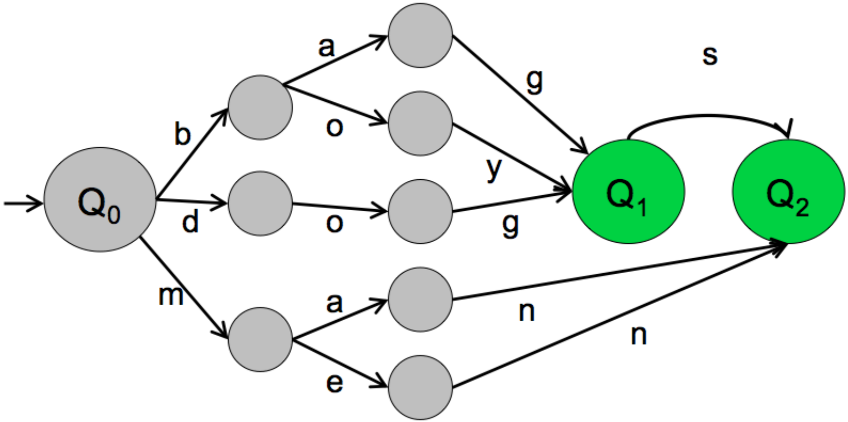
Lexicon

Can we include the lexicon in the FSA?

FSA for nominal inflection in English



After adding a mini-lexicon



Some properties of FSAs: Elegance

- Recognizing problem can be solved in linear time (independent of the size of the automaton)
- There is an algorithm to transform each automaton into a unique equivalent automaton with the least number of states
- An FSA is deterministic iff it has no empty (ϵ) transition and for each state and each symbol, there is at most one applicable transition
- Every non-deterministic automaton can be transformed into a deterministic one

But ...

FSAs are language recognizers/generators.

But ...

FSAs are language recognizers/generators.

We need transducers to build Morphological Analyzers

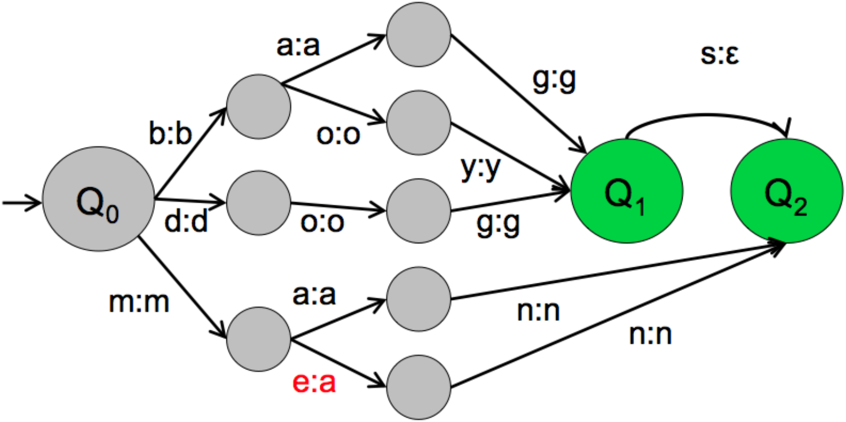
But ...

FSAs are language recognizers/generators.
We need transducers to build Morphological Analyzers

Finite State Transducers

- Translate strings from one language to strings in another language
- Like FSA, but each edge is associated with two strings

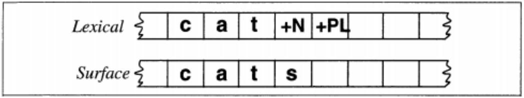
An example FST



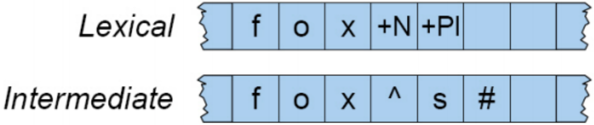
Two-level morphology

Given the input *cats*, we would like to output *cat+N+PL*, talling us that cat is a plural noun.

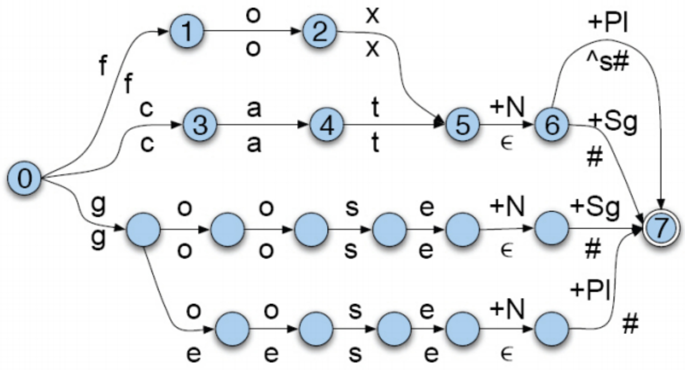
We do this via a version of **two-level morphology**, a correspondence between a lexical level (morphemes and features) to a surface level (actual spelling).



