

Language Modeling Using n -Grams
and
Solutions to the Problem of Data Sparseness

Based on
“Foundations of Statistical NLP” by C. Manning & H. Schütze, ch. 9
MIT Press, 2002

Language Modeling

Problem:

Predict the next word in a (written or spoken) text, given the previous words.

$$P(w_n \mid w_{n-1}, \dots, w_1)$$

Applications:

Optical Character Recognition (OCR)

Speech Recognition

Named Entity Recognition (NER)

Machine Translation (MT)

Spelling Correction

***n*-Grams: Discrimination vs. Reliability**

- **Example:**

Sue swallowed the *large green* ...

(*pill*, but not *car*, *tree*, *mountain*).

- *n* should be sufficiently large — for good **discrimination** but not too large — for statistical **reliability**.

E.g., for 20k words, there are 20K unigrams, 400M bigrams, 8000G trigrams, 16×10^{16} 4-grams, etc.

- **Alternatives:**

Use stems instead of words, semantic classes (from a thesaurus), or more sophisticated models (e.g., use the previous word and the previous predicate).

- **In practice**, although not taking into account the structure of the sentences, 3-gram models are very good.

Maximum Likelihood (ML) Estimates using relative frequencies

- P_{MLE} gives the data the highest likelihood:

$$P_{MLE}(w_1 \dots w_n) = \frac{C(w_1 \dots w_n)}{N}$$

$$P_{MLE}(w_n \mid w_1 \dots w_{n-1}) = \frac{C(w_1 \dots w_n)}{C(w_1 \dots w_{n-1})}$$

- Example:

The dean *comes accross* as an arrogant windbag.

Any rescue worker *comes accross* more human suffering than I can imagine.

The hero of the tale *comes accross* a new chalange every chapter.

That paper *comes accross* more as wishful thinking than real research.

P_{MLE} (as | comes, accross) = ... P_{MLE} (more | comes, accross) = ...

P_{MLE} (a | comes, accross) = ... P_{MLE} (X | comes, accross) = 0

The Zero-Frequency Problem

MLE assigns 0 probability to events unseen in the training data.

- Example (continued):

$$P_{MLE}(\text{the} \mid \text{came, accross}) = 0$$

$$P_{MLE}(\text{some} \mid \text{came, accross}) = 0$$

- Solution (*discounting, smooting*):

Redistribute the probability mass:

Discounting from seen; distribute over unseen.

Smoothing Methods:

1. Laplace, Lidstone, Jeffreys-Perks laws
2. Held-out estimation
3. Cross-validation (deleted estimation)
N-way cross-validation (Leaving-One-Out)
4. Good-Turing estimation

Combining Estimators:

1. Simple linear interpolation
2. Katz's backing-off
3. General linear interpolation

Laplace's Law (“Add One”)

$$P_{Lap}(x) = \frac{C(x) + 1}{N + B}$$

N is the number of training tokens, and B is the number of eventual (“virtual”) types

Notes:

1. This is a Bayes estimator, assuming a prior uniform probability on unseen events.
2. For B far larger than N , too much of the probability mass/space is given to unseen events.

Lidstone's Law

$$P_{Lid}(x) = \frac{C(X) + \lambda}{N + B\lambda} = \mu \frac{C(x)}{N} + (1 - \mu) \frac{1}{B}, \text{ with } \mu = \frac{N}{N + B\lambda}$$

Notes:

1. Wee need a good guess for λ .
2. For $\lambda = \frac{1}{2}$ (quite common), we get the Jeffreys-Perks low.
3. P_{Lid} is linear in the MLE frequency ($\frac{C(x)}{N}$), which is not good for low frequencies.

Held-out Estimation [Jelinek & Mercer, 1985]

Devide the training data into *kept* data and *held-out* data.

For each n -gram $w_1 \dots w_n$, let:

$C_1(w_1 \dots w_n)$ = number of times $w_1 \dots w_n$ occurs in the training data

$C_2(w_1 \dots w_n)$ = number of times $w_1 \dots w_n$ occurs in the held-out data

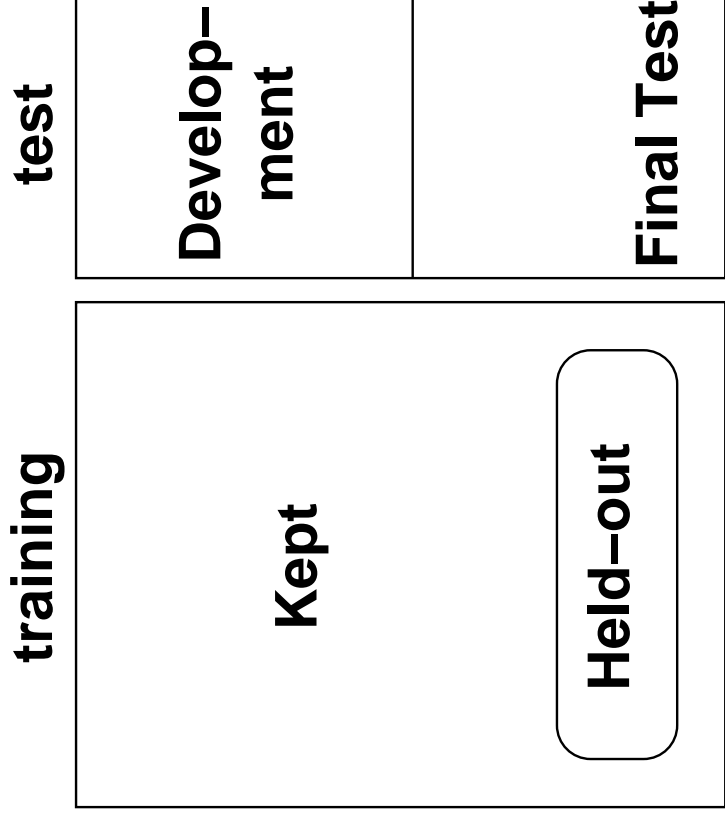
N_r = number of n -grams with frequency r in the training text

