Incorporating Label Dependency for Answer Quality Tagging in Community Question Answering via CNN-LSTM-CRF

Yang Xiang, Xiaoqiang Zhou[1], Qingcai Chen[2], Zhihui Zheng, Buzhou Tang, Xiaolong Wang, and Yang Qin
Intelligent Computation Research Center,
Harbin Institute of Technology Shenzhen Graduate School, China
{xiangyang.hitsz, xiaoqiang.jeseph,qingcai.chen,
zhihui.zchina,tangbuzhou}@gmail.com
wangxl@insun.hit.edu.cn, csyqin@hitsz.edu.cn

Abstract

In community question answering (cQA), the quality of answers are determined by the matching degree between question-answer pairs and the correlation among the answers. In this paper, we show that the dependency between the answer quality labels also plays a pivotal role. To validate the effectiveness of label dependency, we propose two neural network-based models, with different combination modes of Convolutional Neural Networks, Long Short Term Memory and Conditional Random Fields. Extensive experiments are taken on the dataset released by the SemEval-2015 cQA shared task. The first model is a stacked ensemble of the networks. It achieves 58.96% on macro averaged F_1, which improves the state-of-the-art neural network-based method by 2.82% and outperforms the Top-1 system in the shared task by 1.77%. The second is a simple attention-based model whose input is the connection of the question and its corresponding answers. It produces promising results with 58.29% on overall F_1 and gains the best performance on the *Good* and *Bad* categories.

1 Introduction

Community question answering (cQA) provide abundant human-to-human questions and answers, behaving as a good resource for building automatic question answering systems. For example, many users ask about "How to build your body like a model?" in Yahoo Answers[3] and also there are many answerers who are willing to share their experiences. Answer quality tagging refers to automatically identifying whether an answer is good, so as to collect high quality question answer pairs.

The task is challenging in that one should develop useful features that can effectively bridge the semantic gap between the question and answer (QA) pair. The matching degree on content is the primary factor that determines how good an answer is. It can be measured through a series of syntactic and semantic features such as overlapped linguistic elements (Berger et al., 2000; Agichtein et al., 2008; Punyakanok et al., 2004). For example, if the question and answer have several words shared, the answer is likely to be good. However, the content similarity performs well only when both the question and answer are well-structured and overlapped, which are rare to see in cQA sentences.

Many previous studies discover specific features based on the characteristics of cQA systems. For instance, opinion leader or experts would probably contribute more good answers than bad, and the number of thumbs-up towards an answer may reflect how much the users accept it (Agichtein et al.,

[1] The first two authors have equal contributions.
[2] Corresponding author
[3] https://answers.yahoo.com/question/index?qid=20070918225025AA5Jz0G

Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, pages 1231–1241, Osaka, Japan, December 11-17 2016.

2008). The inefficiency of textual similarity can be partly complemented with the help of these specific features (Hu et al., 2013; Tran et al., 2015). But the definition and validation of them need too much laborious cost and might not be appropriate when transferring to newly built systems.

In this paper, we focus on a novel concept of contextual feature, the *label dependency*, which we think plays a pivotal role when predicting the answer quality in cQA. Intuitively, being a rational answerer, when we decide to answer a question, we face the following situations:

- If it is a new question (without an answer), just answer it.
- If it has been answered but all the existed answers are bad, we provide our good answer.
- If it has been answered and some of the existed answers have already satisfied the asker's information needs, we can choose to answer it from a distinct perspective.
- Other actions.

A typical example for cQA is shown in Figure 1 in which A4 and A5 follow several bad or potentially good comments. Although A4 and A5 are both good answers, they provide alternative useful information. Based on the above assumptions, we say that there are possible soft constraints between the quality tags of the answers (i.e. a good answer follows several bad ones). The contextual dependency relation can be affected by the relations among the answer contents as well as the interactions between answer quality tags.

> **Q**: Where can i buy globe roam sim here in qatar?Can anyone tell me where can i buy globe roaming sim? thanks! if your selling globe roaming sim.
>
> **A1**: vivo bonito, did you just cut and paste that from the Globe Website? (Bad)
>
> **A2**: i am not working to any either of the smart or globe.. and that was my opinion when i bought one family sim pack... to where, at my disappointments.. were not all true upon on the run...(sigh)hell network, a misguiding false adverts. (Bad)
>
> **A3**: i heard that globe's signal is not good here and besides calling thru roaming will be more expensive. anyway, in the souq ull find sim roaming for smart, dont know about globe. (Potential)
>
> **A4**: you can go to filipino souq you cab get it for QR 25. (Good)
>
> **A5**: ... available at designated retail outlets in the Philippines. u would easily find it while processing some of ur papers at POEA Ortigas. (Good)

Figure 1: Example for cQA thread.

