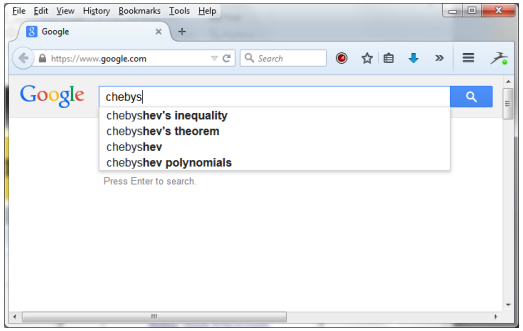
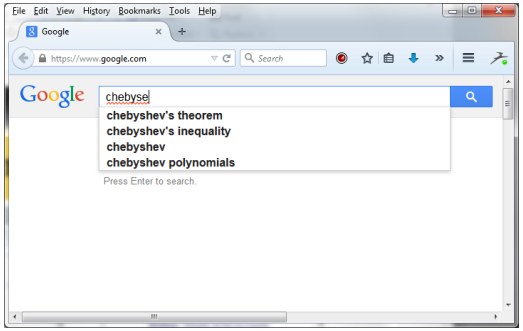
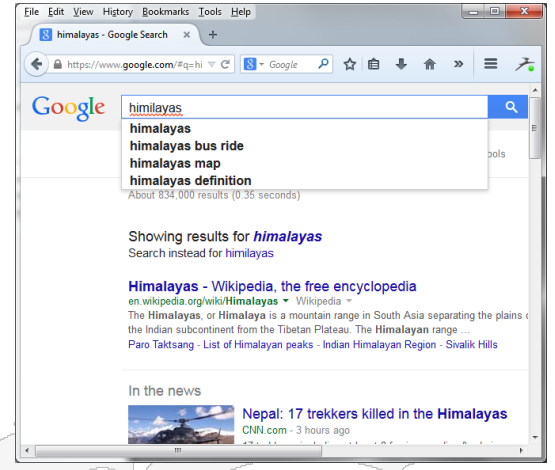
Spell Checking and Correction

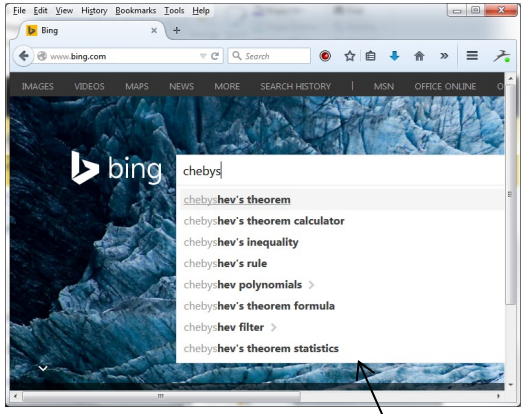
Some Google/Bing Examples

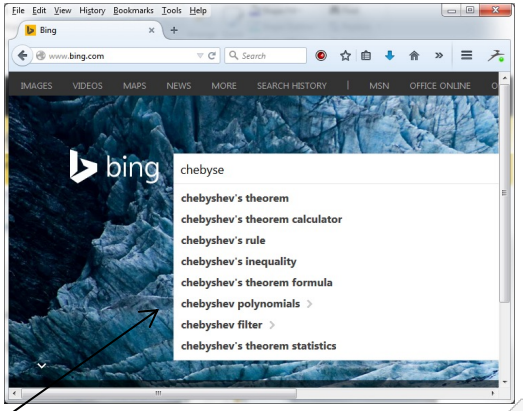


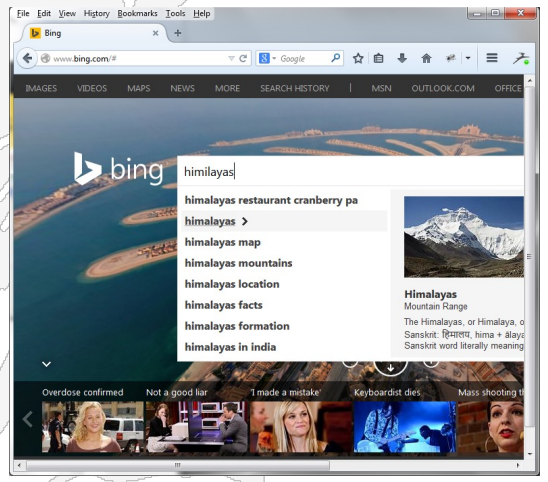


Russian mathematician, notice red underline appears as soon as the first incorrect character is typed





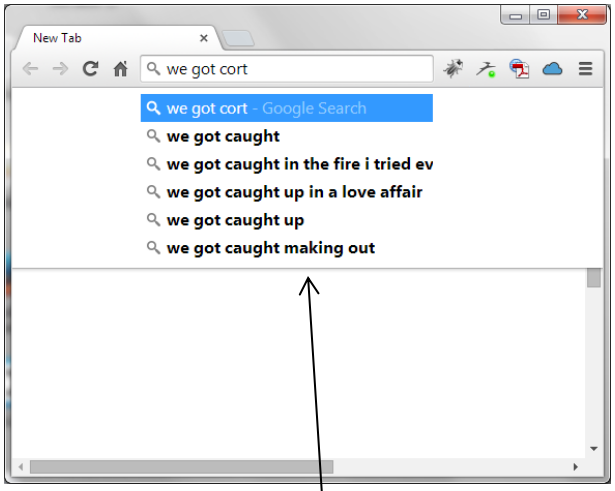




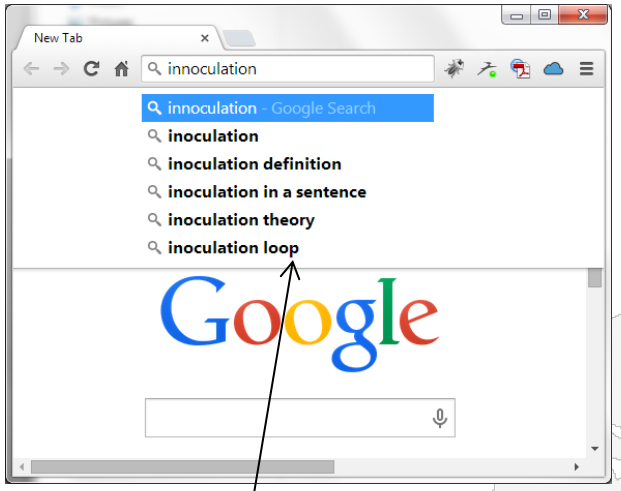
Bing also combines autocomplete with spelling correction but there is no red underline

Himilayas

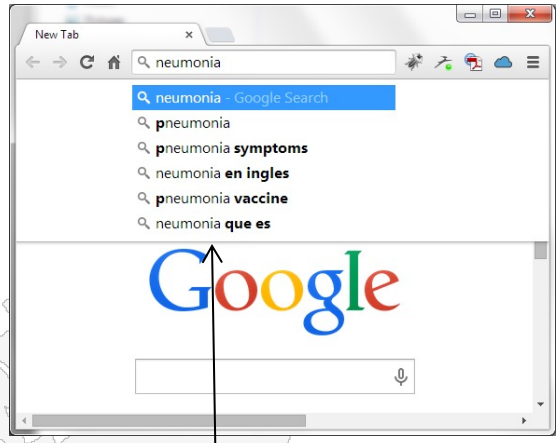
More Examples



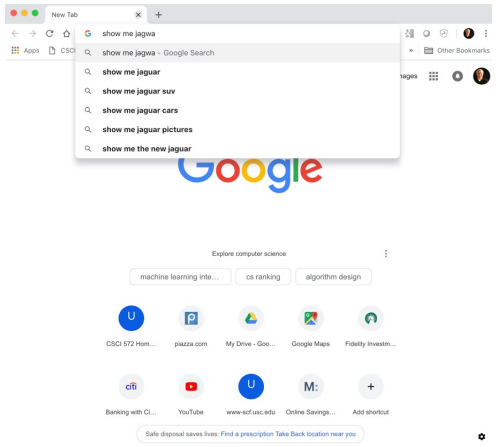
easy for people, but harder for computers to correct, **likely use of n-grams**

