Evaluating Architectural Choices for Deep Learning Approaches for Question Answering over Knowledge Bases

Sherzod Hakimov
Semantic Computing Group
CITEC, Bielefeld University

Soufian Jebbara
Semantic Computing Group
CITEC, Bielefeld University

Philipp Cimiano
Semantic Computing Group
CITEC, Bielefeld University

***Abstract*—The task of answering natural language questions over knowledge bases has received wide attention in recent years. Various deep learning architectures have been proposed for this task. However, architectural design choices are typically not systematically compared nor evaluated under the same conditions. In this paper, we contribute to a better understanding of the impact of architectural design choices by evaluating four different architectures under the same conditions. We address the task of answering simple questions, consisting in predicting the subject and predicate of a triple given a question. In order to provide a fair comparison of different architectures, we evaluate them under the same strategy for inferring the subject, and compare different architectures for inferring the predicate. The architecture for inferring the subject is based on a standard LSTM model trained to recognize the span of the subject in the question and on a linking component that links the subject span to an entity in the knowledge base. The architectures for predicate inference are based on i) a standard softmax classifier ranging over all predicates as output, ii) a model that predicts a low-dimensional encoding of the property and subject entity, iii) a model that learns to score a pair of subject and predicate given the question as well as iv) a model based on the well-known FastText model. The comparison of architectures shows that FastText provides better results than other architectures.**

I. Introduction

The task of Question Answering (QA) has received increasing attention in the last few years. Most research has concentrated on the task of answering factoid questions such as *Who wrote Mildred Pierced?*, yielding the answer *Stuart Kaminsky*. Typically, such answers are extracted from a knowledge base (KB). A frequently used dataset in this context is the SimpleQuestions [2] dataset, which consists of *simple* questions that can be answered with a single fact from the Freebase KB. For instance, the question above can be answered using the following triple from Freebase:

```
Subject:   m.04t1ftb (mildred_pierced)
Predicate: book.written_work.author
Object:    m.03nx4yz (stuart_kaminsky)
```

The system needs to identify the relevant entity (subject), i.e. *mildred_pierced* in the example question, and infer the appropriate predicate, i.e. *book.written_work.author*. In the case of the SimpleQuestions dataset, all questions involve a single triple, with the answer being the corresponding object. Thus, the task involves essentially predicting the subject and predicate of a triple.

Various deep learning architectures have been proposed for this task. However, a systematic comparison of different architectural choices has not been provided so far. In particular, different architectures for the prediction of the property have been proposed. Our goal is to compare and evaluate these different architectural choices with respect to the same model for subject entity prediction, which is based on a sequence-to-sequence NER architecture.

The main contributions in this paper are:

- We evaluate the performance of the individual models we provide on different levels, i.e. subject entity span recognition, entity linking, predicate prediction and answer selection.
- We compare four different architectures for predicate prediction under the same conditions (same subject entity recognition, same entity linking and same index)
- We show the impact of entity linking on the overall performance on the question answering task.

The paper is structured as follows: in the next Section II we describe the NER-based system for predicting the subject entity as well as the four architectures for predicate prediction. Section III presents the results of our evaluation for all components. Before concluding, we discuss the related work.

II. Methods

The task of answering simple questions requires identifying the correct subject entity and the predicate from a given natural language question. In this section, we describe in detail the model for identifying the span of the entity and retrieving candidates. Then, we describe four architectures for predicate prediction that build on this common entity prediction model. All four architectures rely on a candidate retrieval step that extracts candidate pairs of subject and predicate and then score pairs of subject/predicate to predict a query consisting of a single subject and predicate.

A. Named Entity Recognition & Candidate Pair Generation

Inverted Index: We build an inverted index for all entities in Freebase using the *type.object.name* and *common.topic.alias* predicates, storing the frequency with with each string is associated to the given entity or predicate. We rely on *owl:sameAs* links from Freebase to DBpedia entities as provided by the DBpedia release of 2014[1] to extract and add additional labels to the index.

NER: We trained a Named Entity Recognizer (NER) system similar to the one proposed by Chiu and Nichols [4] using weak supervision[2]. We adapt the system to extract a single entity span using an *IO* tagging scheme to mark tokens inside (I) and outside (O) of the single named entity of interest. The architecture is based on Bidirectional LSTMs (BiLSTM) [6] composed of two LSTM [7] layers. The model uses a threesome of features: word and character embeddings along with the case of words (lowercase, uppercase). These features are concatenated and fed into a neural network.