Intermediate tape for Spelling change rules

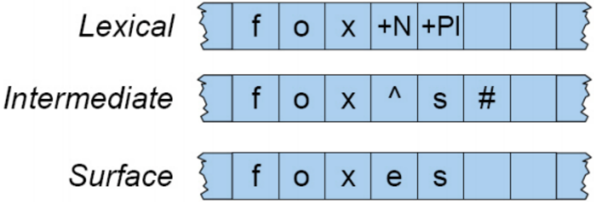


English Nominal Inflection FST



Spelling Handling

A spelling change rule would insert an e only in the appropriate environment.



Rule Handling

Rule Notation

$a \rightarrow b/c\_d$: "rewrite a as b when it occurs between c and d."

Morphological Analysis: Approaches

Two different ways to address phonological/graphemic variations

- Linguistic approach: A phonological component accompanying the simple concatenative process of attaching an ending
- Engineering approach: Phonological changes and irregularities are factored into endings and a higher number of paradigms

Different Approaches: Example from Czech

	woman	owl	draft	iceberg	vapor	fly
S1	žen-a	sov-a	skic-a	kr-a	pár-a	mouch-a
S2	žen-y	sov-y	skic-i	kr-y	pár-y	mouch-y
S3	žen-ě	sov-ě	skic-e	kř-e	pář-e	mouš-e
⋮						
P2	žen-0	sov-0	skic-0	ker-0	par-0	much-0

A linguistic approach

$$\textit{žen} + \begin{cases} \text{a} \\ \text{y} \\ \text{ě} \\ 0 \end{cases} \quad \textit{sov} + \begin{cases} \text{a} \\ \text{y} \\ \text{ě} \\ 0 \end{cases} \quad \textit{skic} + \begin{cases} \text{a} \\ \text{y} \\ \text{ě} \\ 0 \end{cases} \quad \textit{kr} + \begin{cases} \text{a} \\ \text{y} \\ \text{ě} \\ 0 \end{cases} \quad \textit{pár} + \begin{cases} \text{a} \\ \text{y} \\ \text{ě} \\ 0 \end{cases} \quad \textit{mouch} + \begin{cases} \text{a} \\ \text{y} \\ \text{ě} \\ 0 \end{cases}$$

An engineering approach

$$\textit{žen} + \begin{cases} \text{a} \\ \text{y} \\ \text{ě} \\ 0 \end{cases} \quad \textit{sov} + \begin{cases} \text{a} \\ \text{y} \\ \text{ě} \\ 0 \end{cases} \quad \textit{skic} + \begin{cases} \text{a} \\ \text{i} \\ \text{e} \\ 0 \end{cases} \quad \text{k} + \begin{cases} \text{ra} \\ \text{ry} \\ \text{ře} \\ \text{er} \end{cases} \quad \text{p} + \begin{cases} \text{ára} \\ \text{áry} \\ \text{áře} \\ \text{ar} \end{cases} \quad \text{m} + \begin{cases} \text{oucha} \\ \text{ouchy} \\ \text{ouše} \\ \text{uch} \end{cases}$$

Tools Available

- AT&T FSM Library and Lextools
  http://www2.research.att.com/~fsmtools/fsm/
- OpenFST (Google and NYU)
  http://www.openfst.org/

Introduction to POS Tagging

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 4

Part-of-Speech (POS) tagging

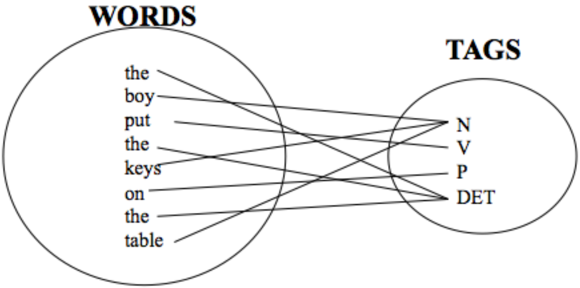
Part-of-Speech (POS) tagging

Task

Given a text of English, identify the parts of speech of each word

Task

Given a text of English, identify the parts of speech of each word



Parts of Speech: How many?

