Automated Concatenation of Embeddings for Structured Prediction

Xinyu Wang, Yong Jiang, Nguyen Bach, Tao Wang, Zhongqiang Huang, Fei Huang, Kewei Tu
Motivation

• Pretrained contextualized embeddings have significantly improved the performance of structured prediction tasks in NLP
• The ever-increasing number of embedding learning methods makes the choice of best embedding concatenation difficult
• Exploring all possible concatenations can be prohibitively demanding in computing resources
Automated Concatenation of Embeddings (ACE)

• Automate the process of finding better concatenations of embeddings
• Formulate the problem as an neural architecture search (NAS) problem
Automated Concatenation of Embeddings (ACE)

- A controller samples a subset of embeddings according to its belief model
Automated Concatenation of Embeddings (ACE)

• A controller samples a concatenation of embeddings according to its belief model
• The concatenated word represents are fed as input of a task model and return the model accuracy after training
Automated Concatenation of Embeddings (ACE)

• A controller samples a concatenation of embeddings according to its belief model

• The concatenated word represents are fed as input of a task model and return the model accuracy after training

• Use the accuracy as a reward signal and update the controller’s belief model
Automated Concatenation of Embeddings (ACE)

- A controller samples a concatenation of embeddings according to its belief model.
- The concatenated word representations are fed as input to a task model and return the model accuracy after training.
- Use the accuracy as a reward signal and update the controller’s belief model.
- Optimization: policy gradient algorithm in reinforcement learning.
Task Model

- **Sequence-structured outputs**
  - BiLSTM-CRF: \( P^{\text{seq}}(y|x) = \text{BiLSTM-CRF}(V, y) \)

- **Graph-structured outputs**
  - BiLSTM-Biaffine: \( P^{\text{graph}}(y|x) = \text{BiLSTM-Biaffine}(V, y) \)

- **Word representation:** \( V = [v_1; \cdots; v_n] \)
  - Embedding concatenation: \( v_i^l = \text{embed}_i^l(x); \quad v_i = [v_i^1; v_i^2; \cdots; v_i^L] \)
Search Space Design

- Decide which embedding candidates are concatenated as word representation $v_i = \{v_i^1, ..., v_i^l, ..., v_i^L\}$
  - The resulting search space contains $2^L$ possible combinations

- Problem: Variable hidden size of word representation making the task model difficult to be shared throughout the training
Search Space Design

• Solution: use a binary vector to mask out embeddings which are not selected

\[ \mathbf{a} = [a_1, \cdots, a_l, \cdots, a_L]; \mathbf{v}_i = [\mathbf{v}_i^1 a_1; \cdots; \mathbf{v}_i^l a_l; \cdots; \mathbf{v}_i^L a_L] \]

• Benefit:
  • The model weights can be shared after applying the embedding mask to all embedding candidates' concatenation
  • We can remove the unused embedding candidates after training
Searching in the Space

• The parameter for the controller: \( \boldsymbol{\theta} = [\theta_1; \theta_2; \ldots; \theta_L] \)

• The probability distribution of selecting a certain concatenation \( \boldsymbol{a} \):
  \[
P^{\text{ctrl}}(\boldsymbol{a}; \boldsymbol{\theta}) = \prod_{l=1}^{L} P^{\text{ctrl}}(a_l; \theta_l)
  \]

• Each element \( a_l \) of \( \boldsymbol{a} \) is sampled independently from a Bernoulli distribution
Optimization

- Policy gradient with accuracy $R$: $J(\theta) = \mathbb{E}_{P_{ctrl}(a;\theta)}[R]$
- Approximate the gradient $J(\theta)$ by sampling only one selection:

$$\nabla_\theta J(\theta) \approx \sum_{l=1}^{L} \nabla_\theta \log P^\text{ctrl}_l(a_l; \theta_l)(R - b)$$
Optimization: Reward Function

• Reward function on how each embedding candidate contributes to accuracy change

\[ r^t = \sum_{i=1}^{t-1} (R_t - R_i) \gamma^{Hamm(a^t, a^i) - 1} |a^t - a^i| \]

- A reward for each embedding
- Accumulated accuracy change
- When many embeddings are switched on/off, we are unsure which should get the credit, so we discount it
- Only those responsible for the accuracy change get the credit
Optimization

• The gradient of $J(\theta)$ is then formulated as:

$$\nabla_{\theta} J_t(\theta) \approx \sum_{l=1}^{L} \nabla_{\theta} \log P_l^{\text{ctrl}}(a^t_l; \theta_l)r^t_l$$
Training

1. Initialization: A dictionary $\mathcal{D}$ to store the searched concatenations and scores. Set time step $t = 0$.
2. Sample a concatenation $a^t$ based on the probability distribution
3. Train the task model with $a^t$ and evaluate the model on the development set to get the accuracy $R_t$.
4. Given the concatenation $a^t$, accuracy $R_t$ and $\mathcal{D}$, compute the gradient of $J(\theta)$ and update the parameters of controller.
5. Add $a^t$ and $R_t$ into $\mathcal{D}$, set $t = t + 1$.
6. Repeat 2~5 until $t$ is larger than a maximum iteration $T$
Experiments

• Structured prediction tasks varying from syntactic tasks to semantic tasks:
  • NER: 5 datasets
  • PoS Tagging: 3 datasets
  • Chunking: 1 dataset
  • Abstract Extraction (AE): 8 datasets
  • Syntactic Dependency Parsing (DP): 1 dataset
  • Semantic Dependency Parsing (SDP): 3 datasets

