A Bayesian End-to-End Model with Estimated Uncertainties for Simple Question Answering over Knowledge Bases

Linhai Zhang[a], Chao Lin[a], Deyu Zhou[a,*], Yulan He[b], Meng Zhang[a]

[a]*School of Computer Science and Engineering, Southeast University, Nanjing, Jiangsu Province, 210096, China*
[b]*Department of Computer Science, University of Warwick, UK*

Abstract

Existing methods for question answering over knowledge bases (KBQA) ignore the consideration of the model prediction uncertainties. We argue that estimating such uncertainties is crucial for the reliability and interpretability of KBQA systems. Therefore, we propose a novel end-to-end KBQA model based on Bayesian Neural Network (BNN) to estimate uncertainties arose from both model and data. To our best knowledge, we are the first to consider the uncertainty estimation problem for the KBQA task using BNN. The proposed end-to-end model integrates entity detection and relation prediction into a unified framework, and employs BNN to model entity and relation under the given question semantics, transforming network weights into distributions. Therefore, predictive distributions can be estimated by sampling weights and forward inputs through the network multiple times. Uncertainties can be further quantified by calculating the variances of predictive distributions. The experimental results demonstrate the effectiveness of uncertainties in both the misclassification detection task and cause of error detection task. Furthermore, the proposed model also achieves comparable performance compared to the existing state-of-the-art approaches on SimpleQuestions dataset.

*Corresponding author.
Email addresses: lzhang472@seu.edu.cn (Linhai Zhang), c.lin@seu.edu.cn (Chao Lin), d.zhou@seu.edu.cn (Deyu Zhou), Yulan.He@warwick.ac.uk (Yulan He), m.zhang@seu.edu.cn (Meng Zhang)

Keywords: question answering over knowledge bases, bayesian neural network, uncertainty estimation

1. Introduction

With the ever-growing amount of data, knowledge bases (KB) such as Freebase [1] and WikiData[1] become larger and larger. The facts in the real world are often represented as triplets (*subject entity, predicate, object entity*) in knowledge bases, where the subject entity and the object entity refer to two real-world entities and predicate refers to the relation between subject entity and object entity. Such a large volume of data and complex structures make it extremely hard for users to access the information efficiently. To address this issue, Question Answering over Knowledge Bases (KBQA) [2, 3, 4, 5, 6, 7, 8, 9] was proposed. KBQA systems aim to automatically translate natural language questions posed by users into structured queries, e.g. SPARQL, and return the entities in KB as the answers which attract massive attention [6, 10]. However, the KBQA problem is far from solved as it involves multiple subtasks such as entity linking [11, 12] and predicate detection [5, 13]. In this paper, we focus on the simple question answering problem, which consists of the majority of KBQA questions. The simple question can be answered with a single fact (subject, predicate, object) in the knowledge base, which constitutes the majority of questions asked on the web. The task can be formulated as finding the best matches of subject and predicate for the given question. For example, for the question "*what is a compatible ingredient with a gluten-free diet?*", the task aims to find the subject-predicate pair (m.034n2g_[Gluten-free Diet], food/dietary_restriction/compatible_ingredients) in KBs. Based on the found pair, final answer (m.057xpf_[Breckland Thyme]) can be easily retrieved in a single fact using SPARQL queries.

There are two mainstream research directions for the KBQA task. One

[1] `https://www.wikidata.org/wiki/Wikidata:Main_Page`

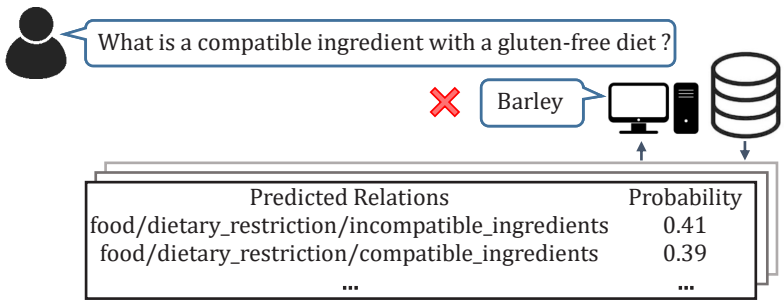

Figure 1: Example of uncertainties in KBQA model.

category is multi-staged methods that tend to break down the KBQA task into subtasks [7, 14, 15]. For example, in AskHow [16], a natural language question was passed through five modules including Part-of-Speech (POS) tagging, template-fitting, relation extraction, token merging, and entity mapping before translated into SPARQL query. However, such approaches suffer from error and uncertainty propagation problems. To deal with these problems, end-to-end based approaches have been developed, leaving all the decisions to the model itself [3, 6]

Despite the promising results, current approaches are unable to estimate uncertainties of their predictions, which is crucial for the model's reliability and interpretability. As shown in Figure 1, for the question "*What is a compatible ingredient with a gluten-free diet?*", the model may be uncertain about two conflicting predicates, *incompatible_ingredients* and *compatible_ingredients* and output a wrong answer. Such wrong prediction may be fatal for gluten-sensitive users. More importantly, as black-box models, neural network-based KBQA systems can provide nothing but the answers, which are not interpretable. This uninterpretability makes even high-performance KBQA systems unreliable. Because people cannot judge when the system makes an error while the cause of such error, as illustrated in the example, may be unacceptable. On the other hand, if we could measure how uncertain the model is in its prediction, more reliable decisions could be made with such information. Take Figure 1 as an example, the model is quite uncertain about the answer *Barley*,

so people could refer to another information source and take the decision. Therefore, instead of solely predicting the answer, it is important to measure the uncertainty in model predictions. Recent developments on Bayesian Neural Network (BNN) [17, 18, 19] make it feasible to quantify such uncertainties. BNNs estimate distributions over the prediction space by placing distributions over network weights. Therefore, uncertainties of the model predictions could be estimated with predictive distributions by calculating their spread.

Moreover, current end-to-end KBQA methods often rely on semantic matching in the embedding space based on the semantic similarity between a given question and the candidate resources including entities and predicates in KBs and return the nearest neighbors as the correct resources for the given question [3, 6]. In such frameworks, (question-subject) and (question-predicate) are often matched separately, ignoring the interaction between each other. For example, for the question "*what is the subject of writing home*", two candidate entities corresponding to Freebase IDs: "$m.02hvp4r$" and "$m.04v0\_pk$" will be extracted. they have the same entity name "*writing home*" which output the same score in the matching procedure of (question-subject). However, they are attached with different predicates *book.written.subjects* and *book.book edition.binding*. These predicates can help to distinguish the entities with the same scores.

