Language Models

Marcello Federico
FBK-irst Trento, Italy

MT Marathon 2010

Outline

- Role of LM in ASR and SMT
- N-gram Language Models
- Evaluation of Language Models
- Frequency Smoothing
- Frequency Discounting
- Is smoothing necessary?
- ARPA LM representation
- New IRSTLM Toolkit

Fundamental Equation of ASR

Goal: find the words $\mathbf{w}^*$ in a speech signal $\mathbf{x}$ such that:

$$\mathbf{w}^* = \arg \underset{\mathbf{w}}{max} \Pr(\mathbf{x} \mid \mathbf{w}) \Pr(\mathbf{w}) \quad (1)$$

Problems:

- language modeling (LM): estimating $\Pr(\mathbf{w})$
- acoustic modeling (AM): estimating $\Pr(\mathbf{x} \mid \mathbf{w})$
- search problem: computing (1)

AM sums over hidden state sequences $\mathbf{s}$ a Markov process of $(\mathbf{x}, \mathbf{s})$ from $\mathbf{w}$

$$\Pr(\mathbf{x} \mid \mathbf{w}) = \sum_{\mathbf{s}} \Pr(\mathbf{x}, \mathbf{s} \mid \mathbf{w})$$

Hidden Markov Model: hidden states "link" speech frames to words.

Fundamental Equation of SMT

Goal: find the English string $\mathbf{f}$ translating the foreign text $\mathbf{f}$ such that:

$$\mathbf{e}^* = \arg \underset{\mathbf{e}}{max} \Pr(\mathbf{f} \mid \mathbf{e}) \Pr(\mathbf{e}) \tag{2}$$

Problems:

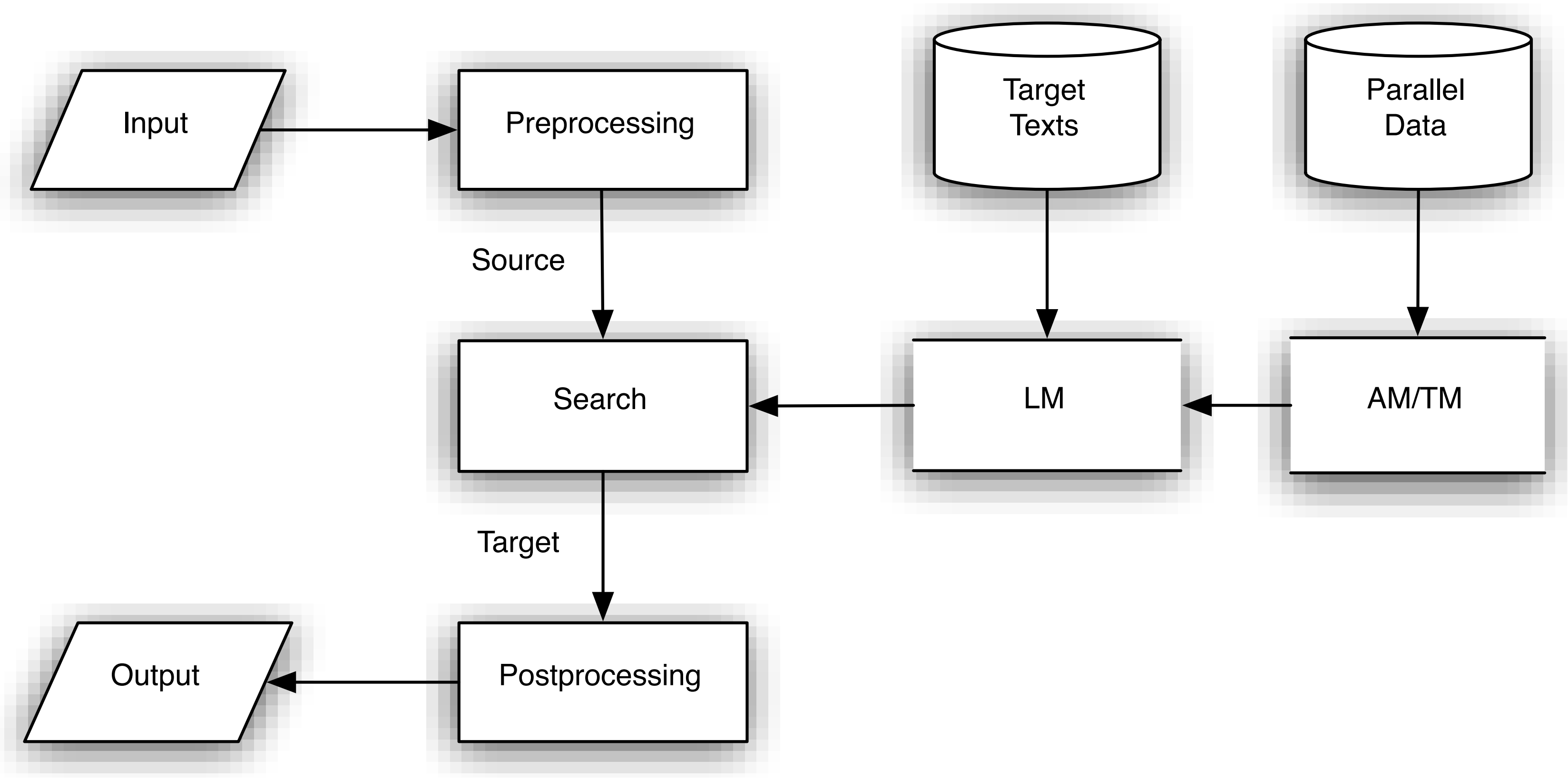
- language modeling (LM): estimating $\Pr(\mathbf{e})$
- translation modeling (TM): estimating $\Pr(\mathbf{f} \mid \mathbf{e})$
- search problem: computing (2)

