n-gram-based MT : what's behind us, what's ahead

F. Yvon and the LIMSI MT crew

LIMSI — CNRS and Université Paris Sud

MT Marathon in Prague, Sep 08th, 2015

Outline

1. overview: MT @ LIMSI
2. *n*-gram-based MT: Basics
3. Continuous space LMs and TMs: SOUL and beyond
4. From n-gram to CRF based TMs
5. Conclusion

Outline

MT @ LIMSI: some facts and numbers

Statistical Machine Learning and Machine Translation (PI: F. Yvon)

- Part of "Spoken Language Processing"
- Joint venture with "Information, Written and Signed Languages"
- Contributors:
  - 5 faculty members (Univ. Paris-Sud) + 2 CNRS researchers
  - 9 Ph.D students
  - 2 post-docs
- Main Theme: Structured Machine learning for multilingual NLP
  - sequence labeling, dependency parsing, WSD
  - weakly supervised learning & cross-lingual transfert
  - alignment models, statistical machine translation

☞ http://www.limsi.fr/tlp [Machine Translation]

MT @ LIMSI: Recent Activities and Contributions

Covering all aspects of Multilingual (spoken and written) NLP

- Some recent contributions
  - Discriminative & sampling-based alignements models [AMTA'10, IWSLT'10, MT'13, MT'14]
  - Contextual models, on-the-fly learning for SMT [IWSLT'13, IWSLT'14]
  - Large-scale continuous space language and translation models [ICASSP'11, NAACL'12, AMTA'14, IWSLT'14, EMNLP'15]
  - Large-scale discriminative learning for SMT [WMT'11, TALN'13]
  - Evaluation: computing oracles, quality estimation [MT'13, ACM TSLP'13, WMT'13...]
  - Ambiguous supervision and cross-lingual transfert [TALN'14, EMNLP'14]
  - Structured learning with large, structured, output spaces [ACL'10, LREC'12, InterSpeech'13, TALN'15, InterSpeech'15, EMNLP'15]
- Current Projects (multi-lingual NLP)
- Evaluation campaigns

MT @ LIMSI: Recent Activities and Contributions

Covering all aspects of Multilingual (spoken and written) NLP

- Some recent contributions
- Current Projects (multi-lingual NLP)
  - QT-21: Quality translation for 21 languages [H2020, +10 academic, TAUS, Tilde...]
  - Transread: towards bilingual reading [French ANR, +CNAM, Reverso]
  - Papyrus: cross-domain and cross-lingual transfert for Information processing [French DGA, +Systran]
  - Bulb: NLP tools for collecting and annotating unwritten languages [German/French ANR, +LPL, LIG, LLACAN, KIT, Uni. Stuttgart]
- Evaluation campaigns

MT @ LIMSI: Recent Activities and Contributions

Covering all aspects of Multilingual (spoken and written) NLP

- Some recent contributions
- Current Projects (multi-lingual NLP)
- Evaluation campaigns
  - WMT Translation [2007-2015], Quality Estimation [2012-2015], Metrics [2015]
    consistently among the top systems for English:French both directions
  - IWSLT Translation [2010, 2011, 2014], Recognition+Translation [2014]
  - SemEval 2015 [Task 13: all word WSDs]
    best system for English

Outline

1. overview: MT @ LIMSI
2. *n*-gram-based MT: Basics
   - Tuples: bilingual units for SMT
   - How is this done ?
   - Order
   - Simplicity of the *n*-gram based approach
3. Continuous space LMs and TMs: SOUL and beyond
   - Towards large-scle CSTMs
   - Discriminative training for NNs
4. From n-gram to CRF based TMs
5. Conclusion
   - Roadmap

Bilingual *n*-grams for Statistical Machine Translation

n-gram LM of tuples

- a bilingual language model as primary translation model
- parallel sentences are sequences of tuples = synchronous phrases

	$u_1 = (f,e)_1$	$u_2 = (f,e)_2$	$u_3 = (f,e)_3$	$u_4 = (f,e)_4$
$\mathbf{f} =$	we	want	translations	perfect
$\mathbf{e} =$	nous	voulons	des traductions	parfaites

- translation context introduced through tuple **n**-gram history

$$P(\mathbf{f},\mathbf{e}) = \prod_{t=1}^{T} P\big((f,e)_t | (f,e)_{t-1},(f,e)_{t-2}\big)$$

with back-off, smoothing, *etc.*

Bilingual *n*-grams for Statistical Machine Translation

n-gram LM of tuples

- a bilingual language model as primary translation model
- parallel sentences are sequences of tuples = synchronous phrases

	$u_1 = (f, e)_1$	$u_2 = (f, e)_2$	$u_3 = (f, e)_3$	$u_4 = (f, e)_4$
$\mathbf{f} =$	we	want	translations	perfect
$\mathbf{e} =$	nous	voulons	des traductions	parfaites

- translation context introduced through tuple **n**-gram history

$$P(\mathbf{f}, \mathbf{e}) = \prod_{t=1}^{T} P\big((f, e)_t | (f, e)_{t-1}, (f, e)_{t-2}\big)$$

with back-off, smoothing, *etc.*

Training and Decoding with *n*-gram TMs

Training

1. identify tuples
2. synchronize bitext

   asymmetric, target oriented
3. train LM
4. train reordering component

Steps 1 and 2 are currently performed simultaneously (but don't need to be)

Decoding

1. generate source reorderings $L(\mathbf{f})$
2. solve:

$$\mathbf{e}^* = \operatorname*{argmax}_{\tilde{\mathbf{f}} \in L(\mathbf{f})} P(\tilde{\mathbf{f}}, \mathbf{e})$$

or use the standard log-linear model

Training and Decoding with n-gram TMs

Training

1. identify tuples
2. synchronize bitext
   asymmetric, target oriented
3. train LM
4. train reordering component

Decoding

1. generate source reorderings $L(\mathbf{f})$
2. solve:

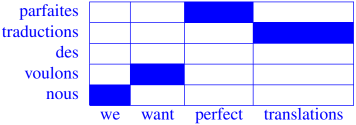
$$\mathbf{e}^* = \underset{\tilde{\mathbf{f}} \in L(\mathbf{f})}{\operatorname{argmax}} P(\tilde{\mathbf{f}}, \mathbf{e})$$

or use the standard log-linear model

Steps 1+2: extract tuples, synchronize phrase pairs

Extracting tuples from word alignments

1. compute (symmetric) word alignments



2. a unique joint segmentation of each sentence pair
3. no NULL on the source side
   - source-NULL can't be predicted
   - attach the target word to the **previous/next** tuple

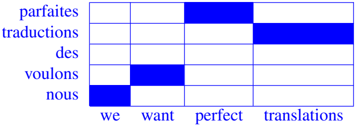
we	want	translations	perfect
nous	voulons	des traductions	parfaites

   - optimizing attachment direction

Steps 1+2: extract tuples, synchronize phrase pairs

Extracting tuples from word alignments

1. compute (symmetric) word alignments



2. a unique joint segmentation of each sentence pair
   - source words are reordered to match target word order
   - no word in a tuple can be aligned outside the tuple
   - maximal segmentation yield minimal tuples

we	want	NULL	translations	perfect
nous	voulons	des	traductions	parfaites

3. no NULL on the source side
   - source-NULL can't be predicted
   - attach the target word to the **previous/next** tuple

we	want	translations	perfect
nous	voulons	des traductions	parfaites

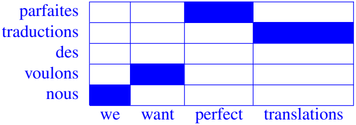
   - optimizing attachment direction

Limsi

Steps 1+2: extract tuples, synchronize phrase pairs

Extracting tuples from word alignments

1. compute (symmetric) word alignments



2. a unique joint segmentation of each sentence pair
3. no NULL on the source side
   - source-NULL can't be predicted
   - attach the target word to the **previous/next** tuple

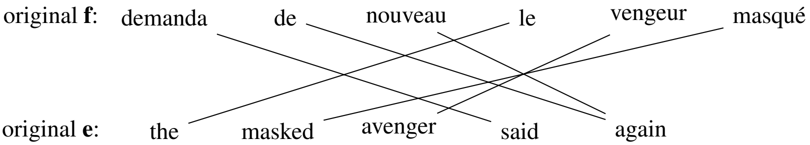
we	want	translations	perfect
nous	voulons	des traductions	parfaites

   - optimizing attachment direction

Bitext synchronization

Reordering and segmenting parallel sentences

1. unfold the word alignments
2. segment into minimal bilingual units → a tuple sequence



Bitext synchronization

Reordering and segmenting parallel sentences

1. unfold the word alignments
2. segment into minimal bilingual units → a tuple sequence

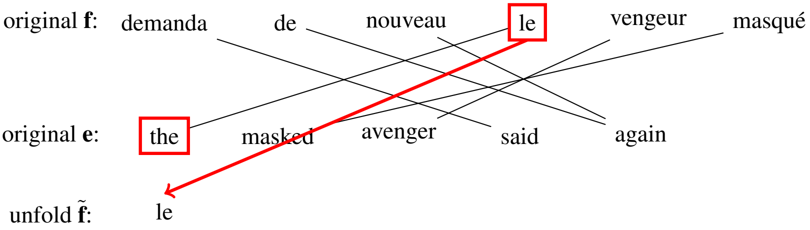
original **f**: demanda de nouveau le vengeur masqué

original **e**: the masked avenger said again

Bitext synchronization

Reordering and segmenting parallel sentences

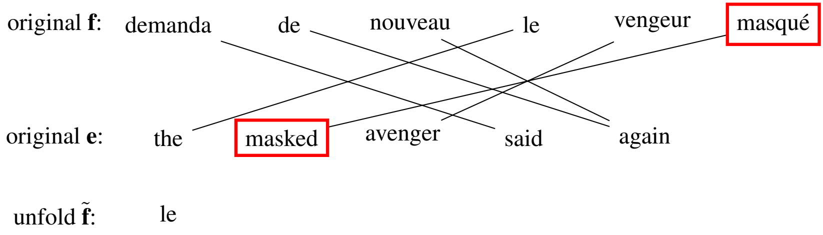
1. unfold the word alignments
2. segment into minimal bilingual units → a tuple sequence



