Automated Concatenation of Embeddings for Structured Prediction

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

上海科技大学
ShanghaiTech University

DAMO
ALIBABA DAMO ACADEMY

Motivation

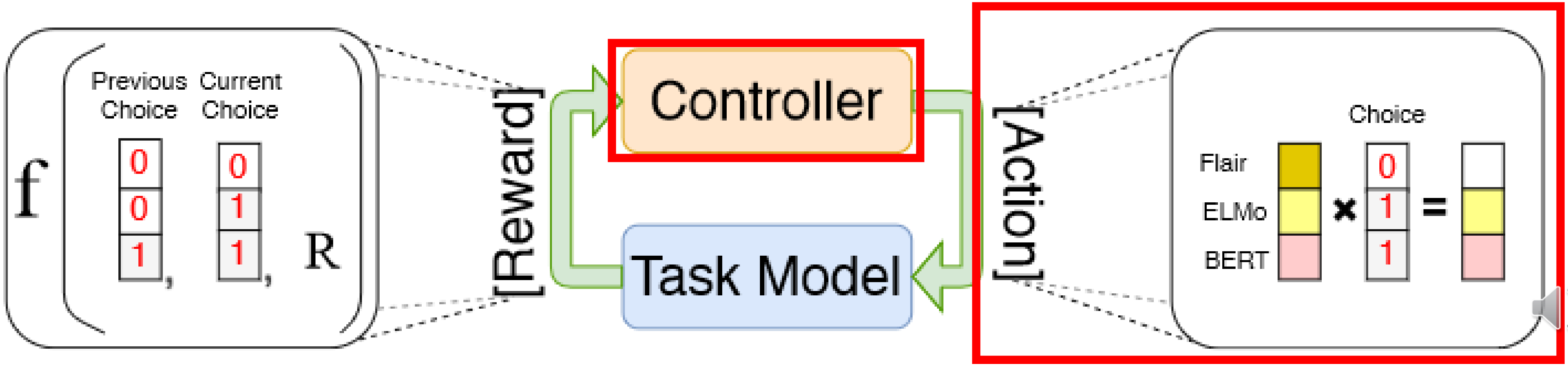
- Pretrained contextualized embeddings have significantly improved the performance of structured prediction tasks in NLP
- The ever-increasing number of embedding learning methods makes the choice of best embedding concatenation difficult
- Exploring all possible concatenations can be prohibitively demanding 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

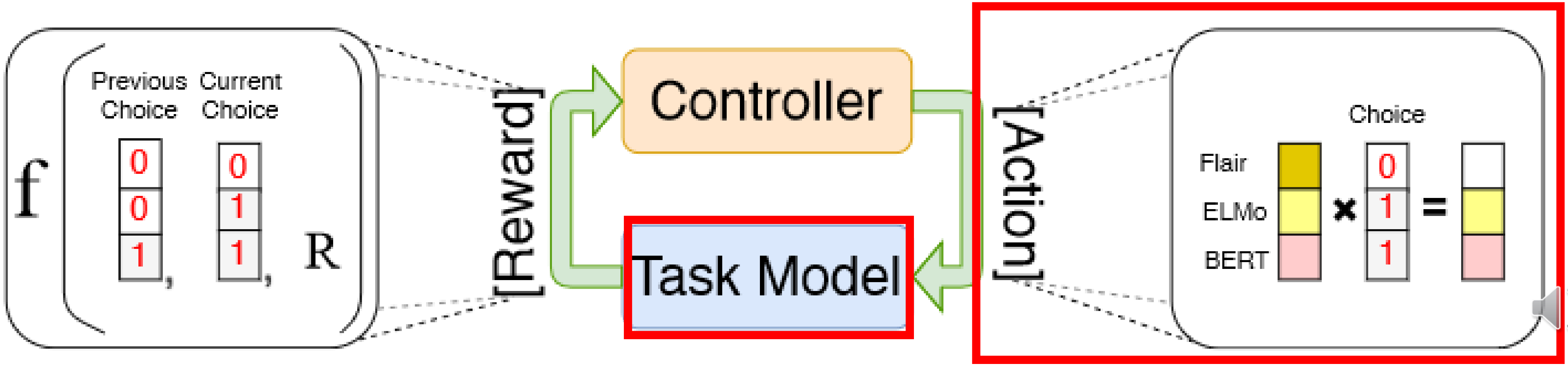
Automated Concatenation of Embeddings (ACE)

