1 Recap

- the simplest way to estimate LMs, MLE, is problematic on sparse data
  - problem with zero counts ⇒ zero probabilities for unseen n-grams
  - high perplexity on test, low performance of the model as a whole
- smoothing and back-off to address to problems
  - add-α (Laplace estimation)

$$p(w_i|w_{i-n+1}^{i-1}) = \frac{\alpha + c(w_{i-n+1}^{i})}{\alpha|V| + \sum_{w_i} c(w_{i-n+1}^{i})}$$

  - deleted estimation
  - Good-Turing estimation

$$r^* = (r+1)\frac{N_{r+1}}{N_r}$$
$$p(w_{i-n+1}^{i}) = r^* / \sum_{r=0}^{\infty} N_r r^*$$

  - interpolation/back-off with lower-order models (Jelinek-Mercer smoothing and Katz estimation)

2 "Witten-Bell"-smoothing

Previously, we computed n-gram probabilities based on relative frequency:

$$p(w_2|w_1) = \frac{\text{count}(w_1, w_2)}{\text{count}(w_1)}$$

"Good Turing"-smoothing adjusts counts c to expected counts c^*:

$$\text{count}^*(w_1, w_2) \le \text{count}(w_1, w_2)$$

We use these expected counts for the prediction model (but 0^* remains 0):

$$\alpha_2(w_2|w_1) = \frac{\text{count}^*(w_1, w_2)}{\text{count}(w_1)}$$

This leaves probability mass for the discounting function:

$$d_2(w_1) = 1 - \sum_{w_2} \alpha_2(w_2|w_1) \qquad (1)$$

Example: Diversity of Predicted Words Consider the bigram histories "spite" and "constant". They both occur 993 times in the Europarl corpus, but:

- only 9 different words follow "spite", which is almost always followed by "of" (979 times), due to the expression "in spite of"
- a total of 415 different words follow "constant", where the most frequent word is "and" (42 times), followed by "concern" (27 times) and "pressure" (26 times). Also, there's a huge tail of singletons (268 different words).

This means it is much more likely to see a new bigram that starts with "constant" than with "spite". "Witten-Bell"-smoothing considers the diversity of predicted words.

2.1 Idea:

"Witten-Bell"-smoothing is an instance of the recursive interpolation method (Jelinek-Mercer smoothing)

$$p_{WB}(w_i|w_{i-n+1}^{i-1}) = \lambda_{w_{i-n+1}^{i-1}} p_{ML}(w^i|w_{i-n+1}^{i-1}) + (1-\lambda_{w_{i-n+1}^{i-1}}) p_{WB}(w_i|w_{i-n+2}^{i-1})$$

- we should use higher-order model if n-gram w_{i-n+1}^{i} was seen in the training data, and back off to lower-order model otherwise.
- interpret $(1-\lambda_{w_{i-n+1}^{i-1}})$ as the probability of recurring to the lower-order model, i.e. that a word not seen after w_{i-n+1}^{i-1} in training data occurs after that history in test data.
- replace this by the number of unique words that follow the history w_{i-n+1}^{i-1} in the training data (i.e., how we would be surprised to see a new word following our n-gram).

2.2 The method

Let us define the number of possible extensions of a history $w_1, ..., w_{n-1}$ in training data as:

$$N_{1+}(w_1, ..., w_{n-1}, \bullet) = |\{w_n : c(w_1, ..., w_{n-1}, w_n) > 0\}| \qquad (2)$$

Based on this, we define the lambda parameters as:

$$1 - \lambda_{w_1,\ldots,w_{n-1}} = \frac{N_{1+}(w_1, ..., w_{n-1}, \bullet)}{N_{1+}(w_1, ..., w_{n-1}, \bullet) + \sum_{w_n} c(w_1, ..., w_{n-1}, w_n)}$$

Let us apply this to our two examples:

$$\begin{aligned} 1 - \lambda_{spite} &= \frac{N_{1+}(\text{spite}, \bullet)}{N_{1+}(\text{spite}, \bullet) + \sum_{w_n} c(\text{spite}, w_n)} \\ &= \frac{9}{9 + 993} = 0.00898 \\ 1 - \lambda_{constant} &= \frac{N_{1+}(\text{constant}, \bullet)}{N_{1+}(\text{constant}, \bullet) + \sum_{w_n} c(\text{constant}, w_n)} \\ &= \frac{415}{415 + 993} = 0.29474 \end{aligned}$$

