Deep Learning for MT

Holger Schwenk

LIUM, University of Le Mans, France

Holger.Schwenk@lium.univ-lemans.fr

MT Marathon – September 11, 2014
Plan of the Talk

Alternative titles

- Neural networks in SMT: past, present and future
- From neural network LMs to fully neural MT systems?

Topics of this talk

- Overview of ongoing research in this area
- Increasing body of research
  → what are the most promising directions (personal view!)
- Vision: can we develop an MT system entirely based on neural networks?
Plan of the Talk

Alternative titles

- Neural networks in SMT: past, present and future
- From neural network LMs to fully neural MT systems?

Topics of this talk

- Overview of ongoing research in this area
- Increasing body of research
  → what are the most promising directions (personal view !)
- Vision: can we develop an MT system entirely based on neural networks?
Plan of the Talk

Alternative titles

- Neural networks in SMT: past, present and future
- From neural network LMs to fully neural MT systems?

Topics of this talk

- Overview of ongoing research in this area
- Increasing body of research
  → what are the most promising directions (personal view!)
- Vision: can we develop an MT system entirely based on neural networks?
Plan of the Talk

Alternative titles

- Neural networks in SMT: past, present and future
- From neural network LMs to fully neural MT systems?

Topics of this talk

- Overview of ongoing research in this area
- Increasing body of research → what are the most promising directions (personal view !)
- Vision: can we develop an MT system entirely based on neural networks?
Plan of the Talk

Alternative titles

- Neural networks in SMT: past, present and future
- From neural network LMs to fully neural MT systems?

Topics of this talk

- Overview of ongoing research in this area
  → what are the most promising directions (personal view!)
- Vision: can we develop an MT system entirely based on neural networks?
Plan of the Talk

Alternative titles

- Neural networks in SMT: past, present and future
- From neural network LMs to fully neural MT systems?

Topics of this talk

- Overview of ongoing research in this area
- Increasing body of research
  → what are the most promising directions (personal view!)
- Vision: can we develop an MT system entirely based on neural networks?
Plan of the Talk

### Alternative titles
- Neural networks in SMT: past, present and future
- From neural network LMs to fully neural MT systems?

### Topics of this talk
- Overview of ongoing research in this area
- Increasing body of research → what are the most promising directions (personal view!)
- Vision: can we develop an MT system entirely based on neural networks?
Applications of Language Models

• Automatic speech recognition (ASR)

\[ \hat{w} = \arg \max_w Pr(w \mid x) = \arg \max_w Pr(w)Pr(x \mid w) \]

• Statistical machine translation (SMT), translate \( f \) to \( e \)

\[ \hat{e} = \arg \max_e Pr(e \mid f) = \arg \max_e Pr(e)Pr(f \mid e) \]

⇒ In both applications we need a LM on the generated word sequence
Applications of Language Models

- **Automatic speech recognition (ASR)**
  \[ \hat{w} = \arg\max_w Pr(w|x) = \arg\max_w Pr(w)Pr(x|w) \]

- **Statistical machine translation (SMT), translate \( f \) to \( e \)**
  \[ \hat{e} = \arg\max_e Pr(e|f) = \arg\max_e Pr(e)Pr(f|e) \]

\[ \Rightarrow \] In both applications we need a LM on the generated word sequence
Applications of Language Models

- Automatic speech recognition (ASR)

\[ \hat{w} = \arg \max_w Pr(w|x) = \arg \max_w Pr(w)Pr(x|w) \]

- Statistical machine translation (SMT), translate \( f \) to \( e \)

\[ \hat{e} = \arg \max_e Pr(e|f) = \arg \max_e Pr(e)Pr(f|e) \]

\[ \Rightarrow \] In both applications we need a LM on the generated word sequence
Applications of Language Models

Speech Recognition

- The LM must choose among a large number of segmentations of the phoneme sequence into words, given the pronunciation lexicon
- The LM must also select among homonyms
- It deals with morphology (gender accordance, ...)
- The word order is given by the sequential processing of speech
Applications of Language Models

Machine translation

- Deal with morphology like for ASR
- The LM helps to choose between different translations
- Translation may require word reordering for certain language pairs
  ⇒ the LM has to sort out the good and the bad ones

Comparison

- It is an interesting question whether language modeling for MT is more or less difficult than for ASR
- One may consider that the semantic level is more important in MT
Applications of Language Models