In this paper, we propose a novel Bayesian end-to-end KBQA model to estimate two types of uncertainties, model uncertainty and data uncertainty, of the predictions, the former one measures the how well the model fits the data and the latter one measure the inherent noise in the data. The model is proposed to select entity and predicate simultaneously considering the relevance between entities and predicates existed in KBs. In specific, both the entity with its context and the candidate predicates are encoded by a Bayesian BiLSTM. The relevance of the entity and predicate pair is calculated based on their representation similarity. Experimental results show that the proposed model outperforms existing state-of-the-art end-to-end approaches. The effectiveness of the proposed uncertainty measures is further confirmed on

the misclassification detection and the cause of error analysis.

The contributions of our work in this paper are listed, more succinctly, as follows:

- From a practical perspective, estimating uncertainties of model prediction is crucial for the QA system, especially in safety-related areas. Traditional KBQA methods ignore the uncertainty existed in data and models which lacks reliability and interpretability. We are interested in exploring a neural network-based model to obtain the answer and its confidence simultaneously. To this end, a novel Bayesian-based KBQA model with uncertainty estimated is proposed. To our best knowledge, we are the first one to incorporate BNN in KBQA. Experimental results on several tasks indicate the efficiency of the proposed uncertainty measures.

- Multi-staged approaches for simple question answering often contain several separate components which cause error and uncertainty propagation problem. Thus we develop a novel end-to-end framework to jointly select entity and predicate, considering their interaction existed in KB in one single training procedure. Furthermore, it can be easily retrained or reused for a different domain. Experimental results on SimpleQuestions dataset show that the proposed model achieves comparable performance compared to the existing state-of-the-art approaches.

The practical significance of this work is that the proposed approach estimates the uncertainties of predictive results for the KBQA system, which is crucial for safety-related areas. Moreover, the proposed approach achieves comparable performance compared with some state-of-the-art approaches and can be easily adapted to other domain because of the end-to-end framework. The rest of the paper is organized as follows. Section 2 reviews the related literature on deep learning for KBQA and uncertainties quantification in deep learning. In Section 3, a detailed description of the proposed approach is presented. Section 4 introduces the experiment details. Finally, the paper is concluded in Section 5.

2. Related Work

Our work is related to two lines of research, described as follows.

2.1. Deep learning for KBQA

With the development of deep learning, most recent approaches to KBQA are based on neural networks. A majority of literature focuses on the ways of measuring the semantic similarity between question and candidate triples in KB as we assume that correct resources in KB should be close to questions in the semantic vector space.

Li and Wei [20] exploited three Convolutional Neural Networks (CNNs) to represent questions differently when facing different aspects. Golub and He [8] developed an attention-enhanced encoder-decoder architecture where the attention was introduced to handle long sequences. Lukovnikov et al. [6] trained an end-to-end model, which employed a word/character-level encoder to alleviate the problem of Out-of-Vocabulary. Yin and Chang [14] split a question into entity mentions and question patterns, and used CNNs to model the textual information of the question and KB resources by incorporating attentive pooling. Dai and Li [7] presented a word-level Recurrent Neural Network (RNN) based approach and used the representations learned specifically for Freebase resources. Yu and Yin [21] focused on predicting predicates and proposed a hierarchical Residual BiLSTM model to compare questions and candidate predicates names via different levels of abstraction. Das et al. [22] used matrix factorization to incorporate corpus into KBs, and LSTM to model the question. Hao et al. [9] learned the distributional representations of questions and candidate answers in a unified deep-learning framework. Hao and Liu [15] used a BiLSTM with CRF-tagging-based model to conduct entity extraction and introduced a pattern-revising procedure to improve the performance. Mohammed et al. [10] viewed each predicate as a label category, and exploited the deep classification model to perform predicate prediction. It should be pointed out that [7, 14, 15] exploited extra information sources including

Freebase entity linking results and learned segmentation models to improve model performances. Huang et al. [23] utilized knowledge graph embedding to enhance the quality of entity representation and predicate representation in the matching model. However, all the aforementioned approaches ignore model uncertainties in model predictions.

2.2. Quantifying Uncertainties in Deep Learning

In general, the ways of estimating uncertainties in deep learning models can be categorized into two classes: non-Bayesian approaches and Bayesian-based approaches. For non-Bayesain approaches, Lakshminarayanan et al. [24] directly trained an ensemble of deep neural networks to obtain a set of predictions and estimate uncertainties. Some researchers focused on explicitly training the model to produce the prediction distribution by minimizing the Kullback-Leibler (KL) divergence between the model output and synthetic data distribution and estimating uncertainties based on the distribution [25, 26].

The other class is based on BNNs [27, 17]. In BNNs, weights in networks are considered as random variables. A prior distribution is placed over the weights in BNNs and posterior distribution is inferred with the Bayes' rule. With the posterior distribution, the prediction distribution can be calculated by integration and the uncertainties can be calculated based on the prediction distribution. Traditional BNN optimization mostly relied on variational inference [28, 29, 18, 30] or stochastic gradient MCMC methods [31, 32] to approximate the intractable posterior distribution. Recently, a new approach called Monte Carlo Dropout (MC dropout) was proposed [19], which proves a neural network with dropout can be regarded as an approximation to a BNN. It keeps the dropout unit activated to get an ensemble of predictions by multiple stochastic forward passes, which has been successfully applied in computer vision [33, 34].

There has been some research trying to model uncertainties in natural language processing tasks. Zhang et al [35] tried to measure uncertainty in document classification with MC dropout. Xiao and Wang [36] quantified model

uncertainty and data uncertainty in a series of NLP tasks including sentiment analysis, named entity recognition and language modeling with BNNs. To our best knowledge, there are no other works on quantifying uncertainties in KBQA models.

