Sinuhe — Statistical Machine Translation with a Globally Trained Conditional Exponential Family Translation Model

Matti Kääriäinen

matti.kaariainen@cs.helsinki.fi

Barcelona, 13 May, 2009.

Talk outline

Machine translation by machine learning:

- Theory:
  - Models
  - Training
  - Prediction
- Practice:
  - The `Sinuhe` machine translation system
  - Experimental results

Part 0: Background – machine learning framework

General framework

Learning to predict:

- Data: examples $(x, y) \in \mathcal{X} \times \mathcal{Y}$
- Task: learn $f\colon \mathcal{X} \rightarrow \mathcal{Y}$
- Goal: $f(x)$ close to y on future examples (x, y)

Structured prediction is a special case:

- Labels $y \in \mathcal{Y}$ have internal structure (e.g., sequence, matching, partition of a set, . . .)
- The problem does not fully decompose over the parts of y

Examples: Sequence labeling, image segmentation, *machine translation*

A structured prediction framework

General linear setting:

- Map (x, y) into features with a *joint feature map* $\phi\colon \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$
- Learn weight vector $w \in \mathbb{R}^d$
- Predict $f_w(x) = \arg\max_{y \in \mathcal{Y}_x} w \cdot \phi(x, y)$, where $\mathcal{Y}_x \subset \mathcal{Y}$ is the set of feasible labels for x.

Binary classification is a special case:

- $\mathcal{Y} = \{\pm 1\}$
- $\phi(x, y) = y\phi(x)$.

Moving parts

Modelling:

- How to define the joint feature map?
- What criteria to use in learning the weight vector $w \in \mathbb{R}^d$?

Computational:

- Algorithms for learning $w \in \mathbb{R}^d$
- Algorithms for predicting $f_w(x) = \arg\max_{y \in \mathcal{Y}_x} w \cdot \phi(x, y) \in \mathcal{Y}$

Part 1: Theory — Models, training, and prediction for machine translation

Machine translation

Special case of structured prediction, where

$$\mathcal{X} = \text{French text}, \mathcal{Y} = \text{English text}$$

To be defined:

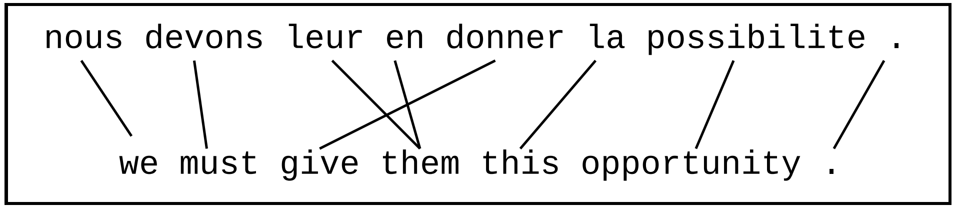
- Joint feature map
- Criterion for learning w
- Algorithms for finding the optimal w
- Algorithms for producing translations $f_w(x)$

Pipeline for extracting biphrase features

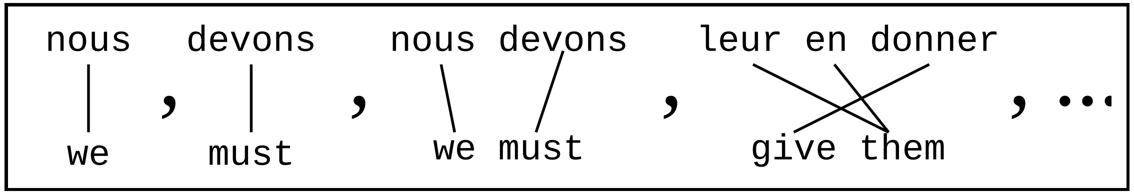
1. Raw data: corpus of sentence pairs $(x, y) \in S_{\text{raw}}$:

```
nous devons leur en donner la possibilite .
we must give them this opportunity .
```

2. Word-alignment: map (x, y) to $(x, a, y) \in S$:



3. Biphrase extraction: extract all compatible biphrases (x', a', y'):



Intuition

Motivating goal:

- Given source sentence x, predict the set of biphrases extracted from it.

Joint feature map

Represent an aligned sentence pair (x, a, y) by the (extracted) biphrases that occur in it:

- $\phi(x, a, y)_{(x',a',y'),i} = 1$ iff the biphrase (x', a', y') occurs at source position i in (x, a, y)
- Projected down features:

$$\tilde{\phi}(x, a, y)_{(x',a',y')} = \sum_i \phi(x, a, y)_{(x',a',y'),i}$$

  The joint feature map is $(x, a, y) \mapsto \tilde{\phi}(x, a, y)$

- Thus: one parameter $w_{(x',a',y')}$ per biphrase feature (x', a', y')

Phrase table pruning: use only biphrases that occur more than once in the training data (leave-one-out motivation)

The translation model

Define:

$$P(\phi(x,a,y)|x) = \frac{\exp(w \cdot \tilde{\phi}(x,a,y))}{\sum_{\phi \in \Phi_x} \exp(w \cdot \tilde{\phi})},$$

where Φ_x is the set of feasible feature vectors for x.

- Proper conditional probability model for (features of) translations
- Φ_x — the feature space equivalent of $\mathcal{Y}_x$ — contains all feature vectors representable by translations (x, a, y) (plus some)
- No reachability problems: the (feature representation of) the training data has non-zero probability!

