Language Models

Marcello Federico
FBK-irst Trento, Italy

MT Marathon, Prague, 2013

Outline

- N-gram Language Models
- Evaluation of Language Models
- Smoothing Schemes
- Discounting Methods
- Class-based LMs
- Maximum-Entropy LMs
- Neural Network LMs
- Toolkits and ARPA file format

Linear phrase-based SMT

- Translation hypotheses are ranked by:

$$\mathbf{e}^* = \arg \underset{\mathbf{e},\mathbf{a}}{max} \sum_i \lambda_i h_i(\mathbf{e}, \mathbf{f}, \mathbf{a})$$

- Phrases are finite strings (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
- LM scores translations hypotheses left to right
  - that incrementally generated by the search algorithm!

N-gram Language Model

Given a text $\mathbf{w} = w_1 \ldots, w_t, \ldots, w_{|\mathbf{w}|}$ we can compute its probability by:

$$\Pr(\mathbf{w}) = \Pr(w_1) \prod_{t=2}^{|\mathbf{w}|} \Pr(w_t \mid h_t) \quad (1)$$

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

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

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

$$PP = 2^{CE} \quad \text{where} \quad CE = -\frac{1}{|\mathbf{w}|} \log_2 p(\mathbf{w}) \tag{2}$$

- $\mathbf{w}$ is a sufficiently long test sample and $p(\mathbf{w})$ 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 news: there is typical strong correlation between PP and BLEU scores!

[1] [Exercise 1. Find PP of 1-gram LM on the sequence `T H T H T H T T H T T H` for $p(\mathtt{T})$=0.3, $p(\mathtt{H})$=0.7 and $p(\mathtt{H})$=0.3, $p(\mathtt{T})$=0.7. Comment the results.]

N-gram Probabilities

Even estimating 3-gram probabilities[2] is not trivial due to:

- model complexity: e.g. 10,000 words correspond to 1 trillion 3-grams!
- data sparseness: e.g. most 3-grams are rare events even in huge corpora.

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

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

where n-gram counts $c(\cdot)$ are taken over the training corpus.

Problem: relative frequencies in general overfit the training data

- if the test sample contains a "new" n-gram PP $\rightarrow +\infty$
- this is largely the most frequent case for $n \geq 3$

We need frequency smoothing!

[2] We will often refer to trigrams just for simplicity, but without loss of generality.

Frequency Smoothing

Issue: $f(w \mid x\ y) > 0$ only for observed n-grams, i.e. $c(x\ y\ w) > 0$

Idea: take off some amount from $f(w \mid x\ y)$ and keep it for new n-grams $x\ y\ \cdot$.