TM sums over hidden alignments $\mathbf{a}$ a stochastic process generating $(\mathbf{f}, \mathbf{a})$ from $\mathbf{e}$.

$$\Pr(\mathbf{f} \mid \mathbf{e}) = \sum_{\mathbf{a}} \Pr(\mathbf{f}, \mathbf{a} \mid \mathbf{e})$$

Alignment Models: hidden alignments "link" foreign words with English words.

ASR and MT Architectures



- Parallel data are samples of observations $(\mathbf{x}, \mathbf{w})$ and $(\mathbf{f}, \mathbf{e})$
- AM and TM can be machine-learned without observing $\mathbf{s}$ and $\mathbf{a}$.

Log-linear phrase-based SMT

- Translation hypotheses are ranked by:

$$\mathbf{e}^* = \arg\max_{\mathbf{e}} \sum_{\mathbf{a}} \sum_{k} \lambda_k \log h_k(\mathbf{e}, \mathbf{f}, \mathbf{a})$$

- Phrases are finite string (cf. n-grams)
- Hidden variable $\mathbf{a}$ embeds:
  – segmentation of $\mathbf{f}$ and $\mathbf{e}$ into phrases
  – alignment of phrases of $\mathbf{f}$ with phrases of $\mathbf{e}$
- Feature functions $h_k()$ include:
  – Translation Model: appropriateness of phrase-pairs
  – Distortion Model: word re-ordering
  – Language Model: fluency of target string
  – Length Model: number of target words
- Role of the LM is exactly the same as for the classical approach:
  – to score translations generated incrementally by the search algorithm!

N-gram Language Model

Goal: given a text $w_1^T = w_1 \ldots, w_t, \ldots, w_T$ we can compute its probability by:

$$\Pr(w_1^T) = \Pr(w_1) \prod_{t=2}^{T} \Pr(w_t \mid h_t) \tag{3}$$

where $h_t = w_1, \ldots, w_{t-1}$ indicates the history of word w_t.

- $\Pr(w_t \mid h_t)$ becomes difficult to estimate as the history h_t grows .
- hence, we take the n-gram approximation $h_t \approx w_{t-n+1} \ldots w_{t-1}$

  e.g. Full history: $\Pr(\texttt{Parliament} \mid \texttt{I declare resumed the session of the European})$
  $3 - gram$: $\Pr(\texttt{Parliament} \mid \texttt{the European})$

The choice of n determines the complexity of the LM (# of parameters):

- bad: no magic recipe about the optimal order n for a given task
- good: language models can be evaluated quite cheaply

Language Model Evaluation

- Indirect: impact on task (e.g. BLEU score for MT)
- Direct: capability of predicting words

The perplexity (PP) measure is defined as: [1]

$$PP = 2^{LP} \quad \text{where} \quad LP = -\frac{1}{M}\log_2 p(w_1^M) \tag{4}$$

- w_1^M is a sufficiently long test sample and $p(w_1^M)$ is the LM probability

Properties:

- $0 \leq PP \leq |V|$ (size of the vocabulary V)
- predictions are as good as guessing among PP equally likely options

Good: there is typical strong correlation between PP and BLUE scores!

[1] [Exercise 1. Find PP of 1-gram LM $p(\texttt{T})$=0.5 on T H T H T H T T H T T H.]

N-gram Probabilities

Estimating n-gram probabilities is not trivial due to:

- parameter space: with 10,000-word V we can form one trillion 3-grams!
- data sparseness: most of 3-grams are rare events even in large corpora.

Relative frequency estimate: MLE of any discrete conditional distribution is:

$$f(w \mid x\ y) = \frac{c(x\ y\ w)}{\sum_w c(x\ y\ w)}$$

where counts $c(\cdot)$ are taken over a large training corpus.

Problem: relative frequencies in general over-fit the training data

- if the test sample contains a "new" n-gram PP $\rightarrow +\infty$
- with 4-grams or 5-grams LM this is largely the most frequent case!