Machine translation

- Deal with morphology like for ASR
- The LM helps to choose between different translations
- Translation may require word reordering for certain language pairs
  \[ \Rightarrow \] the LM has to sort out the good and the bad ones

Comparison

- It is an interesting question whether language modeling for MT is more or less difficult than for ASR
- One may consider that the semantic level is more important in MT
Some Past Research on LM for SMT

There is no better data than more data

- Stupid back-off [Brants et al., EMNLP’07]
- Bloom Filter and randomized LMs [Talbot et al, EMNLP’07; ACL’07; ...]
- Fast KenLM to train on common crawl

...  
+ Usually improved translation performance
- But these models are often intractable for standard users
⇒ Distributed instead of memory-based models

Learn from the data instead of memorizing it
Some Past Research on LM for SMT

New LM approaches

• I’m not aware of new LM smoothing techniques which are able to beat back-off LMs trained on huge amounts of data

• Exception: Model M (but mainly used in ASR)
Continuous Space LM

Theoretical drawbacks of back-off LM:

- Words are represented in a high-dimensional discrete space
- Probability distributions are not smooth functions
- Any change of the word indices can result in an arbitrary change of LM probability

⇒ True generalization is difficult to obtain

Main idea [Y. Bengio, NIPS’01]:

- Project word indices onto a continuous space and use a probability estimator operating on this space
- Probability functions are smooth functions and better generalization can be expected
Continuous Space LM

Theoretical drawbacks of back-off LM:

- Words are represented in a high-dimensional discrete space
- Probability distributions are not smooth functions
- Any change of the word indices can result in an arbitrary change of LM probability

⇒ True generalization is difficult to obtain

Main idea [Y. Bengio, NIPS’01]:

- Project word indices onto a continuous space and use a probability estimator operating on this space
- Probability functions are smooth functions and better generalization can be expected
CSLM - Probability Calculation

- Outputs = LM posterior probabilities of all words: \( P(w_j = i| h_j) \ \forall i \in [1, N] \)
- Context \( h_j = \) sequence of \( n-1 \) points in this space
- Word = point in the \( P \) dimensional space
- Projection onto continuous space
- Inputs = indices of the \( n-1 \) previous words

\[
h_j = w_{j-n+1}, \ldots, w_{j-2}, w_{j-1}
\]
CSLM - Probability Calculation

- Outputs = LM posterior probabilities of all words: $P(w_j = i | h_j) \ \forall i \in [1, N]$
- Context $h_j = \text{sequence of } n-1 \text{ points in this space}$
- Word = point in the $P$ dimensional space
- Projection onto continuous space
- Inputs = indices of the $n-1$ previous words

\[ h_j = w_{j-n+1}, \ldots, w_{j-2}, w_{j-1} \]
CSLM - Probability Calculation

- Outputs = LM posterior probabilities of all words: $P(w_j = i|h_j) \ \forall i \in [1, N]$
- Context $h_j = \text{sequence of } n-1 \text{ points in this space}$
- Word = point in the $P$ dimensional space
- Projection onto continuous space
- Inputs = indices of the $n-1$ previous words

$h_j = w_{j-n+1}, \ldots, w_{j-2}, w_{j-1}$
**CSLM - Probability Calculation**

- Outputs = LM posterior probabilities of all words: 
  \[ P(w_j = i | h_j) \quad \forall i \in [1, N] \]
- Context \( h_j = \text{sequence of } n-1 \text{ points in this space} \)
- Word = point in the \( P \) dimensional space
- Projection onto continuous space
- Inputs = indices of the \( n-1 \) previous words

\[
h_j = w_{j-n+1}, \ldots, w_{j-2}, w_{j-1}
\]
Architecture of the CSLM

Neural Network

Discrete representation: indices in wordlist
Continuous representation: P-dimensional vectors
LM probabilities for all words

\[ p_i = P(w_j = i | h_j) \]
\[ p_1 = P(w_j = 1 | h_j) \]
\[ p_N = P(w_j = N | h_j) \]
CSLM - Math

