Learning to Generate Question by Learning What not to Generate

DATE: 2019/12/13

ADVISOR : JIA-LING, KUO

SPEAKER : GUAN-YU, SHENG

SOURCE : WWW 2019

KDD

Outline

- Introduction
- Method
- Experiment
- conclusion

Goal

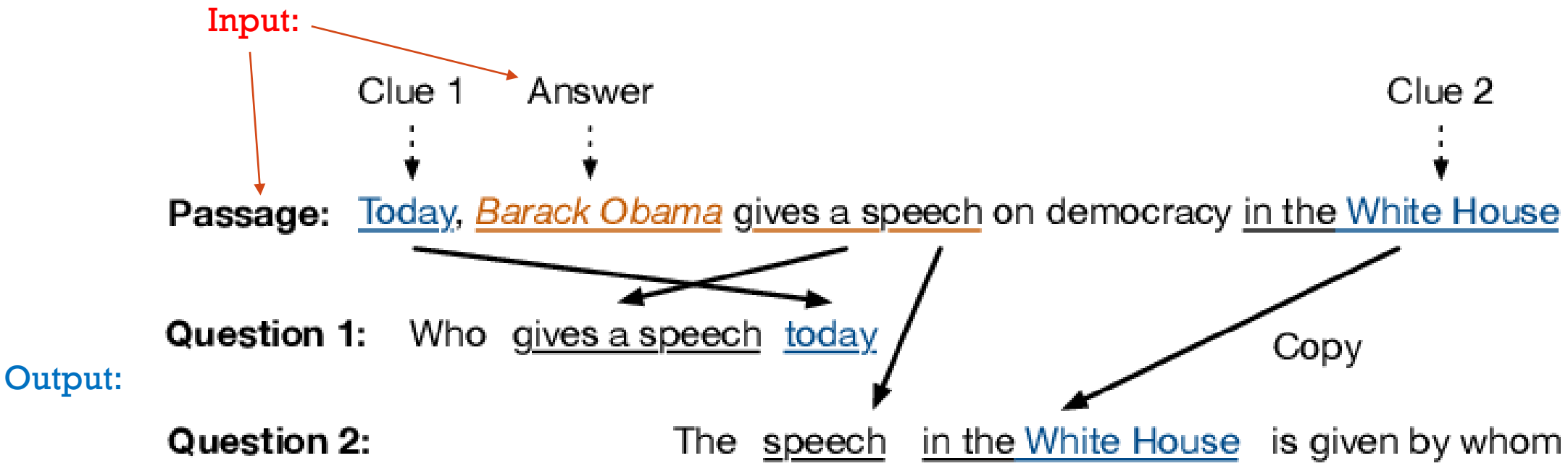


Figure 1: Questions are often asked by repeating some text

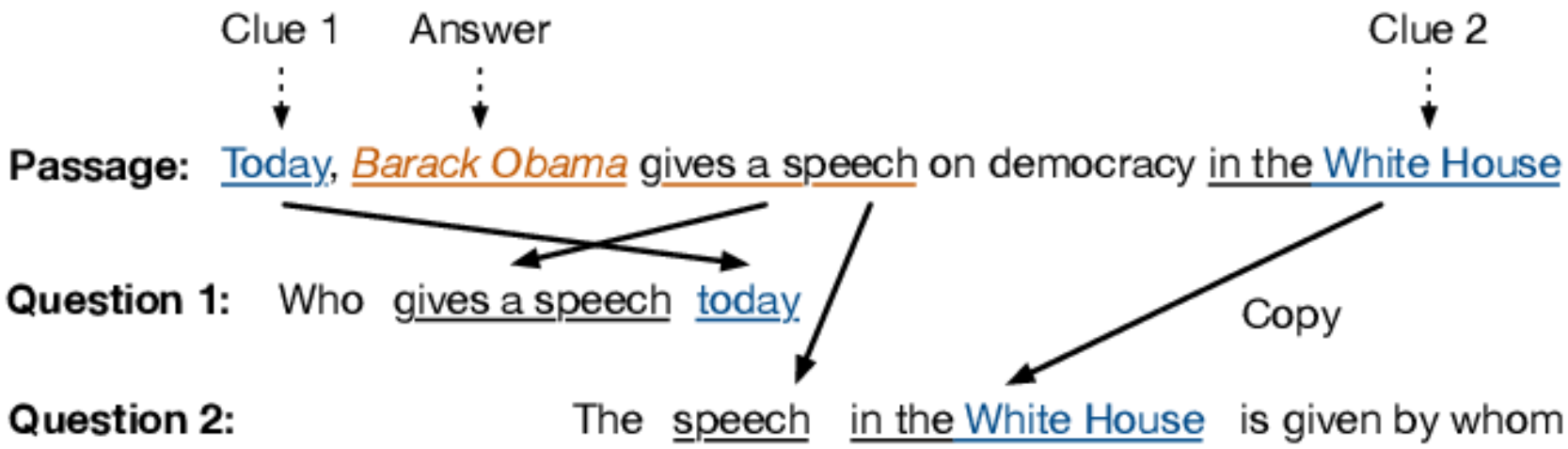
Motivation

- Existing neural question generation models :
  - Inability to properly model the process of how each word in the question is selected.
  - Fail to mark a clear boundary between the set of question words that should be directly copied from the input text and those that should be generated instead.

- We propose Clue Guided Copy Network for Question Generation:
  - Sequence-to-sequence generative model with copying mechanism.
  - Generate questions by learning to identify where each word in the question should come from(copy or generate).

Clue word

- Asking a question about a **passage** and a **given answer** is actually a "one-to-many" mapping problem.



- Solution : Using clue word to guide the way we ask.

