Parameter-Efficient Neural Question Answering Models via Graph-Enriched Document Representations

Anonymous EMNLP submission

Abstract

As it currently stands, a major issue in applying GCNs to NLP is document representation. In this paper, we show that a GCN enriched document representation greatly improves the results seen in HotPotQA, even when using a trivial topology. Our model (gQA), performs admirably when compared to the current SOTA, and requires little to no preprocessing. As the computational footprint of modern NLP systems grows, it becomes increasingly important to arrive at more efficient models. We show that by employing graph convolutional document representation, we can arrive at a question answering system that performs comparably to, and in some cases exceeds the SOTA solutions, while using less than 5% of their resources in terms of trainable parameters.

1 Introduction

In the past decade, Deep Learning (DL) approaches have dominated the field of Artificial Intelligence (AI) across a wide a array of domains. Recent developments in neural information processing methods sparked a revolution in Natural Language Processing (NLP), which resembles advances made in Computer Vision (CV) at the beginning of this new chapter of AI development. This progress was made possible through increasingly more advanced representation methods of natural language inputs. Initially shallow pre-training of early model layers became standard in NLP research through methods such as word2vec (Mikolov et al., 2013). Subsequent progress followed trends similar to those in CV, which naturally led to pre-training of multiple layers of abstraction. These advancements resulted in progressively deeper hierarchical language representations, such as those derived using self-attention mechanisms in transformer-based architectures (Vaswani et al., 2017). Currently SOTA NLP systems use representations derived from pre-training of entire language models on large quantities of raw text, and often involve billions of parameters. Neural language modelling methods came to prominence in recent years due to the development of techniques such as BERT (Devlin et al., 2018). The success of neural network based Machine Learning (ML) models, especially those involving very deep architectures, can be attributed to their ability to derive informative embeddings of raw data into submanifolds of real vector spaces. The common idea behind these developments is that we can learn syntax and semantics of natural languages by training a DL model in a self-supervised fashion on a corpus of raw text. The general language representations of words in vector spaces induced by modern neural NLP models can be transferred to other domain-specific architectures and further tuned for downstream tasks. One could argue that this transfer learning benefit, as well as straightforward applicability to cascaded and multi-task learning, is the most exciting recent development in representation learning research in the context of natural language data.

However, in many applications of practical interest in the area of NLP, a need arises to represent language on a supra-lexical level. In this paper we concern ourselves with the evaluation of modern representation methods on document level. Some canonical tasks requiring such coarser embedding methods include document classification, question answering, and summarization.

Traditionally, document embedding methods combined token-level embeddings by mapping them to a single vector through a variety of models (e.g. vector averaging and word2vec inspired methods (Le and Mikolov, 2014) (Kenter et al., 2016) (Hill et al., 2016) (Arora et al., 2016) (Pagliardini et al., 2017) (Kiros et al., 2015) (Hill et al., 2016) (Chen, 2017) (Logeswaran and Lee, 2018), atten-

tion mechanisms (Vaswani et al., 2017) (Devlin et al., 2018) (Reimers and Gurevych, 2019) (Radford et al., 2018) (Radford et al., 2019), convolutional architectures (Liu et al., 2017), or gated recurrent units (Hochreiter and Schmidhuber, 1997) (Cho et al., 2014)). Most of the previous methods did not impose topological structures on the token embeddings, although recurrent networks impose simple ordering based topologies, and recursive networks could be used to embed sentences with tree-based structures, which can then be combined into a document embedding.

1.1 Arbitrary Topologies

A **Document Topology** is the adjacency matrix that determines relationships between parts of the document (usually sentences or paragraphs). For instance, consider a news article like those found in CNN/Dailymail (See et al., 2017). Within this dataset, the document topology resembles a sparse diagonal matrix, implying that the sentences are meant to be read sequentially in the order they appear.

This is not the case in other settings. Consider an entry in the pushshift.io Reddit dataset (Baumgartner et al., 2020) (2016 election data). In this dataset, Reddit threads are provided where comments are meant to be read in a hierarchical manner. Given that the sentences within a comment are meant to be read sequentially, excluding the possibility of links to other comments, that means the document topology in this dataset is entirely nontrivial. In such cases a more sound way to combine token level or sentence level embeddings into a document vector is via a model incorporating a graph structure. A family of neural embedding models taking graph topology into account can be grouped under the label of graph neural networks (GNN). In recent years multiple instantiations of this idea appeared in literature (Bruna et al., 2013) (Henaff et al., 2015) (Duvenaud et al., 2015) (Li et al., 2015) (Defferrard et al., 2016) (Kipf and Welling, 2016).