The input sentence is tokenized. Each token in the sentence is converted into a word embedding representation using Glove [11] vectors (100 dimensional). Each token is also represented in terms of characters by converting the token into a matrix where each vector corresponds to a one-hot encoding vector of a character. The character matrix is fed into a Convolutional Neural Network (CNN) [9] layer which applies a convolution function to the input vectors. It is followed by a Max-Pooling layer that extracts the most important character features given the token. By applying the NER model on the example sentence, the tokens *Mildred* and *Pierced* are tagged as *I* while the rest of the tokens is tagged as *O*.

Candidate Pair Generation: We query the index with the single entity mention m identified by the NER component. All matching entries are added to the set $S(m)$. Each entry contains a subject URI (Freebase MID) and a frequency value. For example, the following subjects are found: *m.04t1ftb*, *m.01d13qs*, *m.04t_038*, *m.0cgv06r* by querying the detected entity mention *Mildred Pierced*.

We define a KG as a set of triples of the form (s_i, p_i, o_i) that appear in the Freebase-2M dataset. Given a subject s_i, we define the set $Pred(s_i)$ of all the properties that s_i has as

$$Pred(s_i) := \{p_i \mid \exists o_i (s_i, p_i, o_i) \in KB\}. \tag{1}$$

We further define the set of candidate pairs for mention m as:

$$C(m) := \{(s_i, p_i) \mid s_i \in S(m) \wedge p_i \in Pred(s_i)\}. \tag{2}$$

The next step is to find a ranking function that takes an input question text (q), the identified mention m and candidate pairs ($C(m)$=$\{(s_1, p_1)$, (s_2, p_2), (s_3, p_3), $\ldots$, $(s_n, p_n)\}$), and returns the highest ranking pair (s^*, p^*).

We rely on the following probabilistic formulation to infer a predicate and subject entity:

$$(s^*, p^*) = argmax_{(s_i,p_i)\in C(m)} P(s_i, p_i|q; \theta) \tag{3}$$

[1]http://oldwiki.dbpedia.org/Downloads2014\#links-to-freebase

[2]We build on the code available at https://github.com/kamalkraj/Named-Entity-Recognition-with-Bidirectional-LSTM-CNNs

where $P(s_i, p_i)$ computes the probability of a pair s_i and p_i as follows:

$$P(s_i, p_i|q; \theta) = P(p_i|q; \theta)\cdot P(s_i|q; \theta) \tag{4}$$

where $P(p_i|q; \theta)$ is the probability of predicate p_i as computed by our four predicate models described below. $P(s_i|q : \theta)$ is the probability of a subject s_i computed by normalizing the frequency scores retrieved for the mention m.

In the following sections, we describe our proposed models for the prediction of target predicates.

B. Model 1: BiLSTM-Softmax

The first model we investigate is inspired in the approach presented by Ture et al. [12], which is based on a BiLSTM classifier that predicts the target predicate given the question text. Similar to the NER model, the question text is encoded on the word and character level. Word and character embeddings are concatenated and passed through a BiLSTM layer and fed into a feed-forward layer with softmax activation function, which calculates a probability distribution over a set of predicates. The model ranks all predicates based on the input question q and the extracted subject mention m as in $P(p|q, m)$. Before passing the question text to our network, we replace the entity name with a special placeholder token *e* (e.g. “Who wrote e?”) that abstracts away the (inferred) subject mention.

C. Model 2: BiLSTM-KB

Our second model incorporates pre-trained graph embeddings for entities and predicates into the classification. The graph embeddings we use are computed using FastText [8] in a similar fashion to TransE [3]. We phrase the task of learning KB embeddings as a classification task. For each triple $t = (e_i, p, e_j)$ in the KB, we construct training samples for the FastText classifier by treating the predicate p and the object e_j as input tokens and subject e_i as the target class. To create embedding vectors that are aware of the role of an entity in a triple, we generate the training sample using role-specific embeddings: e_i^s, e_j^o and p^s. Here, e_i^s indicates that the target is an entity in the subject position, e_j^o is an input entity in the object position and p^s an input predicate used for predicting a subject entity. Analogously, we create a training sample with the object being the target class. Thus, we have different embeddings for an entity in the subject role and in the object role.

By training a FastText classifier on the generated training samples, we obtain vector representations for all entities and predicates with respect to their role in the triple[3]. We chose FastText as a classifier for its good performance on text classification tasks.

The model architecture for predicting the target predicate is similar to the first model. BiLSTM layers with word and character embeddings are used to encode the given question text.

