Lecture 9 — 04/01/24
Language Modeling (Part II)[1]

1 Introduction

In the strictest sense, ***language models*** (or LMs) are technologies which assign probabilities to sentences or utterances (or occasionally, words) by defining a discrete probability distribution over a finite set of words (or in the case of word models, characters).[2]

Two features of language make this task challenging. First, any corpus of sentences is constrained by the grammatical constraints imposed by serialization, a richly-specified language-specific procedure, but also by many other factors, including semantico-pragmatic plausibility: `I went for a walk but forgot my {phone, torso}...` Second, linguistic wellformedness cannot be reduced to empirical probability of utterance.[3]

While both points are often well-taken, it affected a schism between cognitive-science and engineering approaches to "modeling language", one which persists to this day. Modern engineering solutions: are heuristic in nature, make few affordances for cognitive plausibility, and conflate ill-formed utterances with improbable utterances.

Furthermore, the productive nature of language and the sparse (i.e., highly-skewed) frequencies of words ensure that few sentences in any given corpus will have ever occurred before. If you've taken class with me, you'll no doubt have seen several sparse-data-problem / Zipfian distirbutions plots so I won't repeat another here.

1.1 Applications

Language modeling is essential for automatic speech recognition (ASR). In ASR, the language model is used to select the most plausible of a set of acoustically confusable transcriptions. More formally, here is the somewhat grandiosely named *fundamental theorem of speech recognition* (Jelinek 1998:4f.). Given the observed acoustic sequence $\mathbf{A}$, the recognizer selects a transcript $\hat{\mathbf{W}}$ according to

$$\hat{\mathbf{W}} = \arg\max_{\mathbf{W}} P(\mathbf{A} \mid \mathbf{W}) \cdot P(\mathbf{W}) \tag{1}$$

where the $P(\mathbf{W})$ term is referred to as the language model. Language modeling is applicable to many other tasks, including:

- (mobile) text entry engines (IMEs),
- assistive technologies such as T9 (Grover, King, & Kushler 1998)
- spelling correction (Mays, Damerau, & Mercer 1991)
- optical character recognition (OCR),
- machine translation (MT), and
- natural language generation.

1.2 A note on Software

There are an enormous number of software libraries for creating language models. OpenGrm-NGram[4] (Roark et al. 2012) is a general-purpose command-line library for creating language models and compiling them into OpenFst-compatible weighted finite-state transducers. One popular alternative is KenLM[5] (Heafield 2011). In contrast, discriminative and neural language models are generally a "roll-your-own" technology.

[1] This handout is adapted from Kyle Gorman. In turn, loosely based on Chen and Goodman 1998 and slides by Brian Roark.
[2] Though you will find researchers using technologies which they call "language models" if they rank strings in some way even when they don't **actually** assign probabilities to those strings. IMO this is just not accurate, but you'll find an even looser usage of the term by which any next- or nearby-word prediction model may sometimes be called a language models (e.g., Bender, Gebru, McMillan-Major, & Shmitchell 2021)
[3] Make sure you understand why not, or ask me!
[4] `https://ngram.opengrm.org`
[5] `https://github.com/kpu/kenlm`

2 Formal preliminaries

2.1 Definitions

Let Ω be a set of symbols[6]. Many applications use a closed vocabulary of word-like tokens, whereas others use a character vocabulary (e.g., the 95 printable ASCII characters). A language model P is a discrete probability distribution over strings in Ω^*.

