Language model unit:

- What a language model is
  - Analogy with human language models
  - Extrinsic evaluation
  - Intrinsic evaluation
- Statistics about words
  - *n*-grams
  - Counts, hapaxes, ranks
  - Zipf's law
- Basic language models
  - Training and testing
  - Maximum likelihood
  - Unigram, bigram, trigram
- Smoothing
  - What's wrong with maximum likelihood
  - Laplace smoothing
  - What's wrong with Laplace smoothing
- Good-Turing smoothing
  - Principle
  - Practice
- Linear interpolation
- Alternate approaches

Basic definitions and axioms of probability

- *Event* is a primitive term. Informally, events are the things whose probability we want to compute, estimate, or predict.
- The set of all events that we are modeling is the *event space*.
- For *language models*, the most common event is a word (type) occurring in a given context (being a token). Language models can also be used for predicting other linguistic events, like characters, word sequences, parts of speech, sentences, ...
- For event w, $P(w)$ is the probability of w. Moreover, $0 \leq P(w) \leq 1$
- If V is the event space, then $\sum_{w \in V} P(w) = 1$
- The *conditional probability* of event A in light of event B is

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

- *Bayes's theorem* allows us to convert from one conditional probability to another

$$P(A|B) = \frac{P(B|A)P(A)}{P(B)}$$

Language technology	**Source**	**Channel**	**Observation**
Text decompression	Original text	Compressor	Compressed text
Sentiment analysis	Writer's sentiment	Writing process	Text whose sentiment is to be determined
Spelling correction	Correctly spelled word	Typing process	Possibly misspelled word
POS tagging	POS	Writing process	Word
Machine translation	Text in target language	Writing process	Text in original language

$$\arg\max_S P(S|O) = \arg\max_S \frac{P(O|S)P(S)}{P(O)} = \arg\max_S P(O|S)P(S)$$

It was breakfast time.
Father was eating his egg.
Mother was eating her egg.
Gloria was sitting in a high chair
and eating her egg too.
Frances was eating bread and jam.
"What a lovely egg!" said Father.
"It is just the thing to start the day
off right," said Mother.
Frances did not eat her egg.

Russell Hoban and Lillian Hoban, *Bread and Jam for Frances*. I Can Read edition 2008; originally published 1964

$P(\texttt{Frances did not eat her egg})$

$$
\begin{aligned}
&= P(\texttt{egg}|\texttt{Frances did not eat her}) \cdot P(\texttt{Frances did not eat her}) \\
&= P(\texttt{egg}|\ldots) \cdot P(\texttt{her}|\texttt{Frances did not eat}) \cdot P(\texttt{Frances ...eat})
\end{aligned}
$$

$$P(w_{1:n}) = P(w_1)P(w_2|w_1)P(w_3|w_{1:2}) \cdots P(w_n|w_{1:n-2})$$

$$P(w_n|w_{1:n-1}) \approx P(w_n|w_{n-1}) \text{ or } P(w_n|w_{n-2}w_{n-1})$$

Perplexity (and logs)

$$P(w_0 \cdots w_{K-1})^{\frac{-1}{K}} = \sqrt[K]{\frac{1}{P(w_0 \cdots w_{K-1})}}$$

$$= \left(\prod_{i=0}^{K-1} P(w_i \mid h)\right)^{\frac{-1}{K}} = \sqrt[K]{\frac{1}{\prod_{i=0}^{K-1} P(w_i \mid h)}}$$

$$= 2^{\frac{-1}{K} \sum_{i=0}^{K-1} \lg P(w_i|h)}$$

$$\lg xy = \lg x + \lg y$$

$$\lg x^y = y \lg x$$

$$2^{\lg x} = x$$

Interpretation of perplexity:

> *We suspect that speech recognition people prefer to report on the larger non-logarithmic numbers given by perplexity mainly because it is much easier to impress funding bodies by saying that "we've managed to reduce perplexity from 950 to only 540" than by saying that "we've reduced cross entropy from 9.9 to 9.1 bits." However, perplexity does also have an intuitive reading: a perplexity of k means that you are as surprised on average as you would have been if you had had to guess between k equiprobable choices at each step.*
>
> *Manning and Schütze, Foundations of Statistical Natural Language Processing, pg 78.*

Summary so far (i.e., summary from last time)

- A language model is a probability function for linguistic events.
  Words: $P(w)$ Word sequences: $P(w_0 w_1)$, $P(W_{0:N})$ Words in context: $P(w|h)$
- Humans have a natural language model.
- How good is a language model? We can evaluate a language model
  - Extrinsically: Measure the performance of a tool that uses the language model (example—text decompressor).
  - Intrinsically: Compute the *perplexity* of a language model on a test text.
    - Perplexity is a measure of how "surprised" the language model is by the text text.
    - Perplexity is the inverse probability of the test text normalized (geometric mean) by the size of the text.

