

class-based n-gram models of natural language

Brown et al.

"his facility with some of its concepts to fall into disrepute may profit"
"allies in the struggle to tease from words their linguistic secrets"

N-gram language models:

$$P(w_1^n) = P(w_1) \cdot P(w_2|w_1) \cdots P(w_n|w_1^{n-1})$$

assume that $P(w_n|w_1^{n-1}) = P(w_n|w_{k-n+1}^{k-1})$ so only on previous n-1 words

- 1-gram $V-1$ indep. param.
- 2-gram $P(w_n|w_{n-1})$ $(V-1) \cdot V + (V-1) = V^2 - 1$
- n-gram $V^{n-1} \cdot (V-1)$ for $P(w_n|w_{k-n+1}^{k-1}) + V^{n-1} - 1$ for (n-1)-gram model
- note: no smoothing done yet.

$C(w_1, w_2, \dots, w_n)$ is count of n-gram $w_{1:n}$ in a text t_1^T

"(position matters, so $C(w_1, w_2) \neq C(w_2, w_1)$ "
"not nec. equal")

should use $C_{t_1^T}(w_{1:n})$ i.e., count is always w.r.t. a training data

In general, Maximum Likelihood estimate of parameters is:

$$P(w_n|w_1^{n-1}) = \frac{C(w_1^{n-1}, w_n)}{\sum_w C(w_1^{n-1}, w)} \quad P(w_n) = \frac{C(w_n)}{N} \quad \text{since } \sum_w C(w) = N$$

Problem: too many parameters

- 20k words with a 3-gram $\Rightarrow 8 \times 10^9$ parameters
- Describe Table 1 on slide:
- main problem: Reliability:
as n increases, # of grams that have low counts increases,
- we want high counts of everything, need more training data
 (Lack of dimensionality)

- Partial Interpolation to human robustness

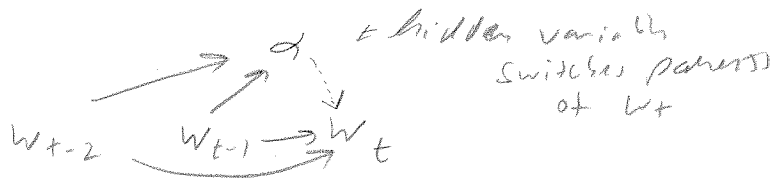
$$\hat{P}(w_i | w_{1:i-1}) = \sum_j \lambda_j(w_{1:i-1}) P_{r, ML}^{(j)}(w_i | w_{1:i-1})$$

- mix between diff ML LMS,

choose weights based on counts of w_i^{i-1} $c(w_i^{i-1})$

if $c(w_i^{i-1})$ is higher, oh to weight that LM higher for that context.

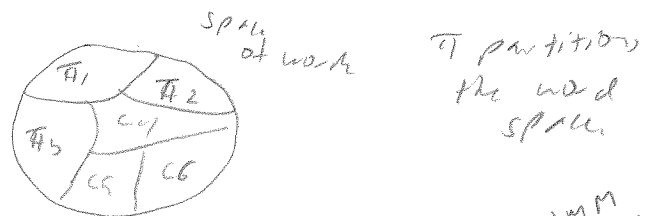
- learn using **EM** (hidden variables, switching networks)



- Backoff approach.

- class-based LMS. Mapping $\pi(w) = c$

form



Assumptions:

A) Each word has only 1 class, & does not live in > 1 class. (so no class overlap). classes are determined by word.

B) classes have multiple words.

Ex: bigram class LM derivation

HMM independence assumptions

$$P(w_t | w_{t-1}) = \sum_{c_t} P(w_t, c_t | w_{t-1})$$

$c_{t+1} = c_t$
 $w_{t+1} = w_t$

$$= \sum_{c_t} P(w_t | c_t, w_{t-1}) \cdot P(c_t | w_{t-1})$$

$$= \sum_{c_t} P(w_t | c_t) \cdot \sum_{c_{t-1}} P(c_t | c_{t-1}, w_{t-1}) P(c_{t-1} | w_{t-1})$$

$$= \sum_{c_t} P(w_t | c_t) \cdot \sum_{c_{t-1}} P(c_t | c_{t-1}) P(c_{t-1} | w_{t-1})$$

0 when $c_t \neq \pi(w_t)$

$\mathbb{1}\{c_{t-1} = \pi(w_{t-1})\}$

$$= P(w_t | \pi(w_t)) \cdot P(\pi(w_t) | \pi(w_{t-1}))$$

- standard class-based language model.
- note same for other histories; just c_t .

parameters.

