

# Sinuhe — Statistical Machine Translation using a Conditional Exponential Family Translation Model

Matti Kääriäinen

`matti.kaariainen@cs.helsinki.fi`

EMNLP, Singapore, 7 Aug, 2009.



# Why *Sinuhe*?

- Mika Waltari: *Sinuhe egyptiläinen* (*Sinuhe the Egyptian*), 1945
- Perfect name for a translation system:
  - Book situated in ancient Egypt
  - Translated to 40+ languages
  - Written by a Finn
  - In 2008, voted for the most beloved book in Finland
  - Inspired by a story from ancient Egyptian literature

# Talk outline

New machine learning inspired phrase-based translation model:

- Theory:
  - Translation model
  - Training the model
  - Predicting translations
- Practice:
  - The `Sinuhe` machine translation system
  - Experimental results

# Machine learning background

- Basic problem:
  - Given  $x \in \mathcal{X}$ , predict  $y \in \mathcal{Y}$

Machine learning solution:

- Learn predictor  $\mathcal{X} \rightarrow \mathcal{Y}$  from a set of examples  $(x, y)$
- Structured prediction:
  - $y$  has structure (graph, sequence, ...)

This talk: Use tools from structured prediction to attack MT

# A framework for structured prediction

- Map  $(x, y)$  into features with a *joint feature map*  $\phi: \mathcal{X} \times \mathcal{Y} \rightarrow \mathbb{R}^d$
- 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 as 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  $w \in \mathbb{R}^d$ ?

Computational:

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

# Machine translation

Special case of structured prediction, where

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

To be defined:

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

# Towards joint feature map

Use the standard phrase extraction heuristic for feature extraction:

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  $(x, a, y) \in S$ :

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

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

nous	devons	nous	devons	leur	en	donner	
	,		,	\	/	,	...
we	must	we	must	give	them		

In Sinuhe: drop singleton biphrases (loo motivation)

# From biphrases to joint feature map

Motivating idea:

- Given source sentence  $x$ , predict the set of biphrases that would be extracted from it.

# Joint feature map

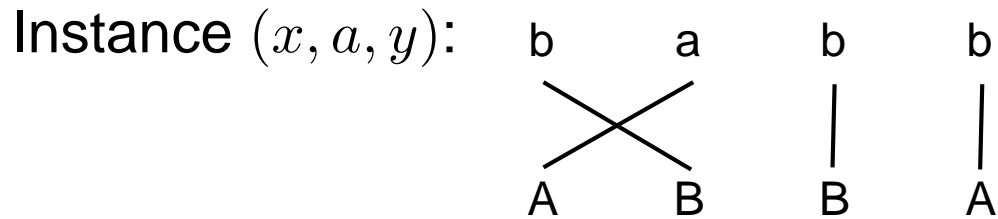
Represent an aligned sentence pair  $(x, a, y)$  by the biphrases that occur in it (and were extracted from the training data):

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

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

Joint feature map:  $(x, a, y) \mapsto \tilde{\phi}(x, a, y)$ .

# Example



Phrase table:

	$\lceil a \rceil$       A   - -	$\lceil a \rceil$       B   - -	$\lceil b \rceil$       B   - -	$\lceil b \quad a \rceil$       A B   - - -	$\lceil b \quad a \rceil$       A B   - - -
$\phi(x, a, y)_{*,0}$	1	0	0	1	0
$\phi(x, a, y)_{*,1}$	0	0	1	0	0
$\phi(x, a, y)_{*,2}$	0	0	1	0	0
$\phi(x, a, y)_{*,3}$	0	0	0	0	0
$\tilde{\phi}(x, a, y)_*$	1	0	2	1	0

# 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  $\Phi_x$  is the set of feasible feature vectors for  $x$ .

Properties:

- Proper conditional probability model for (features of) translations
- Unique “derivation” for each  $(x, a, y)$ , no latent structure
- 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  $P(w)$

Both objectives are convex, no local optima

# 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:

- Stochastic gradients  $\nabla \log P(\phi(x, a, y)|x)$  computed by dynamic programming
- $\tilde{\phi}(x, a, y)$  is sparse, and thus so are the updates
- Easy to parallelize:
  - Apply many stochastic gradient updates asynchronously in parallel

## Predicting translations without a LM

1. Solve  $g_w(x) = \arg \max_{\phi \in \Phi_x} P(\phi|x)$  by dynamic programming
2. Reconstruct  $y = f_w(x)$  from  $g_w(x)$  by glueing together the target phrases in the left-to-right order of their source phrases

Fast, but fluency may be suboptimal (no language model, no reordering model, ...)

## Predicting translations with a LM

- Augment the translation model  $\log P(\phi|x)$  with other features:
  - language model  $\log P(y)$ , fwd/rev lexical translation models, reordering model, translation length
- Find  $y$  by optimizing a weighted combination of the features
  - Beam search
  - Combination weights tuned on development data

Slower and uglier, but 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

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

2. **Model:** conditional exponential probability distribution:

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

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)  $f_w(x) = \text{some } y \text{ reconstructed from } \hat{\phi}(x)$

# Sinuhe — a prototype MT system

- Released under GPLv3 (current version v1.3\_rc2.1)
- Written in C++, about 12000 lines of code (+some scripts)
- Translation model on-disk, in-memory, or **on a server**
  - Distributed training and prediction
- 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  $\approx 200$  CPU cores converges in a few days
- Outputs:  $n$ -best lists, random translations, bag-of-words, ...
- Fast, relatively small memory footprint, reasonable translation quality

# Distributed architecture

- Servers:
  - Multi-threaded `pdb_server` manages queries and updates to components of  $w$
  - Training data stored on `train_data_server`
- Training:
  - Multiple `train` clients connect to servers and run stochastic gradient asynchronously in parallel
- Prediction:
  - Multi-threaded `predict`, either stand-alone or with `pdb_server`

Robust against hardware and software errors, scales well to hundreds of CPUs

# Experimental results

- Systems:
  - Sinuhe: drop singleton biphrases (loo motivation)
  - Moses: use WMT'08 defaults
- Results on Europarl\_v3 data (2000 test sentences, single thread)

	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
Sinuhe <sub>trans</sub>	29.14	27.12	28.74	26.06	22.38	17.14	44.2
Moses <sub>trans</sub>	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 vs. Moses: 26.98

# Conclusions

- `Sinuhe` provides further evidence that MT by ML is feasible:
  - Scalable and fast
  - Better statistical foundations?
  - But BLEU scores still (slightly?) behind state-of-the-art. . .
- Marketing:
  - `Sinuhe` source code and models:
    - \* `http://www.cs.helsinki.fi/u/mtkaaria/sinuhe`
  - Wikipedia demo:
    - \* `http://cosco-demo.hiit.fi/smart`

# Open issues

- Additional features for translation model?
  - Separate weights for maximal biphrases
  - Biphrases over POS tags, wildcards, . . .
- Better integration of LM into the system?
  - Model for  $P(y|\phi)$ ?
  - Noisy channel:  $P(y|x) \propto P(x|y)P(y)$  with *normalized*  $P(x|y)$
- More uses for the probabilistic framework?
- Other tuning (regularization, . . .)?

# Experiments with pruned phrase table

Europarl fr-en data	Sinuhe	Moses <sub>pruned</sub>	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 only
- The effects of pruning and regularization still not quite understood...