$$\sqrt[K]{\frac{1}{P(w_0 \cdots w_{K-1})}} = \left(\prod_{i=0}^{K-1} P(w_i \mid h) \right)^{\frac{-1}{K}} = 2^{\frac{-1}{K} \sum_{i=0}^{K-1} \lg P(w_i|h)}$$

- Language models are trained on textual data. By exploring textual data, we can observe
  - The most common words are function words.
  - The most common non-stopwords reveal subject matter and genre.
  - A type's frequency is proportional to the inverse of its rank (Zipf's law)

$$f \cdot r = C \qquad f \propto \frac{1}{r}$$

  - The frequency of various *n*-grams changes over time.

A **stationary** *process is one that does not change over time. This is clearly wrong for language: new expressions regularly enter the language while others die out. . . . Nevertheless, for a snapshot of text from a certain period, we can assume that the language is near enough to unchanging, and so this is an acceptable approximation to truth.*

Manning and Schütze, Foundations of Statistical Natural Language Processing, pg 76.

The assumption that the probability of a word depends only on the previous word is called a **Markov assumption***. Markov models are the class of probabilistic models that assume we can predict the probability of some future unit without looking too far into the past.*

Jurafsky and Martin, Speech and Language Processing 3e, §3.1

Let V be the vocabulary; we also use V to mean $|V|$ to reduce clutter.

Let N be the size of the training text and K be the size of the test text.

Let $C(w)$ be the number of tokens of type w in the training text; similarly define $C(w_1 w_2)$ etc.

Note that $\sum_{w \in V} C(w) = N$ and $\sum_{winV} P(w) = 1$.

Maximum Likelihood Fallacy

"Which road leads to the Wicked Witch of the West?" asked Dorothy.

"There is no road," answered the Guardian of the Gates. "No one ever wishes to go that way."

"How, then, are we to find her?" inquired the girl.

"That will be easy," replied the man, "for when she knows you are in the country of the Winkies she will find you, and make you all her slaves."

"Perhaps not," said the Scarecrow, "for we mean to destroy her."

"Oh, that is different," said the Guardian of the Gates. "No one has ever destroyed her before, so I naturally thought she would make slaves of you, as she has of the rest. "

F Baum, The Wonderful Wizard of Oz

Perspectives on the maximum likelihood fallacy

- MLE is biased high for rare events and (infinitely) low for unseen events.
- Rare events are given too much probability mass, unseen events are given too little.
- Predicting the future is not the same thing as predicting a randomly chosen past event.
- If a word occurs once in training, what is most likely?
  - That word actually does occur about once every N tokens.
  - The word is actually more common than $\frac{1}{N}$, but got unlucky in training.
  - The word is actually less common than $\frac{1}{N}$, but was lucky to be in the training set at all.

VOLUME 40, PARTS 3 AND 4 DECEMBER 1953

THE POPULATION FREQUENCIES OF SPECIES AND THE ESTIMATION OF POPULATION PARAMETERS

BY I. J. GOOD

A random sample is drawn from a population of animals of various species. (The theory may also be applied to studies of literary vocabulary, for example.) If a particular species is represented r times in the sample of size N, then r/N is not a good estimate of the population frequency, p, when r is small. Methods are given for estimating p, assuming virtually nothing about the underlying population. The estimates are expressed in terms of smoothed values of the numbers n_r ($r = 1, 2, 3, \ldots$), where n_r is the number of distinct species that are each represented r times in the sample. (n_r may be described as 'the frequency of the frequency r'.) Turing is acknowledged for the most interesting formula in this part of the work. An estimate of the proportion of the population represented by the species occurring in the sample is an immediate corollary. Estimates are made of measures of heterogeneity of the population, including Yule's 'characteristic' and Shannon's 'entropy'. Methods are then discussed that do depend on assumptions about the underlying population. It is here that most work has been done by other writers. It is pointed out that a hypothesis can give a good fit to the numbers n_r but can give quite the wrong value for Yule's characteristic. An example of this is Fisher's fit to some data of Williams's on Macrolepidoptera.

1. *Introduction.* We imagine a random sample to be drawn from an infinite population of animals of various species. Let the sample size be N and let n_r distinct species be each represented exactly r times in the sample, so that

Good-Turing smoothing

Let r range over frequencies; $r = C(w)$ for some $w \in V$.

