Cold Fusion: Training Seq2Seq Models together with Language Models

Philipp Meier

Department of Computational Linguistics, Heidelberg
<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Related Work</td>
</tr>
<tr>
<td>2</td>
<td>Cold Fusion</td>
</tr>
<tr>
<td>3</td>
<td>Experiments</td>
</tr>
<tr>
<td>4</td>
<td>Comparison</td>
</tr>
<tr>
<td>5</td>
<td>Questions</td>
</tr>
</tbody>
</table>
Related Work
Motivation

• High quality parallel corpora is essential for good results in various NLP tasks
• Lack of such in low resource language pairs (Turkish-English) or tasks with domain restrictions (SMS chats)
• Integrate a language model in a sequence to sequence model to give ‘hints’ and improve fluency
• Multiple methods to integrate a language model
• Assumption: NMT model and RNN language model have been pre-trained separately before integration
Translation Model Architecture

- Model after Bahdanau et al., 2014
- Encoder consists of forward and backward RNNs
- Decoder: Single Layer RNN with soft alignment mechanism
- Decoders hidden states: \( s_{TM}^t = f_r(s_{TM}^{t-1}, y_{t-1}, c_t) \)
- Deep output layer to compute the conditional distribution:
  \[
  p(y_t|y_{<t}, x) \propto \exp(y_t^\top (W_o f_o(s_{TM}^t, y_{t-1}, c_t) + b_o))
  \]
- Training objective: Maximize the conditional log-likelihood of the bilingual training corpus:
  \[
  \max_{\theta} \frac{1}{N} \sum_{n=1}^{N} \log p_{\theta}(y^{(n)}|x^{(n)})
  \]
Shallow fusion I

• Integration of recurrent neural network language model in the decoder
• Translation Model (TM) proposes set of candidate words at each time step $t$
• Candidates are scored according to the sum of the scores given by the Language Model and Translation Model.
• At each $t$, the model computes the score of every possible next word for each hypothesis of all hypotheses $\{y^{(i)}_{\leq t-1}\}$
• Score = Score of the hypothesis + Score given by NMT to the next word
• Sort new hypotheses according to their scores, select top K as candidates $\{\hat{y}^{(i)}_{\leq t}\}_{i=1,...K}$
Rescore hypotheses with the weighted sum of the scores by the NMT and the LM

Only recompute the score of the 'new word' at the end of each candidate hypothesis

\[
\log p(y_t = k) = \log p_{TM}(y_t = k) + \beta \log p_{LM}(y_t = k)
\]

Figure 1: Shallow Fusion [5]
Deep fusion I

- Stronger connection between decoder of the NMT and language model
- Concatenation of their hidden states
- Finetuned to use both hidden states when computing the output probability of the next word

\[ p(y_t|y_{<t}, x) \propto \exp(y_t^T (W_0 f_0(s_t^{LM}, s_t^{TM}, y_{t-1}, c_t) + b_0)) \]
Deep fusion II

- Balancing of LM and TM via gating mechanism
- \( g_t = \sigma \left( v_g^T s_t^{LM} + b_g \right) \)
- \( s_t^{DF} = [c_t, s_t^{TM}; g_t s_t^{LM}] \)
- \( y_t = \text{softmax}(DNN(s_t^{DF})) \)

**Figure 2:** Deep Fusion [5]
Semi-supervised Learning in Seq2Seq Models

- Backtranslation
  - Combine monolingual training data with automatic backtranslation
  - Backtranslate monolingual target text into source language
  - Increase the parallel training corpus by translating the unlabeled target domain text

- Unsupervised pre-training
  - Weights of encoder and decoder are initialized with weights of two pre-trained language models
  - Finetuned with labeled data
  - Idea: Find a good initialization point
Motivation for Cold Fusion

- Deep Fusion has drawbacks
- Translation and Language model are trained separately, so the decoder of the TM learns from training data labels
- Waste of decoder capacity for redundant information
- Bias towards the training labels of the parallel corpus which limits deployability
Cold Fusion
• Sequence-to-Sequence Models
  • Encoder maps input sequence $x = (x_1, ..., x_T)$ to representation $h$
  • Decoder which generates an output sequence $y = (y_1, ...y_K)$ using $h$

• Inference and Language Model Integration
  • Compute the most likely sequence during inference
    • $\hat{y} = \arg\max_y \log p(y|x)$
  • Use of left-to-right beam search because argmax is intractable
  • Integration through change of the inference task to:
    • $\hat{y} = \arg\max_y \log p(y|x) + \lambda \log p_{LM}(y)$
  • Shallow Fusion uses this way of integration
Cold Fusion Properties