Open class words (content words)

- nouns, verbs, adjectives, adverbs
- mostly content-bearing: they refer to objects, actions, and features in the world
- *open class*, since new words are added all the time

Parts of Speech: How many?

Open class words (content words)

- nouns, verbs, adjectives, adverbs
- mostly content-bearing: they refer to objects, actions, and features in the world
- *open class*, since new words are added all the time

Closed class words

- pronouns, determiners, prepositions, connectives, ...
- there is a limited number of these
- *mostly functional:* to tie the concepts of a sentence together

POS examples

- N noun chair, bandwidth, pacing
- V verb study, debate, munch
- ADJ adj purple, tall, ridiculous
- ADV adverb unfortunately, slowly,
- P preposition of, by, to
- PRO pronoun I, me, mine
- DET determiner the, a, that, those

POS tagging: Choosing a tagset

- To do POS tagging, a standard set needs to be chosen

- To do POS tagging, a standard set needs to be chosen
- Could pick very coarse tagsets
  N, V, Adj, Adv

POS tagging: Choosing a tagset

- To do POS tagging, a standard set needs to be chosen
- Could pick very coarse tagsets
  N, V, Adj, Adv
- More commonly used set is finer grained, “UPenn TreeBank tagset”, 45 tags

POS tagging: Choosing a tagset

- To do POS tagging, a standard set needs to be chosen
- Could pick very coarse tagsets
  N, V, Adj, Adv
- More commonly used set is finer grained, "UPenn TreeBank tagset", 45 tags

A Nice Tutorial on POS tags

https://sites.google.com/site/partofspeechhelp/

UPenn TreeBank POS tag set

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

Using the UPenn tagset

Example Sentence

The grand jury commented on a number of other topics.

Using the UPenn tagset

Example Sentence

The grand jury commented on a number of other topics.

POS tagged sentence

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

Why is POS tagging hard?

Why is POS tagging hard?

Words often have more than one POS: back

- The back door:

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ*
- On my back:

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ*
- On my back: *back/NN*
- Win the voters back:

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ*
- On my back: *back/NN*
- Win the voters back: *back/RB*
- Promised to back the bill:

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ*
- On my back: *back/NN*
- Win the voters back: *back/RB*
- Promised to back the bill: *back/VB*

Why is POS tagging hard?

Words often have more than one POS: back

- The back door: *back/JJ*
- On my back: *back/NN*
- Win the voters back: *back/RB*
- Promised to back the bill: *back/VB*

POS tagging problem

To determine the POS tag for a particular instance of a word

Ambiguous word types in the Brown Corpus

Ambiguity in the Brown corpus

- 40% of word tokens are ambiguous
- 12% of word types are ambiguous

Ambiguous word types in the Brown Corpus

Ambiguity in the Brown corpus

- 40% of word tokens are ambiguous
- 12% of word types are ambiguous
- Breakdown of ambiguous word types:

Unambiguous (1 tag)	35,340
Ambiguous (2–7 tags)	4,100
2 tags	3,760
3 tags	264
4 tags	61
5 tags	12
6 tags	2
7 tags	1 ("still")

How bad is the ambiguity problem?

- One tag is usually more likely than the others.

How bad is the ambiguity problem?

- One tag is usually more likely than the others.
  In the Brown corpus, *race* is a noun 98% of the time, and a verb 2% of the time

How bad is the ambiguity problem?

- One tag is usually more likely than the others.
  In the Brown corpus, *race* is a noun 98% of the time, and a verb 2% of the time
- A tagger for English that simply chooses the most likely tag for each word can achieve good performance

How bad is the ambiguity problem?

- One tag is usually more likely than the others.
  In the Brown corpus, *race* is a noun 98% of the time, and a verb 2% of the time
- A tagger for English that simply chooses the most likely tag for each word can achieve good performance
- Any new approach should be compared against the unigram baseline (assigning each token to its most likely tag)

Deciding the correct POS

Can be difficult even for people

- Mrs./NNP Shaefer/NNP never/RB got/VBD around/_ to/TO joining/VBG.
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/_ the/DT corner/NN.
- Chateau/NNP Petrus/NNP costs/VBZ around/_ 2500/CD.

Deciding the correct POS

Can be difficult even for people

- Mrs./NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG.
- All/DT we/PRP gotta/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN.
- Chateau/NNP Petrus/NNP costs/VBZ around/RB 2500/CD.