Let n_r be the number of types that occur exactly r times in the training text. n_r is the *frequency of frequency* r.

$$n_r = |\{w \in V \mid C(w) = r\}|$$

Thus n_1 is the number of hapaxes, and n_0 is the number of unseen words.

$$\sum_{r=0}^{\infty} n_r = V$$

$$\sum_{r=0}^{\infty} (r \cdot n_r) = N$$

Good-Turing smoothing

Change the question from "What is the probability of seeing type w?" to "What is the probability of seeing a type that I have seen r times before?"

$$\frac{r \cdot n_r}{N}$$

Change the question again to, "What is the probability of seeing a type for the $r+1$st time?"

$$\frac{(r+1) \cdot n_{r+1}}{N}$$

This leads to the language model

$$P_{GT}(w \mid C(w) = r) = \frac{(r+1)n_{r+1}}{N \cdot n_r} = \frac{\frac{(r+1)n_{r+1}}{n_r}}{N} = \frac{\frac{(r+1)n_{r+1}}{N}}{n_r}$$

In particular, for unseen words,

$$P_{GT}(w \mid C(w) = 0) = \frac{n_1}{N \cdot n_0}$$

What's Wrong with Adding One?

William A. Gale
Kenneth W. Church
AT&T Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ, 07974

1. Introduction

One could estimate the probability of a word (or n-gram) in English by collecting a large corpus of English text, counting the number of times the word appears in the corpus, and normalizing by the size of the corpus. Unfortunately this method (known as MLE) produces a poor estimate when the count is small and an unacceptable estimate when the count is zero. This is particularly problematic in many applications because counts are often small and zero is often the most frequent count.

A common engineering practice, first proposed by Jeffries (1948), adds one to the counts in order to "fix" the zeros. This method, though, has severe deficiencies as will be illustrated by example: the estimation of word bigram probabilities in a year of Associated Press newswire (N = 44 million words). The source of these deficiencies can be shown by comparing the Add-One method to the Good-Turing method. This comparison reveals that adding one is correct only if the ratio of unseen to observed types equals the ratio of all types to sample size. Since these ratios will be equal only by coincidence, adding one will obtain the correct answers only by happenstance, if at all.

Axiom 1. Good-Turing smoothing is good.

Axiom 2. Frequency (as frequency rank) vs frequency of frequency follows Zipf's law.

Theorem. Laplace smoothing is bad. (Gale and Church, 1994)

Proof. *Suppose Laplace smoothing is good. Then, by Axiom 1, its formula would reduce to the formula for Good-Turing, that is,*

$$
\begin{aligned}
\frac{(r+1)n_{r+1}}{Nn_r} &= \frac{r+1}{N+V} && \text{by equating Lapace and GT probabilities for some type with count } r \\
\frac{n_{r+1}}{n_r} &= \frac{N}{N+V} \\
n_{r+1} &= \frac{N}{N+V} n_r \\
n_r &= \frac{N}{N+V} n_{r-1} && \text{by change of variable} \\
n_r &= \left(\frac{N}{N+V}\right)^r n_0
\end{aligned}
$$

By Axiom 2, Zipf's law predicts $n_r = \frac{c}{r}$ *for some* c*. Contradiction.* $\square$

Good-Turing Frequency Estimation Without Tears*

William A. Gale and Geoffrey Sampson
AT&T Bell Laboratories, USA and University of Sussex, U.K.

ABSTRACT

Linguists and speech researchers who use statistical methods often need to estimate the frequency of some type of item in a population containing items of various types. A common approach is to divide the number of cases observed in a sample by the size of the sample; sometimes small positive quantities are added to divisor and dividend in order to avoid zero estimates for types missing from the sample. These approaches are obvious and simple, but they lack principled justification, and yield estimates that can be wildly inaccurate. I.J. Good and Alan Turing developed a family of theoretically well-founded techniques appropriate to this domain. Some versions of the Good-Turing approach are very demanding computationally, but we define a version, the Simple Good-Turing estimator, which is straightforward to use. Tested on a variety of natural-language-related data sets, the Simple Good-Turing estimator performs well, absolutely and relative both to the approaches just discussed and to other, more sophisticated techniques.

1. THE USE OF GOOD-TURING TECHNIQUES

Consider a population made up of individuals drawn from a large number of distinct species having diverse frequencies, including a few very common species, many rare species, and intermediate numbers of species of intermediate frequencies. We want to estimate the frequencies

Say that the sample contains N individuals, and that for each species i it includes r_i examples of that species. (The number s of distinct species in the population may be finite or infinite, though N – and consequently the number of distinct species represented in the sample – must be finite.) We call r_i the *sample frequency* of i, and we want to use it in order to estimate the *population frequency* p_i of i, that is the probability that an individual drawn at random from