- Seq2Seq model is trained from scratch together with a pre-trained language model → Early Training Integration
- Cold Fusion uses a char based RNN as language model
- Both hidden states, $s_t$ and $s^{LM}_t$ can be used for the gate computation
- Use of fine-grained gating mechanism, different gate value for each hidden node
- Use of language models probability instead of hidden state for better generalization
Cold Fusion Layer

- \( h_t^{LM} = DNN(\ell_t^{LM}) \)
- \( g_t = \sigma(W[s_t; h_t^{LM}] + b) \)
- \( s_t^{CF} = [s_t; g_t \circ h_t^{LM}] \)
- \( r_t^{CF} = DNN(s_t^{CF}) \)
- \( \hat{P}(y_t|X, y_{<t}) = \text{softmax}(r_t^{CF}) \)
• In Yang et al [8]: Mechanism to combine word-level and character-level representation
• In Cold Fusion:
  • \( g_t = \sigma(W_g[s^{ED}_t; s^{LM}_t] + b_g) \)
  • \( s^{CF}_t = [s^{ED}_t; g_t \circ s^{LM}_t] \)
• In Deep Fusion:
  • \( g_t = \sigma(v^T_g d^{LM}_t + b_g) \)
  • \( d^{DF}_t = [c_t; d_t; g_t d^{LM}_t] \)
• Vector values are used instead of scalar values
Deep & Cold Fusion Architecture

Figure 3: Deep Fusion

Figure 4: Cold Fusion
Experiments
Datasets

- Evaluation on the Speech Recognition Task
- Source: based on search queries, 411,000 utterances, 650 hours of audio
- Target: based on movie transcripts, 345,000 utterances, 676 hours of audio
- Used Amazon Mechanical Turk to provide audio recordings
- Held out of 2048 from each domain for evaluation
**Architectures**

- **Language Model**
  - Trained on 25 million words
  - Three layers of GRU, hidden state dimension of 1024
  - Minimize Cross Entropy of predicting the next character given the previous characters
  - Adam optimizer with a batch size of 512

- **Acoustic Models**
  - Seq2Seq model with soft attention
  - Encoder: 6 BLSTM with a dimension of 480
  - Decoder: Single Layer with 960 dimensional GRU with hybrid attention
## Language Model Perplexity on Dev set

<table>
<thead>
<tr>
<th>Model</th>
<th>Domain</th>
<th>Word Count</th>
<th>Perplexity Source</th>
<th>Perplexity Target</th>
</tr>
</thead>
<tbody>
<tr>
<td>GRU (3*512)</td>
<td>Source</td>
<td>5.73M</td>
<td>2.670</td>
<td>4.463</td>
</tr>
<tr>
<td>GRU (3*512)</td>
<td>Target</td>
<td>5.46M</td>
<td>3.717</td>
<td>2.794</td>
</tr>
<tr>
<td>GRU (3*1024)</td>
<td>Full</td>
<td>25.16M</td>
<td><strong>2.491</strong></td>
<td><strong>2.325</strong></td>
</tr>
</tbody>
</table>

**Table 1:** Perplexities on the Development set [6]
Figure 5: Cross-entropy loss on dev set, baseline (orange), proposed model (purple) [6]
## Speech Recognition Results

<table>
<thead>
<tr>
<th>Model</th>
<th>Train Domain</th>
<th>Test on Source</th>
<th>Test on Target</th>
<th>Domain Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>CER</td>
<td>WER</td>
<td>CER</td>
</tr>
<tr>
<td>BL Attention Model</td>
<td>Source</td>
<td>7.54%</td>
<td>14.68%</td>
<td>23.02%</td>
</tr>
<tr>
<td>BL Attention Model</td>
<td>Target</td>
<td>8.84%</td>
<td>17.61%</td>
<td></td>
</tr>
<tr>
<td>Baseline + Deep Fusion</td>
<td>Source</td>
<td>7.64%</td>
<td>13.92%</td>
<td>22.14%</td>
</tr>
<tr>
<td>+ sAMingate</td>
<td>Source</td>
<td>7.61%</td>
<td>13.92%</td>
<td>21.07%</td>
</tr>
<tr>
<td>+ Fine Grained Gating</td>
<td>Source</td>
<td>7.47%</td>
<td>13.61%</td>
<td>20.29%</td>
</tr>
<tr>
<td>+ ReLU Layer</td>
<td>Source</td>
<td>7.50%</td>
<td>13.54%</td>
<td>21.18%</td>
</tr>
<tr>
<td>Baseline + Cold Fusion</td>
<td>Source</td>
<td>7.25%</td>
<td>13.88%</td>
<td>15.63%</td>
</tr>
<tr>
<td>+ sAMingate</td>
<td>Source</td>
<td>6.14%</td>
<td>12.08%</td>
<td>14.79%</td>
</tr>
<tr>
<td>+ Fine-Grained Gating</td>
<td>Source</td>
<td>5.82%</td>
<td>11.52%</td>
<td>14.89%</td>
</tr>
<tr>
<td>+ ReLU Layer</td>
<td>Source</td>
<td>5.64%</td>
<td>11.87%</td>
<td>13.72%</td>
</tr>
<tr>
<td>+ Probability Projection</td>
<td>Source</td>
<td>5.64%</td>
<td>11.87%</td>
<td>13.72%</td>
</tr>
</tbody>
</table>