3 Absolute discounting

Remember that the idea of smoothing is to redistribute some probability mass by taking from the non-zero counts. Practitioners have noticed that the discount value $1 - d_r$ in the GT smoothing can often be approximated by an "almost constant" value for large r.
Idea: use the Jelinek-Mercer interpolation as before but set $\lambda_{w_{i-n+1}^{i-1}}$ to $\frac{\max\{c(w_{i-n+1}^{i}) - D, 0\}}{\sum_{w_i} c(w_{i-n+1}^{i})}$.
Then

$$1 - \lambda_{w_{i-n+1}^{i-1}} = \frac{D}{\sum_{w_i} c(w_{i-n+1}^{i})} N_{1+}(w_{i-n+1}^{i-1} \bullet), \tag{3}$$

where D can be estimated using deleted estimation.

4 "Kneser-Ney"-smoothing

Previously we always modified the higher-order part of the interpolation (Katz, Witten-Bell); the lower-order part is just as important and should be optimized.

Example: Diversity of Histories Consider the word "York". It is a fairly frequent word in the Europarl corpus, occurring 477 times. This is as frequent as the words "foods", "indicates" or "providers". In a unigram language model, this is a respectable probability. However, it almost always directly follows "New" (473 times). Recall: The unigram model is only used, if the bigram model is inconclusive. "York" is very unlikely to be the second word in an unseen bigram in a back-off unigram model, therefore "York" should have a low probability. "Kneser-Ney"-smoothing is an interpolation method with **absolute discounting** and **modified counts**.

4.1 Idea

Assume for simplicity a two-bigram model. The idea is to set the unigram probability to the number of different words that it follows instead of simply number of occurrences. To formalize this, we constrain the smoothed distribution to have unigram marginals matching the empirical unigram distribution.

$$\frac{c(w_i)}{\sum_{w_i} c(w_i)} = \sum_{w_{i-1}} \hat{p}(w_{i-1}w_i) = \sum_{w_{i-1}} \hat{p}(w_i|w_{i-1})p(w_{i-1}) = \sum_{w_{i-1}} \hat{p}(w_i|w_{i-1}) \frac{c(w_{i-1})}{\sum_{w_{i-1}} c(w_{i-1})},$$

where in the last step we assumed that unigram distribution is the empirical one.
From (3) we have

$$\hat{p}(w_i|w_{i-n+1}^{i-1}) = \frac{\max(c(w_{i-n+1}^i) - D, 0)}{c(w_{i-n+1}^i)} + \frac{DN_{1+}(w_{i-n+1}^{i-1}\bullet)}{\sum_{w_i} c(w_{i-n+1}^i)} \hat{p}(w_i|w_{i-n+2}^{i-1})$$

Substituting into constrain and simplifying:

$$c(w_i) = c(w_i) - N_{1+}(\bullet w_i)D + D\hat{p}(w_i)N_{1+}(\bullet\bullet),$$

where

$$N_{1+}(\bullet w_i) = |\{w_{i-1} : c(w_{i-1}w_i) > 0\}|,$$
$$N_{1+}(\bullet\bullet) = |\{(w_{i-1}, w_i) : c(w_{i-1}w_i) > 0\}|.$$

Solving for $\hat{p}(w_i)$ we obtain

$$\hat{p}(w_i) = \frac{N_{1+}(\bullet w_i)}{N_{1+}(\bullet\bullet)}.$$

For the general case of higher-order models:

$$\hat{p}(w_i|w_{i-n+2}^{i-1}) = \frac{N_{1+}(\bullet w_{i-n+2}^i)}{N_{1+}(\bullet w_{i-n+2}^{i-1}\bullet)}.$$

where

$$N_{1+}(\bullet w_{i-n+2}^i) = |\{w_{i-n+1} : c(w_{i-n+1}^i) > 0\}|,$$
$$N_{1+}(\bullet w_{i-n+2}^{i-1}\bullet) = |\{(w_{i-n+1}, w_i) : c(w_{i-n+1}^i) > 0\}| = \sum_{w_i} N_{1+}(\bullet w_{i-n+2}^i).$$

5 Modified "Kneser-Ney"-smoothing

The most used smoothing for n-gram models in practice .