We need frequency smoothing!

Frequency Smoothing

Issue: $f(w \mid x\ y) > 0$ only if w was observed after $x\ y$ in the training data.

Idea: for each w take off some fraction of probability from $f(w \mid x\ y)$ and redistribute the total to words never observed after $x\ y$.

- the discounted frequency $f^*(w \mid x\ y)$ satisfies:

$$0 \leq f^*(w \mid x\ y) \leq f(w \mid x\ y) \qquad \forall x, y, w \in V$$

  Notice: in general $f^*(w \mid x\ y)$ does not sum up to 1!

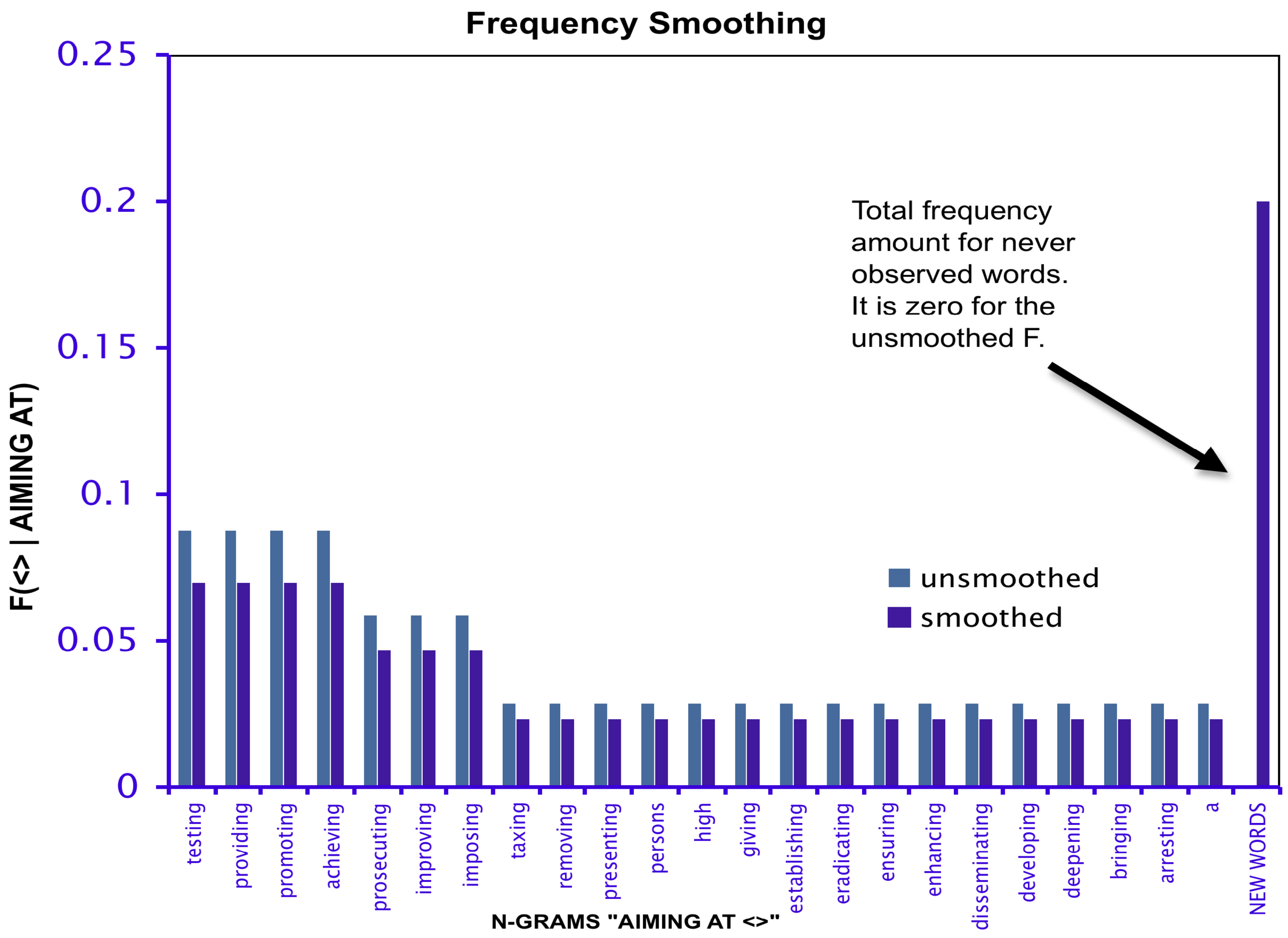
- the "total discount" is called zero-frequency probability $\lambda(x\ y)$[2]:

$$\lambda(x\ y) = 1.0\ -\ \sum_{w \in V} f^*(w \mid x\ y)$$

How to redistribute the total discount?