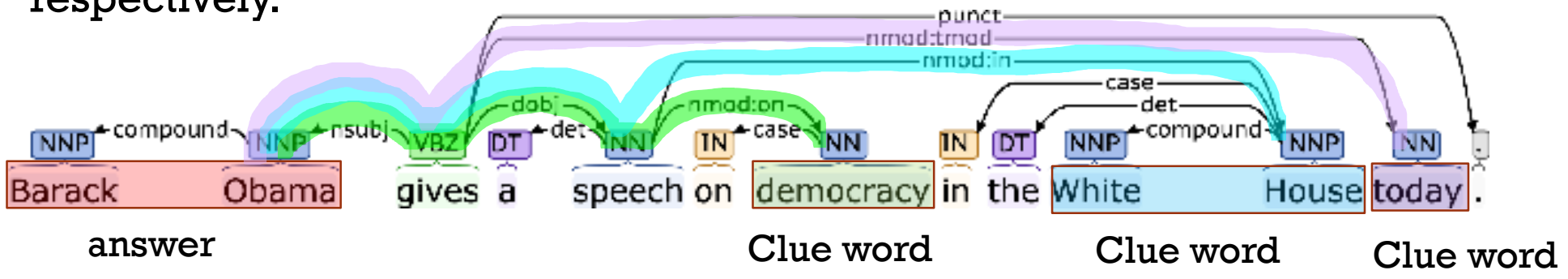
Clue word

- Clue words will be more closely connected to the answer in dependency trees
- The syntactical connection patterns can be learned via **graph convolutional networks**.

For example

the graph path distances between answer "**Barack Obama**" to "**democracy**", "**White House**", and "**today**" are **3**, **3**, and **2**

the corresponding word order distances between them are **5**, **8**, and **10**, respectively.



Copy word

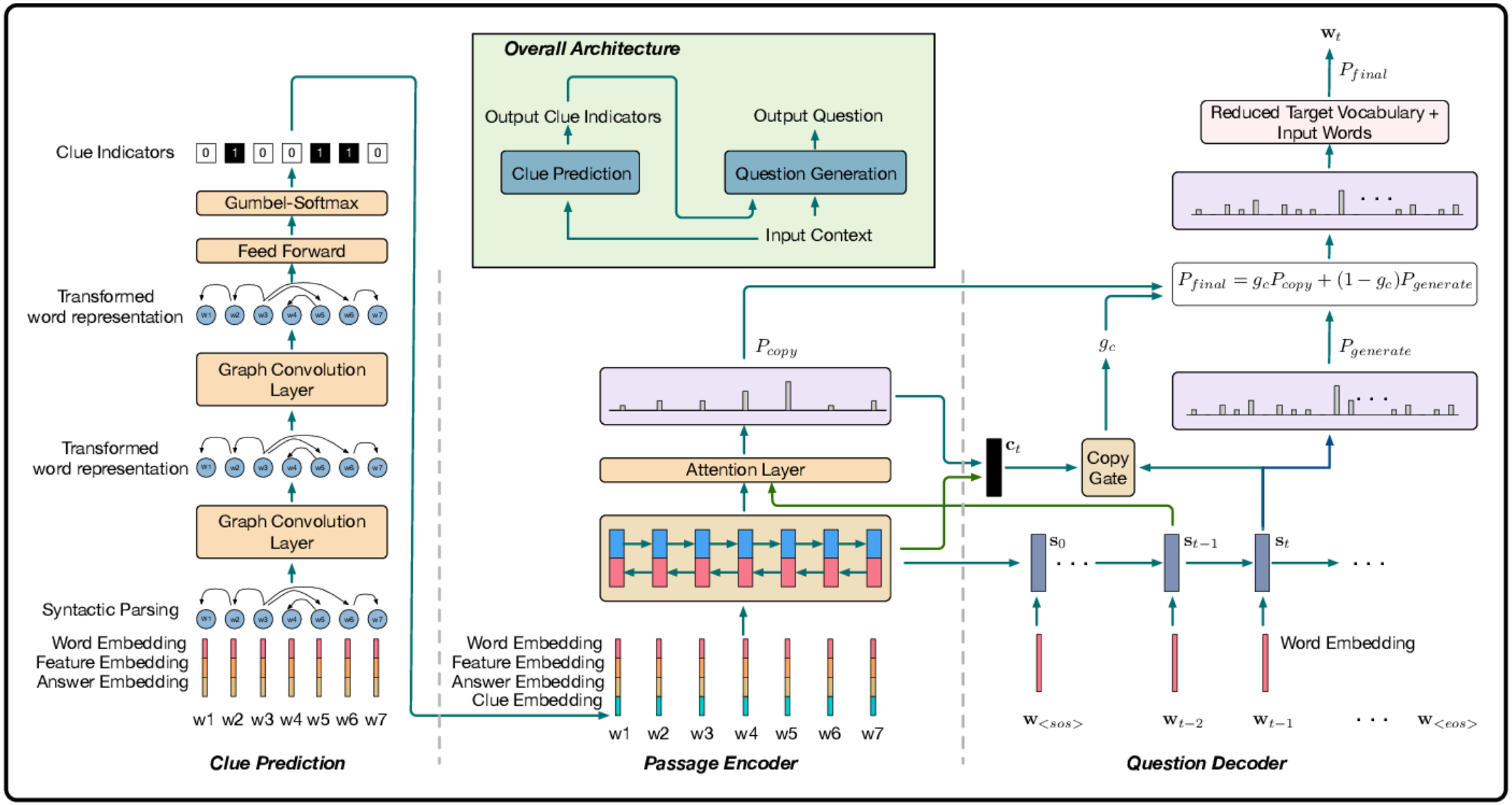
- Given a passage and a question in a training dataset :
- We label a word as copied word if it is
  - **Nonstop word** shared by both the **passage** and the **question.**
  - Its word frequency rank in the vocabulary is **lower than a threshold**.
    - Reduce the size of vocabulary
    - high-frequency word—be generated
    - Low-frequency word—be copied
- We then shortlist the output vocabulary based on frequency analysis of the non-copied question words (according to our labeling criteria).

