Introduction to Information Retrieval

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

The course structure ...

Index construction
Index compression
Efficient boolean querying
Chapter/lecture 1, 2, 4, 5

Spelling correction
Chapter/lecture 3 (mainly some parts)
This lecture (PA #21)

The course structure ...

tf.idf weighting
The vector space model
Gerry Salton
Chapter/lecture 6,7

Probabilistic term weighting
Thursday/next Tuesday
In-class lecture (PA #3!)
Chapter 11

Applications for spelling correction

Word processing

Phones

Web search

Spelling Tasks

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

Types of spelling errors

- Non-word Errors
  - graffe → giraffe
- Real-word Errors
  - Typographical errors
    - three → there
  - Cognitive Errors (homophones)
    - peace → piece,
    - two → too
    - you → you’re

- Non-word correction was historically mainly context insensitive
- Real-word correction almost needs to be context sensitive

Spelling Tasks

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

Types of spelling errors

- Non-word Errors
  - graffe → giraffe
- Real-word Errors
  - Typographical errors
    - three → there
  - Cognitive Errors (homophones)
    - peace → piece,
    - too → two
    - you → you’re

- Non-word correction was historically mainly context insensitive
- Real-word correction almost needs to be context sensitive
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

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 from 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)

The Noisy Channel Model of Spelling

INDEPENDENT WORD SPELLING CORRECTION
Noisy Channel = Bayes’ Rule

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

$$\hat{w} = \arg \max_{w \in V} P(w | x)$$

$$= \arg \max_{w \in V} \frac{P(x | w)P(w)}{P(x)}$$

$$= \arg \max_{w \in V} P(x | w)P(w)$$

History: Noisy channel for spelling proposed around 1990

- IBM

- AT&T Bell Labs

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

<table>
<thead>
<tr>
<th>Error</th>
<th>Candidate Correction</th>
<th>Correct Letter</th>
<th>Error Letter</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>acress</td>
<td>actress</td>
<td>t</td>
<td>–</td>
<td>deletion</td>
</tr>
<tr>
<td>acress</td>
<td>cress</td>
<td>–</td>
<td>a</td>
<td>insertion</td>
</tr>
<tr>
<td>acress</td>
<td>caress</td>
<td>ca</td>
<td>ac</td>
<td>transposition</td>
</tr>
<tr>
<td>acress</td>
<td>access</td>
<td>c</td>
<td>r</td>
<td>substitution</td>
</tr>
<tr>
<td>acress</td>
<td>across</td>
<td>o</td>
<td>e</td>
<td>substitution</td>
</tr>
<tr>
<td>acress</td>
<td>acres</td>
<td>–</td>
<td>s</td>
<td>insertion</td>
</tr>
</tbody>
</table>
**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

**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)

<table>
<thead>
<tr>
<th>word</th>
<th>Frequency of word</th>
<th>(P(w))</th>
</tr>
</thead>
<tbody>
<tr>
<td>actress</td>
<td>9,321</td>
<td>0.000230573</td>
</tr>
<tr>
<td>cress</td>
<td>220</td>
<td>0.000005442</td>
</tr>
<tr>
<td>caress</td>
<td>686</td>
<td>0.000016969</td>
</tr>
<tr>
<td>access</td>
<td>37,038</td>
<td>0.0000916207</td>
</tr>
<tr>
<td>across</td>
<td>120,844</td>
<td>0.0002989314</td>
</tr>
<tr>
<td>acres</td>
<td>12,874</td>
<td>0.000318463</td>
</tr>
</tbody>
</table>

**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 \(\hat{w}\)
  - \(\hat{w} = \arg\max_{w \in V} P(w \mid x)\)
  - \(= \arg\max_{w \in V} \frac{P(x \mid w)P(w)}{P(x)}\)
  - \(= \arg\max_{w \in V} P(x \mid w)P(w)\)

