# NLP Note: Language Models

Lizi Chen

**NLP Challenges:**

1. Sparsity - Common words occur a lot yet many words appear much less often.
2. Long Range Dependencies
3. Multi-Linguality
4. Difficulty of Annotation
5. Ambiguity, Common Sense Reasoning.

| **Words** | **Sequences** | **Models** |
|---|---|---|
| N-gram Models and Smoothing | POS Tagging | Sentiment Analysis |
| Classification and Clustering | Word-Alignments | Summarization |
| | | Generative Models |
| | **Trees** | Discriminative Models |
| **NLP Dev Cycle** | Syntax | Graphical Models |
| Representation Re-design | Semantics | Neural Networks |
| Feature Engineering | Machine Translation | |

## 1 Noisy-Channel Model

Predict a sentence given acoustics, Bayes rule.

$$w^* = \arg\max_w P(w|a)$$

$$\begin{aligned} w^* &= \arg\max_w P(w|a) \\ &= \frac{\arg\max_w P(w,a)}{P(a)} \\ &= \frac{\arg\max_w P(a|w)P(w)}{P(a)} \quad \propto \arg\max_w \underbrace{P(a|w)}_{\text{Acoustic model}} \underbrace{P(w)}_{\text{Language model}} \end{aligned} \tag{1}$$

Example of such Machine Translation approach:
1. Given French sentence f, find English sentence e: $e^* = \arg\max_e P(e|f) = \arg\max_e P(f|e)P(e)$
2. Given characters (with spelling mistakes), find the correct word(s): $\text{correct\_word}^* = P(words|characters) \sim \arg\max_{words} P(characters|words)P(words)$

## 2 Probabilistic Language Models

Assign useful probabilities $P(x)$ to sentences x, i.e., $P$(I saw a van) >> $P$(eyes awe of an).

### 2.1 Naive: Empirical Distribution Over Training Sentences

Estimate the probability of a sentence in terms of the number of times the sentence appears as it is in the corpora[1]. Such approach does NOT generalize at all - useless.

[1] A corpus is just a collection of text, i.e., Penn Treebank: 1M words of parsed WSJ

### 2.2 Slightly Better than Naive: Predict Probability of Words

Decompose the sentences into words gives flexibility of re-combining them in new ways and smoothing. Smoothing allows for possibility of unseen pieces.

- Joint Probability: $P(w) = P(w_1, w_2, \cdots, w_n)$
- Conditional Probability: $P(w) = P(w_n|w_1, w_2, \cdots, w_{n-1})$

### 2.3 N Gram Model

Assume each word depends only on a short linear history, a word can be considered dependent only on a few previous words.

$$P(w_1 w_2 w_3 ... w_n) = \prod_i P(w_i|w_1 w_2 ... w_{i-1})$$

Example:

$$P(\text{please close the door}) = P(\text{please}|\text{START})P(\text{close}|\text{please})...P(\text{STOP}|\text{door}) \quad (2)$$

**General Approach:**

$$\hat{P}(w|w_{-1}) = \frac{count(w_{-1}, w)}{\sum_{w'} count(w_{-1}, w')} \quad (3)$$

, where $w_{-1}$ means the one word before $w$. $\sum_{w'} count(w_{-1}, w')$ means sum over all the count which $w_{-1}$ appears before any word.

**Uni-gram**

$$P(w_1 w_2 w_3 \cdots w_n) \approx \prod_i P(w_i)$$

Problem: $P(the\ the\ the) >> P(I\ like\ ice\ cream)$

**Bi-gram**

Consider only previous word dependent: $P(w_1 w_2 w_3 \cdot w_n) \approx \prod_i P(w_i|w_{i-1})$, just like Example 2.

### 2.4 Problems with Unsmoothed N-Gram Language Models

- Long distance dependencies issue!
- The un-seen words will have probability of 0!

## 3 Measuring Model Quality

### 3.1 Perplexity

is the probability of the test set, normalized by the number of words.
The lower the perplexity, the better the model.

$$perp(W) = P(w_1 w_2 w_3 \cdots w_n)^{-1/N}$$

Or,

$$perp(X, \theta) = 2^{H(X|\theta)}$$

, where $H(X|\theta)$ is called **Entropy** (per-word test log likelihood) and is given by

$$H(X|\theta) = \frac{-1}{\underbrace{|X|}_{\sum_{x \in X} |x|}} \sum_{x \in X} \underbrace{\log_2 P(x|\theta)}_{\sum_i \log P(x_i|x_{i-1},\ \theta)}$$

### 3.2 Word Error Rate:

$$\text{Example in Figure 1, } WER = \frac{insertions + deletions + substitutions}{\text{True Sentence size}}$$

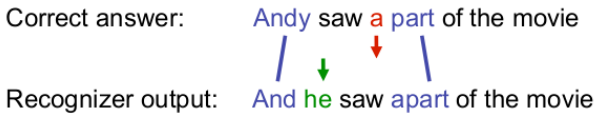


Figure 1: Word Error Rate

### 3.3 Zipf's Law

- It states that given the frequency $f$ of a word and it's rank $r$ in the list of words ordered $f \propto \frac{1}{r}$ or $f \times r = constant$.
- Not special to language: randomly generated character strings have this property too!

