Knowledge graph augmented neural networks

Arpan Mangal
Indian Institute of Science, Bengaluru
arpanmangal@iisc.ac.in

G P Shrivatsa Bhargav
Indian Institute of Science, Bengaluru
bhargavs@iisc.ac.in

Abstract

In this work, we propose to enhance neural models with world knowledge in the form of Knowledge Graph (KG) fact triples for Natural Language Processing (NLP) tasks. Existing models show significant improvement in performance for text classification with News20, other datasets. We propose some modification and intuitive ideas to improve upon that.Our model improves on both training speed and accuracy. We have got significant improvement even when training on just half of the dataset.

1 Introduction

Neural networks have seen great success in various tasks in recent years. They excel at pattern identification and automatically extracting features relevant to the task at hand. However, lots and lots of data is needed to train neural models.The current neural models are thus constrained by the following issues:

1. It is not possible to learn everything about a task just from the training data. Common knowledge like bench is furniture isnt always present in the datasets.

2. Some domains dont have sufficient data. Some prior general information will make it easier to train neural models in these domains. This is analogous to how humans pick up new skills.

In this work, we show that (Annervaz et al., 2018) can be improved in accuracy and training speed by using entity and relation shortlisting schemes instead of clustering.

2 Related Works

1. An approach to enable neural networks to remember facts is to make memory units available to them. Memory Network(Weston et al., 2014) is an architecture that can learn to store key facts about the text in the long term memory. These have been shown to perform well in reading comprehension tasks with multiple facts to remember and reason over. But this approach is not scalable when a lot of information has to be used to perform the task.

2. In the area of knowledge graph representation, DKRL (Xie et al., 2016) is a knowledge graph encoding scheme that emphasizes on semantic description of text. It is based on TransE (Bordes et al., 2013)for encoding structure. If we have an entity e1, a relationship r, then we can get the entity e2 related to e1 by r as e2 = e1 + r.

3. The recent work Learning beyond datasets (Annervaz et al., 2018) successfully integrates neural sequence models with facts from the knowledge graph. The model is tested in a classification setting. In the pretraining step, the entities and relationships are clustered into 20 groups separately. A CNN is used to obtain a representation for each cluster. During training, the sentence is first encoded with three LSTMs (Hochreiter and Schmidhuber, 1997),to produce three encodings : C_e for entity extraction, C_r relationship extraction and C_s for sentence encoding respectively. C_e and C_r are used to compute soft attention (Bahdanau et al., 2014) over the clusters of entities and relations to give E1 and R which are attention weighted combination of the entity and rela-

tion cluster embeddings. As per DKRL (Xie et al., 2016) E2 is obtained as E2 = E1+R. The completed fact triplet is then concatenated with Cs and used to make the final prediction. The model improves accuracy by 3% on News20 and 4% on SNLI compared to the vanilla LSTM baseline.

4. (Türe and Jojic, 2016) uses knowledge graphs to answer first oder factoid questions. The answers have to be a single property of a single entity in the knowledge graph. To answer the question, classifiers are trained to tag named entities and predict relationship type in the question. These are used get the appropriate fact form the knowledge graph.

5. (Lukovnikov et al., 2017) executes N-gram match with input and entitiy names from KG for entity shortlisting. Candidate entities are used to shortlist predicates.

Our work uses some of these ideas. We propose various schemes to shortlist entities and relationships from KG. These ideas are quite intuitive and work well.

3 Knowledge Graph

We have used subset of Freebase Knowledge graph i.e Freebase15k. It contains:

1. Around 15K entities.
2. Around 1345 relations
3. Around 4.9 M fact tripes
4. Entity,relation names and entity descriptions

4 Dataset

We have experimented on **News20** dataset. It has

1. 20 classes (sports,religion,...)
2. Around 18k documents
3. Balanced class distribution
4. Domain overlap with freebase (shown in (Annervaz et al., 2018))

5 Baseline implementation

As a baseline, we have implemented the model described in (Annervaz et al., 2018). We go through the details of the baseline model in this section.

