TableLingo

A Framework for Controlled Table-to-Text Generation

Ashish Bharadwaj Srinivasa

October 2020

CS224U Natural Language Understanding

Literature Review

Introduction

The task of data-to-text generation involves converting structured data, typically in tables or knowledge bases, into natural language text. Data-to-Text has become a critical driver of the AI revolution of the recent years as it actively helps address NLP problems such as question answering, chat bots (dialog), machine translation, etc. As huge amounts of data in businesses is generally stored in large structured databases, extending natural language generation techniques to them requires carefully fine tuned algorithms. Moreover, the databases can be from a wide variety of domains and thus finding generic techniques that work well becomes a challenging problem.

There are two major ways to approach the problem - pipeline and end-to-end based. In the pipeline based models, the table-to-text generation task is broken down into multiple linked smaller tasks such as content selection, lexicalization, text ordering, etc. In recent years there has been a trend towards more end-to-end approaches that take a table as input and converts it to text directly. Two of the more recent end-to-end generation techniques involve using raw highlighted table cells or using semantic triples extracted from tables as the input for natural language generation algorithms. In 2020 there have been multiple popular datasets published for these two approaches - DART [1] and WebNLG2020 [2] contain triples to text examples and ToTTo [3] contains examples of highlighted table cells to text. The datasets come with new text generation challenges of their own and provide baseline metrics alongside performance of current state-of-the-art models.

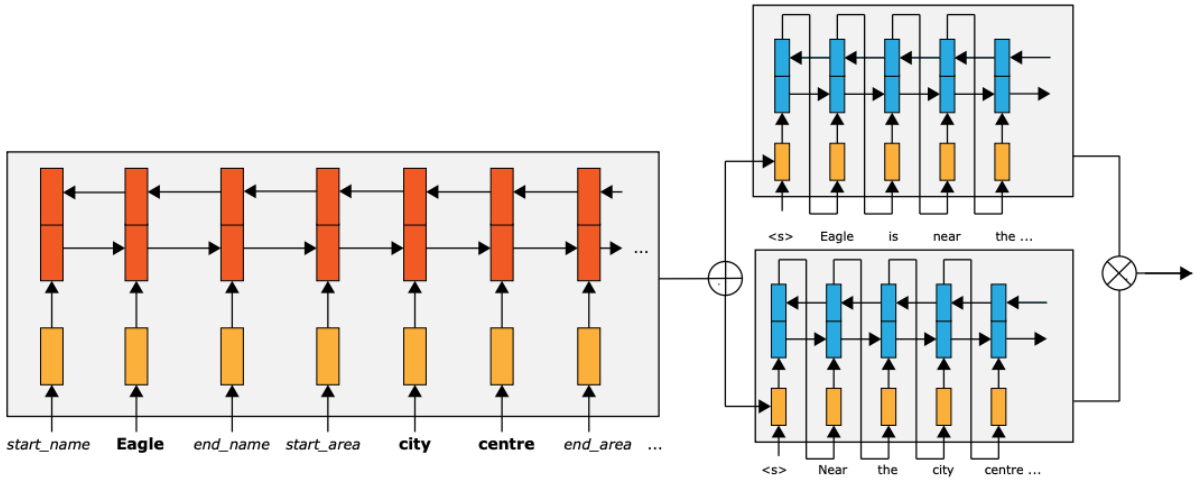
In the following sections, we will dive deeper into the popular approaches to tackling data-to-text generation, state of the art models and from them infer potential future work. We compare and contrast the approaches using BLEU, METEOR and TER metrics which are popularly used for text generation tasks and are also used for the WebNLG challenge.

Related Work

(Gehrmann et al., 2018) End-to-End Content and Plan Selection for Data-to-Text Generation

Gehrmann et al. [5] introduced the idea of using Pointer-Generator networks [11] for table-to-text generation. While it was originally invented for text summarization, it has become popular for data-to-text tasks in the recent years as well. The main idea behind pointer-generator networks is a conditional copy mechanism that allows the decoder to copy tokens from the source sentence. At each output time-step, the decoder makes the decision to either copy a token from source or to generate a new text token. While still being a strictly seq2seq model, the copy mechanism allowed them to easily add text found in the tables into the generated sentence which was difficult earlier due to their unseen (out-of-vocabulary) nature. In the table-to-text setting, they found that enforcing a much harder criteria for copying all text from tables also works really well, as the task demands that the generated sentence must contain information from all the selected table cells.

