5 - Part-of-Speech Tagging

CS 6320

Outline

- English word classes
- Penn Treebank tagset
- Rule-Based POS Tagging
- HMM POS Tagging
- Transformation-Based POS Tagging
- Probabilistic Models of Spelling
  - Noisy Channel Model
  - Applying Bayes Noisy Channel to Spelling

Closed Word Classes

- Prepositions: usually before a NP.
  (on, under, over, at, from, etc.)
- Determiners: usually precede a NP.
  (a, an, the, etc.)
- Pronouns: can replace an NP.
  (she, who, I, others, etc.)
- Conjunctions: join two phrases, clauses or sentences.
  (and, but, or, etc.)
- Auxiliary verbs: followed by another full verb
  (can, may, should, are, etc.)
- Particles: appear after verbs.
  (up, down, on, off, in, out, etc.)
- Numerals
  (one, two, first, second, etc.)

Open Word Classes

- Nouns
- Verbs
- Adjectives
- Adverbs

Prepositions from CELEX

of	540,085	through	14,964	worth	1,563	pace	12
in	331,235	after	13,670	toward	1,390	nigh	9
for	142,421	between	13,275	plus	750	re	4
to	125,691	under	9,525	till	686	mid	3
with	124,965	per	6,515	amongst	525	o'er	2
on	109,129	among	5,090	via	351	but	0
at	100,169	within	5,030	amid	222	ere	0
by	77,794	towards	4,700	underneath	164	less	0
from	74,843	above	3,056	versus	113	midst	0
about	38,428	near	2,026	amidst	67	o'	0
than	20,210	off	1,695	sans	20	thru	0
over	18,071	past	1,575	circa	14	vice	0

English Particles

aboard	aside	besides	forward(s)	opposite	through
about	astray	between	home	out	throughout
above	away	beyond	in	outside	together
across	back	by	inside	over	under
ahead	before	close	instead	overhead	underneath
alongside	behind	down	near	past	up
apart	below	east, etc.	off	round	within
around	beneath	eastward(s),etc.	on	since	without

Conjunctions

and	514,946	yet	5,040	considering	174	forasmuch as	0
that	134,773	since	4,843	lest	131	however	0
but	96,889	where	3,952	albeit	104	immediately	0
or	76,563	nor	3,078	providing	96	in as far as	0
as	54,608	once	2,826	whereupon	85	in so far as	0
if	53,917	unless	2,205	seeing	63	inasmuch as	0
when	37,975	why	1,333	directly	26	insomuch as	0
because	23,626	now	1,290	ere	12	insomuch that	0
so	12,933	neither	1,120	notwithstanding	3	like	0
before	10,720	whenever	913	according as	0	neither nor	0
though	10,329	whereas	867	as if	0	now that	0
than	9,511	except	864	as long as	0	only	0
while	8,144	till	686	as though	0	provided that	0
after	7,042	provided	594	both and	0	providing that	0
whether	5,978	whilst	351	but that	0	seeing as	0
for	5,935	suppose	281	but then	0	seeing as how	0
although	5,424	cos	188	but then again	0	seeing that	0
until	5,072	supposing	185	either or	0	without	0

Tags

- There are various tagsets to choose from.
- POS tagging is the assignment of correct POS tags for each word in an input sentence.
- Penn Treebank partofspeech tags.

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*			

Why is POS Tagging Useful?

- First step of a vast number of practical tasks
- Speech synthesis
  - How to pronounce "lead"?
  - OBject obJECT
  - CONtent conTENT
- Parsing
  - Need to know if a word is an N or V before you can parse
- Information extraction
  - Finding names, relations, etc.
- Machine Translation

Using the Penn Tagset

- The/DT grand/JJ jury/NN commmented/VBD on/IN a/DT number/NN of/IN other/JJ topics/NNS ./.
- Prepositions and subordinating conjunctions marked IN ("although/IN I/PRP..")
- Except the preposition/complementizer "to" is just marked "TO".

Using the Penn Tagset

EXAMPLE ANNOTATIONS:

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**

Preliminary/JJ findings/NNS were/VBD **reported/VBN** in/IN today/NN **'s/POS** New/NNP England/NNP Journal/NNP of/IN Medicine/NNP ./.