5.1 Entity and Relation Embeddings

We have used DKRL embedding (Xie et al., 2016). It uses convolution neural network as encoder.The CNN architecture has five layers, taking the whole description of a certain entity as the input after preprocessing, and output description-based representations of this entity. The entity embedding will then be learned to minimize the energy function of DKRL. Figure 1 depicts the model.

Training: The DKRL model can be stated as a parameter set = (X, W(1) , W(2) , E, R) where X, E, R stand for the embeddings of words, entities and relations, and W (1), W (2) stand for the convolution kernels in different layers. It minimize the following margin-based score function as objective for training:

$$L = \sum_{(h,r,t)\in T} \sum_{(h,r,t)\in T'} max(\gamma + d(h+r,t) - d(h+r,t), 0)$$

where $\gamma > 0$ is a margin hyperparameter, d(h + r, t) is the dissimilarity function between **h+r** and **t**. T' is the negative sampling set of T,in which the head, tail or relation are randomly replaced by another entity or relation in a triple. Stochastic gradient descent(SGD) is used as optimization technique.

5.2 KG Augmented Classifier

The entities and relationships of the KG(Freebase,FB15k) are encoded using DKRL, explained earlier. Using attention mechanism on all the entities and relations wont scale well. So K1 entity clusters and K2 relations clusters are formed before training. We use these clusters in the model. We present details of clustering and learning the cluster representatives in the next section Let $e_i \in \mathbb{R}^m$ stand for the encoding of the i^{th} entity cluster representative and $r_j \in \mathbb{R}^m$ stand for j^{th} relationship cluster representative. The input document in the form of concatenated word vectors (using Glove embedding),$x = (x_1, x_2, ..., x_T)$ is first encoded using an LSTM module as follows,

$$h_t = f(x_t, h_{t1})$$

and

$$o = \frac{1}{T}\sum_{t=1}^{T} h_t$$

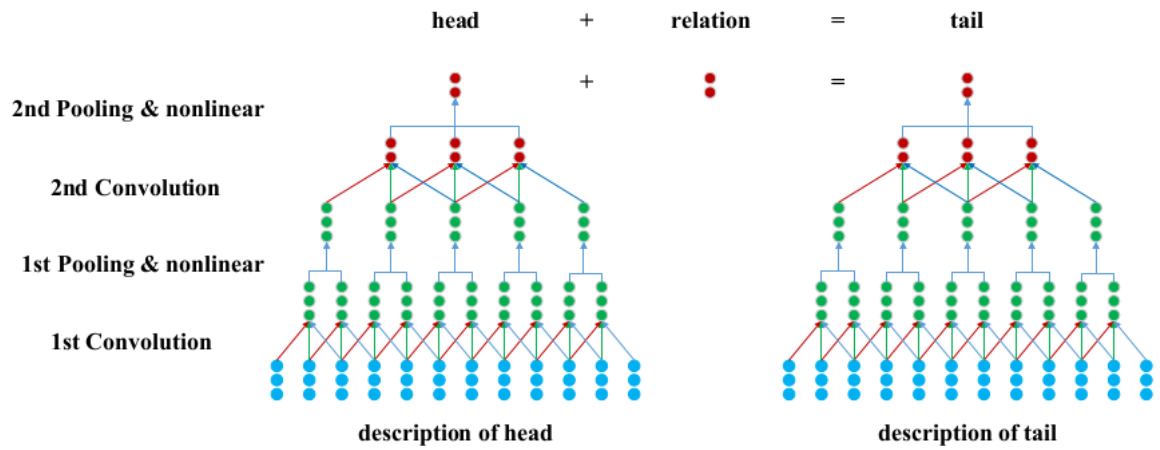

Figure 1: The Convolutional Neural Network Encoder (Xie et al., 2016)

$h_t \in \mathbb{R}$ is the hidden state of the LSTM at time t , f is a non-linear function and T is the sequence length. Then a context vector is formed from o as follows,

$$C = ReLU(o^T W)$$

where,$\mathbf{W} \in \mathbb{R}^{n \times m}$ represent the weight parameters. The same procedure is duplicated with separate LSTMs to form two separate context vectors, one for entity retrieval (C_E) and one for relationship retrieval (C_R).

