Introduction to Natural Language Processing

a course taught as B4M36NLP at Open Informatics

by members of the Institute of Formal and Applied Linguistics

Today: Week 2, lecture
Today’s topic: Language Modelling & The Noisy Channel Model
Today’s teacher: Jan Hajič
E-mail: hajic@ufal.mff.cuni.cz
WWW: http://ufal.mff.cuni.cz/jan-hajic
The Noisy Channel

• Prototypical case:

Input: 0,1,1,1,0,1,0,1,...  
Output (noisy): 0,1,1,0,0,1,1,0,...

The channel (adds noise)

• Model: probability of error (noise):

Example: p(0|1) = .3  p(1|1) = .7  p(1|0) = .4  p(0|0) = .6

• The Task:

known: the noisy output; want to know: the input (decoding)
Noisy Channel Applications

• OCR
  - straightforward: text → print (adds noise), scan → image

• Handwriting recognition
  - text → neurons, muscles (“noise”), scan/digitize → image

• Speech recognition (dictation, commands, etc.)
  - text → conversion to acoustic signal (“noise”) → acoustic waves

• Machine Translation
  - text in target language → translation (“noise”) → source language

• Also: Part of Speech Tagging
  - sequence of tags → selection of word forms → text
Noisy Channel: The Golden Rule of ...

• Recall:

\[
p(A|B) = \frac{p(B|A) \ p(A)}{p(B)} \quad \text{(Bayes formula)}
\]
\[
A_{\text{best}} = \text{argmax}_A p(B|A) \ p(A) \quad \text{(The Golden Rule)}
\]

• \(p(B|A)\): the acoustic/image/translation/lexical model
  – application-specific name
  – will explore later

• \(p(A)\): \textbf{the language model}
The Perfect Language Model

- Sequence of word forms [forget about tagging for the moment]
- Notation: \( A \sim W = (w_1, w_2, w_3, \ldots, w_d) \)
- The big (modeling) question:
  \[ p(W) = ? \]
- Well, we know (Bayes/chain rule →):
  \[ p(W) = p(w_1, w_2, w_3, \ldots, w_d) = \]
  \[ = p(w_1) \times p(w_2|w_1) \times p(w_3|w_1, w_2) \times \ldots \times p(w_d|w_1, w_2, \ldots, w_{d-1}) \]
- Not practical (even short \( W \rightarrow \) too many parameters)
Markov Chain

• Unlimited memory (cf. previous foil):
  – for $w_i$, we know all its predecessors $w_1, w_2, w_3, ..., w_{i-1}$

• Limited memory:
  – we disregard “too old” predecessors
  – remember only $k$ previous words: $w_{i-k}, w_{i-k+1}, ..., w_{i-1}$
  – called “$k^{th}$ order Markov approximation”

• + stationary character (no change over time):
  \[ p(W) \approx \prod_{i=1..d} p(w_i | w_{i-k}, w_{i-k+1}, ..., w_{i-1}), \quad d = |W| \]
n-gram Language Models

• (n-1)th order Markov approximation → n-gram LM:

\[ p(W) = \prod_{i=1..d} p(w_i | w_{i-n+1}, w_{i-n+2}, ..., w_{i-1}) \]

• In particular (assume vocabulary |V| = 60k):
  
  • 0-gram LM: uniform model, \( p(w) = 1/|V| \), 1 parameter
  • 1-gram LM: unigram model, \( p(w) \), \( 6 \times 10^4 \) parameters
  • 2-gram LM: bigram model, \( p(w_i | w_{i-1}) \) \( 3.6 \times 10^9 \) parameters
  • 3-gram LM: trigram model, \( p(w_i | w_{i-2}, w_{i-1}) \) \( 2.16 \times 10^{14} \) parameters
Maximum Likelihood Estimate

• MLE: Relative Frequency...
  – ...best predicts the data at hand (the “training data”)

• Trigrams from Training Data T:
  – count sequences of three words in T: $c_3(w_{i-2}, w_{i-1}, w_i)$
    [NB: notation: just saying that the three words follow each other]
  – count sequences of two words in T: $c_2(w_{i-1}, w_i)$:
    • either use $c_2(y, z) = \sum_w c_3(y, z, w)$
    • or count differently at the beginning (& end) of data! $p(w_i | w_{i-2}, w_{i-1})$

\[
= \text{est. } \frac{c_3(w_{i-2}, w_{i-1}, w_i)}{c_2(w_{i-2}, w_{i-1})}
\]
LM: an Example