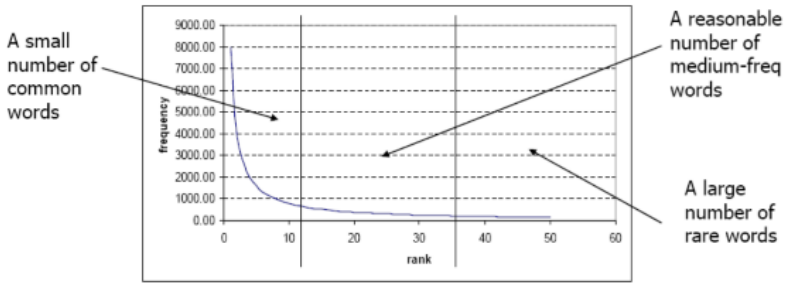


Figure 2: Zipf's Law Diagram

## 4 Smoothing

Maximum likelihood estimates (Equation 3) gives zero probabilities for un-seen instances from the training data. *Smoothing* can somewhat alleviate such problem.

**Discounting Methods:** Allocate probability mass for un-observed events by discounting counts from observed events. i.e., Add-One Smoothing, Good-Turing Smoothing, Katz Smoothing.
**Interpolation Methods:** Approximate counts of N-gram using combination of estimates from related denser histories. i.e., Linear Interpolation Smoothing (插值平滑)
**Back-off Methods:** Approximate counts of un-observed N-gram based on the proportion of back-off events. i.e., Kneser-Ney Smoothing

### 4.1 Add One smoothing (Laplace smoothing/Dirichlet Prior) Useless

We assume that we have seen all words once before actually seeing the data. For example, a uni-gram model with add-1 smoothing is:

$$P_{add_1}(x) = \frac{\text{count}(x) + 1}{\sum_{x'} \text{count}(x') + 1}$$

A bi-gram model:

$$P_{dir}(w|w_{-1}) = \frac{\text{count}(w|w_{-1}) + k\hat{P}(w)}{\sum_{w'} \text{count}(w'|w_{-1}) + k}$$

Problem: Except this does not perform well, because the number of un-seen bi-grams is usually greater than the bi-grams in the training text, which results in a lot of mass of the probability distribution shifted to the portion of un-seen words.

**Held out reweighting:** To-do

### 4.2 Linear Interpolation

Mixtures of related, denser histories; for example, Tri-gram model:

$$P(w|w_{-1}, w_{-2}) = \lambda \hat{P}(w|w_{-1}, w_{-2}) + \lambda' \hat{P}(w|w_{-1}) + \lambda'' \hat{P}(w) \tag{4}$$

* Not entirely clear why it works so much better. Details: Chen and Goodman, 98
* Good ways of learning the mixture weights with EM. Refer in the later chapters.

### 4.3 Absolute Discounting, a mixture of discounting and interpolation.

Instead of multiplying $\lambda$ like in the method of Linear Interpolation, we subtract a fixed constant $\delta \in [0, 1]$ from all the non-zero counts, and assign the left-over probability mass to the lower order model.

$$P_{ad}(w|w') = \frac{max(\text{count}(w', w) - \delta, 0)}{\sum_w \text{count}(w', w)} + \underbrace{\alpha(w')}_{\text{Interpolation weight}} \cdot P(w) \tag{5}$$

, where $w'$ refers to the previous word, and

$$\alpha(w') = \frac{\delta \times \{w : \text{count}(w', w) > 0\}}{\sum_w \text{count}(w', w)} \tag{6}$$

Problem: The property for Maximum likelihood conditional distribution: $\hat{P}(w) = \sum_{w'} \hat{P}(w|w')P(w')$ does not hold for absolute discounting. Better Explain

$$\hat{P}(w) \neq \sum_{w'} \hat{P_{ad}}(w|w')P(w')$$

This discounting basically falls back to the maximum likelihood unigram model if the count is low.