The attention for the i^{th} entity cluster, e_i is given by

$$\alpha_{e_i} = \frac{exp(C_E^T e_i)}{\sum_{j=0}^{|EC|} exp(C_E^T e_j)}$$

where $|EC|$ is the number of entity clusters in the KG.

Similarly the attention for a i^{th} relation cluster r_i is computed as

$$\alpha_{r_i} = \frac{exp(C_R^T r_i)}{\sum_{j=0}^{|RC|} exp(C_R^T r_j)}$$

where $|RC|$ is the number of relation clusters in the KG.The final entity and relation vector retrieval is computed by the weighted sum with the attention values of entity/relation cluster vectors.

$$e = \sum_{i=0}^{|EC|} \alpha_{e_i} e_i$$

and

$$r = \sum_{i=0}^{|RC|} \alpha_{r_i} r_i$$

After the final entity and relation vectors are computed.The KG embedding technique used for the experiment is DKRL which inherently uses the TransE model assumption (h + r = t). Therefore, Thus the fact triplet retrieved is F = [e, r, e + r], where$\mathbf{F} \in \mathbb{R}^{3m}$. Figure 2 shows the block diagram of the model.

5.3 Entity and relation Clustering

We have used standard K-means clustering to form the clusters for entity(and relations). Euclidean distance between vectors is used as distance metric.

Our model requires each cluster to have equal number of vectors. Therefore, after executing K-means clustering, smallest cluster size (s) (which is 352 in our experiment) is used as cluster size, and these many vectors are sampled from each cluster without replacement.

5.4 Entity and Relation Cluster Representation

Attending to all the entities and relationships scales very poorly with the size of the knowledge graph. So clusters of entities and relations are attended to instead. For each cluster, vectors are stacked to form $\mathbf{E}$ - the 2D input to the CNN encoder, where $\mathbf{E} \in \mathbb{R}^{m \times n}$ where m is the number of vectors in each cluster and n is the dimension of the entity/relation embedding. 1-D convolution filters which slide only along the columns of $\mathbf{E}$ are used. Two layers of convolutions with max pooling after each convolution layer are used to com-

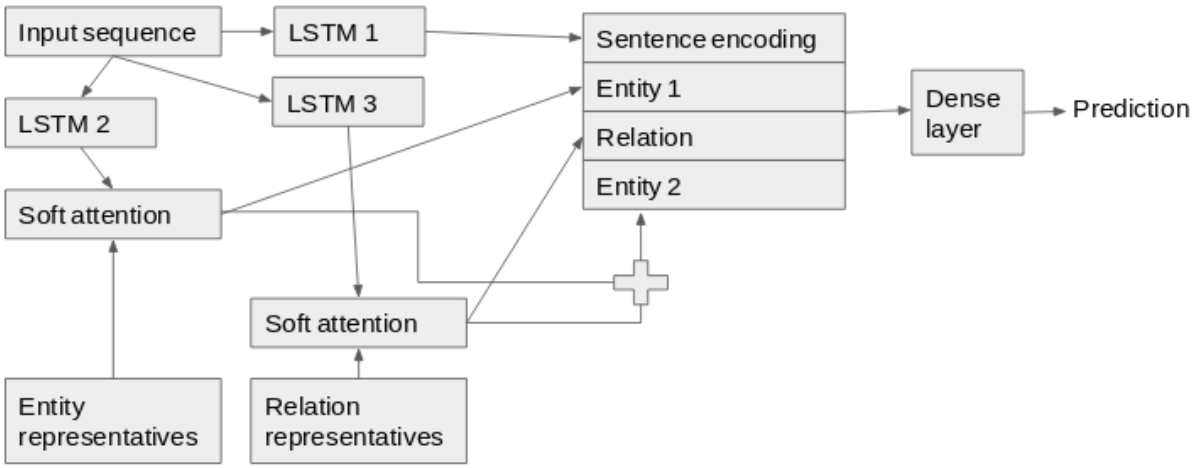

Figure 2: KG augmented classifier(Annervaz et al., 2018)

pute the cluster representatives. This convolution architecture is a part of the architecture described in the previous section so the training happens simultaneously.