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

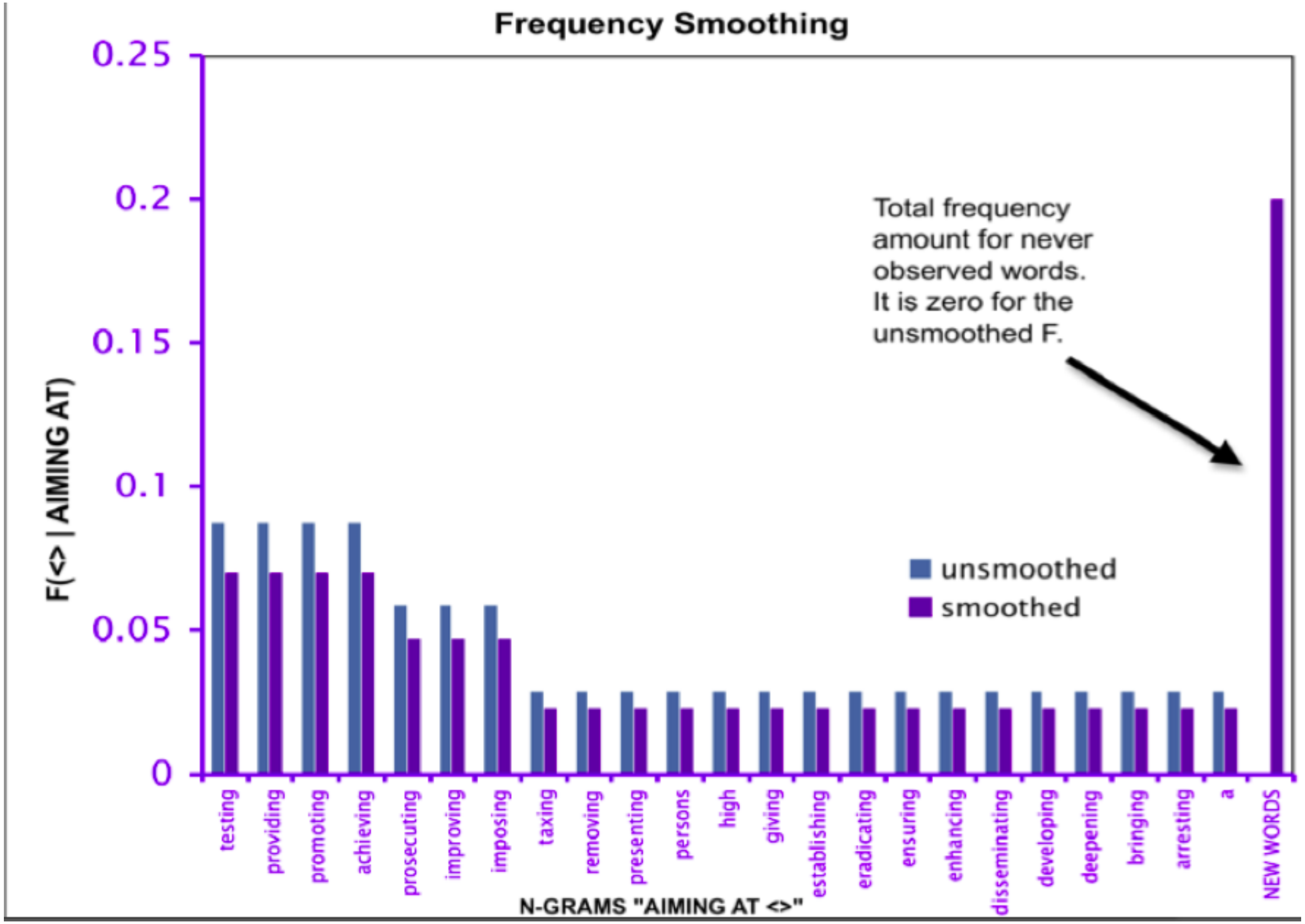
$$0 \leq f^*(w \mid x\ y) \leq f(w \mid x\ y) \qquad \forall x, y, w \in V$$

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

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

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



How to redistribute the total discount?

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:

$$p(w \mid x\ y) = \begin{cases} f^*(w \mid x\ y) & \text{if } \ f^*(w \mid x\ y) > 0 \\ \alpha_{xy} \times \lambda(x\ y) p(w \mid y) & \text{otherwise} \end{cases} \quad (3)$$

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

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

$$p(w \mid x\ y) = f^*(w \mid x\ y) + \lambda(x\ y) p(w \mid y). \quad (4)$$

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 is needed to cope with out-of-vocabulary (OOV) words

Assumptions:

- $|V|$: size of observed vocabulary; N: size of training corpus
- $|U|$: upper-bound estimate of size of true vocabulary
- Laplace smoothing: $f^*(w) = \dfrac{c(w)}{N + |V|}$ $\qquad \lambda = \dfrac{|V|}{N + |V|}$

Then: 1-gram back-off/interpolation schemes collapse to:

$$p(w) = \begin{cases} f^*(w) & \text{if } \; w \in V \\ \lambda \times \dfrac{1}{|U| - |V|} & \text{if } \; w \notin V \;\; \text{(i.e. OOV word)} \end{cases} \tag{5}$$

