Sequence to Sequence Learning Experiment Report

Tang Songkai 515030910595

March 15, 2018

Introduction

Matching natural language sentences is central for many applications such as information retrieval and question answering. As the wide use of Deep Learning method, a lot of neural network have been proposed and shows great performance on this task. In this experiment, We have implemented one traditional model and three popular deep models for Question-Answer Matching. Then we compare result on two tasks and discussed about the shortcoming of deep model.

Methodology

Problem Definition Given a question $q = q_1 \cdots q_n$, where q_i is the i-th word in the question, and a set of candidate answers $\mathscr{A} = a^1, a^2, ..., a^n$, where $a^j = a^j_1 \cdots a^j_m$ and a^j_k is the k-th word in j-th candidate answer, the goal is to identify the most relevant answer a^{best} given a target question q.

Word Embedding Correlation

Shen et al. [2017] used a simple technique named "translation matrix". The translation matrix $\mathbf{M}$ are used to transform words in the answer into words in the question. Then, the best-fit method is used to choose the most related word in answer for each single word in question.

Word-level Correlation Function Given a pair of words (q_i, a_j), their WEC scoring function is defined as:

$$C(q_i, a_j) = cos < v_{q_i}, \mathbf{M}v_{a_j} >= \frac{v^T_{q_i}}{||v_{q_i}||} \frac{\mathbf{M}v_{a_j}}{||\mathbf{M}v_{a_j}||} \tag{1}$$

where v_{q_i} and v_{a_j} represent q_i and a_j's d-dimensional word embedding vectors; $||||$ is Euclidean norm; correlations matrix $\mathbf{M} \in \mathbb{R}^{d \times d}$.

Sentence-level Correlation Function For a Q&A pair (q, a), their correlation score is defined as:

$$C(q, a) = \frac{1}{|a|} \sum_j \max_j C(q_i, a_j) \tag{2}$$

where $|a|$ represents the length of answer a, $C(q_i, a_j)$ is the correlations score of the i-th word in question and the j-th word in the answer. The max-operator choose one most related word in question for each word in answer.

Architecture-II

Convolutional Sentence Model

Hu et al. [2014] proposed a convolutional architecture to generate a sentence embedding. As illustrated in Figure 1, it takes the embedding of words as input (often trained from a large corpus) in the sentence aligned sequentially, and summarize the meaning of a sentence through layers of convolution and pooling, until reaching a fixed length vectorial representation in the final layer. A large feature map is designed to adequately model the rich structures in the composition of words.

Convolution As shown in Figure 1, the convolution in Layer-1 operates on sliding windows of words (width k_1), and the convolutions in deeper layers are defined in a similar way. Generally, with sentence input $\mathbf{x}$, the convolution unit for feature map of type-f (among F_ℓ of them) on Layer-ℓ is

$$z^{(\ell,f)}_i \stackrel{\text{def}}{=} z^{(\ell,f)}_i(\mathbf{x}) = \sigma(\mathbf{w}^{(\ell,f)} \hat{\mathbf{z}}^{(\ell-1)}_i + b^{(\ell,f)}), \quad f = 1, 2, \cdots, F_\ell \tag{3}$$

and its matrix form is $\mathbf{z}^{(\ell)}_i \stackrel{\text{def}}{=} \mathbf{z}^{(\ell)}_i(\mathbf{x}) = \sigma(\mathbf{W}^{(\ell)} \hat{\mathbf{z}}^{(\ell-1)}_i + \mathbf{b}^{(\ell)})$, where