**Table 2:** Speech recognition results [6]
### Table 3: Speech Recognition Results

<table>
<thead>
<tr>
<th>Model</th>
<th>LibriSpeech Test-Clean</th>
<th>LibriSpeech Test-Other</th>
<th>Target Domain Test</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CER</td>
<td>WER</td>
<td>CER</td>
</tr>
<tr>
<td>Wav2Letter + Shallow Fusion</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFCC</td>
<td>6.9%</td>
<td>7.2%</td>
<td></td>
</tr>
<tr>
<td>Power Spectrum</td>
<td>9.1%</td>
<td>9.4%</td>
<td></td>
</tr>
<tr>
<td>Raw Wave</td>
<td>10.6%</td>
<td>10.1%</td>
<td></td>
</tr>
<tr>
<td>Baseline Attention</td>
<td>4.47%</td>
<td>8.94%</td>
<td>11.57%</td>
</tr>
<tr>
<td>Baseline + Deep Fusion</td>
<td>5.01%</td>
<td>9.17%</td>
<td>12.67%</td>
</tr>
<tr>
<td>Baseline + Cold Fusion</td>
<td>3.87%</td>
<td>7.47%</td>
<td>9.28%</td>
</tr>
</tbody>
</table>

Results from models trained on the publicly available Librispeech (reading task) data. Results from the Wav2Letter model [4] are presented for reference. MFCC, Power Spectrum and Raw Wave are names for different features.
<table>
<thead>
<tr>
<th>Model</th>
<th>Decoder size</th>
<th>Source CER</th>
<th>WER</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attention</td>
<td>64</td>
<td>16.33%</td>
<td>33.98%</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>11.14%</td>
<td>24.35%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>8.89%</td>
<td>18.74%</td>
</tr>
<tr>
<td></td>
<td>960</td>
<td>7.54%</td>
<td>14.68%</td>
</tr>
<tr>
<td>Cold Fusion</td>
<td>64</td>
<td>9.47%</td>
<td>17.42%</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>7.96%</td>
<td>15.15%</td>
</tr>
<tr>
<td></td>
<td>256</td>
<td>6.71%</td>
<td>13.19%</td>
</tr>
<tr>
<td></td>
<td>960</td>
<td>5.82%</td>
<td>11.52%</td>
</tr>
</tbody>
</table>

**Table 4:** Effect of decoder dimension [6]
## Fine-tuning for Domain Adaption

<table>
<thead>
<tr>
<th>Model</th>
<th>Target Data</th>
<th>Target CER</th>
<th>Target WER</th>
<th>Domain Gap</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cold Fusion</td>
<td>0%</td>
<td>13.72%</td>
<td>27.50%</td>
<td>38.17%</td>
</tr>
<tr>
<td>Cold Fusion + fine-tuning</td>
<td>0.6%</td>
<td>11.98%</td>
<td>23.13%</td>
<td>21.30%</td>
</tr>
<tr>
<td></td>
<td>1.2%</td>
<td>11.62%</td>
<td>22.40%</td>
<td>18.49%</td>
</tr>
<tr>
<td></td>
<td>2.4%</td>
<td>10.79%</td>
<td>21.05%</td>
<td>13.28%</td>
</tr>
<tr>
<td></td>
<td>4.8%</td>
<td>10.46%</td>
<td>20.46%</td>
<td>11.00%</td>
</tr>
<tr>
<td></td>
<td>9.5%</td>
<td>10.11%</td>
<td>19.68%</td>
<td>7.99%</td>
</tr>
<tr>
<td>Attention*</td>
<td>100%</td>
<td>8.84%</td>
<td>17.61%</td>
<td>0.00%</td>
</tr>
</tbody>
</table>