[2]Notice: by convention $\lambda(x\ y) = 1$ if $f(w \mid x\ y) = 0$ for all w, i.e. $c(x\ y) = 0$.

Discounting Example



Frequency Smoothing

Insight: redistribute $\lambda(x\ y)$ according to the lower-order smoothed frequency.

Two major hierarchical schemes to compute the smoothed frequency $p(w \mid x\ y)$:

- Back-off, i.e. select the best available n-gram approximation:

$$p(w \mid x\ y) = \begin{cases} f^*(w \mid x\ y) & \text{if } f^*(w \mid x\ y) > 0 \\ \alpha_{xy}\lambda(x\ y)p(w \mid y) & \text{otherwise} \end{cases} \quad (5)$$

  where α_{xy} is an appropriate normalization term.[3]

- Interpolation, i.e. sum up the two approximations:

$$p(w \mid x\ y) = f^*(w \mid x\ y) + \lambda(x\ y)p(w \mid y). \quad (6)$$

Smoothed frequencies are learned bottom-up, starting from 1-grams ...

[3][Exercise 2. Find and expression for α_{xy} s.t. $\sum_w p(w \mid x\ y) = 1$.]

Frequency Smoothing of 1-grams

Unigram smoothing permits to treat out-of-vocabulary (OOV) words in the LM.

Assumptions:

- $|U|$ is an upper-bound estimate of the size of language vocabulary
- $f^*(w)$ is strictly positive on the observed vocabulary V
- λ is the total discount reserved to OOV words

Then: 1-gram back-off and interpolation collapse to:

$$p(w) = \begin{cases} f^*(w) & \text{if } \; w \in V \\ \lambda(|U| - |V|)^{-1} & \text{otherwise} \end{cases} \tag{7}$$

Notice: LMs make also other approximations when an OOV word x appears:

$$p(w \mid h_1 \; x \; h_1) = p(w \mid h_2) \;\; \text{and} \;\; p(x \mid h) = p(x)$$

Important: use a common value $|U|$ when comparing/combining different LMs!

Discounting Methods

Witten-Bell estimate (WB) [Witten and Bell, 1991]

- Insight: learn $\lambda(x\ y)$ by counting "new word" events in 3-grams `x y *`
  - corpus: `x y u x x y t t x y u w x y w x y t u x y u x y t`
  - then $\lambda(x\ y) \propto$ number of "new word" events (i.e. 3)
  - and $f^*(w \mid x\ y) \propto$ relative frequency (linear discounting)
- Solution:

$$\lambda(x\ y) = \frac{n(x\ y\ *)}{c(x\ y) + n(x\ y\ *)} \quad \text{and} \quad f^*(w \mid xy) = \frac{c(x\ y\ w)}{c(x\ y) + n(x\ y\ *)}$$

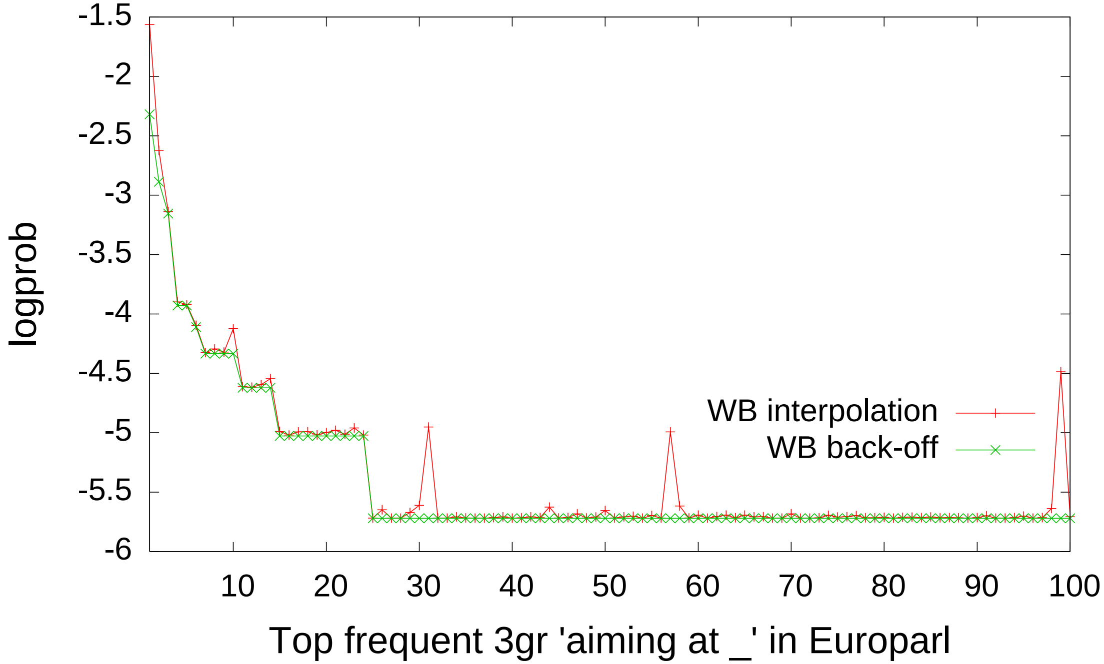
where $c(x\ y) = \sum_w c(x\ y\ w)$ and $n(x\ y\ *) = |\{w : c(x\ y\ w) > 0\}|$. [4]