2.2 Sentence probability

Recall that the joint probability of some sequence $\mathbf{W} = w_1 \ldots w_n$ is given by the product of the conditional probabilities:

$$P(w_1) \cdot P(w_2 \mid w_1) \cdot P(w_3 \mid w_1 w_2) \ldots P(w_n \mid w_1 \ldots w_{n-1}). \tag{2}$$

For example:

$$\begin{aligned} P(\texttt{du hast mich gefragt}) = &\ P(\texttt{du}) \cdot \\ &\ P(\texttt{hast} \mid \texttt{du}) \cdot \\ &\ P(\texttt{mich} \mid \texttt{du hast}) \cdot \\ &\ P(\texttt{gefragt} \mid \texttt{du hast mich}). \end{aligned}$$

Given a token w_i, we refer to the symbols preceding it, $h_i = w_1 \ldots w_{i-1}$, as that token's ***history***. In practice, the inherent sparsity of the data forces us to adopt a simplifying assumption that each symbol is conditioned on only the n preceding words, and is conditionally independent of earlier words. For instance, the 1st-order Markov model—a ***bigram*** model—estimate is given by:

$$\begin{aligned} \hat{P}(\texttt{du hast mich gefragt}) = &\ P(\texttt{du}) \cdot \\ &\ P(\texttt{hast} \mid \texttt{du}) \cdot \\ &\ P(\texttt{mich} \mid \texttt{hast}) \cdot \\ &\ P(\texttt{gefragt} \mid \texttt{mich}). \end{aligned}$$

We refer to hw sequences as ***n-grams***, and language models which adopt a Markov assumption as ***n-gram models***:

- a unigram (zeroth order) model assumes symbols are independent of each other,
- a bigram (first order) model conditions each symbol on the preceding symbol,
- a trigram (second order) model conditions each symbol on the preceding two symbols,
- a 4-gram (third order) model conditions each symbol on the preceding three symbols,

and so on. Compared to the alternatives, n-gram models are:

- **compact**: only observed n-grams and their probabilities must be stored;
- **scalable**: their performance increases as the amount of available data increases;[7]
- **efficient**: they can be efficiently scored and searched using finite-state algorithms; and
- **unreasonably effective**: their performance is still hard to beat for many tasks.

3 N-gram language models

3.1 Maximum likelihood estimation

Given a corpus of strings $w_1 \ldots w_n \subseteq \Omega^*$, let $C \subseteq \Omega^* \times \mathbb{N}$ be a function which gives the frequency of specific n-grams. According to the method of maximum likelihood, the conditional probability of seeing a word w with history h is

[6] other sources may just call this V, presumably for *"Vocabulary"*. In either case, we prefer either Ω or V here rather than Σ to avoid confusion with the summation symbol.

[7] And, most types of language models can be estimated in a data-parallel fashion (Allauzen, Riley, & Roark 2016).

$$\hat{P}(w \mid h) = \frac{C(hw)}{C(h)}. \tag{3}$$

3.1.1 Bigram example

Given a corpus consisting of the following three sentences (shown with padding tokens written explicitly):

```
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>
```

Then we could use eq. 3 to derive the following counts as follows:

$$\hat{P}(\text{I} \mid \text{<s>}) = \frac{2}{3} = 0.67$$
$$\hat{P}(\text{Sam} \mid \text{<s>}) = \frac{1}{3} = 0.33$$
$$\hat{P}(\text{am} \mid \text{I}) = \frac{2}{3} = 0.67$$
$$\hat{P}(\text{</s>} \mid \text{Sam}) = \frac{1}{2} = 0.5$$
$$\hat{P}(\text{Sam} \mid \text{am}) = \frac{1}{2} = 0.5$$
$$\hat{P}(\text{do} \mid \text{I}) = \frac{1}{3} = 0.33$$
$$\ldots$$

However, due to the inherent sparsity of the data, we expect to encounter unattested n-grams, i.e., n-grams hw such that $C(hw) = 0$, even when adopting a Markov assumption. Thus the maximum likelihood estimate will assign zero probability to sentences which contain unattested n-grams!