- A controller samples a subset of embeddings according to its belief model



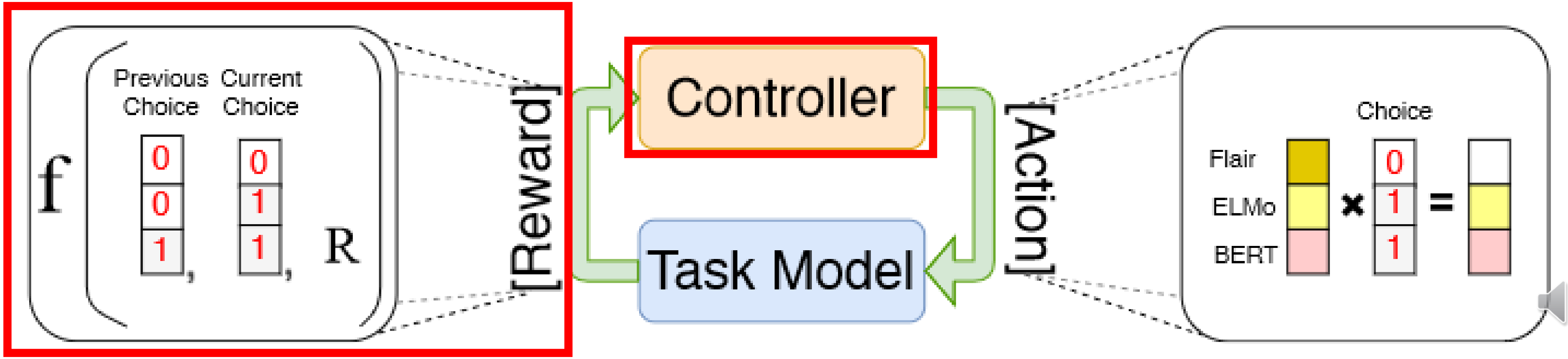
Automated Concatenation of Embeddings (ACE)

- 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



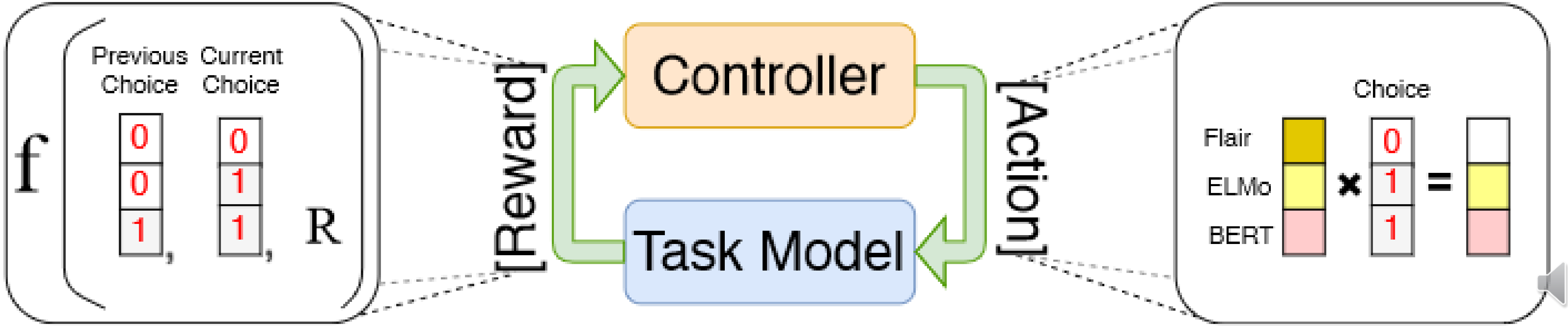
Automated Concatenation of Embeddings (ACE)