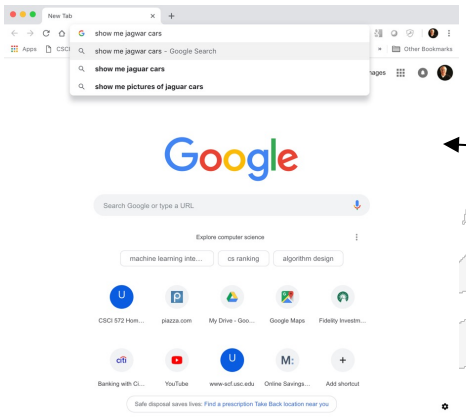


easy for a computer to correct **likely use of a database** of words



computer needs to both identify the error and correct the misspelling

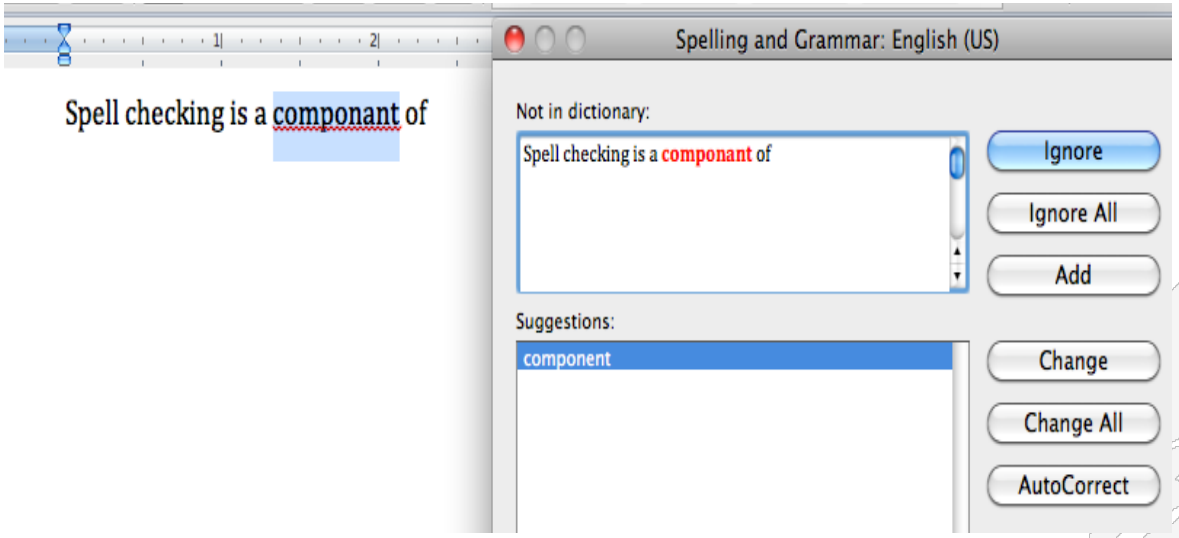




Google combines spelling correction with the **most likely terms** as it comes up with "cars" in autocomplete for the query "jagwa" (misspelled) but leaves the user's misspelling for a while

Spelling Correction is Done in Many Places

1. Word processing



2. Smartphone input

Word processing is the classic application for spelling correction

Word and PowerPoint have mode to auto-correct

- set as the default
- the spell dictionary can be modified

Typing on a virtual keyboard can be doubly difficult (for seniors)

Rates of Spelling Errors

Error rates vary depending upon the application

- **Typing is very error prone, and especially difficult on smartphones**
- **Different studies have produced varying results**

26%: Web queries Wang *et al.* 2003

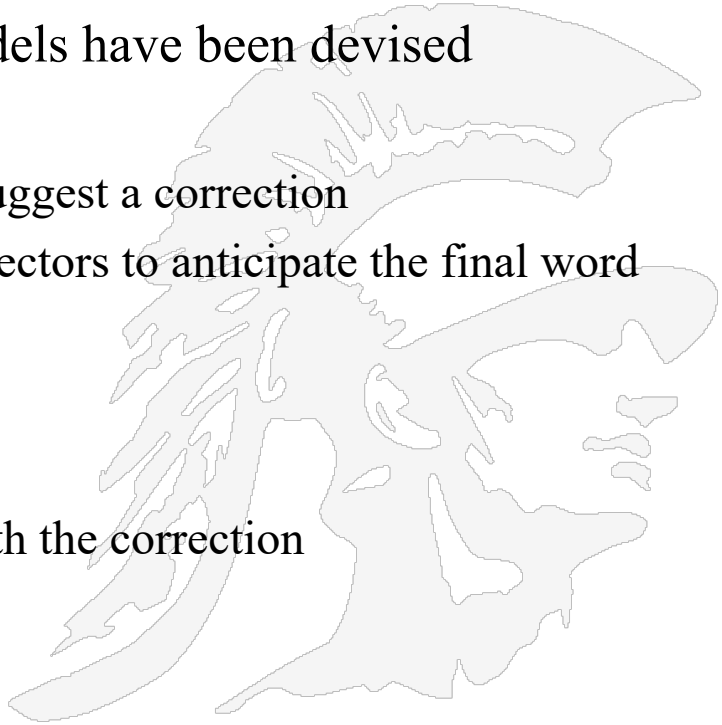
13%: Retyping, no backspace: Whitelaw *et al.* English&German

7%: Words corrected retyping on phone-sized organizer

So seamless spelling correction is an essential component of information retrieval and especially for search engines

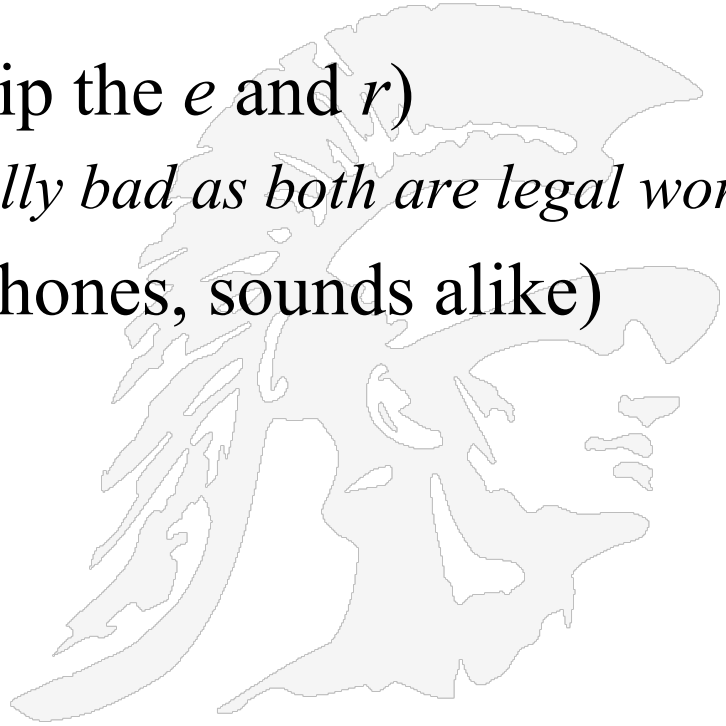
The Two Main Spelling Tasks

1. **Spelling Error *Detection***
   - Obviously we need a big dictionary and the ability to search it quickly
   - Using context may be necessary
     - To do this spelling error models have been devised
2. **Spelling Error *Correction***
   - Web search engines **always** try to suggest a correction
   - Autocomplete requires spelling correctors to anticipate the final word
     - Fast response time is required
   - The two major techniques are
     1. edit distance algorithms or
     2. n-gram matching to come up with the correction



Three Types of Spelling Errors

1. **Non-word errors**
   - *graffe* →*giraffe*
2. **Typographical errors** (flip the *e* and *r*)
   - *three* →*there* *(especially bad as both are legal words)*
3. **Cognitive errors** (homophones, sounds alike)
   - *piece*→*peace*,
   - *too* → *two*
   - *your* →*you're*



Non-Word Spelling Errors

- **Non-word spelling error detection:**
  - Any word not in a *dictionary* is presumed to be an error
  - The larger the dictionary the better
- **Approach to non-word spelling error correction:**
  - Generate candidates from the dictionary that are close to the error
  - **How do we do this?**
    - **Shortest weighted edit distance**
      - **Edit distance** is a way of quantifying how dissimilar two strings (e.g., words) are to one another by counting the minimum number of operations required to transform one string into the other
    - **Highest noisy channel probability**
      - use probabilities to select the most obvious candidates

Causes of Misspellings

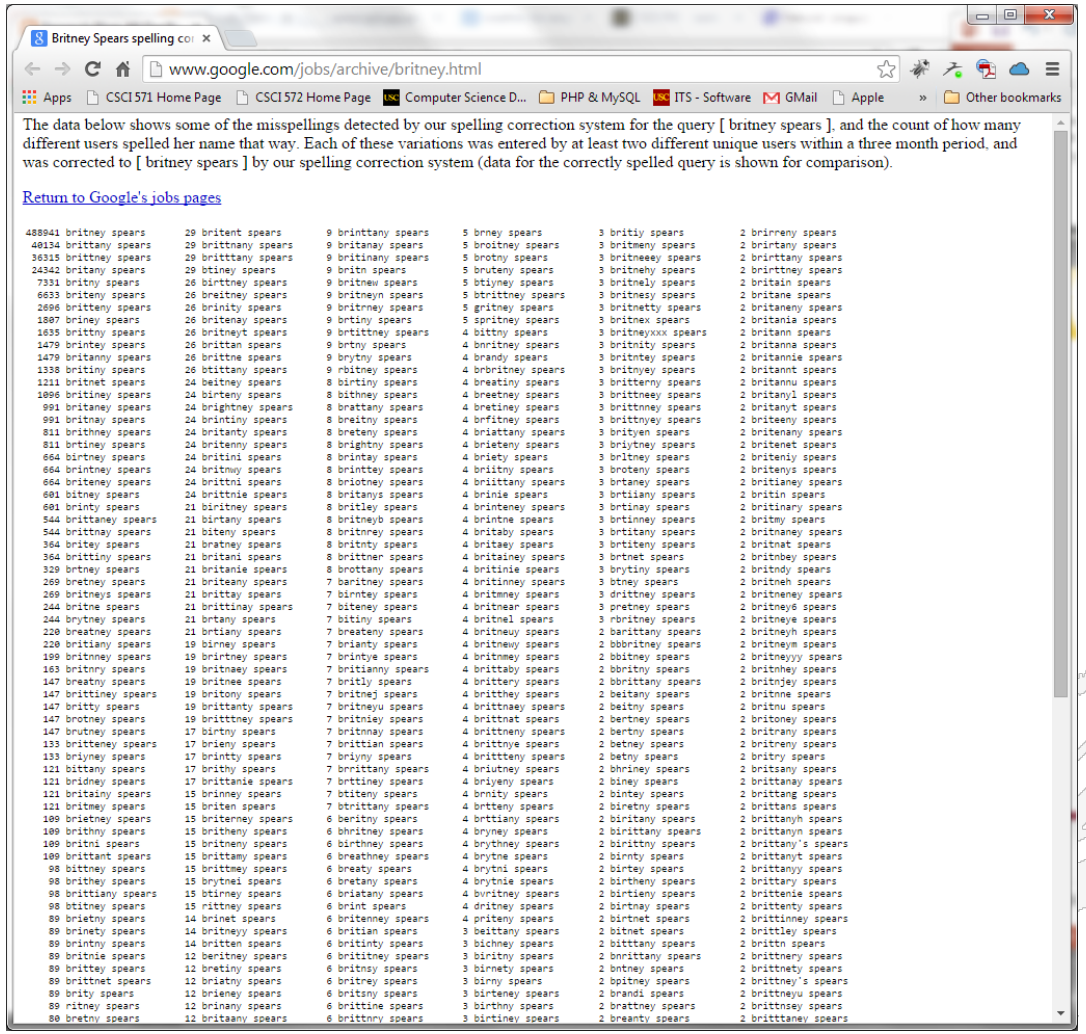
Cause	Misspelling	Correction
typing quickly	exxit mispell	exit misspell
keyboard adjacency	importamt	important
inconsistent rules	conceive concierge	conceive concierge
ambiguous word breaking	silver light	silverlight
new words	kinnect	kinect