- **Input**: \( n - 1 \) context word \( w_{i-n+1} \ldots w_{i-1} \)
- **Vocabulary**: \( V \) words
- **Projection layer**
  - one-out-of-\( n \) encoding \( \rightarrow \) \( V \) dimensional input vector
  - projection = lookup in table
- **Hidden layer**
  \[
  h_i = \tanh\left( \sum_k w_{ij} p_j + b_i \right)
  \]
- **Output layer**
  \[
  o_i = \sum_j w_{ij} h_j + k_i
  \]
- **Softmax normalization over the full vocabulary**
  \[
  p_i = \frac{\exp(o_i)}{\sum_k \exp(o_k)} \quad \forall i = 1 \ldots V
  \]
CSLM - Training

• Backprop training, cross-entropy error

\[ E = \sum_{i=1}^{N} d_i \log p_i \]

+ weight decay

⇒ NN minimizes perplexity on training data

• continuous word codes are also learned (random initialization)
CSLM - Training

- Backprop training, cross-entropy error

\[ E = \sum_{i=1}^{N} d_i \log p_i \]

+ weight decay

⇒ NN minimizes perplexity on training data

- continuous word codes are also learned (random initialization)
CSLM - Training

- Backprop training, cross-entropy error
  \[ E = \sum_{i=1}^{N} d_i \log p_i \]
  + weight decay
  \[ \Rightarrow \text{NN minimizes perplexity on training data} \]
- continuous word codes are also learned
  (random initialization)
CSLM - Variants

Variants of the basic architecture

- Representation of the output layer (input is no issue)
  → short-lists, classes, SOUL, NCE, etc

- Integration into the SMT system
  → N-best rescoring, integration into decode, ...

- Network architecture
  → MLP, RNN, LSTM, ...
  → shallow or flat networks, how much capacity?

- NN training tricks
  → initialization, learning rate, huge data, ...

- Speed of training and inference
  → a lot of engineering, GPUs

- Providing additional (source) information
  → joint model, translation model, ...
CSLM - Representations of the Output Layer

Short-list

- NN only predicts the $N$ most frequent words (8k-32k)
- Remaining ones are done by standard back-off LM
- One special NN output for the cumulative probability mass

$$P(w_t|h_t) = \begin{cases} P_{NN}(w_t|h_t) & \text{if } w_t \in \text{shortlist} \\ P_{BOF}(w_t|h_t) & \text{else} \end{cases}$$

⇒ Neural network redistributes the probability mass
- Easy to implement and use
- Actually quite fast since we can use the brute force of CPU or GPUs

Short-list

- NN only predicts the $N$ most frequent words (8k-32k)
- Remaining ones are done by standard back-off LM
- One special NN output for the cumulative probability mass

$$P(w_t|h_t) = \begin{cases} P_{NN}(w_t|h_t) & \text{if } w_t \in \text{shortlist} \\ P_{BOF}(w_t|h_t) & \text{else} \end{cases}$$

⇒ Neural network redistributes the probability mass
- Easy to implement and use
- Actually quite fast since we can use the brute force of CPU or GPUs
CSLM - Representations of the Output Layer

Classes

- Well known class decomposition of back-off LMs
- All the words can be handled by the NN
- Not necessarily faster due to many operations on small matrices
- Used in RNN toolkit [Mikolov], OxFLM [this Marathon], etc
- Now also available in the CSLM toolkit
CSLM - Representations of the Output Layer

*Structured Output Neural Network Language Model (SOUL)*, Le et al, ICASSP’11

- Hierarchical decomposition:

  - Probably some experience needed to find the optimal decomposition
CSLM - Representations of the Output Layer

Techniques to avoid the costly softmax normalization at the output layer

Noise constrastive estimation (NCE)

- A fast and simple algorithm for training neural probabilistic language models Minh et Teh, ICML’12
- First used for SMT in: Decoding with Large neural networks improves translation, Vaswani et al, EMNLP’13

Self normalization

- Fast and Robust Neural Network Joint Models for Statistical Machine Translation, Devred et al, ACL’14
- Additional regularizer to learn self normalized outputs
CSLM - Representations of the Output Layer

Techniques to avoid the costly softmax normalization at the output layer

Noise constrastive estimation (NCE)

- A fast and simple algorithm for training neural probabilistic language models Minh et Teh, ICML’12
- First used for SMT in: Decoding with Large neural networks improves translation, Vaswani et al, EMNLP’13

Self normalization

- Fast and Robust Neural Network Joint Models for Statistical Machine Translation, Devred et al, ACL’14
- Additional regularizer to learn self normalized outputs
CSLM - Integration into the SMT System