- $z^{(\ell,f)}_i(\mathbf{x})$ gives the output of feature map of type-f for location i in Layer-ℓ;
- $\mathbf{w}^{(\ell,f)}$ is the parameters for f on Layer-ℓ, with matrix form $\mathbf{W}^{(\ell)} \stackrel{\text{def}}{=} [\mathbf{w}^{(\ell,1)}, \cdots, \mathbf{w}^{(\ell,F_\ell)}]$;
- $\sigma(\cdot)$ is the activation function (e.g., Sigmoid or Relu)
- $\hat{\mathbf{z}}^{(\ell-1)}_i$ denotes the segment of Layer-$\ell-1$ for the convolution at location i , while

$$\hat{\mathbf{z}}^{(0)}_i = \mathbf{x}_{i:i+k_1-1} \stackrel{\text{def}}{=} [\mathbf{x}^\top_i,\ \mathbf{x}^\top_{i+1},\ \cdots,\ \mathbf{x}^\top_{i+k_1-1}]^\top$$

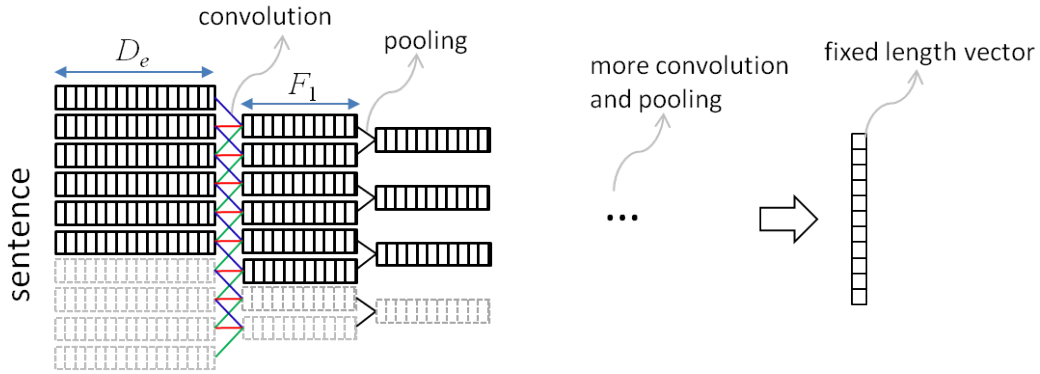

Figure 1: **The over all architecture of the convolutional sentence model.** A box with dashed lines indicates all-zero padding turned off by the gating function.

concatenates the vectors for k_1 (width of sliding window) words from sentence input $\mathbf{x}$.

Max-Pooling Max-pooling is applied in every two-unit window for every f, after each convolution

$$z_i^{(\ell,f)} = \max(z_{2i-1}^{(\ell-1,f)}, z_{2i}^{(\ell-1,f)}), \quad \ell = 2, 4, \cdots .$$

The max-pooling method aims to shrink the feature map size to deal with variable length of input sentence and filter out undesirable composition of words.

Length Variability Convolutional Network is often applied to fix-sized input. The author use the all-zero padding method to deal with variable length of input. Then, a gate function is used to set the output vectors to all-zeros if the input is all zeros. In this way, the boundary caused by zero padding can be eliminated. For any given sentence input $\mathbf{x}$, the output of type-f filter for location i in the ℓ^{th} layer is given by

$$z_i^{(\ell,f)} \stackrel{\text{def}}{=} z_i^{(\ell,f)}(\mathbf{x}) = g(\hat{\mathbf{z}}_i^{(\ell-1)}) \cdot \sigma(\mathbf{w}^{(\ell,f)} \hat{\mathbf{z}}_i^{(\ell-1)} + b^{(\ell,f)}), \tag{4}$$

where $g(\mathbf{v}) = 0$ if all the elements in vector $\mathbf{v}$ equals 0, otherwise $g(\mathbf{v}) = 1$. This gate, working with max-pooling and positive activation function (e.g., Sigmoid), keeps away the artifacts from padding in all layers. Actually it creates a natural hierarchy of all-zero padding (as illustrated in Figure 1), consisting of nodes in the neural net that would not contribute in the forward process (as in prediction) and backward propagation (as in learning).