As an example of a downstream task which benefits from GCN encoding consider question answering on the HotPotQA Dataset, (Yang et al., 2018). This is a multi document question answering dataset where documents may or may not be related to the question. Additionally, some questions may require inferring from "multiple hops" within documents to reach a conclusive answer.

1.2 Advantages to non-naïve representation.

A common theme in contemporary multi-document summarization techniques can be defined as figuring out ways to cleverly combine multiple documents into a single document. For instance, (Lebanoff et al., 2018) suggested selecting sentences which introduced the most unique information, which essentially maximized the MMR metric. However, such approaches do not allow for conflicting opinions, as two similarly stated sentences might have drastically different opinions but due to the requirement of maximizing MMR only one would be selected. This would be a common occurrence in multi-document datasets- particularly in the case of reviews for instance. Furthermore, such approaches to multi-document summarization rely on the assumption that articles are independent. In the case of reviews, this might not be the case- reviewers may echo or argue with the statements given by other reviewers.

While such limitations might not be an issue in select use cases, they do not inherently translate well between domains, and thus cannot be assumed to be synergistic to other NLP tasks.

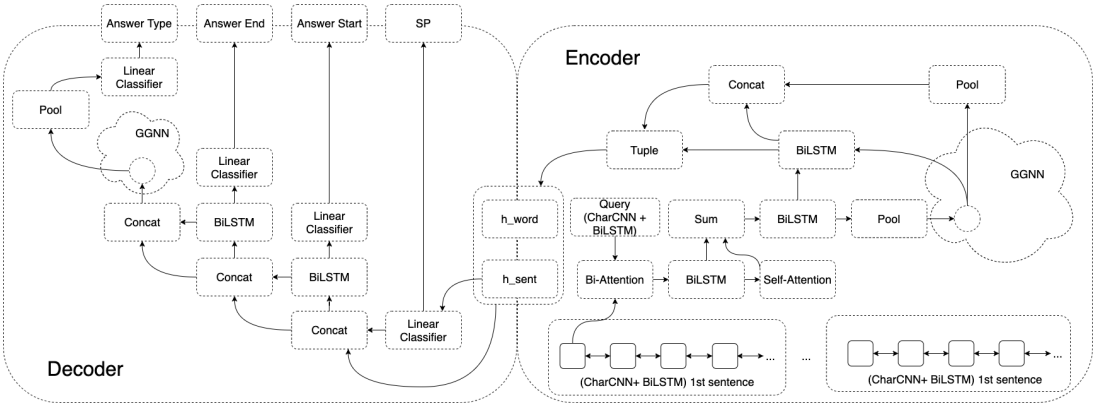
By contrast, other approaches (e.g. (Qian et al., 2018)), suggest methods that allow for the specification of document topology together with the input to the summarization system. Their work is split up into two separate models. In the first model the document topology is specified over the sentences, while in the second it is specified over the words. Given the initial inspiration to model Twitter conversations, where tweets often do not follow a sentence structure, specifying topology over words seems to be an adequate approach.

In the following section, we introduce gQA our test model for multi-document question answering. The goal of this project wasn't to beat SOTA results on QA, but rather to evaluate the difference in document representation for synergistic NLP tasks. However, to our surprise, introducing graphical representation on document topology to a baseline QA approach greatly improved performance, resulting in a model that performs competitively with SOTA systems, while avoiding costly pre-training of over-parametrized language models.

2 Model

In the design of gQA, we aimed for a model that would work over arbitrary input topologies for multi-document question answering.

Figure 1: Diagram of the gQA model. Notice that every sentence is encoded separately, but the sentence embedding from the original encoder is never used.



2.1 Architecture

Since our primary objective wasn't to improve SOTA approaches, we deviated from traditional models utilized for question answering and particularly that of multidoc QA. With the exception of HQN (Fang et al., 2019), multihop QA is very rarely done with the aid of a GCN. In particular we deviated from traditional models in the following ways.

2.2 Separate Encoding of Sentences

Inspired by (Qian et al., 2018), we introduce individual sentence representations. Given that sentences in Wikipedia, and particularly HotPotQA, can be viewed as "supporting facts" for questions rather than outlining some long detailed chronology it made sense to consider these sentences as individual nodes in a larger concept graph.

