Introduction to

# Information Retrieval

CS276: Information Retrieval and Web Search
Christopher Manning and Pandu Nayak

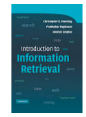
Spelling Correction

---



## The course thus far …

Index construction
Index compression
Efficient boolean querying
  Chapters 1, 2, 4, 5
  Coursera lectures 1, 2, 3, 4




Spelling correction
  Chapter 3
  Coursera lecture 5 (mainly some parts)
  **This lecture** (PA #2!)

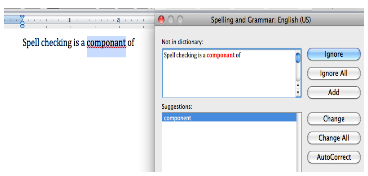


---



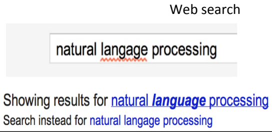
## Applications for spelling correction

Word processing




Phones


Web search






---



## Rates of spelling errors

Depending on the application, ~1–20% error rates

**26%**: Web queries Wang *et al.* 2003
**13%**: Retyping, no backspace: Whitelaw *et al.* English&German
**7%**: Words corrected retyping on phone-sized organizer
**2%**: Words uncorrected on organizer Soukoreff &MacKenzie 2003
**1-2%**: Retyping: Kane and Wobbrock 2007, Gruden et al. 1983



---



## Spelling Tasks

- Spelling Error Detection
- Spelling Error Correction:
  - Autocorrect
    - hte→the
  - Suggest a correction
  - Suggestion lists



---



## Types of spelling errors

- Non-word Errors
  - *graffe* →*giraffe*
- Real-word Errors
  - Typographical errors
    - *three* →*there*
  - Cognitive Errors (homophones)
    - *piece*→*peace*,
    - *too* → *two*
    - *your* →*you're*
- Non-word correction was historically mainly context insensitive
- Real-word correction almost needs to be context sensitive

## Non-word spelling errors

- Non-word spelling error detection:
  - Any word not in a ***dictionary*** is an error
  - The larger the dictionary the better … up to a point
  - (The Web is full of mis-spellings, so the Web isn't necessarily a great dictionary …)
- Non-word spelling error correction:
  - Generate ***candidates***: real words that are similar to error
  - Choose the one which is best:
    - Shortest weighted edit distance
    - Highest noisy channel probability





## Real word & non-word spelling errors

- For each word *w*, generate candidate set:
  - Find candidate words with similar ***pronunciations***
  - Find candidate words with similar ***spellings***
  - Include *w* in candidate set
- Choose best candidate
  - Noisy Channel view of spell errors
  - Context-sensitive – so have to consider whether the surrounding words "make sense"
  - *Flying form Heathrow to LAX → Flying from Heathrow to LAX*





## Terminology

- These are *character bigrams*:
  - *st, pr, an …*
- These are *word bigrams*:
  - *palo alto, flying from, road repairs*
- In today's class, we will generally deal with *word* bigrams
- In the accompanying Coursera lecture, we mostly deal with *character* bigrams (because we cover stuff complementary to what we're discussing here)

Similarly trigrams, *k*-grams etc



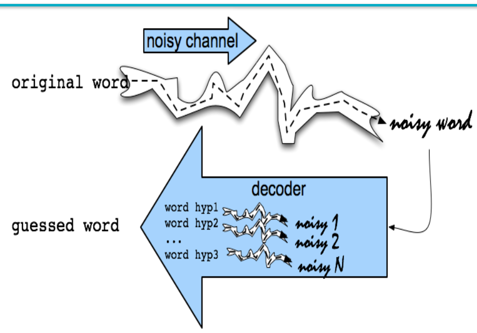


The Noisy Channel Model of Spelling

## INDEPENDENT WORD SPELLING CORRECTION



## Noisy Channel Intuition








## Noisy Channel = Bayes' Rule

- We see an observation $x$ of a misspelled word
- Find the correct word $\hat{w}$