$$\hat{p}(w_i|w_{i-n+1}^{i-1}) = \frac{c(w_{i-n+1}^{i}) - D(c(w_{i-n+1}^{i}))}{c(w_{i-n+1}^{i})} + \gamma(w_{i-n+1}^{i-1})\hat{p}(w_i|w_{i-n+2}^{i-1})$$

Idea Use 3 discount factors D_1, D_2, D_{3+} respectively for uni-, bi-grams and tri(-and-higher)-grams, so that

$$D = D(c) = D_1\mathbf{1}[c = 1] + D_2\mathbf{1}[c = 2] + D_{3+}\mathbf{1}[c > 2].$$

5.0.1 Evaluation of Smoothing Methods

Perplexity for language models trained on the Europarl corpus:

Smoothing method	**bigram**	**trigram**	**4-gram**
Good-Turing	96.2	62.9	59.9
Witten-Bell	97.1	63.8	60.4
Modified Kneser-Ney	95.4	61.6	58.6
Interpolated Modified Kneser-Ney	94.5	59.3	54.0

5.1 Managing the size of the model

PROBLEM: Millions to billions of words are easy to get (trillions of English words available on the web), but huge language models do not fit into RAM.

5.1.1 Estimation on Disk

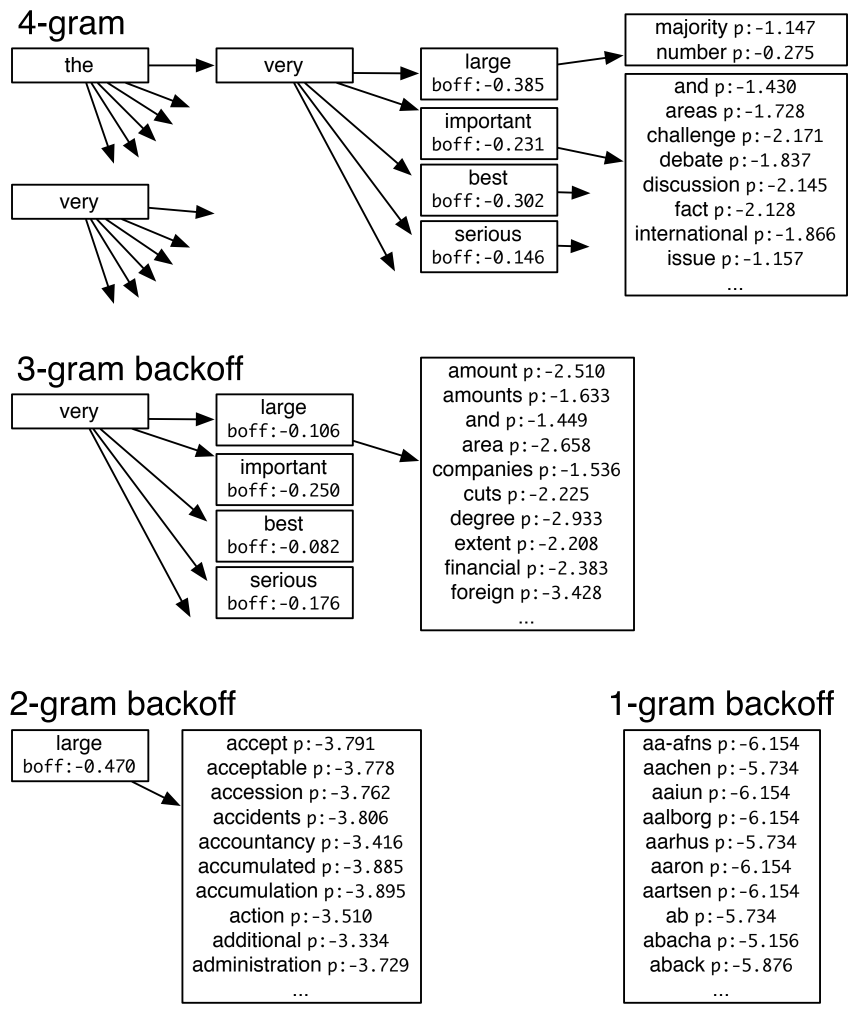
Maximum likelihood estimation:

$$p(w_n|w_1, ..., w_{n-1}) = \frac{\text{count}(w_1, ..., w_n)}{\text{count}(w_1, ..., w_{n-1})} \tag{4}$$

This can be done separately for each history $w_1, ..., w_{n-1}$.
We extract all n-grams into files on-disk and sort by them history on our disk and only keep n-grams with shared history in RAM.
Smoothing techniques may require additional statistics.