To efficiently model the contextual information, we propose two neural network-based models with different combination modes of Convolutional Neural Networks (CNN), Long Short Term Memory (LSTM) and Conditional Random Fields (CRF). The first model (ARC-I) is a stacked ensemble of the above networks, which can be seen as a combination of RCNN (Zhou et al., 2015) and LSTM-CRF (Huang et al., 2015). In ARC-I, LSTM is applied on the sequence of encoded QA matching pairs, with CRF on the final layer to memorize transition probabilities over the tag sequence. The main improvement from Zhou et al. (2015) is the addition of backward LSTM and CRF. And the difference from Huang et al. (2015) is that we adopt the LSTM-CRF model in comment-level (actually sentence-level within this paper) sequence tagging, with the help of CNN sentence modelling (Kim, 2014).

ARC-II is a novel and much simpler model, with the integration of the attention mechanism. In ARC-II, the question and its answers are linearly connected in a sequence and encoded by CNN. An attention-based LSTM is then applied on the encoded sequence. Through attention, the model learns how much the question and the context affect the predicting of the current answer. And similar to ARC-I, a CRF layer is appended at last. Using a simple attention function (described in Section 3), ARC-II reduces the size of the parameter space and trains faster than ARC-I.

We carried out extensive experiments on the dataset released by the SemEval-2015 cQA shared task. By adding bidirectional LSTM (Bi-LSTM) and CRF, we achieve 58.96% on macro averaged F_1 (by ARC-I) for answer quality tagging, which improves the state-of-the-art neural network-based method by 2.82% and outperforms the task winner by 1.77%. ARC-II produces promising results with 58.29% on overall F_1 and gains the best performance on the *Good* and *Bad* categories. The contributions of this

work can be summarized as: 1) experiments show that encoding the label dependency is powerful in answer quality tagging; 2) the proposed models achieve state-of-the-art result and considerable improvements on individual categories; 3) as far as we know, this is the first work using the LSTM-CRF architecture on sentence-level sequence tagging.

The roadmap of this paper is: in Section 2 we briefly introduce the literature. The proposed models are detailedly described in Section 3. Experiments are arranged in Section 4. Discussion and conclusion are at last.

2 Related Work

2.1 Answer Quality Tagging

In the literature, methods for answer quality tagging in cQA can be roughly divided into the following four groups: sparse feature-based methods, translation models, parsing trees, and deep CNNs. Sparse feature-based methods are the mostly widely applied and have the longest research duration. Early studies such as Agichtein et al. (2008) and Suryanoto et al. (2009), and later studies such as Yih et al. (2013) and Hou et al. (2015) all achieved not bad results using simple classifiers with manually constructed sparse features. However, feature engineering is time consuming and has low extensibility to other domains. Translation models relied on large-scale of training pairs and can effectively bridge the semantic gap in many cases (Berger et al., 2000; Riezler et al., 2007; Surdeanu et al., 2008). One difficulty is that large parallel training dataset is hard to obtain. The similarity computed from parsing trees is a direct criterion to measure the semantic correlation between two sentences. Typical works are tree edit distance (Punyakanok et al., 2004; Yao et al., 2013) and convolutional tree kernels (Severyn and Moschitti, 2013). But one of the drawbacks of parsing tree-based methods is that most existed parsers perform badly in low-quality sentences (i.e. spoken style language in cQA or daily dialogues).

In the research field of deep learning, Deep Belief Networks (DBN) can also be classified into sparse feature-based methods since they mainly incorporated sparse encodings for multiple features (Wang et al., 2010). The issue of discontinuous word features are later tackled by CNN (Kim, 2014; Hu et al., 2014), following some applications for answer selection in cQA (Yu et al., 2014; Qiu and Huang, 2015; Shen et al., 2015). Zhou et al. (2015)'s work modelled the content correlations using LSTM along QA sequences but they ignored the constraints among quality tags. Their method is insufficient in context learning also due to the neglect of sequence-level dependency modelling. Joty et al. (2015) proposed a graph-cut approach on the judgement of good or bad answers and gains improvement from the baselines. Joty et al. (2016) is an improved version of Joty et al. (2015) in which the authors introduced jointly learning approaches to capture global dependency. The above two works validated the importance of label dependency but they divided the tagging task into two subtasks and built their models in a distinct way.

2.2 Combining Deep Neural Networks and Graphical Models

Wöllmer et al. (2011) was one of the earliest studies that combine neural networks and graphical models. They appended an LSTM layer on top of a Hidden Markov Model in automatic speech recognition. However, the LSTM they applied was only a shallow architecture. Only in recent two or three years did some researchers begin to explore the combination of deep neural networks and graphical models. Huang et al. (2015) proposed the LSTM-CRF model for POS, chunking and NER, and produced state-of-the-art (or close to) accuracies. Lample et al. (2016) applied character and word embeddings in LSTM-CRF and generated good results on NER for four languages. Ma et al. (2016) added a CNN layer on word and character embedding and outperformed previous works in POS tagging and NER.

