

More on Noisy Channel (From Lecture 5 Slides)

Independent (Non-word) Word Spelling Correction

- The goal: Find the intended word, given a word where the letters have been scrambled in some manner (quoted from Wikipedia)
- Noisy model = Bayes' Rule
 - $\hat{w} = \arg \max_{w \in V} P(w | x) = \arg \max_{w \in V} \frac{P(x|w)P(w)}{P(x)} = \arg \max_{w \in V} P(x | w)P(w)$
- Candidate generation
 - Words with similar spelling: short edit distance
 - * 80% of errors are within edit distance 1
 - * Almost all errors are with edit distance 2
 - * Allow inserting space or hyphen
 - * Allow merging words
 - Words with similar pronunciation
- What's $P(w)$?
 - Language Model: With a big supply of words (A document collection with T tokens), let $P(w) = \frac{C(w)}{T}$, where $C(w)$ is the total occurrence of w in the collection.
 - In other collections, we can simply consider the supply to be the query typed in.
- What's $P(x | w)$?: Probability of the *edit* (deletion, insertion, substitution, transposition)
- Channel Model:
 - Confusion matrix for substitution
 - * $\text{del}[x, y]$: Count xy typed as x
 - * $\text{ins}[x, y]$: Count x typed as xy
 - * $\text{sub}[x, y]$: Count y typed as x
 - * $\text{trans}[x, y]$: Count xy typed as yx
 - If deletion, $P(x | w) = \frac{\text{del}[w_{i-1}, w_i]}{\text{count}[w_{i-1}w_i]}$
 - If insertion, $P(x | w) = \frac{\text{ins}[w_{i-1}, x_i]}{\text{count}[w_{i-1}]}$
 - If substitution, $P(x | w) = \frac{\text{sub}[x_i, w_i]}{\text{count}[w_i]}$
 - If transposition, $P(x | w) = \frac{\text{trans}[w_i, w_{i+1}]}{\text{count}[w_iw_{i+1}]}$
 - Will gonna need Add-1 Smoothing, since unseen errors would be considered impossible otherwise.

Context-sensitive Spelling Correction with the Noisy Channel

- 25-40% of spelling errors are *real words*
- Context-sensitive spelling error fixing
 - Generate candidate set (Pre-computable)
 1. The word itself
 2. All *one*-letter edits that are English words
 3. Words that are *homophones* (i.e. pronounced the same but differ in meaning and spelling)
 - Choose best candidates
 - * Given a sentence $x_1x_2x_3 \cdots x_n$
 - * Generate a set of candidates for each word x_i
 - * Choose the sequence W that maximizes $P(W | x_1, \dots, x_n)$
- We need to a better *language model*, in order to determine whether a particular word is more appropriate correction than another in the given context.
 - *Bigram language model*: Conditions the probability of a word on just the *previous word*
 - * $P(w_1 \cdots w_n) = P(w_1)P(w_2 | w_1) \cdots p(w_n | w_{n-1})$
 - For unigram counts, $P(w)$ is always non-zero, assuming that we construct our dictionary from the document collection.
 - $P(w_k | w_{k-1})$ need to be smoothed
 - * Add-1 smoothing
 - * Interpolate a unigram and a bigram: $P_{\text{bi}}(w_k | w_{k-1}) = \frac{C(w_{k-1}, w_k)}{C(w_{k-1})}$, $P_{\text{li}}(w_k | w_{k-1}) = \lambda P_{\text{uni}}(w_k) + (1 - \lambda)P_{\text{bi}}(w_k | w_{k-1})$
- You might need to work with log probabilities
- 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.
- Where to get the probabilities

- Language model
- Channel model: same as for non-word spelling correction
 - * Need probability for *no error $P(w \mid w)$: This depends strongly on the application