N-best list rescoring

- First decode with standard back-off LM
- Then rescore with the CSLM (addtl feature)
- Many optimizations are possible

Directly in the decoder

- Quite tricky since many thousands/millions of LM requests are needed for each sentence
- One need to delay the LM requests
- Avoid to perform softmax normalization
- Usually only small networks are possible
- Limited context size (hypothesis recombination)
CSLM - Integration into the SMT System

N-best list rescoring

- First decode with standard back-off LM
- Then rescore with the CSLM (addtl feature)
- Many optimizations are possible

Directly in the decoder

- Quite tricky since many thousands/millions of LM requests are needed for each sentence
- One need to delay the LM requests
- Avoid to perform softmax normalization
- Usually only small networks are possible
- Limited context size (hypothesis recombination)
CSLM - Network Architecture

Feed-Forward

- Initial approach of Bengio et al.
- Easy to implement and optimize
- Still an $n$-gram approach

$\Rightarrow$ Context beyond $n – 1$ words is ignored
Recurrent neural networks

- Use of internal state which summarizes the past
- This hidden layer has connections to the input layer and itself
- First used for LM by Mikolov et al., Interspeech’10
- Theoretically better motivated (no n-gram approximation)
- In practice, RNN tend to forget words too far in the past (vanishing gradient problem)
- RNNs are more tricky to optimize since we need to preserve the sequential order
- Normally only n-best rescoring
- Unclear if they are superior on large tasks
LSTM

- Variant of recurrent NN [Hochreiter and Schmidhuber '97]
- Intended to be better suited for long sequences
- Very successful in online handwritten character recognition
- Recently used in a couple of studies for LM
CSLM - Network Architecture

Large context feed-forward architectures

- It is straightforward to train an CSLM with a large context window
  ⇒ We just need to add additional inputs
- Use of a special word for shorter $n$-grams
- No effect of forgetting quickly the past (vanishing gradient in RNN)
- Training and inference complexity is almost not affected ($n$-best rescoring)
- Surprisingly efficient in practice
CSLM - Network Architecture

Large context feed-forward architectures

- BLEU scores in function of the context size (Ar/En and Ar/Fr on NIST-style task)

⇒ Consistent increase of BLEU score with the context size
- +2 BLEU between 4- and 14-gram for Ar/En
CSLM - Network Architecture

Shallow versus deep architectures

- Often, a straight-forward NN architecture seems to be used (input, projection, one hidden and softmax layer)
- One the other hand, the machine learning community is very actively working on deep architectures, i.e. with multiple hidden layers
- These networks consistently outperformed shallow architectures in many tasks
- However, only very few studies concern deep CSLMs

Network capacity

- Another meta-parameter is the size of the hidden layer(s)
- Practice has shown, that we need more capacity than expected (no overfitting observed on large tasks)
Deep NN for SMT
H. Schwenk

Introduction
CSLM
Architecture
Output layer
Integration
Architecture
Training
Summary

Beyond CSLM
Joint Models
CSTM
Encoder/Decoder

CSLM Toolkit

Shared Task

Conclusion

CSLM - Network Architecture

Shallow versus deep architectures

- Often, a straight-forward NN architecture seems to be used (input, projection, one hidden and softmax layer)
- One the other hand, the machine learning community is very actively working on deep architectures, i.e. with multiple hidden layers
- These networks consistently outperformed shallow architectures in many tasks
- However, only very few studies concern deep CSLMs

Network capacity

- Another meta-parameter is the size of the hidden layer(s)
- Practice has shown, that we need more capacity than expected (no overfitting observed on large tasks)
### Study on NN architecture (BOLT Ar/En)