- Pros: easy to compute, robust for small corpora, works with artificial data.
- Cons: underestimates probability of frequent n-grams

[4] [Exercise 3. Compute $f^*(\mathtt{u} \mid \mathtt{x\ y})$ with WB on the above artificial text.]

Discounting Methods

- interpolation and back-off with WB discounting
- trigram LMs estimated on the English Europarl corpus
- logprobs of 3-grams of type `aiming at _` observed in training



- peaks correspond to very probable 2-grams interpolated with f^* respectively: `at that`, `at national`, `at European`
- Practically, interpolation and back-off perform similarly

Absolute Discounting (AD) [Ney and Essen, 1991]

- Insight:
  - discount by subtracting a small constant β $(0 < \beta \leq 1)$ from each counts
  - estimate β by maximizing the leaving-one-out likelihood of the training data
- Solution: (notice: one distinct β for each n-gram order)

$$f^*(w \mid x\ y) = max\left\{\frac{c(xyw) - \beta}{c(xy)}, 0\right\} \text{ which gives } \quad \lambda(xy) = \beta\frac{\sum_{w:c(xyw)>1} 1}{c(xy)}$$

where $\beta \approx \frac{n_1}{n_1+2n_2} < 1$ and $n_r = |\{x\ y\ w : c(x\ y\ w) = r\}|$. [5]

- Pros: easy to compute, accurate estimate of frequent n-grams.
- Cons: problematic with small and artificial samples.

[5] [Exercise 4. Given the text in WB slide find the number of 3-grams, n_1, n_2, β, $f^*(\mathtt{w} \mid \mathtt{x\ y})$ and $\lambda(\mathtt{x\ y})$]

Discounting Methods

Kneser-Ney method (KN) [Kneser and Ney, 1995]

- Insight: 2-grams counts should be correspond to the "back-off" cases
  - count all "back-off" events in 3-grams of type `* y w` (cf. WB method)
  - corpus: `x y w x t y w t x y w u y w t y w u x y w u u y w`
  - corrected counts $n(*\ y\ w)$ = number of observed back-offs (i.e. 3)

- Solution: (for 3-gram normal counts)

$$f^*(w \mid y) = max\left\{\frac{n(*\ y\ w) - \beta}{n(*\ y\ *)}, 0\right\} \text{ which gives } \quad \lambda(y) = \beta\frac{\sum_{w:n(*\ y\ w)>1} 1}{n(*\ y\ *)}$$

where $n(*\ y\ w) = |\{x : c(x\ y\ w) > 0\}|$ and $n(*\ y\ *) = |\{x\ w : c(x\ y\ w) > 0\}|$

- Pros: better back-off probabilities, can be applied to other methods
- Cons: LM cannot be used to compute lower order n-gram probs

Discounting Methods

Modified Kneser-Ney (MKN) [Chen and Goodman, 1999]

- Insight: specific discounting coefficients for unfrequent n-grams
- Solution:

$$f^*(w \mid x\ y) = \frac{c(x\ y\ w) - \beta(c(x\ y\ w))}{c(x\ y)}$$

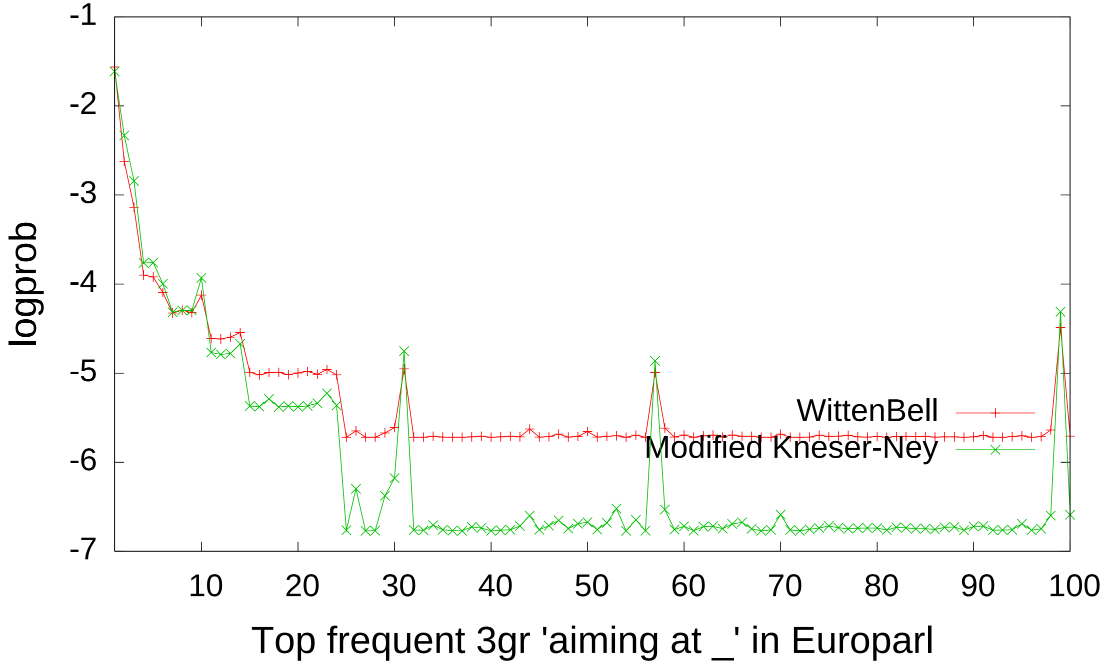
where $\beta(0) = 0$, $\beta(1) = D_1$, $\beta(2) = D_2$, $\beta(c) = D_{3+}$ if $c \geq 3$, coefficients are computed from n_r statistics, corrected counts used for lower order n-grams

- Pros: see previous + more fine grained smoothing
- Cons: see previous + more sensitiveness to noise

Important: LM interpolation with MKN is the most popular training method. Under proper training conditions it gives the best PP and BLEU scores!

Discounting Methods

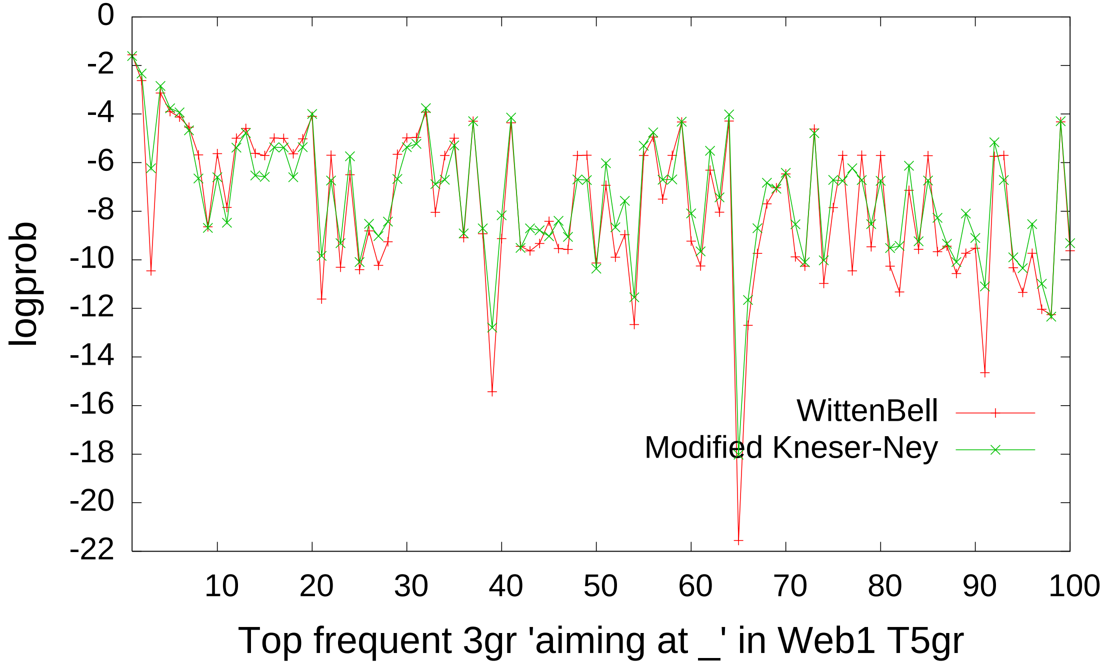
- interpolation with WB and MKN discounting trained on Europarl
- the plot shows the logprob of observed 3-grams of type `aiming at _`



- notice that for less frequent 3-grams WB assigns higher probability
- we have three very high peaks corresponding large corrected counts:
  `n(*at that)=665 n(* at national)=598 n(* at European)=1118`
- also an interesting peak at rank #26: n(* at very)=61

Discounting Methods

- train: interpolation with WB and MKN discounting on Europarl
- test: 3-grams of type `aiming at _` are from the Google 1TWeb sample