- 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



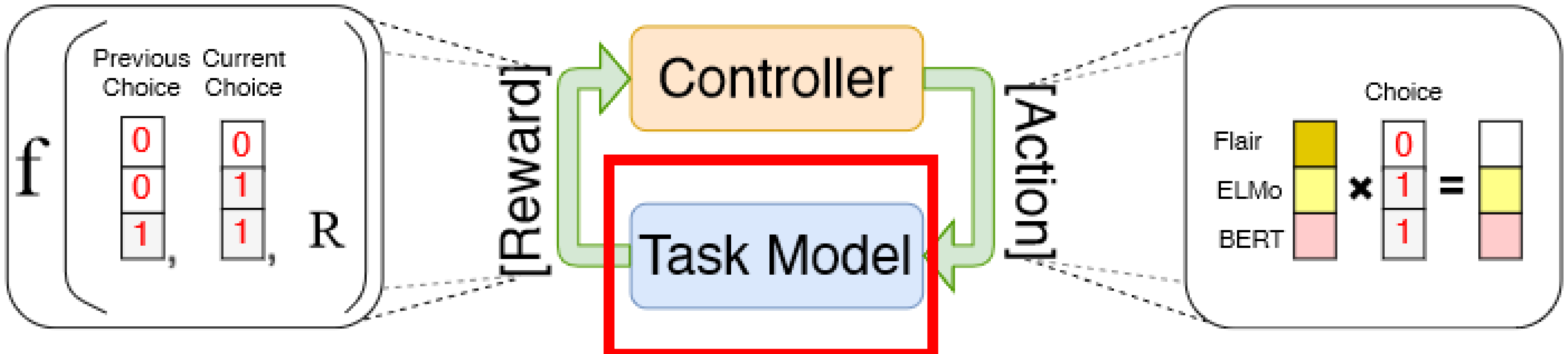
Automated Concatenation of Embeddings (ACE)

- A controller samples a concatenation of embeddings according to its belief model
- The concatenated word 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
- Optimization: policy gradient algorithm in reinforcement learning



Task Model

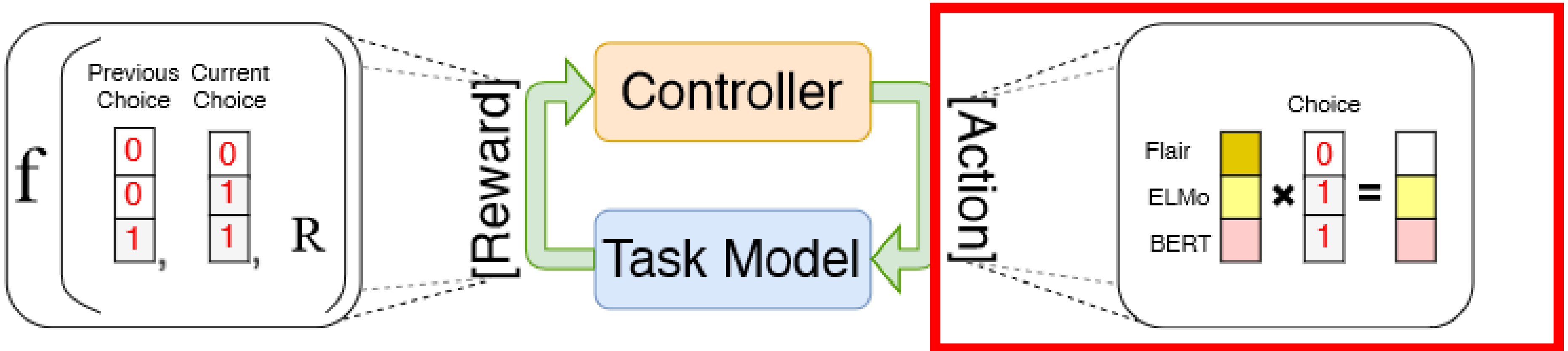
- Sequence-structured outputs
  - BiLSTM-CRF: $P^{\text{seq}}(\boldsymbol{y}|\boldsymbol{x}) = \textbf{BiLSTM-CRF}(\boldsymbol{V}, \boldsymbol{y})$
- Graph-structured outputs
  - BiLSTM-Biaffine: $P^{\text{graph}}(\boldsymbol{y}|\boldsymbol{x}) = \textbf{BiLSTM-Biaffine}(\boldsymbol{V}, \boldsymbol{y})$
- Word representation: $\boldsymbol{V} = [\boldsymbol{v}_1; \cdots; \boldsymbol{v}_n]$
  - Embedding concatenation $\boldsymbol{v}_i^l = \text{embed}_i^l(\boldsymbol{x}); \quad \boldsymbol{v}_i = [\boldsymbol{v}_i^1; \boldsymbol{v}_i^2; \ldots; \boldsymbol{v}_i^L]$