3. Methodologies

3.1. Problem Statement

The task of simple KBQA can be defined as follows. Let $G = (S, P, O)$ be the knowledge bases, where S represents the set of subject entities, O represents the set of object entities and P represents the set of predicates between the subject entities and the object entities. Let Q be the set of simple questions, for each question $q \in Q$, the goal is to automatically return the right object as the answer from a single fact in knowledge bases [2]. That is, in simple question answering setting, a question can be answered with a single <subject, relation/predicate, object> KB tuple. It can also be formulated as finding the correct match of the subject $\hat{s} \in S$ and predicate $\hat{p} \in P$, given the question q.

$$\hat{s}, \hat{p} = \underset{s \in S,\ p \in P}{\arg\max}\ Prob(s, p \mid q) \tag{1}$$

3.2. The Proposed Model

We propose a KBQA model based on BNN. The overall architecture of the proposed KBQA model is shown in Figure 2. To predict the subject entity $\hat{s}$ and predicate $\hat{p}$ for a given question q, it

(A) generates the set of candidate entities $C_s = \{s_1, s_2, \ldots, s_n\}$, the set of candidate predicates $C_p = \{p_1, p_2, \ldots, p_m\}$ and the set of context of entities in question $C_{\tilde{s}} = \{\tilde{s}_1, \tilde{s}_2, \ldots, \tilde{s}_n\}$ based on the question q and the knowledge base G;

(B) represents each candidate entity's context $\tilde{s}_i \in C_{\tilde{s}}$ as a vector $\mathbf{h}^{s_i}$ and each candidate predicate $p_j \in C_p$ as a vector representation $\mathbf{h}^{p_j}$ by the Bayesian BiLSTM encoders which is illustrated in Figure 3.

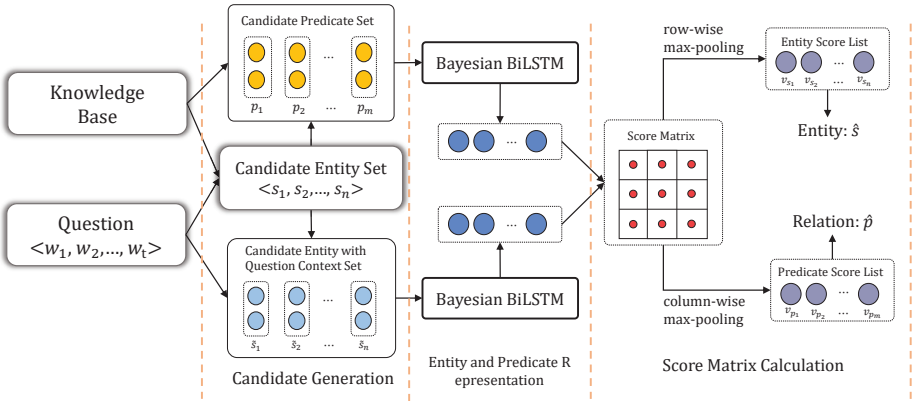

Figure 2: Architecture of the proposed model

(C) calculates a similarity score matrix $\mathbf{T}_{mn}$ by dot product between the representation of each entity $\mathbf{h}^{s_i}$ and predicate $\mathbf{h}^{p_j}$ to jointly select best-matched entity-predicate pair $\langle \hat{s}, \hat{p} \rangle$.

With the jointly selected entity-predicate pair, an entropy-based loss function is applied to optimize the model parameters. Details of each step in the model are elaborated in the following.

3.2.1. Candidate Sets Generation

Since the knowledge base contains tens of thousands of facts, it is extremely time-consuming to perform matching process. As such, it is necessary to generate the candidate sets for entities and predicates based on the question in order to shrink the learning space.

- **Set of Candidate Entities**: Each question is first converted into lower case, from which n-grams (n from 1 to L) are extracted. Candidate entity set C_s for question q is generated by matching n-grams with entity names stored in the knowledge base G.

- **Set of Candidate Predicates**: For each candidate entity s_i in set C_s, we construct its corresponding predicate set C_p by collecting all predicates

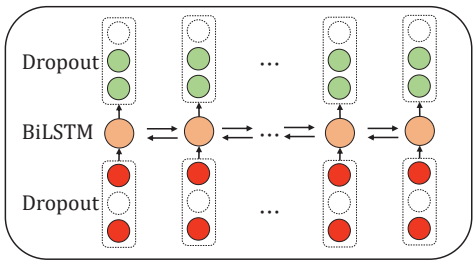

Figure 3: An illustration of Bayesian BiLSTM, where white circles represent masked units. Two dropout layers are introduced between the input and BiLSTM as well as BiLSTM and output. The dropout units are activated during both training and testing processes.

occurred in the knowledge base which have the entity in the candidate entity set as the subject.

- **Set of Entities with Question Context**: In this work, each entity is represented by its context in the question. We construct the entity context set $C_{\tilde{s}}$ according to question q and Candidate entity set C_s. For each entity s_i in C_s, its context in question q is obtained by replacing it with a special token $\langle flag \rangle$ in the question. For example, for the question, "*what film was shawn holly cookson the costume designer of*", the context for the entity '*shawn holly cookson*' is "*what film was* $\langle flag \rangle$ *the costume designer of*".

3.2.2. Entity and Predicate Representation

We first give a brief introduction of BNN, its optimization procedure and the inference of prediction distribution, before presenting the details of representing entities and predicates.

Given a set of N training instances $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, ..., \mathbf{x}_N\}$, each of which is associated with a class label $\mathbf{Y} = \{\mathbf{y}_1, \mathbf{y}_2, ..., \mathbf{y}_N\}$, in BNN, the network is converted into probabilistic form by placing a prior $p(\mathbf{W})$ over network weights $\mathbf{W}$. Given the dataset $\mathbf{X}, \mathbf{Y}$, the posterior distribution is inferred by Bayes' rule:

$$p(\mathbf{W}|\mathbf{X}, \mathbf{Y}) = \frac{p(\mathbf{Y}|\mathbf{X}, \mathbf{W})p(\mathbf{W})}{p(\mathbf{Y}|\mathbf{X})} \tag{2}$$

With the posterior $p(\mathbf{W}|\mathbf{X},\mathbf{Y})$ and new input data $\mathbf{x}^*$, we do prediction by marginalizing over the posterior:

$$p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y}) = \int p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{W})p(\mathbf{W}|\mathbf{X},\mathbf{Y})d\mathbf{W} \tag{3}$$

$$E(\mathbf{y}^*) = \int \mathbf{y}^* p(\mathbf{y}^*|\mathbf{x}^*,\mathbf{X},\mathbf{Y})d\mathbf{y}^* \tag{4}$$