3.2 Smoothing

To avoid this problem, we attempt to reserve some probability mass for unseen n-grams, a technique known as ***smoothing***. All smoothing techniques have a "Robin Hood" or "redistributive" characteristic: some probability mass is taken from attested n-grams ("the rich") and given to less frequent n-grams ("the poor").[8]

3.2.1 Laplace smoothing

One of the simplest forms of smoothing is ***Laplace*** (or ***add***-α) smoothing. We simply add some small positive "pseudocount" $\alpha \in \mathbb{R}_+$ to each n-gram count, including unattested but logically possible n-grams, then modify the denominator to reflect the pseudocounts so as to ensure proper normalization:

$$\hat{P}(w \mid h) = \frac{C(hw) + \alpha}{C(h) + \alpha|V|}. \tag{4}$$

In practice, Laplace smoothing is not a very effective smoothing technique. Modern smoothing methods can be conceptualized in terms of either ***backoff*** or ***interpolation*** strategies.

[8] In the terminology of machine learning, zero probabilities assigned to sequences with unattested n-grams can be thought of as a type of ***overfitting***, and smoothing can be thought of as a special case of ***regularization***.

3.2.2 Backoff

In ***backoff*** smoothing strategies, we derive the estimate for unattested n-grams using the next lowest-order model. Given an n-gram hw, let us define the ***backoff history***, h', such that $h = w'h'$. For instance, for the trigram `du hast mich`, $h =$ `du hast` and $h' =$ `hast`. Thus the history for a trigram is itself a bigram, and its backoff history is a unigram. Thus:

$$\hat{P}(w \mid h) = \begin{cases} \tilde{P}(w \mid h) \text{ if } C(hw) > 0 \\ \alpha_h \hat{P}(w \mid h') \text{ otherwise} \end{cases} . \tag{5}$$

The term α_h, the ***discount***, must be defined so as to ensure proper normalization.

3.2.3 Interpolation

Interpolation conceives of smoothing not as a disjunction between attested and unattested n-gram probabilities, but rather as a mixture—a weighted sum—of lower- and higher-order estimates:

$$\hat{P}(w \mid h) = \lambda_h \tilde{P}(w \mid h) + (1 - \lambda_h)\hat{P}(w \mid h'). \tag{6}$$

Note that $1 - \lambda_h$ corresponds loosely to the discount α_h in (5).

Most smoothing techniques can be interpreted either as some kind of backoff or interpolation, though some people find one more intuitive than the other. I was originally taught that interpolation performs better, though I never saw this claim rigorously tested and backoff may be more commonly used when language models are implemented as weighted finite-state transducers.

4 Evaluation

Since we're building engineering tools and there is often not a *"ground truth"* of the correct language model we should generally perform **extrinsic evaluation** when possible. i.e. comparing different language model implementations using some model-external measure in a downstream task. For instance, if we are developing language models for automatic speech recognition, we can measure ***word error rate*** (i.e., roughly the number of errors in the best transcription).

Frequently though, this is not possible and language models are often evaluated using an **intrinsic** metric called ***perplexity*** (Jelinek, Mercer, Bahl, & Baker 1977), closely related to Shannon (1948) entropy. Given a language model P and a held-out corpus $w_1 \ldots w_n \subseteq V^*$, the probability of the corpus is given by:

$$P(w_1 \ldots w_n) = \prod_{i=1}^{N} P(w_i \mid h_i) \tag{7}$$

Perplexity is simply this quantity exponentiated to $-1/n$:

$$\text{PPX}_P(w_1 \ldots w_n) = P(w_1 \ldots w_n)^{-\frac{1}{n}} \tag{8}$$

Perplexity is the inverse probability of the test set, normalized by the number of words. You may see this "simplified" by expanding the formula, first by flipping the negative exponent:

$$\text{PPX}_P(w_1 \ldots w_n) = \sqrt[N]{\frac{1}{P(w_1 \ldots w_n)}} \tag{9}$$

If we apply the chain rule for a bigram model then we get:

$$\text{PPX}_P(w_1 \ldots w_n) = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{P(w_1 \mid w_{i-1})}} \tag{10}$$

Intuitively, a lower perplexity indicates that the language model P assigns more probability mass to the corpus $w_1 \ldots w_n$. Whenever possible, however, we prefer to compare language models using model-extrinsic metrics derived from downstream tasks. For instance, if we are developing language models for automatic speech recognition, we can measure ***word error rate*** (i.e., roughly the number of errors in the best transcription).