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}}\, P(w \mid x)$$

$$= \underset{w \in V}{\operatorname{argmax}}\, \frac{P(x \mid w)P(w)}{P(x)}$$

Bayes

$$= \underset{w \in V}{\operatorname{argmax}}\, P(x \mid w)P(w)$$

## History: Noisy channel for spelling proposed around 1990

- **IBM**
  - Mays, Eric, Fred J. Damerau and Robert L. Mercer. 1991. Context based spelling correction. *Information Processing and Management*, 23(5), 517–522
- **AT&T Bell Labs**
  - Kernighan, Mark D., Kenneth W. Church, and William A. Gale. 1990. A spelling correction program based on a noisy channel model. Proceedings of COLING 1990, 205-210



## Non-word spelling error example

acress





## Candidate generation

- Words with similar spelling
  - Small *edit distance* to error
- Words with similar pronunciation
  - Small distance of pronunciation to error
- In this class lecture we mostly won't dwell on *efficient* candidate generation
- A lot more about candidate generation in the accompanying Coursera material





## Candidate Testing: Damerau-Levenshtein edit distance

- Minimal edit distance between two strings, where edits are:
  - Insertion
  - Deletion
  - Substitution
  - Transposition of two adjacent letters
- See *IIR* sec 3.3.3 for edit distance





## Words within 1 of acress

| Error | Candidate Correction | Correct Letter | Error Letter | Type |
|---|---|---|---|---|
| acress | actress | t | – | deletion |
| acress | cress | – | a | insertion |
| acress | caress | ca | ac | transposition |
| acress | access | c | r | substitution |
| acress | across | o | e | substitution |
| acress | acres | – | s | insertion |





## Candidate generation

- 80% of errors are within edit distance 1
- Almost all errors within edit distance 2
- Also allow insertion of **space** or **hyphen**
  - thisidea → this idea
  - inlaw → in-law
- Can also allow merging words
  - data base → database
  - For short texts like a query, can just regard whole string as one item from which to produce edits

## How do you generate the candidates?

1. Run through dictionary, check edit distance with each word
2. Generate all words within edit distance ≤ $k$ (e.g., $k$ = 1 or 2) and then intersect them with dictionary
3. Use a character $k$-gram index and find dictionary words that share "most" $k$-grams with word (e.g., by Jaccard coefficient)
   - see *IIR* sec 3.3.4
4. Compute them fast with a Levenshtein finite state transducer
5. Have a precomputed map of words to possible corrections





## A paradigm …

- We want the best spell corrections
- Instead of finding the very best, we
  - Find a subset of pretty good corrections
    - (say, edit distance at most 2)
  - Find the best amongst them
- *These may not be the actual best*
- This is a recurring paradigm in IR including finding the best docs for a query, best answers, best ads …
  - Find a good candidate set
  - Find the top *K amongst them* and return them as the best





## Let's say we've generated candidates: Now back to Bayes' Rule

- We see an observation *x* of a misspelled word
- Find the correct word $\hat{w}$

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}}\, P(w \mid x)$$

$$= \underset{w \in V}{\operatorname{argmax}}\, \frac{P(x \mid w)P(w)}{P(x)}$$

$$= \underset{w \in V}{\operatorname{argmax}}\, P(x \mid w)P(w)$$

What's *P(w)*?





## Language Model

- Take a big supply of words (your document collection with *T* tokens); let *C(w)* = # occurrences of *w*

$$P(w) = \frac{C(w)}{T}$$

- In other applications – you can take the supply to be typed queries (suitably filtered) – when a static dictionary is inadequate





## Unigram Prior probability

Counts from 404,253,213 words in Corpus of Contemporary English (COCA)

| word | Frequency of word | *P(w)* |
|---|---|---|
| actress | 9,321 | .0000230573 |
| cress | 220 | .0000005442 |
| caress | 686 | .0000016969 |
| access | 37,038 | .0000916207 |
| across | 120,844 | .0002989314 |
| acres | 12,874 | .0000318463 |





## Channel model probability

- **Error model probability, Edit probability**
- *Kernighan, Church, Gale 1990*
- *Misspelled word* $x = x_1, x_2, x_3 \ldots x_m$
- *Correct word* $w = w_1, w_2, w_3, \ldots, w_n$
- *P(x|w)* = probability of the edit
  - (deletion/insertion/substitution/transposition)

## Computing error probability: confusion "matrix"

```
del[x,y]:    count(xy typed as x)
ins[x,y]:    count(x typed as xy)
sub[x,y]:    count(y typed as x)
trans[x,y]:  count(xy typed as yx)
```

