Paper Report

周杰　2017/9/20

Attentive Interactive Neural Networks for Answer Selection in Community Question Answering

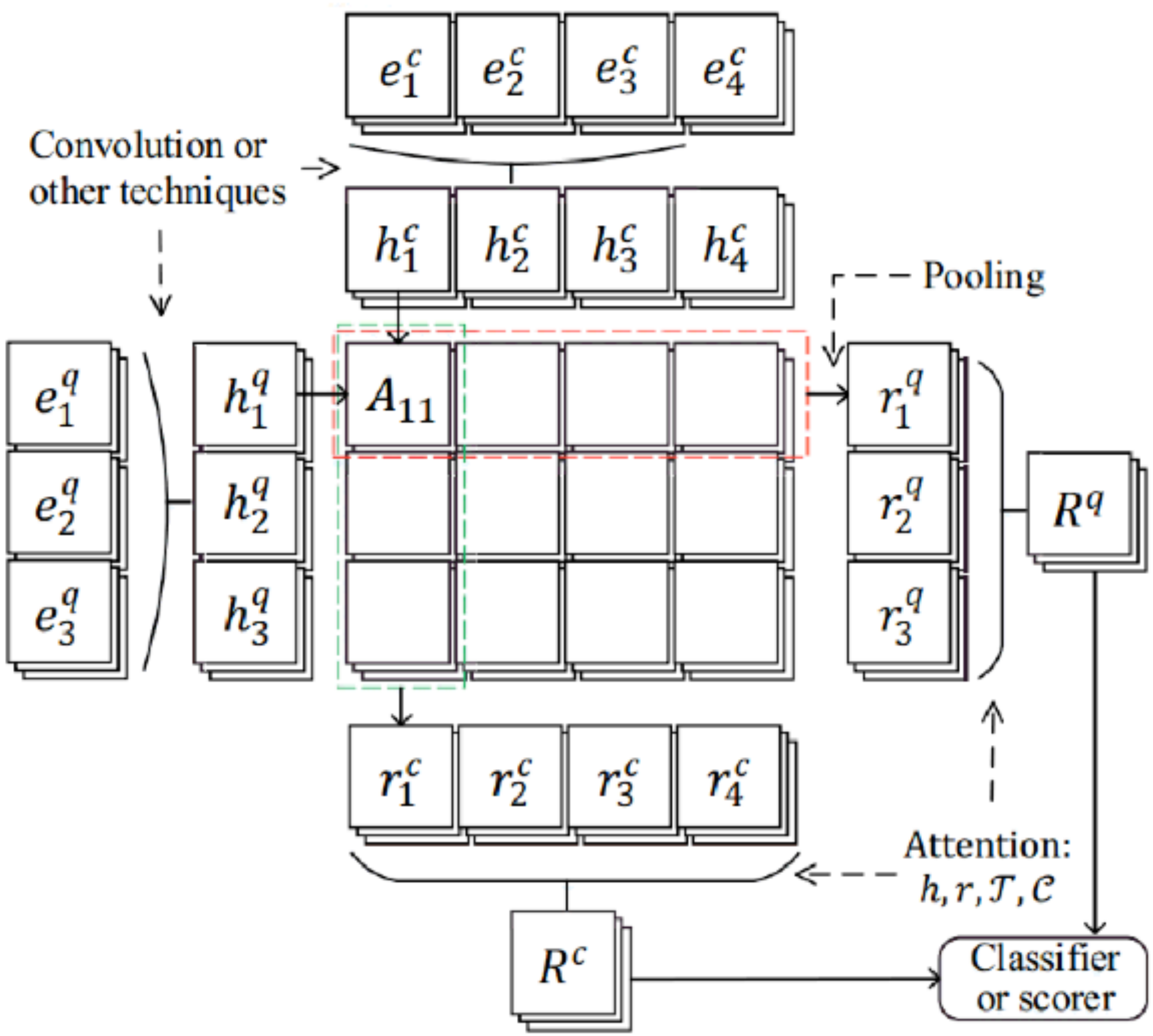
AAAI 2017

Peking University
Xiaodong Zhang

Motivation

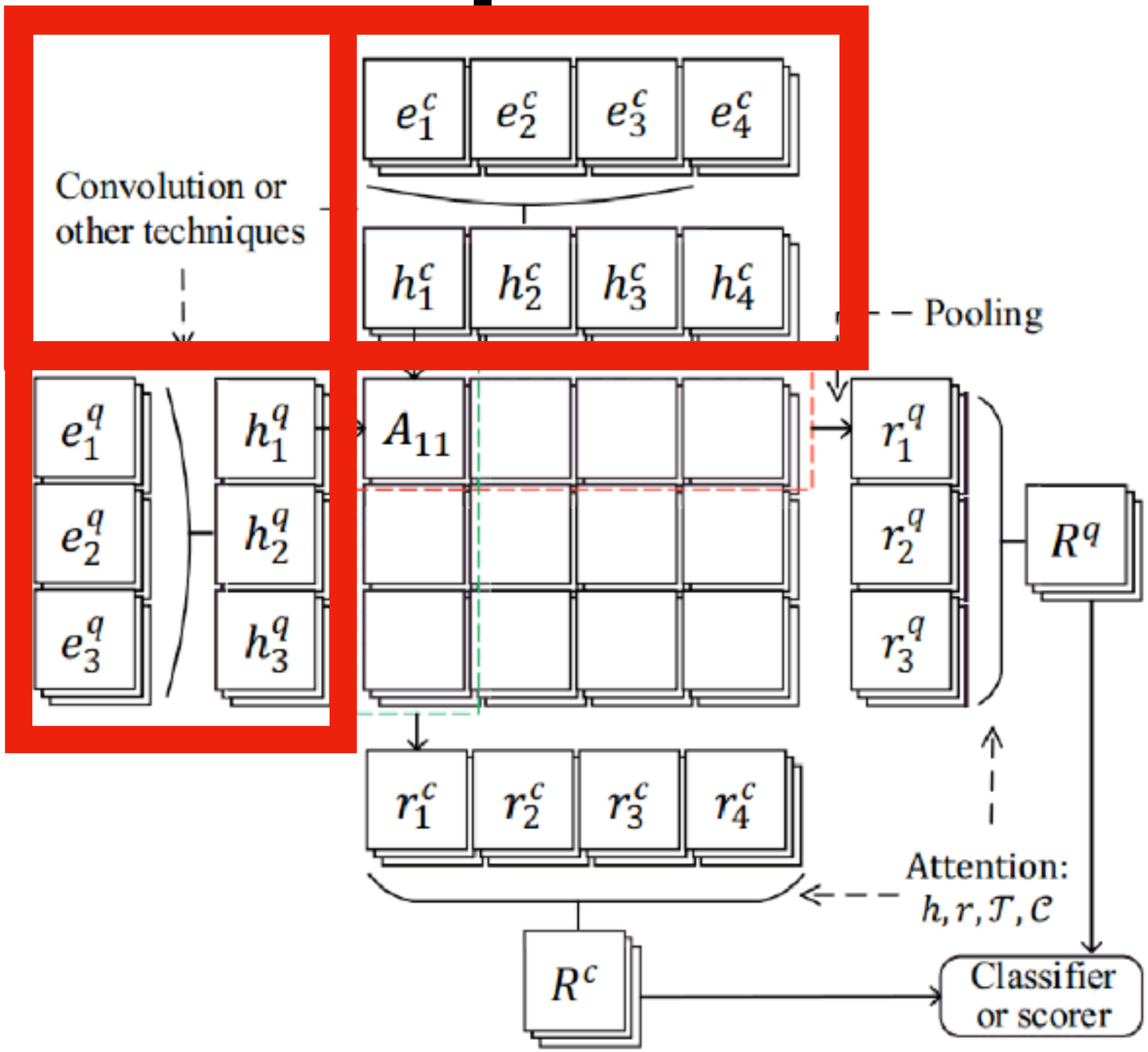
- **Attentive interactive neural network(AI-NN)**
- **Interactive**
  - **Model the relation between each segments of question and answer**
- **Row-wise and column-wise max-pooling**
- **Attention**
  - **Representation of segment**
  - **Question topic**
  - **Question type**

AI-NN



e_1^c
e_2^c
e_3^c
e_4^c
Convolution or other techniques
h_1^c
h_2^c
h_3^c
h_4^c
Pooling
e_1^q
e_2^q
e_3^q
h_1^q
h_2^q
h_3^q
A_{11}
r_1^q
r_2^q
r_3^q
R^q
r_1^c
r_2^c
r_3^c
r_4^c
Attention: $h, r, \mathcal{T}, \mathcal{C}$
R^c
Classifier or scorer

Text Representation



Text Representation

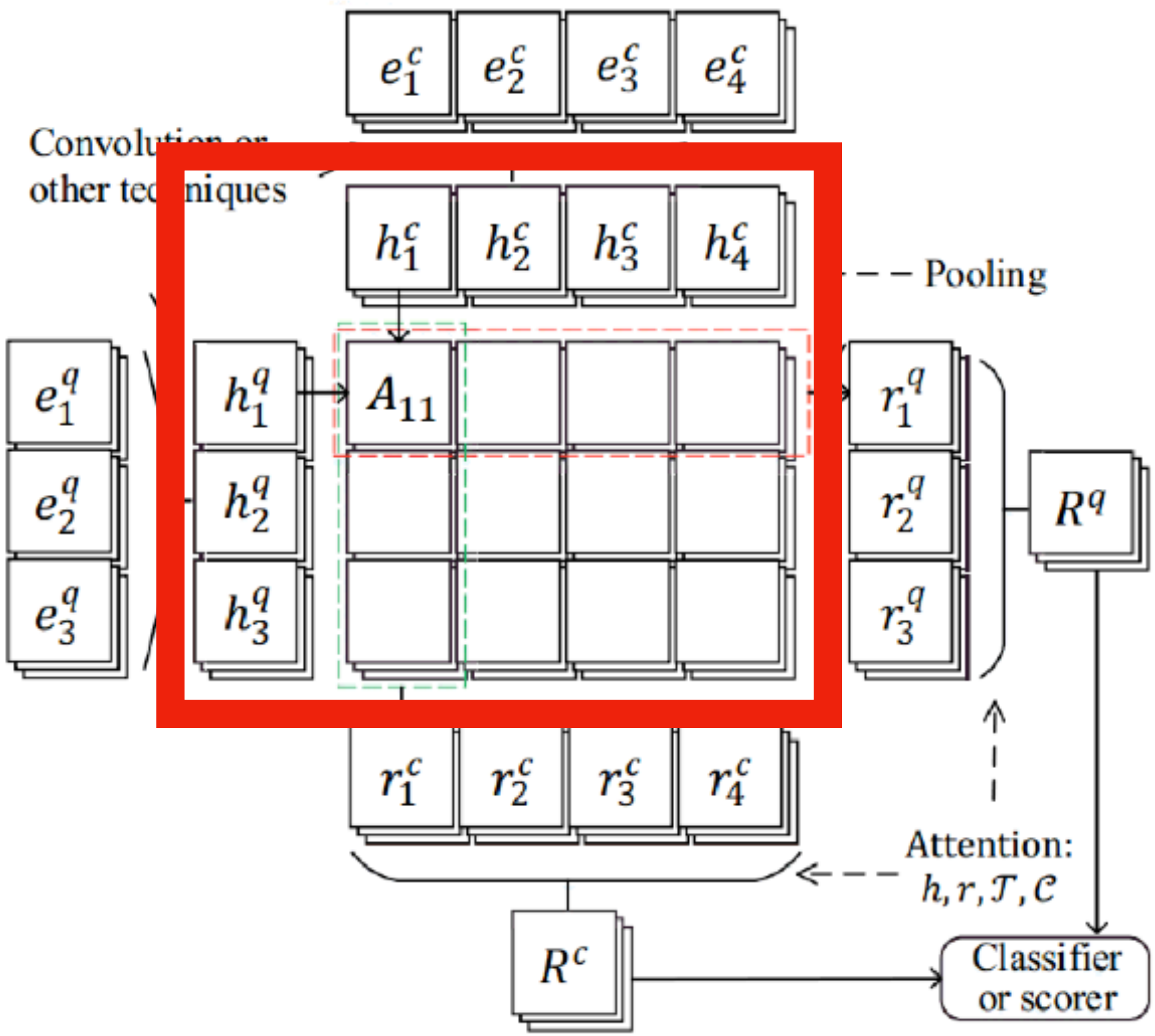
- **The t-th input of convolution layer x_t**

$$x_t = \left[e^q_{t-\lfloor d/2 \rfloor}, ..., e^q_t, ..., e^q_{t+\lceil d/2 \rceil -1}\right]$$

- **The hidden states of convolution layer**

$$h^q_i = \sigma(W^h * x_i + b^h)$$

Interaction

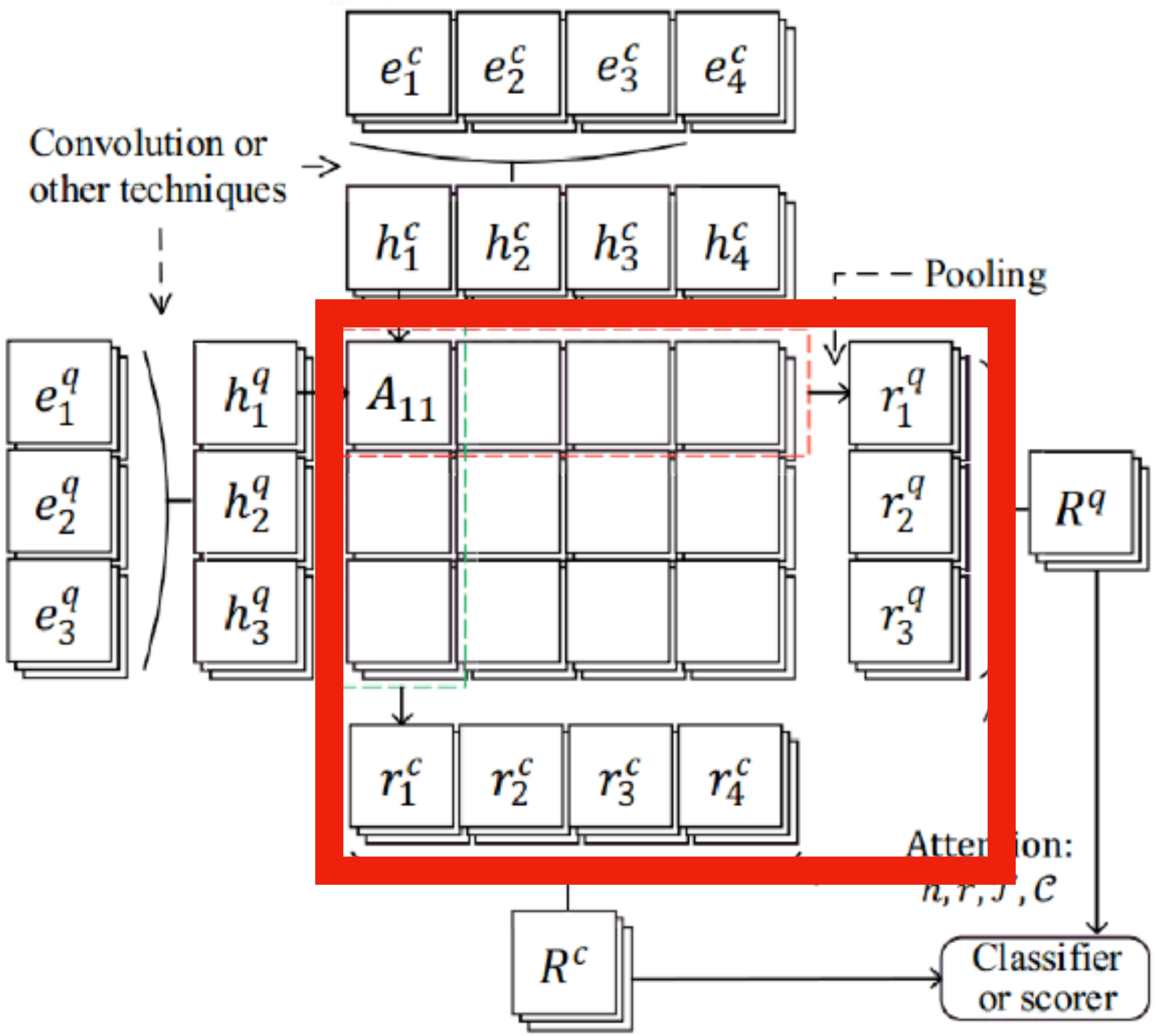


Interaction

- **For the i-th hidden state h_i^q and the j-th hidden state h_j^c their interaction A_{ij}**

$$A_{ij} = \sigma(W^a * [h_i^q, h_j^c] + b^a)$$

Max pooling



Convolution or
other techniques
Pooling
Attention:
Classifier
or scorer

Max pooling

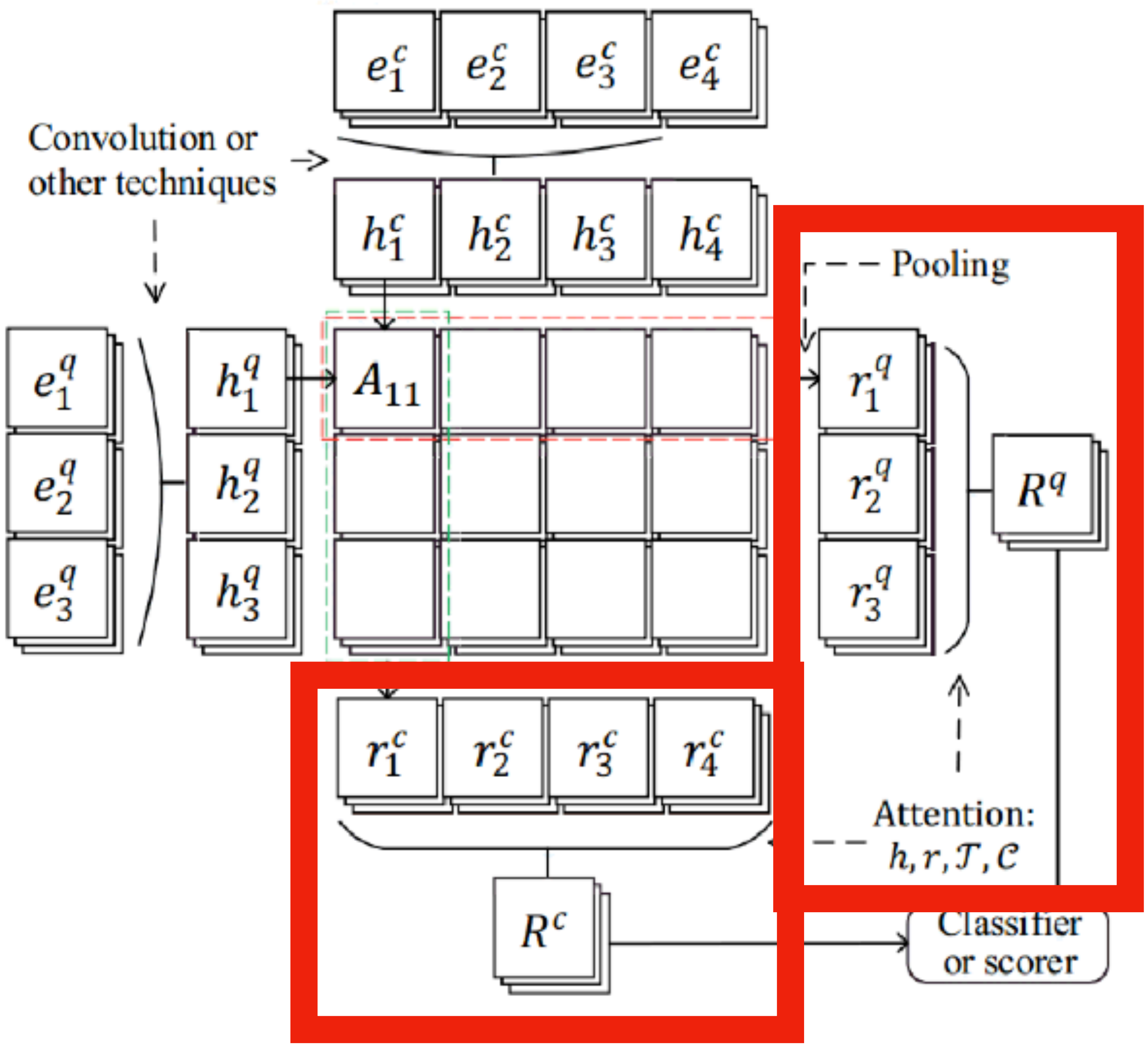
- **Row-wise max pooling**

$$r_i^q = \max_{m \in [1, T^c]} A_{im}$$

- **Column-wise max pooling**

$$r_j^c = \max_{n \in [1, T^q]} A_{nj}$$

Attention


e_1^c e_2^c e_3^c e_4^c
Convolution or other techniques
h_1^c h_2^c h_3^c h_4^c
Pooling
e_1^q e_2^q e_3^q
h_1^q h_2^q h_3^q
A_{11}
r_1^q r_2^q r_3^q
R^q
r_1^c r_2^c r_3^c r_4^c
Attention: $h, r, \mathcal{T}, \mathcal{C}$
R^c
Classifier or scorer

Attention Calculation

$$\alpha_i^q = \frac{\exp(u_i^q)}{\sum_{k=1}^{T^q} \exp(u_k^q)} \qquad R^q = \sum_{l=1}^{T^q} \alpha_l^q r_l^q$$

$$u_i^q = a(h_i^q, r_i^q, \mathcal{T}, \mathcal{C})$$

- **a is a feedforward neural network**
- $\mathcal{T}$ **:question topic**
- $\mathcal{C}$ **:question type**

Additional features

- **Whether answer and question are from the same user**
- **Whether the answer is anonymous(匿名)**
- **The order of an answer**
- **The length of an answer**

$$R = [R^T, R^F]$$
$$R^T = \sigma(W^T * [R^q, R^c] + b^T)$$

Results

- **Dataset: SemEval-2016 Subtask A**

Method	MAP	Acc	F1
ARC-I	77.05	74.07	69.50
ARC-II	77.98	75.26	71.64
AP	77.12	75.47	71.72
Kelp	79.19	75.11	64.36
ConvKN	77.66	75.54	66.16
AI-CNN (w/o features)	79.17	76.30	72.75
AI-CNN	80.14	76.87	73.03

Analysis of Attention

- **Additional features are not used**

Information	MAP	Acc	F1
w/o attention	78.05	75.19	71.50
+ representation	78.83	75.95	72.16
+ interaction	78.75	75.92	72.43
+ question topic	77.91	75.25	71.93
+ question type	78.22	75.22	72.10
+ all	79.17	76.30	72.75

Table 4: Influence of different information to attention calculation.

Analysis of Additional features

Feature	MAP	Acc	F1
w/o feature	79.17	76.30	72.75
+ a) same user	79.62	76.50	72.46
+ b) anonymous	78.87	76.41	72.11
+ c) order	79.54	76.62	72.91
+ d) length	79.32	76.22	72.83
+ all	80.14	76.87	73.03

Table 5: Influence of different features.

Improving Word Embeddings with Convolutional Feature Learning and Subword Information

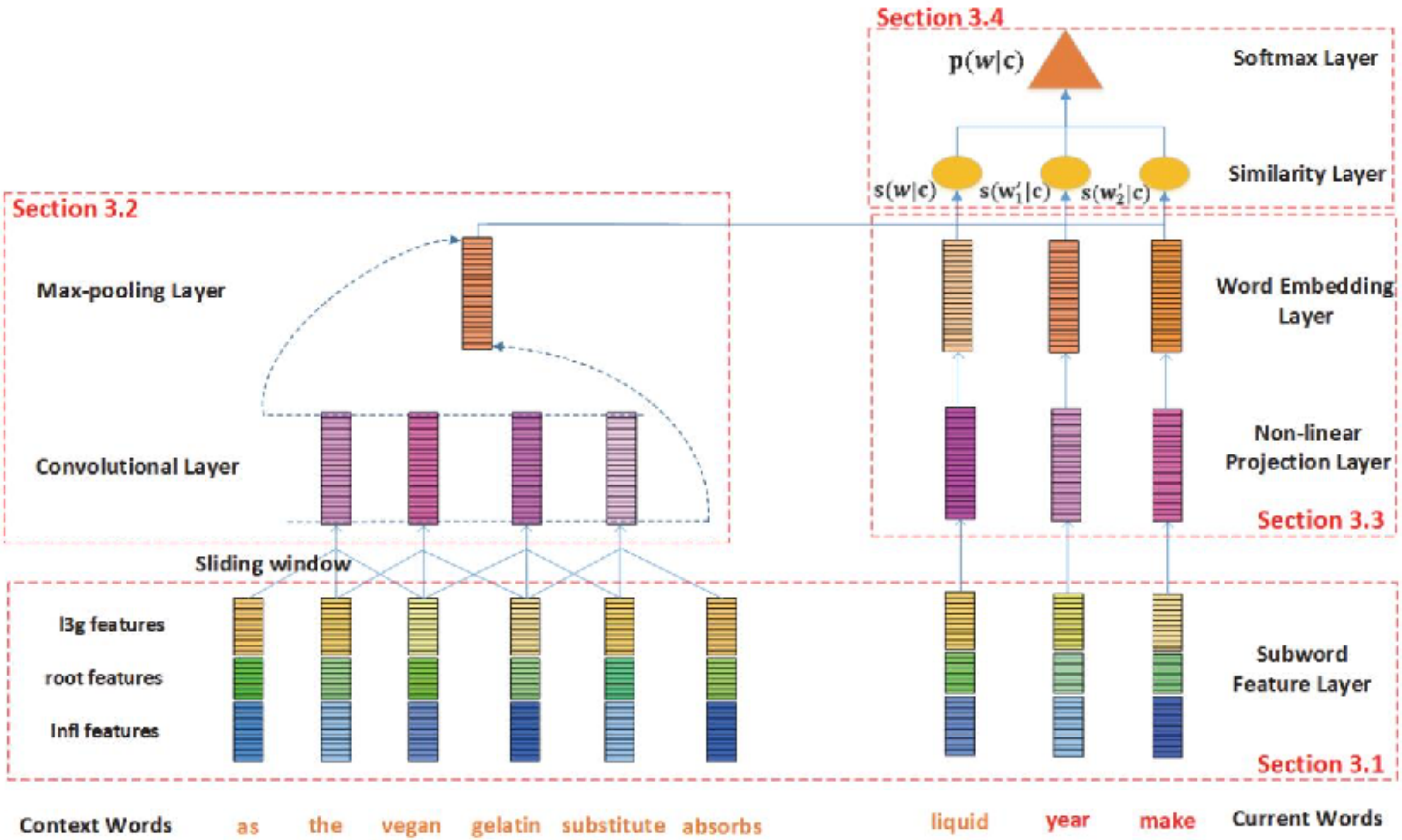
AAAI 2017

Singapore University
Shaosheng Cao

Motivation

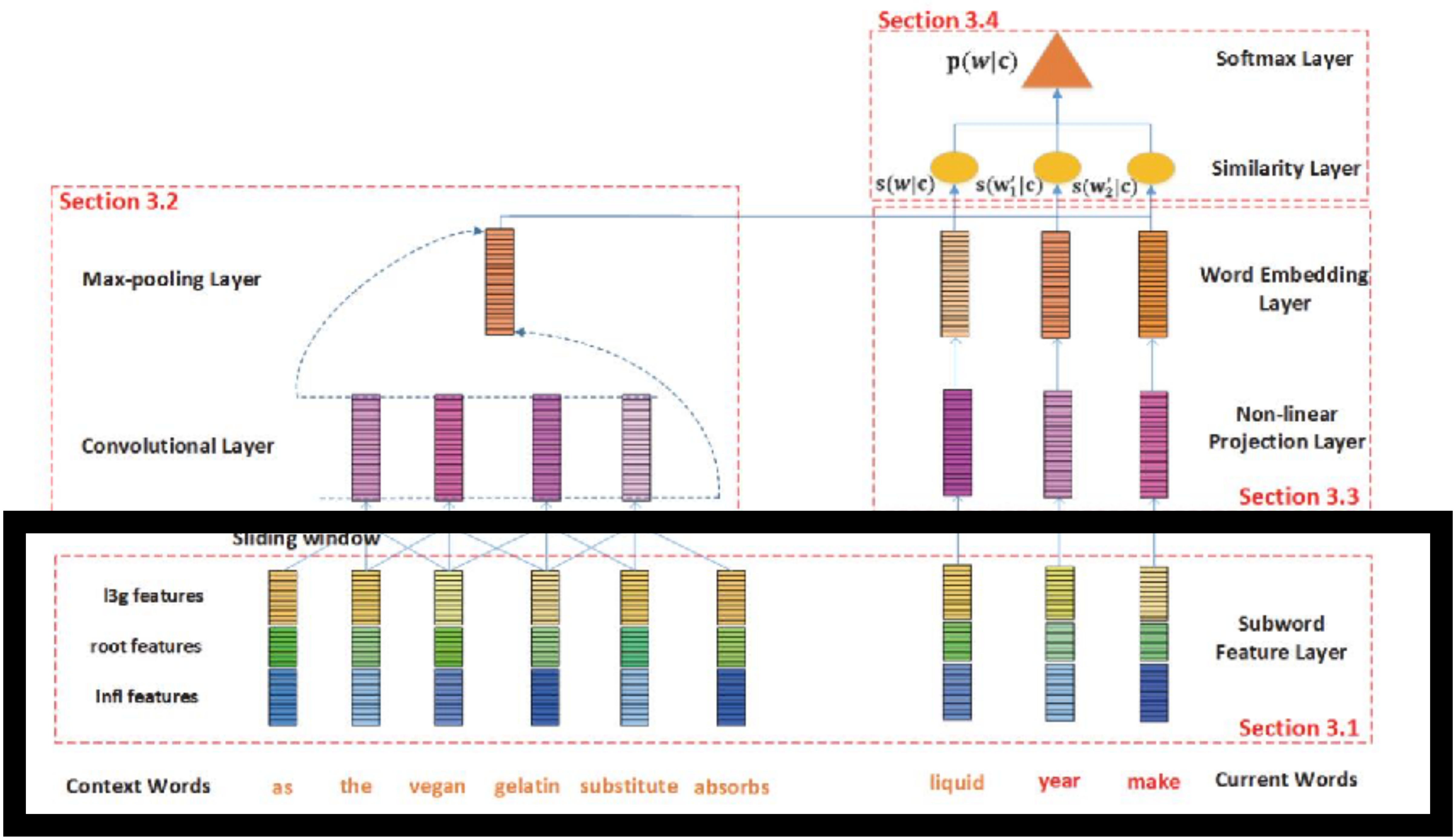
- **Convolutional feature learning**
  - **Capture the structural information of their context**
- **Exploring subword information**
  - **Character n-gram**
  - **Root/affix**
  - **Inflections**

Model

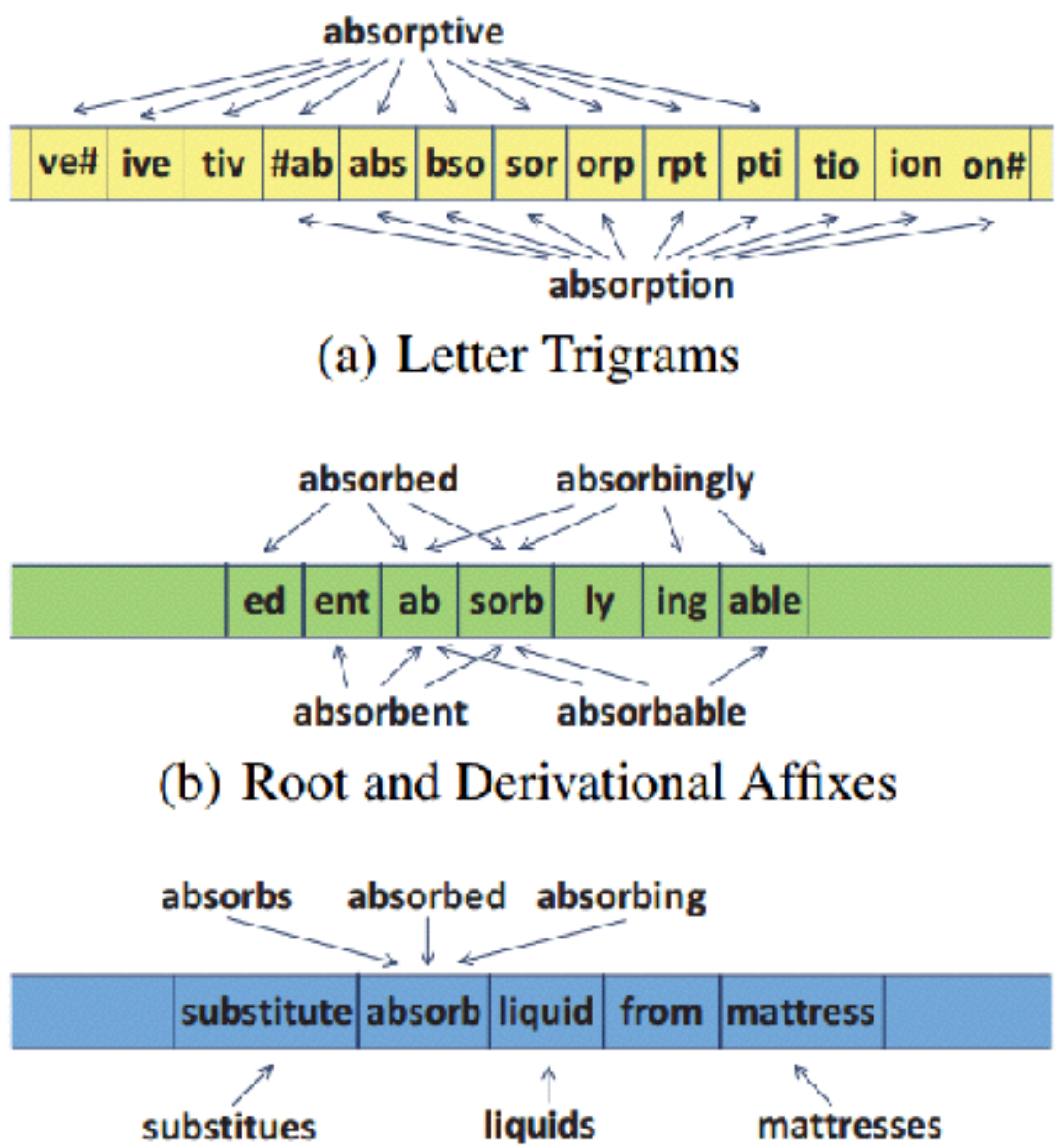


Section 3.4
p(w|c)
Softmax Layer
Similarity Layer
s(w|c)
Section 3.2
Max-pooling Layer
Word Embedding Layer
Convolutional Layer
Non-linear Projection Layer
Section 3.3
Sliding window
l3g features
root features
Infl features
Subword Feature Layer
Section 3.1
Context Words
as
the
vegan
gelatin
substitute
absorbs
liquid
year
make
Current Words

Extracting Subword Features



Extracting Subword Features

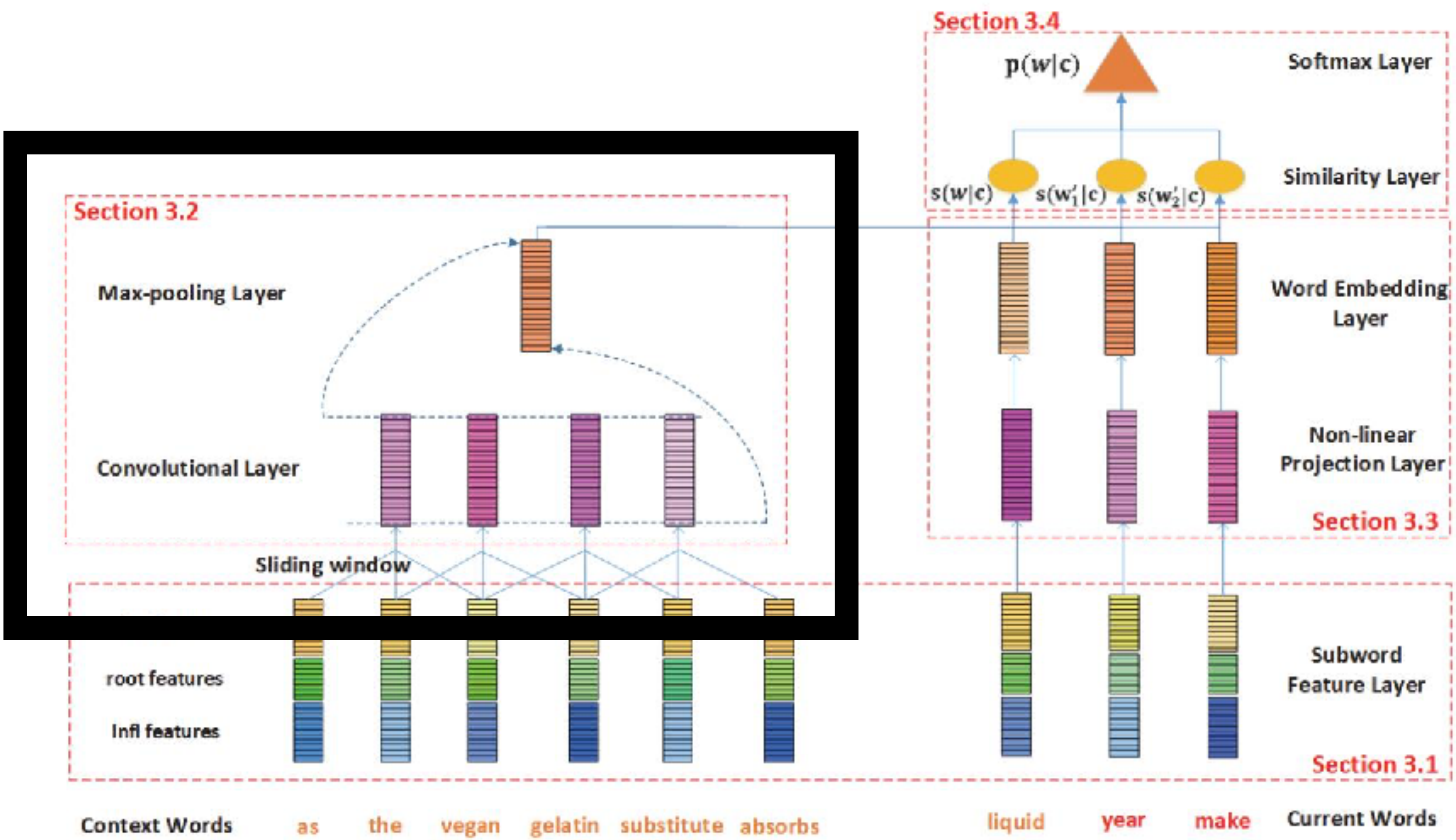


(a) Letter Trigrams

(b) Root and Derivational Affixes

(c) Inflectional Affixes

Learning Context Embedding



Learning Context Embedding

- **Convolutional layer**

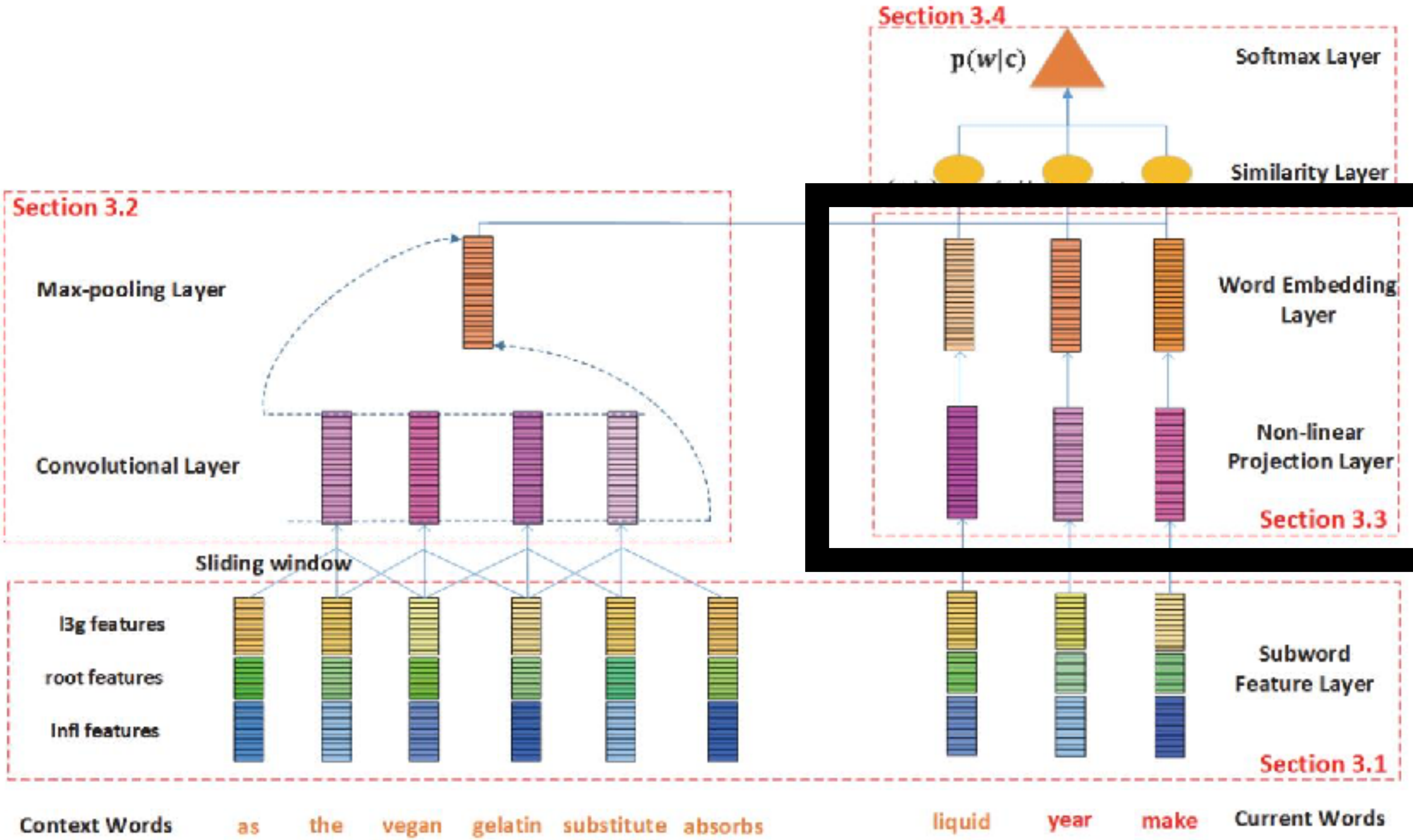
$$y_i = \sigma(\varpi \tilde{x}_i + \xi)$$

$$\tilde{x}_i = x_{i:i+n-1} = [x_i^T, x_{i+1}^T, \ldots, x_{i+n-1}^T]^T$$

- **Max pooling**

$$c(j) = \max_{i=1,2,\ldots,t-n+1} \{y_i(j)\}$$

Learning Current Word Embedding



Learning Current Word Embedding

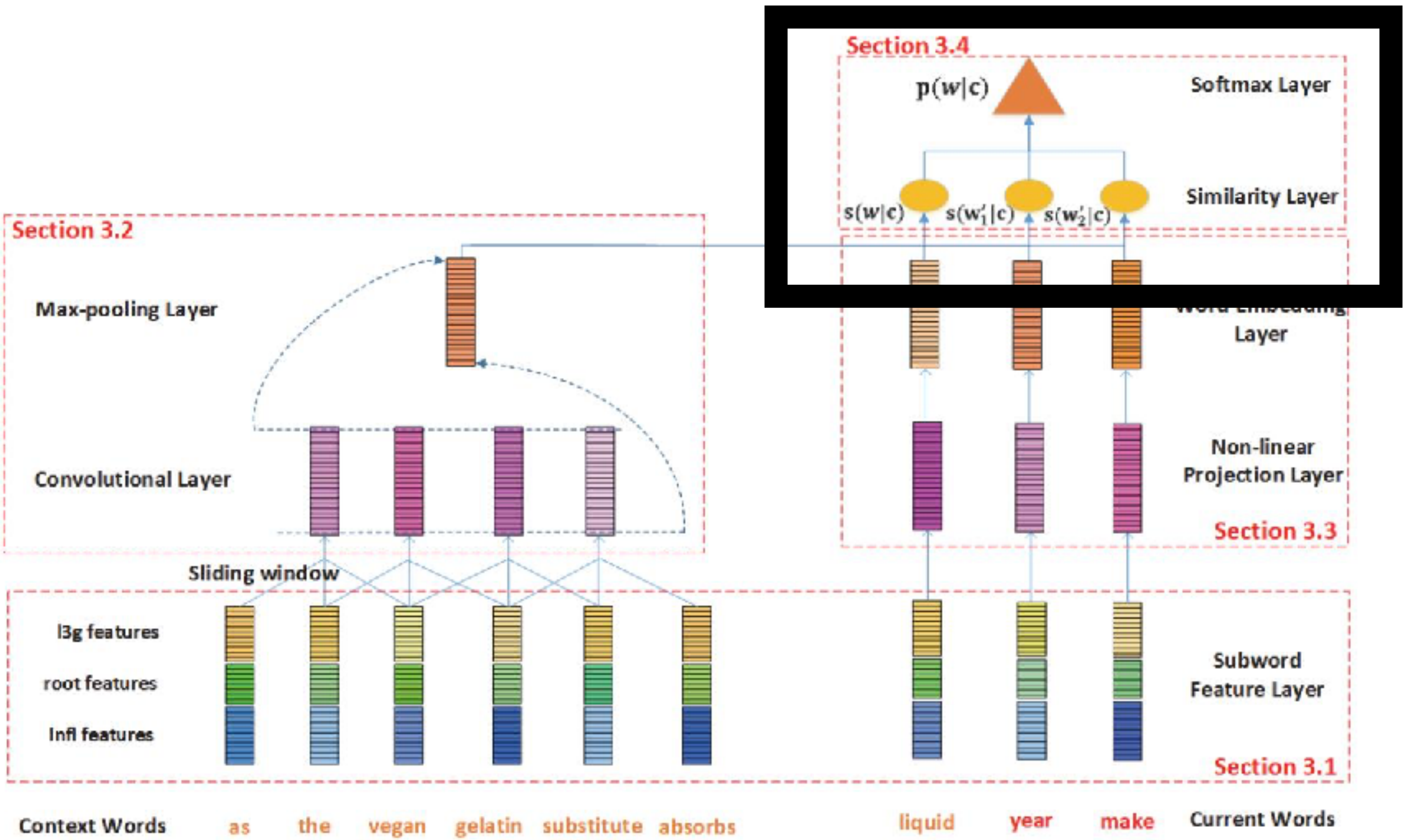
- **First full layer**

$$q = \sigma(\varsigma^1 x + \tau^1)$$

- **Second full layer**

$$w = \sigma(\varsigma^2 q + \tau^2)$$

Conditional Probability Based on Softmax



Conditional Probability Based on Softmax

$$p(\mathbf{w}|\mathbf{c}) = \frac{\exp(\gamma \cdot s(w,c))}{\exp(\gamma \cdot s(w,c)) + \sum_{\mathbf{w}'_j \in \mathbb{W}} \exp(\gamma \cdot s(w'_j, c))}$$

- **s(w,c) :similarity function**
- **w: the current word; c: context word**

$$l(\mathbf{w}, \mathbf{c}; \theta) = -\log p(\mathbf{w}|\mathbf{c}) = \log(1 + \sum_j \exp(-\gamma \cdot \Delta_j(w,c)))$$

$$\Delta_j(w,c) = s(w,c) - s(w'_j, c)$$

$$L(\theta) = \sum_{(\mathbf{w},\mathbf{c}) \in \mathcal{D}} \log(1 + \sum_j \exp(-\gamma \cdot \Delta_j(w,c)))$$

Experiment

- **Word Similarity**
  - **DataSets: WordSim353、MEN、MT、Rel122、RG**
  - **Pairs of words together with their similarity scores**
- **Word Analogy**
  - **DataSets: 3CosAdd and 3CosMul**
  - **“a** is to **b** that is similar to **u** is to **v”**

Word Similarity

Model \ $\rho \times 100$	F. et al. WS353	B. et al. MEN	R. et al. MT	S. et al. Rel122	R. et al. RG
skipgram (neg=10)	63.2	59.8	61.8	53.3	64.0
skipgram (neg=100)	59.5	58.1	60.8	53.8	64.1
char n-gram	59.5	21.7	60.3	51.0	51.3
GloVe	62.6	65.1	60.2	48.8	58.1
DSSM	51.1	41.5	44.1	31.6	37.7
Our Model	**65.7**	**69.0**	**64.8**	**57.6**	**72.7**

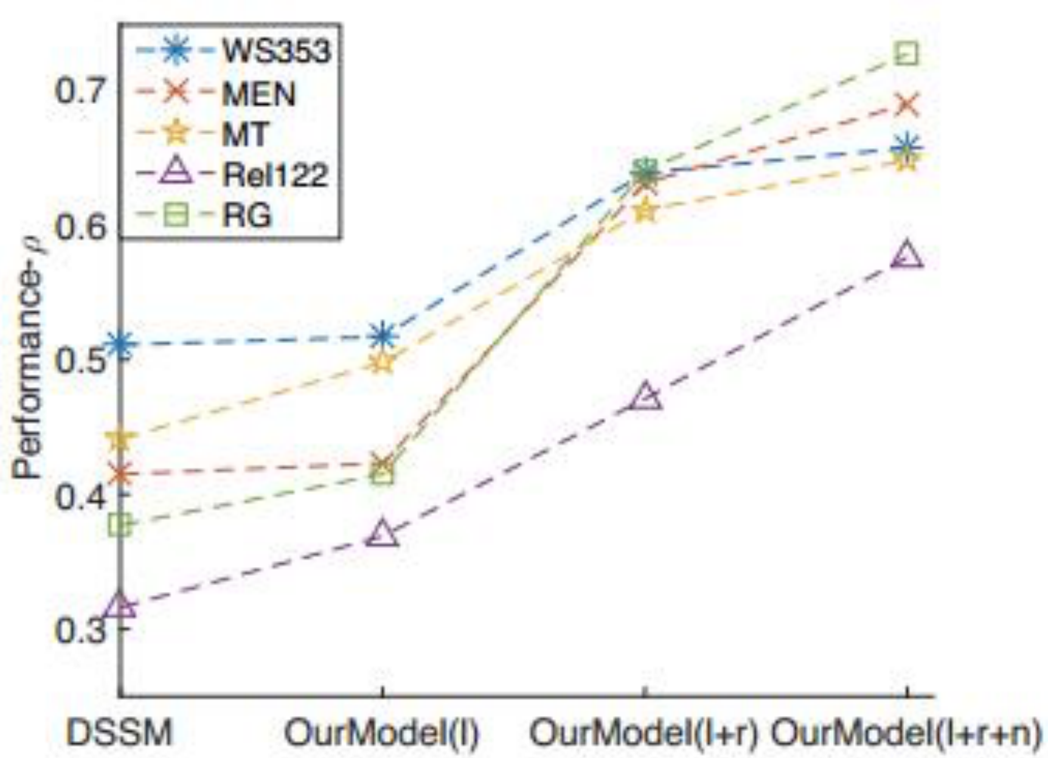
Table 1: Performance on word similarity task

Effect of Convolutional Feature learning

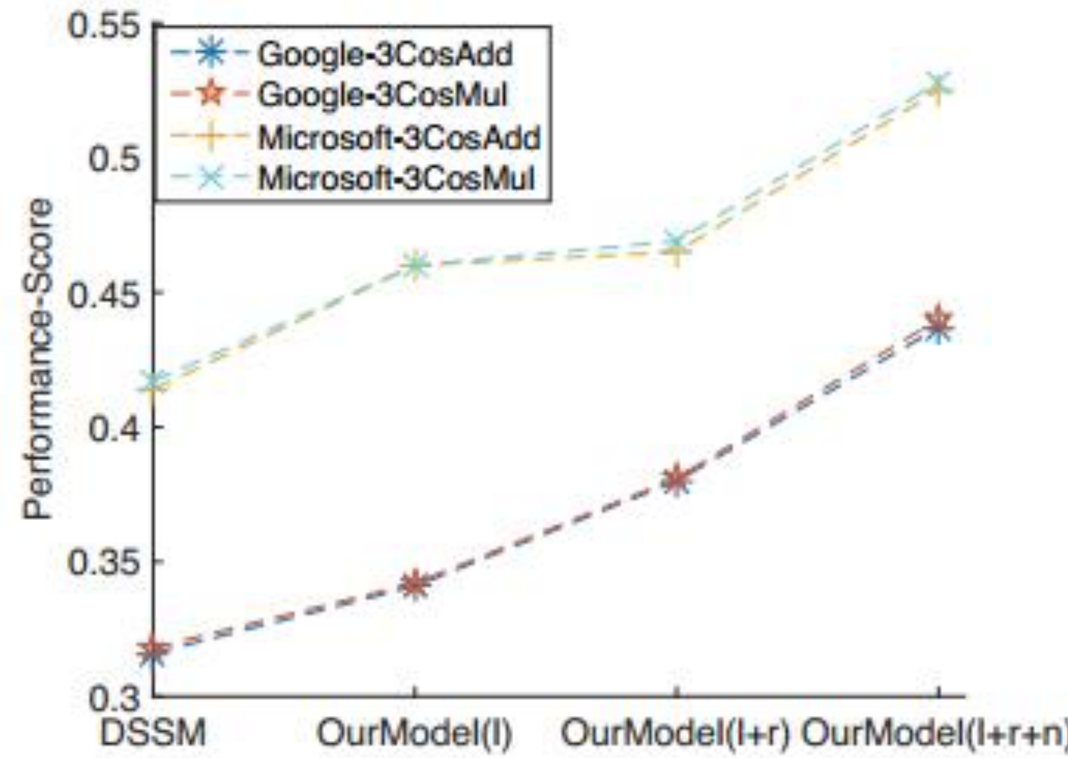
	Google		Microsoft	
Model	3CosAdd	3CosMul	3CosAdd	3CosMul
DSSM	31.6	31.8	41.4	41.7
DSSM (l+r+n)	41.7	41.7	48.5	48.9
Our Model (l+r+n)	**43.7**	**44.0**	**52.5**	**52.8**

Table 3: Performance comparison (on word analogy task) between Our Model (l+r+n) and DSSM (l+r+n) when all subword features are considered

Importance of subword information



(a) Performance on Word Similarity



(b) Performance on Word Analogy

Neural Ranking Models with Weak Supervision

SIGIR 2017
University of Amsterdam
Mostafa Dehghani

Motivation

- **A neural ranking model using weak supervision**
  - **Labels: Unsupervised ranking model, such as BM25**
- **Various learning scenarios**
  - **point-wise & pair-wise models**
- **Different input representation**

Weak Supervision for Ranking

- **Pseudo-Labeling**
  - **pseudo-labeler: Unsupervised ranking model, such as BM25**
- **A set of neural network-based ranking models**

Ranking Architectures

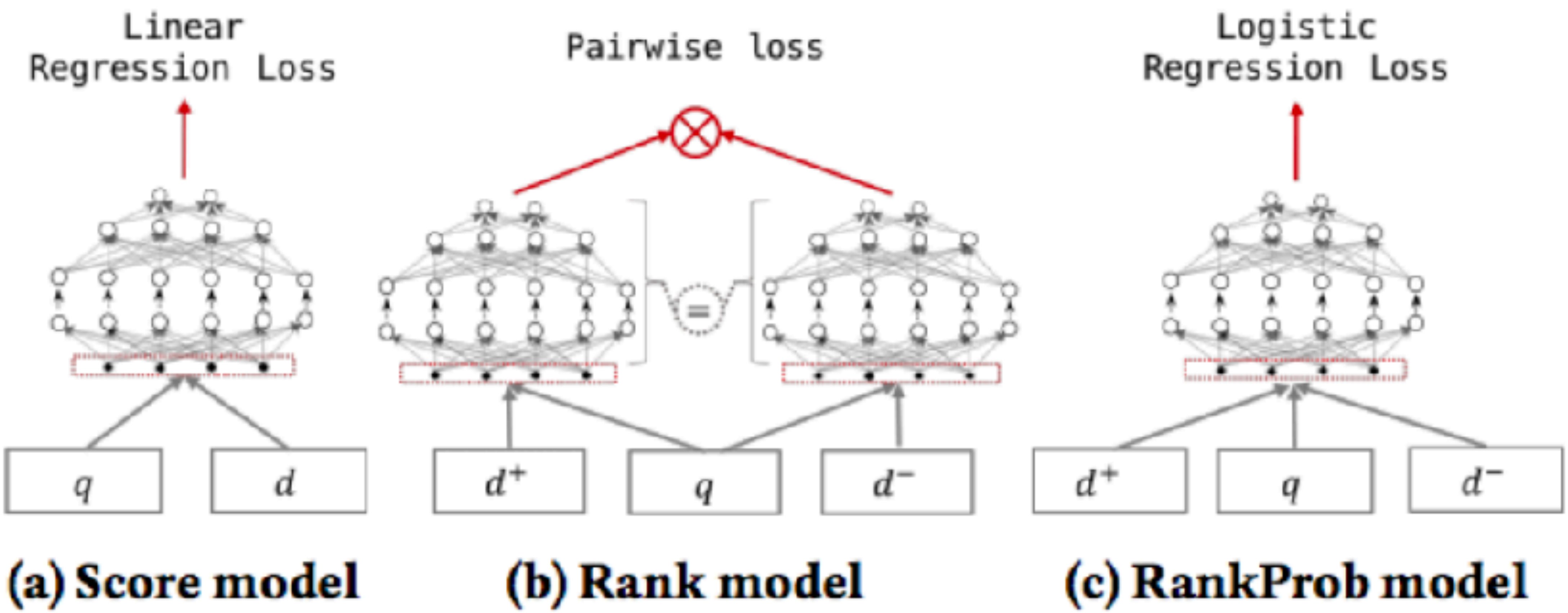


Figure 1: Different Ranking Architectures

Score model

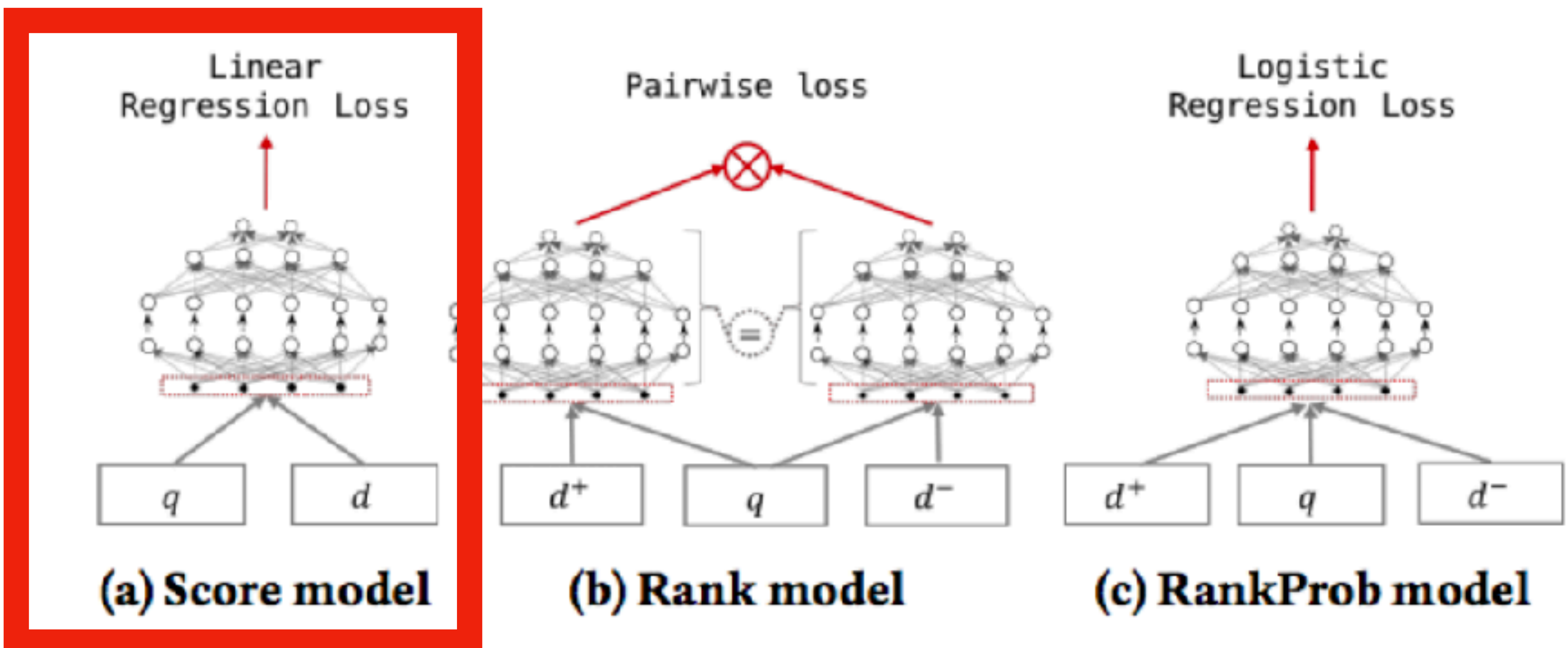


Figure 1: Different Ranking Architectures

Score model

- **Loss function**

$$\mathcal{L}(b;\theta)=\frac{1}{|b|}\sum_{i=1}^{|b|}(\mathcal{S}(\{q,d\}_i;\theta)-s_{\{q,d\}_i})^2$$

- **|b| : batch b**
- $S(q,d;\theta)$ **: retrieval score**
- $s_{\{q,d\}_i}$ **: the relevance score**
- $\tau=(q,d,s_{q,d})$ **: instance**

Rank model

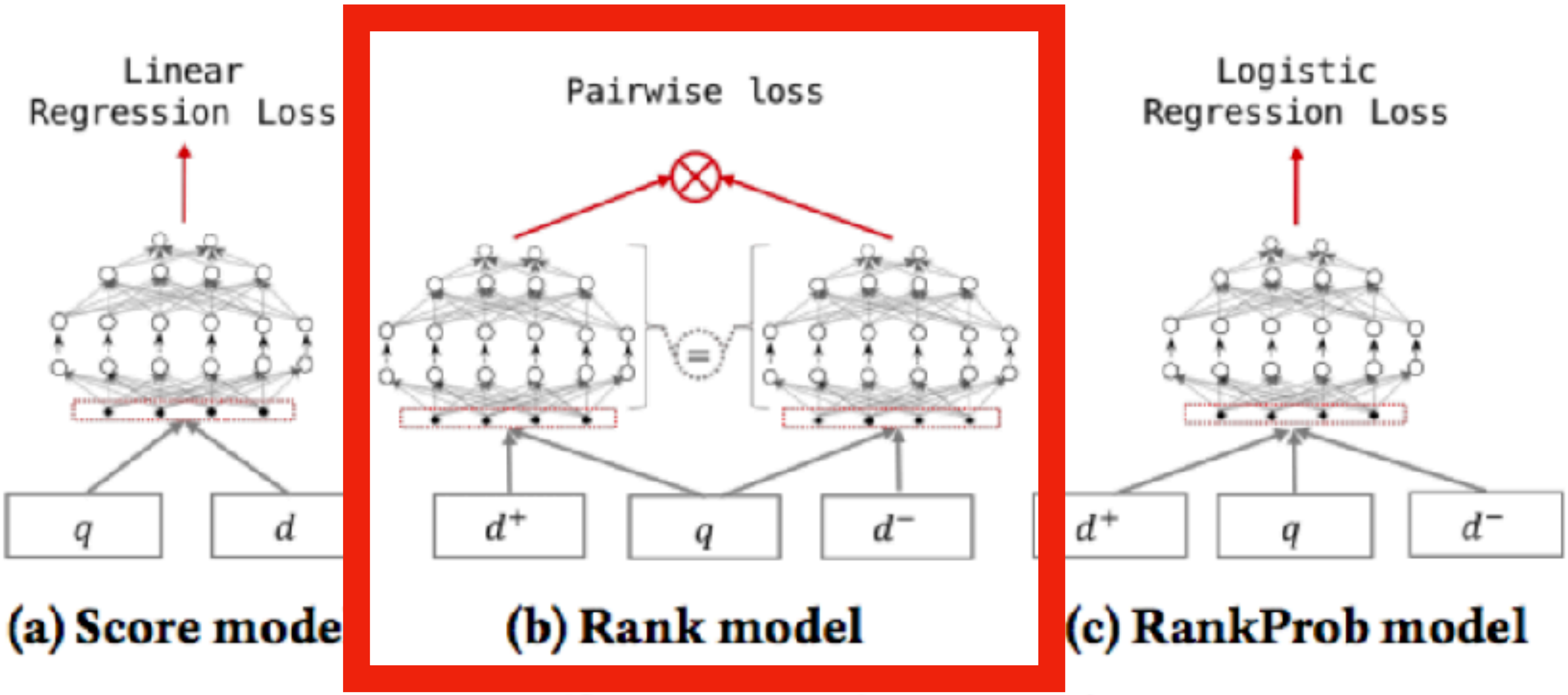


Figure 1: Different Ranking Architectures

Rank model

- **Loss function**

$$\mathcal{L}(b;\theta)=\frac{1}{|b|}\sum_{i=1}^{|b|}\max\{0,\varepsilon-\mathrm{sign}(s_{\{q,d_1\}_i}-s_{\{q,d_2\}_i})$$
$$(\mathcal{S}(\{q,d_1\}_i;\theta)-\mathcal{S}(\{q,d_2\}_i;\theta))\}$$

- **Compress the outputs to the range of [-1,1]**
- $\tau=(p,d_1,d_2,s_{qd_1},s_{qd_2})$ **: instance**

RankProb model

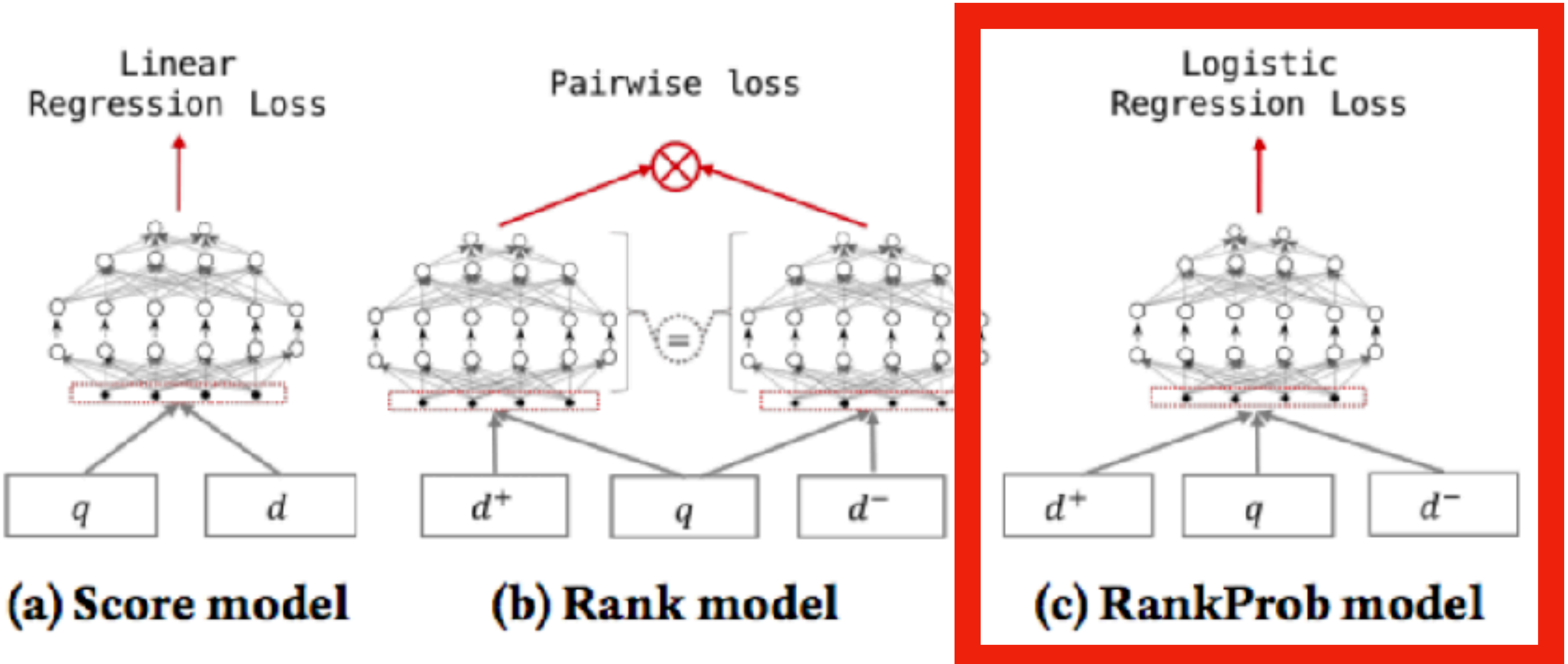


Figure 1: Different Ranking Architectures

RankProb model

- **Loss function_cross-entropy**

$$\mathcal{L}(b;\theta) = -\frac{1}{|b|}\sum_{i=1}^{|b|} P_{\{q,d_1,d_2\}_i}\log(\mathcal{R}(\{q,d_1,d_2\}_i;\theta)) + (1-P_{\{q,d_1,d_2\}_i})\log(1-\mathcal{R}(\{q,d_1,d_2\}_i;\theta))$$

- **The probability of document d1 being ranked higher than d2**

$$P_{\{q,d_1,d_2\}_i} = \frac{s_{\{q,d_1\}_i}}{s_{\{q,d_1\}_i} + s_{\{q,d_2\}_i}}$$

Input Representation

- **Dense vector representation (Dense)**
- **Sparse vector representation (Sparse)**
- **Embedding vector representation (Embed)**

Dense vector representation (Dense)

- **A dense feature vector composed of features used in traditional IR model**

$$\psi(q,d)=[N||avg(l_d)_D||l_d||\{df(t_i)||tf(t_i,d)\}_{1\leq i\leq k}]$$

- **D：the total number of the documents**
- **avg(ld): the average length of documents**
- **ld: the length of the document**
- **df(ti):the frequency of each term**
- **df(ti):document frequency of each term**

Sparse vector representation (Sparse)

- **Bag-of-word representation**

$$\psi(q,d) = [tfv_c || tfv_q || tfv_d]$$

- **tfvc: term frequency vector of collection**
- **tfvq: term frequency vector of query**
- **tfvd: term frequency vector of document**

Embedding vector representation (Embed)

- **Word Embedding**

$$\psi(q,d)=[\odot_{i=1}^{|q|}(\mathcal{E}(t_i^q),\mathcal{W}(t_i^q))||\odot_{i=1}^{|d|}(\mathcal{E}(t_i^d),\mathcal{W}(t_i^d))]$$

$$\odot_{i=1}^{n}(\mathcal{E}(t_i),\mathcal{W}(t_i))=\sum_{i=1}^{n}\widehat{\mathcal{W}}(t_i)\cdot\mathcal{E}(t_i)$$

$$\widehat{\mathcal{W}}(t_i)=\frac{\exp(\mathcal{W}(t_i))}{\sum_{j=1}^{n}\exp(\mathcal{W}(t_j))}$$

Experiment

- **Results**

Method	**Robust04**			**ClueWeb**		
	MAP	*P@20*	*nDCG@20*	*MAP*	*P@20*	*nDCG@20*
BM25	0.2503	0.3569	0.4102	0.1021	0.2418	0.2070
Score + Dense	0.1961▽	0.2787▽	0.3260▽	0.0689▽	0.1518▽	0.1430▽
Score + Sparse	0.2141▽	0.3180▽	0.3604▽	0.0701▽	0.1889▽	0.1495▽
Score + Embed	0.2423▽	0.3501	0.3999	0.1002	0.2513	0.2130
Rank + Dense	0.1940▽	0.2830▽	0.3317▽	0.0622▽	0.1516▽	0.1383▽
Rank + Sparse	0.2213▽	0.3216▽	0.3628▽	0.0776▽	0.1989▽	0.1816▽
Rank + Embed	**0.2811▲**	**0.3773▲**	**0.4302▲**	**0.1306▲**	**0.2839▲**	**0.2216▲**
RankProb + Dense	0.2192▽	0.2966▽	0.3278▽	0.0702▽	0.1711▽	0.1506▽
RankProb + Sparse	0.2246▽	0.3250▽	0.3763▽	0.0894▽	0.2109▽	0.1916
RankProb + Embed	**0.2837▲**	**0.3802▲**	**0.4389▲**	**0.1387▲**	**0.2967▲**	**0.2330▲**

Experiment

- **How useful is learning with weak supervision for supervised ranking?**

Method	**Robust04**			**ClueWeb**		
	MAP	*P@20*	*nDCG@20*	*MAP*	*P@20*	*nDCG@20*
Weakly supervised	0.2837	0.3802	0.4389	0.1387	0.2967	0.2330
Fully supervised	0.1790	0.2863	0.3402	0.0680	0.1425	0.1652
Weakly supervised + Fully supervised	**0.2912**$^{\blacktriangle}$	**0.4126**$^{\blacktriangle}$	**0.4509**$^{\blacktriangle}$	**0.1520**$^{\blacktriangle}$	**0.3077**$^{\blacktriangle}$	**0.2461**$^{\blacktriangle}$