5.1.2 Efficient Data Structures

We need to store large language models both memory-efficiently and in a way that allows us to retrieve probabilities in a time-efficient manner. Typically, this is done using a data structure known as a **trie**.
Imagine we need to store the probabilities for "the very large majority" and "the very large number". Both share the same history "the very large". With this, there is no need to store their history twice.



Language model probabilities stored in a trie.
Backoff from 4-gram to 3-gram:

$$\begin{aligned} p_{\text{LM}}(\text{amount}|\text{the very large}) =& \text{backoff}(\text{the very large}) \\ &\cdot p_3(\text{amount}|\text{very large}) \\ =& \exp(-0.385 + -2.510) \end{aligned} \tag{5}$$

Backoff from 4-gram to 2-gram:

$$\begin{aligned} p_{\text{LM}}(\text{action}|\text{the very large}) =& \text{backoff}(\text{the very large}) \\ &\cdot \text{backoff}(\text{very large}) \\ &\cdot p_2(\text{action}|\text{large}) \\ =& \exp(-0.385 + -0.106 + -3.510) \end{aligned} \tag{6}$$

5.1.3 Fewer Bits to store Probabilities

Index for words
Two bytes allow a vocabulary of $2^{16} = 65,536$ words, but typically more bytes are needed. We can be even more efficient by employing Huffman coding to use fewer bits for frequent words.
Probabilities
Typically probabilities are stored in log format as floats (4 or 8 bytes). Quantization of probabilities allows us to use even less memory, maybe just 4-8 bits.

5.1.4 Reducing Vocabulary Size

Each number is treated as a separate token and there are theoretically infinitely many of them. Since different numbers do not behave that differently from each other, we replace them with a number token: NUM
But we want our language model to prefer:

$$p_{\text{LM}}(\text{I pay 950.00 in May 2007}) > p_{\text{LM}}(\text{I pay 2007 in May 950.00})$$

This is not possible with the number token:

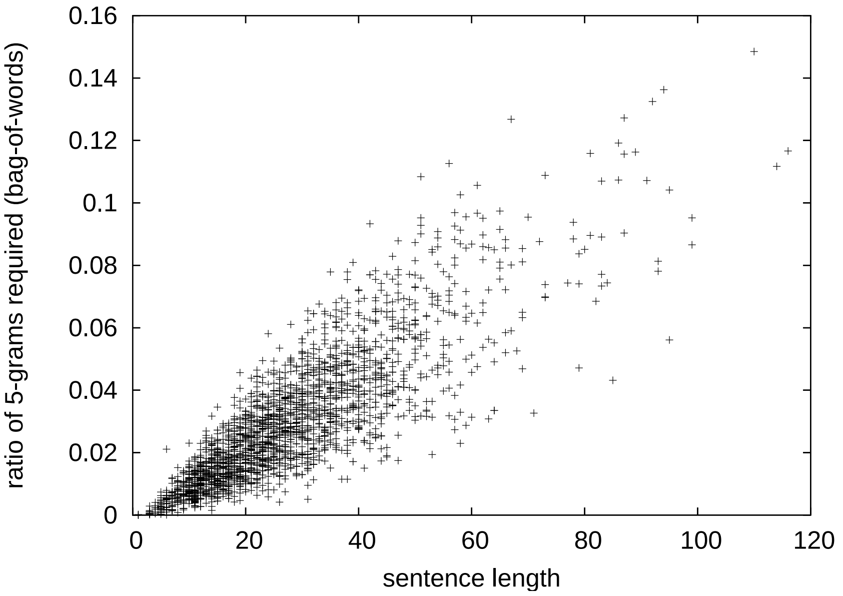
$$p_{\text{LM}}(\text{I pay NUM in May NUM}) = p_{\text{LM}}(\text{I pay NUM in May NUM})$$

Therefore we replace each digit with a unique symbol, e.g., @ or 5, to retain some distinctions:

$$p_{\text{LM}}(\text{I pay 555.55 in May 5555}) > p_{\text{LM}}(\text{I pay 5555 in May 555.55})$$

5.1.5 Filtering irrelevant N-Grams

We use the language model in decoding. We only produce English words in the translation options and filter the language model down to n-grams containing only those words.



Ratio of 5-grams needed to all 5-grams in language model (by sentence length).