Insertion and deletion conditioned on previous character





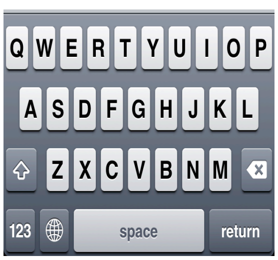
## Confusion matrix for substitution

sub[X, Y] = Substitution of X (incorrect) for Y (correct)

| X \ Y (correct) | a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | 0 | 0 | 7 | 1 | 342 | 0 | 0 | 2 | 118 | 0 | 1 | 0 | 0 | 3 | 76 | 0 | 0 | 1 | 35 | 9 | 9 | 0 | 1 | 0 | 5 | 0 |
| b | 0 | 0 | 9 | 9 | 2 | 2 | 3 | 1 | 0 | 0 | 0 | 5 | 11 | 5 | 0 | 10 | 0 | 0 | 2 | 1 | 0 | 0 | 8 | 0 | 0 | 0 |
| c | 6 | 5 | 0 | 16 | 0 | 9 | 5 | 0 | 0 | 0 | 1 | 0 | 7 | 9 | 1 | 10 | 2 | 5 | 39 | 40 | 1 | 3 | 7 | 1 | 1 | 0 |
| d | 1 | 10 | 13 | 0 | 12 | 0 | 5 | 5 | 0 | 0 | 2 | 3 | 7 | 3 | 0 | 1 | 0 | 43 | 30 | 22 | 0 | 0 | 4 | 0 | 2 | 0 |
| e | 388 | 0 | 3 | 11 | 0 | 2 | 2 | 0 | 89 | 0 | 0 | 3 | 0 | 5 | 93 | 0 | 0 | 14 | 12 | 6 | 15 | 0 | 1 | 0 | 18 | 0 |
| f | 0 | 15 | 0 | 3 | 1 | 0 | 5 | 2 | 0 | 0 | 0 | 3 | 4 | 1 | 0 | 0 | 0 | 6 | 4 | 12 | 0 | 0 | 2 | 0 | 0 | 0 |
| g | 4 | 1 | 11 | 11 | 9 | 2 | 0 | 0 | 0 | 1 | 1 | 3 | 0 | 0 | 2 | 1 | 3 | 5 | 13 | 21 | 0 | 0 | 1 | 0 | 3 | 0 |
| h | 1 | 8 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 0 | 2 | 0 | 12 | 14 | 2 | 3 | 0 | 3 | 1 | 11 | 0 | 0 | 2 | 0 | 0 | 0 |
| i | 103 | 0 | 0 | 0 | 146 | 0 | 1 | 0 | 0 | 0 | 0 | 6 | 0 | 0 | 49 | 0 | 0 | 0 | 2 | 1 | 47 | 0 | 2 | 1 | 15 | 0 |
| j | 0 | 1 | 1 | 9 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 2 | 1 | 0 | 0 | 0 | 0 | 0 | 5 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| k | 1 | 2 | 8 | 4 | 1 | 1 | 2 | 5 | 0 | 0 | 0 | 0 | 5 | 0 | 2 | 0 | 0 | 0 | 6 | 0 | 0 | 0 | 4 | 0 | 0 | 3 |
| l | 2 | 10 | 1 | 4 | 0 | 4 | 5 | 6 | 13 | 0 | 1 | 0 | 0 | 14 | 2 | 5 | 0 | 11 | 10 | 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| m | 1 | 3 | 7 | 8 | 0 | 2 | 0 | 6 | 0 | 0 | 4 | 4 | 0 | 180 | 0 | 6 | 0 | 0 | 9 | 15 | 13 | 3 | 2 | 2 | 3 | 0 |
| n | 2 | 7 | 6 | 5 | 3 | 0 | 1 | 19 | 1 | 0 | 4 | 35 | 78 | 0 | 0 | 7 | 0 | 28 | 5 | 7 | 0 | 0 | 1 | 2 | 0 | 2 |
| o | 91 | 1 | 1 | 3 | 116 | 0 | 0 | 0 | 25 | 0 | 2 | 0 | 0 | 0 | 0 | 14 | 0 | 2 | 4 | 14 | 39 | 0 | 0 | 0 | 18 | 0 |
| p | 0 | 11 | 1 | 2 | 0 | 6 | 5 | 0 | 2 | 9 | 0 | 2 | 7 | 6 | 15 | 0 | 0 | 1 | 3 | 6 | 0 | 4 | 1 | 0 | 0 | 0 |
| q | 0 | 0 | 1 | 0 | 0 | 0 | 27 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| r | 0 | 14 | 0 | 30 | 12 | 2 | 2 | 8 | 2 | 0 | 5 | 8 | 4 | 20 | 1 | 14 | 0 | 0 | 12 | 22 | 4 | 0 | 0 | 1 | 0 | 0 |
| s | 11 | 8 | 27 | 33 | 35 | 4 | 0 | 1 | 0 | 1 | 0 | 27 | 0 | 6 | 1 | 7 | 0 | 14 | 0 | 15 | 0 | 0 | 5 | 3 | 20 | 1 |
| t | 3 | 4 | 9 | 42 | 7 | 5 | 19 | 5 | 0 | 1 | 0 | 14 | 9 | 5 | 5 | 6 | 0 | 11 | 37 | 0 | 0 | 2 | 19 | 0 | 7 | 6 |
| u | 20 | 0 | 0 | 0 | 44 | 0 | 0 | 0 | 64 | 0 | 0 | 0 | 0 | 2 | 43 | 0 | 0 | 4 | 0 | 0 | 0 | 0 | 2 | 0 | 8 | 0 |
| v | 0 | 0 | 7 | 0 | 0 | 3 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 0 | 8 | 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| w | 2 | 2 | 1 | 0 | 1 | 0 | 0 | 2 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 7 | 0 | 6 | 3 | 3 | 1 | 0 | 0 | 0 | 0 | 0 |
| x | 0 | 0 | 0 | 2 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 9 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| y | 0 | 0 | 2 | 0 | 15 | 0 | 1 | 7 | 15 | 0 | 0 | 0 | 2 | 0 | 6 | 1 | 0 | 7 | 36 | 8 | 5 | 0 | 0 | 1 | 0 | 0 |
| z | 0 | 0 | 0 | 7 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 7 | 5 | 0 | 0 | 0 | 0 | 2 | 21 | 3 | 0 | 0 | 0 | 0 | 3 | 0 |