[3]Due to the huge amount of target classes, training the classifier with a full softmax objective is not feasible. Instead, we use the negative sampling objective that is part of the FastText toolkit as an approximation to the softmax objective.

The difference is that this model predicts the pre-computed property embedding vector instead of a categorical encoding. We obtain a probability distribution by computing the cosine similarity to all predicates in Freebase-2M normalized across all properties.

D. Model 3: BiLSTM-Binary

This model is different from the other 2 models explained above (see Section II-B and Section II-C) in terms of the input to the model. While BiLSTM-KB introduces external knowledge about predicates from a knowledge base, this model learns to associate the question text with the tokens in the predicate URI. The input is composed of a question text q and the label of a single predicate p_i and the model outputs a binary decision (0 or 1) indicating if the predicate is correct for the question. By giving the label of a predicate as an input feature, the model can potentially use the similarity between the question text (e.g. *Who **wrote** e?*) and the predicate label (e.g. *book.**written**_work.author*) to determine if the given predicate tokens matches the question text.

The inputs q and p_i are tokenized and fed into encoding layers that use word and character embeddings. The encoding is the same process explained in Section II-B where the tokens are represented by word and character embeddings and fed into 2-layer BiLSTM.

E. Model 4: FastText-Softmax

Our last model relies on FastText to train a classifier to predict the predicate given the question text. The FastText tool implements a linear classifier on top of a bag-of-N-gram representation of a text using word N-grams to preserve local word order and character N-grams for robustness against out-of-vocabulary words. The model outputs a probability for each predicate. The score for a candidate pair is computed using Equation 3. The highest scoring pair is selected as the final output. For a detailed description of the model architecture we refer to [8]. Due to the moderate size of the target vocabulary[4], we can train the classifier with a full softmax objective. We trained the classifier for 50 epochs and a hidden layer size of 100. The classifier uses word N-grams of size 1 and 2 and character N-grams of size 5.

III. Evaluation

In this section we present the results of the evaluation of the different components for NER (subject entity recognition), entity linking and predicate prediction. All results are provided on the given test split of the SimpleQuestions [2] dataset.

A. Named Entity Recognition

The evaluation shows the accuracy for detecting the correct mention of the subject entity in the question. We trained a BiLSTM-CRF NER system on the SimpleQuestions training split. The model was trained for 100 epochs, with 100-dimensional Glove word embeddings and 200-dimensional LSTM layers. To judge whether the NER prediction is correct, we query the index for n-grams in the detected subject entity span and regard the prediction as correct if the corresponding subject entity from the triple is contained in the results of the index lookup. The NER component achieves an accuracy of 0.82.

[4]1629 predicates in the training set.

B. Named Entity Linking

Once the subject mention m has been extracted from the NER system, the next step is to select an entity from the knowledge base that is denoted by the subject entity mention m detected by the NER system. We rank the entities retrieved from the index inversely according to the frequency value stored in the index. We evaluate whether the correct entity is contained in the top k entities retrieved. Table I shows the Recall@k score for different values of k. We see that retrieving more than 10 entities does not increase the recall significantly above what we obtain with the top 10 candidates alone.

TABLE I
Named Entity Linking evaluation on test split using Recall@K

K	Recall@k
1	0.68
2	0.74
5	0.79
10	0.81
25	0.82
100	0.82

C. Predicate Prediction

The results for the different predicate predicting models are given in Table II, showing the results for the best hyperparameters chosen on a development set. The table gives the accuracy for predicate prediction as well as for answer retrieval. It can be observed that the best results are obtained by the FastText-Softmax model, followed at equal distance by the BiLSTM-Softmax and BiLSTM-Binary architectures. The BiLSTM-KB approach performs worst and scores more than 10 points below the FastText-Softmax model.

TABLE II
Evaluation of four models on predicate prediction and answer tasks, the reported results are accuracy for the respective tasks.

Name	Predicate Prediction	Answer Prediction
BiLSTM-Softmax	0.74	0.67
BiLSTM-KB	0.68	0.61
BiLSTM-Binary	0.73	0.66
FastText-Softmax	**0.79**	**0.68**

D. Answer Prediction

The task of question answering on the SimpleQuestions dataset requires a system to output a single triple consisting of a subject and a predicate. We evaluated the four proposed models on prediction of a triple consisting of a subject and a predicate. The predicated pairs are ranked using Equation 3.

IV. Related Work

Bordes et al. [2] have presented the first results on the SimpleQuestions dataset. Their approach is based on Memory Networks [13]. The approach corrupts the dataset to generate negative samples by assigning random questions from the dataset to Freebase entity and predicate pairs. Aghaebrahimian et al. [1] proposed a method that predicts the predicate and subject separately. Their approach uses a two-layered CNN for ranking predicates and named entity linking approach that ranks subjects.