Bitext synchronization

Reordering and segmenting parallel sentences

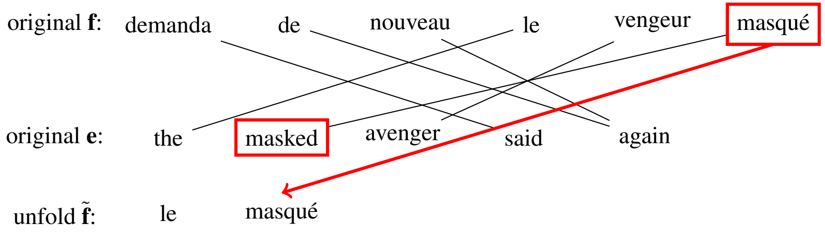
1. unfold the word alignments
2. segment into minimal bilingual units → a tuple sequence



Bitext synchronization

Reordering and segmenting parallel sentences

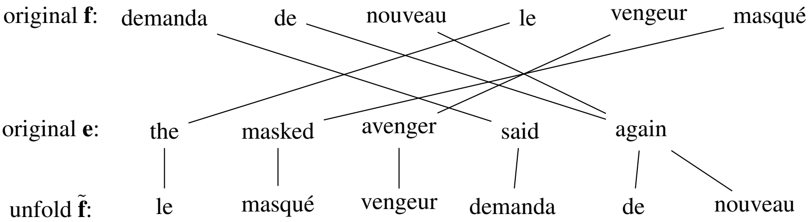
1. unfold the word alignments
2. segment into minimal bilingual units → a tuple sequence



Bitext synchronization

Reordering and segmenting parallel sentences

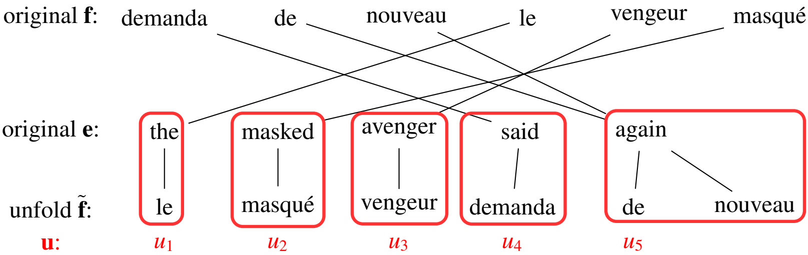
1. unfold the word alignments
2. segment into minimal bilingual units $\rightarrow$ a tuple sequence



Bitext synchronization

Reordering and segmenting parallel sentences

1. unfold the word alignments
2. segment into minimal bilingual units → a tuple sequence



Word (dis)order issues

Towards Dissociating reordering and decoding

Reproducing source reorderings

Solving $\mathbf{e}^* = \operatorname{argmax}_{\tilde{\mathbf{f}} \in L(\mathbf{f})} P(\tilde{\mathbf{f}}, \mathbf{e})$ assumes $L(\mathbf{f})$
$L(\mathbf{f})$ is a set of reordering hypotheses

Generating permutations

Our way: learn rewrite reordering rules from word alignments

Decoding is easy (Finite-State SMT (Bengalore et al, 2000))

Word (dis)order issues

Towards Dissociating reordering and decoding

Reproducing source reorderings

Generating permutations

- $L(\mathbf{f})$ =all ($|\mathbf{f}|!$) permutations is untractable
  permutations make MT NP-hard
- combinatorial reorderings: distance-based, WJ1, IBM, ITG, *etc.*
  computationally effective (polynomial), linguistically risky

Our way: learn rewrite reordering rules from word alignments

Decoding is easy (Finite-State SMT (Bengalore et al, 2000))

Word (dis)order issues

Towards Dissociating reordering and decoding

Reproducing source reorderings

Generating permutations

Our way: learn rewrite reordering rules from word alignments

1. crossing alignment: perfect translations ||| translations perfect
   lexical rules: $r =$ perfect translations $\rightsquigarrow 2\ 1$
   POS rules: $r =$ JJ NN $\rightsquigarrow 2\ 1$
2. compose rules as a reordering transducer $R = \bigcirc_i (r_i \cup Id)$
3. in decoding: $L(\mathbf{f}) = \pi_1(\text{tag}(\mathbf{f}) \circ R)$
   Computes $L(\mathbf{f}$ as a word lattice

Decoding is easy (Finite-State SMT (Bengalore et al, 2000))

Word (dis)order issues

Towards Dissociating reordering and decoding

Reproducing source reorderings

Generating permutations

Our way: learn rewrite reordering rules from word alignments

Decoding is easy (Finite-State SMT (Bengalore et al, 2000))

$$\mathbf{e}^* = \mathit{bestpath}(\pi_2(\mathbf{L}(\mathbf{f}) \circ \mathit{pt}) \circ \mathit{lm})$$

Comparison with (PB)-Moses

- translation units are minimal
- training segmentation is deterministic much smaller models, well-defined transduction models, much less spurious derivations
- static vs dynamic reordering spaces
- different search and pruning strategies

n-gram based approach: pros and cons

- ☺ isolates two main components
  - reordering model (can vary accross language pairs)
  - translation model
- ☺ leverages $\pm$ 20 yrs of LM technologies (and counting)
  (smoothing techniques, adaptation, trigger-based LMs, skip LMs, etc)
- ☺ scales to very-large bitexts (hardly any redundancy in TM + LM compression techniques)
- ☺ decoding (search) is easy - use *generic* finite-state technologies
  generate Nbest, lattices, etc. + larger translation options (reordering is small)
- ☹ source reordering is difficult (and ill-posed)
- ☺ performance $\approx$ to other PB systems for many European language pairs

Recent improvements of N-gram based models

The building blocks

1. identify tuples
2. synchronize bitexts
3. train TM as LM
4. train reordering component
5. include more models

Recent improvements of N-gram based models

The building blocks : what we have tried

1. identify tuples: + discontiguous tuples [Crego and Yvon, 2009]
2. synchronize bitexts: + discriminative alignments [Tomeh et al., 2014]
3. train TM as LM
4. train reordering component
5. include additional models: + lex. reordering, +source LM [Crego and Yvon, 2010]

Leveraging improved LM modeling techniques

- ☑ class-based LMs
- ☑ LM adaptation [Bellagarda, 2001]
- ☑ factored models [Bilmes and Kirchhoff, 2003]
- ☑ compact LMs [Heafield, 2011]
- ☑ continuous-space LMs [Bengio et al., 2003]
- ☑ discriminative LMs [Roark et al., 2004]
- ☐ whole sentence log-linear LMs [Rosenfeld et al., 2001]
- ☐ Bayesian models with HDPs à la [Teh, 2006]
- ☐ M-Models [Chen, 2009]
- ☐ training with fractional counts [Zhang and Chiang, 2014]
  (include incertainty in alignment / segmentation)

Outline

The tuple-based *n*-gram translation model

Can be conventionally learnt with NNs

Training LMs: the lazy way

the *n*-gram translation model ...

$$P(\mathbf{f}, \mathbf{e}) = \prod_{i=1}^{L} P(u_i | u_{i-1}, ..., u_{i-n+1})$$

... is easy to train
(CMU-LM, SriLM, IRSTLM, KenLM, (yes, we even have tried LimsiLM))

The lazy way is the inefficient way

- elementary units are tuples $\Rightarrow$ Very large unit set
- very sparse training data.
- smoothing is a big problem

☞Decompose tuples in smaller parts $\oplus$ use best-known smoothing: NNs

The tuple-based *n*-gram translation model

Can be conventionally learnt with NNs

Training LMs: the lazy way

the *n*-gram translation model ...

$$P(\mathbf{f}, \mathbf{e}) = \prod_{i=1}^{L} P(u_i | u_{i-1}, ..., u_{i-n+1})$$

... is easy to train
(CMU-LM, SriLM, IRSTLM, KenLM, (yes, we even have tried LimsiLM))

The lazy way is the inefficient way

- elementary units are tuples ⇒ Very large unit set
- very sparse training data.
- smoothing is a big problem

☞Decompose tuples in smaller parts ⊕ use best-known smoothing: NNs

The tuple-based n-gram translation model

Can be conventionally learnt with NNs

Training LMs: the lazy way

the n-gram translation model ...

$$P(\mathbf{f}, \mathbf{e}) = \prod_{i=1}^{L} P(u_i | u_{i-1}, \dots, u_{i-n+1})$$

... is easy to train
(CMU-LM, SriLM, IRSTLM, KenLM, (yes, we even have tried LimsiLM))

The lazy way is the inefficient way

- elementary units are tuples $\Rightarrow$ Very large unit set
- very sparse training data.
- smoothing is a big problem

☞Decompose tuples in smaller parts $\oplus$ use best-known smoothing: NNs

The phrase-factored *n*-gram translation model

A novelty of the factored *n*-gram-based TM

Decompose tuples in phrases

org :	....	à	recevoir	le	prix nobel de la paix		
s :	....	$\bar{s}_8$: à	$\bar{s}_9$: recevoir	$\bar{s}_{10}$: le	$\bar{s}_{11}$: nobel de la paix	$\bar{s}_{12}$: prix	....
t :	....	$\bar{t}_8$: to	$\bar{t}_9$: receive	$\bar{t}_{10}$: the	$\bar{t}_{11}$: nobel peace	$\bar{t}_{12}$: prize	....
		u_8	u_9	u_{10}	u_{11}	u_{12}	

Notations:

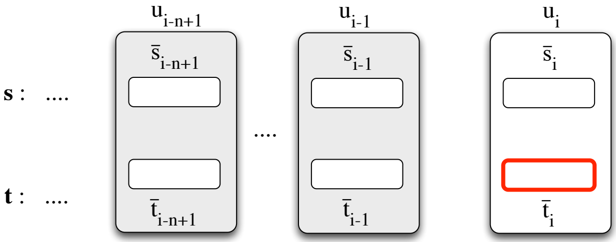
- $u = (\bar{s}, \bar{t})$: a tuple
- $\bar{s}$: the source side of u
- $\bar{t}$: the target side of u

