Introduction to Natural Language Processing I [Statistické metody zpracování přirozených jazyků I] (NPFL067)

http://ufal.mff.cuni.cz/courses/npfl067

prof. RNDr. Jan Hajič, Dr. / doc. RNDr. Pavel Pecina, Ph.D.
ÚFAL MFF UK
{hajic,pecina}@ufal.mff.cuni.cz
http://ufal.mff.cuni.cz/jan-hajic
http://ufal.mff.cuni.cz/~pecina/index.html

LM Smoothing
(And the EM Algorithm)

The Zero Problem

- "Raw" n-gram language model estimate:
  - necessarily, some zeros
    - **!many: trigram model → 2.16×10^{14} parameters, data ~ 10^9 words**
  - which are true 0?
    - **optimal situation: even the least frequent trigram would be seen several times, in order to distinguish it's probability vs. other trigrams**
    - **optimal situation cannot happen, unfortunately (open question: how many data would we need?)**
  - → we don't know
  - we must eliminate the zeros
- Two kinds of zeros: $p(w|h) = 0$, or even $p(h) = 0$!

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

    $H(p) = \infty$: prevents comparing data with > 0 "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 Ω): almost p(w) but no zeros
- Discount w for (some) p(w) > 0: new p'(w) < p(w)

$$\Sigma_{w \in \text{discounted}} (p(w) - p'(w)) = D$$

- 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 $\Sigma_{w \in \Omega} p'(w) = 1$
- 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:

    $p'(w|h) = (c(h,w) + 1) \ / \ (c(h) + |V|)$

    - **for non-conditional distributions: $p'(w) = (c(w) + 1) / (|T| + |V|)$**
  - Problem if $|V| > c(h)$ (as is often the case; even $>> c(h)$!)
- Example: Training data: <s> what is it what is small ? $|T| = 8$
  - **V = { what, is, it, small, ?, <s>, flying, birds, are, a, bird, . }, |V| = 12**
  - **p(it)=.125, p(what)=.25, p(.)=0 p(what is it?) = $.25^2 \times .125^2 \cong$ .001**

    p(it is flying.) = $.125 \times .25 \times 0^2 = 0$
  - **p'(it) =.1, p'(what) =.15, p'(.)=.05 p'(what is it?) = $.15^2 \times .1^2 \cong$ .0002**

    p'(it is flying.) = $.1 \times .15 \times .05^2 \cong .00004$

Adding *less than* 1

- Equally simple:
  - Predicting words w from a vocabulary V, training data T:

    $p'(w|h) = (c(h,w) + \lambda) \ / \ (c(h) + \lambda|V|), \lambda < 1$

    - **for non-conditional distributions: $p'(w) = (c(w) + \lambda) / (|T| + \lambda|V|)$**
- Example: Training data: <s> what is it what is small ? $|T| = 8$
    - **V = { what, is, it, small, ?, <s>, flying, birds, are, a, bird, . }, |V| = 12**
    - **p(it)=.125, p(what)=.25, p(.)=0 p(what is it?) = $.25^2 \times .125^2 \cong$ .001**

      p(it is flying.) = $.125 \times .25 \times 0^2 = 0$
    - **Use $\lambda = .1$:**
    - **p'(it)$\cong$ .12, p'(what)$\cong$ .23, p'(.)$\cong$ .01 p'(what is it?) = $.23^2 \times .12^2 \cong$ .0007**

      p'(it is flying.) = $.12 \times .23 \times .01^2 \cong$ .000003

Good - Turing

- Suitable for estimation from large data
  - similar idea: discount/boost the relative frequency estimate:

    $p_r(w) = (c(w) + 1) \times N(c(w) + 1) / (|T| \times N(c(w)))$,

    where N(c) is the count of words with count c (count-of-counts)

    specifically, for $c(w) = 0$ (unseen words), $p_r(w) = N(1) / (|T| \times N(0))$
  - good for small counts (< 5-10, where N(c) is high)
  - variants (see MS)
  - normalization! (so that we have $\Sigma_w\, p'(w) = 1$)

Good-Turing: An Example

- Example: remember: $p_r(w) = (c(w) + 1) \times N(c(w) + 1) / (|T| \times N(c(w)))$

  Training data: <s> what is it what is small ? $|T| = 8$

  - **V = { what, is, it, small, ?, <s>, flying, birds, are, a, bird, . }, |V| = 12**

    p(it)=.125, p(what)=.25, p(.)=0 p(what is it?) = $.25^2 \times .125^2 \cong$ .001

    p(it is flying.) = $.125 \times .25 \times 0^2 =$ 0

  - **Raw reestimation (N(0) = 6, N(1) = 4, N(2) = 2, N(i) = 0 for i > 2):**

    $p_r(it) = (1+1) \times N(1+1)/(8 \times N(1)) = 2 \times 2/(8 \times 4) = .125$

    $p_r(what) = (2+1) \times N(2+1)/(8 \times N(2)) = 3 \times 0/(8 \times 2) = 0$: keep orig. p(what)

    $p_r(.) = (0+1) \times N(0+1)/(8 \times N(0)) = 1 \times 4/(8 \times 6) \cong .083$