POS Tagging

- Words often have more than one POS: *back*
  - The *back* door = JJ
  - On my *back* = NN
  - Win the voters *back* = RB
  - Promised to *back* the bill = VB
- The POS tagging problem is to determine the POS tag for a particular instance of a word.

These examples from Dekang Lin

How Hard is POS Tagging? Measuring Ambiguity

		87-tag Original Brown		45-tag Treebank Brown	
Unambiguous (1 tag)		**44,019**		**38,857**	
Ambiguous (2–7 tags)		**5,490**		**8844**	
Details:	2 tags	4,967		6,731	
	3 tags	411		1621	
	4 tags	91		357	
	5 tags	17		90	
	6 tags	2	(*well*, *beat*)	32	
	7 tags	2	(*still*, *down*)	6	(*well*, *set*, *round*, *open*, *fit*, *down*)
	8 tags			4	(*'s*, *half*, *back*, *a*)
	9 tags			3	(*that*, *more*, *in*)

POS - Tagging

- We use:
  - A set of tags
  - A dictionary that indicates all possible tags for each word
  - Input text
  - General purpose vs. special purpose

- Methods:
  - Rule-based tagging
  - Statistical tagging
  - Transformation-based tagging

Rule-Based Tagging

- Start with a dictionary
- Assign all possible tags to words from the dictionary
- Write rules by hand to selectively remove tags
- Leaving the correct tag for each word.

Start With a Dictionary

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

- Etc… for the ~100,000 words of English with more than 1 tag

Assign Every Possible Tag

			NN		
			RB		
	VBN		JJ		VB
PRP	VBD	TO	VB	DT	NN
She	**promised**	**to**	**back**	**the**	**bill**

Write Rules to Eliminate Tags

Eliminate VBN if VBD is an option when VBN|VBD follows
"<start> PRP"

			NN		
			RB		
	VBN		JJ		VB
PRP	VBD	TO	VB	DT	NN
She	**promised**	**to**	**back**	**the**	**bill**

Transformation-based Tagging

- The pure rule based approach is too expensive, slow, tedious.
- Brill's Transformation Based Learning (TLB)
- Basic idea is to do a poor job first, and then use learned rules to improve things.
- Example: tag "race"

  Step 1 :

  P(NN|race) = 0.98

  P(VB|race) = 0.02
  - Tag all uses of race as nouns

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

    the/DT race/NN for/IN outer/JJ space/NN

Brill's Tagging 1/4

Step 2:

- Rule: Change NN to VB when the previous tag is TO.
- Assume some tagged training corpus.
- Make the assumption that the word depends only on its tag.

Brill's Tagging 2/4

- How are the rules learned?

1. Tag the corpus with the most likely tag for each word (unigram model)
2. Choose a transformation that deterministically replaces an existing tag with a new tag such that the resulting tagged training corpus has the lowest error rate out of all transformations
3. Apply that transformation to the training set
4. Iterate
5. Return as your tagger one that:
   - First tags using unigrams and then
   - Applies the learned transformations in order.

Brill's Tagging 3/4

- Transformations
- Change Tag a to Tag b when:

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**.

	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	

Brill's Tagging 4/4

```
function TBL(corpus) returns transforms-queue
  INITIALIZE-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)
```

```
function GET-BEST-TRANSFORM(corpus, templates) returns transform
  for each template in templates
    (instance, score) ← GET-BEST-INSTANCE(corpus, template)
    if (score > best-transform.score) then best-transform ← (instance, score)
  return(best-transform)
```

Brill's Tagging 4/4 (Continued)

```
function GET-BEST-INSTANCE(corpus, template) returns transform
 for from-tag ← from tag_1 to tag_n do
  for to-tag ← from tag_1 to tag_n do
   for pos ← from 1 to corpus-size do
    if (correct-tag(pos) == to-tag && current-tag(pos) == from-tag)
        num-good-transforms(current-tag(pos−1))++
    elseif (correct-tag(pos)==from-tag && current-tag(pos)==from-tag)
        num-bad-transforms(current-tag(pos−1))++
   end
   best-Z ← ARGMAX_t(num-good-transforms(t) - num-bad-transforms(t))
   if(num-good-transforms(best-Z) - num-bad-transforms(best-Z)
              > best-instance.score) then
    best.rule ← "Change tag from from-tag to to-tag if prev tag is best-Z"
    best.score ← num-good-transforms(best-Z) - num-bad-transforms(best-Z)
return(best)
```

```
procedure APPLY-TRANSFORM(transform, corpus)
for pos ← from 1 to corpus-size do
 if (current-tag(pos)==best-rule-from)
      && (current-tag(pos−1)==best-rule-prev))
   current-tag(pos) ← best-rule-to
```

Basic Probability Background

- Basic Probability Background

  Prior Probability (or unconditional probability): *P(A)*

- Think of A as a proposition as in some simple propositional logic.