The phrase-factored *n*-gram translation model

$$P(u_i|u_{i-1},\ldots,u_{i-n+1}) = P(\bar{t}_i|\bar{s}_i,\bar{s}_{i-1},\bar{t}_{i-1},\ldots,\bar{s}_{i-n+1},\bar{t}_{i-n+1})$$

$$\times P(\bar{s}_i|\bar{s}_{i-1},\bar{t}_{i-1},\ldots,\bar{s}_{i-n+1},\bar{t}_{i-n+1})$$

Conditional translation model

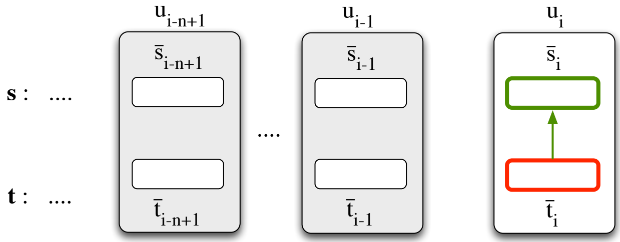


The phrase-factored n-gram translation model

$$P(u_i|u_{i-1},...,u_{i-n+1}) = P(\bar{t}_i|\bar{s}_i, \bar{s}_{i-1}, \bar{t}_{i-1}, ..., \bar{s}_{i-n+1}, \bar{t}_{i-n+1})$$

$$\times P(\bar{s}_i|\bar{s}_{i-1}, \bar{t}_{i-1}, ..., \bar{s}_{i-n+1}, \bar{t}_{i-n+1})$$

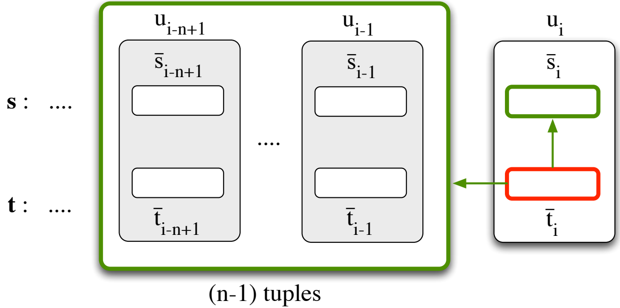
Conditional translation model



The phrase-factored *n*-gram translation model

$$P(u_i|u_{i-1},\dots,u_{i-n+1}) = P(\bar{t}_i|\bar{s}_i, \bar{s}_{i-1},\bar{t}_{i-1},\dots,\bar{s}_{i-n+1},\bar{t}_{i-n+1})$$
$$\times P(\bar{s}_i|\bar{s}_{i-1},\bar{t}_{i-1},\dots,\bar{s}_{i-n+1},\bar{t}_{i-n+1})$$

Conditional translation model

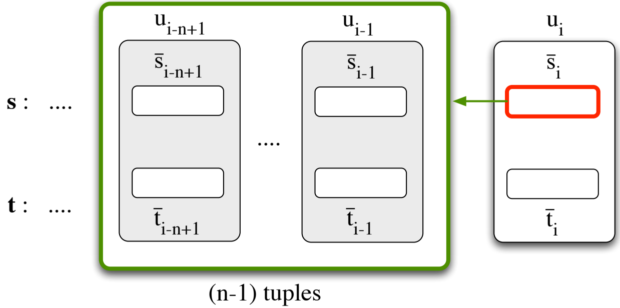


The phrase-factored *n*-gram translation model

$$P(u_i|u_{i-1},\ldots,u_{i-n+1}) = P(\bar{t}_i|\bar{s}_i,\bar{s}_{i-1},\bar{t}_{i-1},\ldots,\bar{s}_{i-n+1},\bar{t}_{i-n+1})$$

$$\times P(\bar{s}_i | \bar{s}_{i-1},\bar{t}_{i-1},\ldots,\bar{s}_{i-n+1},\bar{t}_{i-n+1})$$

A 'distortion' model

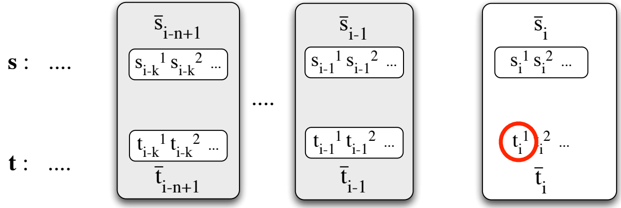


A word-factored n-gram translation model

Decomposing further

$$P(\bar{t}_i|\bar{s}_i, \bar{s}_{i-1}, \bar{t}_{i-1}, \dots, \bar{s}_{i-n+1}, \bar{t}_{i-n+1}) = \prod_{k=1}^{|\bar{t}_i|} P\big(t_i^k \,|\, h^{n-1}(t_i^k), h^{n-1}(s_{i+1}^1) \big)$$

$$P(\bar{s}_i|\bar{s}_{i-1}, \bar{t}_{i-1} \dots, \bar{s}_{i-n+1}, \bar{t}_{i-n+1}) = \prod_{k=1}^{|\bar{s}_i|} P\big(s_i^k | h^{n-1}(s_i^1), h^{n-1}(s_i^k) \big)$$

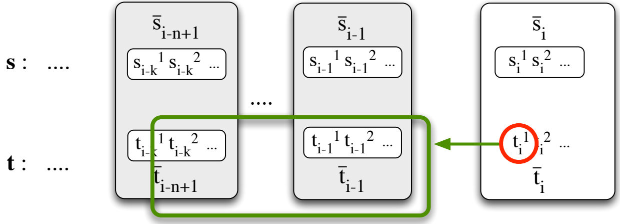


A word-factored *n*-gram translation model

Decomposing further

$$P(\bar{t}_i|\bar{s}_i, \bar{s}_{i-1}, \bar{t}_{i-1}, \ldots, \bar{s}_{i-n+1}, \bar{t}_{i-n+1}) = \prod_{k=1}^{|\bar{t}_i|} P\left(t_i^k \,|\, h^{n-1}(t_i^k), h^{n-1}(s_{i+1}^1) \right)$$

$$P(\bar{s}_i|\bar{s}_{i-1}, \bar{t}_{i-1} \ldots, \bar{s}_{i-n+1}, \bar{t}_{i-n+1}) = \prod_{k=1}^{|\bar{s}_i|} P\left(s_i^k | h^{n-1}(s_i^1), h^{n-1}(s_i^k)\right)$$



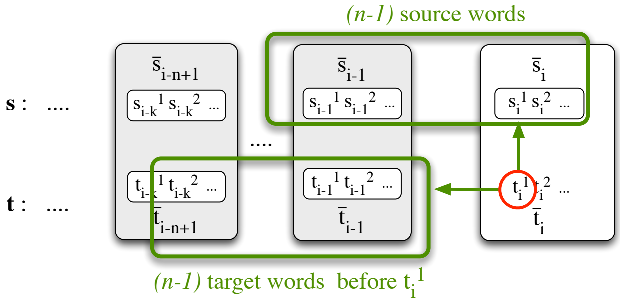
(n-1) target words before t_i^1

A word-factored *n*-gram translation model

Decomposing further

$$P(\bar{t}_i|\bar{s}_i, \bar{s}_{i-1}, \bar{t}_{i-1}, \ldots, \bar{s}_{i-n+1}, \bar{t}_{i-n+1}) = \prod_{k=1}^{|\bar{t}_i|} P\left(t_i^k \,|\, h^{n-1}(t_i^k),\ h^{n-1}(s_{i+1}^1) \right)$$

$$P(\bar{s}_i|\bar{s}_{i-1}, \bar{t}_{i-1} \ldots, \bar{s}_{i-n+1}, \bar{t}_{i-n+1}) = \prod_{k=1}^{|\bar{s}_i|} P\left(s_i^k|h^{n-1}(s_i^1), h^{n-1}(s_i^k)\right)$$

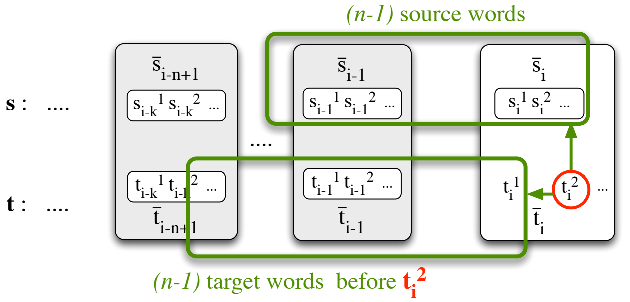


A word-factored n-gram translation model

Decomposing further

$$P(\bar{t}_i|\bar{s}_i, \bar{s}_{i-1}, \bar{t}_{i-1}, \ldots, \bar{s}_{i-n+1}, \bar{t}_{i-n+1}) = \prod_{k=1}^{|\bar{t}_i|} P\left(t_i^k \,\middle|\, \boxed{h^{n-1}(t_i^k)}, \boxed{h^{n-1}(s_{i+1}^1)} \right)$$

$$P(\bar{s}_i|\bar{s}_{i-1}, \bar{t}_{i-1} \ldots, \bar{s}_{i-n+1}, \bar{t}_{i-n+1}) = \prod_{k=1}^{|\bar{s}_i|} P\left(s_i^k | h^{n-1}(s_i^1), h^{n-1}(s_i^k) \right)$$



Three factorization of the n-gram model

Under the n-gram assumption

Three n-gram models of a sentence pair based on different units:

1. tuple-based (u)
2. phrase-factored ($\bar{s}, \bar{t}$)
3. word-factored (s, t)

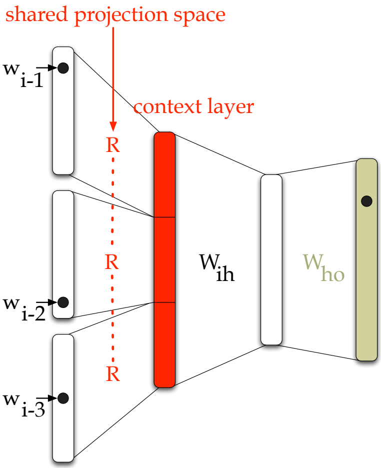
Larger units make sparser models (and conversely)