<table>
<thead>
<tr>
<th>Network architecture</th>
<th>Px</th>
<th>Nbr of parameters</th>
<th>Training time [min]</th>
<th>SMT performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>Back-off 4-gram</td>
<td>76.3</td>
<td>n/a</td>
<td>n/a</td>
<td>26.64 58.38 15.87</td>
</tr>
<tr>
<td>Initial CSLM, 1024-h384</td>
<td>61.0</td>
<td>14.9M</td>
<td>60</td>
<td>27.22 57.94 15.36</td>
</tr>
<tr>
<td><strong>One hidden layer CSLM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>512</td>
<td>62.7</td>
<td>9.4M</td>
<td>26</td>
<td>27.58 57.61 15.01</td>
</tr>
<tr>
<td>1024</td>
<td>59.8</td>
<td>18.8M</td>
<td>46</td>
<td>27.60 57.62 15.01</td>
</tr>
<tr>
<td>2048</td>
<td>60.0</td>
<td>37.5M</td>
<td>80</td>
<td>27.74 57.92 15.09</td>
</tr>
<tr>
<td><strong>Two hidden layer CSLM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1024-1024</td>
<td>57.5</td>
<td>19.8M</td>
<td>46</td>
<td>27.85 57.49 14.82</td>
</tr>
<tr>
<td>1536-1536</td>
<td>57.3</td>
<td>30.5M</td>
<td>58</td>
<td>27.95 57.67 14.86</td>
</tr>
<tr>
<td>3072-1536</td>
<td>57.6</td>
<td>35.8M</td>
<td>68</td>
<td>27.77 57.51 14.87</td>
</tr>
<tr>
<td><strong>Three hidden layer CSLM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>768-768-768</td>
<td>57.4</td>
<td>15.3M</td>
<td>67</td>
<td>27.96 57.27 14.65</td>
</tr>
<tr>
<td>1536-1024-768</td>
<td>57.3</td>
<td>17.9M</td>
<td>74</td>
<td>27.96 57.57 14.80</td>
</tr>
<tr>
<td>1536-1536-1536</td>
<td>56.9</td>
<td>32.8M</td>
<td>63</td>
<td>28.00 57.39 14.69</td>
</tr>
<tr>
<td><strong>Four hidden layer CSLM</strong></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1536-1024-768-768</td>
<td>57.5</td>
<td>18.5M</td>
<td>50</td>
<td>27.95 57.29 14.67</td>
</tr>
<tr>
<td>1536-1536-1024-1024</td>
<td>57.0</td>
<td>24.7M</td>
<td>83</td>
<td>27.78 57.43 14.82</td>
</tr>
<tr>
<td>2048-2048-1536-1536</td>
<td>57.2</td>
<td>38.8M</td>
<td>78</td>
<td>27.86 57.77 14.96</td>
</tr>
<tr>
<td>3072-2048-1536-1536</td>
<td>57.0</td>
<td>42.9M</td>
<td>83</td>
<td>27.78 57.51 14.86</td>
</tr>
</tbody>
</table>

- NN with three hidden layers (total of 7 layers) perform best
- Shallow networks, **with more capacity**, perform worse
- Same tendency observed on many other tasks
- Three hidden layers of about 1024 units seem to be **universal**
Neural Network Training

Optimizing meta-parameters

- There are a lot of buttons to adjust when using NN:
  - learning rate, procedure to decrease it
  - how to initialize the weights
  - regularization, noise injection
  - drop-out, rectifiers, ...

- Many papers were published in the machine learning community on these issues

Application to CSLM

- Training an CSLM on huge corpora can take hours, days, ...
- Can we afford to optimize the meta-parameters?
- Use at least recommended settings (which should be documented in the publications)
Neural Network Training

Optimizing meta-parameters

- There are a lot of buttons to adjust when using NN:
  - learning rate, procedure to decrease it
  - how to initialize the weights
  - regularization, noise injection
  - drop-out, rectifiers, ...

- Many papers were published in the machine learning community on these issues

Application to CSLM

- Training an CSLM on huge corpora can take hours, days, ...
- Can we afford to optimize the meta-parameters?
- Use at least recommended settings (which should be documented in the publications)
Speed of Training and Inference

Speed issues

- An approach to LM should be able to scale to larger corpora (billions of words)
- Quite some engineering is needed to speed-up CSLMs
  - block mode, multi-threading, data selection
  - cache algorithms and redundancy during inference
- GPU cards are a quite cheap and efficient alternative
- Neural network training needs little main memory

⇒ An CSLM can be trained on a **gamer PC**

CSLM toolkit

- network 320-1024-1024-1024-16384: 1h50 for one epoch of 40M examples on Tesla K40
- network 320-1024-1024-1024-32768: 2h30
Speed of Training and Inference

Speed issues

- An approach to LM should be able to scale to larger corpora (billions of words)
- Quite some engineering is needed to speed-up CSLMs
  - block mode, multi-threading, data selection
  - cache algorithms and redundancy during inference
- GPU cards are a quite cheap and efficient alternative
- Neural network training needs little main memory