for $P(w_t | c_t) P(c_t | c_{t-1})$

$$C^{n-1} + C \cdot (V-1) + (V-1)$$

$$= C^{n-1} + CV - C + V - 1$$

but since

$P(w | c)$ looks like

only a total of

$V-C$



tot.:

$$C^{n-1} + V - C \ll V^n - 1$$

parameter estimates for these: given training data t_1^T

$$P(w|c) = \frac{C(w)}{C(c)} \quad C(c) = \text{count of classes, \# of words that have class } c$$

$$= C(c) = \left| \sum_w : \pi(w) = c, w \in t_1^T \right|$$

$$P_c(c) = \frac{C(c)}{T}$$

\Rightarrow class-based model: $P(w) = P(w|c)P(c) = \frac{C(w)}{T}$

unigram \rightarrow

So for unigram, class-based model is same as with out class, \forall classes (only of class)

bi-gram: ML-estimate: $P(c_2|c_1) = \frac{C(c_1, c_2)}{\sum_c C(c_1, c)}$

$\therefore P(c_1, c_2) = P(c_1)P(c_2|c_1) = \frac{C(w)}{T} \cdot \frac{C(c_1, c_2)}{\sum_c C(c_1, c)} = \frac{C(c_1, c_2)}{T} \cdot \frac{C(w)}{\sum_c C(c_1, c)}$

s. $P(c_1, c_2) \rightarrow$ true relation freq. as $T \rightarrow \infty = P(c_1, c_2)$

Goal: Find classes that keeps the (log) probability as high as possible (maximum likelihood setting) of training set

$$L(\pi) = \frac{\log P(t_1^T | \pi)}{T-1} = \frac{1}{T-1} \log \prod_{j=2}^T P_c(t_j | t_{j-1})$$

$$P(t_j | t_{j-1}) = P(t_j | \pi(t_j)) \cdot P(\pi(t_j) | \pi(t_{j-1}))$$

let $c_j = \pi(t_j)$

our estimate of $P(w_2)$, irrespective of class

$$\rightarrow = \sum_{w_1, w_2} \frac{C(w_1, w_2)}{T-1} \log \left[\frac{P(c_2|c_1) P(w_2|c_2) P(c_2)}{P(c_2)} \right]$$

$$= \sum_{c_1, c_2} \frac{C(c_1, c_2)}{T-1} \log \frac{P(c_2|c_1)}{P(c_2)} + \sum_{w_1, w_2} \frac{C(w_1, w_2)}{T-1} \log P(w_2)$$

$$= \sum_{c_1, c_2} \frac{C(c_1, c_2)}{T-1} \log \frac{P(c_2|c_1) P(c_1)}{P(c_2) P(c_1)} + \sum_{w_2} \frac{\sum_{w_1} C(w_1, w_2)}{T-1} \log P(w_2)$$

Complexity of "Avg" MS, $O(V^2)$ cost

$$\equiv \sum_{c_1, c_2} P(c_1, c_2) \log \frac{P(c_1, c_2)}{P(c_1)} + \sum_w P(w) \log P(w)$$

$$= I(c_1; c_2) - H(w)$$

So likelihood under word clustering π corresponds to MI. Increase MS of clustering, and we increase likelihood.

This is a discrete optimization.

Find $\pi^* = \underset{\pi}{\text{argmax}} I_{\pi}(c_1; c_2)$ hard.

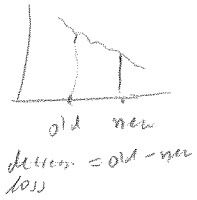
Greedy Strategy, Agglomerative clustering strategy (standard) (Jain & Dubois) decreasing # of clusters at stage k , for $k=V$ down to C (k clusters at stage k)

- Agglomerative clustering**
- 1) Start with V clusters with 1 word to each cluster
 - 2) find a pair of clusters the merger of which leads to the least loss of information.

$$L_k(i, j) = \frac{\text{decrease}}{\text{old}} \text{loss at stage } k \text{ of merging clusters } i \text{ and } j$$

$$= I_k - I_k(i, j)$$

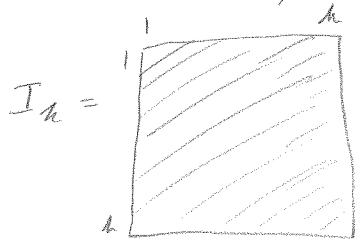
MS at stage k MS after merge i, j gain = new - old
loss = old - new
 - 3) goto 2 until no further loss is found.
- Fixup**
- 4) Iterate through all words and reassign that word to some other cluster if it increases avg MI.
 - 5) repeat 4 until no further increase



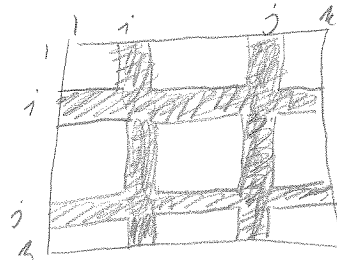
Dynamic Alg for steps 1, 2, and 3
 Repeat $V-C$ times ($k=V$ to C) $O(V)$
 evaluate all $O(V^2)$ potential merges
 To Evaluate a merge - recompute Avg MI, cost $O(V^2)$ } $\Rightarrow O(V^5)$ bad.