Continuous space n-gram models

Overview of the standard model [Bengio et al., 2003, Schwenk, 2007]

Projection in a continuous space

- one-hot encodings (in $\{0, 1\}^{|V|}$)
- linear projections in $\mathbb{R}^d$, ($d \ll |V|$)
- merge context vectors in one history



Continuous space *n*-gram models

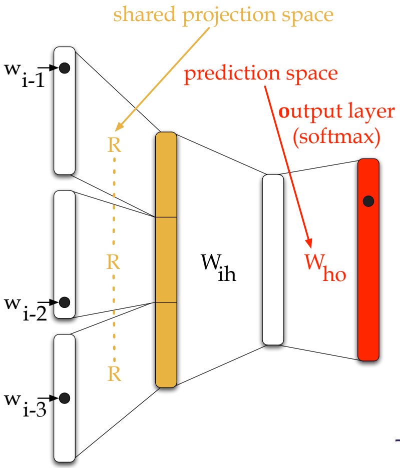
Overview of the standard model [Bengio et al., 2003, Schwenk, 2007]

Projection in a continuous space

- one-hot encodings (in $\{0,1\}^{|V|}$)
- linear projections in $\mathbb{R}^d$, ($d \ll |V|$)
- merge context vectors in one history

Probability estimation

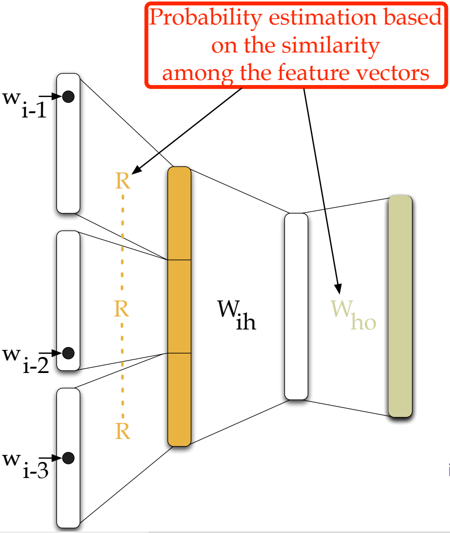
- create a feature vector for the word to be predicted.
- estimate probabilities for all words given history



Large-scale Continuous Space LMs

Key points

- projection in continuous spaces improves smoothing
- joint learning of representation and the prediction layers



i

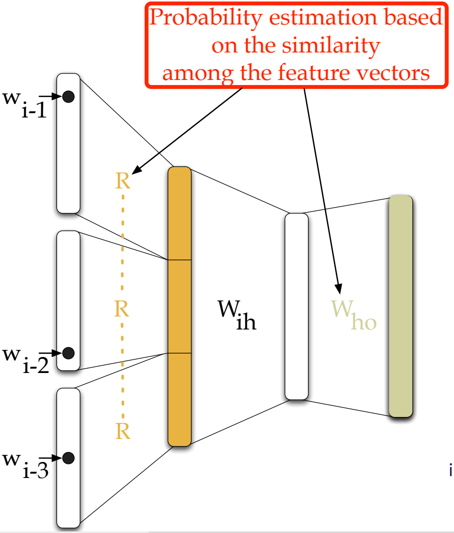
Large-scale Continuous Space LMs

Key points

- projection in continuous spaces improves smoothing
- joint learning of representation and the prediction layers

Complexity issues

- handles arbitrary input vocabularies.
- handles high-order models



i

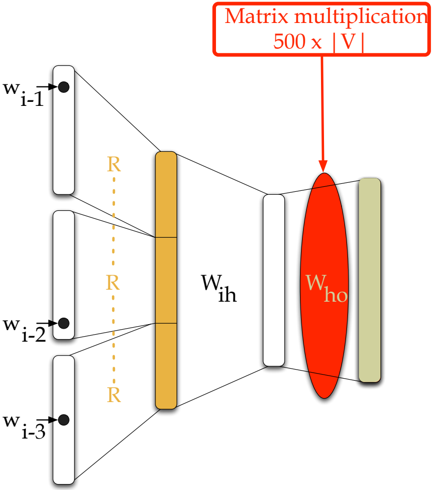
Large-scale Continuous Space LMs

Key points

- projection in continuous spaces improves smoothing
- joint learning of representation and the prediction layers

Complexity issues

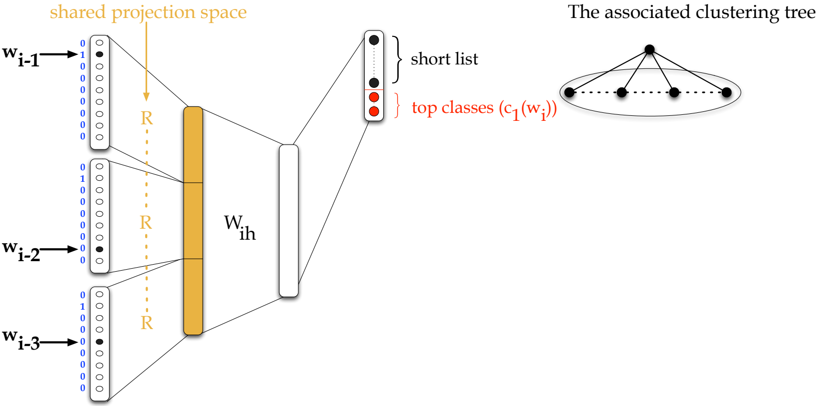
- handles arbitrary input vocabularies.
- handles high-order models
- main bottleneck: output vocabulary size



The SOUL model [Le et al., 2011]

Use a structured output layer

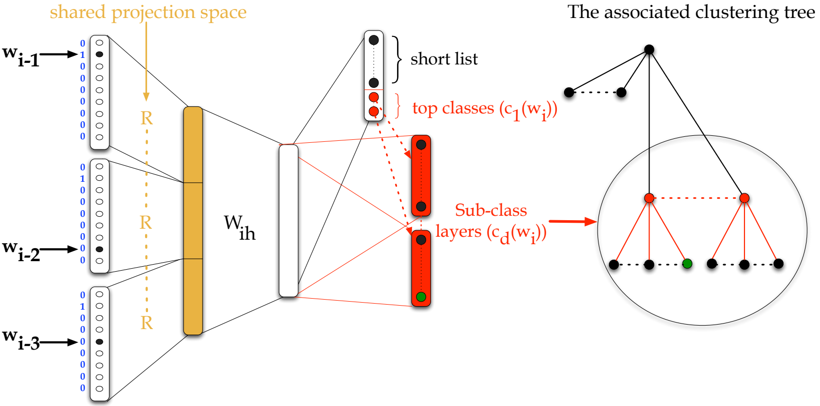
$$P(w_i|h) = P(c_1(w_i)|h)$$



The SOUL model [Le et al., 2011]

Use a structured output layer

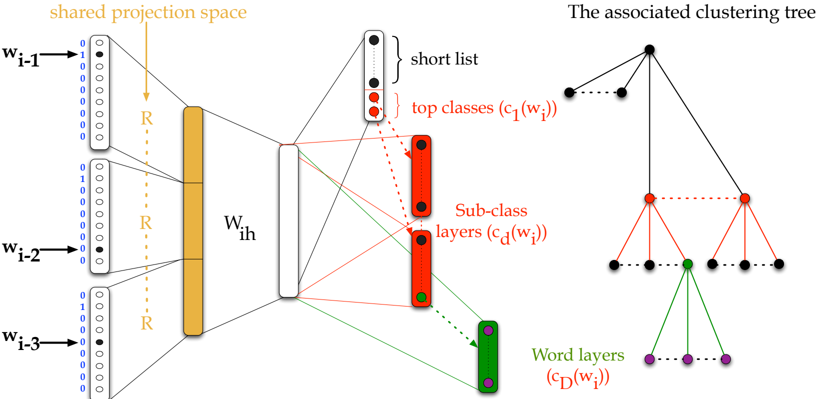
$$P(w_i|h) = P(c_1(w_i)|h) \times \prod_{d=2}^{D} P(c_d(w_i)|h, c_{1:d-1}(w_i))$$



The SOUL model [Le et al., 2011]

Use a structured output layer

$$P(w_i|h) = P(c_1(w_i)|h) \times \prod_{d=2}^{D} P(c_d(w_i)|h, c_{1:d-1}(w_i))$$



Implementing CSLMs with SOUL

The tuple-based *n*-gram translation model

Straightforward implementation (already in [Schwenk et al., 2007])

Phrase and word factored models

They involve two languages and two unit sets:

- the predicted unit is a target phrase (resp. word),
- the context is made of both source and target phrases (resp. words).

☞ use multiple projection matrices ($\mathbf{R_f}$ and $\mathbf{R_e}$).

Implementing CSLMs with SOUL

The tuple-based *n*-gram translation model

Straightforward implementation (already in [Schwenk et al., 2007])

Phrase and word factored models

They involve two languages and two unit sets:

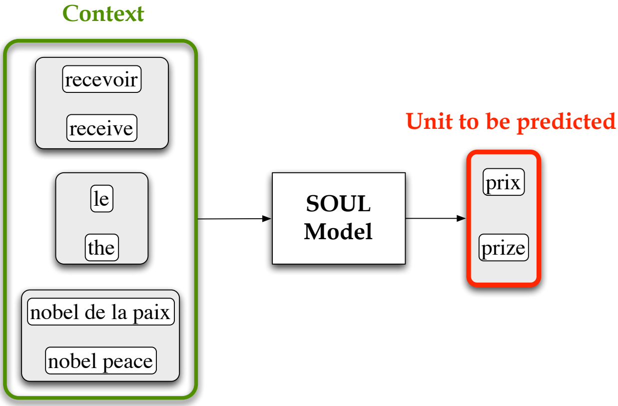
- the predicted unit is a target phrase (resp. word),
- the context is made of both source and target phrases (resp. words).

☞ use multiple projection matrices ($\mathbf{R_f}$ and $\mathbf{R_e}$).