As stated in related work, there are mainly two lines of research on estimating uncertainties in deep learning, BNN-based approaches and non-BNN approaches. For non-BNN methods, they mainly depend on learning an ensemble of neural network or learning a distribution over label space [24, 26]. The former one is timing-consuming to while the latter one replies on the ground-truth distribution over the label space. Therefore in this paper, the BNN-based methods is employed. Within BNN-based methods, there are two paradigms, the Variational Inference-based and the MC dropout-based. The variational-inference based methods require to modify the architecture of neural network and are hard to extend while MC dropout based approaches only requires dropout units in the network [18]. In this work, we use MC dropout [19] to implement a Bayesian BiLSTM. It has been proved that one stochastic forward propagation with dropout units activated in the test time is equivalent to predict with network weights sampled from approximation distribution $q_\theta(\mathbf{W})$ [19]. So the predictive distribution can be approximated by performing M times multiple stochastic forward propagations.

$$\{\mathbf{y_i^*} = \text{BiLSTM}(\mathbf{x}^*|\widehat{\mathbf{W}}_i)\}_{i=1}^M, \widehat{\mathbf{W}}_i \sim q_\theta(\mathbf{W}) \tag{5}$$

In our cases, sampling from $q_\theta(\mathbf{W})$ is equivalent to stochastic forward propagation with dropout units activated. The final prediction of the Bayesian BiLSTM can be computed by performing Monte Carlo integration:

$$E(\mathbf{y}^*) \approx \frac{1}{M}\sum_{i=1}^{M} \text{BiLSTM}(\mathbf{x}^*|\widehat{\mathbf{W}}_i) \tag{6}$$

The Bayesian BiLSTM network used for the encoding of entities and predicates is shown in Figure 3. Dropout layers are introduced between the

input and BiLSTM as well as BiLSTM and output. These two layers work independently and their masks are the same across time steps. The dropout units are activated both during training and testing.

Entity Representation: Previous works often represent entities by their names as well as the labels in the knowledge base. In this work, entities are represented by their contexts in the questions. Suppose question q is expressed as $(w_1, w_2, \ldots, w_n)$, where w_i denotes the ith word in q. For each entity s_i in the candidate set C_s, we extract its context in question q by replacing it with a special token $\langle flag \rangle$ in the question. Here those continues tokens (n-grams) in question which have the shortest Levenshtein distance with an entity are selected to mask. The special token $\langle flag \rangle$ essentially captures the position of the candidate entity in the question. Then, the word embeddings of each constituent word in the context of entities are fed into a Bayesian BiLSTM network. Two hidden state sequences are obtained, one from the forward direction and the other from the backward direction. The final hidden states of the networks in both directions are concatenated which are used as the final representation of entity $\mathbf{h}^s$, with the question context information captured. The distribution on $\mathbf{h}^s$ is obtained by multiple stochastic forward propagations.

$$\{\mathbf{h}_i^s = \text{BiLSTM}((w_1, < flag >, \ldots, w_n)|\widehat{\mathbf{W}}_i)\}_{i=1}^M \tag{7}$$

Predicate Representation: The predicate is represented in two different granularity levels including the word-level and the phrase-level. Let $p_i = \{p_1^{word}, \ldots, p_{N_1}^{word}, p_1^{phrase}, \ldots, p_{N_2}^{phrase}\}$, where the first N_1 tokens are words (e.g., *executive*, *produced*, *by*), and the second N_2 tokens are phrases (e.g., *film*, *executive_produced_by*). The aggregated predicate sequence is randomly initialized and then fed to a Bayesian BiLSTM to derive the representation of predicate p, denoted as $\mathbf{h}^p$. The distribution on $\mathbf{h}^p$ is obtained in a similar way with entity in the previous subsection.

$$\{\mathbf{h}_i^p = \text{BiLSTM}(p_i^{word}, p_i^{phrase})|\widehat{\mathbf{W}}_i)\}_{i=1}^M \tag{8}$$

3.2.3. Score Matrix Calculation

Instead of computing question similarity scores with candidate entities and predicates respectively, we directly match candidate entities with its linked predicates in KB under the context of a given question. Particularly, we generate the score matrix $\mathbf{T}_{mn}$ (m, n represents the number of candidate predicates and entities respectively) with each of its entry taking the value as the dot product between the representations of entity $\mathbf{h}^s$ and predicate $\mathbf{h}^p$.

$$\{\mathbf{T}^i_{mn} = \mathbf{h}^p_i \ \mathbf{h}^s_i | i = 1, 2 \ldots, M\} \tag{9}$$

During training, we perform max-pooling on the score matrix from the row and the column directions respectively and get the score vectors of entity and predicate, v_s and v_p.

$$v^i_p = \text{softmax}(\text{Max-Pooling}_{\text{row-wise}}(\mathbf{T}^i_{mn})) \tag{10}$$

$$v_p = \frac{1}{M} \sum_{i=1}^{M} v^i_p \tag{11}$$

The calculation of v_s is identical to v_p except column-wise max-pooling is performed instead. Here, v_p stores the scores of entity context with different predicates and v_s stores the scores of different entities which measures the probability of each candidate entity linked as the entity mention in a question.

On one hand, we expect the correct entity has the highest score in v_s that implies it appeared in more question-like context (thus allowing the exclusion of wrong candidate entities). On the other hand, the correct predicate is expected to get the highest score in v_p which can help exclude wrong entities with the same lexical form and wrong predicates. Jointly modelling entity linking and predicate prediction can bring mutual benefit to each other. During the testing, entity and predicate $\langle \hat{s}, \hat{p} \rangle$ with the highest value in the score vectors v_s and v_p will be returned. With the entity and predicate jointly selected, the uncertainties of them become comparable because they are calculated simultaneously. If the entity and predicate are predicted separately, their uncertainty values become

incomparable since they are not aligned, which makes it difficult to calculate the overall uncertainty of predictions.

3.2.4. Loss Function and Training Process

The training objective is to maximize the associated score of both the true subject and the true predicate in the entity score vector v_s, and the predicate score vector v_p, respectively. We choose the log-softmax loss function which is formulated as follows:

$$Loss = -(\sum \log \frac{\exp v_s^+}{\sum \exp v_s} + \sum \log \frac{\exp v_p^+}{\sum \exp v_p}) \tag{12}$$

where v_s^+ and v_p^+ are the ground-truth scores in score vectors v_s and v_p respectively. The whole training process of proposed model is shown in Algorithm 1.