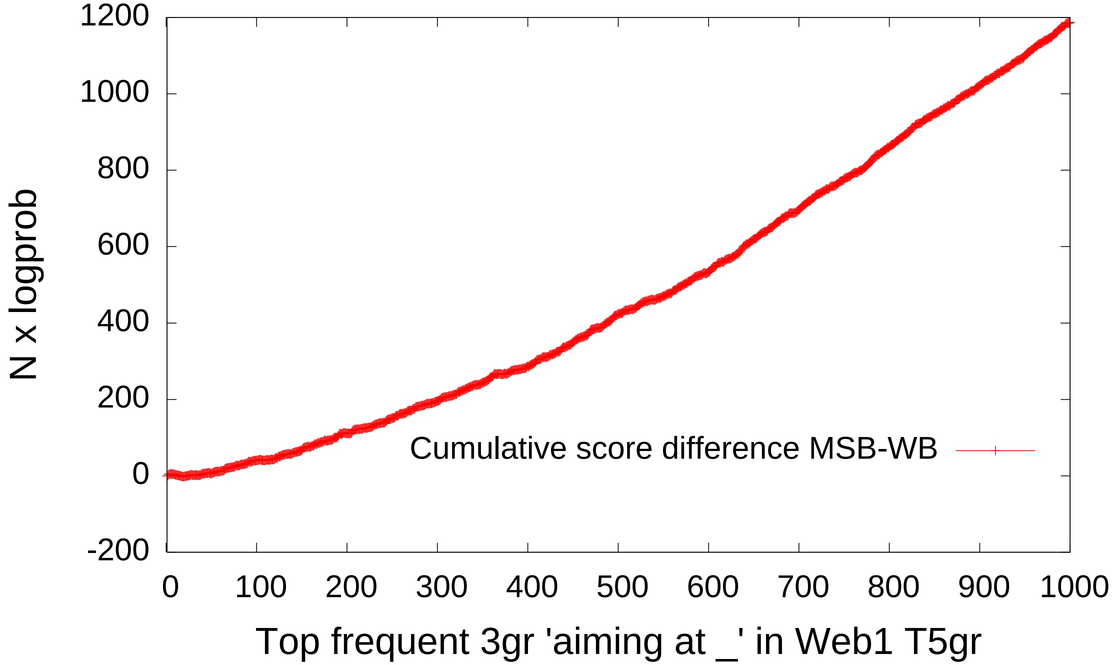


- the trend is the same but MKN outperforms WB smoothing

If you don't believe, check the next slide ....

Discounting Methods

- train: interpolation with WB and MKN discounting on Europarl
- test: 3-grams of type `aiming at _` are from the Google 1TWeb sample
- plot: cumulative score differences between MKN and WB on top 1000 3-grams



Is LM Smoothing Necessary?

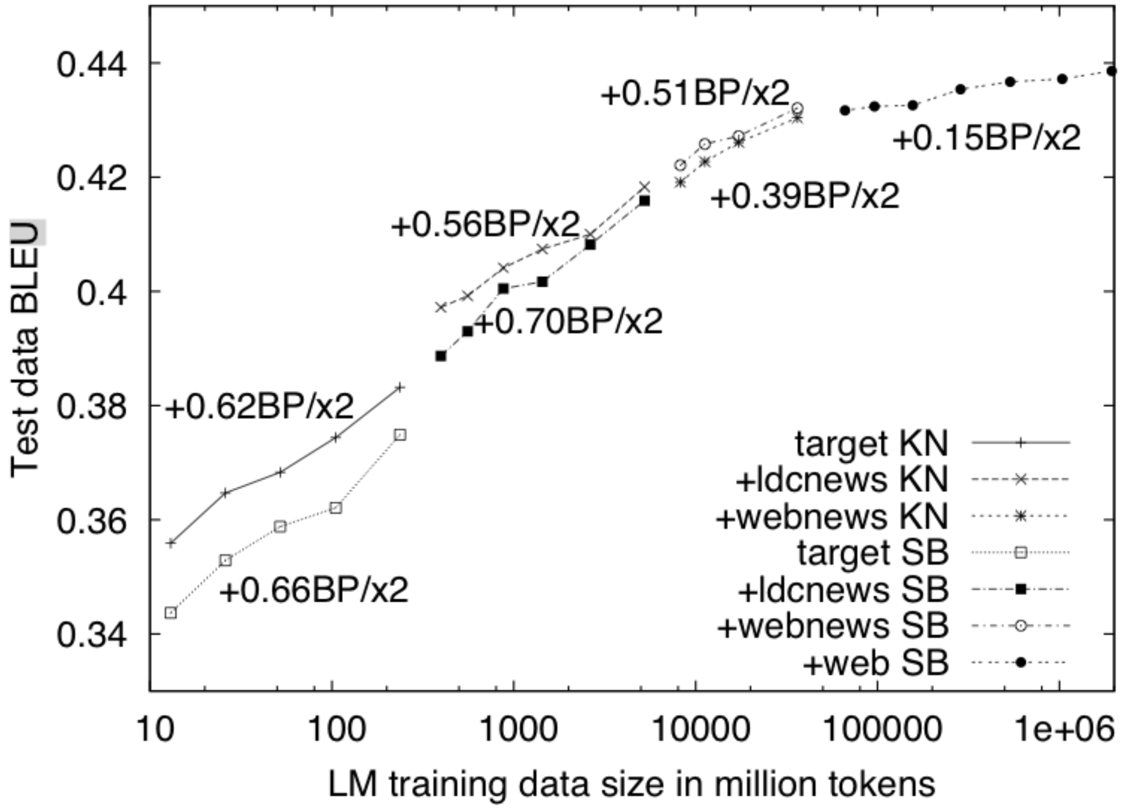
- LM Quantization [Federico and Bertoldi, 2006]
  - Idea: one codebook for each n-gram/back-off level
  - Pros: improves storage efficiency
  - Cons: reduces discriminatory power
  - Experiments with 8bit quantization on ZH-EN NIST task showed:
    * 2.7% BLUE drop with a 5-gram LM trained on 100M-words
    * 1.6% BLUE drop with a 5-gram LM trained on 1.7G words.
- Stupid back-off [Brants et al., 2007]
  - simple smoothing, no correct normalization