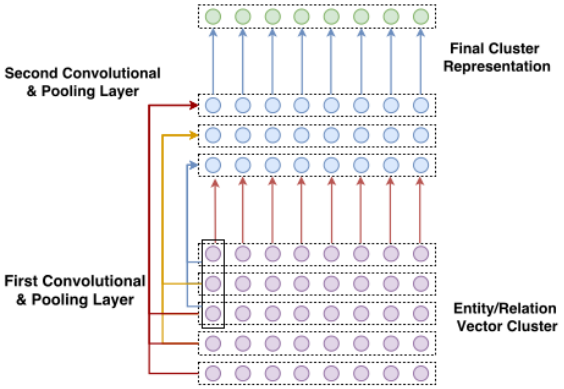

Figure 3: Convolution model cluster representation(Annervaz et al., 2018)

6 Enhancements

Clustering loses a lot of information. This makes the model unsuitable for tasks that need precise information like question answering.
Instead of clustering, we can shortlist entities and relations that are relevant to the task. These shortlisted entities and relations are given to the model in place of the cluster representatives. It will keep the model scalable while making it suitable for tasks that need precision.

We have experimented with the following ideas

- Entity Clusters and Relation Clusters
- Entity Shortlisting and Relation Clusters
- Entity Shortlisting and Relation Shortlisting

6.1 Entity Shortlisting

It is very intuitive that if we use our document to shortlist entities, it will give us more relevant entities than clustering. We have used entity names from the knowledge base to do n-gram match with input document. To generate the set of candidate subjects $\mathbf{C}_s$, all word n-grams of size 1 to L contained in document are retrieved, filtered and used for searching matching entities according to the following rules (Lukovnikov et al., 2017):

- An entity whose label exactly matches a n-gram is added to the set of candidates $\mathbf{C}_s$.
- An n-gram that is fully contained inside a bigger n- gram that matched at least one entity label is discarded, unless the bigger n-gram starts with one of the following words: *the, a, an, of , on, at,by* .
- In case an exact match is not found for an n-gram, we search for entities whose label has an edit distance with the n-gram of at most 1 and add them to $\mathbf{C}_s$.
- If there are several entities matching a n-gram, the entities are ranked by the number

of triples in the KG in which they play the role of the subject. Only the m entities with highest rank are added to the candidate set(in our experiment m = 20).

6.2 Relation Shortlisting

We have used two approaches to shortlist relations from KG.

1. **Most Common K relation**
   In this approach, we extracted K(=20 in our experiment) common relationships that exists between the shortlisted entities. It is intuitive, because entities are shortlisted based on n-gram match with document.

2. **Using Cosine similarity**
   Our previous approach of shortlisting depends on how good entity shortlisting is, therefore it is not that robust. In this approach, we directly shortlisted relations based on their similarity with input document. We have used glove embedding to get vector for relation and document. We have taken average of glove word embedding for document and relationship words. Cosine similarity is used as similarity measure between vectors.

7 Experiments

We have implemented the following models:

- Plain LSTM
- LSTM + KG clusters (prior work)
- LSTM + entity shortlist + relation clusters
- LSTM + entity and relation shortlists

The models are compared in training speed and accuracy in various settings.

Data preprocessing

The documents contain headers(like category:religion) and footers which easily give away the category. We have removed them so that we can assess our models in a real world scenario.