$$T_r = \sum_{\{w_1 \dots w_n : C_1(w_1 \dots w_n) = r\}} C_2(w_1 \dots w_n)$$

$$P_{ho}(w_1 \dots w_n) = \frac{T_r}{N_r T}$$

were T is the number of tokens in the held-out data, and $C_1(w_1 \dots w_n) = r$.

A Typical Pattern in Developing Systems for Statistical NLP



The t -Test for Comparing the Performance of two Systems

1. Compute the mean μ_i and the variance s_i^2 , when dividing the data into n parts.
2. compute the t -value $t = \frac{\mu_1 - \mu_2}{\sqrt{\frac{s^2}{n}}}$, with $s^2 = \frac{s_1^2 + s_2^2}{n-1}$.
3. Find C , the confidence level corresponding to the above-computed t -value in the table, considering $2(n-1)$ degrees of freedom.

See

[Dietrich, 1998] for a systematic discussion,
[Mooney, 1996], a case study for Word Sense Disambiguation.

Cross-Validation (Deleted Estimation)

Divide the training data into 2 parts.

Use each of these parts (in turn) for both kept and held-out data.

$$P_{cv}(w_1 \dots w_n) = \frac{T_r^{01} + T_r^{10}}{N(N_r^0 + N_r^1)} \quad (\text{with } C(w_1 \dots w_n) = r)$$

where

N = number of tokens in the whole training data

N_r^0, N_r^1 = number of tokens occurring r times in the first, respectively second part of training data

T_r^{ab} = number of tokens from part a into part b

Note: On large corpora, cross-validation performs better than held-out estimation.

Living-One-Out Estimation

or: N -way Cross-Validation

[Ney, 1997]

1. Divide the training corpus into 2 parts, containing $N - 1$ items and respectively 1 item.
2. Repeat the process N times, each time for a different token.

(See the link with the Good-Turing estimation...)

Simple Linear Interpolation

For instance, for trigrams:

$$P_{li}(w_n | w_{n-1}, w_{n-2}) = \lambda_1 P_1(w_n) + \lambda_2 P_2(w_n | w_{n-1}) + \lambda_3 P_3(w_n | w_{n-1}, w_{n-2})$$

where $0 \leq \lambda_i \leq 1$ and $\sum_i \lambda_i = 1$.

Note: λ_i are set either by hand or automatically, using the EM algorithm or other numerical algorithms (see [Press et al., 1998]).

Back-Off Models

14.

[Katz, 1987]

Different models are consulted in order, depending on their specificity:

$$P_{bo}(w_i \mid w_{i-n+1} \dots w_{i-1}) =$$

$$= \begin{cases} (1 - d_{w_{i-n+1} \dots w_{i-1}}) \frac{w_{i-n+1} \dots w_i}{C(w_{i-n+1} \dots w_{i-1})} & \text{if } C(w_{i-n+1} \dots w_{i-1}) > k \\ \alpha_{w_{i-n+1} \dots w_{i-1}} P_{bo}(w_i \mid w_{i-n+2} \dots w_{i-1}) & \text{otherwise} \end{cases}$$

where

- usually $k = 0$ or $k = 1$
- $d_{w_{i-n+1} \dots w_{i-1}}$ is for smoothing
- $\alpha_{w_{i-n+1} \dots w_{i-1}}$ is a normalization factor.
(In particular, if $C(w_{i-n+1} \dots w_{i-1}) = 0$ then $\alpha_{w_{i-n+1} \dots w_{i-1}} = 1$.)

General Linear Interpolation

$$P_{Ti}(w | h) = \sum_{i=1}^k \lambda_i(h) P_i(w | h)$$

where $0 \leq \lambda_i(h) \leq 1$ and $\sum_i \lambda_i(h) = 1$.

Note:

In general, the weights $\lambda_i(h)$ are not set according to individual histories (that would worsen the sparse data problem!),

but using some sort of equivalence classes of histories.

E.g., $C(w_{(i-n+1)(i-1)})$ or $\frac{C(w_{(i-n+1)(i-1)})}{|\{w_i: C(w_{(i-n+1)(i)}) > 0\}|}$.

Conclusions

Simple smoothing methods may be appropriate for exploratory studies, but...

Good-Turing estimation, linear interpolation, or back-off represent good current practice.

Active research continues on better ways of combining probability models (in particular for language modeling) and dealing with sparse data.