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 $\mathbf{x}$ be a sequence of acoustic observations, the most probable transcription $\mathbf{w}^*$ is searched through the following statistical decision criterion:

$$\mathbf{w}^* = \arg \max_{\mathbf{w}} \Pr(\mathbf{x} \mid \mathbf{w}) \Pr(\mathbf{w}) \tag{1}$$

The **computational problems of ASR**:

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

The acoustic model is defined by introducing an hidden alignment variable $\mathbf{s}$:

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

corresponding to state sequences of a Markov model generating $\mathbf{x}$ and $\mathbf{s}$.

Fundamental Equation of SMT

Let $\mathbf{f}$ be any text in a Foreign *source* language. The most probable translation into English is searched among texts $\mathbf{e}$ in the *target* language by:

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

The **computational problems of SMT**:

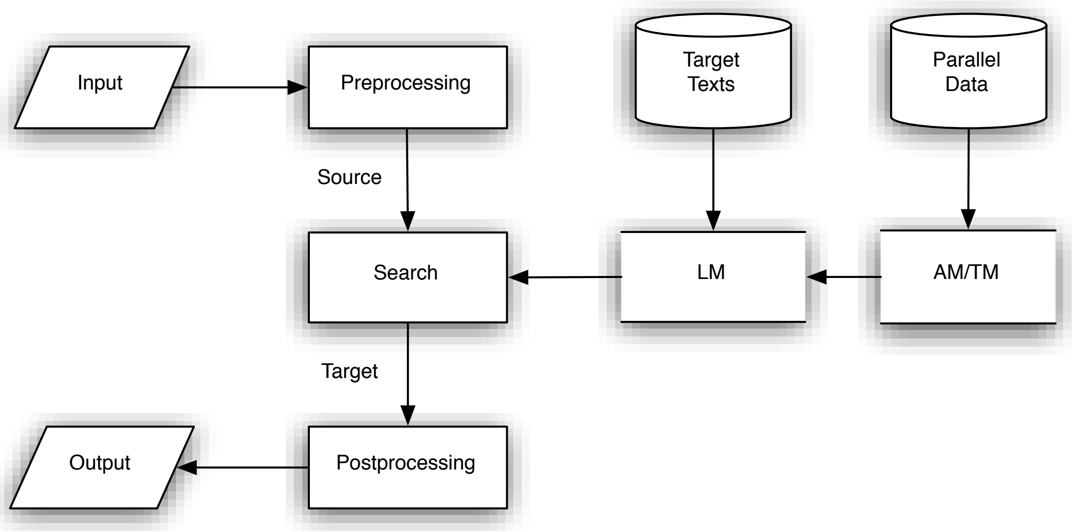
- *language modeling*: estimating the *language model probability* $\Pr(\mathbf{e})$
- *translation modeling*: estimating the *translation model probability* $\Pr(\mathbf{f} \mid \mathbf{e})$
- *search* problem: carrying out the optimization criterion (2)

The translation model is defined in terms of an hidden alignment variable $\mathbf{a}$:

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

that map source to target positions, and a multinomial process for $\mathbf{f}$ and $\mathbf{a}$.

ASR/MT Architecture



Parallel data are *weakly aligned* observations of source and target symbols. In other words, we do not need to observe the hidden variables $\mathbf{s}$ or $\mathbf{a}$.

N-gram LMs

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

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

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

- $\Pr(w_t \mid h_t)$ 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:

$$h_t \approx w_{t-n+1} \ldots w_{t-1}. \tag{4}$$

Normalization Requirement

$$\sum_{T=1}^{\infty} \Pr(T) \sum_{w_1 \dots w_T} \Pr(w_1, \dots, w_T \mid T) = 1$$

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

$$\begin{aligned}
\sum_{w_1 \dots w_T} \prod_{t=1}^{T} \Pr(w_t \mid h_t) &= \sum_{w_1} \Pr(w_1) \sum_{w_2} \Pr(w_2 \mid h_1) \dots \sum_{w_{T-1}} \Pr(w_{T-1} \mid h_{T-1}) \underbrace{\sum_{w_T} \Pr(w_T \mid h_T)}_{=1} \\
&= \sum_{w_1} \Pr(w_1) \sum_{w_2} \Pr(w_2 \mid h_1) \dots \underbrace{\sum_{w_{T-1}} \Pr(w_{T-1} \mid h_{T-1})}_{=1} \cdot 1 \\
&= \dots \\
&= \underbrace{\sum_{w_1} \Pr(w_1)}_{=1} \cdot 1 \dots \cdot 1 \cdot 1 = 1 \qquad (5)
\end{aligned}$$