Settings:

We keep most of the settings fixed across the models so that the difference made by the shortlisting schemes can be studied. We split our dataset as

- Train = 13.5k samples
- Validate = 1.8k samples
- Test = 3.3k samples

Hyper-parameters	Value
KG embedding size	50
GloVe embedding size	300
Sequence length	300
LSTM hidden state dimension	200
Number of epochs	50
Batch size	32
Number of clusters/shortlists	20

Table 1: Hyper-parameters values.

Hyper-parameters values are summarized in the Table 1. These parameters are kept the same across all models so that the effect of entity and relation shortlisting schemes can be studied and compared.

8 Results

We have trained our model in different scenarios. For each, we have tested it on full training data and half of the training data. We have kept validation and test set same across all settings. Accuracy and Training speed is taken as comparison metric. Accuracy in all settings is summarized in the table. Comparison based on Training speed is depicted in Figure 4.

From Table 2, we see that our models outperform the baseline. When the size of the training set is reduced to 50%, the degradation of performance in our models is smaller than that of the baseline's.

Figure 4 shows that our models achieve peak performance in about one fourth of the time taken by the baseline. One of the main reasons for the speedup is that there is no CNN to train our models. The right entities and relations are already available at training time and they(cluster representatives) don't have to be learned. The fact that the model that shortlists entities and uses clustered relations trains as fast as the shortlisting models shows that obtaining the cluster representatives for the entity clusters is hard and impedes training. We examined the clusters and there wasn't a clear pattern. So it is hard to find a cluster representative that suits all the training examples well.

Model	Accuracy(100% data)	Accuracy(50% data)
Plain LSTM	62.30%	56.31%
KG clusters(prior work)	65.10%	61.05%
Entity shortlist, relation clusters	67.04%	63.97%
Entity shortlist, relation shortlist(most common)	66.80%	64.27%
Entity shortlist, relation clusters(cosine similarity)	66.57%	64.15%

Table 2: Experiment Results.