*Note: Do not confuse this inequality with the total law of probability. Absolute discounting does maintain the law of total probability in that if you marginalize over w', the resulting unigram distribution $P_{ad}(w)$ will sum to one: $\sum_w P_{ad}(w) = 1$. The problem is that it is not 'calibrated', i.e., it is not the same as the MLE estimate $\hat{P}(w)$.*

### 4.4 Kneser-Ney Smoothing:

**Intuition:** Lower order distribution needs to be 'altered' to preserve the marginal constraint.

**Clearer Explain:** *Suppose in a context 'San Francisco' is common, but 'Francisco' occurs only after 'San'. 'Francisco' will get a high P in unigram. Absolute Discounting will give a high probability to 'Francisco' appearing after novel bigram histories; however, it's better to give 'Francisco' a low P in unigram because the only time it occurs is after 'San', in which case the bigram model fits well.*

Introduce $P_{\text{continuation}}(w)$ to represent 'how likely is w to appear as a novel continuation'.

- For each word, count the number of bigram types it completes.
- Every bigram type = a novel continuation the first time it was seen.
- $P_{\text{continuation}}(w) \propto |\{w_{i-1} : \text{count}(w_{i-1}, w) > 0\}|$, where $w_{i-1}$ is the number of proceeding words that occurs at least once before $w$.

1. To calculate $P_{\text{continuation}}(w)$, normalized by the total number of word bi-gram types:

$$P_{\text{continuation}}(w) = \frac{|\{w_{i-1} : \text{count}(w_{i-1}, w) > 0\}|}{|\{w_{j-1}, w_j : \text{count}(w_{j-1}, w_j) > 0\}|} \tag{7}$$

2. To calculate $P_{\text{continuation}}(w)$, normalized by the number of words that can precede all words:

$$P_{\text{continuation}}(w) = \frac{|\{w_{i-1} : \text{count}(w_{i-1}, w) > 0\}|}{\sum_{w'} |\{w'_{i-1} : \text{count}(w'_{i-1}, w') > 0\}|} \tag{8}$$

* Equation 7 and 8 have the same denominators as the number of possible bi-gram types is the same as the number of word types that can precede all words.

Show example?

Put together Absolute Discounting with the Kneser-Ney probability for the lower-order n-gram, we have the **Kneser-Ney Smoothing Algorithm (bi-gram model):**

$$P_{KN}(w_i|w_{i-1}) = \frac{max(\text{count}(w_{i-1}, w_i) - d,\ 0)}{\text{count}(w_{i-1})} + \lambda(w_{i-1}) P_{\text{continuation}}(w_i)$$

, where $\lambda(w_{i-1})$ takes the discount $d$ as weight to assign to the probability of the unigram $P_{\text{continuation}}(w_i)$

$$\lambda(w_{i-1}) = \frac{d}{\text{count}(w_{i-1})} |\{w : \text{count}(w_{i-1}, w) > 0\}|$$

- $\frac{d}{\text{count}(w_{i-1})}$ is the normalized discount.
- $|\{w : \text{count}(w_{i-1}, w) > 0\}|$ is the number of word types that can follow $w_{i-1}$, which is also the number of word types we discounted, as well as the number of times we applied normalized discount.

The interpolation part of the algorithm; $\lambda(w_{i-1}) P_{\text{continuation}}(w_i)$, determines how many probability mass can we assign to the continuation of the word.

**Kneser-Ney Smoothing, Recursive Generalized Formulation for N-Gram:**

$$P_{KN}(w_i|w_{i-(n-1)}^{i-1}) = \frac{Max(\text{Count}_{KN}(w_{i-(n-1)}^{i}) - d,\ 0)}{\text{Count}_{KN}(w_{i-(n-1)}^{i})} + \lambda(w_{i-(n-1)}^{i-1}) P_{\text{KN}}(w_i|w_{i-(n-2)}^{i-1}) \tag{9}$$

, where

$$\text{Count}_{KN}(\bullet) = \begin{cases} \text{Count}(\bullet) & \text{for the highest order} \\ \text{ContinuationCount}(\bullet) & \text{for lower order} \end{cases}$$

ContinuationCount(•) is number of unique single word contexts for •.

Code?

### 4.5 Interpolation Smoothing vs. Backoff Smoothing

- Both interpolation (Jelinek-Mercer) and backoff (Katz) involve combining information from higher- and lower-order models.
- **Key Difference:** In determining the probability of N-grams with non-zero counts, interpolation models use information from lower-order models while Backoff models do not. Backoff models uses lower-order models to determine the probability of n-grams with zero counts, so does the Interpolation Models.

Later

## 5 Further Readings:

- Modified Kneser-Ney Model, Chen and Goodman.
- Neural Language Models: Bengio et al. 03 Mikolov et al. 11
- Caching Models
- Topic Models
- Syntactic models: Use tree models to capture long-distance syntactic effects [Chelba and Jelinek, 98]
- Discriminative Models: Set N-gram weights to improve final task accuracy rather than fit training set density [Roark 05, Liang et. al. 06]
- Compressed LMs [Pauls & Klein 11, Heafield 11]