Relevant knowledge for POS tagging

The word itself

- Some words may only be nouns, e.g. *arrow*
- Some words are ambiguous, e.g. *like, flies*
- Probabilities may help, if one tag is more likely than another

Relevant knowledge for POS tagging

The word itself

- Some words may only be nouns, e.g. *arrow*
- Some words are ambiguous, e.g. *like, flies*
- Probabilities may help, if one tag is more likely than another

Local context

- Two determiners rarely follow each other
- Two base form verbs rarely follow each other
- Determiner is almost always followed by adjective or noun

POS tagging: Two approaches

Rule-based Approach

- Assign each word in the input a list of potential POS tags
- Then winnow down this list to a single tag using hand-written rules

POS tagging: Two approaches

Rule-based Approach

- Assign each word in the input a list of potential POS tags
- Then winnow down this list to a single tag using hand-written rules

Statistical tagging

- Get a training corpus of tagged text, learn the transformation rules from the most frequent tags (TBL tagger)
- Probabilistic: Find the most likely sequence of tags T for a sequence of words W

TBL Tagger

Label the training set with most frequent tags

- The can was rusted.

TBL Tagger

Label the training set with most frequent tags

- The can was rusted.
- The/DT can/MD was/VBD rusted/VBD.

TBL Tagger

Label the training set with most frequent tags

- The can was rusted.
- The/DT can/MD was/VBD rusted/VBD.

Add transformation rules to reduce training mistakes

- MD →NN: DT_
- VBD→VBN: VBD_

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d, c)\}$ of paired observations d and hidden classes c.

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d,c)\}$ of paired observations d and hidden classes c.

Different instances of d and c

- **Part-of-Speech Tagging**:

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d,c)\}$ of paired observations d and hidden classes c.

Different instances of d and c

- **Part-of-Speech Tagging**: words are observed and tags are hidden.
- **Text Classification**:

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d,c)\}$ of paired observations d and hidden classes c.

Different instances of d and c

- **Part-of-Speech Tagging**: words are observed and tags are hidden.
- **Text Classification**: sentences/documents are observed and the category is hidden.

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d, c)\}$ of paired observations d and hidden classes c.

Different instances of d and c

- **Part-of-Speech Tagging**: words are observed and tags are hidden.
- **Text Classification**: sentences/documents are observed and the category is hidden.
  Categories can be positive/negative for sentiments ..
  sports/politics/business for documents ...

Probabilistic Tagging: Two different families of models

Problem at hand

We have some data $\{(d,c)\}$ of paired observations d and hidden classes c.

Different instances of d and c

- **Part-of-Speech Tagging**: words are observed and tags are hidden.
- **Text Classification**: sentences/documents are observed and the category is hidden.
  Categories can be positive/negative for sentiments ..
  sports/politics/business for documents ...

What gives rise to the two families?

Whether they generate the observed data from hidden stuff or the hidden structure given the data?

Generative vs. Conditional Models

Generative (Joint) Models

Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d,c)$ in terms of $P(d|c)$

Generative vs. Conditional Models

Generative (Joint) Models

Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d,c)$ in terms of $P(d|c)$
e.g. Naïve Bayes' classifiers, Hidden Markov Models etc.

Generative vs. Conditional Models

Generative (Joint) Models

Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d,c)$ in terms of $P(d|c)$
e.g. Naïve Bayes' classifiers, Hidden Markov Models etc.

Discriminative (Conditional) Models

Take the data as given, and put a probability over hidden structure given the data: $P(c|d)$

Generative vs. Conditional Models

Generative (Joint) Models

Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d,c)$ in terms of $P(d|c)$
e.g. Naïve Bayes' classifiers, Hidden Markov Models etc.

Discriminative (Conditional) Models

Take the data as given, and put a probability over hidden structure given the data: $P(c|d)$
e.g. Logistic regression, maximum entropy models, conditional random fields

Generative vs. Conditional Models

Generative (Joint) Models

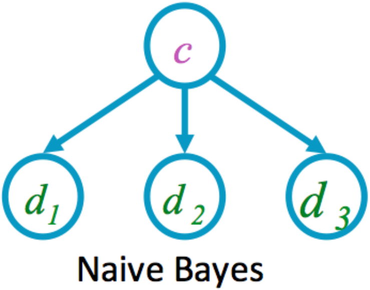
Generate the observed data from hidden stuff, i.e. put a probability over the observations given the class: $P(d,c)$ in terms of $P(d|c)$
e.g. Naïve Bayes' classifiers, Hidden Markov Models etc.