What’s \(P(w)\)?
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”

\[
\text{del}[x,y] = \frac{\text{count}(xy \text{ typed as } x)}{\text{count}(xy)}
\]

\[
\text{ins}[x,y] = \frac{\text{count}(x \text{ typed as } xy)}{\text{count}(x)}
\]

\[
\text{sub}[x,y] = \frac{\text{count}(y \text{ typed as } x)}{\text{count}(y)}
\]

\[
\text{trans}[x,y] = \frac{\text{count}(xy \text{ typed as } yx)}{\text{count}(xy)}
\]

Insertion and deletion conditioned on previous character

Confusion matrix for substitution

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

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

Channel model for across

| Candidate Correction | Correct Letter | Error Letter | x/w  | P(x|w)  | P(w)  | \(10^4 \cdot \frac{P(x|w)}{P(w)}\) |
|----------------------|----------------|--------------|------|---------|-------|---------------------|
| actress t            | t              | c            | .000117 | .0000231 | 2.7   |                     |
| cress                | a              | a            | .00000144 | .00000544 | .00078 |                     |
| caress               | ca             | ac           | .00000164 | .00000170 | .0028  |                     |
| access               | c              | r            | .00000209 | .0000916  | .019   |                     |
| across               | o              | e            | .0000093  | .0000299  | 2.8    |                     |
| acres                | s              | es           | .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 mistakes to go to her house.
- The design as construction of the system.
- Can they save 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

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

\[
P(w_1...w_n) = P(w_1|w_2|w_3)...P(w_n|w_{n-1})
\]

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 \( P(w_1|w_{n-1}) \). 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:

\[
P_{\lambda}(w_1|w_{n-2}) = \lambda P_{un}(w_1) + (1-\lambda)P_{bi}(w_1|w_{n-2})
\]

\[
P_{\lambda}(w_1|w_{n-2}) = C(w_{n-2}, w_1) / C(w_{n-2})
\]

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:  
  - \( \log P(w_1|w_2) = \log P(w_2|w_1) + \log P(w_1|w_{n-1}) \)
  - 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 across whose combination of sass and glamour…”
- Counts from the Corpus of Contemporary American English with add-1 smoothing
  - $P(\text{actress}|\text{versatile}) = 0.00021$  
  - $P(\text{whose}|\text{actress}) = 0.0010$
  - $P(\text{across}|\text{versatile}) = 0.00021$  
  - $P(\text{whose}|\text{across}) = 0.000006$

- $P(\text{"versatile actress whose"}) = 0.00021 \times 0.0010 = 210 \times 10^{-10}$
- $P(\text{"versatile across whose"}) = 0.00021 \times 0.000006 = 1 \times 10^{-10}$

Simplification: One error per sentence

- Out of all possible sentences with one word replaced
  - $w_1, w'_2, w_3, w_4$ two off threw
  - $w_2, w'_3, w_4$ two of the
  - $w''_1, w_2, w'_3, w_4$ too of threw
  - ...
- 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?
  - \( P(\text{“the”} | \text{“the”}) \)
    - 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 “thaw” example**

| x | w | x | w | P(x|w) | P(w) | \(10^0 P(x|w)P(w)\) |
|---|---|---|---|---|---|---|
| thew | the | ew|e | 0.000007 | 0.02 | 144 |
| thew | thew | 0.95 | 0.00000009 | 90 |
| thew | thaw | e|ja | 0.001 | 0.000007 | 0.7 |
| thew | threw | h|hr | 0.000008 | 0.000004 | 0.03 |
| thew | thwe | we | 0.000003 | 0.0000004 | 0.0001 |

**State of the art noisy channel**

- We never just multiply the prior and the error model
- Independence assumptions \(\Rightarrow\) probabilities not commensurate
  - Instead: Weight them
    \[
    \hat{w} = \arg \max_{w \in V} P(x | w) P(w)^{\lambda}
    \]
- Learn \( \lambda \) from a development test set

**Improvements to channel model**

- Allow richer edits (Brill and Moore 2000)
  - en\(\Rightarrow\)ant
  - ph\(\Rightarrow\)f
  - le\(\Rightarrow\)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