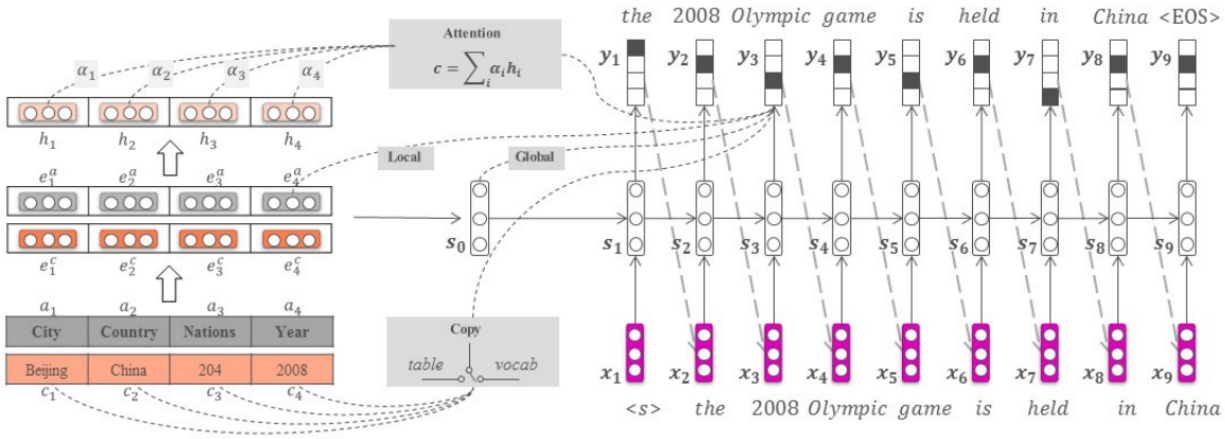
The other important problem that was tackled in this work is the challenge posed by differing sentence templates (active vs passive, for example) in the training labels. This generally forced the models to not be able to generate either sentence. Gehrmann et al., tackle this by learning an ensemble of decoder models each learning a different sentence template. Each model only trains on a subset of the data that was carefully split based on the sentence structure.



The experiments were conducted on the E2E NLG Challenge dataset that contained about 50,000 samples of meaning representation and their corresponding sentences. The models used a simple two-layer bidirectional LSTM. The results show that diverse decoder ensembling provides an efficient way to handle noisy training data. Additionally, the copy mechanism enforced via a coverage penalty works well to copy over the source table text to the generated sentence producing top BLEU scores of 74.3 after fine tuning the hyperparameters.

(Bao et al., 2018) Table-to-Text: Describing Table Region with Natural Language

Bao et al. [6] tackle the more general problem of table-to-text using source tables as the input, compared to the previous approach, which used meaning representations as the input. They also introduce a new open domain dataset called WikiTableText contains 4,692 tables and 13,318 sentences. This work extends the idea of pointer generator networks with copying mechanism to raw table inputs. The major contribution of this paper is the table aware encoder, that tries to capture the rich semantic information available in a table. The work introduces the idea of using a combination of embedding vectors for the table attributes and cell values. Different embedding vectors are learned for each as they represent different types of information. Each cell is represented by an aggregated value of its associated column attribute embedding and the cell value embedding. This approach allows us the use and fine-tuning of pre-trained embedding models as well and helps build more generalized encoder models that still capture a rich feature representation.



The models also use a GRU based architecture for both the encoder and decoder. The table also adds local and global table information to the models. The global information captures information common to the entire table that can be used by the model to differentiate the type of source tables, domains, table name, etc. The local information maps each table entry to a generated word using the above mentioned attention mechanism. The local and global table information is combined with the copy mechanism in the decoder step to produce sentence tokens. The combination of these produces high BLEU scores of 40.26 on the WikiBio dataset proving the importance of these techniques for domain generalization.

(Ferreira et al., 2019) Neural data-to-text generation: A comparison between pipeline and end-to-end architectures