Evaluation of A_{21} can be much faster than $O(N^2)$

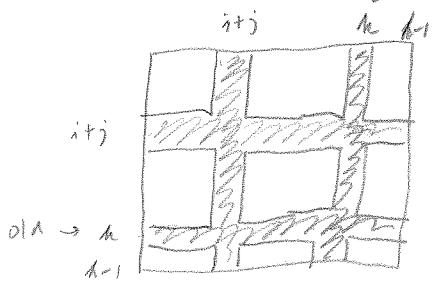
Consider I_k , sum over a $V \times V$ table, in sum to I_{k-1}



to merge class i and j
 subtract out contribution from these classes

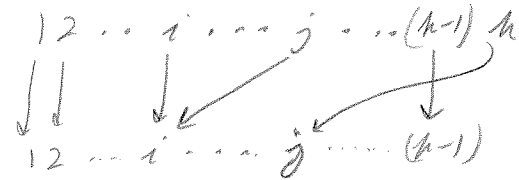


add back in contribution of new class, $i+j$



old (k -class)

new ($k-1$ class)



Equations:

$P_k(l, m) \triangleq P(C_k(l), C_k(m))$, prob. that word class

$C_k(m)$ follows $C_k(l)$ at k 's stage.

$$P_k^l(l) \triangleq \sum_m P_k(l, m) \quad P_k^m(m) = \sum_l P_k(l, m)$$

$$q_k(l, m) = P_k(l, m) \log \frac{P_k(l, m)}{P_k^l(l) P_k^m(m)} \quad I_k = \sum_{l, m} q_k(l, m)$$

prob. after a merge of i and j ($i+j$)

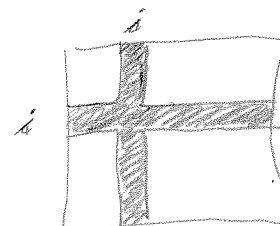
$$P_k(i+j, m) = \frac{q_k(i+j, m)}{T} = \frac{q_k(i, m) + q_k(j, m)}{T} = P_k(i, m) + P_k(j, m)$$

$$q_k(i+j, m) = P_k(i+j, m) \log \frac{P_k(i+j, m)}{P_k(i+j) P_k^m(m)}$$

Define partial computations

$$S_k(i) \triangleq \sum_l q_k(l, i) + \sum_m q_k(i, m) - q_k(i, i)$$

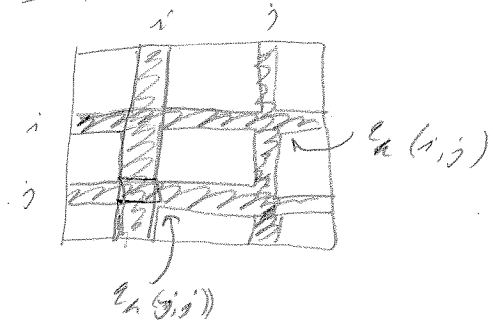
~~col.~~ col. row avoid double counting



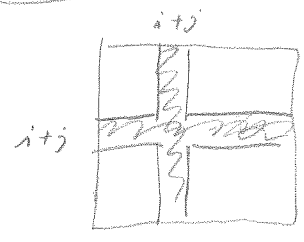
old as old stuff
 then $S_n(i) + S_n(j) - z_n(i,j) - z_n(j,i) \stackrel{\Delta}{=} f_n(i,j)$ 26

cross pattern

avoid double counting



New JVH.



$$g_n(i+j) \stackrel{\Delta}{=} z_n(i+j, i+j) + \sum_{l \neq i+j} z_n(l, i+j) + \sum_{m \neq i+j} z_n(i+j, m)$$

Difference:

$$I_n(1,j) = I_n - f_n(i,j) + g_n(i+j) \quad (\text{eq. 15 in paper})$$

calculating this is $O(V)$ rather than $O(V^2)$

- further optimization: only sum over pairs that are not zero (see table), many terms bi-grams with non-zero probabilities.

rather than 6×10^{10} 2-grams only 14M occur in training data.

claim a paper is that around 56 terms occur $\rightarrow O(1)$

\rightarrow Alg: $O(V^5) \rightarrow O(V^3)$

Results: word trees, 5000 word VOCAB

Fig. 2, shows related words in Title.

When $V > 5000$, run alg. sequentially, sorting words by prob. At each stage, add a new class consisting of next-most prob. word, & do a merge step, retaining best classes along the way.

Sequential Algorithm used for 260k vocab word set.

=> Table 2. 1000 classes (Table 3 random classes)

Avg. class size = 260

show on word with count > 10, or size < 10 words.

- both syntactic & semantic categories are discovered.

- the threat that some class, misspellings of that.

- Table 4: word freq. counts.

fewer grams with low counts.

got small perplexity reduction. 244 - 236

$$P(S) = \frac{1}{|S|}$$

Sticky pairs & semantic class

- $w_1 w_2$ "sticky" if $\log \frac{P(w_1, w_2)}{P(w_1) P(w_2)} \gg 0$

(i.e., if w_2 follows w_1 more often than if they were unrelated)

Table 5 Humpty Dumpty
Kilux Kilan, etc.

- Semantic classes: (Semantically sticky)

$$C_S(w_1, w_2) = \left\{ (w_1, w_2) : \frac{1}{5} < \text{distance}(w_1, w_2) < 1001 \right\}$$

$$P_{near}(w_1, w_2) \propto C_S(w_1, w_2)$$