  - **Normalize (divide by $1.5 = \Sigma_{w \in |V|} p_r(w)$) and compute:**

    p'(it)$\cong$ .08, p'(what)$\cong$ .17, p'(.)$\cong$ .06 p'(what is it?) = $.17^2 \times .08^2 \cong$ .0002

    p'(it is flying.) = $.08 \times .17 \times .06^2 \cong$ .00004

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**

$\longrightarrow$ reliability

$\longleftarrow$ 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 $\lambda=(\lambda_0,\lambda_1,\lambda_2,\lambda_3)$:

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

- Normalize:

  $\lambda_i > 0,\ \Sigma_{i=0..n} \lambda_i = 1$ is sufficient ($\lambda_0 = 1 - \Sigma_{i=1..n} \lambda_i$) (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): $-(1/|D|)\Sigma_{i=1..|D|}\log_2(p'_\lambda(w_i|h_i))$

Held-out 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 λ, $p'_\lambda = \lambda_3 p_{3T} + \lambda_2 p_{2T} + \lambda_1 p_{1T} + \lambda_0/|V|$**
      - remember: $H_T(p'_\lambda) = H(p_{3T}) + D(p_{3T} || p'_\lambda)$;
        - (p_{3T} fixed $\rightarrow$ $H(p_{3T})$ fixed, best)
      - which p'_λ minimizes $H_T(p'_\lambda)$? ... a p'_λ 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 <u>not use the training data for estimation of λ!</u>
    - **must hold out part of the training data (*heldout* data, <u>H</u>):**
    - **...call the remaining data the (true/raw) *training* data, <u>T</u>**
    - **the *test* data <u>S</u> (e.g., for comparison purposes): still different data!**

The Formulas

- Repeat: minimizing $-(1/|H|)\Sigma_{i=1..|H|}\log_2(p'_\lambda(w_i|h_i))$ over λ

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

- "Expected Counts (of lambdas)": j = 0..3

$$\boxed{c(\lambda_j) = \Sigma_{i=1..|H|}\, (\lambda_j p_j(w_i|h_i) / p'_\lambda(w_i|h_i))} \quad !$$

- "Next λ": j = 0..3

$$\boxed{\lambda_{j,next} = c(\lambda_j) / \Sigma_{k=0..3}\, (c(\lambda_k))} \quad !$$

The (Smoothing) EM Algorithm

1. Start with some λ, such that $\lambda_j > 0$ for all $j \in 0..3$.
2. Compute "Expected Counts" for each λ_j.
3. Compute new set of λ_j, using the "Next λ" formula.
4. Start over at step 2, unless a termination condition is met.

- Termination condition: convergence of λ.
  - Simply set an ε, and finish if $|\lambda_j - \lambda_{j,next}| < \varepsilon$ 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 λ instead of one, based on (the frequency of) history: $\lambda(h)$
    - **e.g. for h = (micrograms,per) we will have**

      $\lambda(h) = (.999,.0009,.00009,.00001)$

      (because "cubic" is the only word to follow...)
  - actually: not a separate set for each history, but rather a set for "similar" histories ("bucket"):

    $\lambda(b(h))$, where b: $V^2 \rightarrow N$ (in the case of trigrams)

    b classifies histories according to their reliability (~ 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 ($f_{max}(b)$)
  - gradually fill your buckets from the most frequent bigrams so that the sum of frequencies does not exceed $f_{max}(b)$ (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):

  $p(a) = .25$, $p(b) = .5$, $p(\alpha) = 1/64$ for $\alpha \in \{c..r\}$, $= 0$ for the rest: s,t,u,v,w,x,y,z
- Heldout data: <u>baby</u>; use one set of λ (λ_1: unigram, λ_0: uniform)
- Start with $\lambda_1 = .5$; $p'_\lambda(b) = .5 \text{ x } .5 + .5 / 26 = .27$

  $p'_\lambda(a) = .5 \text{ x } .25 + .5 / 26 = .14$

  $p'_\lambda(y) = .5 \text{ x } 0 + .5 / 26 = .02$

$c(\lambda_1) = .5\text{x}.5/.27 + .5\text{x}.25/.14 + .5\text{x}.5/.27 + .5\text{x}0/.02 = 2.72$

$c(\lambda_0) = .5\text{x}.04/.27 + .5\text{x}.04/.14 + .5\text{x}.04/.27 + .5\text{x}.04/.02 = 1.28$

Normalize: $\lambda_{1,next} = .68$, $\lambda_{0,next} = .32$.

Repeat from step 2 (recompute p'_λ first for efficient computation, then $c(\lambda_i)$, ...)

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

Some More Technical Hints

- Set V = {all words from training data}.
  - **You may also consider $V = T \cup H$, but it does not make the coding in any way simpler (in fact, harder).**
  - **But: you must *never* use the test data for you vocabulary!**
- Prepend two "words" in front of all data:
  - **avoids beginning-of-data problems**
  - **call these index -1 and 0: then the formulas hold exactly**
- When $c_n(w,h) = 0$:
  - **Assign 0 probability to $p_n(w|h)$ where $c_{n-1}(h) > 0$, but a uniform probability ($1/|V|$) to those $p_n(w|h)$ where $c_{n-1}(h) = 0$ [this must be done both when working on the heldout data during EM, as well as when computing cross-entropy on the test data!]**