Studio Ousia's Quiz Bowl Question Answering System at NIPS HCQA 2017

Ikuya Yamada
Studio Ousia
3-27-15 Jingumae, Shibuya, Tokyo, Japan
ikuya@ousia.jp

Ryuji Tamaki
Studio Ousia
3-27-15 Jingumae, Shibuya, Tokyo, Japan
ryuji@ousia.jp

Hiroyuki Shindo
Nara Institute of Science and Technology
8916-5 Takayama, Ikoma, Nara, Japan
shindo@is.naist.jp

Yoshiyasu Takefuji
Keio University
5322 Endo, Fujisawa, Kanagawa, Japan
takefuji@sfc.keio.ac.jp

Abstract

This paper describes the question answering system that we submitted to the Human–Computer Question Answering Competition at the Thirty-first Annual Conference on Neural Information Processing Systems (NIPS). The competition requires participants to address a factoid question answering task referred to as *quiz bowl*. We use two novel neural network models to address the task. We combine these models with conventional information retrieval models using a supervised learning to rank algorithm. Our system achieved the best performance among the systems submitted to the competition.

1 Introduction

This paper presents the system we submitted at the Human–Computer Question Answering Competition held at the Thirty-first Annual Conference on Neural Information Processing Systems (NIPS) 2017. This competition requires a system to address a unique factoid question answering (QA) task referred to as *quiz bowl*. This task has been studied frequently [1, 2, 3, 4]. Given a question, the system is required to guess the entity that is described in the question (see Table 1). One unique characteristic of this task is that the question is given one word at a time, and the system can output an answer at any time. Moreover, the answer must be an entity that exists in Wikipedia.

To address this task, we propose a system that combines the outputs of neural network models and conventional information retrieval (IR) models using supervised machine learning. We adopt a point-wise learning to rank algorithm to combine these outputs; given a question text, our system assigns a relevance score to answers, and output an answer if the score of the top answer exceeds a predefined threshold.

We use two types of neural network models to address this task. Similar to past work [2, 3, 4], our first neural network model directly solves the task by casting it as a text classification problem. As the entities mentioned in the question (e.g., *Gregor Samsa* and *The Metamorphosis* in the question shown in Table 1) play a significant role in guessing the answer, we use words and entities as inputs to the model. We train the neural network model based on Deep Averaging Networks [3] to predict the answer from a set of words and entities that appear in the question.

Given a question, our second neural network model predicts the types of the answers. For example, the expected entity types of the question shown in Table 1 are *author* and *person*. We train the neural network model to predict the entity types of the answer to a question. We address this task as a text

31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA.

Question: The protagonist of a novel by this author is evicted from the Bridge Inn and is talked into becoming a school janitor by a character whose role is often translated as the Council Chairman. A character created by this writer is surprised to discover that he no longer likes the taste of milk, but enjoys eating rotten food. The quest for Klamm, who resides in the title structure, is taken up by K in his novel The Castle. For 10 points, name this author who wrote about Gregor Samsa being turned into an insect in "The Metamorphosis."
Answer: *Franz Kafka*

Table 1: Example of a quiz bowl question

classification problem. In particular, we adopt a convolutional neural network (CNN) [5] to predict the corresponding entity types for a question.

The outputs of these neural network models are used as the features of the learning to rank model. We train the learning to rank model by combining these neural network-based features with other features including the outputs of conventional IR models. The neural network models and learning to rank model are trained using our quiz bowl QA dataset, which is developed based on two existing datasets. As a result, our system achieves the best performance among the systems submitted to the competition.

2 Proposed system

In this section, we provide an overview of the proposed system. We first describe the data used to develop our system and then present the technical details of our system.

2.1 Data

We used several data sources to develop our system. First, we used the Protobowl QA dataset, which is the official dataset provided by the competition organizers.[1] The dataset contained over 100,000 quiz bowl questions and their answers, which were originally obtained from the Protobowl website[2]. The dataset contained several questions whose answers did not exactly match their corresponding Wikipedia titles. We resolved the answers to the corresponding Wikipedia titles using simple string matching methods and a crowd-sourcing service and excluded the questions whose answers could not be matched to Wikipedia.

In addition, we concatenated the Protobowl QA dataset with the public QA dataset provided by Iyyer et al. [2], containing 20,407 quiz bowl questions and their answers.[3] Unlike the Protobowl dataset, the answers contained in this dataset were provided as Wikipedia titles. We removed the duplicated questions from the concatenated dataset. As a result, our final QA dataset contained 101,043 question–answer pairs.