Summary

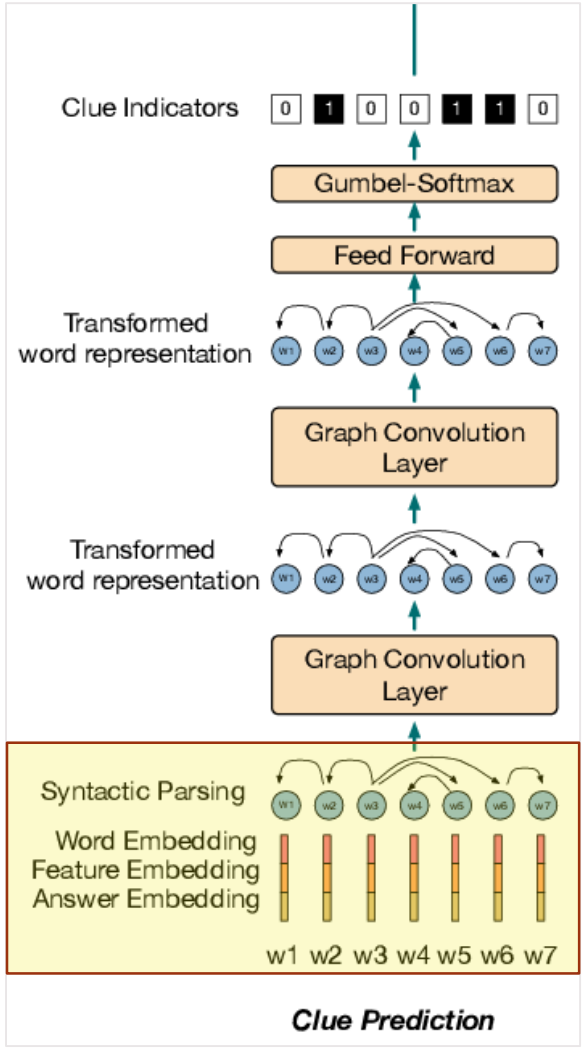
- Clue word:
  - To solving the 1-n answer-questions problem.
  - To guide the way we ask.
  - More closely connected to the answer in dependency trees.
  - Patterns can be learned via graph convolutional networks

- Copy word:
  - Labelled in training set.
  - Nonstop word shared by passage and question.
  - Word frequency rank in the vocabulary is lower than a threshold.

Method-model



Clue prediction

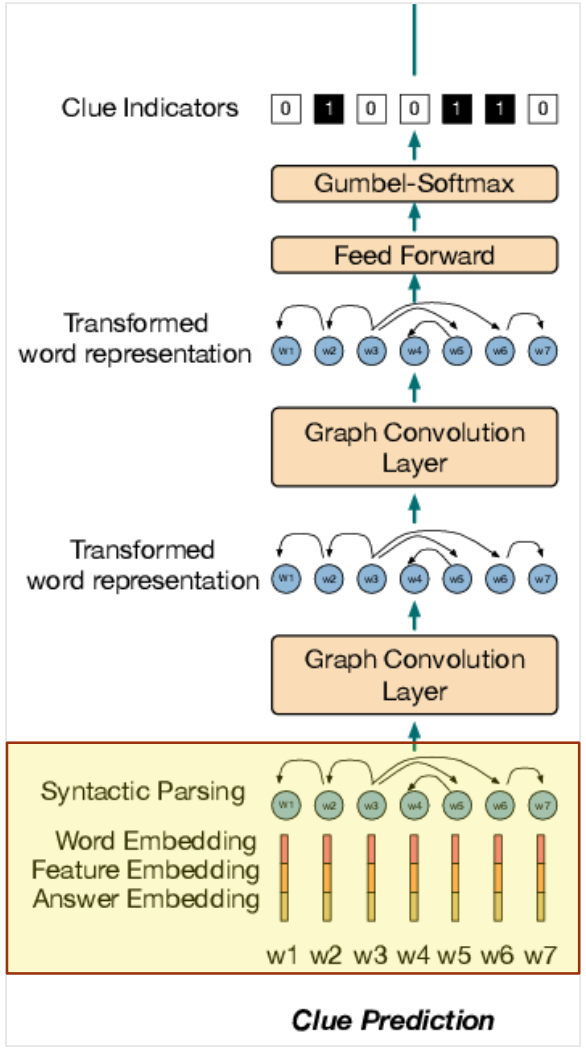


- Input :

  Each word is represented by vector concated by following 5 features

  1. Word Vector: (Glove embedding)
  2. Lexical Features:
     - Named Entity Recognition(NER)
     - Part-of-Speech (POS) tagging
     - DependencyParsing (DEP-using spaCy)
  3. Binary Features

Clue prediction



4. Answer Position:

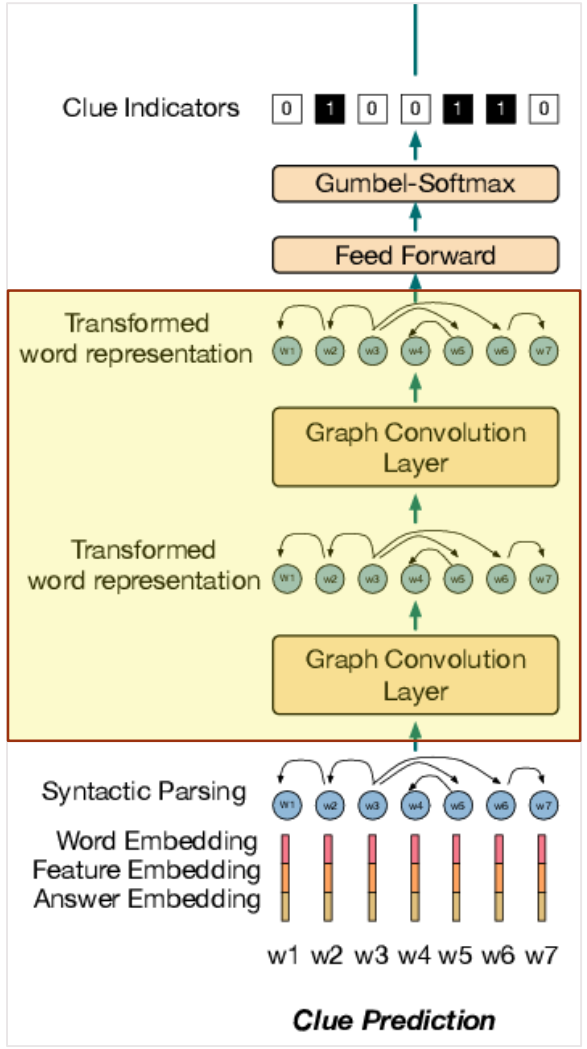
   B/I/O: tagging scheme to label the position of a given answer:

   - B : where a word at the beginning of an answer
   - I : denotes the continuation of the answer
   - O: words not contained in an answer.

5. Word Frequency Feature:

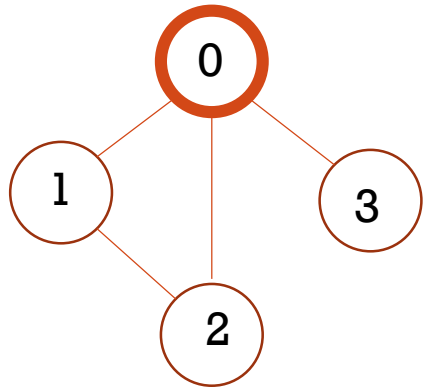
   - Rank all the words(passages and questions in the training dataset).
   - Each word will be assigned tag: L ,H or M.

Clue prediction



- Use *graph convolutional networks* to learn the pattern of clue word.
  - Input:
    - Representing each dependency tree into its corresponding adjacency matrix form.
    - Representing input feature vectors into its corresponding vector matrix form.
  - Output:
    - the hidden vector representation of each word

GCN example



Adjacency matrix: $\tilde{\mathbf{A}} = \mathbf{A} + \mathbf{I}_N$ where $\mathbf{I}_N$ is an $n \times n$ identity matrix.

i \ j	0	1	2	3
0	0	1	1	1
1	1	0	1	0
2	1	1	0	0
3	1	0	0	0

$\tilde{\mathbf{A}}$:

i \ j	0	1	2	3
0	1	1	1	1
1	1	1	1	0
2	1	1	1	0
3	1	0	0	1

$$h_0^{(1)} = \sigma(\mathbf{W}^{(1)} \text{sum} \begin{bmatrix} \text{J=1 , } \underline{1 * [0, 0]} \\ \text{J=2 , } \underline{1 * [1, -1]} \\ \text{J=3 , } \underline{1 * [2, -2]} \\ \text{J=4 , } \underline{1 * [3, -3]} \end{bmatrix} / 4 + b^{(1)})$$

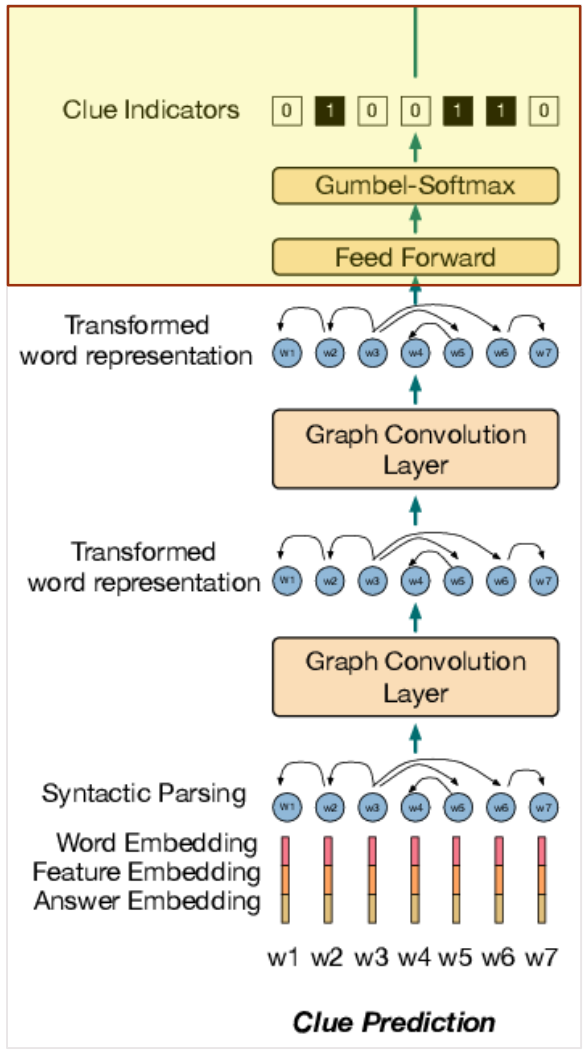
Assume Feature matrix:

i \F	0	1
0	0	0
1	1	-1
2	2	-2
3	3	-3

$d_i = \sum_{j=1}^{N} \tilde{\mathbf{A}}_{ij}$ is the degree of node

$$h_i^{(l)} = \sigma(\sum_{j=1}^{N} \tilde{\mathbf{A}}_{ij} \mathbf{W}^{(l)} h_j^{(l-1)} / d_i + b^{(l)})$$

Clue prediction



- We wanna get a one-hot vector represent whether a word is a clue word, but one-hot representation is undifferentiable.
- Solution : we use Gumbel-softmax to be the activation function.

probability pi of class i is defined as: $p_i = \frac{\exp(\log(\pi_i))}{\sum_{j=1}^{k} \exp(\log(\pi_j))}$

$$z_i = \begin{cases} 1, & i = \arg\max_j (log(\pi_j) + g_j) \\ 0, & \text{otherwise} \end{cases}$$

$$g_i = -\log(-\log(u_i)),$$

$$u_i \sim \text{Uniform}(0, 1)$$

Clue prediction

- Proof of Gumbel Max:

- Assume that the output of the Gembel Max is p_1:
  It means 'log p_1−log(−logε1)' is the largest (in p_1-pk)
  So we can get :

  logp_1−log(−logε1)>logp_2−log(−logε2)
  logp_1−log(−logε1)>logp_3−log(−logε3)
  ⋮
  logp_1−log(−logε1)>logp_k−log(−logεk)

$$z_i = \begin{cases} 1, & i = \arg\max_j (log(\pi_j) + g_j) \\ 0, & \text{otherwise} \end{cases}$$
$$g_i = -\log(-\log(u_i)),$$
$$u_i \sim \text{Uniform}(0, 1)$$

- From logp_1−log(−logε1)>logp2−log(−logε2)
  we get: ε2 < ε1$^{p2/p1}$≤ 1
  ε2~U[0,1], so the probability of (ε2<ε1$^{p2/p1}$) is ε1$^{p2/p1}$
  The probability that all inequalities hold at the same time is:

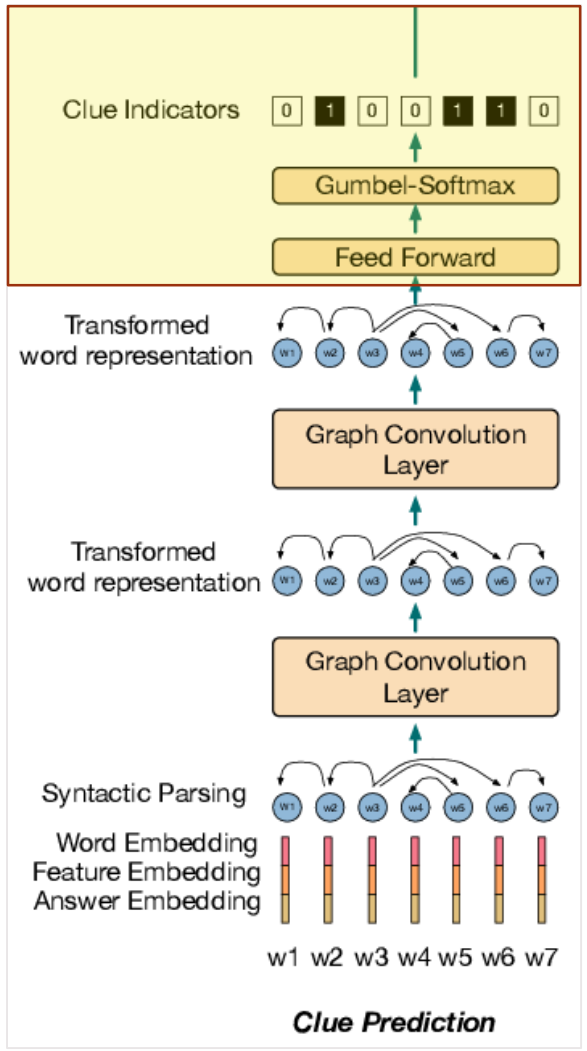
$$\varepsilon_1^{p_2/p_1}\ \varepsilon_1^{p_3/p_1}\ \cdots \varepsilon_1^{p_k/p_1} = \varepsilon_1^{(p_2+p_3+\cdots+p_k)/p_1} = \varepsilon_1^{(1/p_1)-1}$$

- Then average all ε1:

$$\int_0^1 \varepsilon_1^{(1/p_1)-1}\, d\varepsilon_1 = p_1$$

Clue prediction

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)}$$

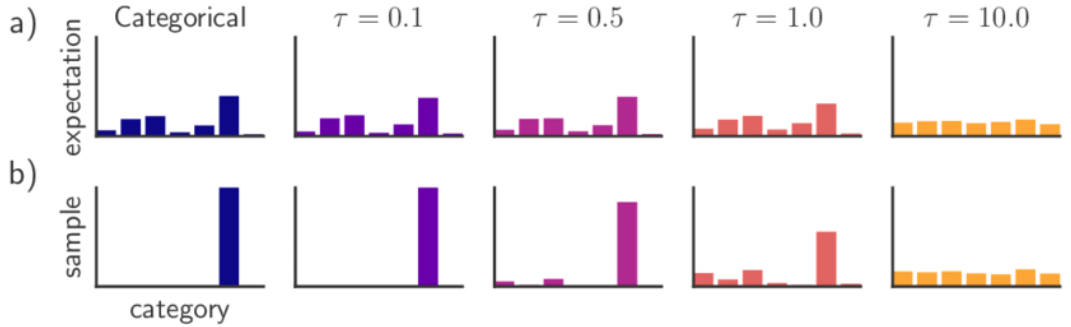


- Parameter τ :

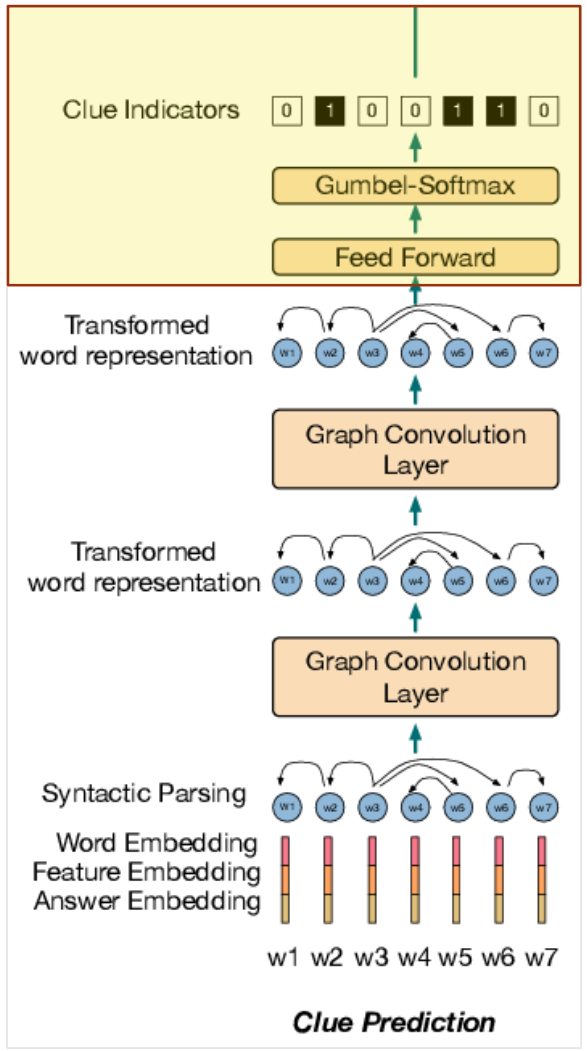
  The lower the parameter is, the representation closer to one-hot.

$$y_i = \frac{\exp((\log(\pi_i) + g_i)/\tau)}{\sum_{j=1}^{k} \exp((\log(\pi_j) + g_j)/\tau)}$$

- Gumbel-Softmax distribution replaces the "argmax" function by differentiable softmax function.
- Therefore, a sample : y = ($y1$, ・ ・ ・ , yk) drawn from Gumbel-Softmax distribution is given by:



Clue prediction



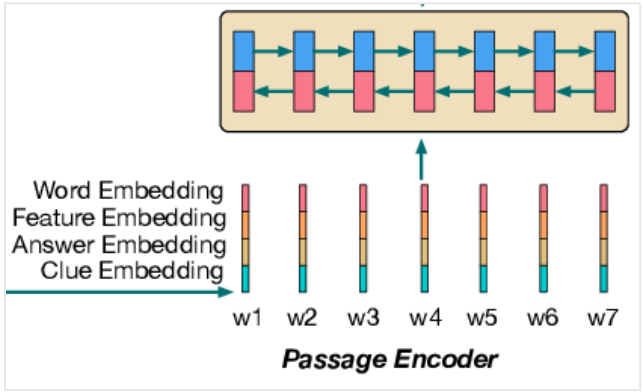
- Straight-Through Gumbel-Softmax:
  - Forward and backward propagation
  - Forward pass : It discretizes a continuous probability vector y sampled from the Gumbel-Softmax distribution into a one-hot vector

$$y^{ST} = (y_1^{ST}, \cdot \cdot \cdot, y_k^{ST})$$

$$y_i^{ST} = \begin{cases} 1, & i = argmax_j\ y_j\ , \\ 0, & \text{otherwise.} \end{cases}$$

  - Backward pass: It uses the continuous y, so that the error signal can still backpropagate.

Passage Encoder

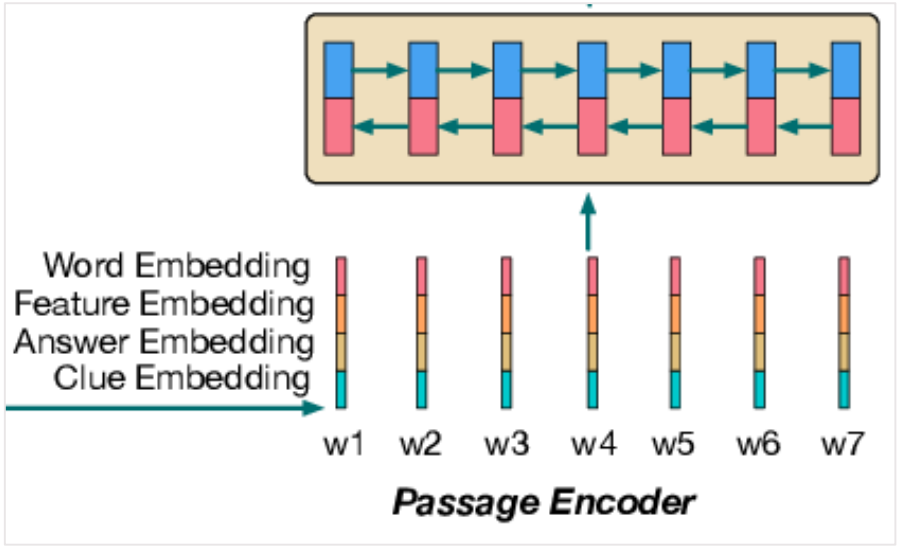


- Input :

  Each word is represented by vector concated by following 6 features

  1. Word Vector (Gloveembedding)
  2. Lexical Features
  3. Binary Features
  4. Answer Position
  5. Word Frequency Feature
  6. Clue Word Feature:
     - the clue predictor assigns a binary value to each word to indicate whether it is a potential clue word or not.

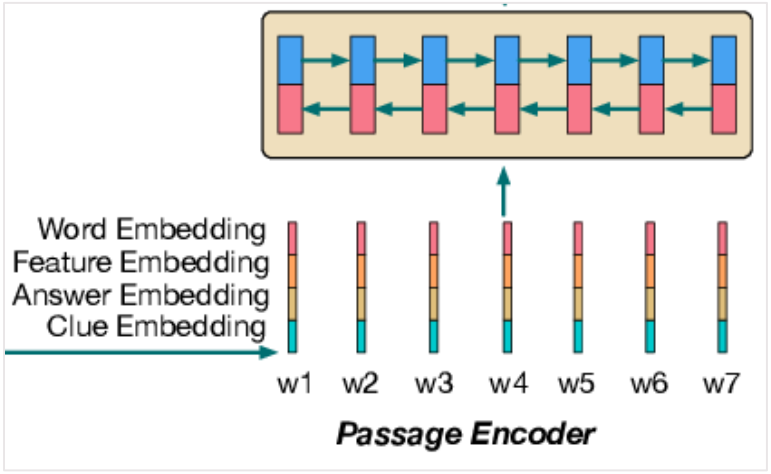
Passage Encoder



- GRU:
  - Each hidden state is a concatenation of a forward representation and a backward representation:

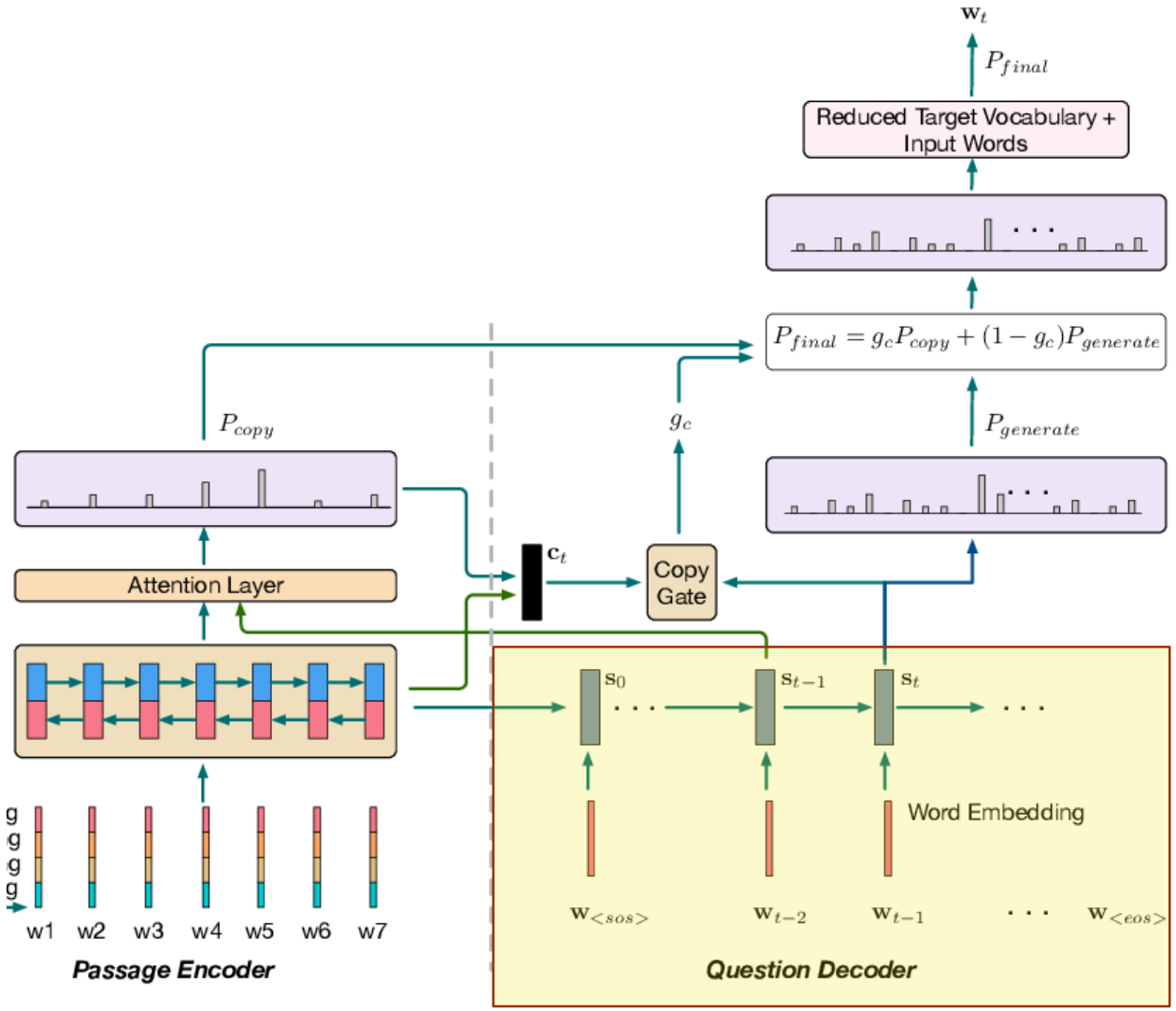
$$h_i = [\overrightarrow{h}_i; \overleftarrow{h}_i],$$
$$\overrightarrow{h}_i = \mathbf{BiGRU}(w_i, \overrightarrow{h}_{i-1}),$$
$$\overleftarrow{h}_i = \mathbf{BiGRU}(w_i, \overleftarrow{h}_{i+1}),$$