## Nearby keys






## Generating the confusion matrix

- Peter Norvig's list of errors
- Peter Norvig's list of counts of single-edit errors
  - All Peter Norvig's ngrams data links: http://norvig.com/ngrams/





## Channel model

Kernighan, Church, Gale 1990

$$P(x|w) = \begin{cases} \dfrac{\text{del}[w_{i-1}, w_i]}{\text{count}[w_{i-1} w_i]}, & \text{if deletion} \\ \dfrac{\text{ins}[w_{i-1}, x_i]}{\text{count}[w_{i-1}]}, & \text{if insertion} \\ \dfrac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}, & \text{if substitution} \\ \dfrac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_i w_{i+1}]}, & \text{if transposition} \end{cases}$$





## Smoothing probabilities: Add-1 smoothing

- But if we use the confusion matrix example, unseen errors are impossible!
- They'll make the overall probability 0. That seems too harsh
  - e.g., in Kernighan's chart q➔a and a➔q are both 0, even though they're adjacent on the keyboard!
- A simple solution is to add 1 to all counts and then if there is a |A| character alphabet, to normalize appropriately:

$$\text{If substitution, } P(x\,|\,w) = \frac{\text{sub}[x,w]+1}{\text{count}[w]+A}$$

## Channel model for acress

| Candidate Correction | Correct Letter | Error Letter | x\|w | P(x\|w) |
|---|---|---|---|---|
| actress | t | - | c\|ct | .000117 |
| cress | - | a | a\|# | .00000144 |
| caress | ca | ac | ac\|ca | .00000164 |
| access | c | r | r\|c | .000000209 |
| across | o | e | e\|o | .0000093 |
| acres | - | s | es\|e | .0000321 |
| acres | - | s | ss\|s | .0000342 |