- Also useful...

$$P(A \wedge B), P(A \vee B), P(\neg A \wedge B), etc$$

Relating Conditionals and Priors

$$P(A \mid B) = \frac{P(A \wedge B)}{P(B)}$$

rearranging yields…

$$P(A \wedge B) = P(A \mid B)P(B)$$

you could also say…

$$P(A \wedge B) = P(B \mid A)P(A)$$

Bayesian Reasoning

- We know … $P(A \wedge B) = P(A \mid B)P(B)$

  and that $P(A \wedge B) = P(B \mid A)P(A)$

  so $P(A \mid B)P(B) = P(B \mid A)P(A)$

  or $P(A \mid B) = \frac{P(B \mid A)P(A)}{P(B)}$

  or $P(B \mid A) = \frac{P(A \mid B)P(B)}{P(A)}$

  This is commonly known as **Bayes law**. The key point is that we can move from *P(A/B)* to *P(B/A)* and back given appropriate information.

POS Tagging as Sequence Classification

- We are given a sentence (an "observation" or "sequence of observations")
  - *Secretariat is expected to race tomorrow*
- What is the best sequence of tags that corresponds to this sequence of observations?
- Probabilistic view:
  - Consider all possible sequences of tags
  - Out of this universe of sequences, choose the tag sequence which is most probable given the observation sequence of n words $w_1 \ldots w_n$.

Using Hidden Markov Models (HMMs)

- We want, out of all sequences of n tags $t_1 \ldots t_n$ the single tag sequence such that $P(t_1 \ldots t_n | w_1 \ldots w_n)$ is highest.

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n \mid w_1^n)$$

- Hat ^ means "our estimate of the best one"
- argmax_x f(x) means "the x such that f(x) is maximized"

Getting to HMMs

- This equation is guaranteed to give us the best tag sequence

$$\hat{t}_1^n = \arg\max_{t_1^n} P(t_1^n \mid w_1^n)$$

- But how to make it operational? How to compute this value?
- Intuition of Bayesian classification:
  - Use Bayes rule to transform this equation into a set of other probabilities that are easier to compute

Using Bayes Rule

$$P(x|y) = \frac{P(y|x)P(x)}{P(y)}$$

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

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

Likelihood and Prior

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

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

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

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

Two Kinds of Probabilities

- Tag transition probabilities $p(t_i|t_{i-1})$
  - Determiners likely to precede adjs and nouns
    - That/DT flight/NN
    - The/DT yellow/JJ hat/NN
    - So we expect P(NN|DT) and P(JJ|DT) to be high
    - But P(DT|JJ) to be low
  - Compute P(NN|DT) by counting in a labeled corpus:

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

$$P(NN|DT) = \frac{C(DT, NN)}{C(DT)} = \frac{56,509}{116,454} = .49$$

Two Kinds of Probabilities

- Word likelihood probabilities $p(w_i|t_i)$
  - VBZ (3sg Pres verb) likely to be "is"
  - Compute P(is|VBZ) by counting in a labeled corpus:

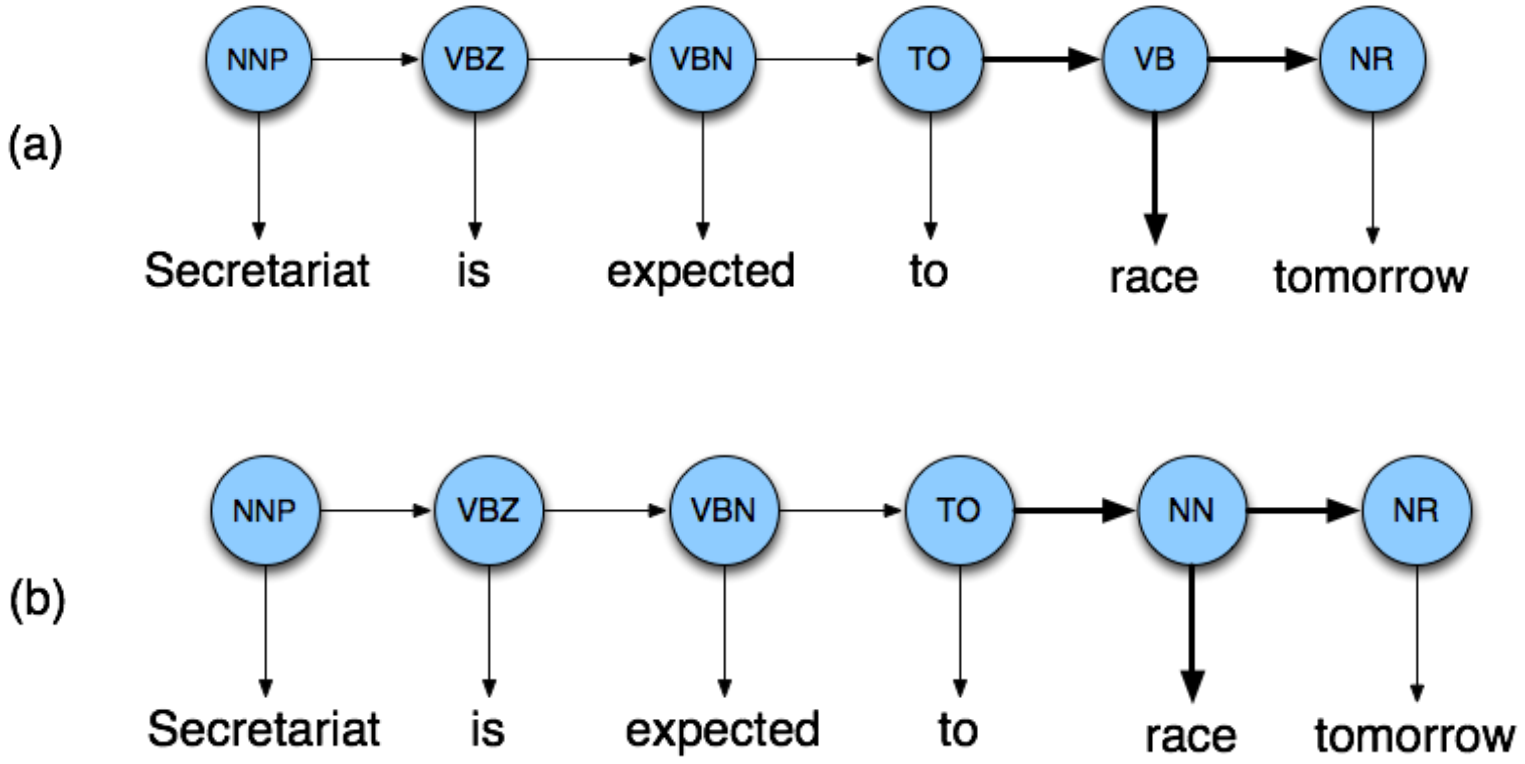
$$P(w_i|t_i) = \frac{C(t_i, w_i)}{C(t_i)}$$

$$P(is|VBZ) = \frac{C(VBZ, is)}{C(VBZ)} = \frac{10,073}{21,627} = .47$$

Example: The Verb "race"

- Secretariat/NNP is/VBZ expected/VBN to/TO **race**/VB tomorrow/NR
- People/NNS continue/VB to/TO inquire/VB the/DT reason/NN for/IN the/DT **race**/NN for/IN outer/JJ space/NN
- How do we pick the right tag?