Important: use a common value $|U|$ when comparing/combining different LMs.
Note: IRSTLM permits to set $|U|$, SRILM uses a fixed $p(w \notin V)$

Discounting Methods

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

- Insight: count how often you would back-off after `x y` in the training data
  - 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`
  - assume $\lambda(x\ y) \propto$ number of back-offs (i.e. 3)
  - hence $f^*(w \mid x\ y) \propto$ 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 or noisy corpora
- Cons: underestimates probability of frequent n-grams

[4] [Exercise 3. Compute $f^*(\mathtt{u} \mid \mathtt{x\ y})$ with WB on the above corpus. Try to relate WB with Laplace smoothing.]

Discounting Methods

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

- Insight:
  - high counts are be more reliable than low counts
  - subtract a small constant β $(0 < \beta \leq 1)$ from each count
  - estimate β via MLE with leaving-one-out on 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 $\beta \approx \frac{n_1}{n_1+2n_2} \leq 1$ 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, $f^*(\mathtt{w} \mid \mathtt{x}\ \mathtt{y})$ and $\lambda(\mathtt{x}\ \mathtt{y})$]

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

- Insight: lower order counts are only used in case of back-off
  - estimate frequency of back-offs to `y w` in the training data (cf. WB)
  - 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`
  - replace $c(y\ w)$ with $n(*\ y\ w) = \#$ of observed back-offs (=3)

- Solution: (for 3-gram use absolute discounting)

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

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

- Pros: corrected counts can be used with other smoothing methods too
- 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 infrequent n-grams
  - introduce more parameters and estimate them with leaving-one-out
- Solution:

$$f^*(w \mid x\ y) = max\{\frac{c(x\ y\ w) - \beta(c(x\ y\ w))}{c(x\ y)}, 0\}$$

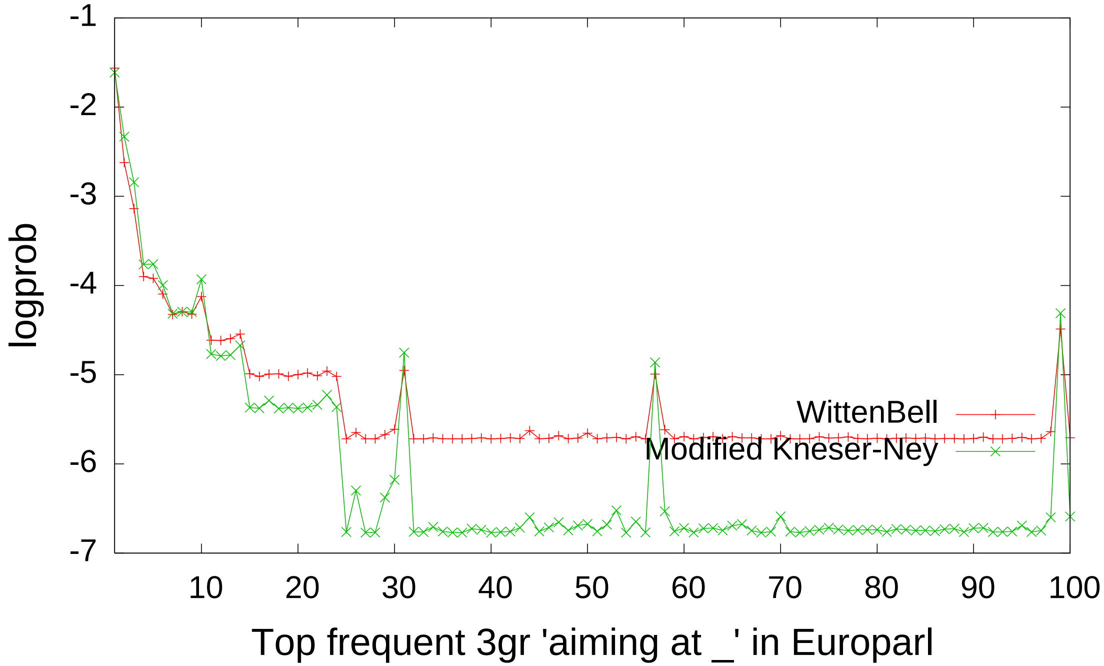
  where $\beta(0) = 0$, $\beta(1) = D_1$, $\beta(2) = D_2$, $\beta(c) = D_{3+}$ if $c \geq 3$, coefficients are computed from n_r statistics, corrected counts used for lower order n-grams