As far as we know, all the related studies were settled on word-level tasks. In most cases, sentence-level sequence labelling is quite distinct from the word-level in that the dependencies between adjacent sentences are not hard. So that the constraints for tags are weak than word-level tasks, demanding additional information to reinforce the constraints (i.e. sentence meanings).

3 Approach

3.1 ARC-I

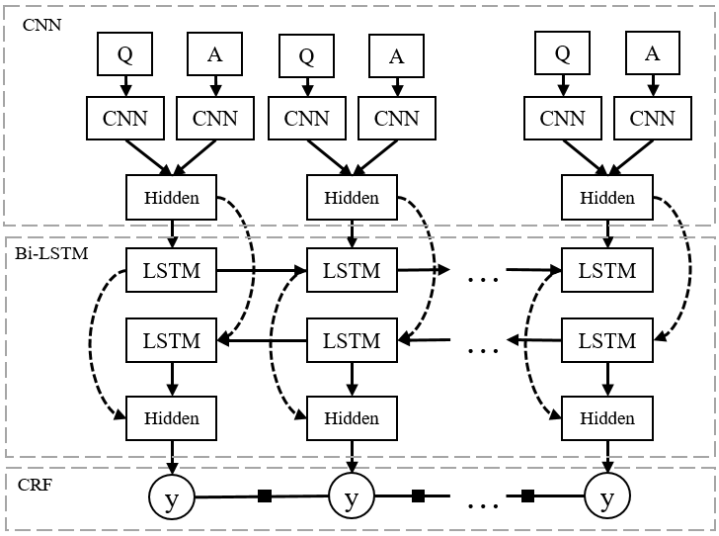

Figure 2: The overview of ARC-I.

The overview of ARC-I is shown in Figure 2. From input to output, the stack of networks are CNN, Bi-LSTM, and CRF. In more detail, in the CNN layer, each QA pair is encoded into a fixed length vector by parallel CNNs together with a following fully connected layer (denoted as *Hidden* in the figures). A Bi-LSTM layer is put next to learn the correlations along the encoded sequence. A fully connected layer and softmax are appended later to generated the predicted tags for the neural network. And at last the cost of the whole network is adjusted using the transition probabilities by a CRF layer over the generated tags.

We incorporate forward linear chain CRF. It jointly models the generative probability (from the output of Bi-LSTM) and the transition probability between adjacent tags from a global perspective, and thus automatically learns the content correlation and constraints among labels.

3.2 ARC-II

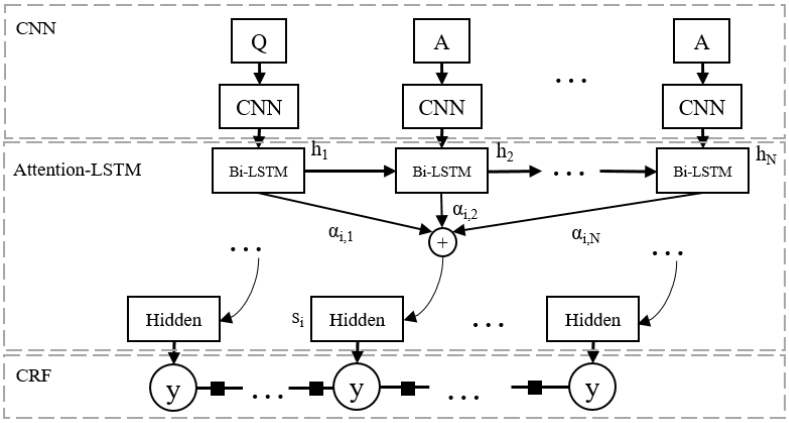

Figure 3: The overview of ARC-II. $\alpha_{i,j}$ stands for the attention weight of the *i*th unit focused on the *j*th encoded element.

Encoding the sequence of QA pairs may introduce extra parameters, the optimization of which can cause much training time. A concise way is to directly learn from the sequence composed of the question and its answers. The model is depicted in Figure 3. In the CNN layer, each question/answer is encoded using a single CNN. In the LSTM layer, an attention based Bi-LSTM is applied to learn the context information along the sequence. Attention mechanism can tackle the bias problem of RNNs (i.e. LSTM/GRU) through computing a weighted distribution of the encoded elements at each time step (Bahdanau et al.,

2015). The distribution reflects the correlations between the current answer and its context. Similar to ARC-I, we also put a CRF layer at last to learn the transitions. To simplify the network, we used a simple form of attention: the attention vector is compromised by the similarities between the current answer and the contextual sentences (the question and the answers).

There are also some variants based on the proposed architectures. By removing backward LSTM or CRF, we validated the contribution of each module to the final result. By replacing CRF with the previously predicted label (denoted as *LP* in the experiment), we tested the superiority of modelling the tag sequence softly over hard encoding. In the following subsections, we explain each applied module in detail.