| Candidate Correction | Correct Letter | Error Letter | x\|w | P(x\|w) | P(w) | $10^9$ * P(x\|w)* P(w) |
|---|---|---|---|---|---|---|
| actress | t | - | c\|ct | .000117 | .0000231 | 2.7 |
| cress | - | a | a\|# | .00000144 | .000000544 | .00078 |
| caress | ca | ac | ac\|ca | .00000164 | .00000170 | .0028 |
| access | c | r | r\|c | .000000209 | .0000916 | .019 |
| across | o | e | e\|o | .0000093 | .000299 | 2.8 |
| acres | - | s | es\|e | .0000321 | .0000318 | 1.0 |
| acres | - | s | ss\|s | .0000342 | .0000318 | 1.0 |



| Candidate Correction | Correct Letter | Error Letter | x\|w | P(x\|w) | P(w) | $10^9$ *P(x\|w)P(w) |
|---|---|---|---|---|---|---|
| actress | t | - | c\|ct | .000117 | .0000231 | 2.7 |
| cress | - | a | a\|# | .00000144 | .000000544 | .00078 |
| caress | ca | ac | ac\|ca | .00000164 | .00000170 | .0028 |
| access | c | r | r\|c | .000000209 | .0000916 | .019 |
| **across** | **o** | **e** | **e\|o** | **.0000093** | **.000299** | **2.8** |
| acres | - | s | es\|e | .0000321 | .0000318 | 1.0 |
| acres | - | s | ss\| | .0000342 | .0000318 | 1.0 |





## Evaluation

- Some spelling error test sets
  - Wikipedia's list of common English misspelling
  - Aspell filtered version of that list
  - Birkbeck spelling error corpus
  - Peter Norvig's list of errors (includes Wikipedia and Birkbeck, for training or testing)





Context-Sensitive Spelling Correction

## SPELLING CORRECTION WITH THE NOISY CHANNEL



## Real-word spelling errors

- …leaving in about fifteen ***minuets*** to go to her house.
- The design ***an*** construction of the system…
- Can they ***lave*** him my messages?
- The study was conducted mainly ***be*** John Black.

- 25-40% of spelling errors are real words Kukich 1992

## Context-sensitive spelling error fixing

- For each word in sentence (phrase, query …)
  - Generate *candidate set*
    - the word itself
    - all single-letter edits that are English words
    - words that are homophones
    - (all of this can be pre-computed!)
- Choose best candidates
    - Noisy channel model





## Noisy channel for real-word spell correction

- Given a sentence $w_1,w_2,w_3,\ldots,w_n$
- Generate a set of candidates for each word $w_i$
  - Candidate($w_1$) = {$w_1$, $w'_1$ , $w''_1$ , $w'''_1$ ,…}
  - Candidate($w_2$) = {$w_2$, $w'_2$ , $w''_2$ , $w'''_2$ ,…}
  - Candidate($w_n$) = {$w_n$, $w'_n$ , $w''_n$ , $w'''_n$ ,…}
- Choose the sequence W that maximizes P(W)



## Incorporating context words: Context-sensitive spelling correction

- Determining whether **actress** or **across** is appropriate will require looking at the context of use
- We can do this with a better **language model**
  - You learned/can learn a lot about language models in CS124 or CS224N
  - Here we present just enough to be dangerous/do the assignment
- A **bigram language model** conditions the probability of a word on (just) the previous word

  $P(w_1 \ldots w_n) = P(w_1)P(w_2|w_1)\ldots P(w_n|w_{n-1})$





## Incorporating context words

- For unigram counts, P($w$) is always non-zero
  - if our dictionary is derived from the document collection
- This won't be true of $P(w_k|w_{k-1})$. We need to **smooth**
- We could use add-1 smoothing on this conditional distribution
- But here's a better way – interpolate a unigram and a bigram:

  $P_{li}(w_k|w_{k-1}) = \lambda P_{uni}(w_k) + (1-\lambda)P_{bi}(w_k|w_{k-1})$
  - $P_{bi}(w_k|w_{k-1}) = C(w_{k-1}, w_k) / C(w_{k-1})$





## All the important fine points

- Note that we have several probability distributions for words
  - Keep them straight!
- You might want/need to work with log probabilities:
  - $\log P(w_1 \ldots w_n) = \log P(w_1) + \log P(w_2|w_1) + \ldots + \log P(w_n|w_{n-1})$
  - Otherwise, be very careful about floating point underflow
- Our query may be words anywhere in a document
  - We'll start the bigram estimate of a sequence with a unigram estimate
  - Often, people instead condition on a start-of-sequence symbol, but not good here
  - Because of this, the unigram and bigram counts have different totals – not a problem