Search Space Design

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

- Problem: Variable hidden size of word representation making the task model difficult to be shared through out the training

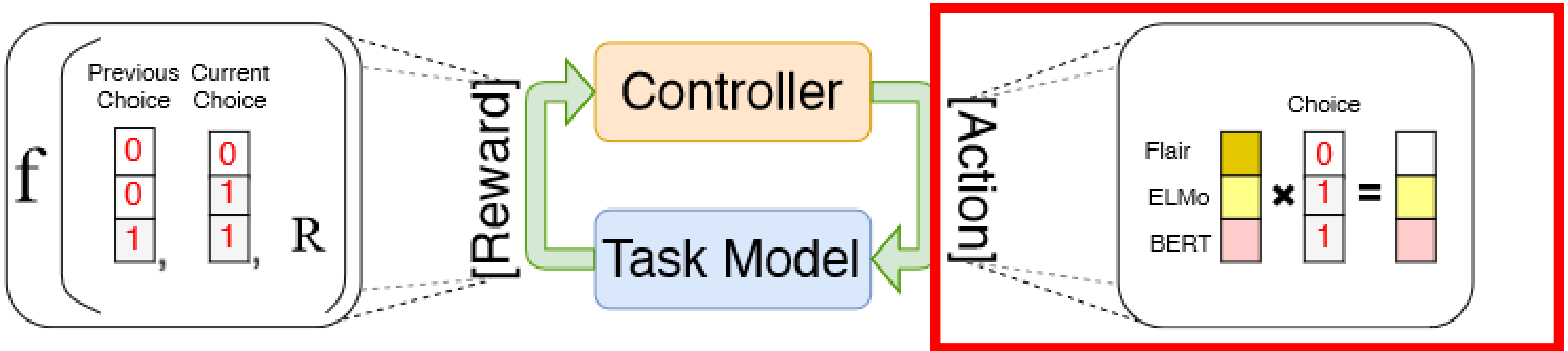


Search Space Design

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

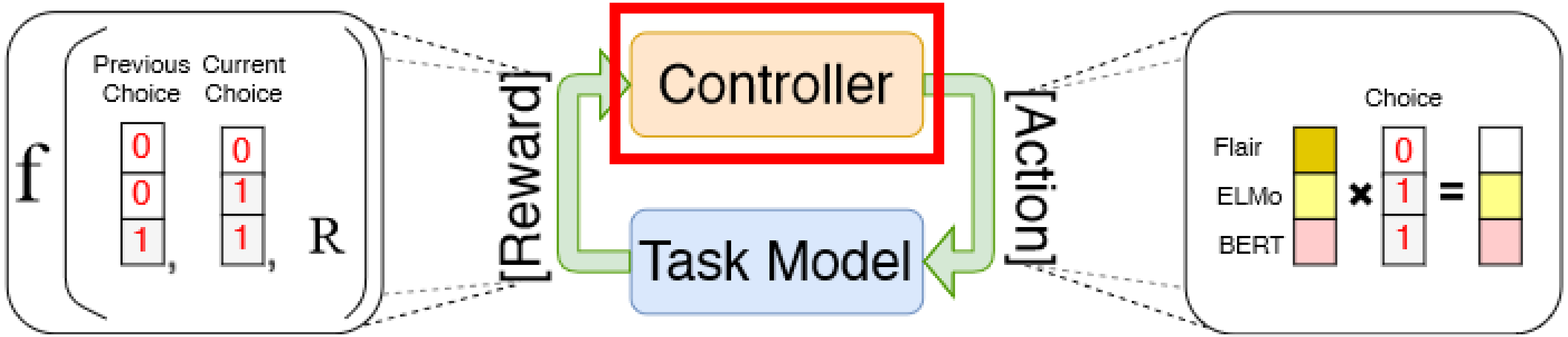
$$\boldsymbol{a} = [a_1, \cdots, a_l, \cdots, a_L]; \boldsymbol{v}_i = [\boldsymbol{v}_i^1 a_1; \ldots; \boldsymbol{v}_i^l a_l; \ldots; \boldsymbol{v}_i^L a_L]$$