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

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

Discounting Methods

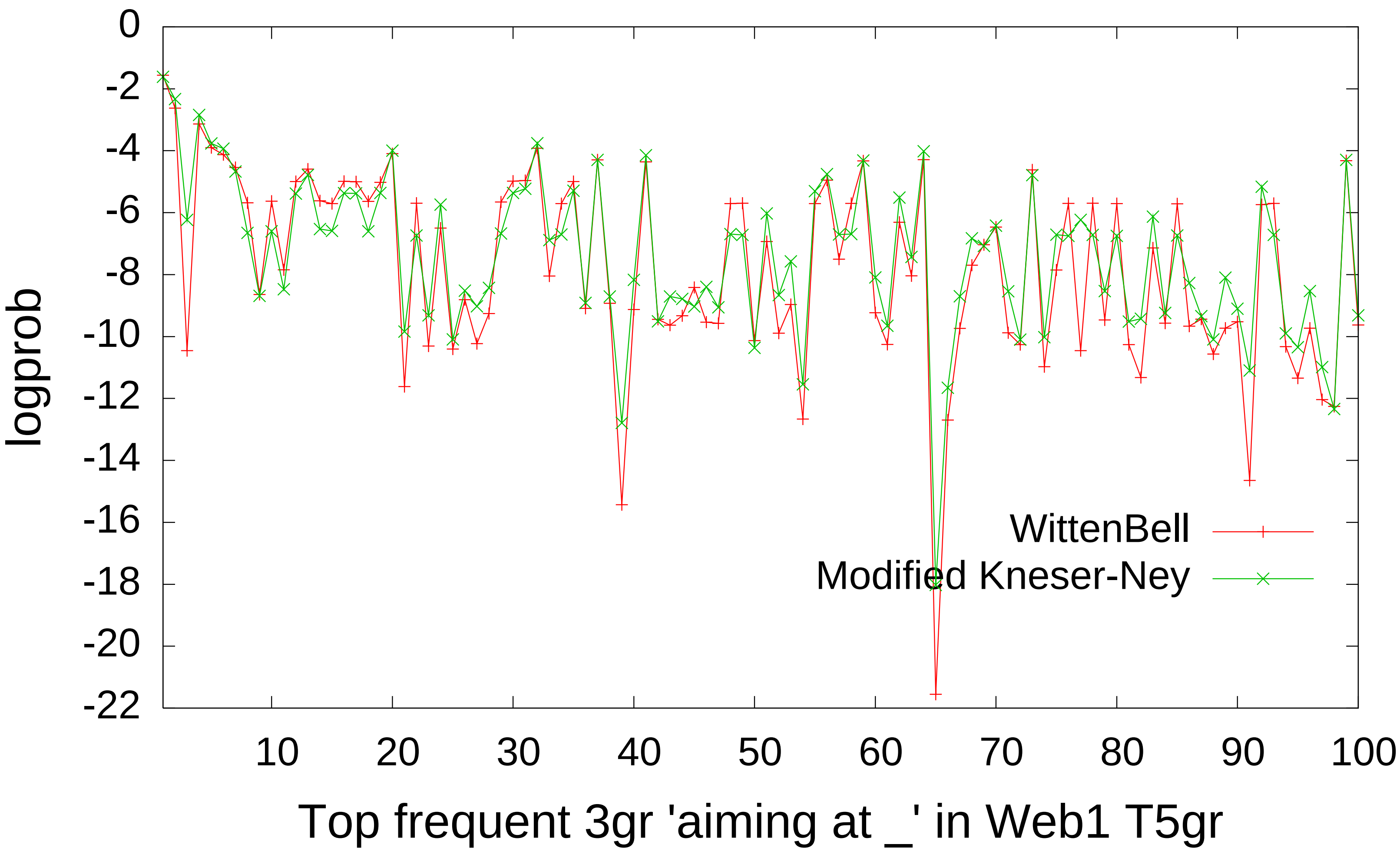
- Interpolation with WB and MKN discounting (Europarl corpus)
- The plot shows the logprob of observed 3-grams of type `aiming at _`



