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:

\[ e^* = \arg \max_{e,a} \sum \lambda_i h_i(e, f, a) \]

- Phrases are finite strings (cf. n-grams)
- Hidden variable \( a \) embeds:
  - segmentation of \( f \) and \( e \) into phrases
  - alignment of phrases of \( f \) with phrases of \( e \)
- 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 $w = w_1 \ldots, w_t, \ldots, w_{|w|}$ we can compute its probability by:

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

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

- $\Pr(w_t | h_t)$ 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: $\Pr($Parliament$ | $I$ declare resumed the session of the European$)$

  3-gram: $\Pr($Parliament$ | $the European$)$

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:

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

- w is a sufficiently long test sample and p(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 H T T H T T H for p(T)=0.3, p(H)=0.7 and p(H)=0.3, p(T)=0.7. Comment the results.]
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 $x, y$.

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

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

- 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. $c(x, y) = 0$. 

M. Federico  
Language Models  
MT Marathon, Prague, 2013
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:

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

where $\alpha_{xy}$ is an appropriate normalization term.\(^3\)

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

\[
p(w \mid x \ y) = f^*(w \mid x \ y) + \lambda(x \ y)p(w \mid y).
\]  

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

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

\[ f^*(w) = \frac{c(w)}{N + |V|} \quad \lambda = \frac{|V|}{N + |V|} \]

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

\[
p(w) = \begin{cases} 
  f^*(w) & \text{if } w \in V \\
  \lambda \times \frac{1}{|U| - |V|} & \text{if } w \notin V \quad \text{(i.e. OOV word)}
\end{cases}
\]

Important: use a common value \(|U|\) when comparing/combining different LMs.
Note: IRSTLM permits to set \(|U|\), SRILM uses a fixed \(p(w \notin V)\)
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 \( \lambda(x \ y) \propto \text{number of back-offs} \) (i.e. 3)
  - hence \( f^{*}(w \mid x \ y) \propto \text{relative frequency} \) (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 \( n(x \ y \ *) = |\{w : c(x \ y \ w) > 0\}|. \)

- **Pros**: easy to compute, robust for small or noisy corpora
- **Cons**: underestimates probability of frequent \( n \)-grams

---

\[4\] [Exercise 3. Compute \( f^{*}(u \mid x \ y) \) 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 $\beta$ $(0 < \beta \leq 1)$ from each count
  - estimate $\beta$ via MLE with leaving-one-out on the training data

- **Solution:** (notice: one distinct $\beta$ for each n-gram order)

\[
 f^*(w \mid x y) = \max \left\{ \frac{c(xyw) - \beta}{c(xy)}, 0 \right\}
\]

which gives
\[
 \lambda(xy) = \beta \frac{\sum_{w : c(xyw) > 1}}{c(xy)}
\]

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

- **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, $n_1$, $n_2$, $\beta$, $f^*(w \mid x y)$ and $\lambda(x y)$]
Discounting Methods

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 \( c(yw) \) with \( n(*yw) = \# \) of observed back-offs (=3)

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

\[
f^*(w \mid y) = \max \left\{ \frac{n(*yw) - \beta}{n(*y*)}, 0 \right\}
\]

which gives

\[
\lambda(y) = \beta \frac{\sum_{w:n(*yw)>1} 1}{n(*y*)}
\]

where \( n(*yw) = |\{x : c(xyw) > 0\}| \) and \( n(*y*) = |\{xw : c(xyw) > 0\}| \)

- **Pros**: corrected counts can be used with other smoothing methods too
- **Cons**: LM cannot be used to compute lower order \( n \)-gram probs
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 \textit{aiming at } \_ \_ (Google 1TWeb corpus)
- The trend is similar but MKN outperforms WB

\begin{figure}
\centering
\includegraphics[width=\textwidth]{chart.png}
\caption{Top frequent 3gr 'aiming at \_' in Web1 T5gr}
\end{figure}
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

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

(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 \( w \) we assume a corresponding class sequence \( g \)
  – ambiguous (e.g. POS) or deterministic (word classes)
• Factored LMs can be **integrated into log-linear models** with:
  – a **word-to-class factored model**: \( f \rightarrow e \rightarrow g \) with features:

\[
h_1(f, e, a) , h_2(e, g) , h_3(e) , h_4(g)
\]

  – a **word-class joint model**: \( f \rightarrow (e, g) \) with features

\[
h_1(f, e, g, a) , h_2(e, g) , h_3(e) , h_4(g)
\]

Features of single sequences are log-probs of standard \( n \)-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), \( \lambda_r \) are free parameters
- Features can model any dependency between \( w \) and \( h \).
- Given feature functions and training sample \( w \), parameters can be estimated [Berger et al., 1996] by maximizing the posterior log-likelihood:

\[
\hat{\lambda} = \arg \max_{\lambda \in \mathbb{R}^m} \sum_{t=1}^{\left| w \right|} \log p_\lambda(w_t \mid h_t) + \log q(\lambda)
\]

- 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

• **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)

---

6http://rnnlm.sourceforge.net
Neural Network LMs

(From [Schwenk et al., 2006])

(M. Federico | Language Models | MT Marathon, Prague, 2013)
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**: \( \log(\text{smoothed-prob}) :: \text{n-gram} :: \log(\text{back-off weight}) \)
- **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\n```

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