5 Finite-state encoding

N-gram language models can be encoded as ***weighted finite-state acceptors*** (WFSAs), in which each transition and final state is associated with a weight. While a finite-state representation of a language model is useful for many applications, it is essential in building finite-state speech recognition systems, as discussed below.

5.1 Weighted FSTs

Finite-state acceptors and transducers can both be extended with numerical weights, including probabilities, so long as the algebraic properties of those weights form a ***semiring***.

5.1.1 Semirings

Before defining semirings, it is necessary to first introduce a related notion. A ***monoid***, is an ordered pair $(\mathbb{K}, \bullet)$ where $\mathbb{K}$ is a set and $\bullet$ is a binary operator over $\mathbb{K}$ with the properties of

1. ***closure***: $\forall a, b \in \mathbb{K} : a \bullet b \in \mathbb{K}$,
2. ***associativity***: $\forall a, b, c \in \mathbb{K} : (a \bullet b) \bullet c = a \bullet (b \bullet c)$, and
3. ***identity***: $\exists e \in \mathbb{K} : e \bullet a = a \bullet e = a$.

A monoid is said to be ***commutative*** if, in addition, $\forall a, b \in \mathbb{K} : a \bullet b = b \bullet a$. Then, a ***semiring*** is a five-tuple $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$ where

1. $(\mathbb{K}, \oplus)$ form a commutative monoid with identity element $\bar{0}$,
2. $(\mathbb{K}, \otimes)$ form a monoid with identity element $\bar{1}$,
3. $\forall a, b, c \in \mathbb{K} : a \otimes (b \oplus c) = (a \otimes b) \oplus (a \otimes c)$, and
4. $\forall a \in \mathbb{K} : a \otimes \bar{0} = \bar{0} \otimes a = \bar{0}$.

These constraints require that $\oplus$ is commutative, that $\bar{0}$ is the additive identity, that $\bar{1}$ is the multiplicative identity, that $\otimes$ distributes over $\oplus$, and that $\bar{0}$ is the multiplicative annihilator (i.e., that any weight multiplied with $\bar{0}$ is $\bar{0}$).

Some common semirings are shown in Table 1. Note that the tropical and log semirings take advantage of underflow-avoiding properties of logarithms discussed last week.

	$\mathbb{K}$	$\oplus$	$\otimes$	$\bar{0}$	$\bar{1}$
Boolean	$\{0, 1\}$	$\vee$	$\wedge$	0	1
Plus-times (“probability”)	$\mathbb{R}_+$	$+$	$\times$	0	1
Max-times (“real”)	$\mathbb{R}_+$	max	$\times$	0	1
Log	$\mathbb{R} \cup \{\pm\infty\}$	$\oplus_{\log}$	$+$	$+\infty$	0
Tropical	$\mathbb{R} \cup \{\pm\infty\}$	min	$+$	$+\infty$	0

Table 1: Some common semirings for finite-state applications; $a \oplus_{\log} b = -\ln(e^{-a} + e^{-b})$.

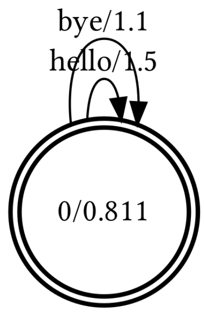

Figure 1: Unigram language model topology

5.2 Weighted finite-state acceptors

Weighted finite-state acceptors are finite acceptors in which transitions—and final states—are associated with weights drawn from a semiring $\mathbb{K}$. A ***weighted finite-state acceptor*** (WFSA) is defined by a six-tuple consisting of

- a finite set of states Q,
- a start or initial state $s \in Q$,
- a **semiring** $(\mathbb{K}, \oplus, \otimes, \bar{0}, \bar{1})$,
- a **final weight function** $\omega \subseteq Q \times \mathbb{K}$,
- an alphabet Σ, and
- a **transition relation** $\delta \subseteq Q \times (\Sigma \cup \{\varepsilon\}) \times \mathbb{K} \times Q$.