Disambiguating "race"



Example

- P(NN|TO) = .00047
- P(VB|TO) = .83
- P(race|NN) = .00057
- P(race|VB) = .00012
- P(NR|VB) = .0027
- P(NR|NN) = .0012
- P(VB|TO)P(NR|VB)P(race|VB) = .00000027
- P(NN|TO)P(NR|NN)P(race|NN)=.00000000032
- So we (correctly) choose the tag verb

Statistical POS Tagging using Trigrams

- In general: $\arg\max \; P(Tag\ Sequence \mid Word\ Sequence)$

- Rewrite this:

$$\text{argmax} \frac{P(WordSequence \mid TagSequence) P(TagSequence)}{P(WordSequence)}$$

$$or$$

$$\hat{T} = \text{argmax} \; P(T)P(W \mid T)$$

$$P(T)P(W \mid T) = \prod_{i=1}^{n} P(w_i \mid w_1 t_1 \cdots w_{i-1} t_{i-1} t_i) P(t_i \mid w_1 t_1 \cdots w_{i-1} t_{i-1})$$

- $P(Tag \quad Sequence \quad) = P(t_1) P(t_2 \mid t_1) \prod_{i=3}^{n} P(t_i \mid t_{i-2} t_{i-1})$

- This is easy to get from simple word counting and smoothing:

$$P(t_i \mid t_{i-2} t_{i-1}) = \frac{C(t_{i-2} t_{i-1} t_i)}{C(t_{i-2} t_{i-1})}$$

Statistical POS Tagging

- Make the assumption that the word depends only on its tag.

$$P(Word\ Sequence \mid Tag\ Sequence)$$

$$P(w_i \mid w_1 t_1 \cdots w_{i-1} t_{i-1} t_i) = P(w_i \mid t_i)$$

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

$$\prod_{i=1}^{n} P(w_i \mid t_i)$$

- Combine the two factors.

$$P(t_1) P(t_2 \mid t_1) \prod_{i=3}^{n} P(t_i \mid t_{i-2} t_{i-1}) \left[\prod_{i=1}^{n} P(w_i \mid t_i) \right]$$

- The state transition probabilities come from the language model.
- The emission probabilities come from P(word|tag).
- Find the best tag sequence - overall.

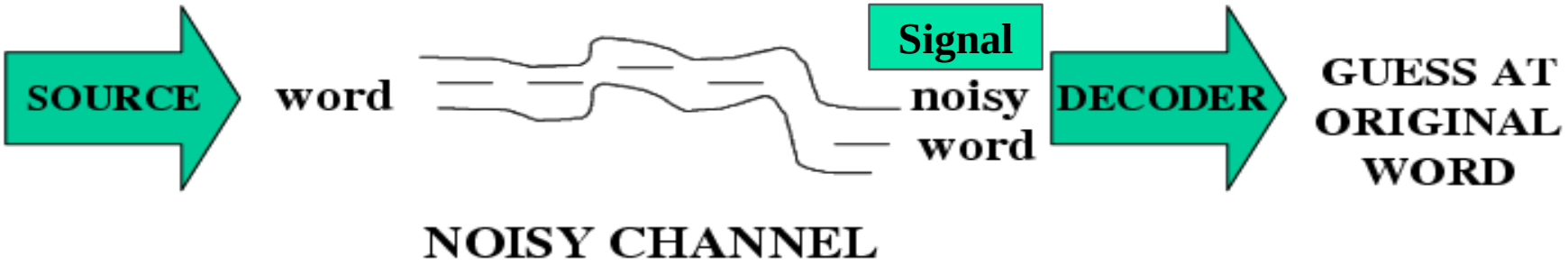
Probabilistic Models of Spelling

Probabilistic Models of Pronunciation and Spelling

- Some problems with Finite State Machines:
  - In the case of a reject, they offer no advice as to why a string was rejected.
  - In the case of global ambiguity (two or more accept conditions or paths) FSAs simply provide all or one.

- So...we're going to make a brief digression away from FSAs to look at alternative. Then we'll find a way back to FSAs by augmenting them slightly.

The Noisy Channel Model



- Many problems in language processing can be viewed as noisy channel problems
  - Optical character recognition
  - Spelling correction
  - Speech recognition
  - Machine translation

Applying Bayes to a Noisy Channel 1/2

- In applying probability theory to a noisy channel what we're looking for is the most probable source (original word) given the observed signal (noisy word). We can denote this as:

$$\arg\max_{Source} P(Source \mid Signal)$$

- Unfortunately, we don't usually know how to compute this, so…

- Rewrite … $P(Source \mid Signal) = \frac{P(Signal \mid Source)P(Source)}{P(Signal)}$

- Giving $\arg\max_{Source} = \frac{P(Signal \mid Source)P(Source)}{P(Signal)}$

Applying Bayes to a Noisy Channel 2/2

How does this help if all we have is the signal and the source is exactly what we don't know?