Passage Encoder



- Masking strategy:
  - Replace the word embeddings of low-frequency words with a special !l. token.
  - The augmented tagging features of a low-frequency word tend to be more influential than the word meaning in question generation.
  - The number of parameters that need be learned is largely reduced.
  - It improves training by reducing the model complexity.

Passage Decoder



Decoder GRU:

Initialize:

$$s_0 = tanh(\mathbf{W}_0 \overleftarrow{h}_1 + b).$$

$$s_t = \mathbf{GRU}([w_{t-1}; c_{t-1}], s_{t-1}).$$

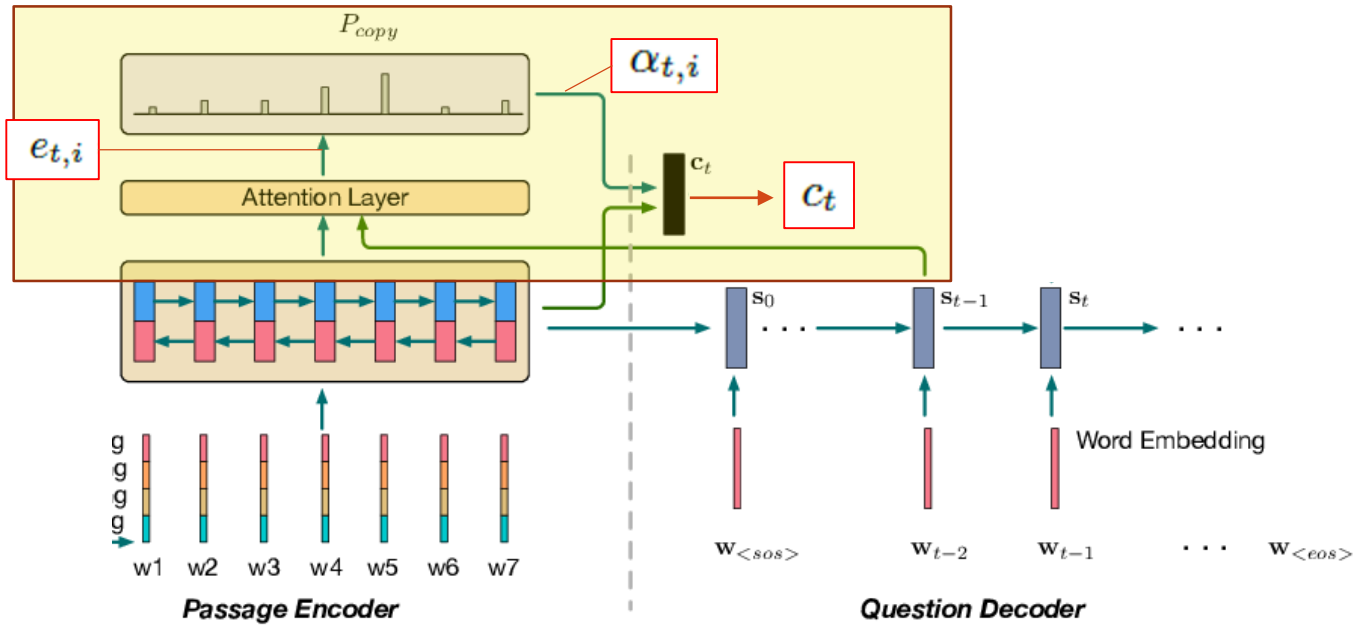
W_{t-1}: previous word embedding

C_{t-1}: previous attention context vector

Passage Decoder

Decoder Attention:

- At time step t, the attention weights and the context vector are calculated as:



$$e_{t,i} = v^{\intercal} tanh(\mathbf{W}_s s_t + \mathbf{W}_h h_i),$$

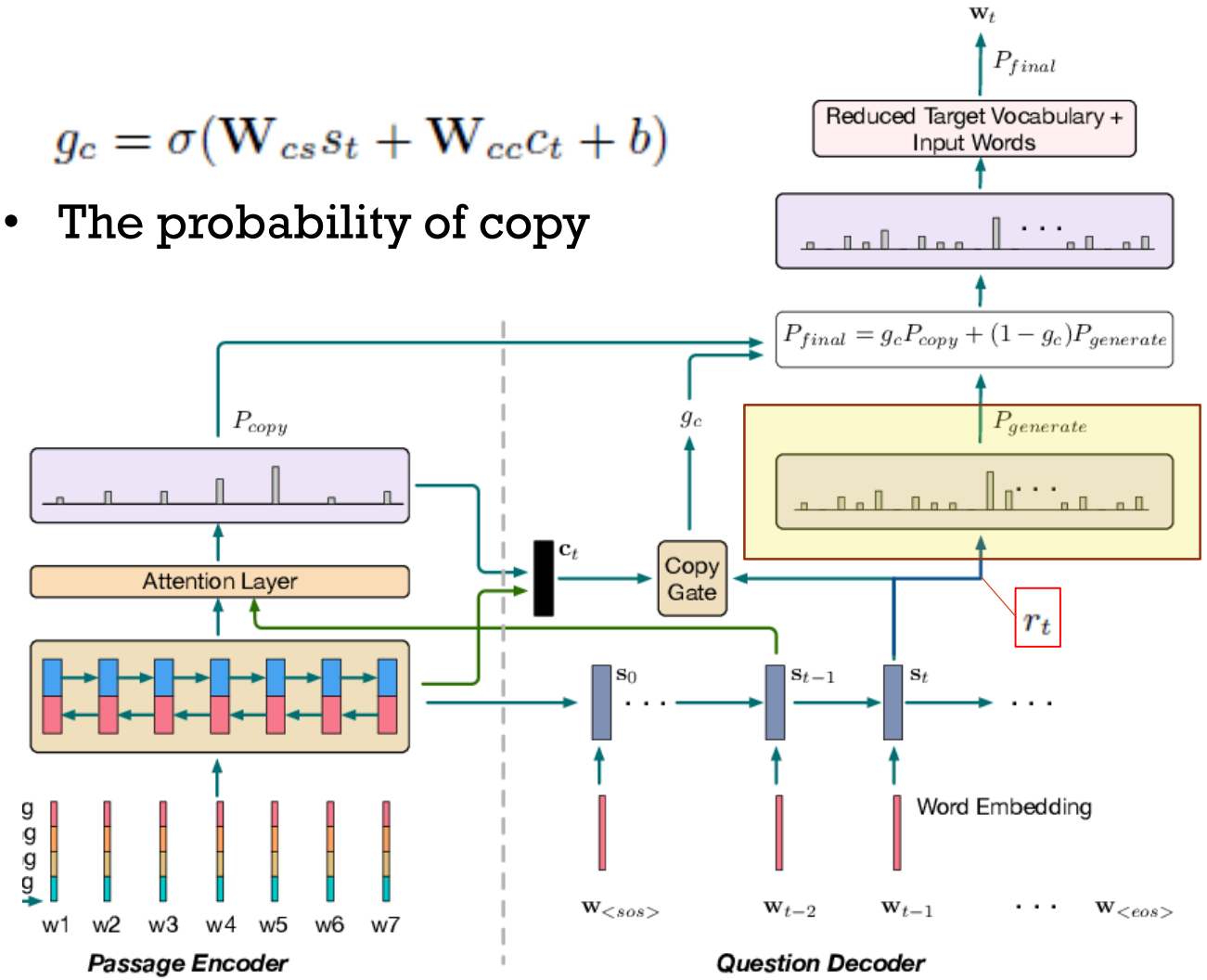
$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{j=1}^{|P|} \exp(e_{t,j})},$$

$$c_t = \sum_{i=1}^{|P|} \alpha_{t,i} h_i.$$

Passage Decoder

$$g_c = \sigma(\mathbf{W}_{cs} s_t + \mathbf{W}_{cc} c_t + b)$$

- The probability of copy

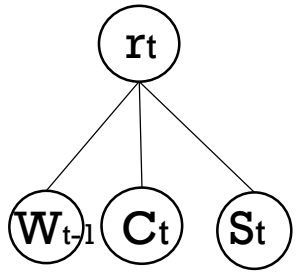


$$r_t = \mathbf{W}_{rw} w_{t-1} + \mathbf{W}_{rc} c_t + \mathbf{W}_{rs} s_t$$

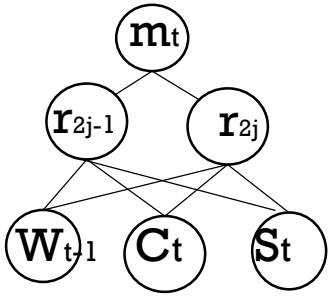
W_{t-1} : previous word embedding
C_t : current attention context vector
S_t : current decoder state

$$m_t = [\max\{r_{t,2j-1}, r_{t,2j}\}]_{j=1,\ldots,d}^{\intercal}$$

MLP:



MLP+Maxout: k=2



$$p(y_t | y1, \cdots, y_{t-1}) = \text{softmax}(\mathbf{W}_o m_t)$$

- Predict the probabilities of the next word over the decoder vocabulary

Experiment

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR
PCFG-Trans*	28.77	17.81	12.64	9.47	31.68	18.97
SeqCopyNet	–	–	–	13.02	44.00	–
seq2seq+z+c+GAN	44.42	26.03	17.60	13.36	40.42	17.70
NQG++*	42.36	26.33	18.46	13.51	41.60	18.18
MPQG	–	–	–	13.91	–	–
Answer-focused Position-aware model	43.02	28.14	20.51	15.64	–	–
s2sa-at-mp-gsa	44.51	29.07	21.06	15.82	44.24	19.67
ASs2s	–	–	–	16.17	–	–
CGC-QG (no feature-rich embedding)	45.50	29.63	21.58	16.38	43.11	20.52
CGC-QG (no target reduction)	45.80	30.27	22.29	17.05	44.09	20.79
CGC-QG (no clue prediction)	45.58	30.07	22.08	16.80	44.51	20.80
CGC-QG	**46.58**	**30.90**	**22.82**	**17.55**	**44.53**	**21.24**

Experiment

Model	BLEU-1	BLEU-2	BLEU-3	BLEU-4	ROUGE-L	METEOR
PCFG-Trans*	16.90	7.94	4.72	3.08	23.78	13.74
MPQG*	35.70	17.16	9.64	5.65	39.85	14.13
NQG++*	40.33	22.47	14.83	9.94	42.25	16.72
CGC-QG (no feature-rich embedding)	39.35	22.10	14.36	9.99	42.00	16.60
CGC-QG (no target reduction)	40.00	22.84	15.01	10.52	42.33	16.89
CGC-QG (no clue prediction)	39.85	22.82	14.96	10.45	43.16	17.11
CGC-QG	**40.45**	**23.52**	**15.68**	**11.06**	**43.16**	**17.43**

Conclusion & Summary

- We demonstrate the effectiveness of teaching the model to make decisions during the question generation process on which words to **generate and to copy**.

- It utilizes the syntactic dependency tree representation of a passage to encode the information of each token in the passage, and sample a clue indicator for each token using a Straight-Through (ST) Gumbel Softmax estimator.

- Our simulation results on the SQuAD dataset and NewsQA dataset show that our model outperforms a range of existing state-of-the-art approaches significantly.