We used Wikipedia and Freebase as the data sources of various models. We used a Wikipedia dump generated in June 2016 and the latest Freebase data dump obtained from the website[4].

2.2 Neural network models

We develop the following two neural network models to solve the QA task: *Neural Quiz Bowl Solver* and *Neural Entity Type Predictor*.

2.2.1 Neural Quiz Bowl Solver

Neural Quiz Bowl Solver addresses the task as a text classification problem over possible answers. We use Deep Averaging Network, which has been reported to perform well on the quiz bowl task [3].

[1] The dataset was obtained from the competition website: `https://sites.google.com/view/hcqa/data`.

[2] `http://protobowl.com/`

[3] The dataset was obtained from the author's website: `https://cs.umd.edu/~miyyer/qblearn/`

[4] `https://developers.google.com/freebase/`

Given the words (w_1, ..., w_N) and Wikipedia entities (e_1, ..., e_K) that appear in question D, our model first computes the word-based vector representation, $\mathbf{v}_{D_w}$, and the entity-based vector representation, $\mathbf{v}_{D_e}$, of question D by averaging the vector representations of the words and entities, respectively.

$$\mathbf{v}_{D_w} = \frac{1}{N}\sum_{n=1}^{N}\mathbf{W}_w\mathbf{a}_{w_n},\ \mathbf{v}_{D_e} = \frac{1}{K}\sum_{n=1}^{K}\mathbf{W}_e\mathbf{b}_{e_n}, \tag{1}$$

where $\mathbf{a}_w \in \mathbb{R}^d$ and $\mathbf{b}_e \in \mathbb{R}^d$ are the vector representations of a word and an entity, respectively, and $\mathbf{W}_w \in \mathbb{R}^{d\times d}$ and $\mathbf{W}_e \in \mathbb{R}^{d\times d}$ are projection matrices. Then, the vector representation of question $\mathbf{v}_D$ is computed as the element-wise sum of $\mathbf{v}_{D_w}$ and $\mathbf{v}_{D_e}$.

$$\mathbf{v}_D = \mathbf{v}_{D_w} + \mathbf{v}_{D_e}. \tag{2}$$

The probability that entity e_t is the answer of question D is defined using the following softmax function:

$$P(e_t|D) = \frac{\exp({\mathbf{c}_{e_t}}^\top \mathbf{v}_D)}{\sum_{e'\in\Gamma}\exp({\mathbf{c}_{e'}}^\top \mathbf{v}_D)}, \tag{3}$$

where Γ is a set containing all answers, and $\mathbf{c}_e \in \mathbb{R}^d$ denotes the vector representation of answer e. Further, we use categorical cross entropy as a loss function.

Because the model requires the list of entities in a question, we automatically annotate entity names using a simple entity linking method. The method is based on *keyphraseness* [6], which is the probability that an entity name is used as an anchor in Wikipedia. We detect an entity name if its keyphraseness is larger than 2%. Furthermore, as an entity name can be ambiguous (e.g., *Washington* can refer to the city and state in the U.S., a person's name, etc.), we use an entity name if it refers to a single entity with a probability of 95% or more in Wikipedia. The entities referred by the detected entity names are used as inputs to the model.

The model is trained by iterating over the QA dataset described in Section 2.1. We randomly select 10% questions from the dataset as a validation set and use the remaining questions to train the model. Furthermore, because a question is given one word at a time, the model must perform accurately for incomplete questions. To address this, we truncate a question at a random position before inputting it to the model during training.

We use Entity-Vector [7], which is a method for learning the vector representations of words and entities from Wikipedia, to initialize the vector representations of words ($\mathbf{a}_w$), entities ($\mathbf{b}_e$), and answers ($\mathbf{c}_e$). We train the representations using the Wikipedia dump described in Section 2.1. Note that we use the same pretrained entity representations to initialize the representations of entities and answers.

The proposed model is implemented using PyTorch[5] and trained using minibatch stochastic gradient descent (SGD) on a GPU. The minibatch size is fixed as 32, the learning rate is automatically controlled by Adam [8], and the number of representation dimensions is set as $d = 300$. In addition, we use early stopping based on the accuracy on the validation set to locate the best epoch. To prevent overfitting, we randomly exclude the words and entities in the question with a probability of 0.5 [3].

We compute two scores for each answer using this model, i.e., (1) the predicted probability ($P(e_t|D)$) and (2) the unnormalized value inputted to the softmax function (${\mathbf{c}_{e_t}}^\top \mathbf{v}_D$).