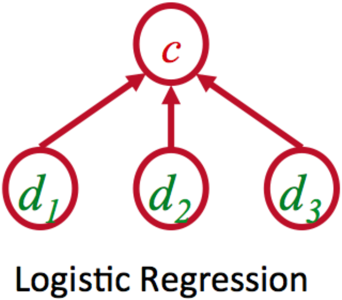
Discriminative (Conditional) Models

Take the data as given, and put a probability over hidden structure given the data: $P(c|d)$
e.g. Logistic regression, maximum entropy models, conditional random fields

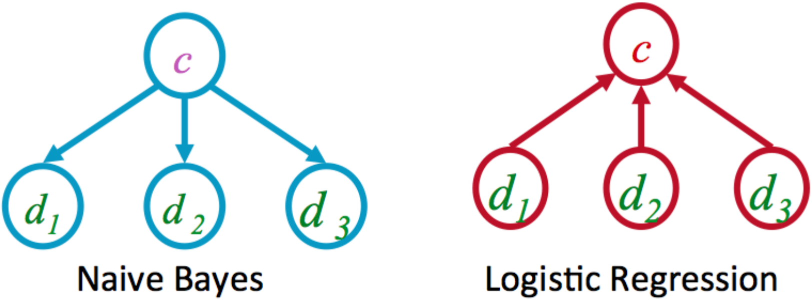
SVMs, perceptron, etc. are discriminative classifiers but not directly probabilistic

Generative vs. Discriminative Models





Generative vs. Discriminative Models



Joint vs. conditional likelihood

- A *joint* model gives probabilities $P(d,c)$ and tries to maximize this joint likelihood.
- A *conditional* model gives probabilities $P(c|d)$, taking the data as given and modeling only the conditional probability of the class.

Hidden Markov Models for POS Tagging

Pawan Goyal

CSE, IITKGP

Week 3: Lecture 5

Probabilistic Tagging

- $W = w_1 \ldots w_n$ - words in the corpus (observed)
- $T = t_1 \ldots t_n$ - the corresponding tags (unknown)

Probabilistic Tagging

- $W = w_1 \ldots w_n$ - words in the corpus (observed)
- $T = t_1 \ldots t_n$ - the corresponding tags (unknown)

Tagging: Probabilistic View (Generative Model)

Find

$$\hat{T} = argmax_T P(T|W)$$

Probabilistic Tagging

- $W = w_1 \dots w_n$ - words in the corpus (observed)
- $T = t_1 \dots t_n$ - the corresponding tags (unknown)

Tagging: Probabilistic View (Generative Model)

Find

$$\begin{aligned}\hat{T} &= argmax_T P(T|W) \\ &= argmax_T \frac{P(W|T)P(T)}{P(W)}\end{aligned}$$

Probabilistic Tagging

- $W = w_1 \ldots w_n$ - words in the corpus (observed)
- $T = t_1 \ldots t_n$ - the corresponding tags (unknown)

Tagging: Probabilistic View (Generative Model)

Find

$$
\begin{aligned}
\hat{T} &= argmax_T P(T|W) \\
&= argmax_T \frac{P(W|T)P(T)}{P(W)} \\
&= argmax_T P(W|T)P(T)
\end{aligned}
$$

Probabilistic Tagging

- $W = w_1 \dots w_n$ - words in the corpus (observed)
- $T = t_1 \dots t_n$ - the corresponding tags (unknown)

Tagging: Probabilistic View (Generative Model)

Find

$$
\begin{aligned}
\hat{T} &= argmax_T P(T|W) \\
&= argmax_T \frac{P(W|T)P(T)}{P(W)} \\
&= argmax_T P(W|T)P(T) \\
&= argmax_T \prod_i P(w_i|w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i|t_1 \dots t_{i-1})
\end{aligned}
$$

Further simplifications

$$\hat{T} = argmax_T \prod_i P(w_i|w_1 \ldots w_{i-1}, t_1 \ldots t_i) P(t_i|t_1 \ldots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag

Further simplifications

$$\hat{T} = argmax_T \prod_i P(w_i|w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i|t_1 \dots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag
  $P(w_i|w_1 \dots w_{i-1}, t_1 \dots t_i) \approx P(w_i|t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag

Further simplifications

$$\hat{T} = argmax_T \prod_i P(w_i|w_1 \dots w_{i-1}, t_1 \dots t_i) P(t_i|t_1 \dots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag
  $P(w_i|w_1 \dots w_{i-1}, t_1 \dots t_i) \approx P(w_i|t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag
  $P(t_i|t_1 \dots t_{i-1}) \approx P(t_i|t_{i-1})$

