Statistical Machine Translation

Marcello Federico
FBK-irst Trento, Italy
Galileo Galilei PhD School – University of Pisa

Pisa, 7-19 May 2008

Part V: Language Modeling

• Comparing ASR and statistical MT
• N-gram LMs
• Perplexity
• Frequency smoothing
• LM representation
• Efficient handling of huge LMs
Fundamental Equation of ASR

Let \( x \) be a sequence of acoustic observations, the most probable transcription \( w^* \) is searched through the following statistical decision criterion:

\[
w^* = \arg \max_w \Pr(x \mid w) \Pr(w)
\]  

(1)

The computational problems of ASR:

- **language modeling**: estimating the language model probability \( \Pr(w) \)
- **acoustic modeling**: estimating the acoustic model probability \( \Pr(x \mid w) \)
- **search** problem: carrying out the optimization criterion (1)

The acoustic model is defined by introducing an hidden alignment variable \( s \):

\[
\Pr(x \mid w) = \sum_s \Pr(x, s \mid w)
\]

corresponding to state sequences of a Markov model generating \( x \) and \( s \).

Fundamental Equation of SMT

Let \( f \) be any text in a Foreign source language. The most probable translation into English is searched among texts \( e \) in the target language by:

\[
e^* = \arg \max_e \Pr(f \mid e) \Pr(e)
\]  

(2)

The computational problems of SMT:

- **language modeling**: estimating the language model probability \( \Pr(e) \)
- **translation modeling**: estimating the translation model probability \( \Pr(f \mid e) \)
- **search** problem: carrying out the optimization criterion (2)

The translation model is defined in terms of an hidden alignment variable \( a \):

\[
\Pr(f \mid e) = \sum_a \Pr(f, a \mid e)
\]

that map source to target positions, and a multinomial process for \( f \) and \( a \).
Parallel data are weakly aligned observations of source and target symbols. In other words, we do not need to observe the hidden variables s or a.

**N-gram LMs**

The purpose of LMs is to compute the probability $\Pr(w_t^T)$ of any sequence of words $w_t^T = w_1 \ldots, w_t, \ldots, w_T$. The probability $\Pr(w_t^T)$ can be expressed as:

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

(3)

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

- $\Pr(w_t | h_t)$ become difficult to estimate as the sequence of words $h_t$ grows.
- We approximate by defining equivalence classes on histories $h_t$.
- $n$-gram approximation let each word depend on the most recent $n - 1$ words:

$$h_t \approx w_{t-n+1} \ldots w_{t-1}.$$  

(4)
Normalization Requirement

\[ \sum_{T=1}^{\infty} \Pr(T) \sum_{w_1 \ldots w_T} \Pr(w_1, \ldots, w_T | T) = 1 \]

N-gram LMs guarantee that probabilities sum up over one, for a given length \( T \):

\[
\sum_{w_1 \ldots w_T} \prod_{t=1}^{T} \Pr(w_t | h_t) = \sum_{w_1} \Pr(w_1) \sum_{w_2} \Pr(w_2 | h_1) \ldots \sum_{w_{T-1}} \Pr(w_{T-1} | h_{T-1}) \sum_{w_T} \Pr(w_T | h_T) = 1 \\
= \sum_{w_1} \Pr(w_1) \sum_{w_2} \Pr(w_2 | h_1) \ldots \sum_{w_{T-1}} \Pr(w_{T-1} | h_{T-1}) \cdot 1 = 1 \\
= \ldots \\
= \sum_{w_1} \Pr(w_1) \cdot 1 \ldots \cdot 1 \cdot 1 = 1 \\
= 1 \\
\]

(5)

String Length Model

Hence we just need a length model \( P(T) \), some examples are:

- **Exponential model**: \( p(T) = (a - 1)a^{-T} \) with any \( a > 1 \)
  - Bad: favors short strings, which are already rewarded by the n-gram product
- **Uniform model**: \( p(T) = 1/K(|f|) \) for lengths up to \( K(|f|) \) and 0 otherwise
  - Good: for a given input \( f \), \( K \) is constant and can be disregarded
  - Shorter sentences are anyway favoured
- **Word Insertion model** is also added to reward any added word
  - feature function: \( h(e, f, a) = \exp(|e|) \)
How to measure LM quality

LMs for ASR are evaluated with respect to their

• Impact on recognition accuracy = Word Error Rate
• Capability of predicting words in a text = Perplexity

The perplexity measure (PP) is defined as follows:

\[ PP = 2^{LP} \quad \text{where} \quad LP = -\frac{1}{M} \log_2 \hat{P}(w^M_1) \]  \hspace{1cm} (6)

• \( w^M_1 = w_1 \ldots w_M \) is a sufficiently long test sample
• \( \hat{P}(w^M_1) \) is the probability of \( w^M_1 \) computed with a given a stochastic LM.