⇒ An CSLM can be trained on a gamer PC

CSLM toolkit

- network 320-1024-1024-1024-16384: 1h50 for one epoch of 40M examples on Tesla K40
- network 320-1024-1024-1024-32768: 2h30
Personal analysis

- Feed-forward or recurrent?
  → Feed-forward with large context window since easier to train/use

- Rescoring or decoding?
  → Integration into decoding may help, but we quite likely loose the benefit of long contexts
CSLM - Summary

Personal analysis

• Feed-forward or recurrent?
  → Feed-forward with large context window since easier to train/use

• Rescoring or decoding?
  → Integration into decoding may help, but we quite likely loose the benefit of long contexts
CSLM - Summary

Personal analysis

• Feed-forward or recurrent?
  → Feed-forward with large context window since easier to train/use

• Rescoring or decoding?
  → Integration into decoding may help, but we quite likely loose the benefit of long contexts
CSLM - Summary

Personal analysis

- Feed-forward or recurrent?
  → Feed-forward with large context window since easier to train/use

- Rescoring or decoding?
  → Integration into decoding may help, but we quite likely loose the benefit of long contexts
CSLM - Summary

My best practice

- Train on GPUs using all the available data
- Use Moore&Lewis data selection to focus on most relevant examples
- Sample in several corpora using appropriate coefficients
- Use deep networks with enough capacity
- Carefully optimize the meta-parameters
- N-best rescoring with long context feed-forward architectures

⇒ Improvements of 2-3 points BLEU in many tasks

- You can use the open-source CSLM toolkit for this
Beyond Neural Language Models

Can we also use neural networks for translation models?

Initial straight-forward attempt

- Consider n-gram tuple translation models
- Translation model = language model over tuple vocabulary

⇒ It’s straight-forward to apply an CSLM

- Smooth bilingual n-gram translation, Schwenk et al., EMNLP’07
Beyond Neural Language Models

Can we also use neural networks for translation models?

Initial straight-forward attempt

- Consider n-gram tuple translation models
- Translation model = language model over tuple vocabulary
  ⇒ It’s straight-forward to apply an CSLM
- *Smooth bilingual n-gram translation*, Schwenk et al., EMNLP’07
Beyond Neural Language Models

Extensions

- Nothing prevents us from using different input and output vocabularies
- It is also possible to add additional information at the input layer
  - context or topic information
  - source text information

⇒ Joint models
Beyond Neural Language Models

Extensions

- Nothing prevents us from using different input and output vocabularies
- It is also possible to add additional information at the input layer
  - context or topic information
  - source text information
  ⇒ Joint models
Joint Neural Language Models

SOUL neural translation model

- *Continuous Space Translation Models with Neural Networks*, Lee et al, NAACL’12
- Factorize translation probability
- Predict the next target word given a sliding window of the preceding source and target words
- Used in $n$-best rescoring

Joint language and Translation Modeling with Recurrent NN, Auli et al, EMNLP’13

- Augment the RNNLM with additional inputs from the source language
- Lattice rescoring using the NN hidden state as context vector
Joint Neural Language Models

SOUL neural translation model

- *Continuous Space Translation Models with Neural Networks*, Lee et al, NAACL’12
- Factorize translation probability
- Predict the next target word given a sliding window of the preceding source and target words
- Used in $n$-best rescoring

*Joint language and Translation Modeling with Recurrent NN*, Auli et al, EMNLP’13

- Augment the RNNLM with additional inputs from the source language
- Lattice rescoring using the NN hidden state as context vector
Joint Neural Language Models

Recurrent Continuous Translation Models, Kalchbrenner & Blunsom, EMNLP’13

- Convolutional RNN
- Rescoring: same result than with standard 14 feature functions

Fast and Robust Neural Network Joint Models for Statistical Machine Translation, Devred et al, ACL’14

- Use of source information
- Various speed-up tricks
- Integration into decoder
- Very significant improvements with combination of 6 models

⇒ Extension: Mapping sequences to sequences
Estimating phrase translation probabilities