Algorithm 1 Training process of proposed model

Require: question set Q, set of candidate entities with question context C_s, set of candidate predicates C_p, number of epochs T.

for t in $1:T$ **do**
  for q in Q **do**
    Generate contextual representation $\mathbf{h}_i^s$ for $\forall S_i \in C_s$;
    Generate representation $\mathbf{h}_j^p$ for $\forall p_j \in C_p$;
    Calculate score matrix $\mathbf{T}_{mn}$ by $\mathbf{h}_i^s \cdot \mathbf{h}_j^p$;
    Generate entity prediction $\hat{s}$ and predicate prediction $\hat{p}$;
    Update network parameters $\mathbf{W}$ with loss function (12).
  end for
end for

3.3. Measuring Uncertainties

In this section, we study how to quantify uncertainties in a KBQA model with predictive distributions. There are mainly two types of uncertainties, model uncertainty and data uncertainty [33]. In particular, model uncertainty

arises from the randomness of BNN model parameters and measures how well the model describes the training data. Model uncertainty is reducible, which can be explained away by supplying more training data. Data uncertainty arises from the inherent data noise, such as measuring error, ambiguity expression or class overlap. For example, the following two predicates, "*location/location/containedby*" and "*location/location/primarily_containedby*", are semantically equal in some situations. It is hard for not only machines but also humen to distinguish between them. Data uncertainty will make the data hard to understand and analyze both for humans and models. Usually, data uncertainty is irreducible unless the noise is removed. Next, we introduce how to quantify these two types of uncertainties in the KBQA models.

3.3.1. Model Uncertainty

In the KBQA data generation process, model uncertainty comes from the randomness of posterior distribution $p(\mathbf{W}|\mathbf{X}, \mathbf{Y})$. Recall that in the KBQA problem, the model must correctly predict both the entity and the predicate to obtain the final right answer. So uncertainty in the prediction of either entity or predicate may cause the model to return wrong answers. We obtain the score vectors of entity and predicate by doing max-pooling on $\mathbf{T}_{mn}$ from two dimensions respectively. For example, to measure the model uncertainty on the prediction of predicates, we can use the average Euclidean distance, where $\overline{v}_p$ is the center of the predictive ensemble:

$$U_m^P = \frac{1}{M} \sum_{i=1}^{M} Euc\_dist(v_p^i, \overline{v}_p) \tag{13}$$

The estimation of model uncertainty on the prediction of entity can be done in a similar way.

$$U_m^S = \frac{1}{M} \sum_{i=1}^{M} Euc\_dist(v_s^i, \overline{v}_s) \tag{14}$$

where $\overline{v}_s$ is the center of the predictive ensemble.

The overall model uncertainty is then defined as the maximum among these

two values:

$$U_m = max(U_m^P, U_m^S) \tag{15}$$

3.3.2. Data Uncertainty

Data uncertainty measures the noises inherent in the data. In the previous study [33], data uncertainty is also called aleatoric uncertainty, which is usually measured by the average variance or average entropy of the predictive score vector ensembles. However, we argue that such a measurement method may fail in some cases for the KBQA system. For example, suppose there are two predictive score vectors for a question involving the classification on four categories, $u_1 = (0.49, 0.49, 0.01, 0.01)$ and $u_2 = (0.4, 0.3, 0.2, 0.1)$. The entropy values of u_1 and u_2 are 0.791 and 1.280 respectively. So according to the previous measurement method, u_2 has a larger data uncertainty. However, the KBQA system would make a rather confident prediction based on u_2, but would be confused by u_1 and could do no better than random guess between the first two categories. To overcome this drawback, we define the data uncertainty of entity and predicate U_d to be the entropy of the top two prediction categories. For the data uncertainty on predicate, we use:

$$U_d^P = \frac{1}{M}\sum_{i=1}^{M} Entropy(\frac{c_p^1}{c_p^1 + c_p^2}, \frac{c_p^2}{c_p^1 + c_p^2}) \tag{16}$$

where c_p^1 and c_p^2 are the scores of the top two classes respectively in u^p. Meanwhile, the data uncertainty for entity can be calculated in a similar way as follows,

$$U_d^S = \frac{1}{M}\sum_{i=1}^{M} Entropy(\frac{c_s^1}{c_s^1 + c_s^2}, \frac{c_s^2}{c_s^1 + c_s^2}) \tag{17}$$

where c_s^1 and c_s^2 are the scores of the top two classes respectively in u^s.

The total data uncertainty is calculated by taking the maximum of these two values:

$$U_d = max(U_d^P, U_d^S) \tag{18}$$

4. Experiments

In this section, we first evaluate the performance of the proposed model on the SimpleQuestions dataset compared with several state-of-the-art methods.

Then we investigate the effectiveness of the estimated uncertainties in the KBQA systems. In particular, we evaluate the performance of these uncertainty measures on tasks of misclassification detection and the detection of causes of errors.

4.1. Performance Comparison

4.1.1. Datasets and Evaluation Metrics

We conduct experiments on the SimpleQuestions dataset [2] in which the question can be solved by a single fact in the knowledge base. The knowledge base we used is the subset of FreeBase[2] with 2M entities (FB2M). All datasets are available online. The detailed statistics are given in Table 1.

	SimpleQuestions	FB2M
# of Training	75,910	-
# of Validation	10,845	-
# of Test	21,687	-
Total Entities	131,681	1,963,130
Total Predicates	1,837	6,701

Table 1: Statistics of SimpleQuestions dataset and FB2M.

SimpleQuestions: The dataset contains over 100,000 questions written in natural language by human English-speaking annotators. Each question has a corresponding fact from FB2M to answer and explain it. The dataset is randomly divided into three parts, with 70% as the training set, 10% as the validation set and the remaining 20% as the test set.

FB2M: Freebase is often used as a reliable knowledge base because its data is collected and filtered mainly by its community members. In this paper, we use a large subset of Freebase, FB2M, as the backend knowledge base. As

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

the Freebase API was no longer available since 2016, we use an entity name collection [7] to map the entity IDs to their names.

In this work, We use accuracy as the evaluation metric. Specifically, the prediction is regarded as correct only when the model successfully predicts the subject entity and predicate at the same time, which can be formulated as follows:

$$Accuracy = \frac{\sum_{i=1}^{N} \mathbf{1}_{[(\hat{s_i},\hat{p_i})=(s_i,p_i)]}}{N} \tag{19}$$

where $\hat{s_i}$ and $\hat{p_i}$ are predicted subject entity and predicted predicate respectively, s_i and p_i are ground-truth subject entity and predicate. $\mathbf{1}_{[.]}$ is the indicator function.

