# Efficient Estimation of Maximum Entropy Language Models with $N$-gram features: SRILM extension

Tanel Alumäe[1], Mikko Kurimo[2]
[1]Institute of Cybernetics, Tallinn University of Technology, Estonia
[2]Adaptive Informatics Research Centre, School of Science and Technology, Aalto University, Helsinki

## Summary

We implemented an extension to the **SRILM** toolkit for training maximum entropy language models with $N$-gram features. Functionality:

- fast estimation
- allows using moderately large training data (building a trigram MaxEnt LM on 500M words takes 20GB RAM and 1 hour training time)
- can be parallelized over multiple cores
- patch available for full SRILM integration (BSD license):
  `http://www.phon.ioc.ee/~tanela/srilm-me`

## Maximum Entropy LM

A conditional MaxEnt model has the following form:

$$P(x|h) = \frac{e^{\sum_i \lambda_i f_i(w,h)}}{Z(h)}$$

where

- $x$ is a word
- $h$ is a context (the word history)
- $Z(h)$ is a normalization factor:

$$Z(h) = \sum_{x_i \in V} e^{\sum_j \lambda_j f_j(x_i,h)}$$

- $f_i$ are (typically binary) feature functions (in our case, $N$-gram features)
- feature weights $\lambda$ are learned via gradient descent
- log conditional likelihood of the data $\mathscr{L}(X;\Lambda)$ is maximized
- smoothing with $\ell_1$ and $\ell_2^2$ prior:

$$\mathscr{L}_{\ell_1+\ell_2^2}(X;\Lambda) = \mathscr{L}(X;\Lambda) - \frac{\alpha}{D}\sum_i^F |\lambda_i| - \frac{1}{2\sigma^2 D}\sum_{i=1}^F \lambda_i^2$$

- where $D$ is the size of training data
- smoothing encourages feature weights with small absolute values

## Motivation

Maximum entropy models are regarded as one of the most promising language modeling techniques.
There are tens of open source toolkits for estimating MaxEnt models with arbitrary features. However, when the size of the output vocabulary in a MaxEnt model is large (which is the case with language models), naive training becomes extremely resource consuming, requiring huge amounts of memory and days or weeks to finish, even with small and medium sized training data.

## Implementation

To speed up training, we take advantage of the fact that $N$-gram features are:

- hierarchical (i.e., if a trigram feature is active, the corresponding bigram feature is also active)
- not overlapping (i.e., only one trigram feature is active at the same time for a given $w|h$).

This allows efficient computation of normalization factors (Wu and Khudanpur, 2002):

$$Z(w_{i-1}, w_{i-2}) =$$
$$\sum_{w_i \in V} e^{f_{w_i}} + \quad (1)$$
$$\sum_{w_i \in V_{w_{i-1}}} (e^{f_{w_{i-1}w_i}} - 1)e^{f_{w_i}} + \quad (2)$$
$$\sum_{w_i \in V_{w_{i-2}w_{i-1}}} (e^{f_{w_{i-2}w_{i-1}w_i}} - 1)e^{f_{w_{i-1}w_i}} \quad (3)$$

In this sum:

(1) is shared by all contexts and can be precomputed;
(2) is shared by all contexts ending with the same word, can be precomputed for each word;
(3) requires only summing over words that occur after the given context.

Additional functionality:

- supports "naive domain adaptation": use the weights learned from an "out of domain" corpus as the prior means for the "in domain" data
- partly parallelized to use multiple CPU cores

## Accuracy

The implementation was tested on two speech recognition tasks:

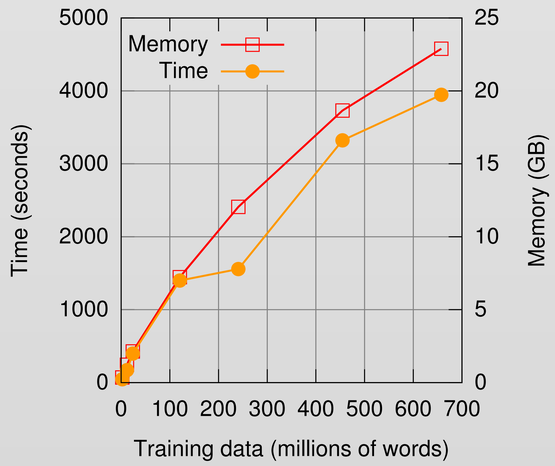
- **English Broadcast News** (2003 NIST Rich Transcription Evaluation Data). For training LM, we used 5M sentences from the Gigaword (2nd ed.) corpus (99.5M words, out-of-domain), and broadcast news transcriptions from the TDT4 corpus (1.19M words, in-domain).
- **Estonian Broadcast Conversations** (40 minutes of live talk programs from Estonian radio). We trained a morpheme-based LM from two sources: about 10M sentences from newspapers (185M morphemes), and transcriptions of 10 hours of radio live talk programs (104K morphemes).

We built Kneser-Ney smoothed trigram models and MaxEnt models with trigram features and used them to evaluate test set perplexity and to rescore $N$-best lists. Results:

| | | Perplexity | | WER | |
|---|---|---|---|---|---|
| Task | Model training | Trigram | Maxent | Trigram | Maxent |
| English | Out-of-domain | 301 | **297** | | |
| | In-domain | 315 | **309** | | |
| | Interpolated | **218** | 219 | **25.7** | 26.0 |
| | Adapted | N/A | **217** | | **25.5** |
| | Adapted ME + Interpolated trigram | **206** | | **25.5** | |
| Estonian | Out-of-domain | 245 | **239** | | |
| | In-domain | **446** | 458 | | |
| | Interpolated | **184** | 187 | 39.4 | **39.2** |
| | Adapted | N/A | 177 | | **38.0** |
| | Adapted ME + Interpolated trigram | **167** | | **37.5** | |

## Training time and memory usage

Time and memory consumption during training with all trigram features, 40K vocabulary




## Parallel training

Training time *vs* number of CPU cores used.