- 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

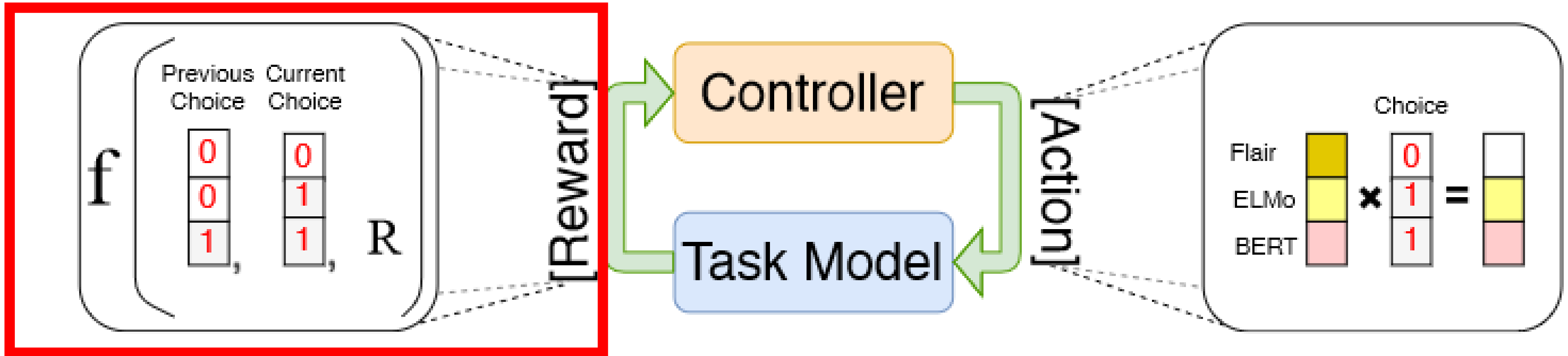
- The parameter for the controller: $\boldsymbol{\theta} = [\theta_1; \theta_2; \ldots; \theta_L]$
- The probability distribution of selecting a certain concatenation $\boldsymbol{a}$: $P^{\text{ctrl}}(\boldsymbol{a}; \boldsymbol{\theta}) = \prod_{l=1}^{L} P_l^{\text{ctrl}}(a_l; \theta_l)$
- Each element a_l of $\boldsymbol{a}$ is sampled independently from a Bernoulli distribution



Optimization

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

$$\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) \approx \sum_{l=1}^{L} \nabla_{\boldsymbol{\theta}} \log P_l^{\text{ctrl}}(a_l; \theta_l)(R - b)$$