## Using a bigram language model

- "a stellar and versatile **acress** whose combination of sass and glamour…"
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- P(actress|versatile)=.000021 P(whose|actress) = .0010
- P(across|versatile) =.000021 P(whose|across) = .000006

- P("versatile actress whose") = .000021*.0010 = 210 x10⁻¹⁰
- P("versatile across whose") = .000021*.000006 = 1 x10⁻¹⁰

## Using a bigram language model

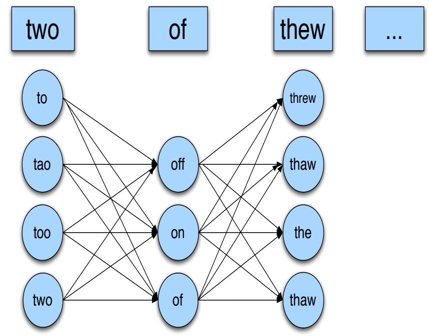
- "a stellar and versatile **acress** whose combination of sass and glamour…"
- Counts from the Corpus of Contemporary American English with add-1 smoothing
- P(actress|versatile)=.000021 P(whose|actress) = .0010
- P(across|versatile) =.000021 P(whose|across) = .000006

- **P("versatile actress whose") = .000021*.0010 = 210 x10$^{-10}$**
- P("versatile across whose") = .000021*.000006 = 1 x10$^{-10}$





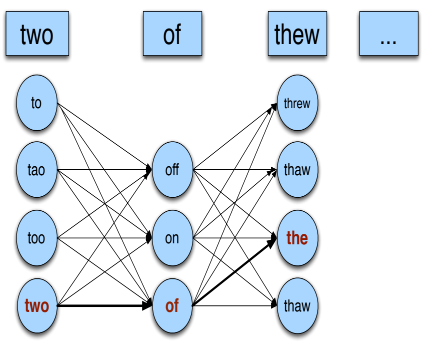
## Noisy channel for real-word spell correction








## Noisy channel for real-word spell correction








## Simplification: One error per sentence

- Out of all possible sentences with one word replaced
  - $w_1$, $\mathbf{w''_2}$, $w_3$, $w_4$ two **off** thew
  - $w_1$, $w_2$, $\mathbf{w'_3}$, $w_4$ two of **the**
  - $\mathbf{w'''_1}$, $w_2$, $w_3$, $w_4$ **too** of thew
  - …
- Choose the sequence W that maximizes P(W)



## Where to get the probabilities

- Language model
  - Unigram
  - Bigram
  - etc.
- Channel model
  - Same as for non-word spelling correction
  - Plus need probability for no error, *P(w|w)*





## Probability of no error

- What is the channel probability for a correctly typed word?
- P("the"|"the")
  - If you have a big corpus, you can estimate this percent correct
- But this value depends strongly on the application
  - .90 (1 error in 10 words)
  - .95 (1 error in 20 words)
  - .99 (1 error in 100 words)

## Peter Norvig's "thew" example

| x | w | x\|w | P(x\|w) | P(w) | $10^9$ P(x\|w)P(w) |
|---|---|---|---|---|---|
| thew | the | ew\|e | 0.000007 | 0.02 | 144 |
| thew | thew | | 0.95 | 0.00000009 | 90 |
| thew | thaw | e\|a | 0.001 | 0.0000007 | 0.7 |
| thew | threw | h\|hr | 0.000008 | 0.000004 | 0.03 |
| thew | thwe | ew\|we | 0.000003 | 0.00000004 | 0.0001 |





## State of the art noisy channel

- We never just multiply the prior and the error model
- Independence assumptions→probabilities not commensurate
- Instead: Weight them

$$\hat{w} = \underset{w \in V}{\operatorname{argmax}}\, P(x \mid w) P(w)^{\lambda}$$

- Learn λ from a development test set





## Improvements to channel model

- Allow richer edits (Brill and Moore 2000)
  - ent→ant
  - ph→f
  - le→al
- Incorporate pronunciation into channel (Toutanova and Moore 2002)
- Incorporate device into channel
  - Not all Android phones need have the same error model
  - But spell correction may be done at the system level