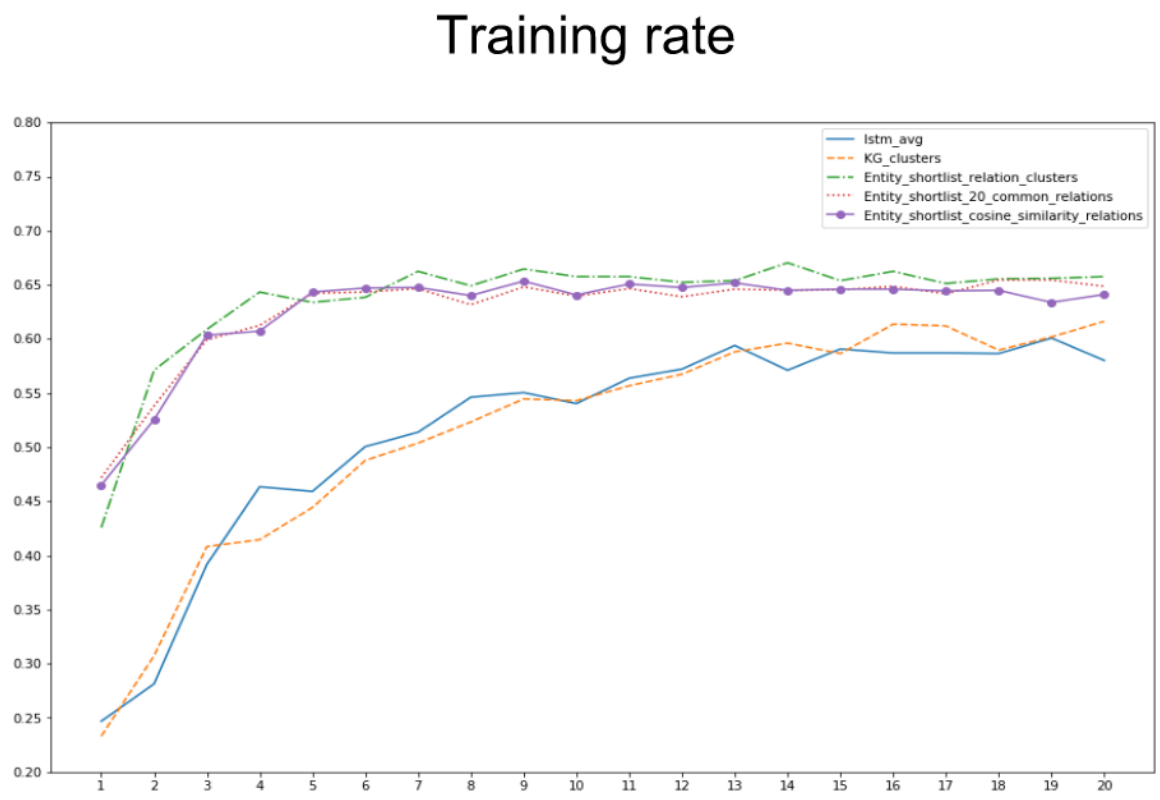

Figure 4: Training rate comparison

8.1 Ablation Test

In order to analyze the importance of entity,relation and complete fact appended to input document encoding, we have performed ablation test. In this we execute models with complete fact triplet, entity + relation and only entity. Results are summarized in the table 3, where,

- Fact Triplet : FT
- Entity + Relation: E + R
- Only Entity : E

It is evident from Table 3 that all the schemes that pick relations perform similarly. The major gains are coming from entity shortlisting. In the last column, where only the shortlisted entities are attended to, the model becomes similar to an LSTM with attention mechanism over the input document because the entities are words from the document itself.

9 Conclusion and future work

Through these experiments we have shown that augmenting neural models with external world knowledge does indeed enhance performance especially when the amount of training data is low. Our entity and and relation shortlisting schemes have improved upon the baseline in accuracy, training speed, scalability and the computational and memory footprint of the model.

Model	FT	E + R	E
Entity shortlist,relation clusters	67.04%	67.02%	
Entity shortlist, relation shortlist(most common)	66.80%	66.48%	64.98%
Entity shortlist, relation clusters(cosine similarity)	66.57%	67.16%	

Table 3: Ablation Test

Our models can be improved further in the following ways:
Entity linkers can be used when available and n-gram matches can be used as a fall back.
The shortlisted entities and relations are being attended to independently. While this works, it could be better to do away with this independence assumption and condition attention mechanism to pick the relation on the chosen entity.
The ablation study shows that all the schemes to pick relations perform similarly. So some more relation shortlisting schemes could be explored next.

References

K M Annervaz, Somnath Basu Roy Chowdhury, and Ambedkar Dukkipati. 2018. Learning beyond datasets: Knowledge graph augmented neural networks for natural language processing https://arxiv.org/abs/1802.05930.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473. http://arxiv.org/abs/1409.0473.

Antoine Bordes, Nicolas Usunier, Alberto Garcia-Duran, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26*, Curran Associates, Inc., pages 2787–2795. http://papers.nips.cc/paper/5071-translating-embeddings-for-modeling-multi-relational-data.pdf.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735.

Denis Lukovnikov, Asja Fischer, Jens Lehmann, and Sren Auer. 2017. Neural network-based question answering over knowledge graphs on word and character level. In *Proceedings of the 26th International World Wide Web Conference*. pages 1211–1220. http://dx.doi.org/10.1145/3038912.3052675.

Ferhan Türe and Oliver Jojic. 2016. Simple and effective question answering with recurrent neural networks. *CoRR* abs/1606.05029. http://arxiv.org/abs/1606.05029.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2014. Memory networks. *CoRR* abs/1410.3916. http://arxiv.org/abs/1410.3916.

Ruobing Xie, Zhiyuan Liu, Jia Jia, Huanbo Luan, and Maosong Sun. 2016. Representation learning of knowledge graphs with entity descriptions. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'16, pages 2659–2665. http://dl.acm.org/citation.cfm?id=3016100.3016273.