*Table 5:* Fine tuning of the acoustic model [6]
Comparison
### Table 6: Comparison between fusion methods

<table>
<thead>
<tr>
<th>Shallow Fusion</th>
<th>Deep Fusion</th>
<th>Cold Fusion</th>
</tr>
</thead>
<tbody>
<tr>
<td>late LM integration</td>
<td>early LM integration</td>
<td>early LM integration</td>
</tr>
<tr>
<td>Separated until score computation</td>
<td>LM directly integrated</td>
<td>LM directly integrated</td>
</tr>
<tr>
<td>late training integration</td>
<td>late training integration</td>
<td>early training integration</td>
</tr>
</tbody>
</table>

Comparison
Architectures & Data

LAS Model

- Encoder: 4-layer pyramidal bidirectional LSTM network, 256 hidden units in each direction of the layer
- Encoder for GVS: 5 unidirectional LSTM layers of 1400 hidden units each
- Decoder: Single Layer unidirectional LSTM with 256 hidden units
- Decoder for GVS: 2 unidirectional LSTM layers of 1024 hidden units each
## Table 7: Word error rates (WER) on Eval2000 [7] for the baseline model and fusion approaches, SWB = switchboard, CH = CallHome, Full = Eval2000

<table>
<thead>
<tr>
<th>Model</th>
<th>Switchboard</th>
<th>Call Home</th>
<th>Full</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAS</td>
<td>17.1</td>
<td>27.9</td>
<td>22.6</td>
</tr>
<tr>
<td>Shallow Fusion</td>
<td>15.6</td>
<td>26.6</td>
<td>21.1</td>
</tr>
<tr>
<td>Deep Fusion</td>
<td>16.3</td>
<td>27.2</td>
<td>21.7</td>
</tr>
<tr>
<td>Cold Fusion</td>
<td>16.3</td>
<td>27.3</td>
<td>21.8</td>
</tr>
</tbody>
</table>
### Table 8: WER (%) on Google voice search (VS14K) and dictation data sets (D15K) for the baseline and fusion approaches. VS14K: 14 000 voice search utterances, D15K: 15 000 dictation utterances [7]

<table>
<thead>
<tr>
<th>Model</th>
<th>VS14K</th>
<th>D15K</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAS</td>
<td>5.6</td>
<td>4.0</td>
</tr>
<tr>
<td>Shallow Fusion</td>
<td>5.3</td>
<td>3.7</td>
</tr>
<tr>
<td>Deep Fusion</td>
<td>5.5</td>
<td>4.1</td>
</tr>
<tr>
<td>Cold Fusion</td>
<td>5.3</td>
<td>3.9</td>
</tr>
</tbody>
</table>
Table 9: Word error rates for rescoring in Google data sets [7], Second pass rescoring with a large, production-scale LM [2]

<table>
<thead>
<tr>
<th>Model</th>
<th>VS14K(oracle)</th>
<th>D15K(oracle)</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAS</td>
<td>5.4 (2.2)</td>
<td>3.9 (1.5)</td>
</tr>
<tr>
<td>Shallow Fusion</td>
<td>5.3 (2.4)</td>
<td>3.7 (1.6)</td>
</tr>
<tr>
<td>Deep Fusion</td>
<td>5.4 (2.0)</td>
<td>4.0 (1.5)</td>
</tr>
<tr>
<td>Cold Fusion</td>
<td>5.0 (1.78)</td>
<td>3.8 (1.2)</td>
</tr>
</tbody>
</table>
Findings

Deep Fusion does not scale well with data → No gain over the baseline for large scale Google sets

Shallow Fusion performs quite well

Cold Fusion has strengths on the oracle task
Questions
• What do the authors mean in 3.3: "LM state is not invariant to the permutation of state hidden nodes"?
• What do the authors mean by: "Since logits can have arbitrary offsets, the maximum value is subtracted off before feeding into the layer" (p.3, lower right)?
• Is it possible (or even necessary) to include multiple language models, for example for translational tasks?
What is ’Adam’ and how does it work?