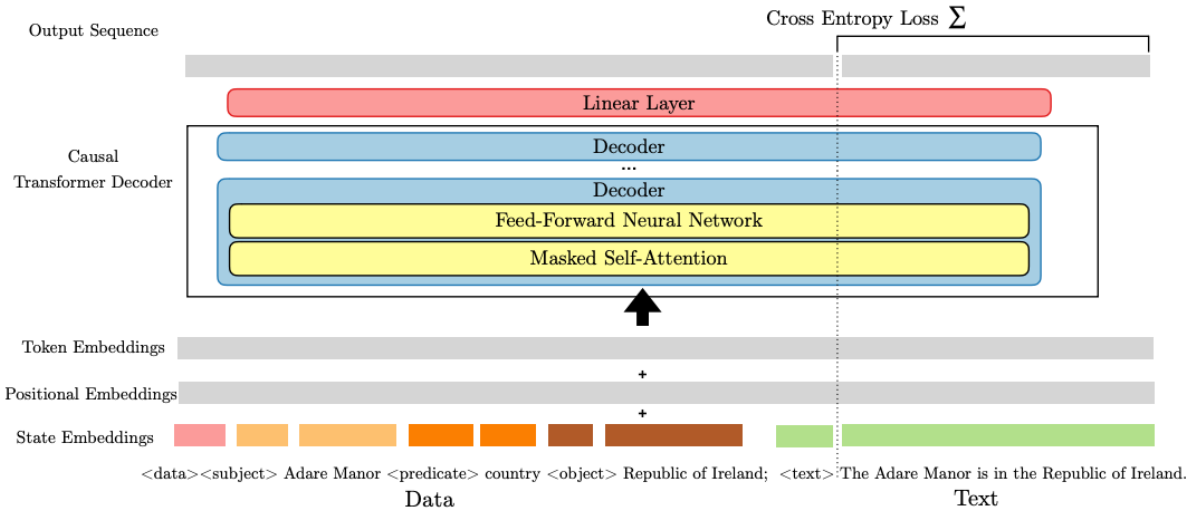
Ferreira et al. [7] compares the pipeline based data-to-text approaches to end-to-end based approaches for the task of generating text from semantic RDF triples. For the pipeline architecture, the work constructs a sequence of 5 tasks. Firstly, the input triple collection undergoes discourse ordering to convert them to an order in which they should be verbalized. Next the ordered triple sequences are lexicalized, or converted to

the appropriate text tokens in the generated sentence. Then the input goes through Referring Expression Generation which generates references in the target sentence to the source table text. Finally, the generated text undergoes textual realization for adjusting verbs, determiners and implications.

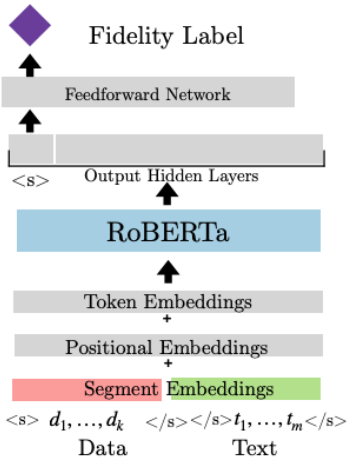
The end-to-end models aim to convert semantic triples into text in a single step. The model architecture makes use of the techniques popularized by Neural Machine Translation Models by combining GRU and transformer architectures. The model avoids explicit intermediate steps. The GRU based models performed best on both the pipeline and end-to-end approaches. In the pipeline approach, GRU based models performed better at the task of ordering non-linguistic input, whereas the transformer based architectures performed better at converting them to text. They also showed the challenges with the end-to-end approaches when it comes to generalizing to new domains. The models also found it difficult to capture non-linguistic information from the source tables in the generated text. Finally, they found that the UPFFORGe models performed the best with BLEU scores on 60.59 on the WebNLG2017 test set. This work shows the trade off between pipeline based approaches with end-to-end approaches, surfacing the challenges with each and leading the path for more investment in GRU and transformer based approaches for table-to-text generation.

(Harkous et al., 2020) Have Your Text and Use It Too! End-to-End Neural Data-to-Text Generation with Semantic Fidelity

Harkous et al. [9] tackles the approach of end-to-end data-to-text generation by trying to address the challenges it faces with new domains and semantically consistent generation. They introduce a two stage approach of generation and reranking which combines a fine tuned language model with a semantic fidelity classifier. For text generation, the work uses a pre-trained GPT2 language model fine tuned on the task. The language model consists of a stack of transformer decoder blocks with multi-headed attention. The data from the table/triples is concatenated with the text and fed as input into the language models. Since GPT uses byte-pair encoding as opposed to characters, it can easily extent to a really large vocabulary of text tokens. The work also introduces the idea of state embeddings that are used with input and positional embeddings in the model.



Secondly, the work introduces the idea of a semantic fidelity classifier to tackle the challenge of generating semantically sound sentences. This enforces that the generated text does not miss any part of the input data while also not adding additional information. This task follows the ideas of Natural Language Inference, by trying to classify if the generated text is “accurate” or contains some “omission”, “repetition”, “hallucination” or “value errors”. Moreover, the idea is built on pre-trained language models such as BERT, RoBERTa that have shown to be very successful in NLI. The work generates the training data for the semantic fidelity classifier using the original data-to-text dataset by applying transformations to synthetically create common errors in data-to-text generation. The text is assigned to one of the 5 buckets mentioned above and a classifier is learned to predict the fidelity label as in the figure above.