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**: $p(T) = 1/K(|\mathbf{f}|)$ for lengths up to $K(|\mathbf{f}|)$ and 0 otherwise
  - Good: for a given input $\mathbf{f}$, 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: $h(\mathbf{e}, \mathbf{f}, \mathbf{a}) = \exp(|\mathbf{e}|)$

FONDAZIONE BRUNO KESSLER

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:

$$PP = 2^{LP} \quad \text{where} \quad LP = -\frac{1}{M}\log_2 \hat{P}(w_1^M) \tag{6}$$

- $w_1^M = w_1 \dots w_M$ is a sufficiently long test sample
- $\hat{\Pr}(w_1^M)$ is the probability of w_1^M 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.

FONDAZIONE BRUNO KESSLER

N-gram LM and data sparseness

Even estimating n-gram probabilities is not a trivial task:

- **high number of parameters**: e.g. a 3-gram LM with a vocabulary of 1,000 words requires, in principle, to estimate 10^9 probabilities!
- **data sparseness** of real texts: i.e. most of correct n-grams are *rare events*

Test samples will contain many *never observed* n-grams: assigning one of them probability zero will rise PP to infinite!

Experimentally, in the 1.2Mw (million word) Lancaster-Oslo-Bergen corpus:

- more than 20% of bigrams and 60% of trigrams occur only once
- 85% of trigrams occur less than five times.
- expected chances of finding new 2-grams is 22%
- expected change of finding new 3-grams is 65%

We need frequency smoothing or discounting!

M. Federico, FBK-irst SMT - Part V Pisa, 7-19 May 2008

Frequency Discounting

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

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

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

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

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

Redistribution is proportional to the less specific $n-1$-gram model $p(w \mid y)$.[1]

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

Smoothing Schemes

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

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

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

  where α_{xy} is an appropriate *normalization term*

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

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

Smoothing Methods

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

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

- **Absolute discounting** [Ney & Essen, 1991]
  subtract constant β $(0 < \beta \leq 1)$ from all observed n-gram counts[2]

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

[2] $\beta \approx \frac{n_1}{n_1 + 2n_2} < 1$ where n_c is # of different n-grams which occurr c 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 $c(\texttt{los}, \texttt{angeles}) = 1000$ and $c(\texttt{angeles}) = 1000$ $\longrightarrow$ corrected count is $c'(\texttt{angeles}) = 1$, hence the unigram prob $p(\texttt{angeles})$ will be small.

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

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

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\
```

```
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

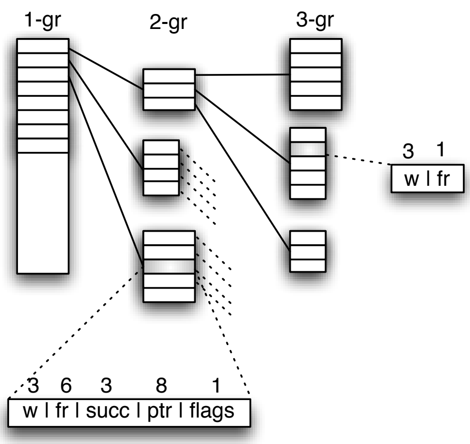
- open-source LGPL library under sourceforge.net
- integrated into the Moses SMT Toolkit and FBK-irst's speech decoder