3.3 CNN

We did not explore deep on sentence modelling such as tuning the depth or dimensions of the nodes, instead we simply adopt the architecture from Kim (2014). The input for CNN is the distributed representation of a sentence, by mapping each word index into its embeddings which were pre-trained in an unsupervised way. Each question or answer is taken as a sentence but not a paragraph due to its length limit (<100 words) in the dataset. To simplify calculation, each sentence is padded to the same length *n* with zero vectors.

We denote the *k*-dimensional embedding for word *j* in a sentence as $z_j \in \mathbb{R}^k$, and thus the sentence can be represented by:

$$z_{1:n} = z_1 \oplus z_2 \oplus ... \oplus z_n \quad (1)$$

where $\oplus$ is the concatenation operator and $z_{i:h}$ stands for $\{z_i, z_{i+1}, \ldots, z_{i+h-1}\}$, i=0,1,…,*n*-*h*+1. The basic CNN operations for sentence modelling include convolution and max-pooling. Convolution is achieved by applying a fixed length sliding window (filter) $w^m \in \mathbb{R}^{h \times k}$ on each word position *i*, such that *n*-*h*+1 convolutional units are generated by:

$$c_i^m = \sigma(w^m \cdot z_{i:i+h} + b^m),\ i = 0,1,...,n-h+1 \quad (2)$$

where σ is the activation function such as Sigmoid or ReLU (Dahl et al., 2013) and b^m is the bias factor for the *m*th layer.

Max-pooling is more popular than mean-pooling in sentence modelling due to the characteristics of natural language. A *d*-max-pooling is to select the maximum unit in every adjacent *d* convolutional units.

$$c_i^m = max(c_j^m, c_{j+1}^m, ..., c_{j+d-1}^m),\ j = 0, d, 2d, ... \quad (3)$$

Following Kim (2014), we applied three convolution filters with lengths *3*, *4* and *5* (with the embedding dimension as the width of the filter), each of which is followed by a max pooling layer to select the most effective structures. Feature maps of size 100 are applied to learn the representations from multiple perspectives.The flattened output vectors for each filter are concatenated as the output of the CNN layer.

3.4 LSTM

LSTM can learn long-distance dependencies through the additions of the *input gate*, *forget gate*, *output gate* and *memory cell* into each recurrent unit. The architecture employed in this paper is analogous to the one introduced by Graves et al. (2013). The input to each LSTM unit is x_t and the output is h_t. The output can be computed through Equations 4-8.

$$i_t = \sigma(W_{xi}x_t + W_{hi}h_{t-1} + b_i) \quad (4)$$

$$f_t = \sigma(W_{xf}x_t + W_{hf}h_{t-1} + b_f) \quad (5)$$

$$c_t = f_t c_{t-1} + i_t \tau(W_{xc}x_t + W_{hc}h_{t-1} + b_c) \quad (6)$$

$$o_t = \sigma(W_{xo}x_t + W_{ho}h_{t-1} + b_o) \quad (7)$$

$$h_t = o_t \theta(c_t) \quad (8)$$

where τ and θ are usually set as the *tanh* function. *W*s and *b*s are weights and biases for each gate. The gate functions can decide what should be passed or retained, thus control whether certain information can be propagated or overwritten across the recurrent network.

In our models, LSTM is applied over the sequence generated by CNN. To bidirectional sequence dependencies, a backward LSTM is arranged over the reverse sequence. We denote the encoded result by forward LSTM as $\vec{h}_{enc}$, backward LSTM as $\overleftarrow{h}_{enc}$ and the concatenation of them as $h_{enc} = [\vec{h}_{enc}, \overleftarrow{h}_{enc}]$.

3.5 Attention-LSTM

Attention mechanism is firstly introduced by Bahdanau et al. (2015) in machine translation. By applying attention in an encoder-decoder framework, the model can figure out the contributions of the encoded elements to the generation of the current unit, using an automatic alignment model. Attention is later popularized in other tasks, such as QA (Hermann et al., 2015) and relation extraction (Liu et al., 2016).

Assume the encoded sequence (question+answers) by LSTM is $h_{enc} \in \mathbb{R}^{(n+1)\times d}$ where d is the output dimension of LSTM. For the ith answer (encoded as s_i), the context vector c_i is computed as a weighted sum of the annotations $\{h_j\}$ (h_j stands for the jth element in h_{enc}):

$$c_i = \sum_{j=1}^{n+1} \alpha_{ij} h_j \tag{9}$$

and the weight α_{ij} is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{n+1} \exp(e_{ik})} \tag{10}$$

where

$$e_{ij} = s_i^T \cdot h_j \tag{11}$$

In ARC-II, we degenerate Eq. 11 into simply computing the similarity between sentences, that is $e_{ij} = h_i^T \cdot h_j$. Thus, the attention context for an answer derives from the similarity with the question as well as the associations with other answers. The correlation with previous answers reflect the information of dependency while the correlation with later answers perhaps convey the information of comments.