4.1.2. Experimental Settings

To compare the proposed model with other methods, we use the same training, validation and test split that was offered in the SimpleQuestions dataset. The dimension of word embeddings is set to 128. The word embeddings are pre-trained on the training data set using GloVe [37]. The number of hidden units in Bayesian BiLSTM is set to 128. The network weights are initialized with Xavier initialization [38]. For training, mini-batch stochastic gradient descent optimizer is used to minimize the loss function. A learning rate of 0.1 is used during training. The dropout rate is set to 0.8, and dropout units are activated both during training and testing to implement the MC dropout.

4.1.3. Baselines

To demonstrate the effectiveness of the proposed model, we include the following state-of-the-art KBQA methods as the baselines:

- Bordes et al.[2]: Questions, entities and predicates are embedded into the same space with a memory network, and new questions and facts are matched with their embeddings.
- Yin et al.[14]: A character-level convolutional neural network (CNN) is trained to match the entity and a word-level CNN with attentive max-pooling is trained to match the predicate.

Approaches	**Test Accuracy%**
Bordes et al. [2]	62.7
Yin et al. [14]	68.3
Golub and He [8]	70.9
Lukovnikov et al. [6]	71.2
Mohammed et al. [10]	73.1
The proposed approach (without BNN)	74.9
The proposed approach	**75.1**

Table 2: Comparison with state-of-the-art methods on SimpleQuestions dataset in end-to-end settings.

- Golub and He[8]: A character-level, attention-based encoder-decoder model is developed where embeddings of questions, entities, and predicates are all jointly learned to directly optimize the likelihood of generating the correct KB query.

- Lukovnikov et al.[6]: Questions and predicates/entities are embedded into the same space to match their similarities with a character-level gated recurrent units neural network.

- Mohammed et al.[10]: Entity and predicate predictions are treated as classification problems and different combinations of neural networks are built as classifiers.

4.1.4. Results

As shown in Table 2, the proposed model achieves comparable results compared to other state-of-the-art methods. Compared to the full model, the proposed model without BNN (dropout units closed during inference) achieve a little bit lower score, demonstrating that the ensemble nature of BNN-based method could make the result robuster .As mentioned in [6, 10], some methods [14, 7] claim better performance, but they used extra information sources based on the Freebase API, which is no longer available. As such,

we report their results in Table 2 without using extra training data. A few multi-staged methods reported better results. However, either a separate segmentation model is employed to split the question into patterns and mentions using extra data [15, 23] or their performance are hard to replicate [39] which has also been pointed out in [10, 23]. Moreover, multi-staged approaches such as [40] consist of multiple related components such as entity detection, entity linking, relation prediction and evidence integration. It is difficult to estimate the uncertainties of such multi-staged approaches for KBQA since (1) uncertainties estimated in each component are hard to aligned and combined; (2) the uncertainties propagation problem between different components make it hard to estimate the uncertainties of the whole framework.

On the contrast, instead of adopting multi-staged methods to improve performance, the proposed approach focuses more on incorporating uncertainties estimation mechanism into KBQA which requires an end-to-end setting to avoid uncertainties flowing.

4.2. Uncertainties in KBQA systems

Task 1: Misclassification Detection

Intuitively, when the model is uncertain about its prediction, it should be more likely to make mistakes. As such, well-estimated uncertainties should be able to distinguish between the correct predictions and wrong ones of the model. In this task, we use several uncertainty measures and their combinations to detect whether a prediction is correct or not.

Concretely, we train a logistic regression model using two types of uncertainty measures (U_m, U_d) and their combination as features respectively (corresponding to each row of Table 3 and 4). The labels are automatically generated by comparing the prediction with ground-truth. The model is trained in the development set and tested on 21,687 instances in the test set. The performance is measured by Area Under the Receiver Operating Characteristic curve (AUC) and Area Under Precision-Recall curve (AUPR).

To show the efficiency of the defined two uncertainty measures U_d and U_m,

we construct three baselines. The baselines are constructed by training the logistic regression models using the following measures as features individually. We compare the results with the following three baselines:

Max Probability (MaxP): the maximum value in the softmax vector is employed as feature.

$$MaxP = max_c P(S_c|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \tag{20}$$

Model uncertainty (MU): the mutual information (MI) [26] between the class label y and the parameters of model, $\mathbf{W}$ is used as feature.

$$MU = MI(\mathbf{y}, \mathbf{W}|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \tag{21}$$

Data uncertainty (DU): the expectation of entropy in the prediction scores [26] is employed as the feature. In practice, we use the average to approximate the expectation.

$$DU = E_{(P(\mathbf{W}|\mathbf{X},\mathbf{Y}))} Entropy(\mathbf{y}|\mathbf{x}^*, \mathbf{X}, \mathbf{Y}) \tag{22}$$

Feature	**AUC**	**AUPR**
MaxP	0.867	0.922
MU [26]	0.749	0.886
DU [26]	0.859	0.922
$[MU, DU]$	0.892	0.953
U_m(Our Proposed)	0.783	0.897
U_d(Our Proposed)	**0.904**	**0.960**
$[U_m, U_d]$	0.900	0.959

Table 3: Evaluation results for misclassification detection with various uncertainty measures.

The results are shown in Table 3. It can be observed that using the proposed data uncertainty measure U_d achieves the best results both in AUC and AUPR compared with all the baselines and also using the proposed model uncertainty measure U_m. Combining the proposed model uncertainty

U_m and data uncertainty measure U_d does not bring any benefit. It shows that data uncertainty plays a more important role than model uncertainty for misclassification detection on this particular dataset.

Task 2: Cause of Error Detection

Sometimes the model makes mistakes because the question itself is ambiguous, corresponding to a high data uncertainty. In other situations, the model makes mistakes because the model has not encountered such a type of questions or predicates during the training process, corresponding to a high model uncertainty. The ability to distinguish between these two types of uncertainties is important since different actions could be taken depending on the source of uncertainty.

We assume that the common errors in this task can be divide into three categories as follows:

Missing Candidate: The correct resource is missing in the candidate set.

Indistinguishability: The predicted predicate is indistinguishable from the correct predicate. For example, the predicates *music/release/track* and *music/release/track_list* are indistinguishable and both of them have been the ground-truth for the same type of questions in the training and test sets.

Unseen predicate: The correct predicate has never occurred in the training set. Moreover, most words in it rarely appeared in the training process. As the predicates are not learned well, high uncertainties might be caused.