2.2.2 Neural Entity Type Predictor

Neural Entity Type Predictor aims to predict the entity types for a question. For example, when the target question is the one shown in Table 1, the target entity types are *person* and *author*. We address this task as a multiclass text classification task over entity types using the CNN proposed by Kim [5]. We use rectified linear units as the activation function and the average binary cross entropy as the loss function.

We use the FIGER entity type set [9], which consists of 112 fine-grained entity types, as the target entity types. We automatically assign entity types to each answer by resolving the answer's Wikipedia

[5] `http://pytorch.org`

entity to its corresponding entity in Freebase and computing FIGER entity types based on the mapping[6] and Freebase data.

We train two separate models with the following different target entity types: all *fine-grained* entity types and only eight *coarse-grained* entity types (i.e., *person*, *organization*, *location*, *product*, *art*, *event*, *building*, and *other*). In the former setting, we address the task as a *multilabel* text classification problem because most answers have multiple entity types. Because the CNN model proposed by Kim does not address multilabel classification, we slightly modify the model by replacing its output softmax layer with a layer consisting of multiple binary sigmoid units, each of which corresponds to each entity type. Furthermore, we use the unmodified version of Kim's CNN model for the latter setting.[7]

These two models are trained using the same settings. The models are trained using SGD on a GPU, the minibatch size is fixed as 32, and the learning rate is controlled by Adamax [8]. We use filter window sizes of 2, 3, 4, and 5 and 1,000 feature maps for each filter. Moreover, we use the GloVe word embeddings [10] trained on the 840 billion Common Crawl corpus to initialize the word representations. We randomly select 10% questions from the dataset as a validation set and use the remaining questions to train the model. We use early stopping based on the validation set to locate the best epoch. Furthermore, similar to the neural network model explained previously, a question is truncated at a random position before it is input to the models. The models are implemented using PyTorch[8]

Given a question and an answer, each model outputs two scores, i.e., the sum and maximum probability, based on the predicted probabilities of the entity types assigned to the answer.

2.3 Information retrieval models

Similar to previous studies [11, 12, 2], we use conventional IR models to enhance the performance of our QA system. In particular, we compute multiple relevance scores against the documents associated to the answer using the words in a question as a query.

Specifically, for each answer contained in the dataset, we create the target documents using the following two types of data sources: (1) *Wikipedia text*, which is the page text in the answer's Wikpedia entry, and (2) *dataset questions*, which are the questions contained in our QA dataset (see Section 2.1) and associated to the answer. Regarding Wikipedia text, we use two methods to create documents for each answer, i.e., treating page text as a single document and treating each paragraph as a separate document. We adopt two similar methods for dataset questions, i.e., creating a single document by concatenating all questions associated to the answer and treating each question as a separate document. Further, because the latter methods of both data sources create multiple documents for each answer, we first compute the relevance scores for all documents and reduce them by selecting their maximum score.

We preprocess the questions and documents by turning all words to lowercase, removing stop words[9], and performing snowball stemming. We use two scoring methods, i.e., Okapi BM25 [12] and the number of common words between the question and document. Further, we generate four types of queries for a question using (1) all of its words, (2) all of its words and bigrams, (3) its noun words, and (4) its proper noun words.[10] There are four target document sets, two scoring methods, and four query types; thus, given a question and an answer, we compute 32 relevance scores.

2.4 Learning to rank

Given a question as an input, the learning to rank model assigns a relevance score to each answer based on the outputs of the neural network models and IR models described above. We use gradient boosted regression trees (GBRT) [13], which is a point-wise learning to rank algorithm that provides state-of-the-art performance [14, 15]. GBRT consists of an ensemble of regression trees, and given

[6]The mapping was obtained from FIGER's GitHub repository: `https://github.com/xiaoling/figer/`

[7]We neglect the answers with multiple coarse-grained entity types during training.

[8]`http://pytorch.org/`

[9]We use the list of stop words contained in the scikit-learn library.

[10]We use Apache OpenNLP to detect noun and proper noun words.

a question, it predicts a relevance score for each answer. We use the GBRT implementation in LightGBM[11], and logistic loss is used as the loss function.

Furthermore, to reduce computational cost, we assign relevance scores only for a small number of top answer candidates. We generate answer candidates using the union of the top five answers with the highest scores among the scores generated by Neural Quiz Bowl Solver and the IR models.