We know the space of possible sources. We can plug each into the equation one by one and compute their probabilities using this equation. The source hypothesis with the highest probability wins.

Applying Bayes Noisy Channel to Spelling

- We have some word that has been misspelled and we want to know the original word (i.e. correctly spelled word). The original word is the source, and the misspelled word is the signal.

- Assume that V is the space of all the words we know.

$\hat{w}$ denotes the correct word, and *s* denotes the misspelling.

$$\hat{w} = \arg\max_{w \in V} \frac{P(s \mid w)P(w)}{P(s)}$$

Getting the Numbers

- It seems like we need...
  - *P(s/w)*
  - *P(w)*
  - *P(s)*
- Let's first consider the *P(s)* in the equation...

$$\hat{w} = \arg\max_{w \in V} \frac{P(s \mid w)P(w)}{P(s)}$$

What about *P(w)*...

$$\hat{w} = \arg\max_{w \in V} \frac{P(s \mid w)P(w)}{P(s)}$$

What about *P(s/w)*...

$$\hat{w} = \arg\max_{w \in V} \frac{P(s \mid w)P(w)}{P(s)}$$

Let's consider the example … collect statistics on the probability of misspelling "actress" and "acress".

Spelling Error Patterns

- It is fruitless to try to collect statistics about the misspellings of individual words given a large dictionary. You'll likely never get enough data.

- We need a way to compute P(s|w) without using direct information.

- This is where spelling error patterns come in...
  - Insertion: ther for the
  - Deletion: ther for there
  - Substitution : noq for now
  - Transportation: teh for the

Spelling Error Statistics 1/2

- Collect statistics for each error type from a large corpus.
- For example... asking for P(acress|actress) is assumed to be the same as asking for the probability that a deletion of “t" happened here.
- So... just collect a large corpus of text (containing errors) and see how often “t" gets deleted.

Spelling Error Statistics 2/2

- For the general case we can use:
  - del[x,y]: number of times in the training set when characters xy were typed as x.
  - ins[x,y]: number of times when x in the correct word was typed as xy.
  - sub[x,y]: number of times when x was typed as y.
  - trans[x,y]: number of times when xy was typed as yx.

Kernighan's Statistics

$$P(t \mid c) = \begin{cases} deletion, \dfrac{del[c_{p-1}, c_p]}{count[c_{p-1}, c_p]} \\ insertion, \dfrac{ins[c_{p-1,}, t_p]}{count[c_{p-1}]} \\ substitution, \dfrac{sub[t_p, c_p]}{count[c_p]} \\ trasposition, \dfrac{trans[c_p, c_{p+1}]}{count[c_p c_{p+1}]} \end{cases}$$

- So… for transposing h and e as in "teh" for "the". Divide the numbers of occurrences of "eh" that are errors by the total number of "eh" occurrences in the corpus.

Kernighan Method…

- Apply all possible single spelling changes to the misspelled word.
- Collect all the resulting strings that are actually words.
- Compute the probability of each of those candidate words.
- Display them ranked to the user.

Results for misspelled word "acress"

$$freq(c) = C(c)$$

$$P(c) = \frac{C(c) + 0.5}{N + 0.5V}$$

$$N = 44Million$$

c	freq(c)	p(c)	p(t|c)	p(t|c)p(c)	%
actress	1343	.0000315	.000117	3.69×10^{-9}	**37%**
cress	0	.000000014	.00000144	2.02×10^{-14}	**0%**
caress	4	.0000001	.00000164	1.64×10^{-13}	**0%**
access	2280	.000058	.000000209	1.21×10^{-11}	**0%**
across	8436	.00019	.0000093	1.77×10^{-9}	**18%**
acres	2879	.000065	.0000321	2.09×10^{-9}	**21%**
acres	2879	.000065	.0000342	2.22×10^{-9}	**23%**

Conclusion: most likely the intended word was "acres"

Problems

- Doesn't incorporate contextual information:
  a stellar and versatile acress whose...
- Chapter 6 will address this.
- There are usually a million ways to instantiate a Bayesian model. It depends on whether you are willing /able to collect the data needed for the model you'd like to choose.
- Big issues in this approach
  - The corpus,
  - Zero counts,
  - What to condition on.