Continuous space translation models for PBSMT, Schwenk, Coling’12

\[
P(\bar{t} | \bar{s}) = P(t_1 \ldots t_p | s_1 \ldots s_q) \\
= P(t_1 | t_2, \ldots, t_p, s_1, \ldots, s_q) \times P(t_2, \ldots, t_p | s_1, \ldots, s_q) \\
= P(t_1 | t_2, \ldots, t_p, s_1, \ldots, s_q) \times P(t_2 | t_3, \ldots, t_p, s_1, \ldots, s_q) \\
\times P(t_3, \ldots, t_p | s_1, \ldots, s_q) \\
= \prod_{k=1}^{p} P(t_k | t_{k+1}, \ldots, t_p, s_1, \ldots, s_q) \\
\approx \prod_{k=1}^{p} P(t_k | s_1, \ldots, s_q) = \prod_{k=1}^{p} P(t_k | \bar{s})
\]

We drop the dependency on the previous target words
Estimating phrase translation probabilities

Different neural network architectures for a CSTM

- **Left**: simple extension of the CSLM.
- **Middle**: addition of a common hidden layer in order to introduce a dependence between the target words.
- **Right**: hierarchical dependence
- Initial experiments on IWSLT task showed small improvements
- Now implemented on multiple GPU cards in CSLM toolkit → we can scale up to large tasks
Encoder/Decoder Framework

Context

- Map a sequence of input words to a sequence of outputs
- The length of both sequences may vary

Encoder

- Map the variable length input sequence to a fixed size high dimensional vector

Decoder

- Create a sequence of output symbols from the high dimensional representation

Use a (deep) neural network for the encoder/decoder (CSTM uses MLPs at the phrase level)
Encoder/Decoder Framework

Context

- Map a sequence of input words to a sequence of outputs
- The length of both sequences may vary

Encoder

- Map the variable length input sequence to a fixed size high dimensional vector

Decoder

- Create a sequence of output symbols from the high dimensional representation

Use a (deep) neural network for the encoder/decoder (CSTM uses MLPs at the phrase level)
Encoder/Decoder Framework

Context

- Map a sequence of input words to a sequence of outputs
- The length of both sequences may vary

Encoder

- Map the variable length input sequence to a fixed size high dimensional vector

Decoder

- Create a sequence of output symbols from the high dimensional representation

Use a (deep) neural network for the encoder/decoder (CSTM uses MLPs at the phrase level)
Encoder/Decoder Framework

Context

- Map a sequence of input words to a sequence of outputs
- The length of both sequences may vary

Encoder

- Map the variable length input sequence to a fixed size high dimensional vector

Decoder

- Create a sequence of output symbols from the high dimensional representation

Use a (deep) neural network for the encoder/decoder (CSTM uses MLPs at the phrase level)
Estimating phrase translation probabilities


- Use of recurrent encoder and decoder (modified LSTM)
- Rescoring of $n$-best list
- WMT’14 En/Fr task

Recent work

- A couple of groups are working on the translation of whole sentences instead of phrases with neural networks
- Some approaches start to match classical systems
- Translation quality seems to decrease with the length of the sentences
Features

- Written in C++, LGPL
- Flexible network architecture (deep)
- Very efficient on CPU and GPU
- Support for large corpora (resampling, etc)
- Continuous space language and translation models
- N-best rescoring (external tool or in Moses)
- Support for many data formats
- Successfully used in industrial applications

http://www-lium.univ-lemans.fr/~cslm
CSLM Toolkit

New release (Sept 2014)

- Classes at output layer (thanks to P. Lamblin)
- Speed improvements
- Support of Google $n$-grams
- Training on multiple GPU cards in parallel
- More learning rate options, ...

Planed

- Recurrent networks (LSTM)
- Sentence translation models
- ...
- External contributions are welcome
CSLM Toolkit

New release (Sept 2014)

- Classes at output layer (thanks to P. Lamblin)
- Speed improvements
- Support of Google $n$-grams
- Training on multiple GPU cards in parallel
- More learning rate options, ...

Planed

- Recurrent networks (LSTM)
- Sentence translation models
- ...
- External contributions are welcome
CSLM Toolkit

New release (Sept 2014)

- Classes at output layer (thanks to P. Lamblin)
- Speed improvements
- Support of Google $n$-grams
- Training on multiple GPU cards in parallel
- More learning rate options, ...