Optimization: Reward Function

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

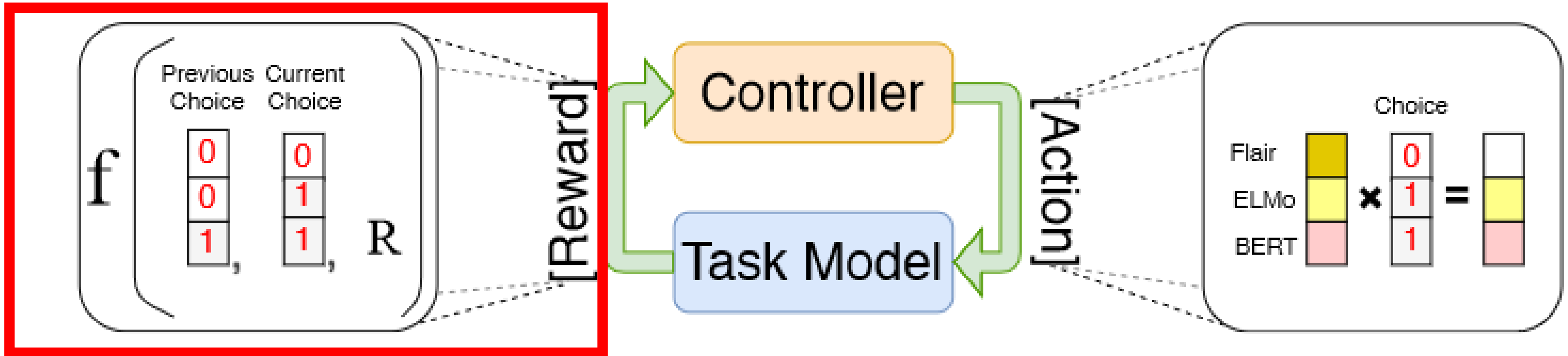
$$\boldsymbol{r}^t = \sum_{i=1}^{t-1} (R_t - R_i)\gamma^{Hamm(\boldsymbol{a}^t,\boldsymbol{a}^i)-1} |\boldsymbol{a}^t - \boldsymbol{a}^i|$$

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

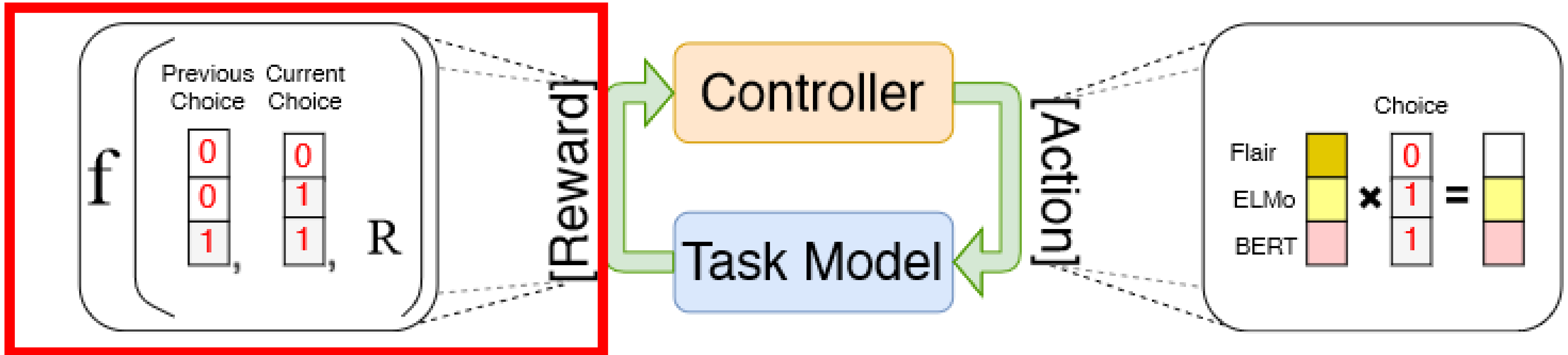
Only those responsible for the accuracy change get the credit



Optimization

- The gradient of $\boldsymbol{J}(\boldsymbol{\theta})$ is then formulated as:

$$\nabla_{\boldsymbol{\theta}} J_t(\boldsymbol{\theta}) \approx \sum_{l=1}^{L} \nabla_{\boldsymbol{\theta}} \log P_l^{\text{ctrl}}(a_l^t; \theta_l) r_l^t$$



Training

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

	NER				POS			AE							
	de	en	es	nl	Ritter	ARK	TB-v2	14Lap	14Res	15Res	16Res	es	nl	ru	tr
ALL	83.1	92.4	**88.9**	89.8	90.6	92.1	94.6	82.7	88.5	74.2	73.2	74.6	75.0	67.1	67.5
RANDOM	84.0	92.6	88.8	91.9	91.3	92.6	94.6	83.6	88.1	73.5	74.7	75.0	73.6	68.0	70.0
ACE	**84.2**	**93.0**	**88.9**	**92.1**	**91.7**	**92.8**	**94.8**	**83.9**	**88.6**	**74.9**	**75.6**	**75.7**	**75.3**	**70.6**	**71.1**