• Training data:

\(<s> <s> \text{He can buy the can of soda.} \)

– Unigram: 
  \( p_1(\text{He}) = p_1(\text{buy}) = p_1(\text{the}) = p_1(\text{of}) = p_1(\text{soda}) = p_1(.) = .125 \)
  \( p_1(\text{can}) = .25 \)

– Bigram: 
  \( p_2(\text{He}|<s>) = 1, p_2(\text{can}|\text{He}) = 1, p_2(\text{buy}|\text{can}) = .5, \)
  \( p_2(\text{of}|\text{can}) = .5, p_2(\text{the}|\text{buy}) = 1, \ldots \)

– Trigram: 
  \( p_3(\text{He}|<s>,<s>) = 1, p_3(\text{can}|<s>,\text{He}) = 1, \)
  \( p_3(\text{buy}|\text{He,can}) = 1, p_3(\text{of}|\text{the,can}) = 1, \ldots, p_3(.)|\text{of,soda}) = 1. \)

– Entropy: 
  \( H(p_1) = 2.75, H(p_2) = .25, H(p_3) = 0 \quad \leftarrow \text{Great?!} \)
LM: an Example (The Problem)

• Cross-entropy:
• $S = \langle s \rangle \langle s \rangle$ It was the greatest buy of all.
• Even $H_S(p_1)$ fails ($= H_S(p_2) = H_S(p_3) = \infty$), because:
  – all unigrams but $p_1(\text{the})$, $p_1(\text{buy})$, $p_1(\text{of})$ and $p_1(\text{.})$ are 0.
  – all bigram probabilities are 0.
  – all trigram probabilities are 0.
• We want: to make all (theoretically possible*) probabilities non-zero.

*in fact, all: remember our graph from day 1?
LM Smoothing
(And the EM Algorithm)
Why do we need Nonzero Probs?

• To avoid infinite Cross Entropy:
  – happens when an event is found in test data which has not been seen in training data
    \[ H(p) = \infty: \text{prevents comparing data with } > 0 \text{ "errors"} \]
• To make the system more robust
  – low count estimates:
    • they typically happen for “detailed” but relatively rare appearances
  – high count estimates: reliable but less “detailed”
Eliminating the Zero Probabilities: Smoothing

- Get new \( p'(w) \) (same \( \Omega \)): almost \( p(w) \) but no zeros
- Discount \( w \) for (some) \( p(w) > 0 \): new \( p'(w) < p(w) \)
  \[ \sum_{w \in \text{discounted}} (p(w) - p'(w)) = D \]
- Distribute \( D \) to all \( w \); \( p(w) = 0 \): new \( p'(w) > p(w) \)
  - possibly also to other \( w \) with low \( p(w) \)
- For some \( w \) (possibly): \( p'(w) = p(w) \)
- Make sure \( \sum_{w \in \Omega} p'(w) = 1 \)
- There are many ways of smoothing
Smoothing by Adding 1

- Simplest but not really usable:
  - Predicting words $w$ from a vocabulary $V$, training data $T$:
    \[ p'(w|h) = \frac{c(h,w) + 1}{c(h) + |V|} \]
  - for non-conditional distributions: $p'(w) = \frac{c(w) + 1}{|T| + |V|}$
  - Problem if $|V| > c(h)$ (as is often the case; even $>> c(h)$!)

- Example: Training data:  
  - $V = \{\text{what, is, it, small, ?, <s>, flying, birds, are, a, bird, . }\}$, $|V| = 12$
  - $p(\text{it}) = .125$, $p(\text{what}) = .25$, $p(.) = 0$
  - $p(\text{what is it?}) = .25^2 \times .125^2 \approx .001$
  - $p(\text{it is flying.}) = .125 \times .25 \times 0^2 = 0$
  - $p'(\text{it}) = .1$, $p'(\text{what}) = .15$, $p'(.) = .05$
  - $p'(\text{what is it?}) = .15^2 \times .1^2 \approx .0002$
  - $p'(\text{it is flying.}) = .1 \times .15 \times .05^2 \approx .00004$
Adding less than 1