3.6 CRF

CRF has been shown superior in many sequence labelling tasks (Lafferty et al., 2001;Sha and Pereira, 2003; Quattoni et al.,2004). Given an observation sequence $X=\{x_1, x_2, \ldots, x_n\}$, CRF jointly models the probability of the entire sequence of labels $Y=\{ y_1, y_2, \ldots, y_n \}$ by using the discriminative probability to y_i given x_i and the transition probability between adjacent labels. The original probability model of CRF is written as:

$$p(Y \mid X; W, b) = \frac{\prod_{i=1}^{n} \psi_i(y_{i-1}, y_i, X)}{\sum_{y' \in \mathcal{Y}(X)} \prod_{i=1}^{n} \psi_i(y'_{i-1}, y'_i, X)} \tag{12}$$

where $\psi_i(y', y, x_i) = \exp(W^T_{y',y} x_i + b_{y',y})$ are potential functions, and $\mathcal{Y}(X)$ denotes the set of possible label sequences given X.

In this work, the observed variable is the encoded sequence (QA sequence) generated by Bi-LSTM. A CRF layer is followed to capture the dependencies between adjacent labels via a state transition matrix. Therefore, Equation 12 is transformed into a simpler form by replacing the potential functions by the outputs of Bi-LSTM. Formally, we denote the outputs of Bi-LSTM as $M=\{m_1, m_2, \ldots, m_n\}$ so as to differentiate from the input of the whole network X (the original word ids). By applying softmax, we obtain the predicted score for each answer on each category.

$$P(y_i = j \mid m_i) = \frac{e^{m_i^T w_j + b_j}}{\sum_k e^{m_i^T w_k + b_k}}, \; j,k = 0,1,2 \tag{13}$$

By adding the transition probability from state y_{i-1} to y_i, the probability of the sequence M is:

$$S(M, Y) = \sum_{i=1}^{n} P(y_i = j \mid m_i) + \sum_{i=1}^{n} T(y_i = j \mid y_{i-1} = k) \tag{14}$$

So the probability for the sequence Y can be yielded by applying a softmax over all possible tag sequences:

$$p(Y \mid M) = \frac{e^{s(M,Y)}}{\sum_{y' \in \mathcal{Y}(M)} e^{s(M,y')}} \tag{15}$$

3.7 Training

We pre-trained the word embeddings with word2vec (Miklov et al., 2013) using the continuous bag-of-words model and the embedding dimension is 100. To achieve fixed length sentence representations (100 in this work), we padded each sentence with zero vectors. The embeddings were not fine-tuned during training because of performance decline in the experiments. We add a dropout layer after CNN with the dropout rate 0.1. The parameters are optimized using AdaDelta (Zeiler, 2012) and the learning rate is initialized as 0.01. The settings for CNN are the same as those by Kim (2014). Following Zhou et al. (2015), we set the dimension of the gates as 360. However, we did not follow the settings of CNN in Zhou et al. (2015)'s work since we found that the settings by Kim (2014) can produce better results when adding Bi-LSTM and CRF. We train the network using a complete end-to-end process. The implementation is under the help of Theano (Al-Rfou et al., 2016) and the tagger codebase[4].

4 Experiments

4.1 Experimental Settings

We carried out the experiments on the dataset released by SemEval-2015 cQA shared task (Nakov et al., 2015). The statistics of the dataset are listed in Table 1. The questions and answers are crawled from Qatar Living Forum. The word embeddings are pre-trained on the untagged cQA data[5]. Each answer is manually tagged as *Good*, *Bad* or *Potential* according to its quality in which *Potential* means the answer is potentially useful to the questioner.

The evaluation metrics include macro averaged precision, recall and F_1 over the target categories, denote as *Prec.*, *Recall*, and F_1 in this section. We also reported the F_1s on individual categories to see whether the models work well on certain labels. The models are trained on the training set, tuned on the development set and tested on the testing set.

	No. of Threads (Q)	No. of Answers (A)	Avg. Length
Train	2600	16541	6.36
Dev	300	1645	5.48
Test	329	1976	6.01

Table 1. Statistical data for SemEval-2015 cQA dataset.

We compare our models with five baselines: 1) The Top-1 system in the shared task which includes almost all the previous popular features, such as translation models and word-based topic models (Tran et al., 2015). This is the state-of-the-art work in the literature. 2) The Top-2 system in the shared task which employs multiple syntactic and semantic features, with ensemble learning as its classification schema (Hou et al., 2015). 3) The state-of-the-art neural network-based system that has an analogous architecture as the proposed models, differing in an insufficient modelling of answer correlation and label dependency (Zhou et al., 2015). 4) A global inference model using graph-cut algorithm but only on binary classification (*Good* or *Bad*) (Joty et al., 2015). 5) An improved version of (Joty et al., 2015) which jointly models the dependencies between tags and learns comment-level classifiers. The main idea of the latter two methods is to divide the tagging task into two subtasks: comment-level tagging and pairwise similarity measuring. However, they still depend on laborious feature engineering.