Finally, the model uses a beam search based decoder with a few tweaks to suit the table-to-text generation task. The models produces state-of-the-art results on 4 different evaluation datasets - LDC2017T10, WebNLG, CleanedE2E, ViGGO, and result in a BLEW score of 52.9 on WebNLG2017. The models also produce semantically sound textual representations for the input data due to the use of the semantic fidelity classifier.

The work also proposes the use of the classifier as a metric to be used to evaluate the generated sentences. Overall, the work shows the importance of using pre-trained language models for table-to-text generation and the need for semantic fine tuning on generated text to tackle previous challenges with open domain generation.

Trends and Future Work

At a high level, there are three major neural language modeling techniques that have been used to tackle table-to-text generation. Firstly, there were seq2seq models with attention which showed promising results. The idea was to use models that were built for text summarization on tables by modifying the input text. Popular techniques that worked were pointer generator networks [11] that use attention on seq2seq modeling with the added ability to copy source tokens. The second wave of models adopted transformer based architectures, which had proven effective on most other NLP tasks. This stage involved techniques such as CTRL [12] and GRU encoder with transformer decoder [7]. The most recent trend towards tackling table-to-text has been using pre-trained language models making use of the ideas of BERT and GPT. The popular architectures of this phase have been BART [13] and Bert2Bert [14]. We also observe the trend from pipeline based approaches towards more end-to-end approaches for table-to-text generation.

In 2020, there have been multiple datasets released for the task of table-to-text generation. The most extensive of these are ToTTo[3] and DART[1]. These two datasets pose slightly different versions of the task and come with new advantages and challenges of their own. Firstly, both the datasets are open domain and thus pose new challenges for model generalization as previous datasets, such as WebNLG2017[2] have been limited in the number of domains covered in the training and evaluation samples. Specifically, ToTTo frames the task as sentence generation given a Wikipedia table, highlighted cells and table metadata (such as table name) to control the generation. Whereas, DART frames the task as sentence generation given a collection of semantic RDF triples with hierarchical ontologies.

The performance of the current state-of-the-art models described in the previous sections on these datasets uncovers interesting new challenges. On both datasets, pre-trained language model based approaches work the best due to their extensive generalization capabilities as they have been trained on large corpora. Specifically, BART works best on DART dataset while Bert2Bert works best on the ToTTo dataset. Interestingly, using only the subtable(highlighted cells) as input performs better on ToTTo compared to using the whole table as input. The intuition behind this is that the whole table might be misleading to the model. Some exploration on effectively using table for providing context might help bridge the gap between the two approaches. Pointer-generator networks perform very well on both the datasets reaffirming the importance of attention based copying mechanisms. Incorporating this capability in pre-trained language models is a potential area for improvement. The hierarchical ontologies extracted in DART pose difficulties to the pre-trained language models due to the linearized nature of their inputs. A good area of exploration is finding richer ways to encode the semantic relationships expressed in the ontologies in the model. Both the datasets also pose challenges of hallucination, where the models generate unfaithful text sequences. Lastly, the open domain nature of both the datasets make it increasingly difficult to model rare and complex domains. But it also makes these datasets very effective in domain adaptation for new and unseen types of table-sentence spaces.

Overall, these new datasets and their accompanying challenges even with the current state-of-the-art models give us good pointers for improving the table-to-text generation techniques.

References

1. (Radev et al., 2020) DART: Open-Domain Structured Data Record to Text Generation
2. WebNLG Challenge 2020
3. (Parikh et al., 2020) ToTTo: A Controlled Table-To-Text Generation Dataset
4. (Wiseman et al., 2017) Challenges in Data-to-Document Generation
5. (Gehrmann et al., 2018) End-to-End Content and Plan Selection for Data-to-Text Generation
6. (Bao et al., 2018) Table-to-Text: Describing Table Region with Natural Language
7. (Ferreira et al., 2019) Neural data-to-text generation: A comparison between pipeline and end-to-end architectures
8. (Chen et al., 2020) Logic2Text: High-Fidelity Natural Language Generation from Logical Forms
9. (Harkous et al., 2020) Have Your Text and Use It Too! End-to-End Neural Data-to-Text Generation with Semantic Fidelity
10. (Zhu et al., 2019) Triple-to-Text: Converting RDF Triples into High-Quality Natural Languages via Optimizing an Inverse KL Divergence
11. (See et al., 2017) Get To The Point: Summarization with Pointer-Generator Networks
12. (Keskar et al., 2019) CTRL: A Conditional Transformer Language Model for Controllable Generation
13. (Lewis et al., 2019) BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension
14. (Rothe et al., 2020) Leveraging Pre-trained Checkpoints for Sequence Generation Tasks