According to basic Information Theory, perplexity indicates that the prediction task of the LM is as difficult as guessing a word among \( PP \) equally likely words.

Example: guessing random digits has \( PP = 10 \).
Frequency Discounting

*Discount* relative frequency to assign some positive prob to every possible $n$-gram

$$0 \leq f^*(w | x y) \leq f(w | x y) \quad \forall x y w \in V^3$$

The *zero-frequency probability* $\lambda(x y)$, defined by:

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

is *redistributed* over the set of words never observed after history $x y$.

Redistribution is proportional to the less specific $n-1$-gram model $p(w | y)$.

---

1 Notice: $c(x, y) = 0$ implies that $\lambda(x y) = 1$.

---

Smoothing Schemes

Discounting of $f(w | x y)$ and redistribution of $\lambda(x y)$ can be combined by:

- **Back-off**, i.e. select the most significant approximation available:

  $$p(w | x y) = \begin{cases} f^*(w | x y) & \text{if } f^*(w | x y) > 0 \\ \alpha_{x y} \lambda(x y)p(w | y) & \text{otherwise} \end{cases} \quad (7)$$

  where $\alpha_{x y}$ is an appropriate *normalization term*

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

  $$p(w | x y) = f^*(w | x y) + \lambda(x y)p(w | y). \quad (8)$$
**Smoothing Methods**

- **Witten-Bell estimate** [Witten & Bell, 1991]
  
  \[ \lambda(xy) \propto n(xy) \text{ i.e. } \# \text{ different words observed after } xy \text{ in the training data:} \]

  \[
  \lambda(xy) = \text{def} \frac{n(xy)}{c(xy) + n(xy)} \quad \text{which gives:} \quad f^*(w \mid xy) = \frac{c(xyw)}{c(xy) + n(xy)}
  \]

- **Absolute discounting** [Ney & Essen, 1991]
  
  subtract constant \( \beta \ (0 < \beta \leq 1) \) from all observed \( n \)-gram counts

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

\[ \beta \approx \frac{n_{1}}{n_{1} + 2n_{2}} < 1 \text{ where } n_{c} \text{ is } \# \text{ of different } n \text{-grams which occur } c \text{ times in the training data.} \]

**Improved Absolute Discounting**

- **Kneser-Ney smoothing** [Kneser & Ney, 1995]
  
  Absolute discounting with **corrected counts for lower order** \( n \)-grams. Rationale: the lower order count \( c(y, w) \) is made proportional to the number of different words \( xy \) follows.

  **Example**: let \( c(\text{los,angeles}) = 1000 \) and \( c(\text{angeles}) = 1000 \) ⟷ corrected count is \( c'(\text{angeles}) = 1 \), hence the unigram prob \( p(\text{angeles}) \) will be small.

- **Improved Kneser-Ney** [Chen & Goodman, 1998]
  
  In addition use **specific discounting coefficients** for rare \( n \)-grams:

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

  where \( \beta(0) = 0, \beta(1) = D_{1}, \beta(2) = D_{2}, \beta(c) = D_{3+} \) if \( c \geq 3 \).
LM representation: ARPA File Format

Contains all the ingredients needed to compute LM probabilities:

\data\ngram 1= 86700
ngram 2= 1948935
ngram 3= 2070512
\1-grams:
-2.88382  !  -2.38764
-2.94351  world -0.514311
-6.09691  pisa  -0.15553
...
\2-grams:
-3.91009  world !  -0.351469
-3.91257  hello world -0.24
-3.87582  hello pisa  -0.0312
...
\3-grams:
-0.00108858  hello world !
-0.000271867  hi hello !
...
\end\n
logPr(!| hello pisa) = -0.0312 + logPr(!| pisa)
logPr(!| pisa) = -0.15553 - 2.88382

Large Scale Language Models

• Availability of large scale corpora has pushed research toward using huge LMs
• At 2006 NIST WS best systems used LMs trained on at least 1.6G words
• Google presented results using a 5-gram LM trained on 1.3T words
• Handling of such huge LMs with available tools (e.g. SRILM) is prohibitive if you use standard computer equipment (4 to to 8Gb of RAM)
• Trend of technology is towards distributed processing using PC farms

We developed IRSTLM, a LM library addressing these needs
• open-source LGPL library under sourceforge.net
• integrated into the Moses SMT Toolkit and FBK-irst’s speech decoder
IRSTLM library (open source)

Important Features

• Distributed training on single machine or SGE queue
  – split dictionary into balanced $n$-gram prefix lists
  – collect $n$-grams for each prefix lists
  – estimate single LMs for each prefix list
  – quickly merge single LMs into one ARPA file

• Space optimization
  – $n$-gram collection uses dynamic storage to encode counters
  – LM estimation just requires reading disk files
  – probs and back-off weights are quantized
  – LM data structure is loaded on demand

• LM caching
  – computations of probs, access to internal lists, LM states, ....