Yin et al. [14] proposed an approach that uses Convolutional Neural Networks (CNN) with attentive max pooling along with an entity detection and linking system. They also proposed to use character embeddings in combination with word embeddings since character embeddings generalize better in handling out-of-vocabulary (OOV) words. Similarly, Golub et al. [5] proposed an approach that uses both LSTM and CNN encoders together with character-level embeddings. The authors show that character embeddings generalize better compared to word embeddings (0.78 vs 0.38). Similarly, Lukovnikov et al. [10] proposed another system that encodes subject and predicate using character and word level embeddings to learn a function that optimizes both subject and predicate assignments by introducing negative samples. Our BiLSTM-Binary uses a similar approach by introducing negative samples for predicate prediction. Ture et al. [12] proposed a rather simple model based on RNNs without any attention mechanism. They essentially propose to use a model with 2-BiGRU layers for prediction of predicates and a model with 2-BiLSTM layers to predict the span for the subject. Our BiLSTM-Softmax for predicting the property is inspired by their model. However, we could not even come close to reproduce their results with a simplified version of their architecture.

V. Conclusion

In this paper, we analyzed four different model architectures for predicate prediction on the task of answering simple questions. We have strived for comparing these for architectures under the same conditions, using the same subject entity detection component as well as the same linking component and the same inverted index. We have shown how well the single components perform on sub-tasks and could show that our model can reach around 82% in subject entity span detection and 82% Recall at about 10 retrieved index entries for entity linking. In some cases the NER system selects an incorrect span when the question contains some proper name which is not part of a target span, e.g. "where is mineral hot springs, colorado?" the expected span is "mineral hot springs" while the NER system recognizes the span "springs, colorado". In spite of the errors made by the NER component, the predicate inference tasks achieves accuracies of close to 80%. The best model is the FastText-Softmax, which outperforms the other three models both on predicate prediction and answer prediction. The worst results are achieved by BiLSTM-KB, hinting at the fact that the prediction of low-dimensional property vectors is not effective. This needs further investigation. In general, an avenue for future work might be to design joint inference systems that move away from a pipeline architecture and predict the subject entity and predicate jointly. The BiLSTM-Binary goes in this direction, but shows only comparable results to the BiLSTM-Softmax model.

Unfortunately, we were not able to reproduce the performance of some published approaches on the task, which clearly shows the need of working towards a fair comparison between approaches within one unified framework and environment.

Acknowledgements

This work was supported by the Cluster of Excellence Cognitive Interaction Technology 'CITEC' (EXC 277) at Bielefeld University, which is funded by the German Research Foundation (DFG).

References

[1] A. Aghaebrahimian and F. Jurčíček, "Open-domain factoid question answering via knowledge graph search," in *Proceedings of the Workshop on Human-Computer Question Answering*, 2016, pp. 22–28.

[2] A. Bordes, N. Usunier, S. Chopra, and J. Weston, "Large-scale simple question answering with memory networks," *arXiv preprint arXiv:1506.02075*, 2015.

[3] A. Bordes, N. Usunier, A. Garcia-Duran, J. Weston, and O. Yakhnenko, "Translating embeddings for modeling multi-relational data," in *Proceedings of Advances in Neural Information Processing Systems (NIPS)*, 2013, pp. 2787–2795.

[4] J. P. Chiu and E. Nichols, "Named entity recognition with bidirectional lstm-cnns," *arXiv preprint arXiv:1511.08308*, 2015.

[5] D. Golub and X. He, "Character-level question answering with attention," in *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2016.

[6] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 6645–6649.

[7] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[8] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, "Bag of tricks for efficient text classification," *arXiv preprint arXiv:1607.01759*, 2016.

[9] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[10] D. Lukovnikov, A. Fischer, J. Lehmann, and 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, pp. 1211–1220.

[11] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proc. of the International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543.

[12] F. Ture and O. Jojic, "No need to pay attention: Simple recurrent neural networks work!" in *Proceedings of the International Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2017, pp. 2866–2872.

[13] J. Weston, S. Chopra, and A. Bordes, "Memory networks," in *Proc. of the 6th International Conference on Learning Representations*, 2015.

[14] W. Yin, M. Yu, B. Xiang, B. Zhou, and H. Schütze, "Simple question answering by attentive convolutional neural network," *Proc. of the 26th International Conference on Computational Linguistics (COLING)*, 2016.