The features used in this model are primarily based on the scores assigned by the neural network models and IR models described above. For each score, we generate three features using (1) the score, (2) its ranking position in the answer candidates, and (3) the margin between the score and the highest score among the scores of the answer candidates. We use three additional features, i.e., the number of words and sentences in the question and the number of FIGER entity types associated to the answer.

The model is trained using our QA dataset (see Section 2.1). To maintain accuracy for incomplete questions, we generate five questions truncated at random positions per question. One problem is that we use the same QA dataset for training the neural network models and the corpus of the IR models; this likely causes severe overfitting. To address this, we use two methods during the training of the learning to rank model. We adopt stacked generalization [16] based on 10-fold cross validation to compute scores based on the neural network models. Regarding the IR models, we dynamically exclude the question used to create the input query from the corpus.

Our system outputs an answer if the relevance score of the top answer exceeds a predefined threshold, which is set as 0.6. Furthermore, as predictions frequently become unstable when the question is short, we restrict the system not to output an answer if the number of words in the question is lower than 15.

3 Competition results

Table 2 shows the accuracy scores of the top 3 systems submitted to the competition. Our system achieves the best performance by a wide margin. Furthermore, to evaluate the actual performance of the systems in the quiz bowl game, simulated pairwise matches were performed between the systems following the official quiz bowl rules. Our system outperformed System 1 (our system: 1220 points; System 1: 60 points) and System 2 (our system: 1145 points; System 2: 105 points) by considerably wide margins.

Name	Accuracy
Our system	**0.85**
System 1	0.675
System 2	0.6
Baseline	0.55

Table 2: Accuracy scores of the top 3 systems

4 Conclusions

In this paper, we describe the system that we submitted to the Human–Computer Question Answering Competition held at NIPS 2017. We proposed two novel neural network models and combined these two models with conventional IR models using a supervised point-wise learning to rank algorithm. Our system achieved the best performance among the systems submitted to the competition.

References

[1] Jordan Boyd-Graber, Brianna Satinoff, He He, and Hal Daume III. Besting the Quiz Master: Crowdsourcing Incremental Classification Games. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 1290–1301, 2012.

[11] `https://github.com/Microsoft/LightGBM`

[2] Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. A Neural Network for Factoid Question Answering over Paragraphs. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 633–644, 2014.

[3] Mohit Iyyer, Varun Manjunatha, Jordan Boyd-Graber, and Hal Daumé III. Deep Unordered Composition Rivals Syntactic Methods for Text Classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1681–1691, 2015.

[4] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Learning Distributed Representations of Texts and Entities from Knowledge Base. *arXiv preprint arXiv:1705.02494v2*, 2017.

[5] Yoon Kim. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1746–1751, Doha, Qatar, 2014.

[6] Rada Mihalcea and Andras Csomai. Wikify!: Linking Documents to Encyclopedic Knowledge. In *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management*, pages 233–242, 2007.

[7] Ikuya Yamada, Hiroyuki Shindo, Hideaki Takeda, and Yoshiyasu Takefuji. Joint Learning of the Embedding of Words and Entities for Named Entity Disambiguation. In *Proceedings of the 20th SIGNLL Conference on Computational Natural Language Learning*, pages 250–259, 2016.

[8] Diederik Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980v9*, 2014.

[9] Xiao Ling and Daniel S. Weld. Fine-Grained Entity Recognition. In *Proceedings of the Twenty-Sixth AAAI Conference on Artificial Intelligence*, pages 94–100, 2012.

[10] Jeffrey Pennington, Richard Socher, and Christopher D Manning. GloVe: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1532–1543, 2014.

[11] Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. Question Answering Using Enhanced Lexical Semantic Models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1744–1753, 2013.

[12] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. Deep Learning for Answer Sentence Selection. *arXiv preprint arXiv:1412.1632v1*, 2014.

[13] Jerome H. Friedman. Greedy Function Approximation: A Gradient Boosting Machine. *The Annals of Statistics*, 29(5):1189–1232, 2001.

[14] O Chapelle and Y Chang. Yahoo! Learning to Rank Challenge Overview. In *Proceedings of the Learning to Rank Challenge*, volume 14 of *Proceedings of Machine Learning Research*, pages 1–24, 2011.

[15] Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly, Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, Jean-Marc Langlois, and Yi Chang. Ranking Relevance in Yahoo Search. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 323–332, 2016.

[16] David H Wolpert. Stacked generalization. *Neural networks*, 5(2):241–259, 1992.