Each path is then associated with, in addition to a state sequence $q_1, q_2, \ldots, q_n \in Q^n$ and a string $z = z_1, z_2, \ldots, z_n \in \Sigma^n$, a weight sequence $k_1, k_2, \times, k_n \in \mathbb{K}^n$, and a complete path must end on a final state whose final weight (given by ω) is non-$\bar{O}$. A WFSA is said to accept a string $z \in \Sigma^n$ with weight

$$\left(\bigotimes_{i=1}^{n} k_i\right) \otimes \omega[q_n] = k_1 \otimes k_2 \otimes \ldots \otimes k_n \otimes \omega[q_n].$$

if there exists a complete path with string z and weight sequence $k_1, k_2, \ldots, k_n$.

Gorman and Sproat (2021:§1.6) provide a similar definition for finite-state transducers (WFSTs), finite transducers augmented with transition and final weights.

5.3 Topology

WFSAs are commonly used to encode language models. Let us assume that $\Omega = \{\texttt{hello}, \texttt{bye}\}$ and that we are deriving counts from the following corpus:

```
hello
bye
hello
bye bye
```

We also use Kneser-Ney smoothing with backoff, encoding weights using the tropical semiring. Figure 1, Figure 2, and Figure 3 illustrate the associated topologies. Note that in the bigram and trigram models, the state labeled 1 is the start state, and the state 0 represents a backoff to the unigram model.