• Equally simple:
  – Predicting words \( w \) from a vocabulary \( V \), training data \( T \):
    \[
    p'(w|h) = \frac{c(h,w) + \lambda}{c(h) + \lambda|V|}, \; \lambda < 1
    \]
  • for non-conditional distributions: \( p'(w) = \frac{c(w) + \lambda}{|T| + \lambda|V|} \)

• Example: Training data: <s> what is it what is small ? \(|T| = 8\)
  • \( V = \{ \text{what, is, it, small, ?, <s>, flying, birds, are, a, bird, . } \} \), \(|V| = 12\)
  • \( p(\text{it}) = .125, \; p(\text{what}) = .25, \; p(.) = 0 \)
  \( p(\text{what is it?}) = .25^2 \times .125^2 \approx .001 \)
  \( p(\text{it is flying.}) = .125 \times .25 \times 0^2 = 0 \)
  • Use \( \lambda = .1 \):
  • \( p'(\text{it}) \approx .12, \; p'(\text{what}) \approx .23, \; p'(.) \approx .01 \)
  \( p'(\text{what is it?}) = .23^2 \times .12^2 \approx .0007 \)
  \( p'(\text{it is flying.}) = .12 \times .23 \times .01^2 \approx .000003 \)
Smoothing by Combination: Linear Interpolation

• Combine what?
  • distributions of various level of detail vs. reliability

• n-gram models:
  • use (n-1)gram, (n-2)gram, ..., uniform

  ___________________________reliability
  ___________________________detail

• Simplest possible combination:
  – sum of probabilities, normalize:
    • p(0|0) = .8, p(1|0) = .2, p(0|1) = 1, p(1|1) = 0, p(0) = .4, p(1) = .6:
    • p'(0|0) = .6, p'(1|0) = .4, p'(0|1) = .7, p'(1|1) = .3
Typical n-gram LM Smoothing

• Weight in less detailed distributions using $\lambda=(\lambda_0, \lambda_1, \lambda_2, \lambda_3)$:

$$p'_{\lambda}(w_i | w_{i-2}, w_{i-1}) = \lambda_0 p_3(w_i | w_{i-2}, w_{i-1}) + \lambda_1 p_2(w_i | w_{i-1}) + \lambda_2 p_1(w_i) + \lambda_3 / |V|$$

• Normalize:

$\lambda_i > 0$, $\sum_{i=0..n} \lambda_i = 1$ is sufficient ($\lambda_0 = 1 - \sum_{i=1..n} \lambda_i$) (n=3)

• Estimation using MLE:

  – fix the $p_3$, $p_2$, $p_1$ and $|V|$ parameters as estimated from the training data
  – then find such $\{\lambda_i\}$ which minimizes the cross entropy (maximizes probability of data): $-(1/|D|) \sum_{i=1..|D|} \log_2(p'_{\lambda}(w_i|h_i))$
Held-out (Cross-validation) Data

• What data to use?
  – try the training data $T$: but we will always get $\lambda_3 = 1$
    • why? (let $p_{iT}$ be an $i$-gram distribution estimated using r.f. from $T$)
    • minimizing $H_T(p'_{\lambda})$ over a vector $\lambda$, $p'_{\lambda} = \lambda_3 p_{3T} + \lambda_2 p_{2T} + \lambda_1 p_{1T} + \lambda_0 / |V|$
      – remember: $H_T(p'_{\lambda}) = H(p_{3T}) + D(p_{3T} || p'_{\lambda})$
        • ($p_{3T}$ fixed $\rightarrow$ $H(p_{3T})$ fixed, best)
      – which $p'_{\lambda}$ minimizes $H_T(p'_{\lambda})$? ... a $p'_{\lambda}$ for which $D(p_{3T} || p'_{\lambda}) = 0$
      – ...and that’s $p_{3T}$ (because $D(p||p) = 0$, as we know).
      – ...and certainly $p'_{\lambda} = p_{3T}$ if $\lambda_3 = 1$ (maybe in some other cases, too).
        • ($p'_{\lambda} = 1 \times p_{3T} + 0 \times p_{2T} + 0 \times p_{1T} + 0 / |V|$)
    – thus: do not use the training data for estimation of $\lambda$!
      • must hold out part of the training data (heldout data, $H$):
      • ...call the remaining data the (true/raw) training data, $T$
      • the test data $S$ (e.g., for comparison purposes): still different data!
The Formulas