	CHUNK	DP		SDP						AVG
	CoNLL 2000	UAS	LAS	DM-ID	DM-OOD	PAS-ID	PAS-OOD	PSD-ID	PSD-OOD	
ALL	96.7	96.7	95.1	94.3	90.8	**94.6**	92.9	82.4	81.7	85.3
RANDOM	96.7	96.8	95.2	94.4	90.8	**94.6**	93.0	82.3	81.8	85.7
ACE	**96.8**	**96.9**	**95.3**	**94.5**	**90.9**	94.5	**93.1**	**82.5**	**82.1**	**86.2**

Compare with SotA (sequence-structured tasks)

	NER						POS		
	de	de_{06}	en	es	nl		Ritter	ARK	TB-v2
Baevski et al. (2019)	-	-	93.5	-	-	Owoputi et al. (2013)	90.4	93.2	94.6
Straková et al. (2019)	85.1	-	93.4	88.8	92.7	Gui et al. (2017)	90.9	-	92.8
Yu et al. (2020)	86.4	90.3	93.5	90.3	93.7	Gui et al. (2018)	91.2	92.4	-
Yamada et al. (2020)	-	-	94.3	-	-	Nguyen et al. (2020)	90.1	94.1	95.2
XLM-R+Fine-tune$^{\diamond}$	87.7	91.4	94.1	89.3	95.3	XLM-R+Fine-tune	92.3	93.7	95.4
ACE+Fine-tune	**88.3**	**91.7**	**94.6**	**95.9**	**95.7**	ACE+Fine-tune	**93.4**	**94.4**	**95.8**

	CHUNK		AE							
	CoNLL 2000		14Lap	14Res	15Res	16Res	es	nl	ru	tr
Akbik et al. (2018)	96.7	Xu et al. (2018)[†]	84.2	84.6	72.0	75.4	-	-	-	-
Clark et al. (2018)	97.0	Xu et al. (2019)	84.3	-	-	78.0	-	-	-	-
Liu et al. (2019b)	**97.3**	Wang et al. (2020)	-	-	-	72.8	74.3	72.9	71.8	59.3
Chen et al. (2020)	95.5	Wei et al. (2020)	82.7	87.1	72.7	77.7	-	-	-	-
XLM-R+Fine-tune	97.0	XLM-R+Fine-tune	85.9	90.5	76.4	78.9	77.0	77.6	77.7	74.1
ACE+Fine-tune	**97.3**	ACE+Fine-tune	**87.4**	**92.0**	**80.3**	**81.3**	**79.9**	**80.5**	**79.4**	**81.9**

Compare with SotA (Graph-structured Tasks)

	DP			SDP					
	PTB			DM		PAS		PSD	
	UAS	LAS		ID	OOD	ID	OOD	ID	OOD
Zhou and Zhao (2019)†	97.2	95.7	He and Choi (2020)‡	94.6	90.8	96.1	94.4	86.8	79.5
Mrini et al. (2020)†	97.4	96.3	D & M (2018)	93.7	88.9	93.9	90.6	81.0	79.4
Li et al. (2020)	96.6	94.8	Wang et al. (2019)	94.0	89.7	94.1	91.3	81.4	79.6
Zhang et al. (2020)	96.1	94.5	Jia et al. (2020)	93.6	89.1	-	-	-	-
Wang and Tu (2020)	96.9	95.3	F & G (2020)	94.4	91.0	95.1	93.4	82.6	82.0
XLNET+Fine-tune	97.0	95.6	XLNet+Fine-tune	94.2	90.6	94.8	93.4	82.7	81.8
ACE+Fine-tune	**97.2**	**95.8**	ACE+Fine-tune	**95.6**	**92.6**	**95.8**	**94.6**	**83.8**	**83.4**

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