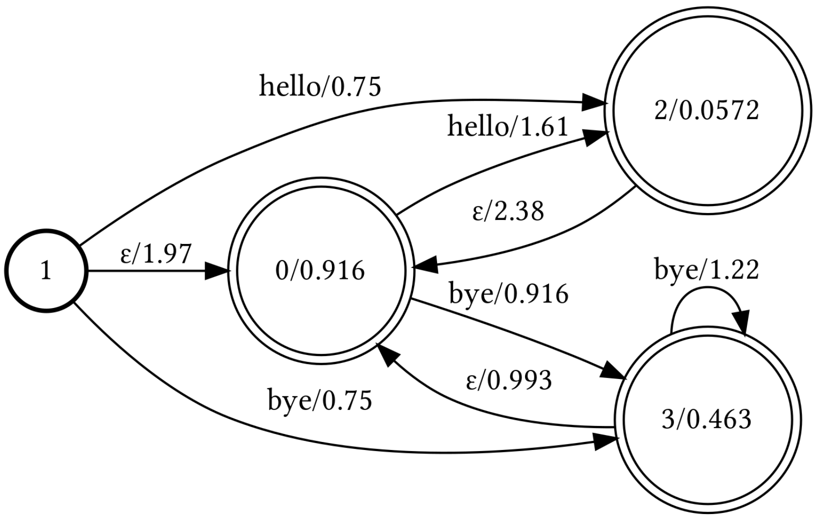

Figure 2: Bigram language model topology

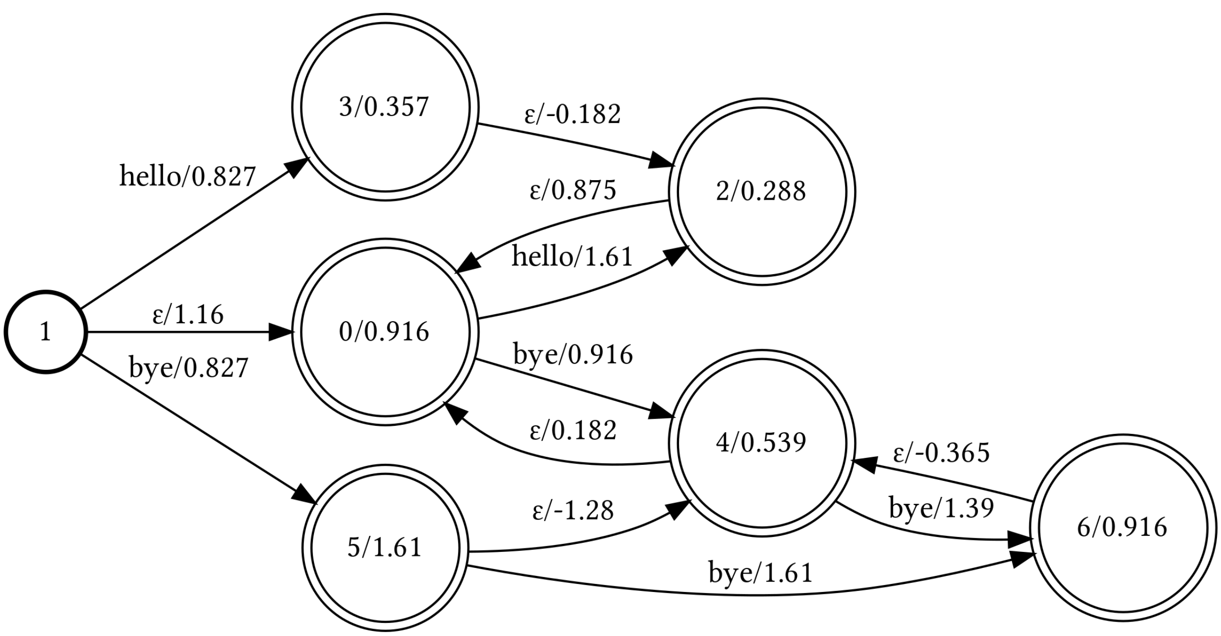

Figure 3: Trigram language model topology

References

Allauzen, C., Riley, M., & Roark, B. (2016). Distributed representation and estimation of WFST-based n-gram models. In ***Acl workshop on statistical nlp and weighted automata*** (p. 32-41).

Bell, T. C., Cleary, J. G., & Witten, I. H. (1990). ***Text compression***. Prentice Hall.

Bender, E. M., Gebru, T., McMillan-Major, A., & Shmitchell, S. (2021). On the dangers of stochastic parrots: can language models be too big? In ***Proceedings of the 2021 acm conference on fairness, accountability, and transparency*** (p. 610-623).

Brants, T. (2000). TnT – a statistical part-of-speech tagger. In ***Sixth applied natural language processing conference*** (p. 224-231).

Carpenter, B. (2005). Scaling high-order character language models to gigabytes. In ***ACL workshop on software*** (p. 86-99).

Chen, S. F., & Goodman, J. (1998). An empirical study of smoothing techniques for language modeling. ***Computer Speech & Language***, ***13***(4), 359-394.

Gorman, K., & Sproat, R. (2021). ***Finite-state text processing***. Morgan & Claypool.

Grover, D. L., King, M. T., & Kushler, C. A. (1998). ***Reduced keyboard disambiguating computer.*** US Patent 5,818,437.

Heafield, K. (2011). KenLM: faster and smaller language model queries. In ***Proceedings of the sixth workshop on statistical machine translation*** (p. 187-197).

Jelinek, F. (1998). ***Statistical methods for speech recognition***. MIT Press.

Jelinek, F., Bahl, L. R., & Mercer, R. L. (1983). A maximum likelihood approach to continuous speech recognition. ***IEEE Transactions on Pattern Analysis & Machine Intelligence***, ***5***, 179-190.

Jelinek, F., Mercer, R. L., Bahl, L. R., & Baker, J. K. (1977). Perplexity: a measure of the difficulty of speech recognition tasks. ***Journal of the Acoustical Society of America***, ***62***(S1), S63.

Katz, S. M. (1975). Estimation of probabilities from sparse data for the language model component of a speech recognizer. ***IEEE Transactions on Acoustics, Speech and Signal Processing***, ***35***(3), 400-401.

Mays, E., Damerau, F. J., & Mercer, R. L. (1991). Context based spelling correction. ***Information Processing & Management***, ***27***(5), 517–522.