Further, to validate the role of each component, we compare several sub-networks based on the proposed architectures. The sub-networks are denoted as CNN+LSTM (without backward LSTM and CRF), CNN+LSTM+CRF (without backward LSTM), CNN+Bi-LSTM (without CRF), and CNN+Bi-LSTM+CRF (with all components). To test the importance of CRF in modelling label dependency, we

[4] https://github.com/glample/tagger
[5] http://alt.qcri.org/semeval2015/task3/index.php?id=data-and-tools

replace CRF with the previously predicted label (LP) which is encoded using one-hot format and initialized with zero vectors. The naming rule for LP-based models is similar to those for CRF. The parameters were trained on the training set, and tuned on the development set. Finally, the optimal set of parameters were tested on the testing set.

4.2 Multiclass Tagging

ARC-I The results for multiclass (*Good*, *Bad* and *Potential*) tagging are shown in Table 2[6]. From the fourth column, we see that the best F_1 value is achieved by ARC-I with CNN+Bi-LSTM+CRF, which outperforms the state-of-the-art (Tran et al., 2015) by 1.77% and the best neural network-based method on this dataset (Zhou et al. 2015) by 2.82%. Meanwhile, the optimal values on precision and recall are also generated by this setting. The results imply the importance of both Bi-LSTM (for content dependency) and CRF (for label dependency).

We also notice that the addition of LP gains improvement for both the two variants (LSTM vs. Bi-LSTM), but in varying degrees. Without backward LSTM, LP enhance the baseline by 1.63%, while on the other side, the improvement is about 0.2%.One possible explanation is that backward LSTM plays a more important part than the constraint from the previous label. The F_1 values by both the settings with CRF outperform the baselines heavily, and are also better than other settings without label dependency, indicating the effectiveness of applying CRF. It is also remarkable that the optimal overall precision overpasses 60%, which is a significant improvement over the baselines (2.5%). A good precision is helpful in many real world applications such as QA knowledge base building.

Methods	**Prec.**	**Recall**	**F_1**	**F-Good**	**F-Bad**	**F-Pot.**
Baselines						
(Tran et al., 2015)	57.31	57.20	57.19	78.96	78.24	14.36
(Hou et al., 2015)	57.83	56.82	56.41	76.52	74.32	18.41
(Zhou et al., 2015)	56.41	56.16	56.14	77.31	75.88	15.22
ARC-I						
CNN+LSTM	55.74	55.86	55.75	78.29	77.66	11.30
CNN+LSTM+LP	57.26	57.72	57.32	76.37	77.32	18.28
CNN+LSTM+CRF	58.97	58.41	58.61	78.91	77.06	**19.87**
CNN+Bi-LSTM	57.61	56.98	56.75	79.13	78.44	12.70
CNN+Bi-LSTM+LP	57.22	56.84	56.96	78.23	76.11	16.17
CNN+Bi-LSTM+CRF	**60.33**	**58.86**	**58.96**	79.80	78.63	18.46
ARC-II						
CNN+LSTM	58.45	57.07	56.48	79.84	78.84	10.76
CNN+LSTM+LP	56.34	56.67	55.23	**81.71**	**79.95**	4.04
CNN+LSTM+CRF	59.62	57.98	57.92	81.29	79.07	13.39
CNN+Bi-LSTM	56.47	56.06	55.43	80.63	77.68	7.96
CNN+Bi-LSTM+LP	56.89	56.73	56.39	80.78	78.11	10.28
CNN+Bi-LSTM+CRF	59.83	58.41	58.29	81.22	79.60	14.05

Table 2: Experiment results by the proposed models and baselines. The optimal value for each column is marked bold. F-Pot stands for the F_1 value for the *Potential* category.

ARC-II ARC-II has considerable improvement from the baselines. We notice that with the full network components, it outperforms (Zhou et al., 2015) by 2.15% and (Tran et al. 2015) by 1.10%. Similar trends in the addition of LP or CRF also indicate the importance of label dependency and the superiority of CRF to LP. We also test the model without the attention part, but got very bad results, regardless of

[6] The result files can be downloaded from https://github.com/o0laika0o/CNN-LSTM-CRF-for-cQA-answer-tagging

the addition of backward LSTM. It is mainly due to the loss of the question information from the sequence. We think using attention over the post sequence is a neat way to capture relations from both the question and the context.

A statistical test (*t*-test) is further conducted to evaluate the significance of the improvements by the CRF-based methods (compared with the existed state-of-the-art by Tran et al. (2015)). The results show that the improvement of ARC-I with Bi-LSTM and CRF is mildly significant (p<0.08), and all the other three CRF-based results are statistically significant (p<0.05).