e before i
i before e

According to Cucerzan and Brill, **more than 10% of search engine queries are misspelled**
"*Spelling Correction as an iterative process that exploits the collective knowledge of web users*"
http://csci572.com/papers/Cucerzan.pdf (advocates using query logs to guess the correct spelling)

How Many Error Variations of a Word May Actually Appear

http://www.netpaths.net/blog/britney-spears-spelling-variations/



Britney Spears spelling cor

www.google.com/jobs/archive/britney.html

Apps | CSCI 571 Home Page | CSCI 572 Home Page | Computer Science D... | PHP & MySQL | ITS - Software | GMail | Apple | Other bookmarks

The data below shows some of the misspellings detected by our spelling correction system for the query [britney spears], and the count of how many different users spelled her name that way. Each of these variations was entered by at least two different unique users within a three month period, and was corrected to [britney spears] by our spelling correction system (data for the correctly spelled query is shown for comparison).

Return to Google's jobs pages

488941 britney spears	29 britent spears	9 brinttany spears	5 brney spears	3 britiy spears	2 brirreny spears
40134 brittany spears	29 brittnany spears	9 britanay spears	5 broitney spears	3 britmeny spears	2 brirtany spears
36315 brittney spears	29 britttany spears	9 britinany spears	5 brotny spears	3 britneeey spears	2 brirttney spears
24342 britany spears	29 btiney spears	9 britn spears	5 bruteny spears	3 britnehy spears	2 brirttney spears
7331 britny spears	26 birttney spears	9 britnew spears	5 btiyney spears	3 britnely spears	2 britain spears
6633 briteny spears	26 breitney spears	9 britneyn spears	5 btrittney spears	3 britnesy spears	2 britane spears
2690 britteny spears	26 brinity spears	9 britrney spears	5 gritney spears	3 britnetty spears	2 britaney spears
1807 briney spears	26 britenay spears	9 britny spears	5 spritney spears	3 britnex spears	2 britania spears
1635 brittny spears	26 britneyt spears	9 brtittney spears	4 bittny spears	3 britneyxxx spears	2 britann spears
1479 brintey spears	26 brittan spears	9 brtny spears	4 bnritney spears	3 britnity spears	2 britanna spears
1479 britanny spears	26 brittne spears	9 brytny spears	4 brandy spears	3 britntey spears	2 britannie spears
1338 britiny spears	26 brittany spears	9 rbitney spears	4 brbritney spears	3 britnyey spears	2 britannt spears
1211 britnet spears	24 beitney spears	8 birtiny spears	4 breatiny spears	3 britterny spears	2 britannu spears
1096 britiney spears	24 birteny spears	8 bithney spears	4 breetney spears	3 brittneey spears	2 britanyl spears
991 britaney spears	24 brightney spears	8 brattany spears	4 bretiney spears	3 brittnney spears	2 britanyt spears
991 britnay spears	24 brintiny spears	8 breitny spears	4 brfitney spears	3 brittnyey spears	2 briteeny spears
811 brittney spears	24 britanty spears	8 bretany spears	4 briattany spears	3 brityen spears	2 britenany spears
811 brtiney spears	24 britenny spears	8 brightny spears	4 brieteny spears	3 briytney spears	2 britenet spears
664 birtney spears	24 britini spears	8 brintay spears	4 briety spears	3 brltney spears	2 briteniy spears
664 brintney spears	24 britnwy spears	8 brinttey spears	4 briitny spears	3 broteny spears	2 britenys spears
664 briteney spears	24 brittni spears	8 briotney spears	4 briittany spears	3 brtaney spears	2 britianey spears
601 bitney spears	24 brittnie spears	8 britanys spears	4 brinie spears	3 brtiiany spears	2 britin spears
601 brinty spears	21 biritney spears	8 britley spears	4 brinteney spears	3 brtinay spears	2 britinary spears
544 brittaney spears	21 birtany spears	8 britneyb spears	4 brintne spears	3 brtinney spears	2 britmy spears
544 brittnay spears	21 biteny spears	8 britnrey spears	4 britaby spears	3 brtitany spears	2 britnaney spears
364 britey spears	21 bratney spears	8 britnty spears	4 britbay spears	3 brtitney spears	2 britnat spears
364 brittiny spears	21 britani spears	8 brittner spears	4 britainey spears	3 brtnet spears	2 britnbey spears
329 brtney spears	21 britanie spears	8 brottany spears	4 britinie spears	3 brytiny spears	2 britndy spears
269 bretney spears	21 briteany spears	7 baritney spears	4 britinney spears	3 btney spears	2 britneh spears
269 britneys spears	21 brittay spears	7 birntey spears	4 britmey spears	3 drittney spears	2 britneney spears
244 britne spears	21 brittinay spears	7 biteney spears	4 britnear spears	3 pretney spears	2 britney6 spears
244 brytney spears	21 brtany spears	7 bitiny spears	4 britnel spears	3 rbritney spears	2 britneye spears
220 breatney spears	21 brtiany spears	7 breateny spears	4 britneuy spears	2 barittany spears	2 britneyh spears
220 britiany spears	19 birney spears	7 brianty spears	4 britnewy spears	2 bbbritney spears	2 britneym spears
199 britney spears	19 biritney spears	7 brintye spears	4 britnney spears	2 bbitney spears	2 britneyyy spears
163 britnry spears	19 britnaey spears	7 britiiny spears	4 brittaby spears	2 bbritny spears	2 britnhey spears
147 breatny spears	19 britnee spears	7 britly spears	4 brittery spears	2 bbrittany spears	2 britnjey spears
147 brittiney spears	19 britony spears	7 britnej spears	4 britthey spears	2 beitany spears	2 britnne spears
147 britty spears	19 brittanty spears	7 britneyu spears	4 brittnaey spears	2 beitny spears	2 britnu spears
147 brotney spears	19 britttney spears	7 britniey spears	4 brittnat spears	2 bertney spears	2 britoney spears
147 brutney spears	17 birtny spears	7 britnnay spears	4 brittneny spears	2 bertny spears	2 britrany spears
133 britteney spears	17 brieny spears	7 brittian spears	4 brittnye spears	2 betney spears	2 britreny spears
133 briyney spears	17 brintty spears	7 briyny spears	4 brittteny spears	2 betny spears	2 britry spears
121 bittney spears	17 britny spears	7 brittany spears	4 britutney spears	2 bhntney spears	2 britsany spears
121 bridney spears	17 brittanie spears	7 brttiney spears	4 briyeny spears	2 biney spears	2 brittanay spears
121 britainy spears	15 brinney spears	7 bttiny spears	4 brnity spears	2 bintey spears	2 brittang spears
121 britney spears	15 briten spears	7 btrittany spears	4 brtteny spears	2 biretny spears	2 brittans spears
109 brietney spears	15 britterney spears	6 beritny spears	4 brttiany spears	2 biritany spears	2 brittanyh spears
109 brithny spears	15 britheny spears	6 bhritney spears	4 bryney spears	2 birittany spears	2 brittanyn spears
109 britni spears	15 britneny spears	6 birthney spears	4 brythney spears	2 birittny spears	2 brittany's spears
109 brittant spears	15 brittamy spears	6 breathney spears	4 brytne spears	2 birnty spears	2 brittanyt spears
98 bittney spears	15 brittney spears	6 breaty spears	4 brytni spears	2 birtey spears	2 brittanyy spears
98 brithey spears	15 brytnel spears	6 bretany spears	4 brytnie spears	2 birtheny spears	2 brittary spears
98 brittany spears	15 btirney spears	6 britaany spears	4 brtritney spears	2 birtieny spears	2 brittenie spears
98 btitney spears	15 rittney spears	6 brint spears	4 dritney spears	2 birtnay spears	2 brittenty spears
89 brietny spears	14 brinet spears	6 britenney spears	4 pritany spears	2 birtnet spears	2 brittinney spears
89 brinety spears	14 britneyy spears	6 britian spears	3 beittany spears	2 bitnet spears	2 brittley spears
89 brintny spears	14 britten spears	6 britinty spears	3 bichney spears	2 bitttany spears	2 brittn spears
89 britnie spears	12 beritney spears	6 britittney spears	3 biritny spears	2 bnrittany spears	2 brittnery spears
89 brittey spears	12 bretiny spears	6 britnsy spears	3 birnety spears	2 bntney spears	2 brittnety spears
89 brittnet spears	12 briatny spears	6 britrey spears	3 birny spears	2 bpitney spears	2 brittney's spears
89 britty spears	12 brieney spears	6 britzny spears	3 birteney spears	2 brandi spears	2 brittneyu spears
89 ritney spears	12 brinany spears	6 britzine spears	3 birtiny spears	2 brattney spears	2 brittnsey spears
80 bretny spears	12 britaany spears	6 brittnry spears	3 birtiney spears	2 breanty spears	2 britttaney spears