Training example

For a «4-gram» model

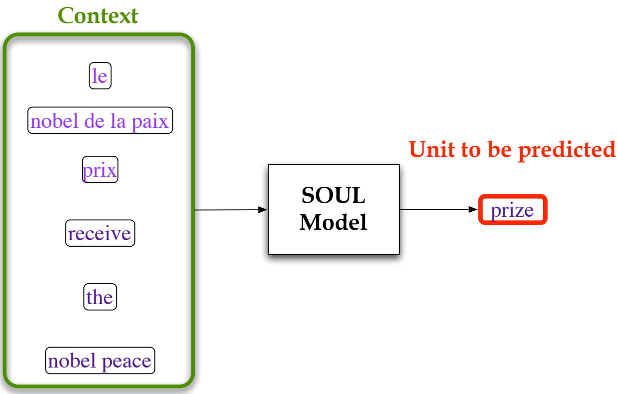
Tuple-based model



Training example

For a «4-gram» model

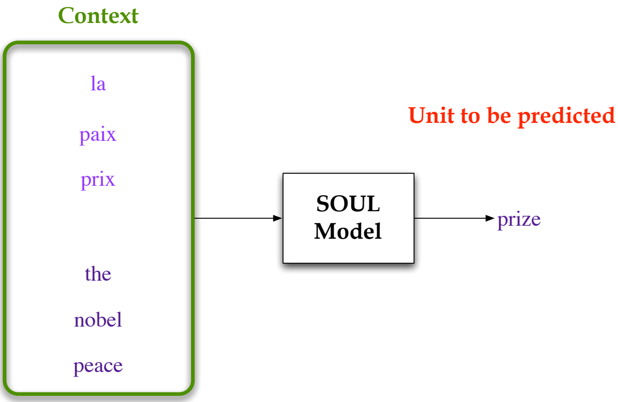
Phrase-based model



Training example

For a «4-gram» model

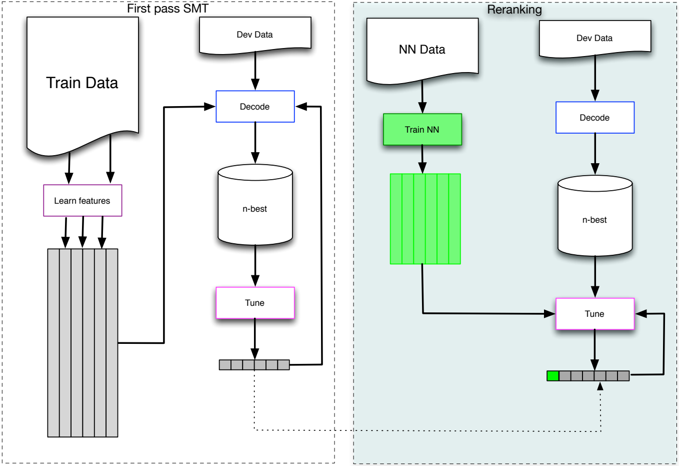
Word-based model



Inference with SOUL

Use wo steps decoding

1. Generate a *k*-best list with the baseline system
2. Re-rank the *k*-best hypotheses (additional feature)



SOUL: promisses and caveats

- ☺ guaranted large BLEU improvements across the board
  see LIMSI@(IWSLT'11 – WMT'15)
- ☺ compatible with any SMT architecture

- ☹ complex training and inference
- ☹ inadequate training objective
- ☹ computationally unsustainable - burns a lot of energy
- ☹ irrealistic in decoding (large histories + computational cost of normalization)
  possible with the "generation" trick

SOUL: promisses and caveats

- ☺ guaranted large BLEU improvements across the board

  see LIMSI@(IWSLT'11 – WMT'15)
- ☺ compatible with any SMT architecture

- ☹ complex training and inference
- ☹ inadequate training objective
- ☹ computationally unsustainable - burns a lot of energy
- ☹ irrealistic in decoding (large histories + computational cost of normalization)

  possible with the "generation" trick

Training objectives for NNLMs and NNTMs

Two generic learning objectives

1. Train NNLMS
   - negated conditional likelihood (including RNN, SOUL, etc):

$$\ell(\boldsymbol{\theta}) = \sum_{(w,h)} -\log P_{\boldsymbol{\theta}}(w|h)(+\mathcal{R}(\boldsymbol{\theta})), \text{ with } P_{\boldsymbol{\theta}}(w|h) = \frac{\exp b_{\boldsymbol{\theta}}(w,h)}{\sum_{w'} \exp b_{\boldsymbol{\theta}}(w',h)}$$

   - NCE: for each observed (h, w), generate k negative samples $(x_1 \ldots x_k)$; optimize:

$$\ell(\boldsymbol{\theta}) = -\sum_{h} \Big(\log P_{\boldsymbol{\theta}}(w|h) - \log(P_{\boldsymbol{\theta}}(w|h) + kP_N(w)) + \sum_{i} \log(P_N(x_i)) - \log(P_\theta(x_i|h) + kP_N(x_i)) \Big)$$

   $P_\theta(w|h)$ unnormalized; $P_N()$ a noise distribution (eg. unigram) [Mnih and Teh, 2012].

2. Train scoring function (log-linear combination) with MERT, MIRA, etc.
3. rerank hypotheses $\mathbf{e}$ with $G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{f},\mathbf{e},\mathbf{a}) = F_{\boldsymbol{\lambda}}(\mathbf{f},\mathbf{e},\mathbf{a}) - \lambda_{k+1} \log(P_{\boldsymbol{\theta}}(\mathbf{e}))$

Training objectives for NNLMs and NNTMs

Two generic learning objectives

1. Train NNLMS
2. Train scoring function (log-linear combination) with MERT, MIRA, etc.
3. rerank hypotheses **e** with $G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{f},\mathbf{e},\mathbf{a}) = F_{\boldsymbol{\lambda}}(\mathbf{f},\mathbf{e},\mathbf{a}) - \lambda_{k+1}\log(P_{\boldsymbol{\theta}}(\mathbf{e}))$

Issues

- step 1 very costly (in training)
- $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$ trained separately
- $\boldsymbol{\theta}$ trained with an inadequate objective

A new ranking objective

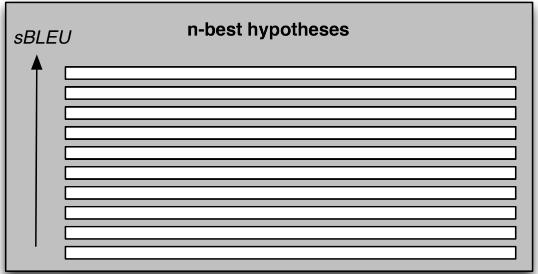
Learning to rank with a margin criterium

BLEU-based cost function

$$\text{cost}_\alpha(\mathbf{h} = (\mathbf{a}, \mathbf{e})) = \alpha\Big(\text{sBLEU}(\mathbf{e}^*) - \text{sBLEU}(\mathbf{e})\Big) \text{ where}$$
$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} \text{sBLEU}(\mathbf{f}) \text{ is the best hypothesis}$$

$(\text{cost}_\alpha(h) \geq 0)$

A Max-margin objective



A new ranking objective

Learning to rank with a margin criterium

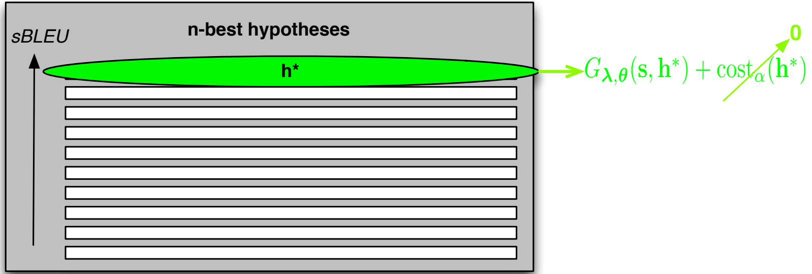
BLEU-based cost function

$$\text{cost}_\alpha(\mathbf{h} = (\mathbf{a}, \mathbf{e})) = \alpha\Big(\text{sBLEU}(\mathbf{e}^*) - \text{sBLEU}(\mathbf{e})\Big) \text{ where}$$

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} \text{sBLEU}(\mathbf{f}) \text{ is the best hypothesis}$$

$(\text{cost}_\alpha(h) \geq 0)$

A Max-margin objective



A new ranking objective

Learning to rank with a margin criterium

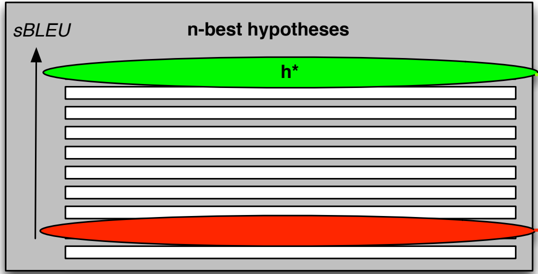
BLEU-based cost function

$$\text{cost}_\alpha(\mathbf{h} = (\mathbf{a}, \mathbf{e})) = \alpha\Big(\text{sBLEU}(\mathbf{e}^*) - \text{sBLEU}(\mathbf{e})\Big) \text{ where}$$

$$\mathbf{e}^* = \underset{\mathbf{e}}{\text{argmax}}\ \text{sBLEU}(\mathbf{f}) \text{ is the best hypothesis}$$

$(\text{cost}_\alpha(h) \geq 0)$

A Max-margin objective



$G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\text{s}, \text{h}^*) + \text{cost}_\alpha(\text{h}^*)$ → 0

$\text{argmax}_{\mathbf{h}} \left[G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}) + \text{cost}_\alpha(\mathbf{h})\right]$

A new ranking objective

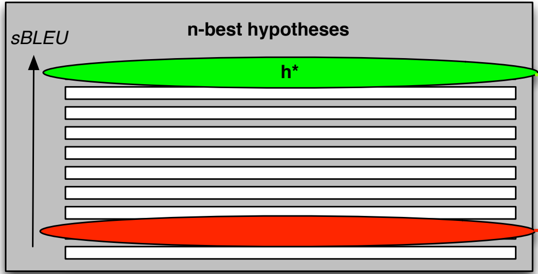
Learning to rank with a margin criterium