Planed

- Recurrent networks (LSTM)
- Sentence translation models
- ...
- External contributions are welcome
Shared Task for Research on NN for SMT

Context

- Many papers are published and it’s difficult to compare the various approaches in a fair way (undocumented tricks)
- We provide a complete Moses En/Fr SMT system trained on WMT’14 data
- Large scale task, with common bitext and LM data (data selection already applied)
- $n$-best lists for rescoring are provided
- Plan: common MERT/MIRA tuning
- Everybody can try his own approach and enter the results into a public table

http://www-lium.univ-lemans.fr/~schwenk/nnmt-shared-task
Conclusion

- Exploding interest in neural networks for SMT
  - Continuous space LMs are now well established: they consistently and significantly improve translation performance in many studies
  - It is important to use all the tricks of the NN community
  - We need to scale up!
    → Buy a couple of GPU cards
  - Many directions to extend those ideas to the translation model, complete sentences, ... 
  - Challenges:
    - should we integrate NN into the current framework or aim for fully neural solutions?
    - how to efficiently encode/decode long sequences?
  - I expect very promising approaches within the next years ...
Conclusion

- Exploding interest in neural networks for SMT
- Continuous space LMs are now well established: they consistently and significantly improve translation performance in many studies
  - It is important to use all the tricks of the NN community
  - We need to scale up!
  - Buy a couple of GPU cards
- Many directions to extend those ideas to the translation model, complete sentences, . . .
- Challenges:
  - should we integrate NN into the current framework or aim for fully neural solutions?
  - how to efficiently encode/decode long sequences?
- I expect very promising approaches within the next years...
Conclusion

- Exploding interest in neural networks for SMT
- Continuous space LMs are now well established: they consistently and significantly improve translation performance in many studies
- It is important to use all the tricks of the NN community
  - We need to scale up!
    - Buy a couple of GPU cards
- Many directions to extend those ideas to the translation model, complete sentences, . . .
- Challenges:
  - should we integrate NN into the current framework or aim for fully neural solutions?
  - how to efficiently encode/decode long sequences?
- I expect very promising approaches within the next years ...
Conclusion

- Exploding interest in neural networks for SMT
- Continuous space LMs are now well established: they consistently and significantly improve translation performance in many studies
- It is important to use all the tricks of the NN community
- We need to scale up!
  → Buy a couple of GPU cards
- Many directions to extend those ideas to the translation model, complete sentences, ...
- Challenges:
  - should we integrate NN into the current framework or aim for fully neural solutions?
  - how to efficiently encode/decode long sequences?
- I expect very promising approaches within the next years...
Conclusion

- Exploding interest in neural networks for SMT
- Continuous space LMs are now well established: they consistently and significantly improve translation performance in many studies
- It is important to use all the tricks of the NN community
- We need to scale up!
  - Buy a couple of GPU cards
- Many directions to extend those ideas to the translation model, complete sentences, . . .
- Challenges:
  - should we integrate NN into the current framework or aim for fully neural solutions?
  - how to efficiently encode/decode long sequences?
- I expect very promising approaches within the next years ...
• Exploding interest in neural networks for SMT
• Continuous space LMs are now well established: they consistently and significantly improve translation performance in many studies
• It is important to use all the tricks of the NN community
• We need to scale up!
  → Buy a couple of GPU cards
• Many directions to extend those ideas to the translation model, complete sentences, ... 
• Challenges:
  • should we integrate NN into the current framework or aim for fully neural solutions?
  • how to efficiently encode/decode long sequences?
• I expect very promising approaches within the next years ...
Conclusion

• Exploding interest in neural networks for SMT
• Continuous space LMs are now well established: they consistently and significantly improve translation performance in many studies
• It is important to use all the tricks of the NN comunity
• We need to scale up!
  → Buy a couple of GPU cards
• Many directions to extend those ideas to the translation model, complete sentences, ...
• Challenges:
  • should we integrate NN into the current framework or aim for fully neural solutions?
  • how to efficiently encode/decode long sequences?
• I expect very promising approaches within the next years...
Conclusion

- Exploding interest in neural networks for SMT
- Continuous space LMs are now well established: they consistently and significantly improve translation performance in many studies
- It is important to use all the tricks of the NN community
- We need to scale up!
  - Buy a couple of GPU cards
- Many directions to extend those ideas to the translation model, complete sentences, ... 
- Challenges:
  - should we integrate NN into the current framework or aim for fully neural solutions?
  - how to efficiently encode/decode long sequences?
- I expect very promising approaches within the next years ...
Conclusion

Cooperations

- I would be happy to collaborate with other groups
- We also have openings at all levels (internship, PhD, postdoc, senior, ...)