Google's list of spelling errors for "Britney Spears" includes a count of how many different users spelled her name in various way, e.g. 488,941 people used the correct spelling "britney spears" while 40,134 used "britnay spears"

there are 593 different variations that actually occurred

Spelling Errors Needing Context

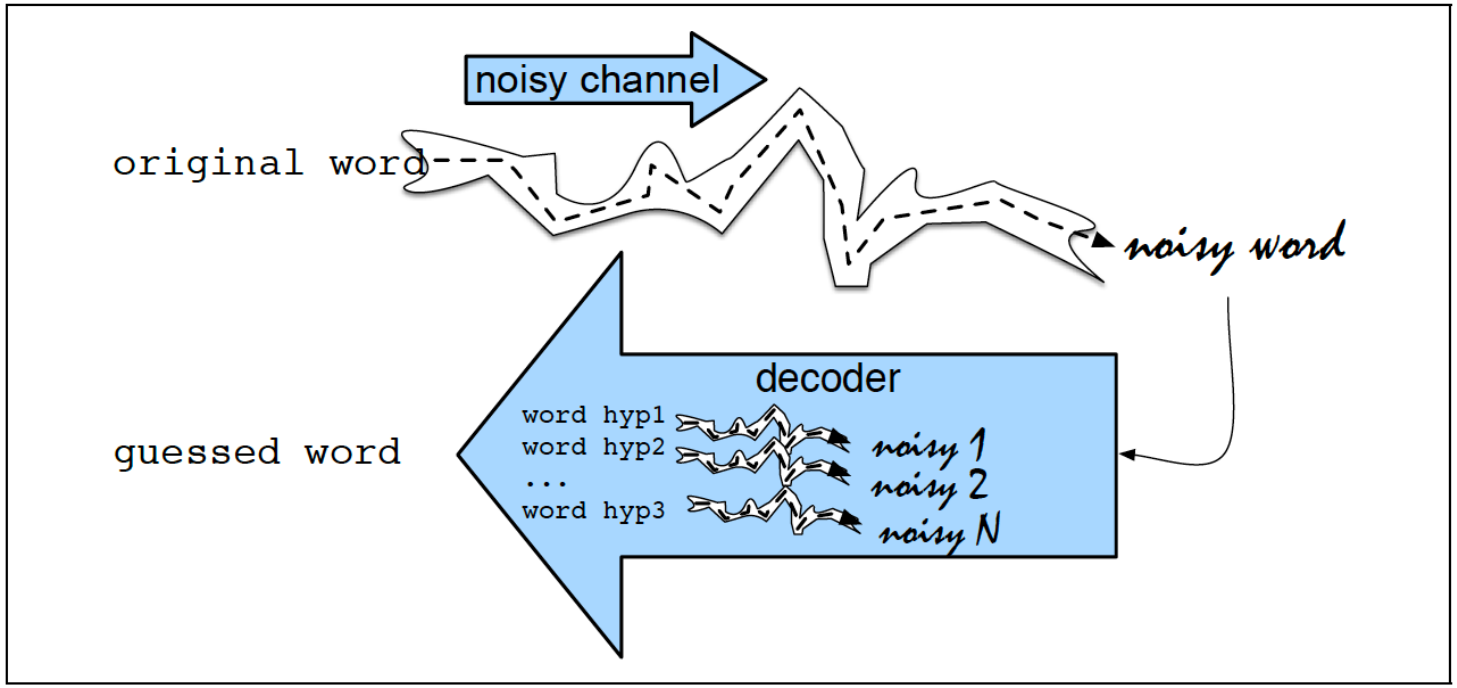
- **Some misspellings require context to disambiguate**
  1. **consider whether the surrounding words "make sense" for your candidate set, e.g.**
     - *Flying form Heathrow to LAX → Flying from Heathrow to LAX*
     - *Power crd → power cord versus Video crd → video card*
  2. **For candidate words with similar sounds but different spellings and different meanings, context is needed**
     - *e.g. there, their*
     - *N-grams are most useful here*
  3. **To resolve the above, candidate words must be found with similar *pronunciations***
     - *use the Soundex algorithm*

More Challenges for Identifying Spelling Errors

- **Some additional challenges**
  4. **Allow for insertion of a space or hyphen**
     - `thisidea → this idea`
     - `inlaw → in-law`
     - `chat inspanich → chat in spanish` `(2 corrections)`
  5. **Allow for deletion of a space**
     - `power point slides → powerpoint slides`
  6. **Watch out for words NOT in the Lexicon that are valid, e.g.**
     - amd processors
       - AMD is a company that makes processors for laptops
       - Another example where context is needed

The Noisy Channel Model

- This model suggests treating the misspelled word as if a correctly spelled word has been distorted by being passed through a noisy communication channel
- Noise in this case refers to substitutions, insertions or deletions of letters



Bayesian Inference Implies We Can Use Previous Combinations to Predict the Correct Word

- We see an observation *x* (a misspelled word) and our job is to find the correct word *w* that generated this misspelled word
- Out of all possible words in the vocabulary *V* we want to find the word *w* such that the probability that w is the correct word, *P(w|x)*, is highest. We use the hat notation ˆ to mean "our estimate of the correct word". We state this in the following way:

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

- Out of all words in the vocabulary, we want the particular word that maximizes the right-hand side *P(w|x)*
- ***Bayes Rule:***

$$P(a|b) = \frac{P(b|a)P(a)}{P(b)}$$

The probability of *a* given *b* is equal to the probability of *b* given *a*, times the probability of *a*, divided by the probability of *b*

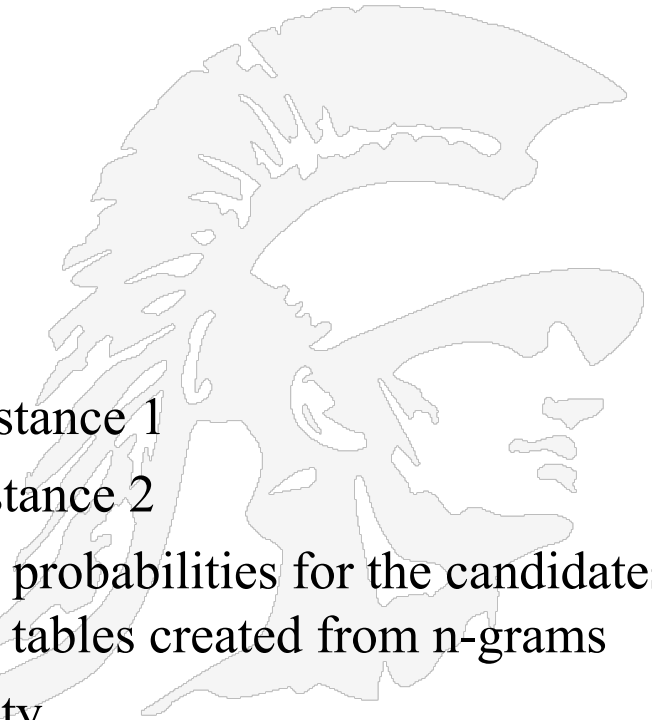
implies $\hat{w} = \underset{w \in V}{\operatorname{argmax}} \frac{P(x|w)P(w)}{P(x)}$ and simplifying gives $\hat{w} = \underset{w \in V}{\operatorname{argmax}} P(x|w)\,P(w)$

Note: *P(x|w)* is a constant so we can ignore it

Conclusion: we need a table of probabilities

The Basic Spelling Correction Algorithm

1. **Initial step**: Create a dictionary and encode it for fast retrieval
2. When a query is submitted, the spell checker examines each word and for words not in the dictionary looks for possible character edits, namely
   - insertions,
   - deletions,
   - substitutions, and occasionally
   - transpositions
   - **Observation:**
     - 80% of errors are within edit distance 1
     - Almost all errors within edit distance 2
3. Take the output of step 2 and compute probabilities for the candidates using previously identified probability tables created from n-grams
4. Select the result with highest probability



The Basic Spelling Correction Algorithm Refined

- **Edit distance** is a way of quantifying how dissimilar two strings (e.g. words) are to one another by counting the minimum number of operations required to transform one string into the other
  - different algorithms assume slightly different operations
  - e.g., Levenshtein uses: removal, insertion, substitution of a character

1. **Check each query term against the dictionary**
2. **For each term NOT found in the dictionary, generate all words within edit distance ≤ *k* (e.g., *k* = 1 or 2) and then intersect them with dictionary**
   - Compute them fast with a Levenshtein algorithm
3. **Use a character *n*-gram index and find dictionary words that share "most" *n*-grams with word**
4. **Select the word with the highest probability** (largest n-gram count)
5. **For speed, have a pre-computed map of words to possible corrections**

Use Edit Distance To Produce Candidate Corrections

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

Six words within 1 of acress

Context check is necessary to choose the appropriate word

(try this yourself in Google/Bing)

For the word "acress" there are six dictionary words all within edit distance 1

Now Apply Probabilities

- **We now need to compute the prior probability of each occurrence**
- **We can do this using unigrams, bigrams, trigrams, etc**
- **Using the Corpus of Contemporary English, 404,253,213 words we get the following**
- ***Across* is the most likely choice, followed by**

For fun try:
I need acress to . . .
I love the acress . . .
a kiss and a acress . . .