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

Discounting Methods

- Train: interpolation with WB and MKN discounting (Europarl corpus)
- Test: 3-grams of type `aiming at _` (Google 1TWeb corpus)
- The trend is similar but MKN outperforms WB



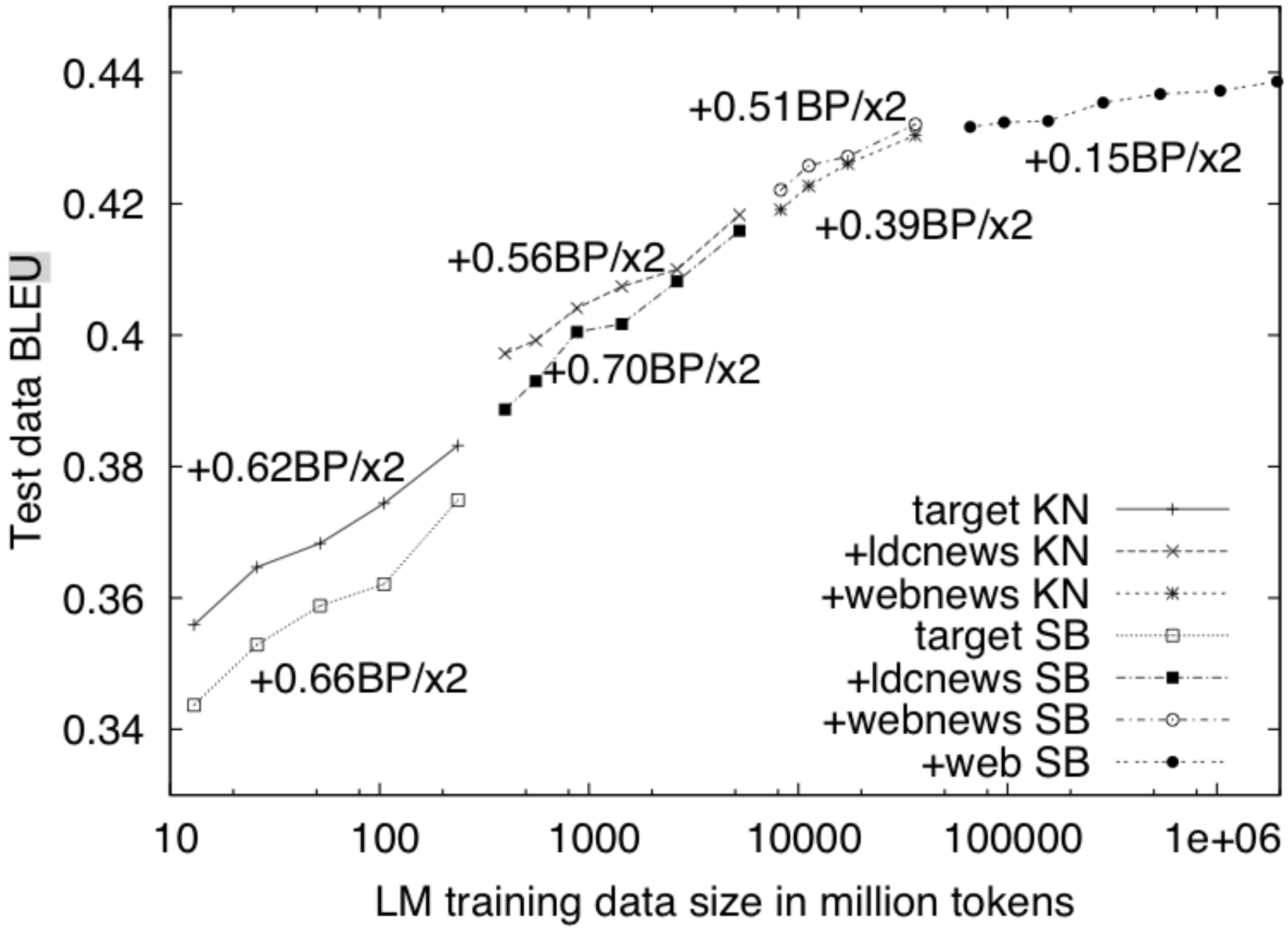
Approximate Smoothing

- 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% BLEU drop with a 5-gram LM trained on 100M-words
    * 1.6% BLEU drop with a 5-gram LM trained on 1.7G words.
- Stupid back-off [Brants et al., 2007]
  - no discounting, no corrected counts, no back-off normalization

$$p(w \mid x\ y) = \begin{cases} f(w \mid x\ y) & \text{if } f(w \mid x\ y) > 0 \\ k \cdot p(w \mid y) & \text{otherwise} \end{cases} \quad (6)$$

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

Is LM Smoothing Necessary?



From [Brants et al., 2007]. SB=stupid back-off, KN=modified Kneser-Ney

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

Class-based LMs

- Use less sparse representation of words than surface form words
  - e.g. part-of-speech, semantic classes, lemmas, automatic clusters
- Higher chance to match longer n-grams in test sequences
  - allows to model longer dependencies, to capture more syntax structure
- For a text $\mathbf{w}$ we assume a corresponding class sequence $\mathbf{g}$
  - ambiguous (e.g. POS) or deterministic (word classes)
