Using Local Knowledge Graph Construction to Scale Seq2Seq Models to Multi-Document Inputs

Angela Fan
FAIR / LORIA

Claire Gardent
CNRS / LORIA

Chloé Braud
CNRS / LORIA

Antoine Bordes
FAIR

angelafan,abordes@fb.com
claire.gardent,chloe.braud@loria.fr

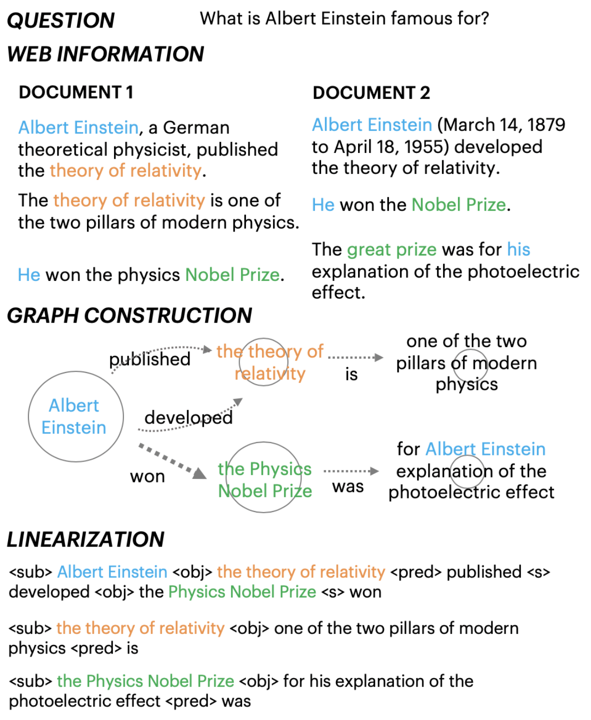
Abstract

Query-based open-domain NLP tasks require information synthesis from long and diverse web results. Current approaches extractively select portions of web text as input to Sequence-to-Sequence models using methods such as TF-IDF ranking. We propose constructing a local graph structured knowledge base for each query, which compresses the web search information and reduces redundancy. We show that by linearizing the graph into a structured input sequence, models can encode the graph representations within a standard Sequence-to-Sequence setting. For two generative tasks with very long text input, long-form question answering and multi-document summarization, feeding graph representations as input can achieve better performance than using retrieved text portions.

1 Introduction

Effective information synthesis is at the core of many Natural Language Processing applications, such as open-domain question answering and multi-document summarization. In such tasks, a fundamental challenge is the ability to distill relevant knowledge from hundreds of thousands of tokens of noisy and redundant input such as webpages. Current approaches predominantly conduct TF-IDF-based information extraction to identify key portions of the information, and then provide this as sequential input to a Sequence-to-Sequence (Seq2Seq) model. The sub-selected portions are limited to a few thousand words, as models often struggle to encode much longer sequences.

In this work, we propose a mechanism to restructure free text into local knowledge graphs that are then linearized into sequences, creating a canonical form in which information is presented to models. By constructing a graph intermediary, redundant information can be merged or discarded, producing substantially compressed input


QUESTION What is Albert Einstein famous for?

WEB INFORMATION

DOCUMENT 1

Albert Einstein, a German theoretical physicist, published the theory of relativity.

The theory of relativity is one of the two pillars of modern physics.

He won the physics Nobel Prize.

DOCUMENT 2

Albert Einstein (March 14, 1879 to April 18, 1955) developed the theory of relativity.

He won the Nobel Prize.

The great prize was for his explanation of the photoelectric effect.

GRAPH CONSTRUCTION

Albert Einstein
published
developed
won
the theory of relativity
is
one of the two pillars of modern physics
the Physics Nobel Prize
was
for Albert Einstein explanation of the photoelectric effect

LINEARIZATION

<sub> Albert Einstein <obj> the theory of relativity <pred> published <s> developed <obj> the Physics Nobel Prize <s> won

<sub> the theory of relativity <obj> one of the two pillars of modern physics <pred> is

<sub> the Physics Nobel Prize <obj> for his explanation of the photoelectric effect <pred> was

Figure 1: **Multi-Document Input to Linearized Graph** Multi-document input resulting from web search queries are converted to a graph structured knowledge base using coreference resolution and information extraction, then linearized into a sequence for Seq2Seq models. Color indicates coreference resolution. Node weight is indicated by circle radius and edge weight by line thickness.

— small enough to be fully encoded by Seq2Seq models. Such a method can be seen as merging previous work on symbolic knowledge bases for information extraction with newer approaches using deep neural networks to encode knowledge.

Our approach, shown in Figure 1, takes a query and its corresponding multi-document web search results and builds for each query a specific local knowledge graph. We present several modeling contributions to effectively encode the entire graph as a sequence and attend to the most relevant portions within this linearization. We demonstrate the effectiveness of this approach on two large-scale generative tasks with both long and noisy multi-document web input and paragraph length output: long-form question answering on the ELI5 dataset (Fan et al., 2019) and Wikipedia lead paragraph generation as a multi-document summarization problem (Liu et al., 2018b).

2 Related Work

Interest in generative sequence modeling has intensified due to recent improvements (Peters et al., 2018; Devlin et al., 2018; Radford et al., 2019), making the challenge of information synthesis more relevant. In contrast to extractive tasks which only require models to identify spans and can do so effectively on long documents by looking at the paragraphs independently, generative sequence models must combine multiple pieces of evidence from long and noisy multi-document input to generate correct and convincing responses.

2.1 Multi-Document Input

Previous work in multi-document summarization (Barzilay et al., 1999) applies various techniques to handle long input, including clustering to find similar information (Honarpisheh et al., 2008), extractive methods to select relevant sentences (Daumé III and Marcu, 2002; Gillick and Favre, 2009; Berg-Kirkpatrick et al., 2011; Di Fabbrizio et al., 2014; Bing et al., 2015; Cao et al., 2017) including maximal marginal relevance (Fabbri et al., 2019), and incorporating queries (Baumel et al., 2018) and graphs (Ganesan et al., 2010; Yasunaga et al., 2017). However, there are few large scale multi-document summarization datasets and many approaches have focused on extractive selection or hybrid extractive-abstractive models. In this work, we use graph construction to re-structure multi-document input for abstractive generation.

Advancements in question answering have examined performance on datasets with multi-document input, such as TriviaQA (Joshi et al., 2017). Various approaches have been proposed, including leveraging TF-IDF and bigram hashing with an RNN to find relevant information (Chen et al., 2017), models that score individual paragraphs for sub-selection (Clark and Gardner, 2017), and nearest neighbor search with paragraph re-ranking (Das et al., 2018a). However, these approaches have been applied to extractive question answering tasks that require span identification, rather than abstractive text generation in an information synthesis setting.

2.2 Using Knowledge Bases

Previous work has explored various ways of representing information in knowledge bases (Bordes et al., 2011) and improving these representations (Chen et al., 2013). Knowledge bases have been leveraged to improve performance on various tasks, from coreference resolution (Ng and Cardie, 2002) and question answering (Zheng, 2003; Bao et al., 2014; Cui et al., 2017; Sun et al., 2018) to signal processing (Bückner et al., 2002). Various works convert text into Abstract Meaning Representations (Liu et al., 2018a) for domains such as news (Vossen et al., 2015; Rospocher et al., 2016) and link nodes to large knowledge bases such as DBPedia (Auer et al., 2007). Wities et al. (2017) combine open information extraction with coreference and lexical inference to build knowledge representations. They apply this to tweets and analyze the accuracy on various aspects of graph construction. Das et al. (2018b) construct graphs from procedural text to track entity position to answer when and if entities are created, destroyed, or moved. In contrast, we build graphs from substantially longer multi-document input and use them for multi-sentence text generation.

Recently, many have explored neural architectures that can encode graph structured input (Bruna et al., 2013; Kipf and Welling, 2016; Beck et al., 2018; Zhou et al., 2018; Xu et al., 2018; Lai et al., 2019). These models often represent graphs as adjacency matrices to generalize architectures such as convolutional networks to graph inputs. Rather than encoding a static knowledge graph or leveraging external knowledge graphs, we build a local graph for each query and model these using standard Seq2Seq models. We leave the incorporation of graph networks for future work.

3 Graph Construction

We describe how symbolic graph representations of knowledge can be constructed from text. Our approach assumes a multi-document input (such as web pages) resulting from the execution of a query. The graph construction process (1) compresses the web search input to a significantly smaller size, allowing models to encode the entirety of the compression, and (2) reduces redundancy through merge operations, allowing relevant information to be more easily identified.

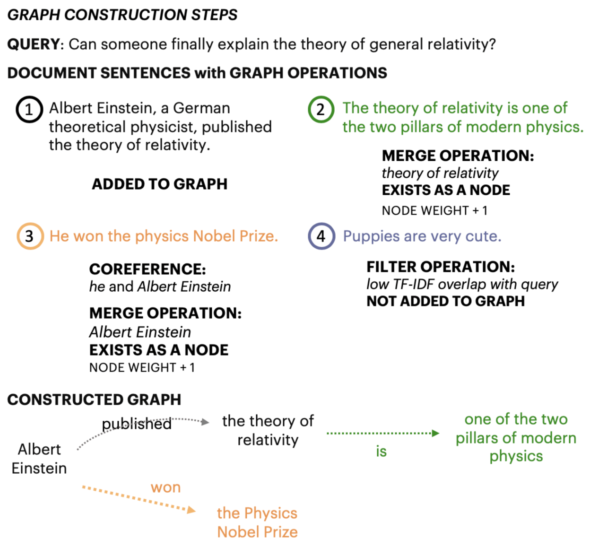

GRAPH CONSTRUCTION STEPS
QUERY: Can someone finally explain the theory of general relativity?
DOCUMENT SENTENCES with GRAPH OPERATIONS
1 Albert Einstein, a German theoretical physicist, published the theory of relativity.
ADDED TO GRAPH
2 The theory of relativity is one of the two pillars of modern physics.
MERGE OPERATION: theory of relativity EXISTS AS A NODE NODE WEIGHT + 1
3 He won the physics Nobel Prize.
COREFERENCE: he and Albert Einstein
MERGE OPERATION: Albert Einstein EXISTS AS A NODE NODE WEIGHT + 1
4 Puppies are very cute.
FILTER OPERATION: low TF-IDF overlap with query NOT ADDED TO GRAPH
CONSTRUCTED GRAPH
Albert Einstein
published
the theory of relativity
is
one of the two pillars of modern physics
won
the Physics Nobel Prize

Figure 2: Steps of **Graph Construction**. Color relates the document sentence used to produce the graph output.

Text to Triples to Graph Graph construction proceeds in several steps outlined in Figure 2. We apply *Coreference Resolution* (Clark and Manning, 2016a,b)[1] and *Open Information Extraction* (Stanovsky et al., 2018)[2] to convert sentences into a *Triple* of the form (subject, predicate, object). The sentence *Albert Einstein, a German theoretical physicist, won the Nobel Prize* would become (*Albert Einstein, won, the Nobel Prize*).

A graph is constructed using the triples by representing subjects and objects as nodes connected by predicates as directed edges. For example, the triple would become *Albert Einstein* $\xrightarrow[\text{won}]{}$ *the Nobel Prize*. Nodes and edges have a *name* property that is the text they represent. They also have a *weight* property that denotes the number of times that node or edge has appeared. For example, in Figure 1, the node with name *Albert Einstein* has weight 4 and edge with name *won* has weight 2.

Merging Nodes and Edges When subsequent triples are added to the graph, they are merged with the existing graph if they already exist to reduce information replication. To merge nodes, the TF-IDF overlap of the new node's name is calculated with the existing graph node names, and the new node is merged into an existing node if the TF-IDF is higher than some threshold (see Figure 2, steps 2 and 3 for example merge operations). Edges are merged similarly with existing edges between the same two nodes. Such merge operations allow strings such as *the Nobel Prize* and *Nobel Prize* to be represented as one node rather than separately. Similarly, coreference resolution aids in merging — by identifying that *Albert Einstein* and *He* refer to the same entity and thus merging them, the construction of the graph reduces redundancy. The size of the graph can be modified by controlling which triples are added using TF-IDF overlap (see Figure 2, step 4). TF-IDF overlap of the triple with the question can be used to determine if the triple contains relevant information. This improves robustness to noisy web search input and helps filter entirely irrelevant portions, such as scraped HTML tags.

[1]We use the implementation available here: `https://github.com/huggingface/neuralcoref`

[2]We use the implementation available here: `https://github.com/gabrielStanovsky/supervised-oie`

4 Modeling Graphs as Sequences

Current models for text generation often use Seq2Seq architectures such as the Transformer (Vaswani et al., 2017). These models are designed to encode sequences rather than graphs. We describe now how to convert a graph into a structured input sequence. Our complete model will take as input a linearized graph by encoding graph attributes such as node and edge weight as embeddings. We add hierarchical and memory-compressed attention mechanisms to scale Seq2Seq models to encode the full graph and attend to the most relevant information within it (Figure 4), and finally we integrate the benefits of language modeling using multi-task training.

4.1 Graph to Sequence

Linearization To represent the graph as a sequence for Seq2Seq, it is linearized into a structured standard form of subject node, object node, and predicate edge, separated by indicator tokens, as shown in Figure 1. For example, the linearization *<sub> Albert Einstein <obj> the Nobel Prize <pred> won* would be created. The linearization is accomplished through graph traversal in a breadth-first manner following the directed edges formed by predicates and starting from the node with the largest weight as the root. For two nodes that are connected by multiple predicates, the predicates are concatenated (shown in Figure 1), so a linearization such as *<pred> won <s> received* would indicate that Albert Einstein both won and received the Nobel Prize.

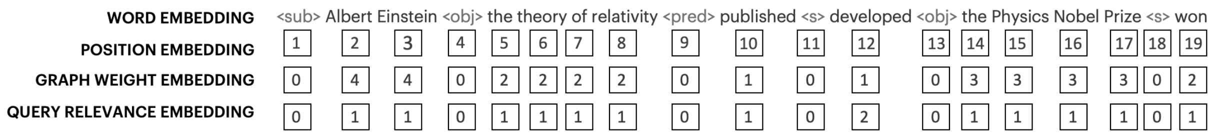

WORD EMBEDDING <sub> Albert Einstein <obj> the theory of relativity <pred> published <s> developed <obj> the Physics Nobel Prize <s> won
POSITION EMBEDDING 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19
GRAPH WEIGHT EMBEDDING 0 4 4 0 2 2 2 2 0 1 0 1 0 3 3 3 3 0 2
QUERY RELEVANCE EMBEDDING 0 1 1 0 1 1 1 1 0 1 0 2 0 1 1 1 1 0 1

Figure 3: **Graph Attribute Embeddings**. In addition to word and position embeddings, models receive a Graph Weight embedding to encode node and edge weight and a Query Relevance embedding that encodes search result rank.

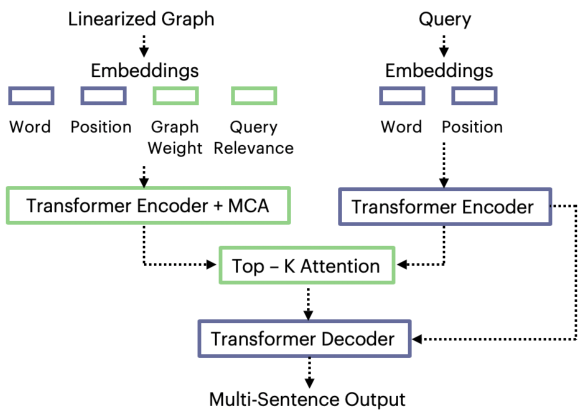


Figure 4: **Model Architecture**. Gray indicates standard Transformer elements, Green indicates modification.

Encoding Graph Information Transformer Seq2Seq models have two embeddings: a word embedding and a position embedding. Simply linearizing the graph, however, loses attribute information such as node and edge weight. Instead, we encode these attributes as embeddings in addition to the word and position embeddings.

To represent *Graph Weight* (GW), node and edge weight is provided as an embedding for each token. The node weight and edge weight are equivalent to the number of merge operations + 1. For example, if *Albert Einstein* occurred 4 times in the text, the GW embedding for the tokens *Albert* and *Einstein* would be 4, as shown in Figure 3.

We encode a *Query Relevance* (QR) embedding to represent the relevance of the web search to the query as ranked by the information retrieval system (e.g. search engine). Information from the top web search results is likely more relevant than information from the last web search results, so providing this embedding allows the model to distinguish between these different information sources. In Figure 3, tokens representing sentences from the first document have QR embedding 1, and tokens from the second document have value 2.

Models now have access to several different types of embeddings, but all embedding information contributes equally as there is no mechanism to distinguish between them. We introduce a mechanism for models to scale the graph embeddings. We denote the embedding for position t as e_t, such that e_t^{word} is the word embedding.

For the GW embedding, models learn a gating function g based on the word and GW embeddings. Such a mechanism provides capacity for the model to decide when the additional embeddings are useful based on the words in the input. The gate is calculated by applying an MLP W to the concatenation of the word and GW embeddings. The learned gate is then applied to GW embeddings to create the output h:

$$g_t^{\text{GW}} = W[e_t^{\text{GW}}; e_t^{\text{word}}]$$
$$h_t^{\text{GW}} = g_t^{\text{GW}} \circ e_t^{\text{GW}}$$

Models learn a gating mechanism for the QR embedding in a similar manner. The full embedding the model receives is as follows:

$$e_t^{\text{word}} + e_t^{\text{pos}} + [h_t^{\text{GW}}; h_t^{\text{QR}}]$$

4.2 Hierarchical Attention

One challenge in modeling long sequences is that attention mechanisms struggle to make sharp selections when softmax-ing over long sequences (Fan et al., 2018). When attention is blurry, there lacks a strong distinction between noise and relevant information.

We assume that graphs are constructed from query-based web search input and thus leverage this query to learn a subselection operation using *hierarchical top-k attention*, depicted in Figure 4. The query is encoded with a Transformer encoder and the linearized graph with another Transformer encoder. We model the interaction between the query and the input sequence (e.g. web search results or linearized graph) by computing an attention distribution between the question and the input, then selecting the top k positions with the most attention weight. Such a mechanism can be thought of as building a query-dependent representation of the most relevant knowledge, which is commonly done in question answering architectures like BiDAF (Seo et al., 2017). The top k

operation limits the number of tokens, making the attention mechanism sharper.

4.3 Scaling to Encode the Graph

Recent progress has improved the ability of language models to process longer sequences (Sukhbaatar et al., 2019; Dai et al., 2019), but models remain limited in their capacity to encode long documents. The multi-document results of query-based web search have hundreds of thousands of tokens, beyond the limit of current Seq2Seq models to handle. For example, the ELI5 dataset provides an average of 200K tokens of web search input. However, by compressing the web search results into a knowledge graph, we significantly reduce the number of tokens by an order of magnitude and make it possible for a model to access the entirety of the search information.

To represent the full graph, models must scale to encode around 10K input tokens. The attention mechanism in Transformer architectures becomes computationally expensive for sequences of this length. Instead, we experiment with the *Memory-Compressed Attention* (MCA) mechanism proposed for language models in (Liu et al., 2018b)[3] and apply it to the encoder side of Seq2Seq models. At each self-attention layer, MCA alternates between (1) local attention, computed between smaller chunks of tokens and (2) strided convolutions to reduce the number of keys and values used in attention computation. By adding the MCA mechanism to the encoder (*E-MCA*), we are able to encode the complete linearized graph.

4.4 Multi-tasking with KB Completion

Fan et al. (2019) used multi-task training on language modeling and various Seq2Seq tasks to incorporate the benefits of language modeling in Seq2Seq models. We extend this by training additionally on *knowledge graph completion*. Models receive at training time sequences of a linearized graph with nodes, edges, or both selectively masked and must predict the missing content words. For example, models might receive as input *<sub> Albert Einstein <obj> [mask] <pred> won* and need to predict *the Nobel Prize*. This can be seen as both a multi-word extension of the masked language model training proposed in (Devlin et al., 2018) and as learning the task of knowledge base completion (Lacroix et al., 2018; Bordes et al., 2011). At training time, the tasks are distinguished using an indicator token in the input.

[3]In (Liu et al., 2018b), the mechanism is termed *DMCA* as it is applied on the decoder side

5 Experimental Setup

We evaluate our approach on two datasets with multi-document web input and multi-sentence abstractive output. We use Seq2Seq models that leverage a query concatenated with web search results that have been processed into a supporting document — e.g. TF-IDF subselection, linearized graph, etc. — to generate long output.

5.1 Datasets and Evaluation

ELI5 First, we experiment with the *Explain Like I'm Five* (ELI5) (Fan et al., 2019) question answering dataset of 270K complex questions paired with multi-sentence, explanatory answers (130 words on average). To facilitate question answering, the dataset provides the top 100 web search hits from querying the question, which results in 200K words on average. See Appendix for examples.

Previous work (Fan et al., 2019) used TF-IDF to find sentences in the web search that have the largest overlap with the question and created a *TF-IDF extraction* of about 850 words as input for Seq2Seq models. Instead, we construct a local knowledge graph for each question from the 100 web search hits. Following the average length of the TF-IDF support document constructed in (Fan et al., 2019), we experiment with modeling the first $N = 850$ tokens of the linearized graph, then scale to encode the entire graph using E-MCA.

WikiSum Second, we experiment on the *WikiSum CommonCrawl* (Liu et al., 2018b) summarization dataset[4] with 1.5 million examples. This task formulates Wikipedia lead paragraph generation as a multi-document summarization problem, where the paragraph must be generated using the cited article references and other queried content from web search. The query used is the title of the Wikipedia article. See Appendix for examples.

Previous work (Liu et al., 2018b) applied *TF-IDF Ranking* to order the paragraphs of web search given a query. Models receive the re-ordered paragraphs ranked by TF-IDF as input. Liu et al. (2018b) model the first $N = 500$ words of this re-ranking and then $N = 11,000$ using

[4]https://github.com/tensorflow/tensor2tensor/tree/master/tensor2tensor/data_generators/wikisum

MCA. We construct the knowledge graph for each Wikipedia article from the first 200K words of the ranked web search results[5], and experiment with encoding 500 and 11,000 tokens.

Evaluation Both tasks evaluate the multi-sentence generation output against the gold output using F1 ROUGE. On WikiSum, we report only ROUGE-L following (Liu et al., 2018b).

We conduct a comparative human evaluation on the ELI5 dataset. We use crowdworkers to examine the responses of two models on 300 different questions from the test set. For each question, 3 evaluators are shown two answers and asked to choose the one they prefer. To reduce variance, answers are standardized at 150 words each.

5.2 Training and Generation

To reduce the vocabulary size of varied web document content, we apply byte-pair encoding (Sennrich et al., 2016) to generate 40K codes for each dataset. We implement our models in fairseq-py (Ott et al., 2019) using the Transformer Big architecture and training schedule described in (Vaswani et al., 2017). Detailed parameters are listed in the Appendix. For generation, we use beam search with beam size 5 and tune a minimum and maximum length on the validation set.

5.3 Baselines

We compare our results to the Transformer sequence models presented in (Fan et al., 2019) for ELI5 and (Liu et al., 2018b) for WikiSum.

We evaluate three additional baseline models:

- *Sentence Selection with Maximal Marginal Relevance:* (Fan et al., 2019) used TF-IDF to identify relevant sentences in the web documents to form a support document of around 850 words. However, recent work (Fabbri et al., 2019) has shown that using maximal marginal relevance is an effective strategy for selecting relevant information while reducing redundancy. We explore using MMR to select sentences from the web text to concatenate to form a support document.
- *Seq2Seq Multi-task Triples:* To examine the impact of solely restructuring the input into Open IE Triples but not leveraging graph construction to reduce redundancy, we experiment with a Triples Only baseline. The triples are concatenated to form the input.
- *Seq2Seq Multi-task Top 20 Triples:* As an alternative to using graph construction to compress the full set of Open IE Triples, we explore using TF-IDF overlap with the query to find the most relevant information. We select the top 20 triples to concatenate as input.

[5]Average length of provided web input is around 50K words, and maximum length is around 900K words

Model	Input Length	ROUGE 1	2	L
Q + D to A*, TF-IDF	avg 850	28.3	5.1	22.8
Q + D to A, MMR	avg 850	28.1	5.0	22.9
Multi-task*	avg 850	28.9	5.4	23.1
Multi-task Triples	850	29.0	5.2	23.2
Multi-task Top20 Trip.	avg 570	28.8	5.3	23.2
Q + D to A Graph	850	28.8	5.3	23.3
Multi-task Graph	850	29.5	5.6	23.6
+ Top-100 Attention	850	29.7	5.7	23.8
+ E-MCA	11K	30.0	5.8	24.0

Table 1: **Answer Generation on ELI5** using Seq2Seq models receiving the **Q**uestion and a support **D**ocument (e.g. TF-IDF selection, Triples, Linearized Graph) to produce the **A**nswer. * denotes results from (Fan et al., 2019).

Model	InputLen	ROUGE-L
T + D to P*	500	34.2
LM + D-MCA*	11K	36.2
T + D to P	500	33.8
Multi-task	500	34.4
Multi-task Graph	500	34.9
+ Top-100 Attention	500	35.2
+ E-MCA	11K	36.5
LM + D-MCA + MoE-256*	7.5K	38.8

Table 2: **Lead Paragraph Generation on WikiSum CommonCrawl** using Seq2Seq models receiving the **T**itle and a support **D**ocument (e.g. TF-IDF ranking, Linearized Graph) to produce the Lead **P**aragraph. * denotes results from (Liu et al., 2018b) that use data scraped from unrestricted web search, not the static CommonCrawl version.

6 Results

We examine the performance of our proposed approach and the choices made in graph construction and modeling. We analyze the quality of the compression created by graph construction and the robustness and interpretability of this process.

6.1 Linearized Graph Improves Performance

In Table 1, we compare our methods to various baselines on the ELI5 dataset. Using MMR to select the most relevant non-redundant input is similar to the TF-IDF baseline from Fan et al. (2019).

The Seq2Seq Multi-task Triples baseline standardizes the input by forming triples but does not remove redundant triples. It produces marginally better results compared to the baseline Multi-Task model. Sub-selecting to the Top 20 Triples is harmful, as similar text has high TF-IDF overlap with the query so redundant information is selected. Creating the graph structure brings an improvement of around 0.6 ROUGE-1.

Similar trends are seen for the WikiSum dataset in Table 2, where graph construction improves the Multi-task model by 0.5 ROUGE-1. These improvements are statistically significant at the 95% confidence level.

For both datasets, a further improvement is seen by using the hierarchical attention mechanism to attend to only the most relevant information in the linearized graph input. For ELI5, it brings an additional 0.2 ROUGE-1 improvement and on WikiSum a 0.3 ROUGE-1 improvement.

By using MCA to scale Seq2Seq models to encode the entire graph, further gains can be seen. Particularly in information synthesis tasks, prior work has shown the importance of reading more information. Liu et al. (2018b) achieved a 2-point ROUGE improvement by reading 11K tokens instead of 500. In our setting, E-MCA improves our results around 0.3 ROUGE on ELI5 and 1.3 ROUGE on WikiSum. We display random generations from both datasets in the Appendix.

We use human evaluation to compare the Multi-task baseline to the Multi-task Graph + Top-k Attention model. **57.4%** of evaluations prefer the Multi-task Graph + Top-k Attention model. We conduct a two-tailed binomial test and find this result is statistically significant with $p = 0.003$.

6.2 Analysis of Modeling Choices

Ablation on Model Components Table 3(a) sequentially removes the graph embeddings, the knowledge-base completion multi-tasking, and the multi-tasking from (Fan et al., 2019) and reveals that each of these is important for performance.

Graph Attribute Embeddings Table 3(b) displays the effect of removing the graph attribute embeddings and gating mechanism. Removing each is slightly harmful, and the combination of all three together provide the best performance.

More Web Search Documents Figure 7 (right) shows that graph construction with more web

Model	**ROUGE-1**
(a) Iterative Removal of Model Components	
Multi-task Graph	29.4
- Graph Embeddings	29.1
- KB Completion Multi-tasking	28.9
- LM Multi-tasking from (Fan et al., 2019)	28.4
(b) Removing Graph Embedding Components	
Graph	
+ Gated Graph Weight + Query Relevance	28.6
No Graph Weight Embedding	28.4
No Query Relevance Embedding	28.3
No Gating	28.4
(c) Varying Number of Hits in Graph	
Multi-task Graph + Top-k Attention + E-MCA	
with Graph on 5 Search Hits	28.8
with Graph on 10 Search Hits	29.3
with Graph on 50 Search Hits	29.6
with Graph on 100 Search Hits	29.9
(d) Varying k in Hierarchical Top-k Atttention	
Multi-task Graph + E-MCA +	
Top-$k = 50$	29.1
Top-$k = 100$	29.5
Top-$k = 250$	29.4
Top-$k = 500$	29.3
Top-$k = 1000$	29.2

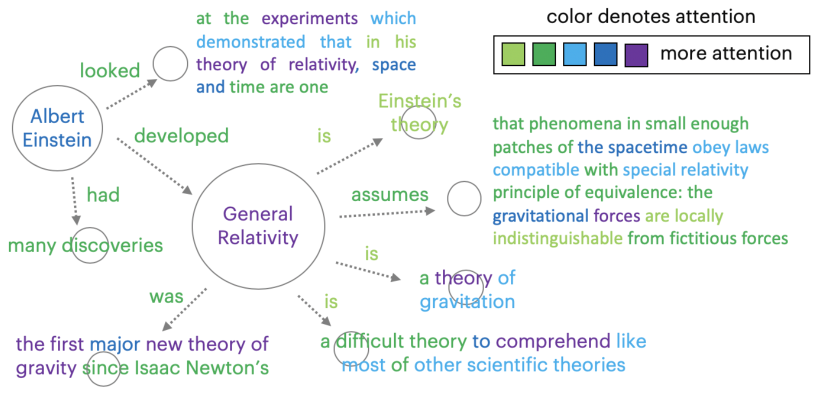
Table 3: **Ablations** on the ELI5 Validation Set

Model	**Input**	**ROUGE-1**
Seq2Seq Q + D to A	TF-IDF	28.3
Seq2Seq Q + D to A	Web	25.9
+ Shuffle	Web Shuffle	24.4
Seq2Seq Q + D to A Graph	Web	28.5
+ Shuffle	Web Shuffle	28.2

Table 4: Importance of **Web Search Relevance** on Validation for ELI5, modeling 850 input words.

search information is important for answer token coverage. The graph on the top search hit alone is missing 64% of the answer tokens, but this decreases as more search hits are used. Table 3(c) indicates that this lack of coverage of the answer tokens correlates with generation quality. Models receiving a graph built on the first 5 search hits alone are substantially worse than all 100 hits.

Top-k Attention Table 3(d) shows the effect of the Top-k Hierarchical Attention mechanism for various values of k. Attending to too many tokens lowers ROUGE — for the ELI5 task of writing approximately 130 word answers, attending to 1000 input tokens likely means the model is focusing on irrelevant information and 50 tokens is too few.



at the experiments which demonstrated that in his theory of relativity, space and time are one
looked
Albert Einstein
developed
had
many discoveries
General Relativity
was
the first major new theory of gravity since Isaac Newton's
is
Einstein's theory
assumes
that phenomena in small enough patches of the spacetime obey laws compatible with special relativity principle of equivalence: the gravitational forces are locally indistinguishable from fictitious forces
a theory of gravitation
a difficult theory to comprehend like most of other scientific theories
color denotes attention
🟩🟩🟦🟦🟪 more attention

Question: *Can someone finally explain the theory of general relativity?*

Generated Answer: It's a bit complicated to explain. General relativity is a theory of Albert Einstein. Einstein developed this theory. Relativity is measure of how fast something moves relative to the reference frame. In theory, the speed of light is constant. If you are traveling at a constant speed, and then you move at a speed relative to an observer, there is a constant rate for that object to move. General relativity is observed in the experiments of Albert Einstein. It is about space, and space, and gravity.

Figure 5: **Interpretable Attention** of Seq2Seq models on a subgraph when answering a question in ELI5

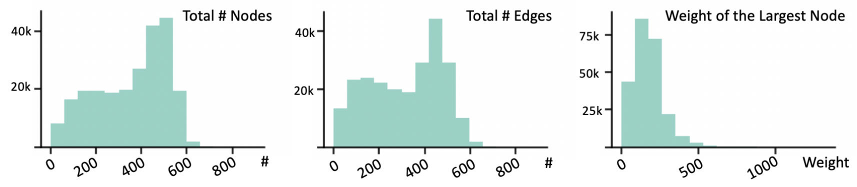


Figure 6: (Left) Distribution of Number of **Nodes**, (Middle) Number of **Edges**, (Right) **Weight** of the Largest Node in graph construction on the ELI5 training set.

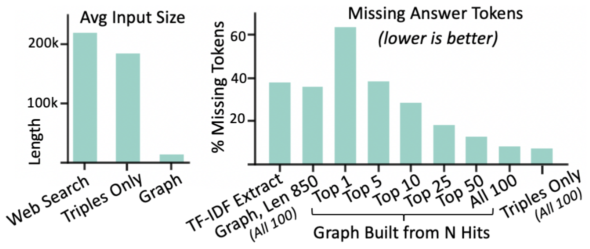


Figure 7: (Left) Graph construction drastically reduces input size by an order of magnitude. (Right) Graph construction encodes more tokens present in the answer compared to TF-IDF extraction and building the graph from more search hits increases answer token coverage. Analysis on ELI5 for both plots.

6.3 Graph Improves Answer Token Coverage Despite Compression

Figure 6 displays the distribution of the number of nodes, edges, and the largest node weight for each local graph built on the ELI5 dataset. The 100 web search results are compressed to a few hundred nodes. By merging redundancy and trimming irrelevant triples from the graph, the input is reduced by an order of magnitude (Figure 7, left).

Despite compression, the graph retains more answer tokens than TF-IDF subselection. Figure 7 (right) displays the percentage of answer tokens not present in the input. The TF-IDF Extraction from (Fan et al., 2019) is missing 38% of tokens. The graph constructed on all 100 web search results is only missing 8.7% of tokens, but has around 10K words. When analyzing just the first 850 tokens to match the average length of the TF-IDF extraction, the graph is better (only missing 35% of tokens). Further, the merging and discarding operations done during graph construction do not have a large effect on answer token coverage — the full set of triples marginally reduces the percentage of answer tokens missing to 7.3% instead of 8.7%. This indicates that much of the information in the full set triples is redundant and unnecessary for good answer token coverage.

6.4 Graph Representation is More Robust to Poor Search Relevance Ordering

We analyze the robustness of our approach to the ordering of web search results in Table 4. Instead of constructing the graph from the first web search result to the last, we shuffle the web search results and construct the graph on this shuffled input. We compare this to modeling the web search results directly (no TF-IDF retrieval) and a model that receives this shuffled web search input. The graph is more robust to shuffling — as more information can be encoded in the graph due to its compression effect, the search hit ordering is less critical.

6.5 Interpretable Attention on Subgraphs

Figure 5 shows an example of the nodes and edges the model focuses upon most when answering a question on ELI5. To construct this visualization, we calculate the top nodes the model attends to and then their top edges. The model attention on a sub-portion of the linearized input can be visualized as an interpretable graph that corresponds well to the model's generated answer. For example, the relationship *General Relativity* $\underset{\text{is}}{\rightarrow}$ *Einstein's theory* becomes the generated sentence *General Relativity is a theory of Albert Einstein*.

7 Conclusion

Many open-domain NLP tasks rely upon multi-document input from the web to facilitate tasks such as answering questions or writing summaries, but current approaches struggle to encode the entirely of this information. We propose constructing one knowledge graph per query and show that this method compresses information and reduces redundancy. We show on two abstractive generation tasks that using the linearized graph achieves better performance than TF-IDF retrieval.

References

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. 2007. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer.

Junwei Bao, Nan Duan, Ming Zhou, and Tiejun Zhao. 2014. Knowledge-based question answering as machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 967–976.

Regina Barzilay, Kathleen R McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics*.

Tal Baumel, Matan Eyal, and Michael Elhadad. 2018. Query focused abstractive summarization: Incorporating query relevance, multi-document coverage, and summary length constraints into seq2seq models. *arXiv preprint arXiv:1801.07704*.

Daniel Beck, Gholamreza Haffari, and Trevor Cohn. 2018. Graph-to-sequence learning using gated graph neural networks. *arXiv preprint arXiv:1806.09835*.

Taylor Berg-Kirkpatrick, Dan Gillick, and Dan Klein. 2011. Jointly learning to extract and compress. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*, pages 481–490. Association for Computational Linguistics.

Lidong Bing, Piji Li, Yi Liao, Wai Lam, Weiwei Guo, and Rebecca J Passonneau. 2015. Abstractive multi-document summarization via phrase selection and merging. *arXiv preprint arXiv:1506.01597*.

Antoine Bordes, Jason Weston, Ronan Collobert, and Yoshua Bengio. 2011. Learning structured embeddings of knowledge bases. In *Twenty-Fifth AAAI Conference on Artificial Intelligence*.

Joan Bruna, Wojciech Zaremba, Arthur Szlam, and Yann LeCun. 2013. Spectral networks and locally connected networks on graphs. *arXiv preprint arXiv:1312.6203*.

Jürgen Bückner, Martin Pahl, O Stahlhut, and C-E Liedtke. 2002. A knowledge-based system for context dependent evaluation of remote sensing data. In *Joint Pattern Recognition Symposium*, pages 58–65. Springer.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2017. Improving multi-document summarization via text classification. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open-domain questions. In *ACL*.

Danqi Chen, Richard Socher, Christopher D Manning, and Andrew Y Ng. 2013. Learning new facts from knowledge bases with neural tensor networks and semantic word vectors. *arXiv preprint arXiv:1301.3618*.

Christopher Clark and Matt Gardner. 2017. Simple and effective multi-paragraph reading comprehension. *arXiv preprint arXiv:1710.10723*.