Criteria for learning w

Two natural probabilistic criteria:

- Maximum likelihood (ML): maximize $\prod_{(x,a,y)\in S} P(\phi(x,a,y)|x)$
  - Overfitting?
- Maximum a posteriori (MAP): maximize

$$P(w|S) \propto \prod_{(x,a,y)\in S} P(\phi(x,a,y)|x,w) \times P(w),$$

  where $P(w)$ is a prior on the parameters
  - Control overfitting by a proper choice of $P(w)$

Surprisingly, ML and MAP (with L1 or L2 regularization) seem to give similar translation quality.

Learning w

For Gaussian priors, MAP parameters can be found by minimizing

$$\mathcal{L}(w) = \sum_i \frac{w_i^2}{2\sigma_i^2} - \sum_{(x,a,y)\in S} \log P(\phi(x,a,y)|x) + C$$

The optimization problem is strictly convex, and can be solved by stochastic gradient:

- Gradients computed by dynamic programming
- The sparsity of $\tilde{\phi}(x,a,y)$ leads to sparse updates, regularization can be done lazily
- Easy to parallelize: apply many stochastic gradient updates asynchronously in parallel

Predicting translations

- Vanilla version:
  1. Solve $g_w(x) = \arg\max_{\phi \in \Phi_x} P(\phi|x)$
  2. Reconstruct $y = f_w(x)$ from $g_w(x)$

  Potential problems: No language model, no reordering model

- Alternative version:
  – Augment $\log P(\phi|x)$ with other features (language model $\log P(y)$, lexical translation features, reordering model, . . .)
  – Find y by optimizing a weighted combination of the features
    * beam search
    * combination weights tuned on development data

The former is conceptually clean and fast, but the latter produces more fluent translations.

Recap: MT system on one slide

1. **Features:** biphrases from phrase-based SMT:
   (a) Primary features $(\phi(x,a,y))_{(x',a',y'),i} =$
   1 iff (x',a',y') occurs in (x,a,y) at position i
   (b) Projected down features $\tilde{\phi}_{(x',a',y')} = \sum_i \phi_{(x',a',y'),i}$
2. **Model:** conditional exponential probability distribution:
   $$P(\phi(x,a,y)|x) = \frac{\exp(w \cdot \tilde{\phi}(x,a,y))}{\sum_{\phi \in \Phi_x} \exp(w \cdot \tilde{\phi})},$$
   where Φ_x is the set of feasible feature vectors for x.
3. **Training:** find MAP parameters, scaled Gaussian prior
4. **Prediction (without an LM):**
   (a) $\hat{\phi}(x) = \arg\max_{\phi(x,a,y)\,:\,x \text{ covered}} P(\phi(x,a,y)|x)$
   (b) $f_w(x) =$ some y reconstructed from $\hat{\phi}(x)$

Part 2: Practice — implementation and experiments

Sinuhe — a prototype MT system

- Released under GPLv3 (current version v1.3beta2)
- Written in C++, about 12000 lines of code (+some scripts)
- Distributed training and prediction:
  - Queries and updates to components of a shared w managed by a server
  - Multiple train and predict clients, communication over TCP
- Scales to large data:
  - GigaFrEn corpus with $22 \cdot 10^6$ sentence pairs crawled from the web, 10^9 words, $w \in \mathbb{R}^{10^8}$
  - Parallel training using ≈ 200 CPU cores converges in a week
- Fast, relatively small memory footprint, good (?) translation quality

Experimental results

- Comparison point: fully tuned `Moses`, no phrase table pruning
- BLEU scores for Europarl data (≈1M sentence pairs for training, 2000 sentence test set):

	es-en	en-es	fr -en	en-fr	de-en	en-de	time (s)
`Sinuhe`	31.38	30.94	31.50	28.91	25.03	19.26	338.0
`Moses`	32.18	31.88	32.63	29.92	27.30	20.57	3729.5
$\texttt{Sinuhe}_{\texttt{trans}}$	29.14	27.12	28.74	26.06	22.38	17.14	44.2
$\texttt{Moses}_{\texttt{trans}}$	24.32	22.75	23.84	21.22	19.62	13.59	1321.5

- BLEU scores for GigaFrEn data (fr-en, WMT09 test set):
  - `Sinuhe`: 26.32
  - `Moses`: 26.98

Experiments with pruned phrase table

Last week results (by Esther Galbrun):

Europarl fr-en data	Sinuhe	Moses$_{\text{pruned}}$	Moses
BLEU score	30.84	30.90	33.05
translation model size (gzipped)	42.6 MB	44.1 MB	1.1 GB
translation time	5 min	47 min	94 min

- For Sinuhe, using the full phrase table seems to help with morphologically rich languages, but not with Spanish to English
- The effects of pruning and regularization still not completely understood

Conclusions

- `Sinuhe` demonstrates feasibility of MT by ML:
  - Faster, smaller memory requirements
  - BLEU scores only slightly behind state-of-the-art
  - Better statistical foundations
- Marketing:
  - `Sinuhe`:
    - `http://www.cs.helsinki.fi/u/mtkaaria/sinuhe`
  - Wikipedia demo:
    - `http://cosco-demo.hiit.fi/smart`