BLEU-based cost function

$$\text{cost}_\alpha(\mathbf{h} = (\mathbf{a}, \mathbf{e})) = \alpha\Big(\text{sBLEU}(\mathbf{e}^*) - \text{sBLEU}(\mathbf{e})\Big) \text{ where}$$
$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} \text{sBLEU}(\mathbf{f}) \text{ is the best hypothesis}$$

$(\text{cost}_\alpha(h) \geq 0)$

A Max-margin objective



$G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathrm{s}, \mathrm{h}^*) + \text{cost}_\alpha(\mathrm{h}^*)$ 0

$\text{argmax}_{\mathbf{h}} \left[G_{\boldsymbol{\lambda},\boldsymbol{\theta}}(\mathbf{s}, \mathbf{h}) + \text{cost}_\alpha(\mathbf{h})\right]$

A new ranking objective

Learning to rank with a margin criterium

BLEU-based cost function

$$\mathrm{cost}_\alpha(\mathbf{h} = (\mathbf{a}, \mathbf{e})) = \alpha\Big(\mathrm{sBLEU}(\mathbf{e}^*) - \mathrm{sBLEU}(\mathbf{e})\Big) \text{ where}$$

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} \mathrm{sBLEU}(\mathbf{f}) \text{ is the best hypothesis}$$

$(\mathrm{cost}_\alpha(h) \geq 0)$

A Max-margin objective

In practice, minimize:

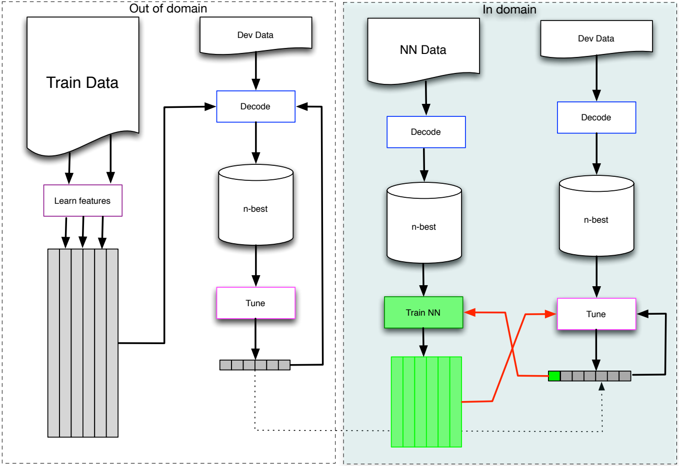
$$\ell(\theta) = \sum_{(i,k)} G_{\lambda,\theta}(\mathbf{f}, \mathbf{h}_k) + \mathrm{cost}_\alpha(\mathbf{h}_k) - G_{\lambda,\theta}(\mathbf{f}, \mathbf{h}_i) - \mathrm{cost}_\alpha(\mathbf{h}_i)$$

where $(\mathbf{h}_i, \mathbf{h}_k)$ are pairs of $(good, bad)$ hypotheses (wrt. sBLEU)

Training discriminative NN: the global view

Still uses two steps decoding

1. generate k-best list with the baseline system for all the training and dev data
2. jointly train re-ranker and NN



Training algorithm

A rather abstract representation

Init. $\boldsymbol{\lambda}$ and $\boldsymbol{\theta}$
for N Iterations **do**
  for M NN-train batches **do**
    Compute sub-gradient of $\ell(\boldsymbol{\theta})$ for each sentence $\mathbf{f}$ in batch
    update $\boldsymbol{\theta}$ $\triangleright$ $\boldsymbol{\lambda}$ fixed
  end for
  update $\boldsymbol{\lambda}$ on dev. set (MERT, MIRA) $\triangleright$ $\boldsymbol{\theta}$ fixed
end for

Some experimental results

NCE vs. CLL

Data and Condition:

- Out-of-domain: WMT en-fr system
- In-domain: TED Talks

Full details in EMNLP paper [Do et al., 2015]

Some experimental results

NCE vs. CLL

	dev	**test**
Baseline	$33,9$	$27,6$
Continuous space models training		
+ SOUL/CLL	$35,1$ ($+1,2$)	$28,9$ ($+1,3$)
+ NCE	$35,0$ ($+1,1$)	$28,8$ ($+1,2$)

Full details in EMNLP paper [Do et al., 2015]

Some experimental results

NCE vs. CLL

	dev	**test**
Baseline	$33,9$	$27,6$
NNs in reranking		
+ NCE	$35,0$	$28,8$
Discriminative training		
+ DT	$35,3$ $(+1,4)$	$29,0$ $(+1,4)$
+ Init. NCE + DT	**35,4** $(+1,5)$	**29,7** $(+2,1)$

comparable results when initializing with SOUL

Full details in EMNLP paper [Do et al., 2015]

Outline

Motivations and inspirations

Motivations

- n-gram models $P(\tilde{\mathbf{f}}, \mathbf{e})$ - Yet $\mathbf{f}$ is known in advance !
  ☞ learn $P(\mathbf{e}|\tilde{\mathbf{f}})$ instead (cf. previous part)
- n-gram models are trained generatively
  ☞ learn TM towards good translations
- n-gram models are "surfacist"
  ☞ integrate reach linguistic features
  cf. factored models in LM and TMs
- Get rid of log-linear combination, tuning, etc.

Motivations and inspirations

Motivations

- n-gram models $P(\tilde{\mathbf{f}}, \mathbf{e})$ - Yet $\mathbf{f}$ is known in advance !
  ☞ learn $P(\mathbf{e}|\tilde{\mathbf{f}})$ instead (cf. previous part)
- n-gram models are trained generatively
  ☞ learn TM towards good translations
- n-gram models are "surfacist"
  ☞ integrate reach linguistic features
  cf. factored models in LM and TMs
- Get rid of log-linear combination, tuning, etc.

Motivations and inspirations

Motivations

- n-gram models $P(\tilde{\mathbf{f}}, \mathbf{e})$ - Yet $\mathbf{f}$ is known in advance !
  ☞ learn $P(\mathbf{e}|\tilde{\mathbf{f}})$ instead (cf. previous part)
- n-gram models are trained generatively
  ☞ learn TM towards good translations
- n-gram models are "surfacist"
  ☞ integrate reach linguistic features
  cf. factored models in LM and TMs
- Get rid of log-linear combination, tuning, etc.

Motivations and inspirations

Motivations

- n-gram models $P(\tilde{\mathbf{f}}, \mathbf{e})$ - Yet $\mathbf{f}$ is known in advance !
  ☞ learn $P(\mathbf{e}|\tilde{\mathbf{f}})$ instead (cf. previous part)
- n-gram models are trained generatively
  ☞ learn TM towards good translations
- n-gram models are "surfacist"
  ☞ integrate reach linguistic features
  cf. factored models in LM and TMs
- Get rid of log-linear combination, tuning, etc.

From n-gram to CRF-based TMs

Implementation

Training

1. identify tuples
2. synchronize bitext
   asymmetric, target oriented
3. train LM
4. train reordering rules

Steps 1 and 2 are performed simultaneously

Decoding

1. generate source reorderings $L(\mathbf{f})$
2. solve:

$$\mathbf{e}^* = \operatorname*{argmax}_{\tilde{\mathbf{f}} \in L(\mathbf{f})} P_{\boldsymbol{\theta}}(\tilde{\mathbf{f}}, \mathbf{e})$$

or use the standard log-linear model

From n-gram to CRF-based TMs

Implementation

Training

1. identify tuples
2. synchronize bitext
   asymmetric, target oriented
3. train CRF
4. train reordering rules

Decoding

1. generate source reorderings $L(\mathbf{f})$
2. solve:

$$\mathbf{e}^* = \operatorname*{argmax}_{\tilde{\mathbf{f}} \in L(\mathbf{f})} P_{\boldsymbol{\theta}}(\mathbf{e}|\tilde{\mathbf{f}})$$

and that is all there is !

The CRF Translation Model

Basic formulation: known tuple alignment (inc. segmentation and reordering)

$$P_{\boldsymbol{\theta}}(\mathbf{e}, \mathbf{a} | \tilde{\mathbf{f}}) = \frac{\exp\left(\boldsymbol{\theta}^{\top} \Phi(\mathbf{e}, \mathbf{a}, \tilde{\mathbf{f}})\right)}{\sum_{\mathbf{e}', \mathbf{a}'} \exp\left(\boldsymbol{\theta}^{\top} \Phi(\mathbf{e}', \mathbf{a}', \tilde{\mathbf{f}})\right)}$$

$$\text{with } \Phi(\mathbf{e}, \mathbf{a}, \tilde{\mathbf{f}})) = \sum_i \Phi(\bar{t}_i, \bar{t}_{i-1}, \tilde{\mathbf{f}}, i)$$

With marginalization (reorderings and segmentations unobserved)

$$P_{\theta}(\mathbf{e}|\mathbf{f}) = \sum_{\tilde{\mathbf{f}} \in L(\mathbf{f})} \sum_{\mathbf{a} \in S(\tilde{\mathbf{f}})} P(\mathbf{e}, \mathbf{a} | \tilde{\mathbf{f}})$$

The CRF Translation Model

Basic formulation: known tuple alignment (inc. segmentation and reordering)

$$P_{\boldsymbol{\theta}}(\mathbf{e},\mathbf{a}|\tilde{\mathbf{f}}) = \frac{\exp\left(\boldsymbol{\theta}^\top \Phi(\mathbf{e},\mathbf{a},\tilde{\mathbf{f}})\right)}{\sum_{\mathbf{e}',\mathbf{a}'} \exp\left(\boldsymbol{\theta}^\top \Phi(\mathbf{e}',\mathbf{a}',\tilde{\mathbf{f}})\right)}$$

$$\text{with } \Phi(\mathbf{e},\mathbf{a},\tilde{\mathbf{f}})) = \sum_i \Phi(\bar{t}_i, \bar{t}_{i-1}, \tilde{\mathbf{f}}, i)$$