5.2 Summary

Language models:
How likely is a string of English words good English?
- N-gram models (Markov assumption)
- Perplexity

Count smoothing
- add-one, add-α
- deleted estimation

Interpolation and back-off
- Good Turing
- Witten-Bell
- Kneser-Ney

Managing the size of the model

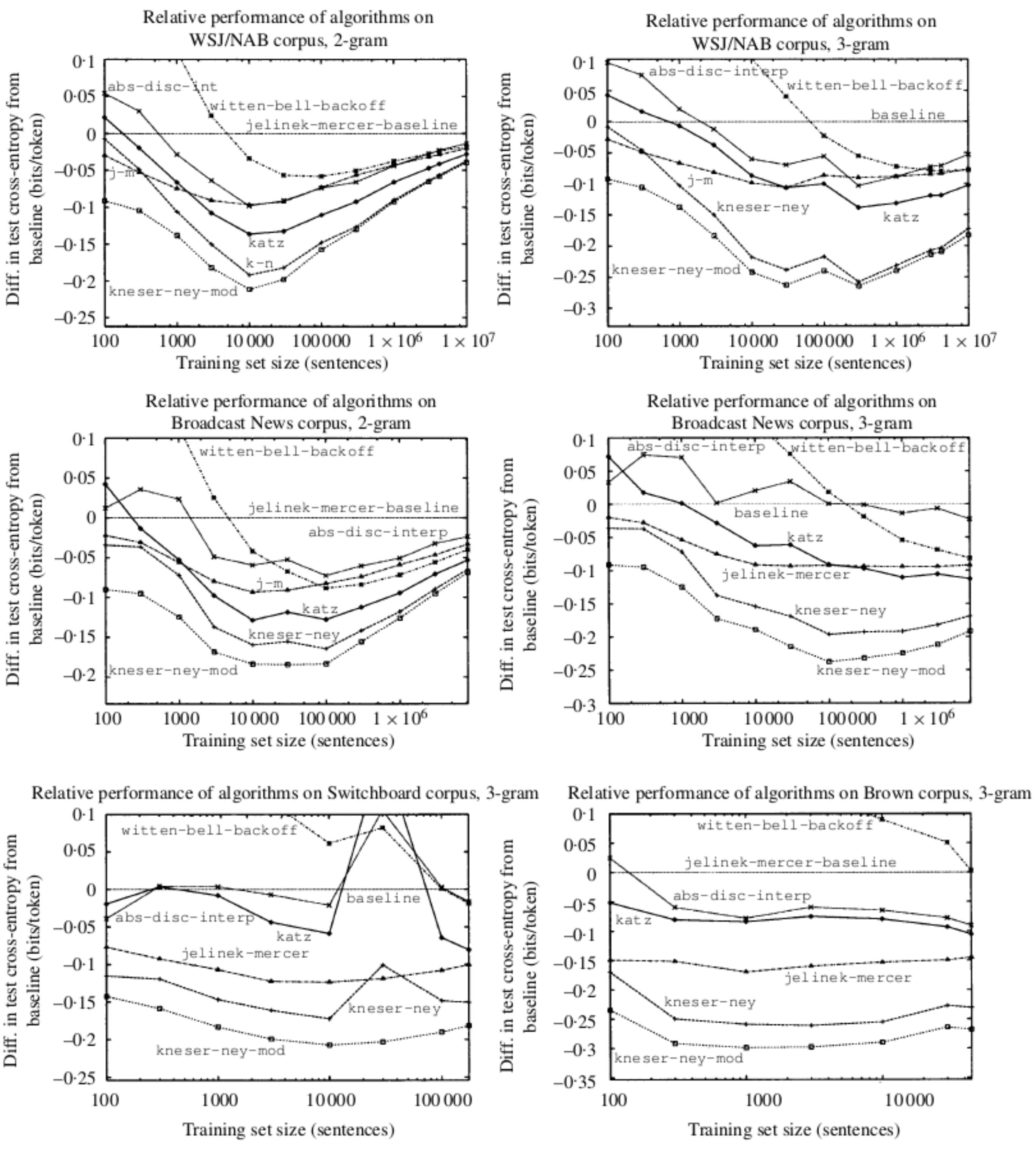


Figure 6. Performance relative to baseline of various algorithms on all four corpora, bigram and trigram models, over various training set sizes.

Abbildung 1: From Chen&Goodman'99