Direction Interaction Architecture

Architecture-II (ARC-II) is built directly on the interaction space between two sentences. It has the desirable property of letting two sentences meet before their own high-level representations mature, while still retaining the space for the individual development of abstraction of each sentence. Basically, in Layer-1, we take sliding windows on both sentences, and model all the *possible* combinations of them through "one-dimensional" (1D) convolutions. For segment i on S_X and segment j on S_Y, we have the feature map

$$z_{i,j}^{(1,f)} \stackrel{\text{def}}{=} z_{i,j}^{(1,f)}(\mathbf{x}, \mathbf{y}) = g(\hat{\mathbf{z}}_{i,j}^{(0)}) \cdot \sigma(\mathbf{w}^{(1,f)} \hat{\mathbf{z}}_{i,j}^{(0)} + b^{(1,f)}), \tag{5}$$

where $\hat{\mathbf{z}}_{i,j}^{(0)} \in \mathbb{R}^{2k_1 D_e}$ simply concatenates the vectors for sentence segments for S_X and S_Y:

$$\hat{\mathbf{z}}_{i,j}^{(0)} = [\mathbf{x}_{i:i+k_1-1}^\top, \ \mathbf{y}_{j:j+k_1-1}^\top]^\top.$$

Clearly the 1D convolution preserves the location information about both segments. After that in Layer-2, it performs a 2D max-pooling in non-overlapping 2×2 windows (illustrated in Figure 2)

$$z_{i,j}^{(2,f)} = \max(\{z_{2i-1,2j-1}^{(1,f)}, z_{2i-1,2j}^{(1,f)}, z_{2i,2j-1}^{(1,f)}, z_{2i,2j}^{(1,f)}\}). \tag{6}$$

In Layer-3, we perform a 2D convolution on $k_3 \times k_3$ windows of output from Layer-2:

$$z_{i,j}^{(3,f)} = g(\hat{\mathbf{z}}_{i,j}^{(2)}) \cdot \sigma(\mathbf{W}^{(3,f)} \hat{\mathbf{z}}_{i,j}^{(2)} + b^{(3,f)}). \tag{7}$$

This could go on for more layers of 2D convolution and 2D max-pooling, analogous to that of convolutional architecture for image input **?**.

The 2D-Convolution After the first convolution, we obtain a low level representation of the interaction between the two sentences, and from then we obtain a high level representation $\mathbf{z}_{i,j}^{(\ell)}$ which encodes the information from both sentences. The general two-dimensional convolution is formulated as

$$\mathbf{z}_{i,j}^{(\ell)} = g(\hat{\mathbf{z}}_{i,j}^{(\ell-1)}) \cdot \sigma(\mathbf{W}^{(\ell)} \hat{\mathbf{z}}_{i,j}^{(\ell-1)} + \mathbf{b}^{(\ell,f)}), \quad \ell = 3, 5, \cdots \tag{8}$$

where $\hat{\mathbf{z}}_{i,j}^{(\ell-1)}$ concatenates the corresponding vectors from its 2D receptive field in Layer-ℓ–1. This pooling has different mechanism as in the 1D case,

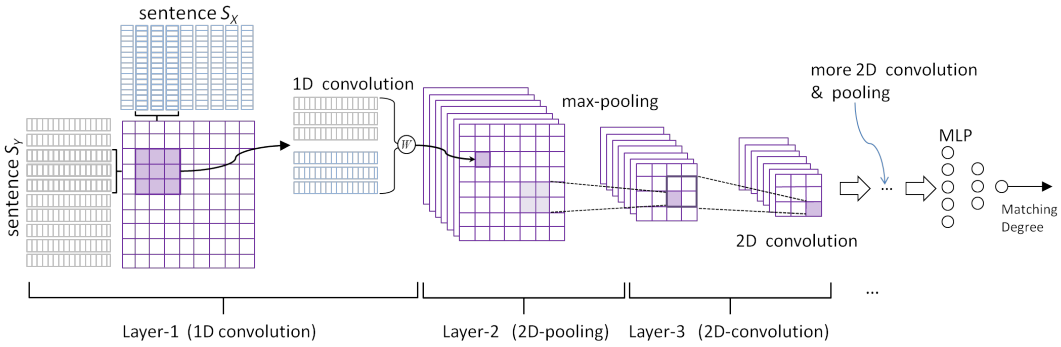

Figure 2: Architecture-II (ARC-II) of convolutional matching model

for it selects *not only* among compositions on different segments but also among different local matchings. This pooling strategy resembles the dynamic pooling in a similarity learning context, but with two distinctions: 1) it happens on a fixed architecture and 2) it has much richer structure than just similarity.

Convolutional Neural Tensor Network

Qiu and Huang [2015] also uses CNN to generate sentence embedding.

Dynamic CNN

The original idea is adopted from Kalchbrenner et al. [2014].

Wide Convolution Given an input sentence, to obtain the first layer of the DCNN we take the embedding $\mathbf{w}_i \in \mathbb{R}^d$ for each word in the sentence and construct the sentence matrix $\mathbf{s} \in \mathbb{R}^{d \times s}$. The values in the embeddings $\mathbf{w}_i$ are parameters that are optimised during training. A convolutional layer in the network is obtained by convolving a matrix of weights $\mathbf{m} \in \mathbb{R}^{d \times m}$ with the matrix of activations at the layer below. For example, the second layer is obtained by applying a convolution to the sentence matrix $\mathbf{s}$ itself. Dimension d and filter width m are hyper-parameters of the network. We let the operations be *wide* one-dimensional convolutions as described in. The resulting matrix $\mathbf{c}$ has dimensions $d \times (s + m - 1)$.

(Dynamic) k-Max Pooling In deep neural network, max pooling is a general technology to extract information. In Computer Vision, max pooling is based on an infinitely strong prior says that the weights for one hidden unit must be identical to the weights of its neighbor, but shifted in space. Goodfellow et al. [2016] thinks In natural language processing, this priority may not be hold because sentences which built from words is much more discrete than images. The k max-pooling selects the top k active feature and keep its order due to position information is also important in natural language processing. To make a smooth extraction of higher order and longer-range features, the parameter k is set to be adaptively:

$$k_l = \max(k_{top}, \left\lceil \frac{L-l}{L} s \right\rceil) \tag{9}$$

where k_{top} is set to be n_s, the embedding size of sentence representation and l and L is the order of convolution layer and total number of convolution layer. The whole model contains three convolution layer with three max pooling layer. In the last max pooling layer, k is fixed to n_s to output a fix-sized vector of sentence embedding.
Note: In my implementation of this model, due to the size of input are often less than n_s, which is set to 50. We adopt the method used in Kalchbrenner et al. [2014]'s work and use a huge size of feature map with $k_{top} = 3$. Then we use a Dense layer to generate a vector with a compatible size to next layer. The performance is discussed in experiment result.

Tensor Match Layer

The output of CNN is two fixed-sized vector which represent sentence embedding. We can use all metrics of vectors similarity function. The author uses a tensor layer to compute the similarity score. Given two sentence embedding vector $\mathbf{v_q}$ and $\mathbf{v_a}$

$$s(q, a) = \mathbf{u}^T f(\mathbf{v_q}^T \mathbf{M}^{[1:c]} \mathbf{v_a} + V \begin{bmatrix} \mathbf{v_q} \\ \mathbf{v_a} \end{bmatrix} + \mathbf{b}), \tag{10}$$

where f is a standard nonlinearity applied elementwise, $\mathbf{M}^{[1:c]} \in \mathbb{R}^{n_s \times n_s \times r}$ is a tensor and the bilinear tensor product $\mathbf{v_q}^T \mathbf{M}^{[1:r]} \mathbf{v_a}$ results in a vector $h \in \mathbb{R}^r$, where each entry is computed by one slice $i = 1, \cdots, r$ of the tensor: $hi = \mathbf{v_q}^T \mathbf{M}^i \mathbf{v_a}$; the other parameters are the standard form of a neural network: $V \in \mathbb{R}^{r \times 2n_s}$, $b \in \mathbb{R}^r$ and $u \in \mathbb{R}^r$.

MV-LSTM

The LSTM network are often used to modeling sentences because it is designed to take variable length of input. Wan et al. [2016] uses Bi-LSTM to generate word embedding with positional information and use them to represent a sentence.

Bi-LSTM

Given an input sentence $S = (x_0, x_1, \cdots, x_T)$, where x_t is the word embedding at position t. LSTM outputs a representation h_t for position t. In bidirectional architecture, we can obtain two vectors $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ for each position. For each word in a sentence, we can generate its positional representation:

$$p_t = [\overrightarrow{h_t}^T, \overleftarrow{h_t}^T]^T \tag{11}$$

Sentence Interaction and Aggregation

On the basis of positional sentence representations, we can model the interactions between a pair of sentences from different positions.

Tensor Matching Layer Give two sentences S_X and S_Y, similarity vector function $s(p_{Xi}, p_{Yj})$ will give the interactions between p_{Xi} and p_{Yj}, where p_{Xi} and p_{Yj} stand for the i and j-th positional sentence representations for two sentences respectively:

$$s(u, v) = f(u^T M^{[1:c]} v + W_{uv} \begin{bmatrix} u \\ v \end{bmatrix} + b), \tag{12}$$

where $M^i, i \in [1, ..., c]$ is one slice of the tensor parameters, W_{uv} and b are parameters of the linear part. f is a non-linear function, and we use rectifier $f(z) = \max(0, z)$ since it always outputs a positive value which is compatible as a similarity.

k-Max Pooling The author also used k-Max pooling to automatically extract top k strongest interactions in the matrix/tensor, similar to Kalchbrenner et al. [2014]. Instead of find top feature to represent a sentence, k-Max pooling in this paper is to find interaction between sentence pair, which can be viewed as different attribute of similarity. The top k values of each slice of the interaction tensor are returned to form a vector. Finally, these vectors are further concatenated to a single vector q. k-Max pooling is also found to be easier to detect where the best matching position lies, and whether we need to aggregate multiple interactions from different positions for matching. The experiments show that better results can be obtained by leveraging matchings on multiple positions.

MultiLayer Perception Finally, a MLP is used to output the matching score by aggregating such strong interaction signals filtered by k-Max pooling. Specifically, the feature vector q obtained by k-Max pooling is first feed into a full connection hidden layer to obtain a higher level representation r. Then the matching score s is obtained by a linear transformation:

$$r = f(W_r q + b_r), \ s = W_s r + b_s, \tag{13}$$

where W_r and W_s stands for the parameter matrices, and b_r and b_s are corresponding biases.

Experiment

Loss Function For simplicity, we formalize the task as a ranking problem and use a common training method for these four models. We utilize pairwise ranking loss as hinge loss for training. Given a triple (S_X, S_Y^+, S_Y^-), where S_Y^+ is ranked higher than S_Y^- when matching with S_X, the loss function is defined as:

$$\mathcal{L}(S_X, S_Y^+, S_Y^-) = \max(0, 1 - s(S_X, S_Y^+) + s(S_X, S_Y^-)) \tag{14}$$

where $s(S_X, S_Y^+)$ and $s(S_X, S_Y^-)$ are the corresponding matching scores.

Data Preparation Yang et al. [2015] has proposed a new dataset named WIKIQA for Open-Domain Question Answering. This dataset is built from Microsoft Bing query logs. It selected question-like queries using simple heuristics, such as queries starting with a WH-word (e.g., "what" or "how") and queries ending with a question mark and is issued by at least 5 unique users and have clicks to Wikipedia. Then, it filtered out some entity queries that satisfy the rules, such as the TV show "how I met your mother.". Candidate answers are generated from the summary section of target Wikipedia page. The dataset contains 3050 QA pairs in total. In my perspective, the search query often requires more common domain knowledge than question from UGC like Yahoo Answers. Questions containing too much personal information like "What are you come from?" are not likely to be included in this data set.
The choice of word embedding is also important. Lai et al. [2016] found that using an small in-domain corpus significantly improves performance for a given task, whereas using a large corpus in an unsuitable domain can decrease performance. It also shows that for NLP tasks that utilize an embedding as a feature or for initialization, a dimensionality of 50 is sufficient. For these two reason, we choose Glove.6B(Pennington et al. [2014]), which is

trained from Wikipedia 2014 + Gigaword 5 corpus with 6B tokens, 400K vocab. The word embedding layer of four models outputs 50d vector and will not be fine-tuned during training. As for the training input, we generate 11(one positive + ten negative) pairs for each question. The total dataset are selected by the ratio of the length of answer text over the length of question text, which will be explained in next subsection.

Evaluation Metrics We use the mean average precision (MAP) and normalized discounted cumulative gain (NDCG) to evaluate the results. A set of candidate answers is created with size 11 (one positive + ten negative) for each question in the test data. Then the performance of ranking quality will be calculated over these ten answers. The premise of NDCG is that the misplaced result should be penalised logarithmically proportional to the distance from real position to current position.

Parameter Setting

- Global setting:
  - Optimizer: adadelta
  - Learning Rate: 0.01
  - Hinge Loss Margin: 1.0
  - Word embedding: 50
- ARC-II
  - 1d kernel count: 20
  - 1d kernel size: 3
  - num conv2d layers: 1
  - 2d kernel counts: [20]
  - 2d kernel sizes: [[3, 3]]
  - 2d mpool sizes: [[3, 3]]
  - dropout rate: 0.9
- CNTN
  - 1d kernel size: 3
  - 1d kernel num: 50,100,200
  - top k: 5
  - sentence embedding: 50
  - r: 5
- MV-LSTM
  - LSTM hidden unit size: 50
  - top k: 100
  - dropout rate: 0.5

Normal QA Pair

We use a normal dataset which is a subset directly chosen from original set. The training set size is 20360 and the validation and testing set all contains about 1000 pairs. The performance measure are shown in Figure3a

Short-Long QA Pair

In this data set, the QA pair are chosen by the ratio of answer length over question length. Only pairs with ratio over 3 are selected. The training set size is 8351 and the validation and testing set all contains about 1000 pairs. The performance measure are shown in Figure3b

Discussion

The huge performance of WEC model is somewhat tricky because it outperformed other deep models with a simple architecture and short training time. In normal sentence task, ARC-II outperforms all other models and WEC model is far away from all deep models. In short-long task, WEC model is way ahead, followed by ARC-II. We collect all question which has at least one mis-paired answers from test set, some of them are shown in Figure 4. From sampled mis-matched pairs, we can see that WEC model are good at "answer" question with obvious intention like "What is something?". The answer to such question can be viewed as "definition", which is a long text of description and refers to the word be defined many times. Under such condition, the direct interaction between words is much more important.

Besides, the merit of deep model cannot be neglected, as it can grasp semantic information in a high level. The LSTM model acts like a man reading sentence word by word and the 1D convolution acts like a grammar parsing tree, which can learning from not only single words, but phases.