- Factored LMs can be integrated into log-linear models with:
  - a word-to-class factored model: $\mathbf{f} \to \mathbf{e} \to \mathbf{g}$ with features:

$$h_1(\mathbf{f}, \mathbf{e}, \mathbf{a}) \ , h_2(\mathbf{e}, \mathbf{g}) \ , \underline{h_3(\mathbf{e})} \ , \underline{h_4(\mathbf{g})}$$

  - a word-class joint model $\mathbf{f} \to (\mathbf{e}, \mathbf{g})$ with features

$$h_1(\mathbf{f}, \mathbf{e}, \mathbf{g}, \mathbf{a}) \ , h_2(\mathbf{e}, \mathbf{g}) \ , \underline{h_3(\mathbf{e})}, \underline{h_4(\mathbf{g})}$$

Features of single sequences are log-probs of standard n-gram LMs.

Maximum Entropy N-gram LMs

- The n-gram prob is modeled with log-linear model [Rosenfeld, 1996]:

$$p_\lambda(w \mid h) = \frac{\exp\left(\sum_{r=1}^m \lambda_r h_r(h,w)\right)}{\sum_{w'} \exp\left(\sum_{r=1}^m \lambda_r h_r(h,w')\right)} = \frac{1}{Z(h)} \exp\left(\sum_{r=1}^m \lambda_r h_r(h,w)\right)$$

- $h_r(\cdot)$ are feature functions (arbitrary statistics), λ_r are free paramenters
- Features can model any dependency between w and h.
- Given feature functions and training sample $\mathbf{w}$, parameters can be estimated [Berger et al., 1996] by maximizing the posterior log-likelihood:

$$\hat{\lambda} = \arg \underset{\lambda \in \mathbf{R}^m}{max} \sum_{t=1}^{|\mathbf{w}|} \log p_\lambda(w_t \mid h_t) + \log q(\lambda)$$