With marginalization (reorderings and segmentations unobserved)

$$P_{\boldsymbol{\theta}}(\mathbf{e}|\mathbf{f}) = \sum_{\tilde{\mathbf{f}}\in L(\mathbf{f})} \sum_{\mathbf{a}\in S(\tilde{\mathbf{f}})} P(\mathbf{e},\mathbf{a}|\tilde{\mathbf{f}})$$

Training and Inference

Training: optimize CLL

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_i \log P_{\boldsymbol{\theta}}(\mathbf{e}_i | \mathbf{f}_i)$$

Caveat: objective no longer convex - still doable with gradient based techniques

Approximate inference: find optimal derivation

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} P_{\theta}(\mathbf{e}|\mathbf{f})$$ NP hard

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e},\mathbf{a},\tilde{\mathbf{f}}} P_{\theta}(\mathbf{e}, \mathbf{a}|\tilde{\mathbf{f}})$$ "Viterbi" decoding

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} \sum_{i=1}^{N} P_{\theta}(\mathbf{e}_i, \mathbf{a}_i|\mathbf{f})$$ approx. marginalization with N-Bests

Training and Inference

Training: optimize CLL

$$\theta^* = \operatorname*{argmax}_{\theta} \sum_i \log P_{\theta}(\mathbf{e}_i|\mathbf{f}_i)$$

Caveat: objective no longer convex - still doable with gradient based techniques

Approximate inference: find optimal derivation

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} P_{\theta}(\mathbf{e}|\mathbf{f})$$ NP hard

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e},\mathbf{a},\tilde{\mathbf{f}}} P_{\theta}(\mathbf{e},\mathbf{a}|\tilde{\mathbf{f}})$$ "Viterbi" decoding

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} \sum_{i=1}^{N} P_{\theta}(\mathbf{e}_i,\mathbf{a}_i|\mathbf{f})$$ approx. marginalization with N-Bests

Training and Inference

Training: optimize CLL

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_i \log P_{\boldsymbol{\theta}}(\mathbf{e}_i|\mathbf{f}_i)$$

Caveat: objective no longer convex - still doable with gradient based techniques

Approximate inference: find optimal derivation

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} P_{\boldsymbol{\theta}}(\mathbf{e}|\mathbf{f})$$ NP hard

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e},\mathbf{a},\tilde{\mathbf{f}}} P_{\boldsymbol{\theta}}(\mathbf{e},\mathbf{a}|\tilde{\mathbf{f}})$$ "Viterbi" decoding

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} \sum_{i=1}^{N} P_{\theta}(\mathbf{e}_i,\mathbf{a}_i|\mathbf{f})$$ approx. marginalization with N-Bests

Training and Inference

Training: optimize CLL

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \sum_i \log P_{\boldsymbol{\theta}}(\mathbf{e}_i|\mathbf{f}_i)$$

Caveat: objective no longer convex - still doable with gradient based techniques

Approximate inference: find optimal derivation

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} P_{\boldsymbol{\theta}}(\mathbf{e}|\mathbf{f})$$ NP hard

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e},\mathbf{a},\tilde{\mathbf{f}}} P_{\boldsymbol{\theta}}(\mathbf{e},\mathbf{a}|\tilde{\mathbf{f}})$$ "Viterbi" decoding

$$\mathbf{e}^* = \operatorname*{argmax}_{\mathbf{e}} \sum_{i=1}^{N} P_{\boldsymbol{\theta}}(\mathbf{e}_i,\mathbf{a}_i|\mathbf{f})$$ approx. marginalization with N-Bests

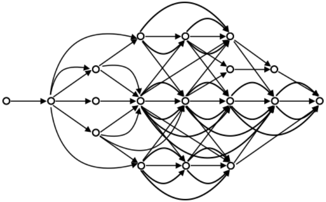
Training: the true story

Training: optimize CLL

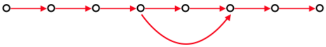
$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \sum_i \log P_{\boldsymbol{\theta}}(\mathbf{e}_i|\mathbf{f}_i) + \alpha||\boldsymbol{\theta}||^2$$

- gradients computed as differences of expectations

$$\frac{\nabla \ell}{\theta_k} = \sum_i \mathbb{E}_{P_{\boldsymbol{\theta}}}(\Phi_k(\mathbf{e}, \mathbf{a}, \mathbf{f}_i)) - \mathbb{E}_{\tilde{P}}(\Phi_k(\mathbf{e}, \mathbf{a}, \mathbf{f}_i))$$



“Possibility” lattice



“Reference” lattice

- reference reachability: reference $\mathbf{e}_i$ not in model

Training: the true story

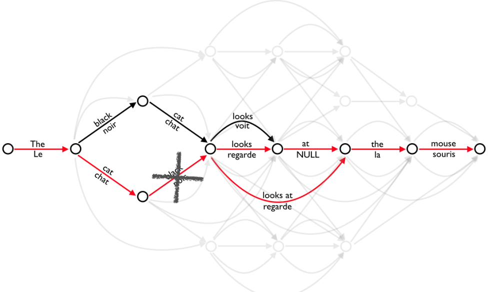
Training: optimize CLL

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \sum_i \log P_{\boldsymbol{\theta}}(\mathbf{e}_i|\mathbf{f}_i) + \alpha||\boldsymbol{\theta}||^2$$

- gradients computed as differences of expectations

$$\frac{\nabla \ell}{\theta_k} = \sum_i \mathbb{E}_{P_{\boldsymbol{\theta}}}(\Phi_k(\mathbf{e},\mathbf{a},\mathbf{f}_i)) - \mathbb{E}_{\tilde{P}}(\Phi_k(\mathbf{e},\mathbf{a},\mathbf{f}_i))$$

- reference reachability: reference $\mathbf{e}_i$ not in model



Training: the true story

Training: optimize CLL

$$\boldsymbol{\theta}^* = \operatorname*{argmax}_{\boldsymbol{\theta}} \ell(\boldsymbol{\theta}) = \sum_i \log P_{\boldsymbol{\theta}}(\mathbf{e}_i|\mathbf{f}_i) + \alpha||\boldsymbol{\theta}||^2$$

- gradients computed as differences of expectations

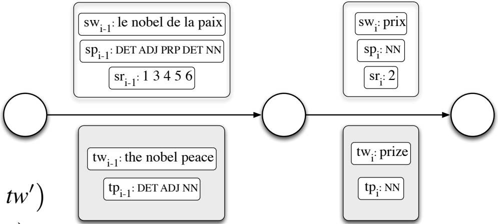
$$\frac{\nabla \ell}{\theta_k} = \sum_i \mathbb{E}_{P_{\boldsymbol{\theta}}}(\Phi_k(\mathbf{e}, \mathbf{a}, \mathbf{f}_i)) - \mathbb{E}_{\tilde{P}}(\Phi_k(\mathbf{e}, \mathbf{a}, \mathbf{f}_i))$$

- reference reachability: reference $\mathbf{e}_i$ not in model
  ☞ use "oracle" derivations instead

caveat: oracles need a goodness measure eg. sBLEU

Feature engineering

Includes LM, TM, RM, and more



1. **LM:uni-tphr**: $\mathbb{I}\left(tw_i = tw\right)$
2. **LM:uni-tpos** : $\mathbb{I}\left(tp_i = tp\right)$
3. **LM:big-tphr** : $\mathbb{I}\left(tw_i = tw \wedge tw_{i-1} = tw'\right)$
4. **LM:big-tpos** : $\mathbb{I}\left(tp_i = tp \wedge tp_{i-1} = tp'\right)$
5. **TM:ci-phrp** : $\mathbb{I}\left(tw_i = tw \wedge sw_i = sw\right)$
6. **TM:ci-posp** : $\mathbb{I}\left(tp_i = tp \wedge sp_i = sp\right)$
7. **TM:ci-mixp** : $\mathbb{I}\left(tw_i = tw \wedge sp_i = sp\right)$
8. **TM:cd-phrs** : $\mathbb{I}\left(tw_i = tw \wedge sw_i = sw \wedge sw_{i-1} = sw'\right)$
9. **TM:cd-poss** : $\mathbb{I}\left(tp_i = tp \wedge sp_i = sp \wedge sp_{i-1} = sp'\right)$
10. **TM:cd-phrt** : $\mathbb{I}\left(tw_i = tw \wedge tw_{i-1} = tw' \wedge sw_i = sw\right)$
11. **TM:cd-post** : $\mathbb{I}\left(tp_i = tp \wedge tp_{i-1} = tp' \wedge sp_i = sp\right)$

CRF-based ngrams: successes and failures

A success story: translating BTEC into French Lavergne et al. [2013]

Configuration	*devel03*	*test09*	*test10*
n-gram-based			
n-gram TM $n = 2$	68.7	61.1	–
n-gram TM $n = 3$	68.0	61.6	53.4
CRF-based			
Viterbi-decoding	64.0	58.8	51.5
+ marginalisation	64.7	59.3	52.0
+ target LM	67.7	**61.7**	**53.9**

Remember: no dense features, no MERT, just plain CRF training on parallel data

A more bumpy road: train on Newsco, translate NewsTest

- Basic config. hardly tractable: > 50B "basic (lexical) features
- "Debug" config: Ncode lattices as proxy search space

CRF-based ngrams: successes and failures

A success story: translating BTEC into French Lavergne et al. [2013]

A more bumpy road: train on Newsco, translate NewsTest

- Basic config. hardly tractable: > 50B “basic (lexical) features
- “Debug” config: Ncode lattices as proxy search space

CRF-based ngrams: successes and failures

A success story: translating BTEC into French Lavergne et al. [2013]

A more bumpy road: train on Newsco, translate NewsTest

- Basic config. hardly tractable: > 50B "basic (lexical) features
- "Debug" config: Ncode lattices as proxy search space

	En→Fr		Fr→En	
	BLEU	BP	*BLEU*	BP
n-gram TM $n = 2$	22.05	0.990	21.99	1.000
CRF (basic)	15.31	0.969	13.96	0.884
CRF (+LM, +p)	16.65	0.970	14.80	0.857
CRF (+dense)	17.52	0.963	16.73	0.881