4.3 Binary Classification

Methods	Prec.	Recall	F_1	Acc
(Joty et al., 2015)	78.30	82.93	80.55	79.80
(Joty et al., 2016)	77.3	**86.2**	81.5	80.5
ARC-I	81.54	81.29	81.28	81.33
ARC-II	**82.29**	82.22	**82.22**	**82.24**

Table 3: Results for binary classification.

The last two competitors' methods are binary classification, which removes the *Potential* class from the original target category set. Determining *Good* or *Bad* is a more direct way for the answer selection task and is much closer to real world applications. Joty et al. (2015) and Joty et al. (2016) gain improvements from baselines and the latter one is the state-of-the-art work on binary classification.

We carried out experiments on binary classification and show the results in Table 3. It is notable that we have much better precision and accuracy than their methods. And ARC-II gets the best F_1, which outperforms (Joty et al., 2016) by 0.72%. We may say that ARC-II is more appropriate for binary classification albeit it is inferior in predicting the *Potential* class (seen from Table 2, even 4.04% in F_1 for *Potential* while around 80% for *Good* and *Bad*). From this perspective, although the overall F_1 for multiclass tagging is not the best, ARC-II has its unique advantage.

5 Discussion

We can draw a conclusion that backward LSTM and CRF are both good contributors and complement each other in determining the quality tags and CRF plays a more important part. The backward LSTM intends to capture the comments to the previous answers (i.e. a negative comment perhaps follows a bad answer) and we notice that it improves the original model by 1% in ARC-I. However, in ARC-II, the improvement is not as obvious as the former, mainly due to that the attention mechanism already takes the future steps into account. The employment of CRF brings over 2% promotion from the baselines and can be seen as the most vital factor (the role of LSTM is not so significant which can be drawn from Zhou et al. (2015)). For ARC-II, we also tried other forms of attention such as using more parameters to replace the element-wise dot in this work, but failed to get better results. We think a possible explanation is that the explosive parameters cannot be effectively trained given the current size of the training corpus.

From Table 2, we also recognize that the addition of LP is helpful in some cases but in other cases the improvement is not obvious or even LP does bad to the result (CNN+LSTM+LP for ARC-II). The comparisons between methods with LP and CRF prove that soft constraints are more powerful than hard constraints. Moreover, it is noticed that in most cases, label dependency can boost the performance on *Potential*, which indicates that this class rely more on the contextual information.

6 Conclusion

This paper introduces two models for answer quality tagging in cQA, one with a hierarchical architecture from input to output, and the other with attention mechanism integrated. Through the combination of CNN, Bi-LSTM and CRF, we focus on the modelling of context information, including content correlation and label dependency. Experiments show that we achieve the state-of-the-art overall precision, recall, F_1, as well as the best performance on individual classes. Through the comparisons on label dependency, we discover that CRF is superior to others by learning global constraints. Future development may rise from the import of extra features, such as the user metadata, and a thorough pre-processing

towards the noisy input. With adjustment to the architectural elements or training procedures, we believe the models can be incrementally improved further.

Acknowledgement

This paper is supported in part by National 863 Program of China (2015AA015405), National Natural Science Foundation of China (61402128, 61473101 and 61272383), and Strategic Emerging Industry Development Special Funds of Shenzhen (JCYJ20140417172417105 and JCYJ20140508-161040764). We thank the reviewers for their constructive suggestions on this paper.

Reference

Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. 2008. Finding High-quality Content in Social Media. In Proceedings of the 2008 International Conference on Web Search and Data Mining, pages 183–194. ACM.

Rami Al-Rfou, Guillaume Alain, et al. Theano: A Python Framework for Fast Computation of Mathematical Expressions. 2016. arXiv preprint arXiv:1605.02688. MLA.

Dzmitry Bahdanau, KyungHyun Cho, Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. arXiv preprint arXiv:1409.0473.

Adam Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu Mittal. 2000. Bridging the Lexical Chasm: Statistical Approaches to Answer-finding. In Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, pages 192–199. ACM.

George E Dahl, Tara N Sainath, and Geoffrey E Hinton. 2013. Improving Deep Neural Networks for LVCSR using Rectified Linear Units and Dropout. In Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on, pages 8609–8613. IEEE.

Alan Graves, Navdeep Jaitly, and Abdel-rahman Mohamed. 2013. Hybrid Speech Recognition with Deep Bidirectional LSTM. In Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on, pages 273–278. IEEE.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, et al. 2015. Teaching Machines to Read and Comprehend. Advances in Neural Information Processing Systems. 2015: 1693-1701.

Yongshuai Hou, Cong Tan, Xiaolong Wang, Yaoyun Zhang, Jun Xu, and Qingcai Chen. 2015. Hitszicrc: Exploiting Classification Approach for Answer Selection in Community Question Answering. In Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval, volume 15, pages 196–202.

Haifeng Hu, Bingquan Liu, Baoxun Wang, Ming Liu, and Xiaolong Wang. 2013. Multimodal DBN for Predicting High-quality Answers in cQA Portals. In Proceedings of ACL, pages 843–847.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In Proceedings of Advances in Neural Information Processing Systems, pages 2042–2050.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF Models for Sequence Tagging. Computer Science.