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

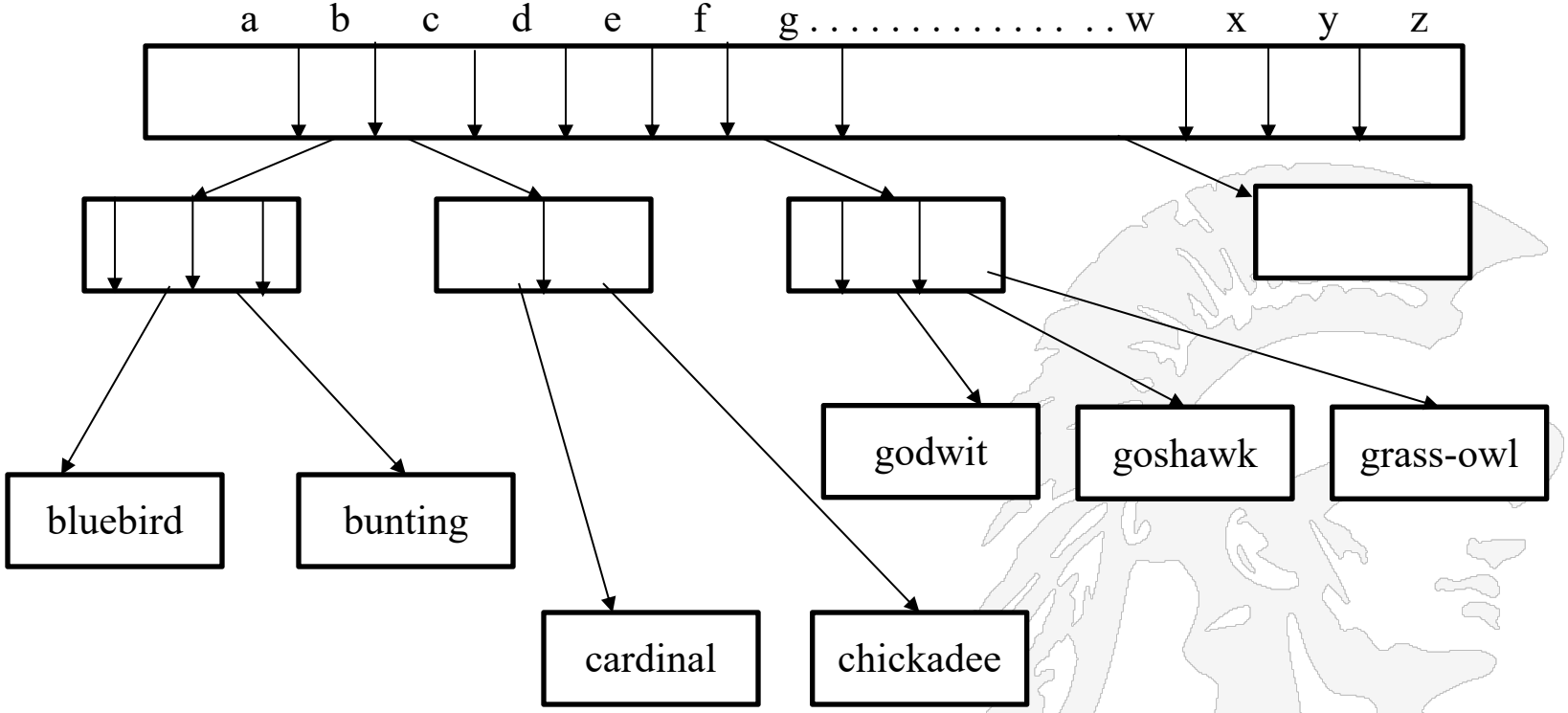
"***across***" is the most likely correction →

The Spelling Correction Dictionary and Autocomplete

- **The search for corrections is carried out from left-to-right**
  - At each point, a partial hypothesis is expanded with every character which could follow the partial hypothesis and lead to one of the known words (the user input is always allowed as an output hypothesis).
  - Thus the branching factor controls the amount of time required to search for spelling corrections.
- **The terms of the lexicon must be stored in a data structure that affords efficient *prefix matching***
  - Often a trie data structure is used

The Spelling Correction Dictionary Example of a Trie

- **a prefix tree (sometimes called a trie from the word re**trie**val) is a tree of degree >=2 in which the branching at any level is determined by a portion of the key**



branch nodes take you down the tree to element nodes; At any stage one is pointing at all keyword matches that contain the same prefix; Computing time for retrieval is *O(m)* where *m* is the length of the string, at the expense of increased storage

The Spelling Correction Dictionary Error Test Sets

- To enhance a lexicon one can include a table of common misspellings
- there are many possible spelling error test sets, e.g.
  - Wikipedia's list of common English misspelling
    - https://en.wikipedia.org/wiki/Wikipedia:Lists_of_common_misspellings
  - Aspell filtered version of that list
    - http://aspell.net/ is the spell program
  - Birkbeck spelling error corpus
    - https://ota.bodleian.ox.ac.uk/repository/xmlui/handle/20.500.12024/0643, 1.8MBs
- These sets are primarily used to correct typographical errors

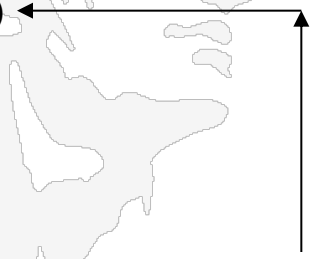
Using N-Grams For Spelling Correction

- An ***n*-gram model** is a type of probabilistic language model for predicting the next item in a sequence
- Two benefits of *n*-gram models (and algorithms that use them) are simplicity and scalability – with larger *n*, a model can store more context with a well-understood space–time tradeoff, enabling small experiments to scale up efficiently

- **Sample of 3-gram sequences**
- ceramics collectables fine (130)
- ceramics collected by (52)
- ceramics collectible pottery (50)
- ceramics collectibles cooking (45)

Sample of 4-gram sequences and # of times appeared

serve as the incoming (92)
serve as the incubator (99)
serve as the independent (794)
serve as the index (223)
serve as the indication (72)
serve as the indicator (120)



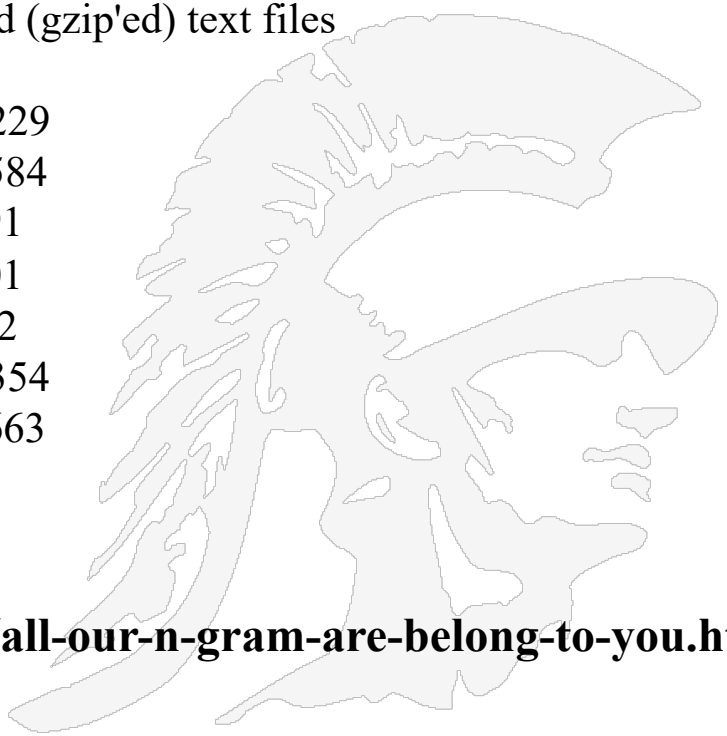
a query such as "serve as the indapendant" would match the above

Google's N-Gram Data

- Google has collected and uses a great deal of N-gram data
- Google is using the Linguistics Data Consortium to distribute more than one trillion words they have extracted from public web pages
- Below is a statistical summary of the data they are distributing

File sizes: approx. 24 GB compressed (gzip'ed) text files

Number of tokens:	1,024,908,267,229
Number of sentences:	95,119,665,584
Number of unigrams:	13,588,391
Number of bigrams:	314,843,401
Number of trigrams:	977,069,902
Number of fourgrams:	1,313,818,354
Number of fivegrams:	1,176,470,663



http://googleresearch.blogspot.com/2006/08/all-our-n-gram-are-belong-to-you.html

Handling N-Gram Data

- **Efficiency considerations are important when building language models that use such large sets of N-grams.**
- **Rather than store each word as a string, it is generally represented in memory as a 64-bit hash number, with the words themselves stored on disk.**
- **Probabilities are generally quantized using only 4-8 bits (instead of 8-byte floats), and N-grams are stored in reverse tries.**
- **N-grams can also be shrunk by pruning, for example only storing N-grams with counts greater than some threshold (such as the count threshold of 40 used for the Google N-gram release)**

- See *Handling Massive N-Gram Datasets Efficiently*, G. Pibiri and R. Venturini which improves upon *Elias-Fano encoding*, https://www.antoniomallia.it/sorted-integers-compression-with-elias-fano-encoding.html

Applying the N-Gram Model to Spelling Correction

- Let's say you are using 4-grams to calculate the probability of a next word in text.
  - You have "this is a very" followed by "sunny".
  - But assume "sunny" does not occur in the n-gram set that includes "this is a very"
  - *Implication*: so for the 4-gram model "sunny" has probability 0,

- Back-off means you go back to a *n-1*-gram level to calculate the probabilities when you encounter a word with probability = 0.
  - So in the above you will use a 3-gram model to calculate the probability of "sunny" in the context "is a very".

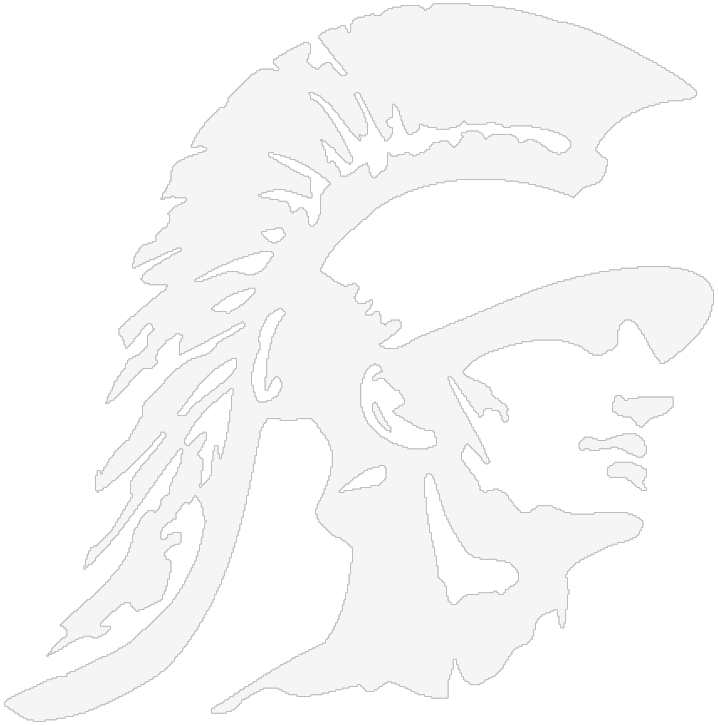
- Whenever you go back one level you multiply the odds by an empirically derived number, in this case 0.4. So if sunny exists in the 3-gram model the probability would be
  *0.4 * P("sunny"|"is a very")*

Generalizing, this leads to an algorithm for handling n-grams to do spelling correction, called *The Stupid Back-off Algorithm*

1. If a higher-order N-gram has a zero count, we simply backoff to a lower order N-gram, weighed by a fixed (context-independent) weight
2. The backoff terminates in the unigram

- **See Wikipedia's description of the Back-off Model**

A Complete Spelling Correction Program



Peter Norvig's Spelling Corrector written in Python

```
import re, collections
def words(text): return re.findall('[a-z]+', text.lower())
def train(features):
    model = collections.defaultdict(lambda: 1)
    for f in features:
        model[f] += 1
    return model
NWORDS = train(words(file('big.txt').read()))
alphabet = 'abcdefghijklmnopqrstuvwxyz'
def edits1(word):
   splits     = [(word[:i], word[i:]) for i in range(len(word) + 1)]
   deletes    = [a + b[1:] for a, b in splits if b]
   transposes = [a + b[1] + b[0] + b[2:] for a, b in splits if len(b)>1]
   replaces   = [a + c + b[1:] for a, b in splits for c in alphabet if b]
   inserts    = [a + c + b     for a, b in splits for c in alphabet]
   return set(deletes + transposes + replaces + inserts)

def known_edits2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1) if e2 in NWORDS)

def known(words): return set(w for w in words if w in NWORDS)

def correct(word):
    candidates = known([word]) or known(edits1(word)) or known_edits2(word) or [word]
    return max(candidates, key=NWORDS.get)
```

Correct takes a word and returns a likely correct form of the word:

```
correct('speling')
Spelling
correct('korrecter')
corrector
```

http://norvig.com/spell-correct.html

How the Program Works

- The algorithm and its description can be found at
- http://norvig.com/spell-correct.html
- The file big.txt contains a million words
  - It includes text of books from Project Gutenberg, Wiktionary, British National Corpus
- Extract the individual words (function `words` converts everything to lower case)
- *The* and *the* are the same, but "don't" is seen as "don" and "t"
- The program counts how many times each word occurs using function `train`
- `NWORDS[w]` holds a count of how many times the word w has been seen
- For words that are not in our set we set their occurrence to a default, non-zero value of 1 using the Python hash table statement `collections.defaultdict`
- Edit distance is the number of changes it would take to turn one word into another
- An edit can be one of: {deletion, transposition, alteration, insertion}
- The 7 lines that end in `return set(deletes + transposes + replaces + inserts)` is a function that returns a set of all words c that are one edit away from w
  - This can be a large set
- **The literature on spelling correction claims that 80-95% of spelling errors are an edit distance of 1 from the target**

How the Program Works (cont'd)

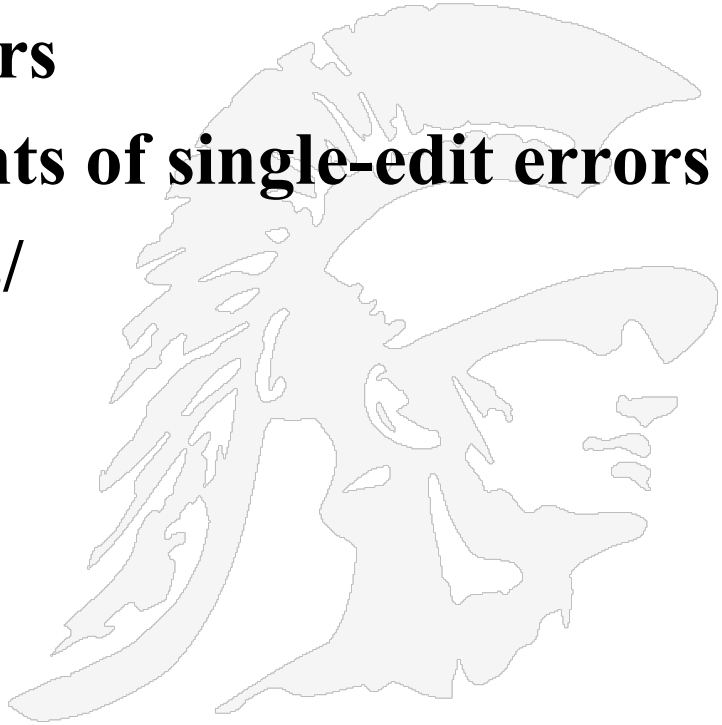
- To compute edit distance 2, just apply edits1 to all the results of edits1

```
Def edit2(word):
    return set(e2 for e1 in edits1(word) for e2 in edits1(e1))
```

- The function known_edits2 removes words that are not in our original set and greatly reduces the size of edits2
  - E.g. all edit 2 combinations for "something" produces 114,324 words, but pruning leaves only 4 words: smoothing, seething, something, and soothing
- Now he assumes his probability model, which is "all known words of edit distance 1 are infinitely more probable than known words of edit distance 2
- He defines function `correct` which chooses as the set of candidate words the set with the shortest edit distance to the original word as long as the set has some known words
- See his web page for an evaluation of his program

Natural Language Corpus Data

- Peter Norvig has provided a web page that is full of useful data for a spelling corrector
  - http://norvig.com/spell-correct.html
- Peter Norvig's list of errors
- Peter Norvig's list of counts of single-edit errors
- http://norvig.com/ngrams/



Some References

- ***How Difficult is it to Develop a Perfect Spell-checker? A Cross-linguistic Analysis through Complex Network Approach*, Monojit Choudhury1, Markose Thomas2, Animesh Mukherjee1, Anupam Basu1, and Niloy Ganguly1**

http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=52A3B869596656C9DA285DCE83A0339F?doi=10.1.1.146.4390&rep=rep1&type=pdf

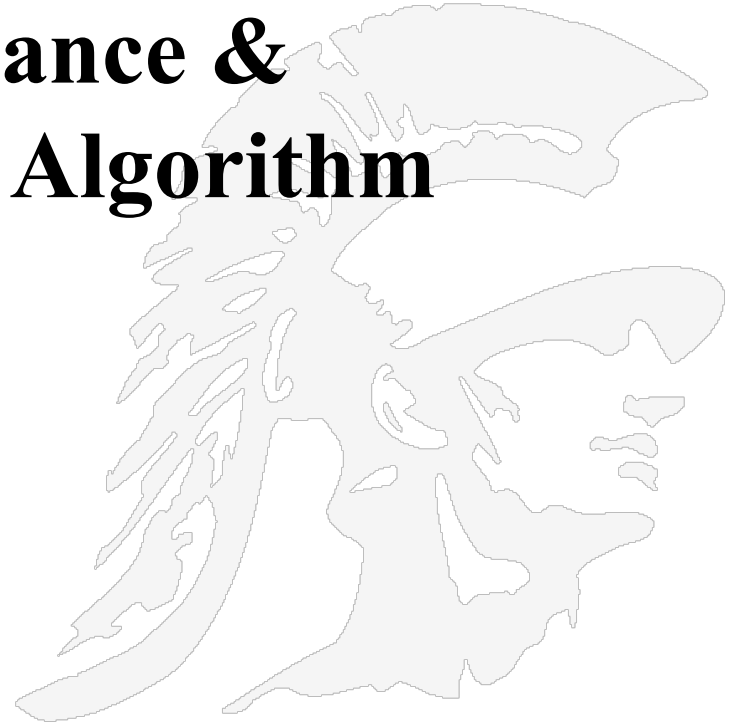
- ***Using the web for language independent spellchecking and autocorrection* by C. Whitelaw et al Proc. 2009 Conf. on Empirical Methods in Natural Language Processing, pp890-899**

http://static.googleusercontent.com/external_content/untrusted_dlcp/research.google.com/en/us/pubs/archive/36180.pdf

Spell Checking by Computer, by Roger Mitton,

http://www.dcs.bbk.ac.uk/~roger/spellchecking.html

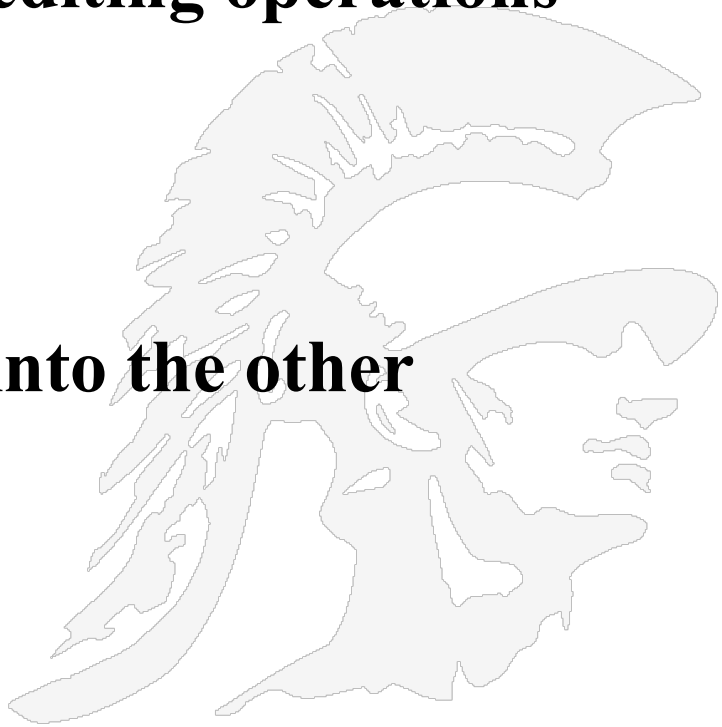
Edit Distance & Levenshtein Algorithm



Edit Distance

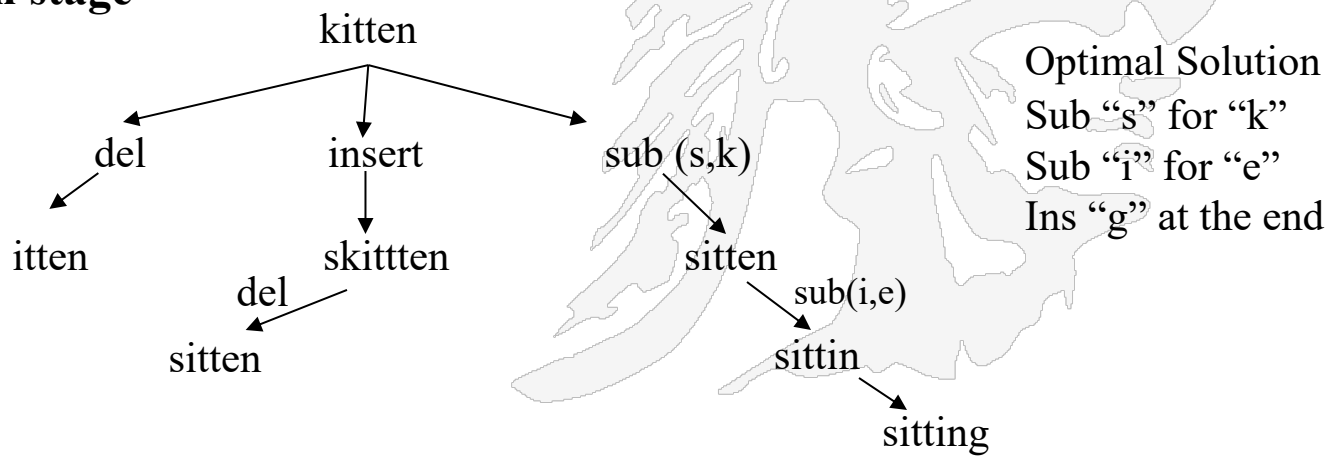
- the minimum edit distance between two strings is the minimum number of editing operations
  - insertion
  - deletion
  - substitution

  needed to transform one into the other



How to Find the Minimum Edit Distance

- Searching for a path (sequence of edits) from the start string to the final string
  - initial state: word we're transforming (e.g. kitten)
  - operators: insert, delete, substitute
  - goal state: the word we're trying to get to (e.g. sitting)
  - path cost: what we want to minimize, the number of edits
- If we blindly generate all possible paths in an effort to produce the goal state, our algorithm will take exponentially long
- But we realize that we needn't do that, we may only follow the path that is optimal at each stage



Dynamic Programming for Minimum Edit Distance

- **Dynamic programming: a technique for solving optimization problems by combining solutions to sub-optimal problems bottom-up**
- **Given two strings: *X* of length *n* and *Y* of length *m*, define**
- ***d(i, j)* as**
  - **the minimum edit distance between *X[1..i]* and *Y[1..j]***
    - **i.e. the first *i* characters of *X* and the first *j* characters of *Y***
  - **Then the minimum edit distance between X and Y is thus *D(n,m)***
- **The dynamic programming algorithm will**
  - **compute *D(i,j)* for small *i, j***
  - **and compute larger *D(i,j)* based on previously computed smaller values**
  - **i.e. compute *D(i,j)* for *all i* $(0 < i < n)$ and *j* $(0 < j < m)$**

- **Note**: the dynamic programming technique was invented by Richard Bellman while at the Rand Corp, later a professor at USC

Pseudocode Implementation of Levenshtein Distance

```
function LevenshteinDistance(char s[1..m], char t[1..n]):
//for all i and j, d[i,j] will hold the Levenshtein distance between
//the first i characters of s and the first j characters of t
declare int d[0..m, 0..n]
//Set each element in d to zero
for i from 1 to m, j from 1 to n: d[i,j] := 0
Starting with empty character source and target can be easily produced by inserts
for i from 1 to m: d[i,0] := i
for j from 1 to n: d[0,j] := j
//main loop
for j from 1 to n:
        for i from 1 to m:
                if s[i] = t[j] then substitutionCost := 0 else substitutionCost := 1
                d[i,j] := min (d[i-1,j] + 1,                    // deletion
                               d[i,j-1] + 1,                    //insertion
                              d[i-1, j-1] + substitutionCost  //substitution
                             )
return d[m,n]
```

Levenshtein Example

- **Consider the strings: sitting, kitten**
- **Here is the initial matrix**

	#	K	I	T	T	E	N
#	0	1	2	3	4	5	6
S	1						
I	2						
T	3						
T	4						
I	5						
N	6						
G	7						

is the null string
Looking at the first column, to go from # to S requires 1 insert; to go from # to SI requires 2 inserts; to go from # to SIT requires 3 inserts;
Similarly for the first row, to go from # to K requires 1 insert, etc.

Levenshtein Allows For Insertion, Deletion and Substitution

- **To get the value in the 1,1 position use the formula:**

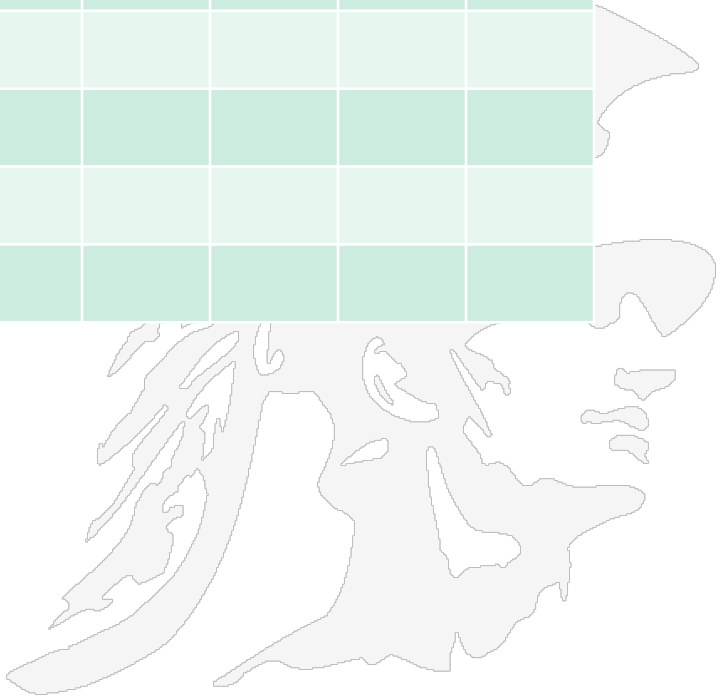
d(i , j) = min { d(i-1,j) + 1, d(i, j-1) + 1, d(i-1, j-1) + 1 if chars not equal, 0 otherwise}

We can fill in the second row and the second column observing that at each step there is a single deletion followed by insertions

	#	K	I	T	T	E	N
#	0	1	2	3	4	5	6
S	1	1	2	3	4	5	6
I	2	2	1	2	3	4	5
T	3	3	2	1	2	3	4
T	4	4	3	2	1	2	3
I	5	5	4	3	2	2	3
N	6	6	5	4	3	3	2
G	7	7	6	5	4	4	3

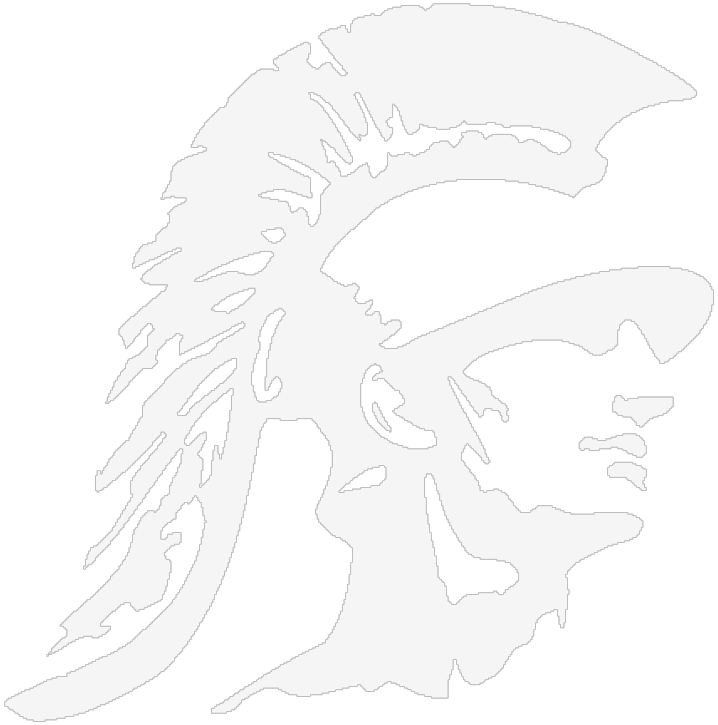
Levenshtein Example 2

	#	S	A	T	U	R	D	A	Y
#	0	1	2	3	4	5	6	7	8
S	1	0	1	2	3	4	5	6	7
U	2	1	1	2	2				
N	3								
D	4								
A	5								
Y	6								



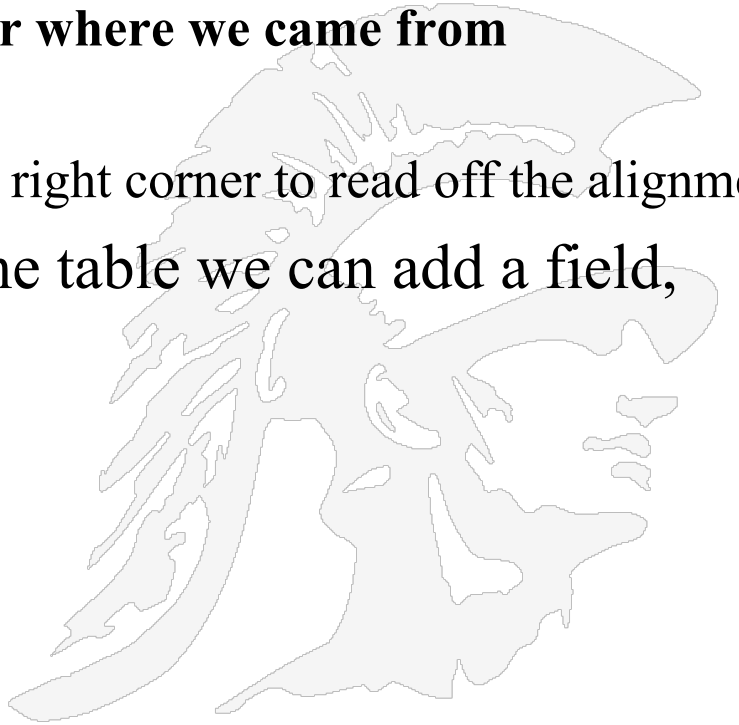
Performance

- **Time: *O(nm)***
- **Space: *O(nm)***
- **Backtrace: *O(n + m)***



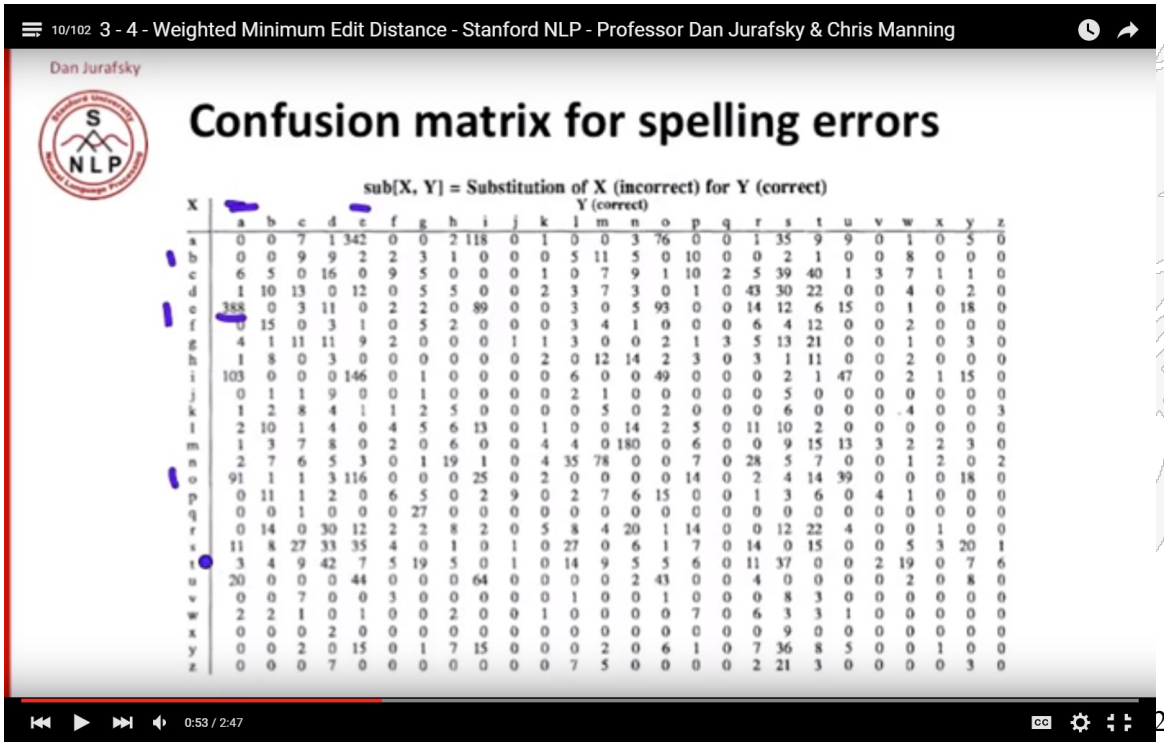
Computing Alignments

- **Edit distance isn't sufficient**
  - we often need to align each character of the two strings to each other
- **we do this by keeping a "backtrace"**
- **every time we enter a cell, remember where we came from**
- **so when we reach the end,**
  - trace back the path from the upper right corner to read off the alignment
- For example, for each field of the table we can add a field, ptr(i, j) whose value is either
  - LEFT, for insertion
  - DOWN, for deletion
  - DIAG, for substitution



Weighted Edit Distance

- **why would we add weights to the computation?**
  - **spell correction: some letters are more likely to be mistyped than others**
- a **confusion matrix** is a specific table layout that allows visualization of the performance of an algorithm; each column of the matrix represents the instances in a predicted class while each row represents the instances in an actual class (or vice-versa)



the confusion matrix for spelling errors shows us, e.g. that "e" is most often confused with "a",
and that "i" is often confused with both "e" and "a"