In GraphIE, an identical approach is utilized by considering twitter accounts as individual nodes and performing NER over the entire graph.

2.3 Encoder

By contrast, when compared to GraphIE, we consider the GNN as well as the "decoder" BiLSTM as part of our encoder. That is to say, we encode the sentences disjointly, perform inter-sentence enrichment via the Gated Graph Neural Network (GGNN) (Li et al., 2017), and then intra-sentence enrichment by initializing a BiLSTM to the enriched sentence embeddings and feeding the word embeddings as input.

The propagation and output model for the GGNN is as follows

$$h_v^{(1)} = [x_v^\top, 0]^\top$$

$$a_v^{(t)} = A_v^\top [h_1^{(t-1)\top} ... h_N^{(t-1)\top}]^\top + b$$

$$z_v^t = \sigma(W^z a_v^{(t)} + U^z h_v^{(t-1)})$$

$$r_v^t = \sigma(W^r a_v^{(t)} + U^r h_v^{(t-1)})$$

$$\widetilde{h_v^{(t)}} = tanh(W a_v^{(t)} + U(r_v^t \odot h_v^{(t-1)}))$$

$$h_v^{(t)} = (1 - z_v^t) \odot h_v^{(t-1)} + z_v^t \odot h_v^{(t)}$$

$$h_G = tanh(\sum_{v \in V} \sigma(i(h_v^{(T)}, x_v)) \odot tanh(j(h_v^{(T)}, x_v)))$$

[1]

In order to ablate the contribution of document representation, we wanted this to be a naïve implementation of GNNs on top of the baseline HotPotQA. For this purpose, we introduce a Bi-Attention and Self-Attention layer preceding the GNN. Bi-Attention is done per sentence and is conditioned on the question, while self-attention is global over the entire set of documents. Excluding the GNN, our encoder is identical to that presented in the baseline of HotPotQA (Yang et al., 2018).

For the first layer of our encoder, we used a BiLSTM + CharCNN, with GloVe embeddings.

2.4 Sentence Level Attention

An issue that GraphIE faced with respect to their graph module is applying an attention mechanism to the sentences that represent twitter accounts. Namely, by applying the attention mechanism to sentences Pre-GCN, the GCN would no longer con-

[1] A soft attention mechanism is applied with $\sigma(i(h_v^T, x_v))$, which in this case is the sentence level attention, i and j denote neural networks.

verge.

We circumvented this by utilizing a gated graph neural network (GGNN) with a passthrough. Namely, the graph neural network is simply an extension of the BiLSTM encoder, and as such GGNN output is equivalent to an attention pooling layer on the encoder. By comparison, GraphIE does not feed the output of its GGNN back through a BiLSTM encoder-rather, they immediately decode the GCN output.

The attention weights are as follows

$$e_i^s = (W_1 h_i^s + b_1) + (W_2 h_i^{\tilde{s}} + b_2) + W_3 h^q$$

$$\text{out} = \text{softmax}(v^t \tanh\left(e_i^s\right)) \cdot h_i^{\tilde{s}}$$

Where h_i^s and $h_i^{\tilde{s}}$ refers to the hidden embedding of the i^{th} sentence Pre and Post GNN respectively, h^q refers to the hidden embedding of the query.

2.5 Decoder

The head utilized is nearly identical to that presented in HotPotQA's baseline, with the exclusion of how "answer_type" is computed.

2.5.1 Supporting Sentences

Given the output of the post-GNN BiLSTM encoder, the sentence embeddings are fed to a linear classification layer in order to determine the supporting sentences.

The contextualized word embeddings and contextualized sentence embeddings for the supporting sentences are then concatenated onto h_{word} and h_{sent} respectively.

2.5.2 Answer Start

Given $(h_{\text{word}}, h_{\text{sent}})$ from the supporting sentence computation, a sentence level BiLSTM is initialized to h_{sent} and given h_{word} as input.

The word level output of this BiLSTM is fed to a linear classification layer, and likewise to above the BiLSTM output is concatenated onto the latent word and sentence embeddings respectively.

2.5.3 Answer End

Answer end is computed identically to answer start, including the output being concatenated onto the latent embeddings.

2.5.4 Answer Type

A GGNN is fed h_{sent} as input, which is then fed to a sentence level attention pooling layer and eventually another linear classification layer.