Kevin Clark and Christopher D Manning. 2016a. Deep reinforcement learning for mention-ranking coreference models. *arXiv preprint arXiv:1609.08667*.

Kevin Clark and Christopher D Manning. 2016b. Improving coreference resolution by learning entity-level distributed representations. *arXiv preprint arXiv:1606.01323*.

Wanyun Cui, Yanghua Xiao, Haixun Wang, Yangqiu Song, Seung-won Hwang, and Wei Wang. 2017. Kbqa: learning question answering over qa corpora and knowledge bases. *Proceedings of the VLDB Endowment*, 10(5):565–576.

Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. *arXiv preprint arXiv:1901.02860*.

Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. 2018a. Multi-step retriever-reader interaction for scalable open-domain question answering.

Rajarshi Das, Tsendsuren Munkhdalai, Xingdi Yuan, Adam Trischler, and Andrew McCallum. 2018b. Building dynamic knowledge graphs from text using machine reading comprehension. *arXiv preprint arXiv:1810.05682*.

Hal Daumé III and Daniel Marcu. 2002. A noisy-channel model for document compression. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 449–456. Association for Computational Linguistics.

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

Giuseppe Di Fabbrizio, Amanda Stent, and Robert Gaizauskas. 2014. A hybrid approach to multi-document summarization of opinions in reviews. In *Proceedings of the 8th International Natural Language Generation Conference (INLG)*, pages 54–63.

Alexander R Fabbri, Irene Li, Tianwei She, Suyi Li, and Dragomir R Radev. 2019. Multi-news: a large-scale multi-document summarization dataset and abstractive hierarchical model. *arXiv preprint arXiv:1906.01749*.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. 2019. Eli5: Long form question answering. In *Proceedings of ACL 2019*.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. *arXiv preprint arXiv:1805.04833*.

Kavita Ganesan, ChengXiang Zhai, and Jiawei Han. 2010. Opinosis: A graph based approach to abstractive summarization of highly redundant opinions. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 340–348.

Dan Gillick and Benoit Favre. 2009. A scalable global model for summarization. In *Proceedings of the Workshop on Integer Linear Programming for Natural Langauge Processing*, pages 10–18. Association for Computational Linguistics.

Mohamad Ali Honarpisheh, Gholamreza Ghassem-Sani, and Ghassem Mirroshandel. 2008. A multi-document multi-lingual automatic summarization system. In *Proceedings of the Third International Joint Conference on Natural Language Processing: Volume-II*.

Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. 2017. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. In *ACL*.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Timothée Lacroix, Nicolas Usunier, and Guillaume Obozinski. 2018. Canonical tensor decomposition for knowledge base completion. *arXiv preprint arXiv:1806.07297*.

Yuxuan Lai, Yansong Feng, Xiaohan Yu, Zheng Wang, Kun Xu, and Dongyan Zhao. 2019. Lattice cnns for matching based chinese question answering. *arXiv preprint arXiv:1902.09087*.

Fei Liu, Jeffrey Flanigan, Sam Thomson, Norman Sadeh, and Noah A Smith. 2018a. Toward abstractive summarization using semantic representations. *arXiv preprint arXiv:1805.10399*.

Peter J. Liu, Mohammad Saleh, Etienne Pot, Ben Goodrich, Ryan Sepassi, Lukasz Kaiser, and Noam Shazeer. 2018b. Generating wikipedia by summarizing long sequences. In *ICLR*.

Vincent Ng and Claire Cardie. 2002. Improving machine learning approaches to coreference resolution. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 104–111. Association for Computational Linguistics.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Marco Rospocher, Marieke van Erp, Piek Vossen, Antske Fokkens, Itziar Aldabe, German Rigau, Aitor Soroa, Thomas Ploeger, and Tessel Bogaard. 2016. Building event-centric knowledge graphs from news. *Journal of Web Semantics*, 37:132–151.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *ACL*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Bidirectional attention flow for machine comprehension. In *ICLR*.

Gabriel Stanovsky, Julian Michael, Luke Zettlemoyer, and Ido Dagan. 2018. Supervised open information extraction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 885–895.

Sainbayar Sukhbaatar, Edouard Grave, Piotr Bojanowski, and Armand Joulin. 2019. Adaptive attention span in transformers.

Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W Cohen. 2018. Open domain question answering using early fusion of knowledge bases and text. *arXiv preprint arXiv:1809.00782*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NIPS*.

Piek Vossen, Tommaso Caselli, and Yiota Kontzopoulou. 2015. Storylines for structuring massive streams of news. In *Proceedings of the First Workshop on Computing News Storylines*, pages 40–49.

Rachel Wities, Vered Shwartz, Gabriel Stanovsky, Meni Adler, Ori Shapira, Shyam Upadhyay, Dan Roth, Eugenio Martínez-Cámara, Iryna Gurevych, and Ido Dagan. 2017. A consolidated open knowledge representation for multiple texts. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 12–24.

Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, Michael Witbrock, and Vadim Sheinin. 2018. Graph2seq: Graph to sequence learning with attention-based neural networks. *arXiv preprint arXiv:1804.00823*.

Michihiro Yasunaga, Rui Zhang, Kshitijh Meelu, Ayush Pareek, Krishnan Srinivasan, and Dragomir Radev. 2017. Graph-based neural multi-document summarization. *arXiv preprint arXiv:1706.06681*.

Zhiping Zheng. 2003. Question answering using web news as knowledge base. In *Proceedings of the tenth conference on European chapter of the Association for Computational Linguistics-Volume 2*, pages 251–254. Association for Computational Linguistics.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. 2018. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*.