Ney, H., Essen, U., & Kneser, R. (1994). On structuring probabilistic dependences in stochastic language modelling. ***Computer Speech & Language***, ***8***(1), 1-38.

Roark, B., Sproat, R., Allauzen, C., Riley, M., Sorensen, J., & Tai, T. (2012). The OpenGrm open-source finite-state grammar software library. In ***Proceedings of the acl 2012 system demonstrations*** (p. 61-66).

Shannon, C. (1948). A mathematical theory of communication. ***Bell System Technical Journal***, ***27***, 379-423.

A Additional smoothing techniques

A.1 Katz smoothing

Katz (1975) smoothing is based on the Good-Turing probability estimation technique, which is inspired by the following two intuitions:

- **Uniformitarianism**: All n-grams that have the same frequency should have the same probability.
- **Leave-one-out estimation**: The expected frequency of an n-gram with frequency n_r is roughly that of the observed frequency of n-grams with frequency n_{r+1}.

Definition Let n_r be the number of types that occur r times in the sample. Thus, $(r+1)n_{r+1}$ is the total count of all words which occur $r+1$ times. To obtain the probability estimate, we simply divide by the number of words N and by n_r, the number of words of frequency r:

$$\tilde{P}(w) = \frac{r+1}{N}\frac{n_{r+1}}{n_r} \text{ where } r = C(w). \tag{11}$$

Usually only infrequent n-grams are so smoothed (and the normalization corrected to reflect).

Usage Katz smoothing is still widely used used for high-order large-vocabulary models in speech recognition.

A.2 Jelinek-Mercer smoothing

Definition In Jelinek-Mercer smoothing (Jelinek, Bahl, & Mercer 1983), the interpolation coefficient λ_h is determined empirically—using maximum likelihood estimation or expectation maximization—on a held-out development set.

Usage This form of smoothing is not widely used for language modeling anymore, but it is sometimes employed to estimate the $P(\mathbf{T})$ term in part of speech taggers (e.g., Brants 2000).

A.3 Witten-Bell smoothing

In Witten-Bell smoothing (Bell, Cleary, & Witten 1990), λ_h is computed using statistical properties of h. The intuition here is that the MLE estimate is of higher quality—i.e., closer to the true probability—and requires less smoothing when its history:

- is more frequent, or
- has few unique continuations.

For instance, if `Rolls` is a common history, or if it is followed by few words other than `Royce`, the estimate $\hat{P}(\texttt{Royce} \mid \texttt{Rolls})$ is given more weight than the backoff probability $\tilde{P}(\texttt{Royce})$.

Definition Let $|hw_*|$ be the number of unique words $w \in V$ for which $C(hw) > 0$. Then

$$\lambda_h = \frac{C(h)}{C(h) + \kappa |hw_*|} \tag{12}$$

where κ is a hyperparameter (though usually $\kappa = 1$).

Usage Witten-Bell smoothing is generally thought to be the best choice for high-order character language models (e.g., Carpenter 2005), but is rarely used for large-vocabulary models.

A.4 Kneser-Ney smoothing

In Witten-Bell smoothing, the degree of smoothing is conditioned in part by the "promiscuity" of the history h. In Kneser-Ney smoothing (Ney, Essen, & Kneser 1994), however, the smoothed estimate is instead conditioned by the "promiscuity" of the continuation w. For instance, if `Royce` is preceded by few words other than `Rolls`, the MLE estimate is given more weight.

Definition Given w and a backoff history h', let $|w_* h' w_i|$ be the number of unique histories of the form $h = w' h'$ for which $C(h) > 0$. Then:

$$\tilde{P}(w \mid h) = \lambda_h \hat{P}(w \mid h) + (1 - \lambda_h)\tilde{P}(w \mid h') \text{ where } \tilde{P}(w \mid h') \propto |w_* h' w|. \tag{13}$$

Usually, only lower-order distributions are so smoothed.

Usage Variants of Kneser-Ney smoothing are generally thought to be one of the best methods for medium- and large-vocabulary models (e.g., Chen & Goodman 1998).