We manually label some wrong predictions with these three types of errors and randomly draw 60, 70, 70 examples from each category. All types of errors in the dataset are mutually exclusive. In this experiment, we assume that the first two types of errors will cause high data uncertainty and the third type of errors will cause high model uncertainty. Therefore, there are two types of labels, "data" and "model". The logistic classifier with the same features in the previous task is employed to classify whether the error is caused by model uncertainty or data uncertainty. We also use AUC and AUPR as the evaluation metrics.

Feature	AUC	AUPR
MaxP	0.731	0.762
MU [26]	0.490	0.595
DU [26]	0.795	0.641
$[MU, DU]$	0.828	0.868
U_m(Our Proposed)	0.580	0.670
U_d(Our Proposed)	0.771	0.817
$[U_m, U_d]$	**0.842**	**0.881**

Table 4: Evaluation results for cause of error detection using different uncertainty measures.

The results are shown in Table 4. It can be observed that using the proposed data uncertainty U_d consistently achieves better performance compared to using the model uncertainty U_m. The classifier using the combination of the proposed model and data uncertainty measures achieves the best results on AUC and AUPR compared to all the other methods.

4.3. Result Analysis

In the previous section, the quantitative experimental results show that the estimated uncertainties could be employed to measure model confidence and detect the causes of errors. In this section, some examples are presented to further explain what model uncertainty and data uncertainty capture.

Four types of prediction results with examples are shown in Table 5. The first example is the prediction with low model uncertainty and low data uncertainty, indicating that the model is familiar with the question and the question itself is expressed clearly. The predicted predicate is very likely to be correct. The second example is the prediction with high model uncertainty since the predicate does not exist in the training set. The third example is the prediction with high data uncertainty since both of the top two predicted predicates have been the ground truth of the same type of questions in the training set and they are equivalent in semantics given the question. So it is difficult for both the model

Low U_d **Low** U_m	q: what country is lake ka-ho in?
	pred: *location/location/containedby*
	gold: location/location/containedby
Low U_d **High** U_m	q: who wrote soviet union?
	pred: *book/written_work/author*
	gold: olympics/olympic_event_competition/medalists
High U_d **Low** U_m	q: who was the publisher of metal marines?
	pred: *cvg/game_version/publisher*
	gold: cvg/computer_videogame/publisher
High U_d **High** U_m	q: What US state is glendale found in?
	pred: *location/location/containedby*
	gold: location/location/primarily_containedby

Table 5: Examples of questions and predicated predicates with top 1% highest/lowest model uncertainty/data uncertainty, where q represents questions, *pred* represents predicated relation for q and *gold* represents ground-truth relation for q.

and human to distinguish between these two predicates. The last example has high value both in model and data uncertainty because the predicted predicate is hard to distinguish and this type of questions is rare in the training set.

5. Conclusion

We have proposed a novel Bayesian simple KBQA model in which uncertainties can be estimated. Specifically, model uncertainty and data uncertainty are estimated with a BNN implemented by the Monte Carlo dropout and a novel way of calculating uncertainties in KBQA systems is proposed. Furthermore, a novel end-to-end framework is proposed to jointly select entity and predicate in one single training procedure, avoiding the uncertainty propagation problem. Experimental results show that our model outperforms some state-of-the-art KBQA methods and our proposed uncertainty measures could be employed to detect the misclassification and the causes of errors in KBQA systems.

Acknowledgements

This work was funded by the National Key Research and Development Program of China (2016YFC1306704), the National Natural Science Foundation of China (61772132) and Innovate UK (grant no. 103652).

References

[1] K. Bollacker, C. Evans, P. Paritosh, T. Sturge, J. Taylor, Freebase: A collaboratively created graph database for structuring human knowledge, in: Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data, SIGMOD 2008, New York, NY, USA, 2008, pp. 1247–1250. doi:10.1145/1376616.1376746.

[2] A. Bordes, N. Usunier, S. Chopra, J. Weston, Large-scale simple question answering with memory networks, ArXiv abs/1506.02075.

[3] W.-t. Yih, M.-W. Chang, X. He, J. Gao, Semantic parsing via staged query graph generation: Question answering with knowledge base, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, ACL 2015, Beijing, China, 2015, pp. 1321–1331. doi:10.3115/v1/P15-1128.

[4] H. Bast, E. Haussmann, More accurate question answering on freebase, in: Proceedings of the 24th ACM International Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, 2015, pp. 1431–1440.

[5] S. Shin, X. Jin, J. Jung, K.-H. Lee, Predicate constraints based question answering over knowledge graph, Information Processing & Management 56 (3) (2019) 445 – 462. doi:https://doi.org/10.1016/j.ipm.2018.12.003.

[6] D. Lukovnikov, A. Fischer, J. Lehmann, S. Auer, Neural network-based question answering over knowledge graphs on word and character level, in: Proceedings of the 26th International Conference on World Wide Web, WWW 2017, Republic and Canton of Geneva, Switzerland, 2017, pp. 1211–1220. doi:10.1145/3038912.3052675.

[7] Z. Dai, L. Li, W. Xu, CFO: Conditional focused neural question answering with large-scale knowledge bases, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, 2016, pp. 800–810. doi:10.18653/v1/P16-1076.

[8] D. Golub, X. He, Character-level question answering with attention, in: Proceedings of the 2016 conference on empirical methods in natural language processing, EMNLP 2016, Austin, Texas, USA, 2016, pp. 1598–1607.

[9] Z. Hao, B. Wu, W. Wen, R. Cai, A subgraph-representation-based method for answering complex questions over knowledge bases, Neural Networks 119 (2019) 57 – 65. doi:https://doi.org/10.1016/j.neunet.2019.07.014.

[10] S. Mohammed, P. Shi, J. Lin, Strong baselines for simple question answering over knowledge graphs with and without neural networks, in: Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, ACL 2018, New Orleans, Louisiana, 2018, pp. 291–296. doi:10.18653/v1/N18-2047.

[11] G. Zhao, J. Wu, D. Wang, T. Li, Entity disambiguation to wikipedia using collective ranking, Information Processing & Management 52 (6) (2016) 1247 – 1257. doi:https://doi.org/10.1016/j.ipm.2016.06.002.

[12] A. Pappu, R. Blanco, Y. Mehdad, A. Stent, K. Thadani, Lightweight multilingual entity extraction and linking, in: Proceedings of the Tenth

ACM International Conference on Web Search and Data Mining, WSDM 2017, New York, NY, USA, 2017.