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

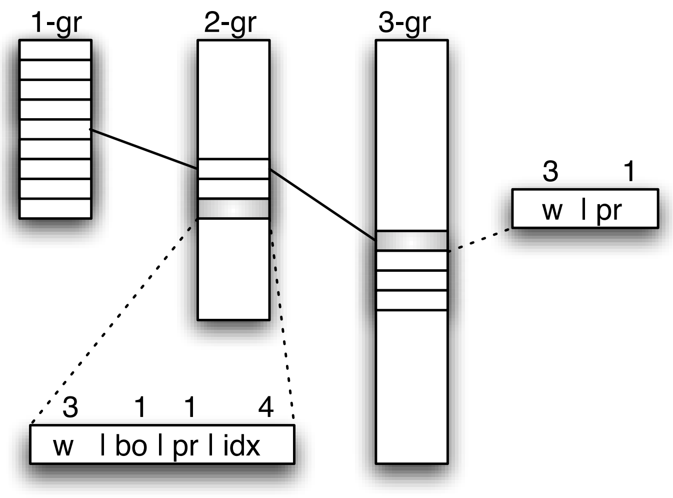


- Dynamic prefix-tree data structure
- Successor lists are allocated on demand through memory pools
- Storage of counts from 1 to 6 bytes, according to max value
- Permits to manage few huge counts, such as in the google n-grams

Distributed Training on English Gigaword

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

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

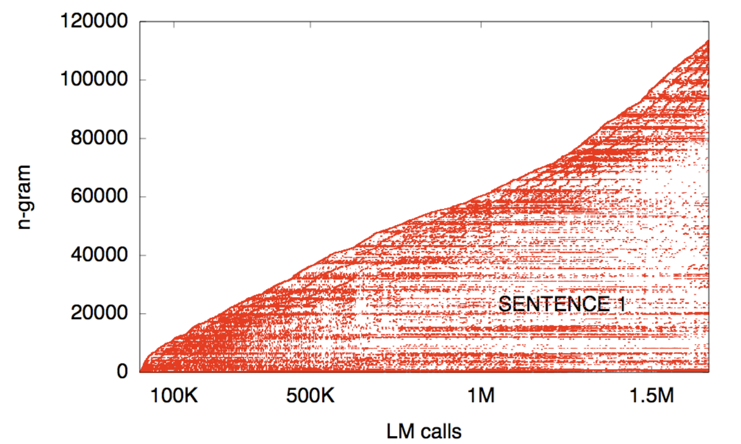
Notice: a codebook of n centers means a *quantization level* of $\log_2 n$ bits.

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]

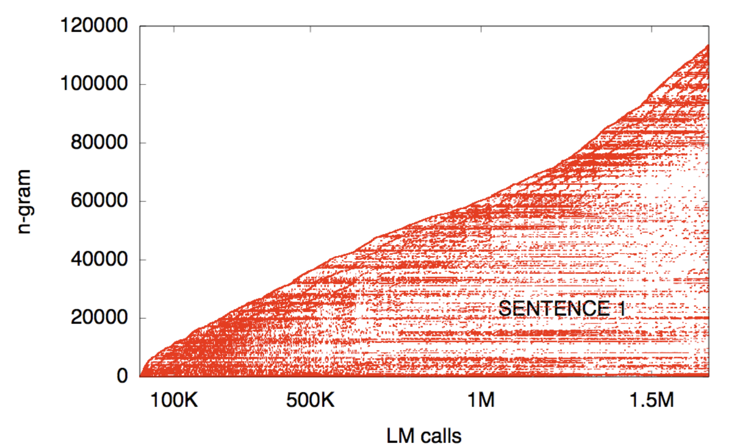
LM Accesses by Search Algorithm



SMT Decode calls to a 3-gram LM while decoding from German to English the text:

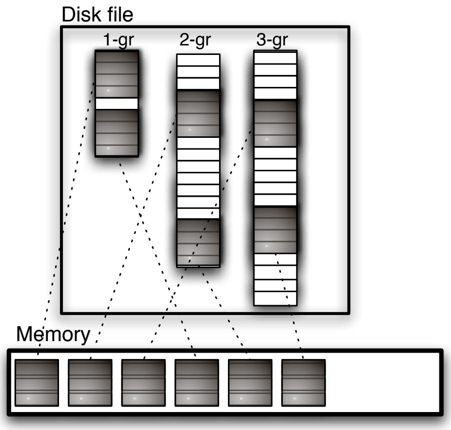
`ich bin kein christdemokrat und glaube daher nicht an wunder . doch ich möchte dem europäischen parlament , so wie es gegenwürtig beschaffen ist , für seinen grossen beitrag zu diesen arbeiten danken.`

LM Accesses by SMT Search Algorithm



- 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