Katz's K cut-off

Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer

SLAVA M. KATZ

Abstract—**The description of a novel type of *m*-gram language model is given. The model offers, via a nonlinear recursive procedure, a computation and space efficient solution to the problem of estimating probabilities from sparse data. This solution compares favorably to other proposed methods. While the method has been developed for and successfully implemented in the IBM Real Time Speech Recognizers, its generality makes it applicable in other areas where the problem of estimating probabilities from sparse data arises.**

Sparseness of data is an inherent property of any real text, and it is a problem that one always encounters while collecting frequency statistics on words and word sequences (*m*-grams) from a text of finite size. This means that even for a very large data collection, the maximum likelihood estimation method does not allow us to adequately estimate probabilities of rare but nevertheless possible word sequences—many sequences occur only once (''singletons''); many more do not occur at all. Inadequacy of the maximum likelihood estimator and the necessity to estimate the probabilities

and the Turing's estim

where

$c^*($

It follows from (1)
(5), for the set of w
sample is

w_1^m

This, in turn, leads
some previously un
tons'' in the text:

On the other hand,

$\sum_{w_1^m : c(w_1^m) > 0}$

Good-Turing with Katz's k cut off:

$$P_{GT-Katz}(w) = \begin{cases} \frac{n_1}{Nn_0} & \text{if } C(w) = 0 \quad \text{(unseen words)} \\ \frac{(r+1)\frac{n_{r+1}}{n_r} - r\frac{(k+1)\cdot n_{k+1}}{n_1}}{N\left(1 - \frac{(k+1)\cdot n_{k+1}}{n_1}\right)} & \text{if } 1 \leq r = C(w) \leq k \quad \text{(rare words)} \\ \frac{C(w)}{N} & \text{otherwise} \quad \text{(common words)} \end{cases}$$

Linear interpolation

Given k language models P_0, P_1, $\ldots P_{k-1}$ and weights $\boldsymbol{\lambda} = [\lambda_0, \lambda_1, \ldots \lambda_{k-1}]$, the interpolated model is

$$P_{LI}(w) = \sum_{j=0}^{k-1} \lambda_j P_j(w) = \lambda_0 P_0(w) + \lambda_1 P_1(w) + \cdots + \lambda_{k-1} P_{k-1}(w)$$

Symbol and variable summary:

- Let M be the size of the held-out set (number of tokens).
- Let k be the number of constituent models
- Let j and jj range over constituent models let λ_j be the weight given to constituent model j
- Let z_{ij} be the expected value of model j's contribution to the probability of token w_i
- Let $\ell(\boldsymbol{\lambda})$ is the average log likelihood

Theorem. If each constituent language model P_j is a proper language model and $\sum_{j=0}^{k-1} \lambda_j = 1$, then P_{LI} is a proper language model.

Proof. *Suppose all that. Then*

$$\begin{aligned}
\sum_{w \in V} P_{LI}(w) &= \sum_{w \in V} \sum_{j=0}^{k-1} \lambda_j P_j(w) \\
&= \sum_{j=0}^{k-1} \sum_{w \in V} \lambda_j P_j(w) \\
&= \sum_{j=0}^{k-1} \lambda_j \sum_{w \in V} P_j(w) \\
&= \sum_{j=0}^{k-1} \lambda_j = 1 \qquad \square
\end{aligned}$$

The "degenerate EM" algorithm for finding optimal linear interpolation coefficients λ_i

Language and Statistics, Spring 2007

February 5, 2007

Input: A set of probability streams $P_j(w_i|h_i)$ for $1 \leq j \leq K, 1 \leq i \leq N$ where j is an index into the set of K (possibly unknown) language models, and i is an index into the N words of text over which the weights are to be optimized.

Output: $\lambda opt = \{\lambda_1^{opt}, ..., \lambda_K^{opt}\}$ such that $\forall j, \lambda_j^{opt} \geq 0$ and $\sum_{j=1}^{K} \lambda_j^{opt} = 1$. These are the weights which optimize the average log-likelihood of the data.

The algorithm

0. *Initialize:*
   Set $\lambda_j^{(0)}$ such that $\forall j, 0 < \lambda_j^{(0)} < 1$ and $\sum_{j=1}^{K} \lambda_j^{(0)} = 1$.
   Set the iteration counter $t = 0$.

1. *Update:*

$$\forall j, \lambda_j^{(t+1)} = \frac{1}{N} \sum_{i=1}^{N} \frac{\lambda_j^{(t)} P_j(w_i|h_i)}{\sum_{k=1}^{K} \lambda_k^{(t)} P_k(w_i|h_i)}$$

Coming up:

- Reading from J&M, Sections 3.(0-8) (Mon, Sept 15) At least 3.(0& 1) by duedate; you may spread out the rest
- Language model quiz (Tues, Sept 23)
- Language model programming assignment (Fri, Sept 26)

- Reading from J&M, Sections 8.(0–4) (Wed, Sept 24)