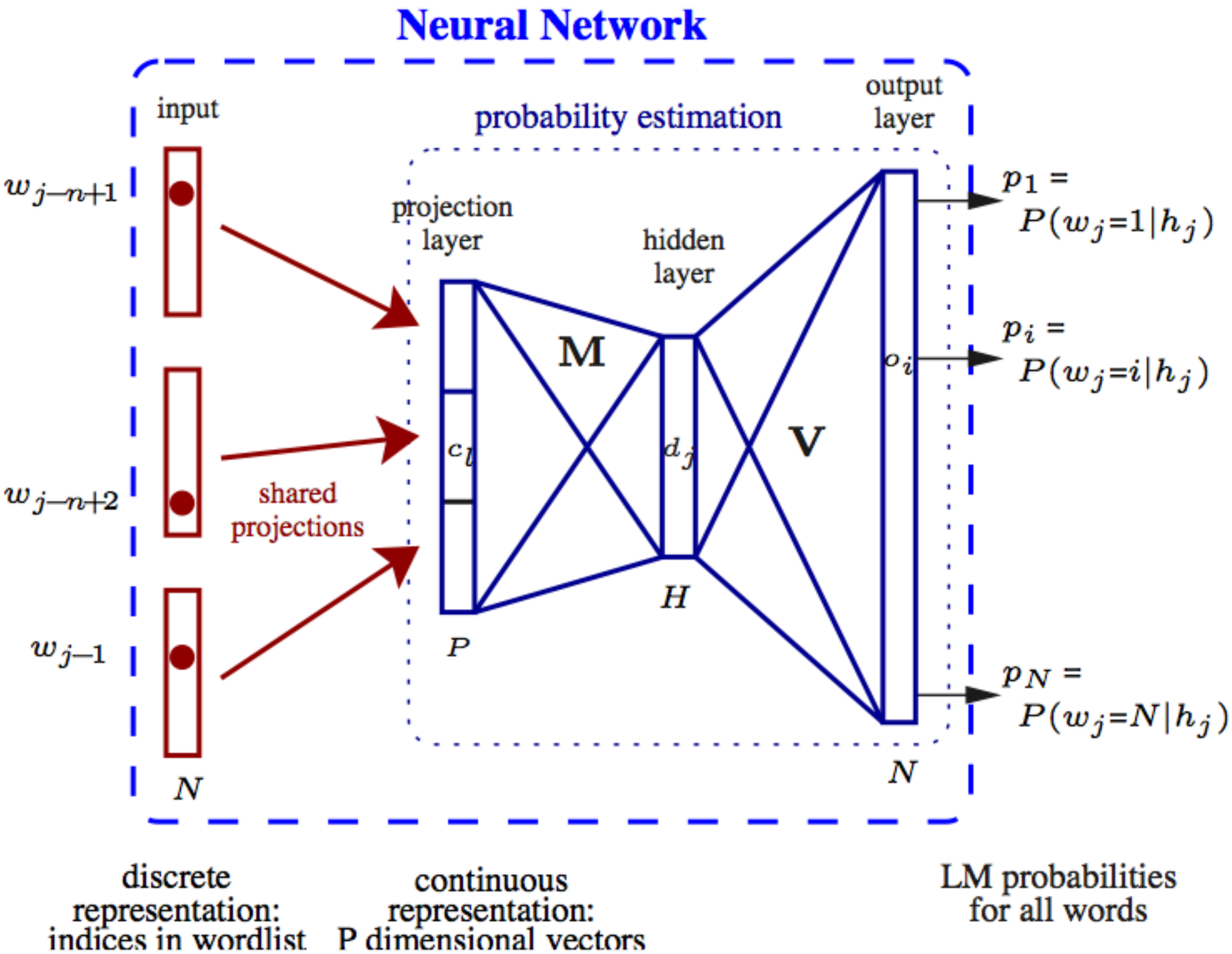
- where the second term is a regularizing Gaussian prior
- ME n-grams are rarely used: perform comparably but at higher computational costs, because of the partition function $Z(h)$.