CRF-based ngrams: successes and failures

A success story: translating BTEC into French Lavergne et al. [2013]

A more bumpy road: train on Newsco, translate NewsTest

- Basic config. hardly tractable: > 50B "basic (lexical) features
- "Debug" config: Ncode lattices as proxy search space
- oracles (pseudo-refs) a problem $\Rightarrow$ length issues (?)
- overtraining a problem
- log-loss a poor objective
- next steps: fix length issue, fix regularization issues, add more features, try alternative losses (eg. soft-max margin)

Discriminative TMs: what we know, what we dont

Confirmation of many studies

1. marginalize nuisance variables if you can
   already well documented
2. the pay-offs of discriminative training
   use translation metrics / cost (eg. *BLEU* in your objective)
3. beware of "dangerous" references
   use hope derivations instead [Chiang, 2012]
4. avoid oracle / pseudo-references if you can
   use ranking [Flanigan et al., 2013] or Expected-*BLEU* [He and Deng, 2012, Gao and He, 2013] etc.
5. sparse or sparse+dense features ?
   Probably an ill-posed alternative, but can we do better ?
6. still the right way to go ?
   time will tell

Discriminative TMs: what we know, what we dont

Confirmation of many studies

1. marginalize nuisance variables if you can
   already well documented
2. the pay-offs of discriminative training
   use translation metrics / cost (eg. *BLEU* in your objective)
3. beware of “dangerous” references
   use hope derivations instead [Chiang, 2012]
4. avoid oracle / pseudo-references if you can
   use ranking [Flanigan et al., 2013] or Expected-*BLEU* [He and Deng, 2012, Gao and He, 2013] etc.
5. sparse or sparse+dense features ?
   Probably an ill-posed alternative, but can we do better ?
6. still the right way to go ?
   time will tell

Discriminative TMs: what we know, what we dont

Confirmation of many studies

1. marginalize nuisance variables if you can
   already well documented
2. the pay-offs of discriminative training
   use translation metrics / cost (eg. *BLEU* in your objective)
3. beware of "dangerous" references
   use hope derivations instead [Chiang, 2012]
4. avoid oracle / pseudo-references if you can
   use ranking [Flanigan et al., 2013] or Expected-*BLEU* [He and Deng, 2012, Gao and He, 2013] etc.
5. sparse or sparse+dense features ?
   Probably an ill-posed alternative, but can we do better ?
6. still the right way to go ?
   time will tell

Discriminative TMs: what we know, what we dont

Confirmation of many studies

1. marginalize nuisance variables if you can
   already well documented
2. the pay-offs of discriminative training
   use translation metrics / cost (eg. *BLEU* in your objective)
3. beware of “dangerous” references
   use hope derivations instead [Chiang, 2012]
4. avoid oracle / pseudo-references if you can
   use ranking [Flanigan et al., 2013] or Expected-*BLEU* [He and Deng, 2012, Gao and He, 2013] etc.
5. sparse or sparse+dense features ?
   Probably an ill-posed alternative, but can we do better ?
6. still the right way to go ?
   time will tell

Discriminative TMs: what we know, what we dont

Confirmation of many studies

1. marginalize nuisance variables if you can
   already well documented
2. the pay-offs of discriminative training
   use translation metrics / cost (eg. *BLEU* in your objective)
3. beware of "dangerous" references
   use hope derivations instead [Chiang, 2012]
4. avoid oracle / pseudo-references if you can
   use ranking [Flanigan et al., 2013] or Expected-*BLEU* [He and Deng, 2012, Gao and He, 2013] etc.
5. sparse or sparse+dense features ?
   Probably an ill-posed alternative, but can we do better ?
6. still the right way to go ?
   time will tell

Discriminative TMs: what we know, what we dont

Confirmation of many studies

1. marginalize nuisance variables if you can
   already well documented
2. the pay-offs of discriminative training
   use translation metrics / cost (eg. *BLEU* in your objective)
3. beware of "dangerous" references
   use hope derivations instead [Chiang, 2012]
4. avoid oracle / pseudo-references if you can
   use ranking [Flanigan et al., 2013] or Expected-*BLEU* [He and Deng, 2012, Gao and He, 2013] etc.
5. sparse or sparse+dense features ?
   Probably an ill-posed alternative, but can we do better ?
6. still the right way to go ?
   time will tell

Outline

n-gram based TMs: a simple and effective implementation of PBMT

What we have

1. open full pipeline for *n*-gram-based MT
2. effective implementation for large-scale NNLMs
3. generic implementation for “generalized” CRFs
   (with latent variable and arbitrary costs) - coming soon

Where we look

1. fix CRF-based model
2. include morpheme-based LMs
3. develop formal characterisation of gappy derivations
4. tick more boxes on slide 17

n-gram based TMs: a simple and effective implementation of PBMT

What we have

1. open full pipeline for n-gram-based MT
2. effective implementation for large-scale NNLMs
3. generic implementation for “generalized” CRFs
   (with latent variable and arbitrary costs) - coming soon

Where we look

1. fix CRF-based model
2. include morpheme-based LMs
3. develop formal characterisation of gappy derivations
4. tick more boxes on slide 17

n-gram based TMs: a simple and effective implementation of PBMT

What we have

1. open full pipeline for *n*-gram-based MT
2. effective implementation for large-scale NNLMs
3. generic implementation for "generalized" CRFs
   (with latent variable and arbitrary costs) - coming soon

Where we look

1. fix CRF-based model
2. include morpheme-based LMs
3. develop formal characterisation of gappy derivations
4. tick more boxes on slide 17

Roadmap

- Improved learning and decoding
  - faster NN training and adaptation with task-related objectives
  - large-scale discriminative learning with sparse features
  - learning to translate with RL / ILR (and very long histories)
- More realistic models
  - more syntax in reordering
  - morphologically aware units for translation
  - optimizing speech segmentation / recognition for MT
  - contextual / discourse level features in MT
- Do more with less resources
  - cross-lingual transfert (in MT and elsewhere)
  - learn tuples from comparable corpora (caveat: require sparse features)
- Better translation environnements
  - improved UIs for the translator workbench
  - seamless online learning, with pre- and post-edition

References I

Jérôme R. Bellagarda. An overview of statistical language model adaptation. In *Proceedings of the ISCA Tutorial and Research Workshop (ITRW) on Adaptation Methods for Speech Recognition*, pages 165–174, Sophia Antipolis, France, 2001.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *JMLR*, 3:1137–1155, 2003. ISSN 1532-4435.

Jeff A. Bilmes and Katrin Kirchhoff. Factored language models and generalized parallel backoff. In *NAACL '03: Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology*, pages 4–6, 2003.

Stanley Chen. Shrinking exponential language models. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 468–476, Boulder, Colorado, 2009.

David Chiang. Hope and fear for discriminative training of statistical translation models. *J. Mach. Learn. Res.*, 13(1):1159–1187, April 2012.

Josep Maria Crego and François Yvon. Gappy translation units under left-to-right SMT decoding. In *Proceedings of the meeting of the European Association for Machine Translation (EAMT)*, pages 66–73, Barcelona, Spain, 2009.

Josep Maria Crego and François Yvon. Improving reordering with linguistically informed bilingual n-grams. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010: Posters)*, pages 197–205, Beijing, China, 2010. Coling 2010 Organizing Committee.

References II

Quoc Khanh Do, Alexandre Allauzen, and François Yvon. A discriminative training procedure for continuous translation models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP 2015)*, page 7, Lisboa, Portugal, 17/09 au 21/09 2015.

Jeffrey Flanigan, Chris Dyer, and Jaime Carbonell. Large-scale discriminative training for statistical machine translation using held-out line search. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 248–258, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Jianfeng Gao and Xiaodong He. Training mrf-based phrase translation models using gradient ascent. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 450–459, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

Xiaodong He and Li Deng. Maximum expected bleu training of phrase and lexicon translation models. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers - Volume 1*, ACL '12, pages 292–301, Stroudsburg, PA, USA, 2012. Association for Computational Linguistics.

Kenneth Heafield. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*, pages 187–197, Edinburgh, Scotland, July 2011. Association for Computational Linguistics.

References III

Thomas Lavergne, Alexandre Allauzen, and François Yvon. Un cadre d'apprentissage intégralement discriminant pour la traduction statistique. In *Actes de la Conférence sur le Traitement Automatique des Langues Naturelles (TALN)*, page 14p, Les Sables d'Olonne, 2013. URL `sources/Lavergne13cadre.pdf`.

Hai-Son Le, Ilya Oparin, Alexandre Allauzen, Jean-Luc Gauvain, and François Yvon. Structured output layer neural network language model. In *Proceedings of ICASSP'11*, pages 5524–5527, 2011.

Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. In *Proceedings of the 29th International Conference on Machine Learning*, pages 1751–1758, 2012.

Brian Roark, Murat Saraclar, Michael Collins, and Mark Johnson. Discriminative language modeling with conditional random fields and the perceptron algorithm. Barcelona, Spain, 2004.

Ronald Rosenfeld, Stanley F. Chen, and Xiaojin Zhu. Whole-sentence exponential language models: a vehicle for linguistic-statistical integration. *Computer, Speech and Language*, 15:55–73, 2001.

Holger Schwenk. Continuous space language models. *Computer Speech and Language*, 21(3):492–518, 2007. ISSN 0885-2308. doi: http://dx.doi.org/10.1016/j.csl.2006.09.003.

Holger Schwenk, Marta R. Costa-jussa, and Jose A. R. Fonollosa. Smooth bilingual *n*-gram translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 430–438, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

References IV

Yeh Weh Teh. A hierarchical Bayesian language model based on Pitman-Yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 985–992, 2006.

Nadi Tomeh, Alexandre Allauzen, and François Yvon. Maximum-entropy word alignment and posterior-based phrase extraction for machine translation. *Machine Translation*, 28(1):19–56, 2014. ISSN 0922-6567. doi: 10.1007/s10590-013-9146-4.

Hui Zhang and David Chiang. Kneser-ney smoothing on expected counts. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 765–774, Baltimore, Maryland, 2014.