Model	EM	F_1
gQA-S	68.47	76.85
Baseline	44.60	59.02
HQN	66.07	79.36
C2F Reader	67.98	81.24
Unsup. Decomp.	66.33	79.34

Table 1: Comparison of gQA small and to the baseline and others. At the time of our experiments, C2F Reader was SOTA.

3 Preprocessing

We define the adjacency matrix used in our GGNNs by the following formula

$$M = \frac{1}{2}(B+(\lambda)(S-I)+L)^T(B+(\lambda)(S-I)+L)$$

Where B is a rotated diagonal matrix, $\lambda(S-I)$ is the regularization parameter [2], and L is an adj. matrix that connects the last sentence of every document to the first sentence of every other.

4 Results

gQA-S (Small) uses 200-dim word embeddings and is 9.04M parameters. Notice that gQA-S is comparable to HQN with respect to answer exact match and answer F_1. Training the model takes about 12hrs on a GTX 1080. We used a 90/5/5 split over 90400 examples.

5 Discussion

A question remains about the performance of gQA-S when enriched with a large language model, for instance RoBERTa, thus creating gQA-L.

Furthermore an improved pooling mechanism, like multi-head attention, would significantly aid in this direction. We have not included it here, since our purpose was to investigate naïve extension to the baseline model in order to gauge the contribution of graph representation.

6 Future Work

We have presented a minimal extension to the original baseline of HotPotQA that performs comparably to SOTA with minimal preprocessing and no hyper parameterization. The model presented here has the ability to easily transfer to single-doc QA, as the only difference would be the adjacency matrix. This might be explored in future work.

[2] $S-I$ refers to the cosine similarity matrix with a zero'ed diagonal. During training, $\lambda = 0.5$ but during testing $\lambda = 0.05$.

References

Sanjeev Arora, Yingyu Liang, and Tengyu Ma. 2016. A simple but tough-to-beat baseline for sentence embeddings.

Jason Baumgartner, Savvas Zannettou, Brian Keegan, Megan Squire, and Jeremy Blackburn. 2020. The pushshift reddit dataset.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs.

Minmin Chen. 2017. Efficient vector representation for documents through corruption. *arXiv preprint arXiv:1707.02377*.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Michal Defferrard, Xavier Bresson, and Pierre Vandergheynst. 2016. Convolutional neural networks on graphs with fast localized spectral filtering.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. 2015. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232.

Yuwei Fang, Siqi Sun, Zhe Gan, Rohit Pillai, Shuohang Wang, and Jingjing Liu. 2019. Hierarchical graph network for multi-hop question answering.

Mikael Henaff, Joan Bruna, and Yann LeCun. 2015. Deep convolutional networks on graph-structured data.

Felix Hill, Kyunghyun Cho, and Anna Korhonen. 2016. Learning distributed representations of sentences from unlabelled data. *arXiv preprint arXiv:1602.03483*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Tom Kenter, Alexey Borisov, and Maarten De Rijke. 2016. Siamese cbow: Optimizing word embeddings for sentence representations. *arXiv preprint arXiv:1606.04640*.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks.

Ryan Kiros, Yukun Zhu, Russ R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-thought vectors. In *Advances in neural information processing systems*, pages 3294–3302.

Quoc Le and Tomas Mikolov. 2014. Distributed representations of sentences and documents. In *International conference on machine learning*, pages 1188–1196.

Logan Lebanoff, Kaiqiang Song, and Fei Liu. 2018. Adapting the neural encoder-decoder framework from single to multi-document summarization.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2015. Gated graph sequence neural networks.

Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard Zemel. 2017. Gated graph sequence neural networks. *arXiv:1511.05493*.

Chundi Liu, Shunan Zhao, and Maksims Volkovs. 2017. Learning document embeddings with cnns. *arXiv preprint arXiv:1711.04168*.

Lajanugen Logeswaran and Honglak Lee. 2018. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Matteo Pagliardini, Prakhar Gupta, and Martin Jaggi. 2017. Unsupervised learning of sentence embeddings using compositional n-gram features. *arXiv preprint arXiv:1703.02507*.

Yujie Qian, Enrico Santus, Zhijing Jin, Jiang Guo, and Regina Barzilay. 2018. Graphie: A graph-based framework for information extraction.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. *URL https://s3-us-west-2. amazonaws. com/openai-assets/researchcovers/languageunsupervised/language understanding paper. pdf*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Zhilin Yang, Peng Qi, and Saizheng Zhang. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering.