Machine Reading Comprehension: Matching and Orders

Ao Liu
zeitmond@gmail.com
University of Electronic Science and Technology of China
Chengdu, Sichuan, China

Lizhen Qu*
Lizhen.Qu@monash.edu
Laboratory for Dialogue Research
Monash University
Melbourne, Victoria, Australia

Junyu Lu
cs.junyu@gmail.com
University of Electronic Science and Technology of China
Chengdu, Sichuan, China

Chenbin Zhang
aleczhang13@gmail.com
University of Electronic Science and Technology of China
Chengdu, Sichuan, China

Zenglin Xu
zenglin@gmail.com
University of Electronic Science and Technology of China
Chengdu, Sichuan, China

ABSTRACT

In this paper, we study the machine reading comprehension of temporal order in text. Given a document of instruction sequences, a model aims to find out the most coherent sequences of activities matching the document among all answer candidates. To tackle the task, we propose *OrdMatch* model, which is able to match each activity in a sequence to the corresponding instruction in the document and regularizes the partial order of activities to match the order of instructions. We evaluate the task using the RecipeQA dataset, which includes step-by-step instructions of cooking recipes. Our model outperforms the state-of-the-art models with a wide margin. The experimental results demonstrate the effectiveness of our novel ordering regularizer. Our code will be made available at https://github.com/Aolius/OrdMatch.

CCS CONCEPTS

• **Computing methodologies → Natural language processing**.

KEYWORDS

Machine Reading Comprehension, RecipeQA, Activity Ordering

ACM Reference Format:
Ao Liu, Lizhen Qu, Junyu Lu, Chenbin Zhang, and Zenglin Xu. 2019. Machine Reading Comprehension: Matching and Orders. In *The 28th ACM International Conference on Information and Knowledge Management (CIKM '19), November 3–7, 2019, Beijing, China.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3357384.3358139

1 INTRODUCTION

Machine Reading Comprehension (MRC) requires systems to understand one or multiple passages of text and to answer arbitrary questions. Existing MRC corpora cover a wide range of genres, ranging from Wikipedia articles [16], conversations [17], to movie plots [18]. However, almost all of them have not targeted at understanding temporal order of events or activities in MRC, despite a large amount of texts, such as recipes and user manuals, are rich in temporal knowledge. To fill the gap, this paper focuses on reading comprehension of temporal order in text.

There is ample of work studying temporal aspects of information. The task QA TempEval [10] in SemEval is the most close to ours. The participating systems are required to annotate plain documents with *events* and *temporal relations*, and answer *yes/no* questions based on extracted temporal information. However, the task has three major limitations: i) an event is defined as a single word such as "came" and "attack", but even simple activities, e.g. "go to a cinema", are described with more than single words; ii) the built systems are difficult to move to new domains, because they require rich manual annotations of temporal information for their temporal information extraction systems; iii) yes/no questions as an evaluation method is not sufficient to check all possible mistakes of temporal order. A large body of works on temporal ordering and reasoning [2, 12, 13] are based on the same definition of events as in TempEval. They do not tackle the challenge of correctly associating arbitrary text description of events with text passages, and some systems need extracted temporal expressions [2, 11]. Another similar task is TempQuestions [7], which is designed for QA over structured knowledge bases. In contrast, our setting needs only plain texts.

The release of RecipeQA corpus [21] provides an opportunity to evaluate comprehension of temporal order. As shown in Table 1, each cooking recipe consists of a sequence of text-based instructions. And the understanding of recipes are evaluated through a set of QAs categorized into four tasks. In order to thoroughly evaluate comprehension of order information, inspired by the sentence ordering task [1, 3, 5], we modify the Textual Cloze task of RecipeQA into a multi-choice activity ordering task. The question is to find the most coherent activity sequence given a recipe. Each of its answers is a sequence of cooking activities, which take the form of short phrases.

The comprehension of temporal order in text imposes two challenges: (1) *Matching*: for each activity in an answer, identify if there is a matched instruction in the recipe. The activities may be abstract, such as "prepare ingredients", such that no exactly matched

*Corresponding author. This work is partially completed while he worked at Data61.

phrases can be found in the instructions. (2) *Ordering*: identify the right order of cooking activities. It cannot be directly solved by mapping activities to instructions, because i) the matching between activities and instructions is ambiguous; ii) one activity may be mapped to more than one instruction; iii) some activities in answers are distractors that are not described in the corresponding recipes.

To tackle the above challenges, we propose a joint model *OrdMatch* with two modules: i) the hierarchical matching module; and ii) the attended ordering module. The former aims to identify matching instruction for each activity, while the latter acts as a regularizer to ensure the activities in an answer matching the order of instructions. To the best of our knowledge, this paper is the first to propose the task of reading comprehension of temporal order. Experimental results show that our model achieves superior performance than competitive baselines.

Table 1: An example recipe and its temporal ordering question and answers. The true answer is in bold.

Mexican Chicken Alfredo: *1. Chop everything up to make it much easier and more fun when actually cooking.*   *2. First I cooked the onions and peppers. Once they were cooked, I added the chicken and spices. Lastly, and while the pasta was boiling I added some wine to the bottom of the pan to break up the tasty bits form the bottom of my pan, and to mix into the food. One tip when cooking with wine, don't use anything you would not drink.*   *3. To this mix, I added one can of spicy salsa.*   *4. As the pasta water begins to boil I add the can of sauce and let it simmer until the pasta is cooked.*   *5. Once the pasta is cooked to your desired tenderness, drain, and immediately add the pasta sauce. Toss, add cheese and eat. I will tell you, it may not look like much, but it is a bowl full of awesome.*
Q: Choose the most coherent activity sequence of the following choices.   (i) *Prepare ingredients. Mix and eat. Cooking ingredients. Add salsa.*   (ii) *Prepare ingredients. Cooking ingredients. Add enchilada sauce. Mix and eat.*   (iii) ***Prepare ingredients. Cooking ingredients. Add salsa. Mix and eat.***   (iv) *Cooking ingredients. Add salsa. Mix and eat. Prepare ingredients.*   ...

2 METHODOLOGY

Our model *OrdMatch* consists of a hierarchical matching module and an attended ordering module.

2.1 Problem Formalization

Formally, we are given a document $D = [S_1, S_2, \dots, S_N]$ of instructions, in which each instruction $S_n = [s_1^{(n)}, s_2^{(n)}, \dots, s_{L_S}^{(n)}]$ is a sequence of tokens. For each question, there are K possible answers $A = \{a_1, a_2, ..., a_K\}$. Each answer $a_k = [\alpha_1, \alpha_2, \dots, \alpha_M]$ consists of M activities, where α_i denotes an activity composed of L_h words: $\alpha_i = [t_1^{(i)}, t_2^{(i)}, \dots, t_{L_h}^{(i)}]$. Our goal is to learn a model to predict the correct answers according to the conditional distribution $P(a_i|D)$.

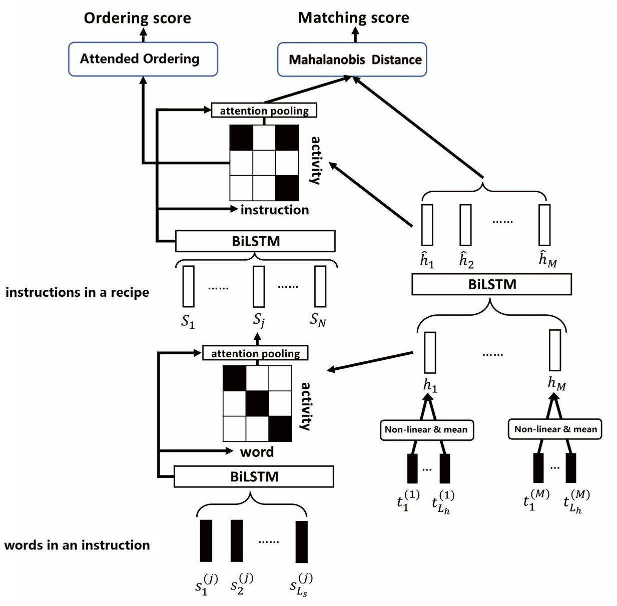

Figure 1: An overview of our proposed model.

2.2 Hierarchical Matching Module

This module identifies alignment between activities and instructions without considering order of activities. The key idea herein is to utilize attention mechanism to find aligned instructions for each activity in answers.

Given an answer a_k, we start with mapping each word in an activity phrase to a word embedding, followed by encoding each activity α_i into an l-dimensional vector $\mathbf{h}_{\alpha_i}$ by performing the following non-linear transformation:

$$\mathbf{h}_{\alpha_i} = \frac{1}{L_h} \sum_{j=1}^{L_h} \text{ReLU}(\mathbf{W}\mathbf{e}_{t_j}), \tag{1}$$

where $\mathbf{e}$ denotes a d-dimensional word embedding, and $\mathbf{W} \in \mathbb{R}^{l \times d}$ is a weight matrix to project word embeddings to a latent activity space. In order to capture information regarding other activities in the same answer, we run a bidirectional LSTM [6, 9](Bi-LSTM) on the activity representation sequences $\mathbf{H}_{a_k} = [\mathbf{h}_{\alpha_1}, \dots, \mathbf{h}_{\alpha_M}]$ to obtain the contextual activity representation $\mathbf{U}_{a_k} \in \mathbb{R}^{l \times M}$. Note that we set the output dimension of the Bi-LSTM as $l/2$ to obtain an $l \times M$ output.

Each instruction S_n is viewed as a word sequence. To make instructions comparable to answers, we apply the same Bi-LSTM and same word embeddings used in answers to transform each instruction to a sequence of contextual representations $\mathbf{U}_{S_n} = [\mathbf{u}_1, \mathbf{u}_2, \ldots, \mathbf{u}_{L_S}] \in \mathbb{R}^{l \times L_S}$.

We characterize the alignment between an activity α_i and an instruction S_n by using both the word-level and the instruction-level attention. At the word level, the attention score is computed by $\mathbf{u}_{s_i}^\top \mathbf{h}_{\alpha_i}$. As in [4], the normalized attention scores are computed by Softmax($\mathbf{U}_{S_n}^\top \mathbf{h}_{\alpha_i}$). We further obtain a representation w.r.t. an activity α_i for each instruction by

$$\mathbf{z}_{S_n}^{\alpha_i} = \mathbf{U}_{S_n} \cdot \text{Softmax}(\mathbf{U}_{S_n}^\top \mathbf{h}_{\alpha_i}) \quad (2)$$

As a result, for each recipe, we obtain a sequence $\mathbf{Z}^{\alpha_i} = [\mathbf{z}_{S_1}^{\alpha_i}, \ldots, \mathbf{z}_{S_N}^{\alpha_i}]$ w.r.t. an activity α_i as its activity specific representation.

In order to capture more contextual information, we run an instruction-level Bi-LSTM on $\mathbf{Z}^{\alpha_i}$ to obtain the contextual representation sequences $\mathbf{V}^{\alpha_i} \in \mathbb{R}^{l \times S_N}$. To calculate how much an activity correlates to each instruction in the recipe, we apply the Activity-to-Instruction attention as follows to construct the representation of a document w.r.t. each activity α_i:

$$\mathbf{d}^{\alpha_i} = \mathbf{V}^{\alpha_i} \cdot \text{Softmax}(\mathbf{V}^{\alpha_i \top} \mathbf{u}_{\alpha_i}), \quad (3)$$

where we use contextual representations for both instructions and activities.

In training, for each recipe, we take the negative of the Mahalanobis distance [20] as the matching score between each activity contextual representation $\mathbf{u}_{\alpha_i}$ and the document vector $\mathbf{d}^{\alpha_i}$:

$$\mathcal{M}(\alpha_i, D) = -\|\mathbf{L}(\mathbf{u}_{\alpha_i} - \mathbf{d}^{\alpha_i})\|^2, \quad (4)$$

where $\mathbf{L} : \mathbb{R}^d \to \mathbb{R}^d$ is a linear transformation. Then the matching score between an answer a_k and a recipe D is the sum of all its activity matching scores. $f_m(a_k, D) = \sum_{i=1}^{M} \mathcal{M}(\alpha_i, D)$.

2.3 Attended Ordering Module

Our attended ordering module aims to constrain the sequence of activities in the true answer to match the order of their corresponding instructions in a recipe. This is achieved by formulating the problem as an optimization problem with order constraints.

The Activity-to-Instruction attention matrix $\mathbf{P} \in \mathbb{R}^{N \times M}$ computed by Equation (3) forms the basis of computing order constraints. Its cell p_{ij} can be viewed as a conditional probability $P(S_i|\alpha_j)$. If $\alpha_i \prec \alpha_j$ with $\prec$ denoting the partial order, we expect $\arg\max_{S_n} P(S_n|\alpha_i) \prec \arg\max_{S_v} P(S_v|\alpha_j)$. In another word, if an answer is a coherent activity sequence, we have $\arg\max_k p_{ki} < \arg\max_{k'} p_{k'j}$, if $i < j$. Let $r_m = \arg\max_k p_{km}$ be the index of the most likely aligned instruction of the activity m, for all the following activities j with $j > m$, we expect that $p_{ij} < p_{r_m m}$ for all $i < r_m$. Otherwise, it would be highly possible to select a matching instruction preceding the instruction r_m, leading to ordering errors. Thus, for a particular activity m with $r_m = \arg\max_k p_{km}$, its ordering error is reduced by minimizing the following quantity:

$$\mathcal{E}_{ij}^m = \max_k p_{km} - p_{ij}, \forall i > r_m, j > m \quad (5)$$

The ordering error of all activities in an answer a_k is given by:

$$f_o(a_k) = \min_m \min_{i > r_m, j > m} \mathcal{E}_{ij}^m. \quad (6)$$

2.4 Joint Training and Prediction

In training, we minimize an interpolated loss $\mathcal{L} = (1-\lambda)\mathcal{L}_M + \lambda\mathcal{L}_O$,

$$\mathcal{L}_M = \sum_D -\log \frac{\exp(f_m(a_k, D))}{\sum_{k'=1}^K \exp(f_m(a_{k'}, D))}, \quad (7)$$

$$\mathcal{L}_O = \sum_D -\log \frac{\exp(f_o(a_k, D))}{\sum_{k'=1}^K \exp(f_o(a_{k'}, D))} \quad (8)$$

where $\lambda \in \mathbb{R}^+$ is a hyperparameter, and the loss of the ordering module serves as a regularizer because it enforces the correct order by using gold standard. As a result, we predict answers only with the matching module:

$$P(a_k|D) = \frac{\exp(f_m(a_k, D))}{\sum_{k'=1}^K \exp(f_m(a_{k'}, D))} \quad (9)$$

3 EXPERIMENTS

Dataset. The recipes in RecipeQA dataset contain instruction sequences. We generate answers in two ways. First, we modify the Textual Cloze task of RecipeQA to generate four activity sequences. A question in Textual Cloze task asks which activity among all four answers can fill the blank of an activity sequence. We use each answer to fill the blank to obtain four sequences. Three of them include distractor activities. Second, the original recipes have a title for each instruction. All instruction titles in the original order form a correct activity sequence for a recipe. We shuffle the order of the titles to generate the remaining answers. Note that, the recipes for our target task *do not* include instruction titles. We further divide the answers into three sets: (1)**TC**, includes only the four activity sequences per recipe derived from the original *Textual Cloze* task; (2)**AO8**, all activity sequences in TC and sampled another four sequences randomly shuffled sequences generated from instruction titles; (3)**AO24**, all 24 permutations of the correct sequence derived from instruction titles. *AO8* and AO24 have the same data splits as *TC* : 7837 for training and 961 for testing.

Baselines. We compare our model with three baselines. Two of them are proposed by [21], which are text-only **Impatient Reader** and **Hasty Student**, respectively. In addition, we implemented **Hier-Co-Matching** [19], a state-of-the-art model for Multiple Choice Machine Comprehension task for comparison.

Implementation Details. We implement our model with PyTorch [14] and use 300 dimensional pretrained Glove[15] embeddings. The batch size is fixed to 5 and the Adamax [8] optimizer is used for optimization with initial learning rate 2e-3.

Table 2: Accuracy on the validation set of TC, AO8 and AO24.

	TC	AO8	AO24
HUMAN	73.60	-	-
Hasty Student	24.35	-	-
Impatient Reader (Text only)	28.51	-	-
Impatient Reader (Multimodal)	33.09	-	-
Hier-Co-Matching	69.09	52.97	43.70
OrdMatch (M)	**75.13**	**56.82**	**51.72**
OrdMatch (M+O)	**75.55**	**58.58**	**57.02**

Results and Analysis. Table 2 lists the performance of our model and the baselines on TC, AO8 and AO24, respectively. We can see that our model achieves the best result on all three datasets. Particularly, it surpasses the human performance on the TC task. Also, we report the results of our matching only model (denoted as *OrdMatch(M)*) and our full model (*OrdMatch(M+O)*) with the ordering regularizer. The results demonstrate the effectiveness of our ordering regularizer.

To investigate whether our ordering loss helps recognize the correct order of activities, we visualize the Activity-to-Instruction attention matrix. In Figure 2, darker color means larger attention weights, the x-axis and the y-axis indicate activities and instructions, respectively. We randomly draw an example with four instructions

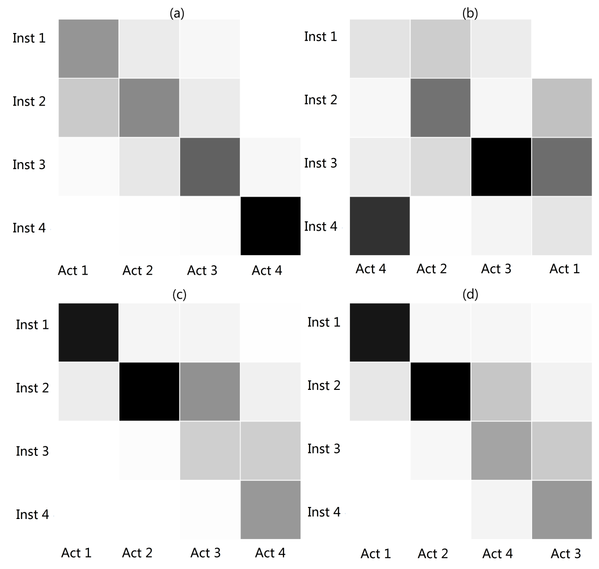

Figure 2: Visualization of activity-to-instruction attention matrix.

and four activities from AO24. The correct order of activity sequence is *Act 1, Act 2, Act 3, Act 4*. We drew the attention matrices on this example with four different model settings, as shown in Figure 2. Matrix (a) and Matrix (c) show the alignment scores based on the true answer extracted from the full model and the matching model, respectively. As Matrix (a) shows, the full model captures the exact alignment between activities and instructions, showing a dark-colored diagonal. Matrix (c) shows that the matching model correctly aligns the activity 1,2 and 4 with the corresponding instructions but mistakenly matches the activity 3 also to the 2nd instruction. Matrix (b) shows the attention matrix of the full model based on a wrong answer. Its attention scores are able to reflect the corresponding mistakes. Matrix (d) is resulted by the predicted answer of the matching model, which fails to find the right order between activity 3 and activity 4. Overall, those matrices demonstrate the effectiveness of our ordering regularizer, which encodes order constraints into the model parameters.

4 CONCLUSION

In this paper, we proposed the task of reading comprehension of temporal order along with an *OrdMatch* model to address this task. The model consists of a matching module and an ordering regularizer. The experiments showed the superior performance of our model over baselines on this task. The visualization experiment further demonstrates the effectiveness of our ordering module.

ACKNOWLEDGMENTS

This work was partially supported by NSF China (Project No. 615721 11) and a Research Fund for the Central Universities of China (Project No. ZYGX2016Z003).

REFERENCES

[1] Harsh Agrawal, Arjun Chandrasekaran, Dhruv Batra, Devi Parikh, and Mohit Bansal. 2016. Sort Story: Sorting Jumbled Images and Captions into Stories. *CoRR* abs/1606.07493 (2016).
[2] Steven Bethard, Guergana Savova, Wei-Te Chen, Leon Derczynski, James Pustejovsky, and Marc Verhagen. 2016. Semeval-2016 task 12: Clinical tempeval. In *SemEval-2016*. 1052–1062.
[3] Qian Chen, Xiaodan Zhu, Zhen-Hua Ling, Si Wei, and Hui Jiang. 2016. Enhancing and Combining Sequential and Tree LSTM for Natural Language Inference. *CoRR* abs/1609.06038 (2016).
[4] Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, Hui Jiang, and Diana Inkpen. 2016. Enhanced lstm for natural language inference. *arXiv preprint arXiv:1609.06038* (2016).
[5] Jingjing Gong, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2016. End-to-End Neural Sentence Ordering Using Pointer Network. *CoRR* abs/1611.04953 (2016).
[6] Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation* 9, 8 (1997), 1735–1780.
[7] Zhen Jia, Abdalghani Abujabal, Rishiraj Saha Roy, Jannik Strötgen, and Gerhard Weikum. 2018. TempQuestions: a Benchmark for Temporal Question Answering. In *Companion of the The Web Conference 2018 on The Web Conference 2018*. ACM, 1057–1062.
[8] Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2014).
[9] Hao Liu, Lirong He, Haoli Bai, Bo Dai, Kun Bai, and Zenglin Xu. 2018. Structured Inference for Recurrent Hidden Semi-markov Model. In *IJCAI*. ijcai.org, 2447–2453.
[10] Hector Llorens, Nathanael Chambers, Naushad UzZaman, Nasrin Mostafazadeh, James F. Allen, and James Pustejovsky. 2015. SemEval-2015 Task 5: QA TempEval - Evaluating Temporal Information Understanding with Question Answering. In *SemEval@NAACL-HLT 2015*. The Association for Computer Linguistics, 792–800.
[11] Junyu Lu, Chenbin Zhang, Zeying Xie, Guang Ling, Tom Chao Zhou, and Zenglin Xu. 2019. Constructing Interpretive Spatio-Temporal Features for Multi-Turn Responses Selection. In *ACL*. Association for Computational Linguistics, 44–50.
[12] Yuanliang Meng, Anna Rumshisky, and Alexey Romanov. 2017. Temporal Information Extraction for Question Answering Using Syntactic Dependencies in an LSTM-based Architecture. In *EMNLP*. Association for Computational Linguistics, 887–896.
[13] Anne-Lyse Minard, Manuela Speranza, Eneko Agirre, Itziar Aldabe, Marieke van Erp, Bernardo Magnini, German Rigau, and Ruben Urizar. 2015. SemEval-2015 Task 4: TimeLine: Cross-Document Event Ordering. In *SemEval@NAACL-HLT 2015*. The Association for Computer Linguistics, 778–786.
[14] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in PyTorch. In *NIPS-W*.
[15] Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *EMNLP*. 1532–1543.
[16] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. SQuAD: 100,000+ Questions for Machine Comprehension of Text. In *EMNLP*. 2383–2392.
[17] Siva Reddy, Danqi Chen, and Christopher D Manning. 2019. Coqa: A conversational question answering challenge. *TACL* 7 (2019), 249–266.
[18] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. 2016. MovieQA: Understanding Stories in Movies through Question-Answering. In *CVPR*. 4631–4640.
[19] Shuohang Wang, Mo Yu, Jing Jiang, and Shiyu Chang. 2018. A Co-Matching Model for Multi-choice Reading Comprehension. In *ACL (Volume 2: Short Papers)*. 746–751.
[20] Kilian Q Weinberger and Lawrence K Saul. 2009. Distance metric learning for large margin nearest neighbor classification. *Journal of Machine Learning Research* 10, Feb (2009), 207–244.
[21] Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. 2018. RecipeQA: A Challenge Dataset for Multimodal Comprehension of Cooking Recipes. In *EMNLP*. 1358–1368.