• 6 tasks over 21 datasets
Embeddings

- ELMo: monolingual
- Flair: monolingual + multilingual
- BERT: monolingual + multilingual
- XLM-R: multilingual
- GLoVe: English
- fastText: monolingual
- Character embeddings: train over the task

- The size of search space (for English): $2^{11} - 1 = 2047$
Baselines

• All
  • The concatenation of all the embeddings
  • Let the task model learn by itself the contribution of each embedding candidate

• Random
  • Randomly search the concatenation of embeddings
  • A strong baseline in NAS
## Compare with Baselines

<table>
<thead>
<tr>
<th></th>
<th>NER</th>
<th>POS</th>
<th>AE</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>de</td>
<td>en</td>
<td>es</td>
</tr>
<tr>
<td>ALL</td>
<td>83.1</td>
<td>92.4</td>
<td><strong>88.9</strong></td>
</tr>
<tr>
<td>RANDOM</td>
<td>84.0</td>
<td>92.6</td>
<td>88.8</td>
</tr>
<tr>
<td>ACE</td>
<td>84.2</td>
<td>93.0</td>
<td><strong>88.9</strong></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>CHUNK</th>
<th>DP</th>
<th>SDP</th>
<th>Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CoNLL 2000</td>
<td>UAS</td>
<td>LAS</td>
<td>DM-ID</td>
</tr>
<tr>
<td>ALL</td>
<td>96.7</td>
<td>96.7</td>
<td>95.1</td>
<td>94.3</td>
</tr>
<tr>
<td>RANDOM</td>
<td>96.7</td>
<td>96.8</td>
<td>95.2</td>
<td>94.4</td>
</tr>
<tr>
<td>ACE</td>
<td>96.8</td>
<td><strong>96.9</strong></td>
<td><strong>95.3</strong></td>
<td><strong>94.5</strong></td>
</tr>
</tbody>
</table>
## Compare with SotA (sequence-structured tasks)

<table>
<thead>
<tr>
<th></th>
<th>NER</th>
<th>POS</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>de</td>
<td>de\textsuperscript{06}</td>
</tr>
<tr>
<td>Baevski et al. (2019)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Straková et al. (2019)</td>
<td>85.1</td>
<td>-</td>
</tr>
<tr>
<td>Yu et al. (2020)</td>
<td>86.4</td>
<td>90.3</td>
</tr>
<tr>
<td>Yamada et al. (2020)</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XLM-R+Fine-tune\textsuperscript{∞}</td>
<td>87.7</td>
<td>91.4</td>
</tr>
<tr>
<td>ACE+Fine-tune</td>
<td>88.3</td>
<td>91.7</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Akbik et al. (2018)</td>
<td>96.7</td>
<td>Xu et al. (2018)\textsuperscript{†}</td>
<td>84.2</td>
<td>84.6</td>
<td>72.0</td>
<td>75.4</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Clark et al. (2018)</td>
<td>97.0</td>
<td>Xu et al. (2019)</td>
<td>84.3</td>
<td>-</td>
<td>-</td>
<td>78.0</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Liu et al. (2019b)</td>
<td>97.3</td>
<td>Wang et al. (2020)</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>72.8</td>
<td>74.3</td>
<td>72.9</td>
<td>71.8</td>
</tr>
<tr>
<td>Chen et al. (2020)</td>
<td>95.5</td>
<td>Wei et al. (2020)</td>
<td>82.7</td>
<td>87.1</td>
<td>72.7</td>
<td>77.7</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>XLM-R+Fine-tune</td>
<td>97.0</td>
<td>XLM-R+Fine-tune</td>
<td>85.9</td>
<td>90.5</td>
<td>76.4</td>
<td>78.9</td>
<td>77.0</td>
<td>77.6</td>
<td>77.7</td>
</tr>
<tr>
<td>ACE+Fine-tune</td>
<td>97.3</td>
<td>ACE+Fine-tune</td>
<td>87.4</td>
<td>92.0</td>
<td>80.3</td>
<td>81.3</td>
<td>79.9</td>
<td>80.5</td>
<td>79.4</td>
</tr>
</tbody>
</table>
Compare with SotA (Graph-structured Tasks)

<table>
<thead>
<tr>
<th></th>
<th>DP PTB</th>
<th>SDP</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>UAS</td>
<td>LAS</td>
</tr>
<tr>
<td>Zhou and Zhao (2019)†</td>
<td>97.2</td>
<td>95.7</td>
</tr>
<tr>
<td>Mrini et al. (2020)†</td>
<td>97.4</td>
<td>96.3</td>
</tr>
<tr>
<td>Li et al. (2020)</td>
<td>96.6</td>
<td>94.8</td>
</tr>
<tr>
<td>Zhang et al. (2020)</td>
<td>96.1</td>
<td>94.5</td>
</tr>
<tr>
<td>Wang and Tu (2020)</td>
<td>96.9</td>
<td>95.3</td>
</tr>
<tr>
<td>XLNET+Fine-tune</td>
<td>97.0</td>
<td>95.6</td>
</tr>
<tr>
<td>ACE+Fine-tune</td>
<td>97.2</td>
<td>95.8</td>
</tr>
</tbody>
</table>
Conclusion

• We propose Automated Concatenation of Embeddings
• A simple search space and a novel reward function to guide the search
• ACE outperforms strong baselines and achieves state-of-the-art performance in 6 tasks over 21 datasets