Further simplifications

$$\hat{T} = argmax_T \prod_i P(w_i|w_1 \ldots w_{i-1}, t_1 \ldots t_i) P(t_i|t_1 \ldots t_{i-1})$$

- The probability of a word appearing depends only on its own POS tag
  $P(w_i|w_1 \ldots w_{i-1}, t_1 \ldots t_i) \approx P(w_i|t_i)$
- Bigram assumption: the probability of a tag appearing depends only on the previous tag
  $P(t_i|t_1 \ldots t_{i-1}) \approx P(t_i|t_{i-1})$
- Using these simplifications:
  $\hat{T} = argmax_T \prod_i P(w_i|t_i) P(t_i|t_{i-1})$

Computing the probability values

Tag Transition probabilities $p(t_i|t_{i-1})$

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

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = 0.49$$

Computing the probability values

Tag Transition probabilities $p(t_i|t_{i-1})$

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

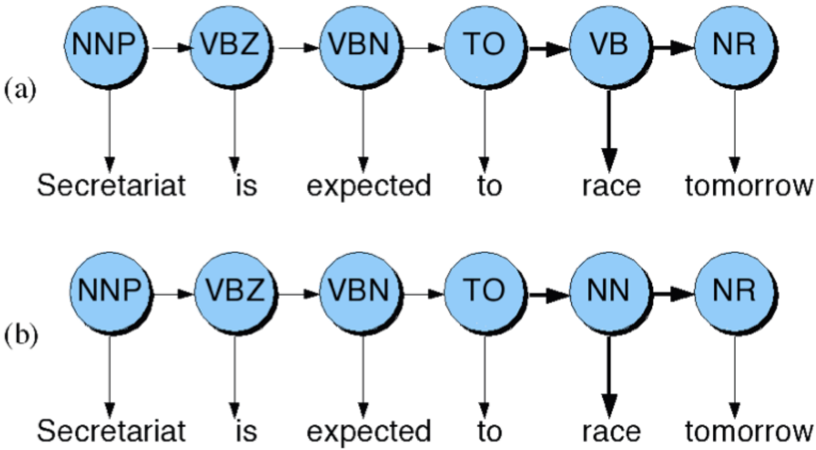
$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = 0.49$$

Word Likelihood probabilities $p(w_i|t_i)$

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

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = 0.47$$

Disambiguating "race"



Disambiguating "race"

Difference in probability due to

- $P(VB|TO)$ vs. $P(NN|TO)$
- $P(race|VB)$ vs. $P(race|NN)$
- $P(NR|VB)$ vs. $P(NR|NN)$

Disambiguating "race"

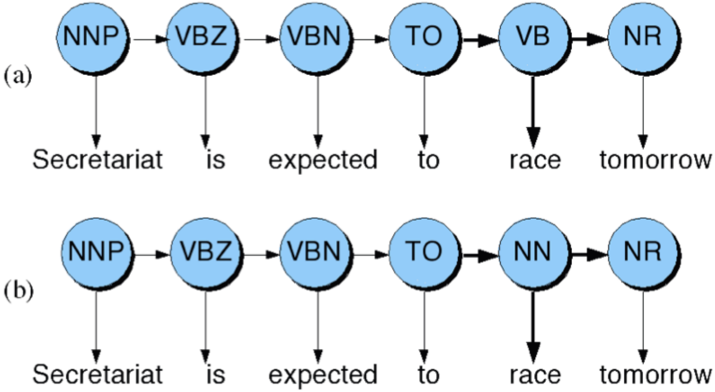
Difference in probability due to

- $P(VB|TO)$ vs. $P(NN|TO)$
- $P(race|VB)$ vs. $P(race|NN)$
- $P(NR|VB)$ vs. $P(NR|NN)$

After computing the probabilities

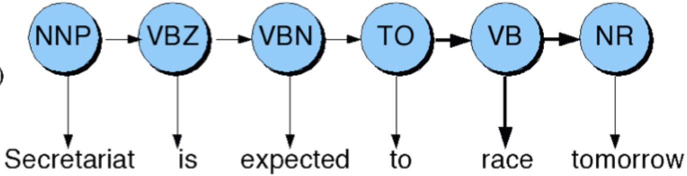
- $P(NN|TO)P(NR|NN)P(race|NN) = 0.0047 \times 0.0012 \times 0.00057 = 0.00000000032$
- $P(VB|TO)P(NR|VB)P(race|VB) = 0.83 \times 0.0027 \times 0.00012 = 0.00000027$

What is this model?