Shafiq Joty, Alberto Barrón-Cedeno, Giovanni Martino Da San, et al. 2015. Global Thread-level Inference for Comment Classification in Community Question Answering. In Proceedings of the Conference on Empirical Methods in Natural Language Processing (EMNLP). 2015, 15.

Shafiq Joty, Lluıs, and Preslav Nakov. Joint Learning with Global Inference for Comment Classification in Community Question Answering. Proceedings of NAACL-HLT. 2016.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. arXiv preprint arXiv:1408.5882.

John Lafferty, Andrew McCallum, and Fernando Pereira. 2001. Conditional Random Fields: Probabilistic Models for Segmenting and Labeling Sequence Data. In Proceedings of ICML.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, Chris Dyer. 2016. Neural Architectures for Named Entity Recognition. arXiv preprint arXiv:1603.01360

Yankai Liu, Shiqi Shen, Zhiyuan Liu, Huanbo Luan, Maosong Sun. 2016. Neural Relation Extraction with Selective Attention over Instances. To be appeared in Proceedings of ACL 2016.

Xuezhe Ma and Eduard Hovy. End-to-end Sequence Labeling via Bi-directional LSTM-CNNs-CRF. 2016. arXiv preprint arXiv:1603.01354.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv preprint arXiv:1301.3781.

Preslav Nakov, Llu´ ıs Marquez, Walid Magdy, Alessandro Moschitti, James Glass, and Bilal Randeree. 2015. SemEval-2015 task 3: Answer Selection in Community Question Answering. SemEval-2015, page 269.

Vasin Punyakanok, Dan Roth, and Wen-tau Yih. 2004. Mapping Dependencies Trees: An Application to Question Answering. In Proceedings of AI&Math 2004, pages 1–10.

Xipeng Qiu and Xuanjing Huang. 2015. Convolutional Neural Tensor Network Architecture for Community based Question Answering. In Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI), pages 1305–1311.

Ariadna Quattoni, Michael Collins, and Trevor Darrell. Conditional Random Fields for Object Recognition. Advances in Neural Information Processing Systems. 2004.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu Mittal, and Yi Liu. 2007. Statistical Machine Translation for Query Expansion in Answer Retrieval. In Annual Meeting-Association for Computational Linguistics, volume 45, page 464.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic Feature Engineering for Answer Selection and Extraction. In EMNLP, pages 458–467.

Fei Sha and Fernando Pereira. Shallow Parsing with Conditional Random Fields. In Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1. Association for Computational Linguistics, 2003.

Yikang Shen, Wenge Rong, Zhiwei Sun, Yuanxin Ouyang, and Zhang Xiong. 2015. Question/answer Matching for cQA System via Combining Lexical and Sequential Information. In Twenty-Ninth AAAI Conference on Artificial Intelligence.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. 2008. Learning to Rank Answers on Large Online QA Collections. In ACL, volume 8, pages 719–727.

Maggy Anastasia Suryanto, Ee Peng Lim, Aixin Sun, and Roger HL Chiang. 2009. Quality-aware Collaborative Question Answering: Methods and Evaluation. In Proceedings of the second ACM International Conference on Web Search and Data Mining, pages 142–151. ACM.

Quan Hung Tran, Vu Tran, Tu Vu, Minh Nguyen, and Son Bao Pham. 2015. Jaist: Combining Multiple Features for Answer Selection in Community Question Answering. In Proceedings of the 9th International Workshop on Semantic Evaluation, SemEval, volume 15, pages 215–219.

Baoxun Wang, Xiaolong Wang, Chengjie Sun, Bingquan Liu, and Lin Sun. 2010. Modelling Semantic Relevance for Question-answer Pairs in Web Social Communities. In Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, pages 1230–1238. ACL.

Martin Wöllmer, Erick Marchi, Stefano Squartini, and Bjö rn Schuller. 2011. Multi-stream LSTM-HMM Decoding and Histogram Equalization for Noise Robust Keyword Spotting. Cognitive Neurodynamics, 2011, 5(3): 253-264.

Xuchen Yao, Benjamin Van Durme, Chris CallisonBurch, and Peter Clark. 2013. Answer Extraction as Sequence Tagging with Tree Edit Distance. In HLTNAACL, pages 858–867. Citeseer.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question Answering using Enhanced Lexical Semantic Models.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. arXiv preprint arXiv:1412.1632.

Matthew D Zeiler. 2012. Adadelta: An Adaptive Learning Rate Method. arXiv preprint arXiv:1212.5701.

Xiaoqiang Zhou, Baotian Hu, Qingcai Chen, Buzhou Tang, and Xiaolong Wang. 2015. Answer Sequence Learning with Neural Networks for Answer Selection in Community Question Answering. arXiv preprint arXiv:1506.06490.