Data Structure to Collect N-grams

• Dynamic prefix-tree data structure
• Successor lists are allocated on demand through memory pools
• Storage of counts from 1 to 6 bytes, according to max value
• Permits to manage few huge counts, such as in the google $n$-grams
Distributed Training on English Gigaword

<table>
<thead>
<tr>
<th>list index</th>
<th>dictionary size</th>
<th>number of 5-grams: observed distinct non-singletons</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>4</td>
<td>217M 44.9M 16.2M</td>
</tr>
<tr>
<td>1</td>
<td>11</td>
<td>164M 65.4M 20.7M</td>
</tr>
<tr>
<td>2</td>
<td>8</td>
<td>208M 85.1M 27.0M</td>
</tr>
<tr>
<td>3</td>
<td>44</td>
<td>191M 83.0M 26.0M</td>
</tr>
<tr>
<td>4</td>
<td>64</td>
<td>143M 56.6M 17.8M</td>
</tr>
<tr>
<td>5</td>
<td>137</td>
<td>142M 62.3M 19.1M</td>
</tr>
<tr>
<td>6</td>
<td>190</td>
<td>142M 64.0M 19.5M</td>
</tr>
<tr>
<td>7</td>
<td>548</td>
<td>142M 66.0M 20.1M</td>
</tr>
<tr>
<td>8</td>
<td>783</td>
<td>142M 63.3M 19.2M</td>
</tr>
<tr>
<td>9</td>
<td>1.3K</td>
<td>141M 67.4M 20.2M</td>
</tr>
<tr>
<td>10</td>
<td>2.5K</td>
<td>141M 69.7M 20.5M</td>
</tr>
<tr>
<td>11</td>
<td>6.1K</td>
<td>141M 71.8M 20.8M</td>
</tr>
<tr>
<td>12</td>
<td>25.4K</td>
<td>141M 74.5M 20.9M</td>
</tr>
<tr>
<td>13</td>
<td>4.51M</td>
<td>141M 77.4M 20.6M</td>
</tr>
<tr>
<td>total</td>
<td>4.55M</td>
<td>2.2G 951M 289M</td>
</tr>
</tbody>
</table>

Data Structure to Compute LM Probs

- First used in *CMU-Cambridge LM Toolkit* (Clarkson and Rosenfeld, 1997)
- Slower access but less memory than structure used by *SRILM Toolkit*
- *IRSTLM* in addition compresses probabilities and back-off weights into 1 byte!
Compression Through Quantization

How does quantization work?
1. Partition observed probabilities into regions (clusters)
2. Assign a code and probability value to each region (codebook)
3. Encode the probabilities of all observations (quantization)

We investigate two quantization methods:

- **Lloyd’s K-Means Algorithm**
  - first applied to LM for ASR by [Whittaker & Raj, 2000]
  - computes clusters minimizing average distance between data and centroids

- **Binning Algorithm**
  - first applied to term-frequencies for IR by [Franz & McCarley, 2002]
  - computes clusters that partition data into uniformly populated intervals

Notice: a codebook of $n$ centers means a quantization level of $\log_2 n$ bits.

LM Quantization

- **Codebooks**
  - One codebook for each word and back-off probability level
  - For instance, a 5-gram LM needs in total 9 codebooks.
  - Use codebook of at least 256 entries for 1-gram distributions.

- **Motivation**
  - Distributions of these probabilities can be quite different.
  - 1-gram distributions contain relatively few probabilities
  - Memory cost of a few codebooks is irrelevant.

- **Composition of codebooks**
  - LM probs are computed by multiplying entries of different codebooks
  - actual resolution of lower order $n$-grams is higher than that of its codebook!

Very little loss in performance with 8 bit quantization [Federico & Bertoldi ’06]
SMT Decode calls to a 3-gram LM while decoding from German to English the text:
ich bin kein christdemokrat und glaube daher nicht an wunder. doch ich möchte dem europäischen parlament, so wie es gegenwärtig beschaffen ist, für seinen grossen beitrag zu diesen arbeiten danken.

- 1.7M calls only involving 120K different 3-grams
- Decoder tends to access LM n-grams in nonuniform, *highly localized patterns*
- First call of an n-gram is easily followed by other calls of the same n-gram.
Memory Mapping of LM on Disk

- Our LM structure permits to exploit so-called *memory mapped* file access.
- Memory mapping permits to include a file in the address space of a process, whose access is managed as virtual memory
- Only memory pages (grey blocks) that are accessed by decoding are loaded