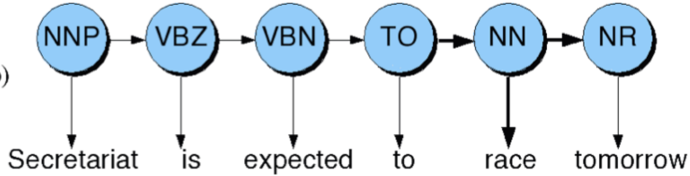


What is this model?

(a)



(b)



This is a Hidden Markov Model

Hidden Markov Models

- Tag Transition probabilities $p(t_i|t_{i-1})$
- Word Likelihood probabilities (emissions) $p(w_i|t_i)$

Hidden Markov Models

- Tag Transition probabilities $p(t_i|t_{i-1})$
- Word Likelihood probabilities (emissions) $p(w_i|t_i)$
- What we have described with these probabilities is a hidden markov model.
- Let us quickly introduce the Markov Chain, or observable Markov Model.

Markov Chain = First-order Markov Model

Weather example

- Three types of weather: *sunny, rainy, foggy*

Markov Chain = First-order Markov Model

Weather example

- Three types of weather: *sunny, rainy, foggy*
- q_n: variable denoting the weather on the n^{th} day

Markov Chain = First-order Markov Model

Weather example

- Three types of weather: *sunny, rainy, foggy*
- q_n: variable denoting the weather on the n^{th} day
- We want to find the following conditional probabilities:

$$P(q_n | q_{n-1}, q_{n-2}, \ldots, q_1)$$

Markov Chain = First-order Markov Model

Weather example

- Three types of weather: *sunny, rainy, foggy*
- q_n: variable denoting the weather on the n^{th} day
- We want to find the following conditional probabilities:

$$P(q_n|q_{n-1}, q_{n-2}, \ldots, q_1)$$

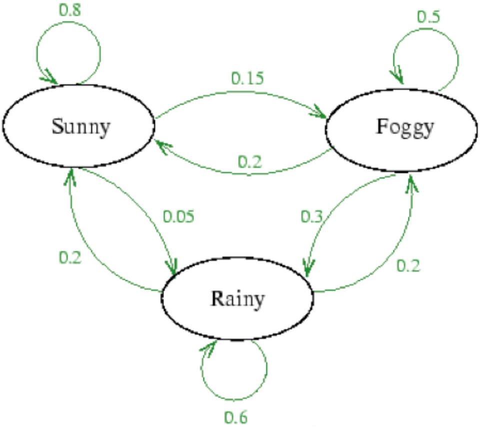
First-order Markov Assumption

$$P(q_n|q_{n-1}, q_{n-2}, \ldots, q_1) = P(q_n|q_{n-1})$$

Markov Chain Transition Table

Table 1: Probabilities $p(q_{n+1}|q_n)$ of tomorrow's weather based on today's weather

	Tomorrow's weather		
Today's weather			
	0.8	0.05	0.15
	0.2	0.6	0.2
	0.2	0.3	0.5



Using Markov Chain

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

Using Markov Chain

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

$$P(q_2 = sunny, q_3 = rainy | q_1 = sunny)$$

Using Markov Chain

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

$$P(q_2 = sunny, q_3 = rainy | q_1 = sunny)$$

$$= \quad P(q_3 = rainy | q_2 = sunny, q_1 = sunny) \times P(q_2 = sunny | q_1 = sunny)$$

- Given that today the weather is sunny, what is the probability that tomorrow is sunny and day after is rainy?

$$P(q_2 = sunny, q_3 = rainy | q_1 = sunny)$$

$$
\begin{aligned}
&= P(q_3 = rainy | q_2 = sunny, q_1 = sunny) \times P(q_2 = sunny | q_1 = sunny) \\
&= P(q_3 = rainy | q_2 = sunny) \times P(q_2 = sunny | q_1 = sunny) \\
&= 0.05 \times 0.8 \\
&= 0.04
\end{aligned}
$$

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states
  'sunny' weather is both observable and state

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states
  'sunny' weather is both observable and state
- But in POS tagging

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states
  'sunny' weather is both observable and state
- But in POS tagging
  The output symbols are words

Hidden Markov Model

- For Markov chains, the output symbols are the same as the states
  'sunny' weather is both observable and state
- But in POS tagging
  The output symbols are words
  But the hidden states are POS tags
- A Hidden Markov Model is an extension of a Markov chain in which the output symbols are not the same as the states
- We don't know which state we are in