- Repeat: minimizing $-(1/|H|)\sum_{i=1..|H|}\log_2(p_\lambda'(w_i|h_i))$ over $\lambda$

  $$p_\lambda'(w_i|h_i) = p_\lambda'(w_i|w_{i-2},w_{i-1}) = \lambda_3 p_3(w_i|w_{i-2},w_{i-1}) + \lambda_2 p_2(w_i|w_{i-1}) + \lambda_1 p_1(w_i) + \lambda_0 /|V|$$

- “Expected Counts (of lambda)”: $j = 0..3$

  $$c(\lambda_j) = \sum_{i=1..|H|} (\lambda_j p_j(w_i|h_i) / p_\lambda(w_i|h_i))$$

  $$\lambda_{i,\text{next}} = c(\lambda_j) / \sum_{k=0..3} (c(\lambda_k))$$

E-step

M-step
The (Smoothing) EM Algorithm

1. Start with some $\lambda$, such that $\lambda_j > 0$ for all $j \in 0..3$.
2. Compute “Expected Counts” for each $\lambda_j$.
3. Compute new set of $\lambda_j$, using the “Next $\lambda$” formula.
4. Start over at step 2, unless a termination condition is met.
   • Termination condition: convergence of $\lambda$.
     – Simply set an $\varepsilon$, and finish if $|\lambda_j - \lambda_{j,\text{next}}| < \varepsilon$ for each $j$ (step 3).
   • Guaranteed to converge:
     follows from Jensen’s inequality, plus a technical proof.
Remark on Linear Interpolation Smoothing

• “Bucketed” smoothing:
  – use several vectors of $\lambda$ instead of one, based on (the frequency of) history: $\lambda(h)$
  • e.g. for $h = \text{(micrograms, per)}$ we will have
    \[ \lambda(h) = (0.999, 0.0009, 0.00009, 0.00001) \]
    (because “cubic” is the only word to follow...)
  – actually: not a separate set for each history, but rather a set for “similar” histories (“bucket”):
    \[ \lambda(b(h)), \text{ where } b : V^2 \rightarrow N \text{ (in the case of trigrams)} \]
    $b$ classifies histories according to their reliability ($\sim$ frequency)
Bucketed Smoothing: The Algorithm

• First, determine the bucketing function $b$ (use heldout!):
  – decide in advance you want e.g. 1000 buckets
  – compute the total frequency of histories in 1 bucket ($f_{\text{max}}(b)$)
  – gradually fill your buckets from the most frequent bigrams so that the sum of frequencies does not exceed $f_{\text{max}}(b)$ (you might end up with slightly more than 1000 buckets)

• Divide your heldout data according to buckets

• Apply the previous algorithm to each bucket and its data
Simple Example

- Raw distribution (unigram only; smooth with uniform):
  \[ p(a) = .25, \quad p(b) = .5, \quad p(\alpha) = \frac{1}{64} \text{ for } \alpha \in \{c..r\}, \quad = 0 \text{ for the rest: } s,t,u,v,w,x,y,z \]

- Heldout data: baby; use one set of \( \lambda \) (\( \lambda_1 \): unigram, \( \lambda_0 \): uniform)

- Start with \( \lambda_1 = .5 \): \[ p'_{\lambda}(b) = .5 \times .5 + .5 / 26 = .27 \]

  \[ p'_{\lambda}(a) = .5 \times .25 + .5 / 26 = .14 \]

  \[ p'_{\lambda}(y) = .5 \times 0 + .5 / 26 = .02 \]

\[ c(\lambda_1) = .5 \times .5 / .27 + .5 \times .25 / .14 + .5 \times .5 / .27 + .5 \times 0 / .02 = 2.72 \]

\[ c(\lambda_0) = .5 \times .04 / .27 + .5 \times .04 / .14 + .5 \times .04 / .27 + .5 \times .04 / .02 = 1.28 \]

  Normalize: \( \lambda_{1,\text{next}} = .68, \quad \lambda_{0,\text{next}} = .32. \)

  Repeat from step 2 (recompute \( p'_{\lambda} \) first for efficient computation, then \( c(\lambda_i), \ldots \))

Finish when new lambdas almost equal to the old ones (say, < 0.01 difference).