Natural Language Processing
Session 2

Nick Kadochnikov

THE UNIVERSITY OF CHICAGO
GRAHAM SCHOOL
CONTINUING LIBERAL AND PROFESSIONAL STUDIES

Session 2 Agenda

- Tokenization
- Stemming & Lemmatization
- Part-of-speech Tagging
- Sentence segmentation
- Minimum Edit Distance
- N-Grams
- Spelling Correction and the Noisy Channel

Basic Text Processing

Word tokenization

Text Normalization

- Every NLP task needs to do text normalization:
  1. Segmenting/tokenizing words in running text
  2. Normalizing word formats
  3. Segmenting sentences in running text

How many words?

- I do uh main- mainly business data processing
  - Fragments, filled pauses
- Seuss's cat in the hat is different from other cats!
  - **Lemma**: same stem, part of speech, rough word sense
    - cat and cats = same lemma
  - **Wordform**: the full inflected surface form
    - cat and cats = different wordforms

How many words?

they lay back on the San Francisco grass and looked at the stars and their

- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.
- How many?
  - 15 tokens (or 14)
  - 13 types (or 12) (or 11?)

How many words?

N = number of tokens

V = vocabulary = set of types

|*V*| is the size of the vocabulary

Church and Gale (1990): $|V| > O(N^{1/2})$

	Tokens = N	Types = |V|
Switchboard Telephone Speech Corpus	2.4 million	20 thousand
Shakespeare	884,000	31 thousand
Google N-grams	1 trillion	13 million

THE UNIVERSITY OF CHICAGO
GRAHAM SCHOOL
CONTINUING LIBERAL AND PROFESSIONAL STUDIES

Tokenization in Python

Issues in Tokenization

- Finland's capital → Finland Finlands Finland's ?
- what're, I'm, isn't → What are, I am, is not
- Hewlett-Packard → Hewlett Packard ?
- state-of-the-art → state of the art ?
- Lowercase → lower-case lowercase lower case ?
- San Francisco → one token or two?
- m.p.h., PhD. → ??

Tokenization: language issues

- French
  - ***L'ensemble*** $\rightarrow$ one token or two?
    - ***L*** ? ***L'*** ? ***Le*** ?
    - Want ***l'ensemble*** to match with ***un ensemble***
- German noun compounds are not segmented
  - ***Lebensversicherungsgesellschaftsangestellter***
  - 'life insurance company employee'
  - German information retrieval needs **compound splitter**

Tokenization: language issues

- Chinese and Japanese no spaces between words:
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
  - Sharapova now lives in US southeastern Florida
- Further complicated in Japanese, with multiple alphabets intermingled
  - Dates/amounts in multiple formats

フォーチュン500社は情報不足のため時間あた$500K(約6,000万円)

Katakana Hiragana Kanji Romaji

End-user can express query entirely in hiragana!

Word Tokenization in Chinese

- Also called **Word Segmentation**
- Chinese words are composed of characters
  - Characters are generally 1 syllable and 1 morpheme.
  - Average word is 2.4 characters long.
- Standard baseline segmentation algorithm:
  - Maximum Matching (also called Greedy)

Maximum Matching Word Segmentation Algorithm

- Given a wordlist of Chinese, and a string.

1) Start a pointer at the beginning of the string
2) Find the longest word in dictionary that matches the string starting at pointer
3) Move the pointer over the word in string
4) Go to 2

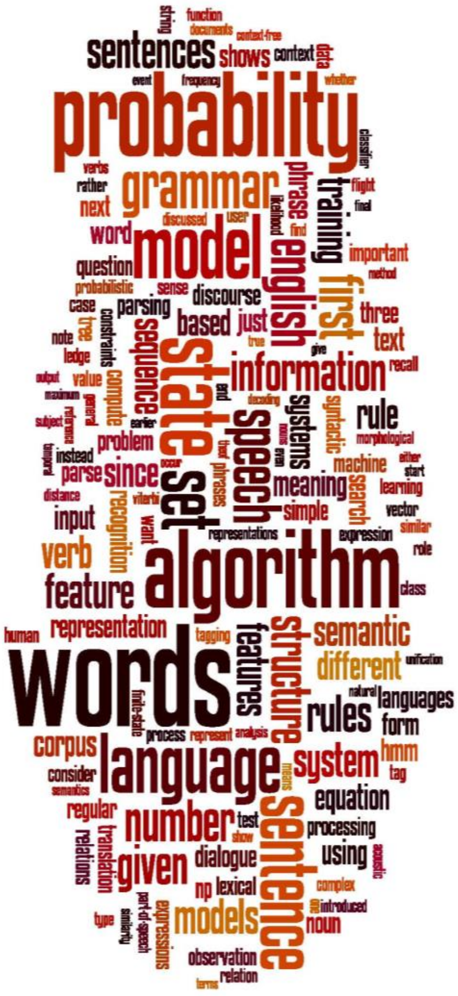
Max-match segmentation illustration

- Thecatinthehat — the cat in the hat
- Thetabledownthere — the table down there

  theta bled own there

- Doesn’t generally work in English!

- But works astonishingly well in Chinese
  - 莎拉波娃现在居住在美国东南部的佛罗里达。
  - 莎拉波娃 现在 居住 在 美国 东南部 的 佛罗里达
- Modern probabilistic segmentation algorithms even better



Basic Text Processing

Word Normalization, Stemming and Lemmatization

Normalization

- Need to "normalize" terms
  - Information Retrieval: indexed text & query terms must have same form.
    - We want to match ***U.S.A.*** and ***USA***
- We implicitly define equivalence classes of terms
  - e.g., deleting periods in a term
- Alternative: asymmetric expansion:
  - Enter: ***window*** Search: ***window, windows***
  - Enter: ***windows*** Search: ***Windows, windows, window***
  - Enter: ***Windows*** Search: ***Windows***
- Potentially more powerful, but less efficient

Case folding

- Applications like IR: reduce all letters to lower case
  - Since users tend to use lower case
  - Possible exception: upper case in mid-sentence?
    - e.g., ***General Motors***
    - ***Fed*** vs. ***fed***
    - ***SAIL*** vs. ***sail***
- For sentiment analysis, MT, Information extraction
  - Case is helpful (***US*** versus ***us*** is important)

Lemmatization

- Reduce inflections or variant forms to base form
  - *am, are, is* $\rightarrow$ *be*
  - *car, cars, car's, cars'* $\rightarrow$ *car*
- *the boy's cars are different colors* $\rightarrow$ *the boy car be different color*
- Lemmatization: have to find correct dictionary headword form
- Machine translation
  - Spanish quiero ('I want'), quieres ('you want') same lemma as querer 'want'

Morphology

- **Morphemes**:
  - The small meaningful units that make up words
  - **Stems**: The core meaning-bearing units
  - **Affixes**: Bits and pieces that adhere to stems
    - Often with grammatical functions

Stemming

- Reduce terms to their stems in information retrieval
- *Stemming* is crude chopping of affixes
  - language dependent
  - e.g., ***automate(s), automatic, automation*** all reduced to ***automat***.

for example compressed and compression are both accepted as equivalent to compress.

→

for exampl compress and compress ar both accept as equival to compress

Porter's algorithm The most common English stemmer

Step 1a

```
sses → ss     caresses → caress
ies  → i      ponies   → poni
ss   → ss     caress   → caress
s    → ø      cats     → cat
```

Step 1b

```
(*v*)ing → ø  walking   → walk
              sing      → sing
(*v*)ed  → ø  plastered → plaster
…
```

Step 2 (for long stems)

```
ational→ ate  relational→ relate
izer→ ize     digitizer → digitize
ator→ ate     operator  → operate
…
```

Step 3 (for longer stems)

```
al   → ø  revival    → reviv
able → ø  adjustable → adjust
ate  → ø  activate   → activ
…
```

Viewing morphology in a corpus
Why only strip –ing if there is a vowel?

(*v*)ing → ø   walking → walk

sing → sing

Viewing morphology in a corpus
Why only strip –ing if there is a vowel?

```
(*v*)ing → ø  walking   → walk
              sing      → sing
```

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep 'ing$' | sort | uniq -c | sort -nr
```

```
1312 King           548 being
 548 being          541 nothing
 541 nothing        152 something
 388 king           145 coming
 375 bring          130 morning
 358 thing          122 having
 307 ring           120 living
 152 something      117 loving
 145 coming         116 Being
 130 morning        102 going
```

```
tr -sc 'A-Za-z' '\n' < shakes.txt | grep '[aeiou].*ing$' | sort | uniq -c | sort -nr
```

Dealing with complex morphology is sometimes necessary

- Some languages requires complex morpheme segmentation
  - Turkish
  - Uygarlastiramadiklarimizdanmissinizcasina
  - `(behaving) as if you are among those whom we could not civilize'
  - Uygar `civilized' + las `become'

    + tir `cause' + ama `not able'

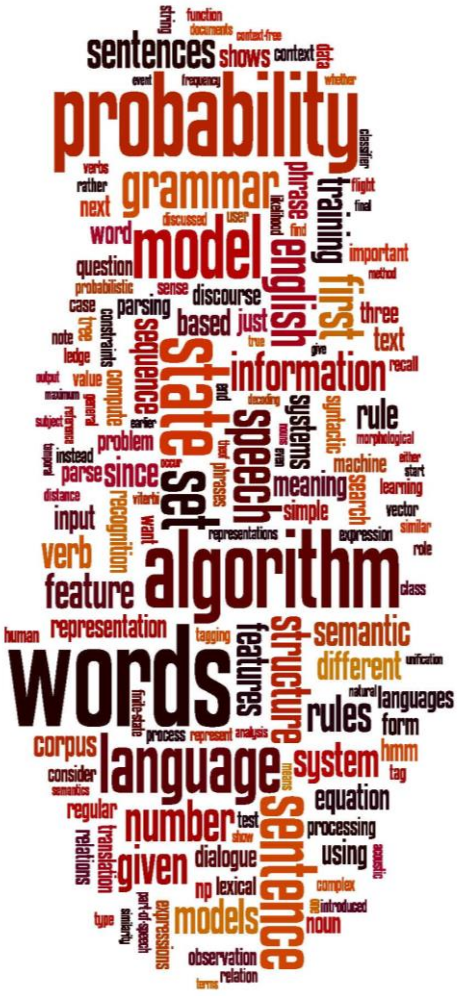
    + dik `past' + lar 'plural'

    + imiz 'p1pl' + dan 'abl'

    + mis 'past' + siniz '2pl' + casina 'as if'

THE UNIVERSITY OF CHICAGO
GRAHAM SCHOOL
CONTINUING LIBERAL AND PROFESSIONAL STUDIES

Stemming and Lemmatization in Python

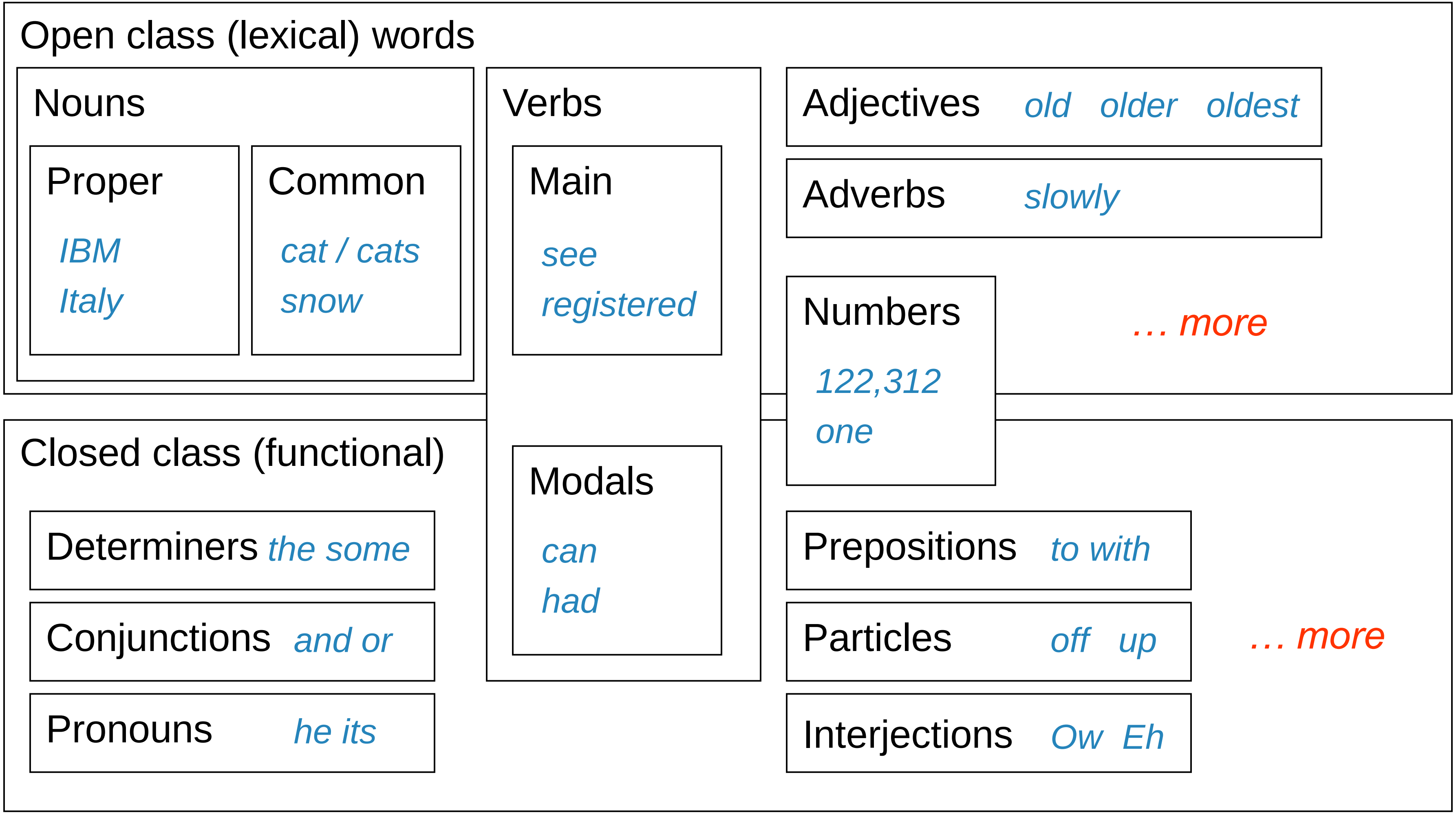


Part-of-speech tagging

A simple but useful form of linguistic analysis

Parts of Speech

- Perhaps starting with Aristotle in the West (384–322 BCE), there was the idea of having parts of speech
  - a.k.a lexical categories, word classes, “tags”, POS
- It comes from Dionysius Thrax of Alexandria (c. 100 BCE) the idea that is still with us that there are 8 parts of speech
  - But actually his 8 aren’t exactly the ones we are taught today
    - Thrax: noun, verb, article, adverb, preposition, conjunction, participle, pronoun
    - School grammar: noun, verb, adjective, adverb, preposition, conjunction, pronoun, interjection


Open class (lexical) words
Nouns
Proper
IBM
Italy
Common
cat / cats
snow
Verbs
Main
see
registered
Adjectives old older oldest
Adverbs slowly
Numbers
122,312
one
… more
Closed class (functional)
Modals
can
had
Determiners the some
Conjunctions and or
Pronouns he its
Prepositions to with
Particles off up
Interjections Ow Eh
… more

Open vs. Closed classes

- Open vs. Closed classes
  - Closed:
    - determiners: ***a, an, the***
    - pronouns: ***she, he, I***
    - prepositions: ***on, under, over, near, by, …***
    - Why “closed”?
  - Open:
    - Nouns, Verbs, Adjectives, Adverbs.

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.

POS Tagging

• Input:	Plays	well	with	others	
• Ambiguity:	NNS/VBZ	UH/JJ/NN/RB	IN	NNS	
• Output:	Plays/VBZ well/RB with/IN others/NNS				

Penn Treebank POS tags

- Uses:
  - Text-to-speech (how do we pronounce "lead"?)
  - Can write regexps like (Det) Adj* N+ over the output for phrases, etc.
  - As input to or to speed up a full parser
  - If you know the tag, you can back off to it in other tasks

POS tagging performance

- How many tags are correct? (Tag accuracy)
  - About 97% currently
  - But baseline is already 90%
    - Baseline is performance of stupidest possible method
      - Tag every word with its most frequent tag
      - Tag unknown words as nouns
  - Partly easy because
    - Many words are unambiguous
    - You get points for them (*the*, *a*, etc.) and for punctuation marks!

Deciding on the correct part of speech 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 250/CD

How difficult is POS tagging?

- About 11% of the word types in the Brown corpus are ambiguous with regard to part of speech
- But they tend to be very common words. E.g., *that*
  - I know *that* he is honest = IN
  - Yes, *that* play was nice = DT
  - You can’t go *that* far = RB
- 40% of the word tokens are ambiguous

Sources of information

- What are the main sources of information for POS tagging?
  - Knowledge of neighboring words
    - Bill saw that man yesterday
    - NNP NN DT NN NN
    - VB VB(D) IN VB NN
  - Knowledge of word probabilities
    - *man* is rarely used as a verb….
- The latter proves the most useful, but the former also helps

More and Better Features → Feature-based tagger

- Can do surprisingly well just looking at a word by itself:
  - Word — the: the → DT
  - Lowercased word — Importantly: importantly → RB
  - Prefixes — unfathomable: un- → JJ
  - Suffixes — Importantly: -ly → RB
  - Capitalization — Meridian: CAP → NNP
  - Word shapes — 35-year: d-x → JJ
- Then build a maxent (or whatever) model to predict tag
  - Maxent P(t|w): 93.7% overall / 82.6% unknown

Overview: POS Tagging Accuracies

- Rough accuracies:
  - Most freq tag: ~90% / ~50%
  - Trigram HMM: ~95% / ~55%
  - Maxent P(t|w): 93.7% / 82.6%
  - TnT (HMM++): 96.2% / 86.0%
  - MEMM tagger: 96.9% / 86.9%
  - Bidirectional dependencies: 97.2% / 90.0%
  - Upper bound: ~98% (human agreement)

Most errors on unknown words

How to improve supervised results?

- Build better features!

<table>
<tr><td></td><td></td><td>RB</td><td></td><td></td><td></td><td></td><td></td></tr>
<tr><td>PRP</td><td>VBD</td><td>IN</td><td>RB</td><td>IN</td><td>PRP</td><td>VBD</td><td>.</td></tr>
<tr><td>They</td><td>left</td><td>as</td><td>soon</td><td>as</td><td>he</td><td>arrived</td><td>.</td></tr>
</table>

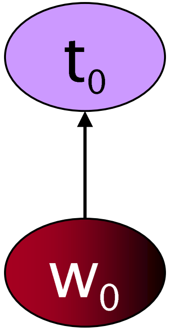
  - We could fix this with a feature that looked at the next word

<table>
<tr><td>JJ</td><td></td><td></td><td></td><td></td></tr>
<tr><td>NNP</td><td>NNS</td><td>VBD</td><td>VBN</td><td>.</td></tr>
<tr><td>Intrinsic</td><td>flaws</td><td>remained</td><td>undetected</td><td>.</td></tr>
</table>

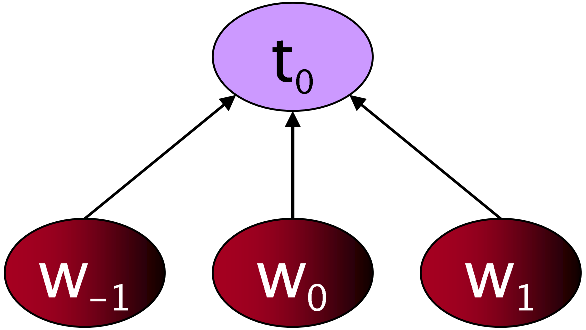
  - We could fix this by linking capitalized words to their lowercase versions

Tagging Without Sequence Information

Baseline



Three Words



Model	Features	Token	Unknown	Sentence
Baseline	56,805	**93.69%**	82.61%	26.74%
3Words	239,767	**96.57%**	86.78%	48.27%

Using words only in a straight classifier works as well as a basic (HMM or discriminative) sequence model!!

Summary of POS Tagging

For tagging, the change from generative to discriminative model **does not by itself** result in great improvement

One profits from models for specifying dependence on **overlapping features of the observation** such as spelling, suffix analysis, etc.

An MEMM allows integration of rich features of the observations, but can suffer strongly from assuming independence from following observations; this effect can be relieved by adding dependence on following words

This additional power (of the MEMM ,CRF, Perceptron models) has been shown to result in improvements in accuracy

The **higher accuracy** of discriminative models comes at the price of **much slower training**

THE UNIVERSITY OF CHICAGO
GRAHAM SCHOOL
CONTINUING LIBERAL AND PROFESSIONAL STUDIES

Part of Speech Tagging in Python

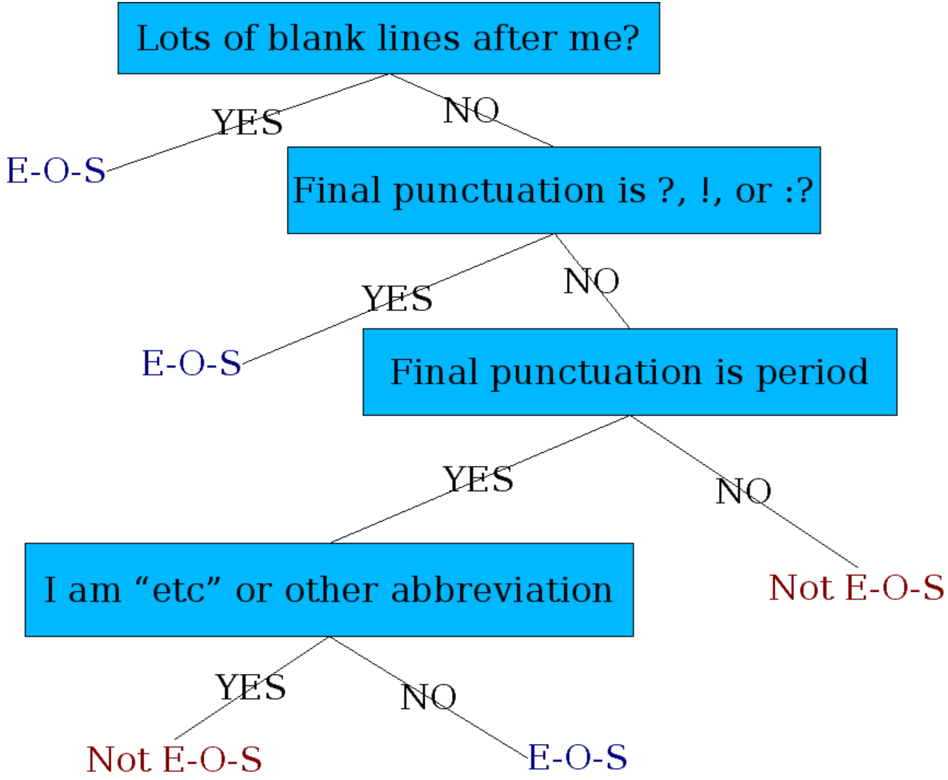
Basic Text Processing

Sentence Segmentation and Decision Trees

Sentence Segmentation

- !, ? are relatively unambiguous
- Period "." is quite ambiguous
  - Sentence boundary
  - Abbreviations like Inc. or Dr.
  - Numbers like .02% or 4.3
- Build a binary classifier
  - Looks at a "."
  - Decides EndOfSentence/NotEndOfSentence
  - Classifiers: hand-written rules, regular expressions, or machine-learning

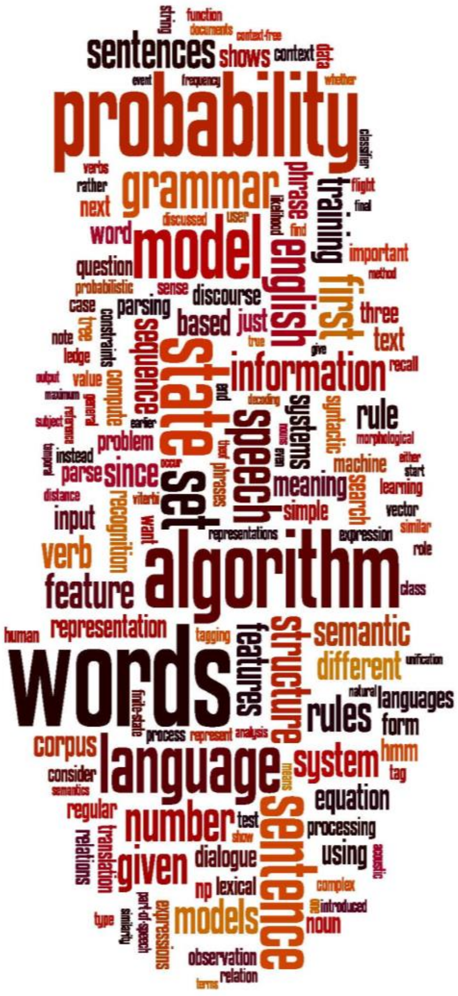
Determining if a word is end-of-sentence: a Decision Tree



More sophisticated decision tree features

- Case of word with ".": Upper, Lower, Cap, Number
- Case of word after ".": Upper, Lower, Cap, Number

- Numeric features
  - Length of word with "."
  - Probability(word with "." occurs at end-of-s)
  - Probability(word after "." occurs at beginning-of-s)



Minimum Edit Distance

Definition of Minimum Edit Distance

How similar are two strings?

- Spell correction
  - The user typed “graffe”

  Which is closest?
    - graf
    - graft
    - grail
    - giraffe

- Computational Biology
  - Align two sequences of nucleotides

    AGGCTATCACCTGACCTCCAGGCCGATGCCC
    TAGCTATCACGACCGCGGTCGATTTGCCCGAC

  - Resulting alignment:

    -AGGCTATCACCTGACCTCCAGGCCGA--TGCCC---
    TAG-CTATCAC--GACCGC--GGTCGATTTGCCCGAC

- Also for Machine Translation, Information Extraction, Speech Recognition

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

Minimum Edit Distance

- Two strings and their **alignment**:

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
```

Minimum Edit Distance

```
I N T E * N T I O N
| | | | | | | | | |
* E X E C U T I O N
d s s   i s
```

- If each operation has cost of 1
  - Distance between these is 5
- If substitutions cost 2 (Levenshtein)
  - Distance between them is 8

Dan Jurafsky

Other uses of Edit Distance in NLP

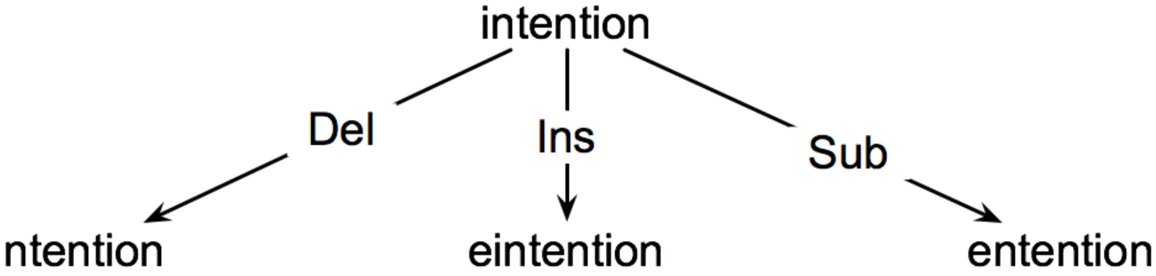
- Evaluating Machine Translation and speech recognition

R	Spokesman	confirms		senior	government	adviser	was	shot	
H	Spokesman	said	the	senior		adviser	was	shot	dead
		S	I		D				I

- Named Entity Extraction and Entity Coreference
  - IBM Inc. announced today
  - IBM profits
  - Stanford President John Hennessy announced yesterday
  - for Stanford University President John Hennessy

How to find the Min Edit Distance?

- Searching for a path (sequence of edits) from the start string to the final string:
  - **Initial state**: the word we're transforming
  - **Operators**: insert, delete, substitute
  - **Goal state**: the word we're trying to get to
  - **Path cost**: what we want to minimize: the number of edits

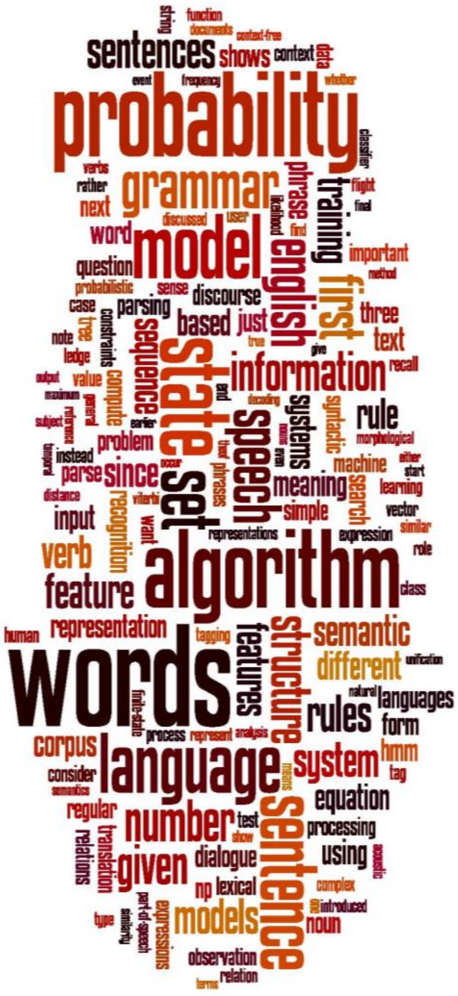


Minimum Edit as Search

- But the space of all edit sequences is huge!
  - We can’t afford to navigate naïvely
  - Lots of distinct paths wind up at the same state.
    - We don’t have to keep track of all of them
    - Just the shortest path to each of those revisted states.

Defining Min Edit Distance

- For two strings
  - X of length *n*
  - Y of length *m*
- We define D(*i*,*j*)
  - the 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
  - The edit distance between X and Y is thus D(*n*,*m*)



Minimum Edit Distance

Weighted Minimum Edit Distance

Weighted Edit Distance

- Why would we add weights to the computation?
  - Spell Correction: some letters are more likely to be mistyped than others
  - Biology: certain kinds of deletions or insertions are more likely than others

Confusion matrix for spelling errors

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

Y (correct)

X	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


~ ` ! 1 @ 2 # 3 $ 4 % 5 ^ 6 & 7 * 8 (9) 0 — - + =
Tab Q W E R T Y U I O P { [}]
Caps Lock A S D F G H J K L : ; " ' Enter
Shift Z X C V B N M < , > . ? / Shift
Ctrl Alt Alt

Language Modeling

Introduction to N-grams

Probabilistic Language Models

- Today's goal: assign a probability to a sentence

Why?

- Machine Translation:
  - P(**high** winds tonite) > P(**large** winds tonite)
- Spell Correction
  - The office is about fifteen **minuets** from my house
    - P(about fifteen **minutes** from) > P(about fifteen **minuets** from)
- Speech Recognition
  - P(I saw a van) >> P(eyes awe of an)
- + Summarization, question-answering, etc., etc.!!

Probabilistic Language Modeling

- Goal: compute the probability of a sentence or sequence of words:

  $P(W) = P(w_1,w_2,w_3,w_4,w_5 \ldots w_n)$

- Related task: probability of an upcoming word:

  $P(w_5|w_1,w_2,w_3,w_4)$

- A model that computes either of these:

  $P(W)$ or $P(w_n|w_1,w_2 \ldots w_{n-1})$ is called a **language model**.

- Better: **the grammar** But **language model** or **LM** is standard

How to compute P(W)

- How to compute this joint probability:
  - P(its, water, is, so, transparent, that)
- Intuition: let's rely on the Chain Rule of Probability

Reminder: The Chain Rule

- Recall the definition of conditional probabilities

Rewriting:

- More variables:

$$P(A,B,C,D) = P(A)P(B|A)P(C|A,B)P(D|A,B,C)$$

- The Chain Rule in General

$$P(x_1,x_2,x_3,\ldots,x_n) = P(x_1)P(x_2|x_1)P(x_3|x_1,x_2)\ldots P(x_n|x_1,\ldots,x_{n-1})$$

The Chain Rule applied to compute joint probability of words in sentence

$$P(w_1 w_2 \dots w_n) = \prod_i P(w_i \mid w_1 w_2 \dots w_{i-1})$$

P("its water is so transparent") =

P(its) × P(water|its) × P(is|its water)

× P(so|its water is) × P(transparent|its water is so)

How to estimate these probabilities

- Could we just count and divide?

$$P(\text{the} \mid \text{its water is so transparent that}) = \frac{Count(\text{its water is so transparent that the})}{Count(\text{its water is so transparent that})}$$

- No! Too many possible sentences!
- We'll never see enough data for estimating these

Markov Assumption

Andrei Markov

- Simplifying assumption:

$$P(\text{the} \mid \text{its water is so transparent that}) \text{ » } P(\text{the} \mid \text{that})$$

- Or maybe

$$P(\text{the} \mid \text{its water is so transparent that}) \text{ » } P(\text{the} \mid \text{transparent that})$$

Markov Assumption

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i \mid w_{i-k} \ldots w_{i-1})$$

- In other words, we approximate each component in the product

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-k} \ldots w_{i-1})$$

Simplest case: Unigram model

$$P(w_1 w_2 \ldots w_n) \approx \prod_i P(w_i)$$

Some automatically generated sentences from a unigram model

```
fifth, an, of, futures, the, an, incorporated, a,
a, the, inflation, most, dollars, quarter, in, is,
mass

thrift, did, eighty, said, hard, 'm, july, bullish

that, or, limited, the
```

Bigram model

- Condition on the previous word:

$$P(w_i \mid w_1 w_2 \ldots w_{i-1}) \approx P(w_i \mid w_{i-1})$$

texaco, rose, one, in, this, issue, is, pursuing, growth, in, a, boiler, house, said, mr., gurria, mexico, 's, motion, control, proposal, without, permission, from, five, hundred, fifty, five, yen

outside, new, car, parking, lot, of, the, agreement, reached

this, would, be, a, record, november

N-gram models

- We can extend to trigrams, 4-grams, 5-grams
- In general this is an insufficient model of language
  - because language has **long-distance dependencies**:

  "The computer which I had just put into the machine room on the fifth floor crashed."

- But we can often get away with N-gram models

Google N-Gram Release, August 2006

AUG 3

All Our N-gram are Belong to You

Posted by Alex Franz and Thorsten Brants, Google Machine Translation Team

Here at Google Research we have been using word n-gram models for a variety of R&D projects,

…

That's why we decided to share this enormous dataset with everyone. We processed 1,024,908,267,229 words of running text and are publishing the counts for all 1,176,470,663 five-word sequences that appear at least 40 times. There are 13,588,391 unique words, after discarding words that appear less than 200 times.

Google N-Gram Release

- `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`
- `serve as the indicators 45`
- `serve as the indispensable 111`
- `serve as the indispensible 40`
- `serve as the individual 234`

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

Google Book N-grams

- http://ngrams.googlelabs.com/

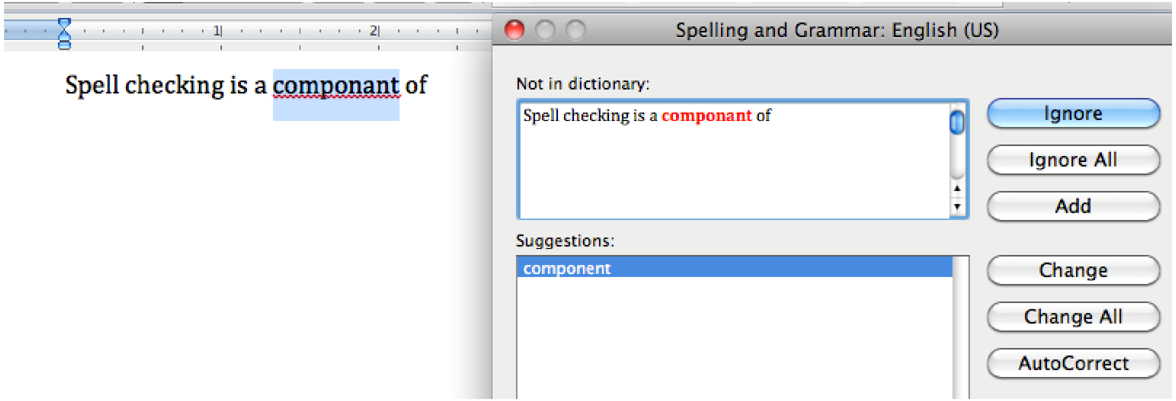
N-Grams in Python

Spelling Correction and the Noisy Channel

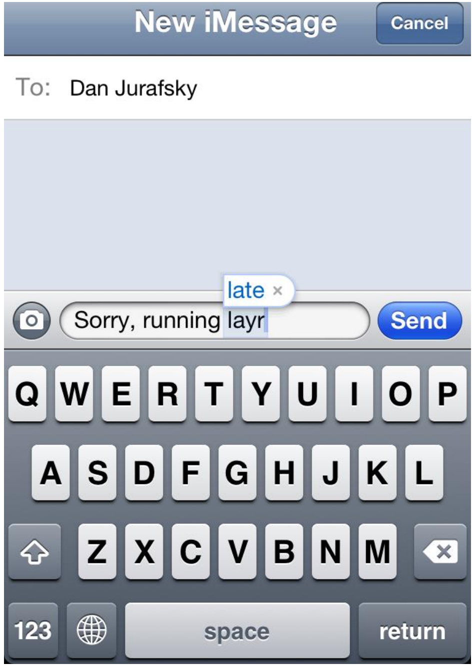
The Spelling Correction Task

Applications for spelling correction

Word processing



Phones



Web search

Showing results for natural ***language*** processing

Search instead for natural langage processing

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*

Rates of spelling errors

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

13%: Typing, no delete/ backspace: Whitelaw *et al.* English&German

7%: Words corrected retyping on phone-sized organizer

2%: Words uncorrected on organizer Soukoreff &MacKenzie 2003

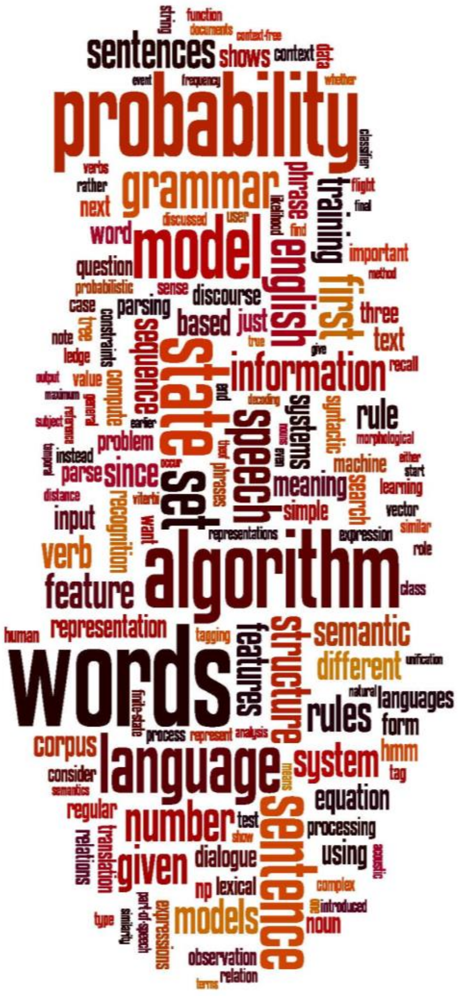
1-2%: Retyping: Kane and Wobbrock 2007, Gruden et al. 1983

Non-word spelling errors

- Non-word spelling error detection:
  - Any word not in a ***dictionary*** is an error
  - The larger the dictionary the better
- 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 spelling errors

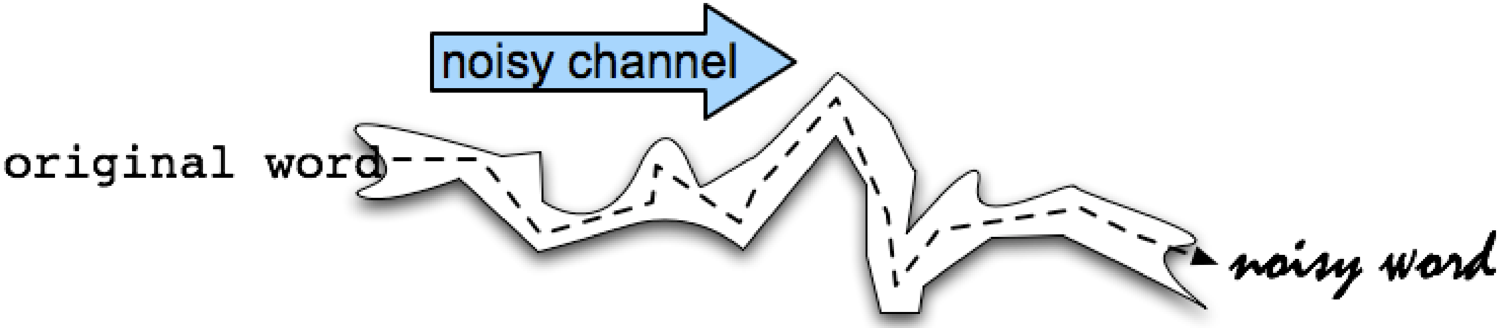
- For each word *w*, generate candidate set:
  - Find candidate words with similar ***pronunciations***
  - Find candidate words with similar ***spelling***
  - Include *w* in candidate set
- Choose best candidate
  - Noisy Channel
  - Classifier



Spelling Correction and the Noisy Channel

The Noisy Channel Model of Spelling

Noisy Channel Intuition



Noisy Channel

- We see an observation x of a misspelled word
- Find the correct word w

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

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

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

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 edit distance of pronunciation to error

Damerau-Levenshtein edit distance

- Minimal edit distance between two strings, where edits are:
  - Insertion
  - Deletion
  - Substitution
  - Transposition of two adjacent letters

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`

Language Model

- Use any of the language modeling algorithms we've learned
- Unigram, bigram, trigram
- Web-scale spelling correction
    - Stupid backoff

Unigram Prior probability

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

word	Frequency of word	P(word)
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(x typed as y)
trans[x,y]:   count(xy typed as yx)
```

Insertion and deletion conditioned on previous character

Confusion matrix for spelling errors

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

Generating the confusion matrix

- Peter Norvig's list of errors
- Peter Norvig's list of counts of single-edit errors

Channel model

Kernighan, Church, Gale 1990

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

Channel model for `acress`

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)
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

Noisy channel probability for acress

<table>
<tr><th>Candidate Correction</th><th>Correct Letter</th><th>Error Letter</th><th>x|w</th><th>P(x|word)</th><th>P(word)</th><th>10^9 *P(x|w)P(w)</th></tr>
<tr><td>actress</td><td>t</td><td>-</td><td>c|ct</td><td>.000117</td><td>.0000231</td><td>2.7</td></tr>
<tr><td>cress</td><td>-</td><td>a</td><td>a|#</td><td>.00000144</td><td>.000000544</td><td>.00078</td></tr>
<tr><td>caress</td><td>ca</td><td>ac</td><td>ac|ca</td><td>.00000164</td><td>.00000170</td><td>.0028</td></tr>
<tr><td>access</td><td>c</td><td>r</td><td>r|c</td><td>.000000209</td><td>.0000916</td><td>.019</td></tr>
<tr><td>across</td><td>o</td><td>e</td><td>e|o</td><td>.0000093</td><td>.000299</td><td>2.8</td></tr>
<tr><td>acres</td><td>-</td><td>s</td><td>es|e</td><td>.0000321</td><td>.0000318</td><td>1.0</td></tr>
<tr><td>[illegible]</td><td>[illegible]</td><td>[illegible]</td><td>[illegible]</td><td>[illegible]</td><td>[illegible]</td><td>[illegible]</td></tr>
</table>

Noisy channel probability for acress

Candidate Correction	Correct Letter	Error Letter	x|w	P(x|word)	P(word)	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
[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]

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}

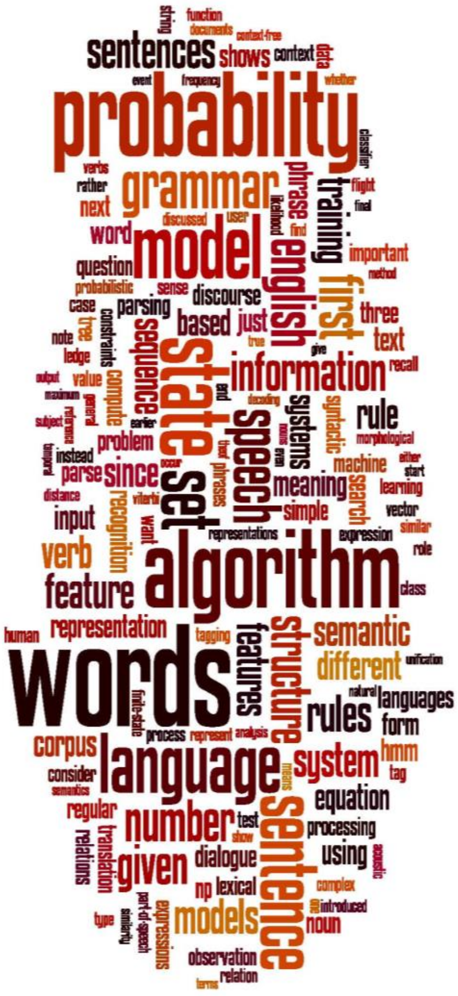
Using a bigram language model

- "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}

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)



Spelling Correction and the Noisy Channel

Real-Word Spelling Correction

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

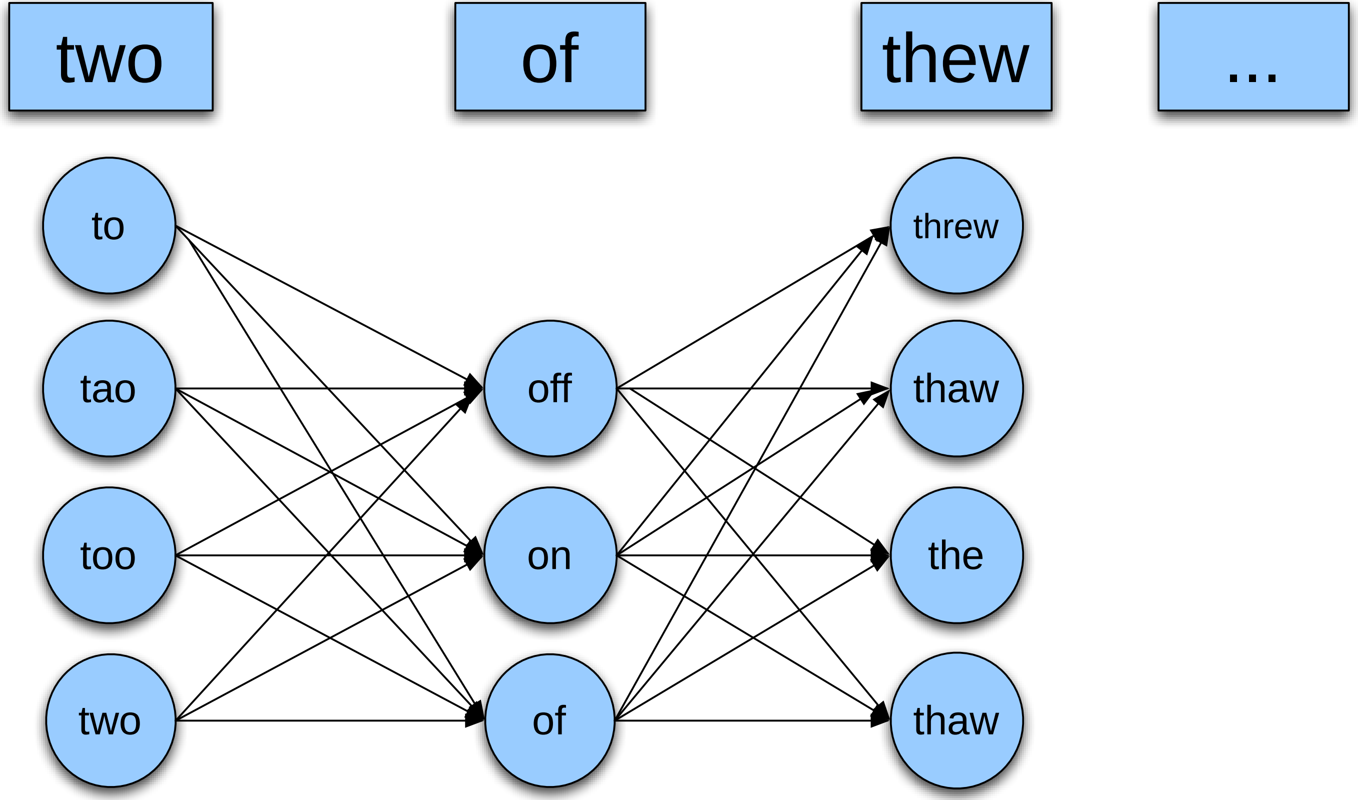
Solving real-world spelling errors

- For each word in sentence
  - Generate *candidate set*
    - the word itself
    - all single-letter edits that are English words
    - words that are homophones
- Choose best candidates
    - Noisy channel model
    - Task-specific classifier

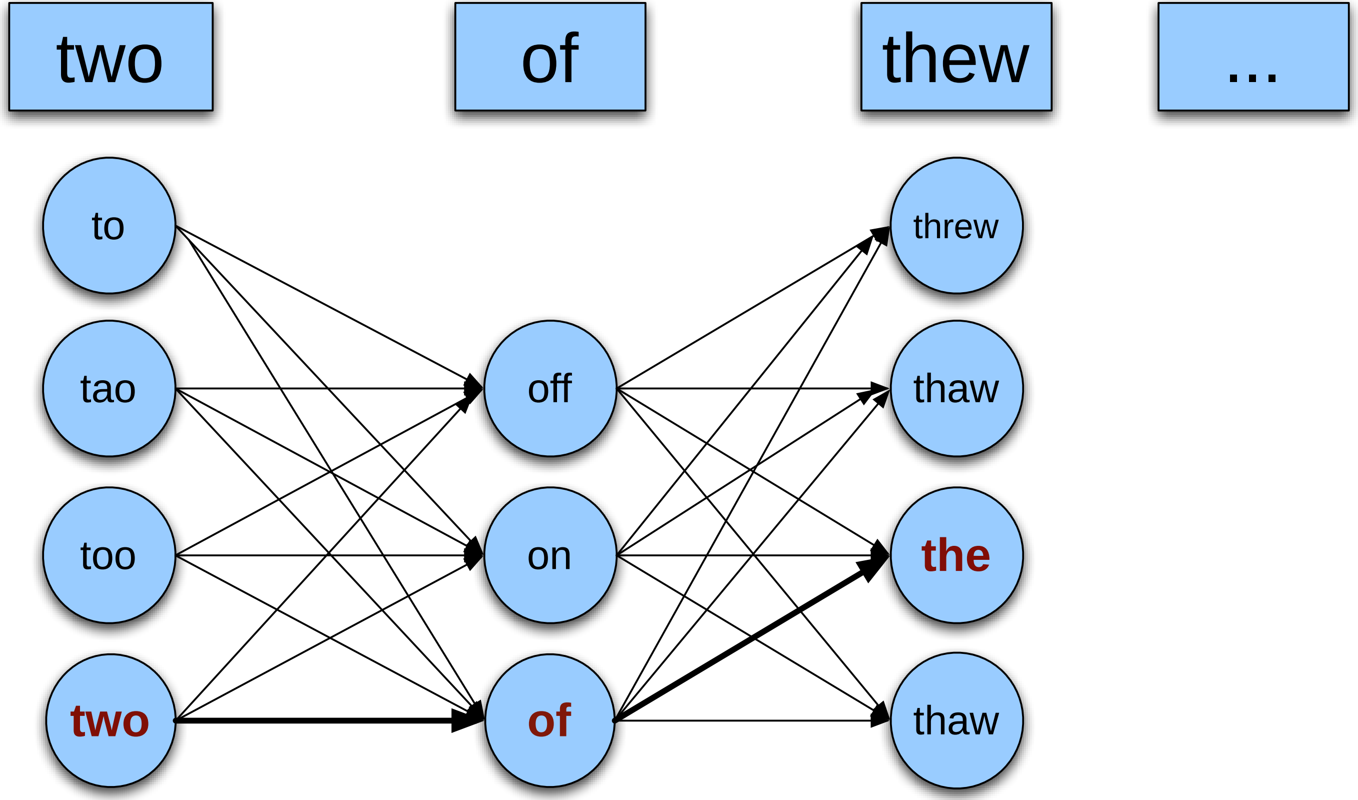
Noisy channel for real-word spell correction

- 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) = $\{w_1, w'_1, w''_1, w'''_1, \ldots\}$
  - Candidate(w_2) = $\{w_2, w'_2, w''_2, w'''_2, \ldots\}$
  - Candidate(w_n) = $\{w_n, w'_n, w''_n, w'''_n, \ldots\}$
- Choose the sequence W that maximizes P(W)

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")

- Obviously this depends on the application
  - .90 (1 error in 10 words)
  - .95 (1 error in 20 words)
  - .99 (1 error in 100 words)
  - .995 (1 error in 200 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

Spelling Correction and the Noisy Channel

State-of-the-art Systems

HCI issues in spelling

- If very confident in correction
  - Autocorrect
- Less confident
  - Give the best correction
- Less confident
  - Give a correction list
- Unconfident
  - Just flag as an error

HCI = Human Computer Interaction

State of the art noisy channel

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

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

- Learn λ from a development test set

Phonetic error model

- Metaphone, used in GNU aspell
  - Convert misspelling to metaphone pronunciation
    - “Drop duplicate adjacent letters, except for C.”
    - “If the word begins with 'KN', 'GN', 'PN', 'AE', 'WR', drop the first letter.”
    - “Drop 'B' if after 'M' and if it is at the end of the word”
    - …
  - Find words whose pronunciation is 1-2 edit distance from misspelling’s
  - Score result list
    - Weighted edit distance of candidate to misspelling
    - Edit distance of candidate pronunciation to misspelling pronunciation

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)

Channel model

- Factors that could influence p(misspelling|word)
  - The source letter
  - The target letter
  - Surrounding letters
  - The position in the word
  - Nearby keys on the keyboard
  - Homology on the keyboard
  - Pronunciations
  - Likely morpheme transformations

Nearby keys

Classifier-based methods for real-word spelling correction

- Instead of just channel model and language model
- Use many features in a classifier (next lecture).
- Build a classifier for a specific pair like:

whether/weather

  - “cloudy” within +- 10 words
  - ___ to VERB
  - ___ or not

THE UNIVERSITY OF CHICAGO
GRAHAM SCHOOL
CONTINUING LIBERAL AND PROFESSIONAL STUDIES

Thank You!