- Optimization Method, derived from ’adaptive moment estimation.’
- Computes a learning rate per parameter, instead of a strict learning rate like in SGD
- Compute decaying averages of past and past squared gradients $m_t$ and $v_t$
  
- $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$
- $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$
  
- To counteract biases a bias-correct fist and second moment estimates are computed
  
- $\hat{m}_t = \frac{m_t}{1 - \beta_1^t}; \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$
  
- Update rule: $w_t = w_{t-1} - \eta \frac{\hat{m}_t}{\sqrt{\hat{v}_t + \epsilon}}$ [1]

- $w = \text{weights}, \eta = \text{stepsize}$
Questions

• Why this architecture is called "Cold Fusion"
  • Due to the early training integration of the Language Model.

• Why not integrate a language model on the encoder side as well?

• Also, the language models are different: The one used in the Cold Fusion paper seems to be purely character based, whereas the one in the Deep Fusion paper (Gulcehre et al.) is also character based but constructed from a word-based language model. I am wondering whether the purely character based one is able to adequately represent phrase-level structures such as grammar. Besides, are whitespaces even part of the "raw" ASR output? So are the predictions listed in Table 1 the actual outputs or are they tokenised?
Lastly, one question raised towards the end of the last session was whether there is an equivalent to residual connections for RNN-like models that mitigates the problem of vanishing gradients. My first intention was that Bahdanau-Attention achieves something like this by making constant the “distance” in terms of weights and nonlinearities from each input to each output. If this is not correct, I would of course receive thankfully being corrected.
• In 2.3, why do the authors state that during warmstarting “… training on the parallel corpus could end up effectively erasing the knowledge of the language models.”?
• In 3.2, how are the “different gate value for each hidden node” realised?
• Are the changes in model architecture from deep fusion to cold fusion necessary to train the seq2seq model alongside with the Language model. Why can’t we do that with the deep fusion architecture?
• This is a rather general question: I don’t see why you would use specific domains (such as legal or medical documents (example in Sec. 1), search queries or movie transcripts (evaluation, Sec. 4.1)) for training in the first place. If you are going to use different domains and/or try to adapt to new domains anyway, doesn’t it make more sense to train on broad corpora directly, so that they contain as many domains as possible in one go?

• “Since logits can have arbitrary offsets, the maximum value is subtracted off before feeding into the layer.” (Sec. 3) - What is the maximum value in this context, and why would it be subtracted off? Can we see this anywhere in Formula 4?
• In the third step, the language model’s hidden state is replaced with the language model probability, does it mean that in case of RNN, each cell receives the time step input and the prediction of the previous time step instead of the hidden state of the previous time step?

• Deep Fusion was also intended to help in cases of low resource language pairs. Though Cold Fusion can be easily used to transfer a model to a different domain, what about its usefulness in such a case? Maybe I’ve managed to miss it, but only the smaller training time of the decoder seems to be mentioned and the little data used to fine-tune to a different domain.
• The DNN layer of Cold Fusion is a single affine layer with ReLU activation and then softmax. Any idea/intuition why this works better than anything else?
  • Unfortunately, this point is not further mentioned in the paper
  • Maybe to explicitly lead the softmax function?
· Can you briefly explain the 'hybrid attention' mechanism? I didn’t really understand how it works and what the difference to 'normal' attention is.
  · Extension of the attention mechanism with speech recognition features
  · Combines content and location information to select the next candidate for decoding
Figure 1: Two steps of the proposed attention-based recurrent sequence generator (ARSG) with a hybrid attention mechanism (computing $\alpha$), based on both content ($h$) and location (previous $\alpha$) information. The dotted lines correspond to Eq. (1), thick solid lines to Eq. (2) and dashed lines to Eqs. (3)–(4).

Figure 6: Hybrid Attention [3]

- 1) $\alpha = \text{Attend}(s_{i-1}, \alpha_{i-1}, h)$
- 2) $g_i = \sum_{j=1}^{L} i_j, h_j$
- 3) $y \sim \text{Generate}(s_{i-1}, g_i)$
- 4) $s_i = \text{Reccurency}(s_{i-1}, g_i, y_i)$
In section 3, in the third bullet point, they say they don’t use the hidden states but the language model probability. And I assume they mean the logits (offset by the largest logit value?). Why not use the LM’s softmax output? In my intuition, that would provide a probability representation that is more constant across language models.

In section 4.1, they explain that they collect their own dataset. Why do they do that in the first place? Why not use an already existing dataset?
Questions

• How is the computation of the Domain Gap performed?
• When showing the impressive results for the domain Adaptation, the LM is trained on full domain. In the scenario were the LM trained only on the source domain, could we expect similar results?