$$p(w \mid x\ y) = \begin{cases} f(w \mid x\ y) & \text{if } f(w \mid x\ y) > 0 \\ k \cdot p(w \mid y) & \text{otherwise} \end{cases} \tag{8}$$

where $k = 0.4$ and $p(w) = c(w)/N$.

Is LM Smoothing Necessary?

Stupid back-off (SB) versus Modified Kneser-Ney (KN)



From [Brants et al., 2007].

- Conclusion: proper smoothing useful up to 1 billion word training data?

ARPA File Format (srilm, irstlm)

Represents both interpolated and back-off n-gram LMs

- format: log(smoothed-freq) :: n-gram :: log(back-off weight)
- computation: look first for smoothed-freq, otherwise back-off

```
\data\
ngram 1= 86700
ngram 2= 1948935
ngram 3= 2070512
\1-grams:
-2.88382        !       -2.38764
-2.94351        world       -0.514311
-6.09691        dublin       -0.15553
...
\2-grams:
-3.91009     world !     -0.351469
-3.91257     hello world  -0.24
-3.87582     hello dublin    -0.0312
..
\3-grams:
-0.00108858     hello world  !
-0.000271867    , hi hello  !
...
\end\
```

```
logPr(!| hello dublin) = -0.0312 + logPr(!| dublin)
logPr(!| dublin) = -0.15553 - 2.88382
```

New IRSTLM Toolkit

Main Features [Federico et al., 2008]

- Single thread training for standard LMs
  - all major smoothing methods: WB, AD, MKN, ...
  - LM pruning, internal/external interpolation, adaptation
- Distributed training for huge LMs
  - simple smoothing methods: interpolation with WB, KN
  - split dictionary into balanced n-gram prefix lists
  - collect n-grams for each prefix lists
  - estimate and merge single LMs for each prefix list
- Space optimization
  - n-gram collection uses dynamic storage to encode counters
  - distributed LM estimation just requires reading disk files
  - probs and back-off weights are quantized
  - run-time LM data structure is loaded on demand
- LM caching
  - computations of probs, access to internal lists, LM states, ...

References

[Brants et al., 2007] Brants, T., Popat, A. C., Xu, P., Och, F. J., and Dean, J. (2007). Large language models in machine translation. In Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, pages 858–867, Prague, Czech Republic.

[Chen and Goodman, 1999] Chen, S. F. and Goodman, J. (1999). An empirical study of smoothing techniques for language modeling. Computer Speech and Language, 4(13):359–393.

[Federico and Bertoldi, 2006] Federico, M. and Bertoldi, N. (2006). How many bits are needed to store probabilities for phrase-based translation? In Proceedings on the Workshop on Statistical Machine Translation, pages 94–101, New York City. Association for Computational Linguistics.

[Federico et al., 2008] Federico, M., Bertoldi, N., and Cettolo, M. (2008). Irstlm: an open source toolkit for handling large scale language models. In Proceedings of Interspeech, Brisbane, Australia.

[Kneser and Ney, 1995] Kneser, R. and Ney, H. (1995). Improved backing-off for m-gram language modeling. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, volume 1, pages 181–184, Detroit, MI.

[Ney and Essen, 1991] Ney, H. and Essen, U. (1991). On smoothing techniques for bigram-based natural language modelling. In Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing, pages S12.11:825–828, Toronto, Canada.

[Witten and Bell, 1991] Witten, I. H. and Bell, T. C. (1991). The zero-frequency problem: Estimating the probabilities of novel events in adaptive text compression. IEEE Trans. Inform. Theory, IT-37(4):1085–1094.