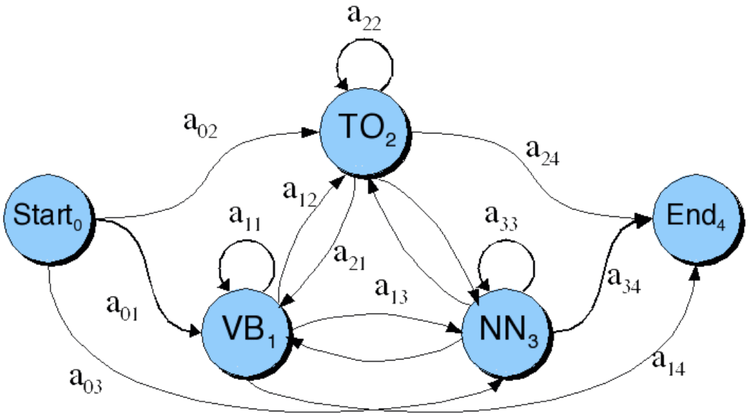
Hidden Markov Models (HMMs)

Elements of an HMM model

- A set of states (here: the tags)
- An output alphabet (here: words)
- Initial state (here: beginning of sentence)
- State transition probabilities (here $p(t_n|t_{n-1})$)
- Symbol emission probabilities (here $p(w_i|t_i)$)

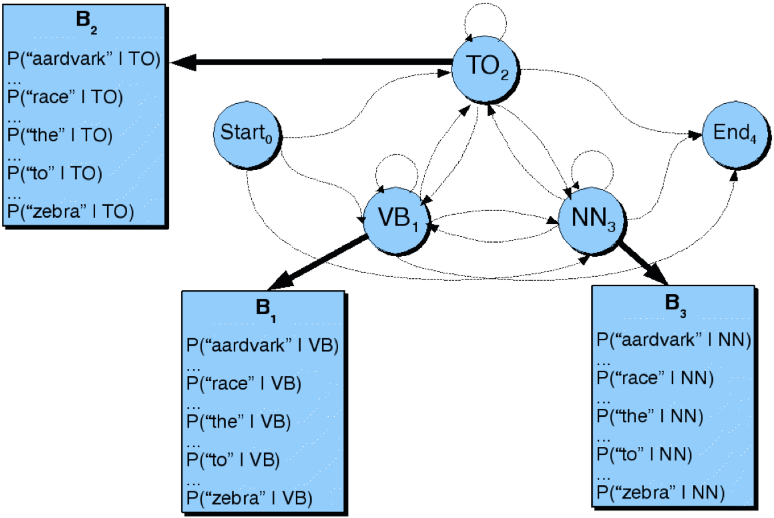
Graphical Representation

When tagging a sentence, we are walking through the state graph:

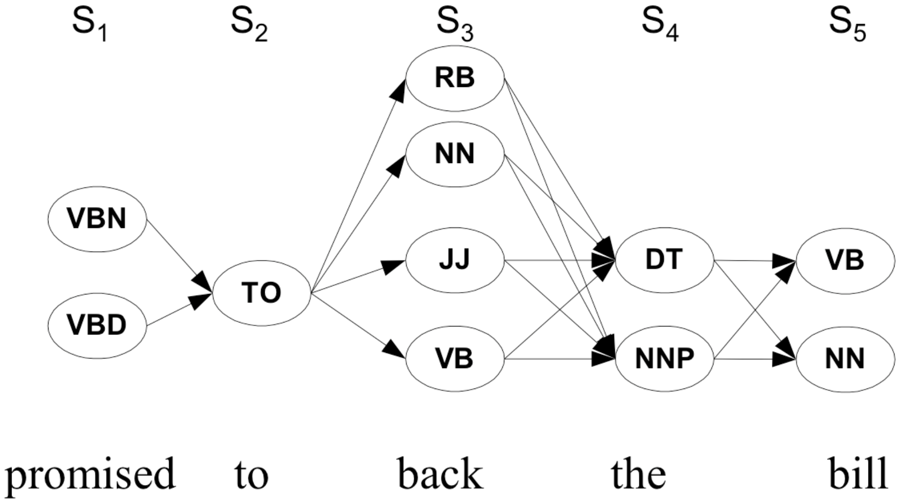


Edges are labeled with the state transition probabilities: $p(t_n|t_{n-1})$

At each state we emit a word: $P(w_n|t_n)$



Walking through the states: best path



Walking through the states: best path