[13] M. Yu, W. Yin, K. S. Hasan, C. dos Santos, B. Xiang, B. Zhou, Improved neural relation detection for knowledge base question answering, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, 2017, pp. 571–581. doi:10.18653/v1/P17-1053.

[14] W. Yin, M. Yu, B. Xiang, B. Zhou, H. Schütze, Simple question answering by attentive convolutional neural network, in: Proceedings of the 26th International Conference on Computational Linguistics: Technical Papers, COLING 2016, Osaka, Japan, 2016, pp. 1746–1756.

[15] Y. Hao, H. Liu, S. He, K. Liu, J. Zhao, Pattern-revising enhanced simple question answering over knowledge bases, in: Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, 2018, pp. 3272–3282.

[16] M. Dubey, S. Dasgupta, A. Sharma, K. Höffner, J. Lehmann, Asknow: A framework for natural language query formalization in sparql, in: European Semantic Web Conference, ESWC 2016, Heraklion, Crete, Greece, 2016, pp. 300–316.

[17] R. M. Neal, Bayesian learning for neural networks, Vol. 118, Springer Science & Business Media, 2012.

[18] C. Blundell, J. Cornebise, K. Kavukcuoglu, D. Wierstra, Weight uncertainty in neural networks, in: Proceedings of the 32Nd International Conference on International Conference on Machine Learning, ICML 2015, Lille, France, 2015, pp. 1613–1622.

[19] Y. Gal, Z. Ghahramani, Dropout as a bayesian approximation: Representing model uncertainty in deep learning, in: Proceedings of the

33rd International Conference on International Conference on Machine Learning, ICML 2016, New York, NY, USA, 2016, pp. 1050–1059.

[20] L. Dong, F. Wei, M. Zhou, K. Xu, Question answering over freebase with multi-column convolutional neural networks, in: Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, Beijing, China, 2015, pp. 260–269.

[21] K. Xu, S. Reddy, Y. Feng, S. Huang, D. Zhao, Question answering on freebase via relation extraction and textual evidence, in: Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Berlin, Germany, 2016.

[22] R. Das, M. Zaheer, S. Reddy, A. McCallum, Question answering on knowledge bases and text using universal schema and memory networks, in: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, 2017, pp. 358–365. doi:10.18653/v1/P17-2057.

[23] X. Huang, J. Zhang, D. Li, P. Li, Knowledge graph embedding based question answering, in: Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining, WSDM 2019, New York, NY, USA, 2019, pp. 105–113. doi:10.1145/3289600.3290956.

[24] B. Lakshminarayanan, A. Pritzel, C. Blundell, Simple and scalable predictive uncertainty estimation using deep ensembles, in: Proceedings of the 31st International Conference on Neural Information Processing Systems, NIPS 2017, 2016, pp. 6405–6416.

[25] K. Lee, H. Lee, K. Lee, J. Shin, Training confidence-calibrated classifiers for detecting out-of-distribution samples, in: International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, 2018.

[26] A. Malinin, M. Gales, Predictive uncertainty estimation via prior networks, in: Proceedings of the 32Nd International Conference on Neural Information Processing Systems, NIPS 2018, USA, 2018, pp. 7047–7058.

[27] D. J. C. MacKay, A practical bayesian framework for backpropagation networks, Neural Comput. 4 (3) (1992) 448–472. doi:10.1162/neco.1992.4.3.448.

[28] A. Graves, Practical variational inference for neural networks, in: Proceedings of the 24th International Conference on Neural Information Processing Systems, NIPS 2011, USA, 2011, pp. 2348–2356.

[29] D. P. Kingma, T. Salimans, M. Welling, Variational dropout and the local reparameterization trick, in: Proceedings of the 28th International Conference on Neural Information Processing Systems, NIPS 2015, Cambridge, MA, USA, 2015, pp. 2575–2583.

[30] C. Louizos, M. Welling, Structured and efficient variational deep learning with matrix gaussian posteriors, in: Proceedings of the 33rd International Conference on International Conference on Machine Learning, ICML 2016, New York, NY, USA, 2016, pp. 1708–1716.

[31] T. Chen, E. B. Fox, C. Guestrin, Stochastic gradient hamiltonian monte carlo, in: Proceedings of the 31st International Conference on International Conference on Machine Learning, ICML 2014, Beijing, China, 2014, pp. II–1683–II–1691.

[32] A. Korattikara, V. Rathod, K. Murphy, M. Welling, Bayesian dark knowledge, in: Proceedings of the 28th International Conference on Neural Information Processing Systems, NIPS 2015, Cambridge, MA, USA, 2015, pp. 3438–3446.

[33] A. Kendall, Y. Gal, What uncertainties do we need in bayesian deep learning for computer vision?, in: Proceedings of the 31st International

Conference on Neural Information Processing Systems, NIPS 2017, USA, 2017, pp. 5580–5590.

[34] A. Kendall, Y. Gal, R. Cipolla, Multi-task learning using uncertainty to weigh losses for scene geometry and semantics, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2018, Salt Lake City, Utah, 2018, pp. 7482–7491.

[35] X. Zhang, F. Chen, C.-T. Lu, N. Ramakrishnan, Mitigating uncertainty in document classification, in: Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, ACL 2019, Minneapolis, Minnesota, 2019, pp. 3126–3136. doi:10.18653/v1/N19-1316.

[36] Y. Xiao, W. Y. Wang, Quantifying uncertainties in natural language processing tasks, in: Proceedings of the AAAI Conference on Artificial Intelligence, AAAI 2019, Honolulu, Hawaii, USA, 2019, pp. 7322–7329.

[37] J. Pennington, R. Socher, C. Manning, Glove: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, Doha, Qatar, 2014, pp. 1532–1543. doi:10.3115/v1/D14-1162.

[38] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, AISTATS 2010, Chia Laguna Resort, Sardinia, Italy, 2010, pp. 249–256.

[39] F. Ture, O. Jojic, No need to pay attention: Simple recurrent neural networks work!, in: Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, emnlp 2017, Copenhagen, Denmark, 2017, pp. 2866–2872. doi:10.18653/v1/D17-1307.

[40] M. Petrochuk, L. Zettlemoyer, SimpleQuestions nearly solved: A new upperbound and baseline approach, in: Proceedings of the 2018 Conference

on Empirical Methods in Natural Language Processing, Brussels, Belgium, 2018, pp. 554–558. doi:10.18653/v1/D18-1051.