From my perspective, the enhancement to current model can be viewed as a ensemble learning approach: We can integrate identity block into the whole neural network, and at the end of output, the score from translation matrix(WEC Model) and

Models	NDCG@3	NDCG@5	MAP
WEC	0.485437	0.546745	0.508369
ARC-II	0.546709	0.616266	0.569121
CNTN	0.539489	0.606684	0.560798
MV-LSTM	0.540077	0.611622	0.558051

(a)

Models	NDCG@3	NDCG@5	MAP
WEC	0.834926	0.852815	0.810173
ARC-II	0.496673	0.564462	0.507174
CNTN	0.383719	0.477619	0.437992
MV-LSTM	0.442138	0.543960	0.488529

(b)

Figure 3: Evaluation in test set. (a) Result from normal pair. (b) Result from short-long pair

deep models will be calculated separately. The aggregation of two scores can be adaptive to question: If the length of question is short, the model tend to use prediction from translation matrix and if the length of question is long, more semantic information from deep model will be used.
The idea of direct mapping reminds me of He et al. [2016]'s ResNet, which is used in very deep architecture of image recognition. In practice, our network may adopt some design idea from it. The problem is how to do it in a neat way, i.e how to build a single network with careful design rather than combine two model directly. I will keep doing experiments on it and I am looking forward to follow the instructions and suggestions!

References

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd*

Question	(How)day st. day came
Correct Answer	saint patrick 's day feast saint patrick (, " day festival patrick ") cultural religious holiday celebrated 17 march .
WEC	saint patrick 's day feast saint patrick (, " day festival patrick ") cultural religious holiday celebrated 17 march .
CNTN	monument includes acres (32 square miles) , much recommended congress designation wilderness .
MV-LSTM	period consists 12 hours numbered : 12 (acting zero) , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 11 .
ARC-II	oscar grant fatally shot bart police officer johannes mehserle oakland , california , united states , early morning hours new year 's day 2009

(a)

Question	(What is)points mortgage
Correct Answer	points , sometimes also called " discount point " , form interest .
WEC	points , sometimes also called " discount point " , form interest .
CNTN	points , sometimes also called " discount point " , form interest .
MV-LSTM	steelers founded pittsburgh pirates july 8 , 1933 , art rooney , taking original name baseball team name , common practice nfl teams time .
ARC-II	exception bond , , perpetuity , i.e . bond maturity .

(b)

Question	(Who) use semicolon
Correct Answer	italian printer aldus manutius elder established practice using semicolon separate words opposed meaning indicate interdependent statements .
WEC	inside mainly sugar , corn syrup , palm kernel oil along fruit juice , citric acid , natural artificial .
CNTN	tourette syndrome (also called tourette 's syndrome , tourette 's disorder , gilles de la tourette syndrome , gts , commonly , simply tourette 's) inherited disorder onset childhood , characterized multiple physical (motor) tics least one vocal ( ) tic .
MV-LSTM	tourette syndrome (also called tourette 's syndrome , tourette 's disorder , gilles de la tourette syndrome , gts , commonly , simply tourette 's) inherited disorder onset childhood , characterized multiple physical (motor) tics least one vocal ( ) tic .
ARC-II	italian printer aldus manutius elder established practice using semicolon separate words opposed meaning indicate interdependent statements .

(c)

Figure 4: Some sample question-answer pair

Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/P14-1062.

S. Lai, K. Liu, S. He, and J. Zhao. How to generate a good word embedding. *IEEE Intelligent Systems*, 31(6):5–14, Nov 2016. ISSN 1541-1672. doi: 10.1109/MIS.2016.45.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL http://www.aclweb.org/anthology/D14-1162.

Xipeng Qiu and Xuanjing Huang. Convolutional neural tensor network architecture for community-based question answering. In *IJCAI*, pages 1305–1311, 2015.

Yikang Shen, Wenge Rong, Nan Jiang, Baolin Peng, Jie Tang, and Zhang Xiong. Word embedding based correlation model for question/answer matching. In *AAAI*, pages 3511–3517, 2017.

Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. A deep architecture for semantic matching with multiple positional sentence representations. In *AAAI*, volume 16, pages 2835–2841, 2016.

Yi Yang, Wen-tau Yih, and Christopher Meek. WikiQA: A challenge dataset for open-domain question answering. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Lisbon, Portugal, September 2015. Association for Computational Linguistics.