Neural Network LMs

- Most promising among recent development on n-gram LMs.
- Idea: Map single word into a $|V|$-dimensional vector space
  - Represent n-gram LM as a map between vector spaces
- Solution: Learn map with neural network (NN) architecture
  - one hidden layer compress information (projection)
  - second hidden layer performs the n-gram prediction
  - other architectures are possible: e.g. recurrent NN
- Implementations:
  - Continuous Space Language Model [Schwenk et al., 2006]
  - Recurrent Neural Network Language Modeling Toolkit [6]
- Pros:
  - Improves SMT performance when used jointly with conventional LM
- Cons:
  - Computational cost of training phase (requires GPU)
  - Not easy to integrate into search algorithm (mainly used for re-scoring)

[6] http://rnnlm.sourceforge.net

Neural Network LMs



(From [Schwenk et al., 2006])

Language Modelling Toolkits

- Availability of large scale corpora has pushed research toward using huge LMs
- MT systems set for evaluations use LMs with over a billion of 5-grams
- Estimating accurate large scale LMs is computationally costly
- Querying large LMs can be carried out rather efficiently (with adequate RAM)

Available LM toolkits

- SRILM [Stolcke, 2002]: Moses support, open source (no commercial)
- IRSTLM [Federico et al., 2008]: Moses support, open source
- KENLM [Heafield, 2011]: MKN training, Moses support, open source

Interoperability

- The standard for n-gram LM representation is the so-called ARPA file format.

ARPA File Format

Represents both interpolated and back-off n-gram LMs

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

```
ngram 1= 86700
ngram 2= 1948935
ngram 3= 2070512

\1-grams:
-2.94351     world    -0.51431
-6.09691     friends  -0.15553
-2.88382     !        -2.38764

  ...

\2-grams:
-3.91009     world !        -0.3514
-3.91257     hello world    -0.2412
-3.87582     hello friends  -0.0312

  ...

\3-grams:
-0.00108     hello world  !
-0.00027     hi hello  !
  ...

\end\
```

ARPA File Format

Represents both interpolated and back-off n-gram LMs

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

```
ngram 1= 86700
ngram 2= 1948935
ngram 3= 2070512

\1-grams:
-2.94351      world     -0.51431
-6.09691      friends   -0.15553
-2.88382      !         -2.38764

  ...

\2-grams:
-3.91009      world !         -0.3514
-3.91257      hello world     -0.2412
-3.87582      hello friends -0.0312

  ...

\3-grams:
-0.00108      hello world   !
-0.00027      hi hello   !
  ...

\end\
```

Query: Pr(! / hello friends)?

1. look-up logPr(hello friends !)
   failed! then back-off
2. look-up logBow(hello friends)
   res=-0.0312
3. look-up logPr(friends !)
   failed! then back-off
4. look-up logBow(friends)
   res=res-0.15553
5. look-up logPr(!)
   res=res-2.88382
6. prob=exp(res)=0.04640

References

[Berger et al., 1996] Berger, A., Pietra, S. A. D., and Pietra, V. J. D. (1996). A maximum entropy approach to natural language processing. Computational Linguistics, 22(1):39–71.

[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.

[Heafield, 2011] Heafield, K. (2011). KenLM: faster and smaller language model queries. In Proceedings of the Sixth Workshop on Statistical Machine Translation, pages 187–197, Edinburgh, Scotland, United Kingdom.

[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.

[Rosenfeld, 1996] Rosenfeld, R. (1996). A maximum entropy approach to adaptive statistical language modeling. Computer Speech and Language, 10:187–228.

[Schwenk et al., 2006] Schwenk, H., Dechelotte, D., and Gauvain, J.-L. (2006). Continuous space language models for statistical machine translation. In Proceedings of the COLING/ACL 2006 Main Conference Poster Sessions, pages 723–730, Sydney, Australia. Association for Computational Linguistics.

[Stolcke, 2002] Stolcke, A. (2002). Srilm - an extensible language modeling toolkit. In Proceedings of ICSLP, Denver, Colorado.

[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.