**Doctoral Dissertation**

# Teaching Machines to Write from Data

Hayate Iso

February 26, 2021

Graduate School of Information Science
Nara Institute of Science and Technology

A Doctoral Dissertation
submitted to Graduate School of Information Science,
Nara Institute of Science and Technology
in partial fulfillment of the requirements for the degree of
Doctor of ENGINEERING

Hayate Iso

Thesis Committee:

| | |
|---|---|
| Professor Eiji Aramaki | (Supervisor) |
| Professor Taro Watanabe | (Co-supervisor) |
| Associate Professor Shoko Wakamiya | (Co-supervisor) |
| Professor Hiroya Takamura | (Advanced Industrial Science and Technology) |

# Teaching Machines to Write from Data*

Hayate Iso

**Abstract**

In the field of artificial intelligence (AI), natural language generation (NLG) has been studied to create automated language production systems. This crucial research area has been vigorously studied since the dawn of AI. It poses critical perspectives for the development of general-purpose AI, such as representing non-linguistic forms of information in forms that machines can understand and translating back into the form that humans can understand. Despite the progress of NLG systems over the past decades, conventional NLG systems still have great difficulty in generating fluent and diverse texts: many systems rely on template-based generation systems, which require excessive human effort to build a template database.

Recently advances in neural network-based language generation has led to the production of texts that are so fluent as to be indistinguishable from human-written texts. However, neural network-based generation systems also pose new challenges; they generate inconsistent sentences depending on input, and it becomes more challenging to process expressions with increasing input length.

This dissertation addresses problems involved in data-to-text generation systems and a text editing system related to the challenges described above. First, we explore the construction of a data-to-text generation system that can generate document-scale text from redundant data records [51]. Second, we introduce a new text editing task, referred to as fact-based text editing, in which the goal is to revise a given document to better describe facts in a knowledge base (e.g., several triples) [52]. We also develop a more accurate, efficient, and interpretable model for fact-based text editing compared to the standard encoder-decoder model.

*Doctoral Dissertation, Graduate School of Information Science,
Nara Institute of Science and Technology, February 26, 2021.

**Keywords:**

Text Generation, Natural Language Generation, Data-to-Text Generation, Summarization, Text Editing

# Contents

# List of Figures

# List of Tables

# Acknowledgements

First, I must thank my advisor Dr. Eiji Aramaki, who is the exceptional professor I've ever worked up until now and from now on. I have been impressed with the research attitude that always being mindful of the real world's impact. It will continue to influence my future life. I am also indebted for his understanding that I have the chance to cooperate with talented researchers around the world in various other research institutes.

I would like to extend thanks to Dr. Shoko Wakamiya. She spent countless times revising my first paper in 2016. Without her generous support, my first paper would never have been accepted.

I would like to thank Dr. Hiroya Takamura, who warmly served as a mentor at AIST and a thesis committee. I started my career as a natural language generation researcher during the research stay at AIST. This opportunity dramatically changed my life trajectory.

I would like to thank Dr. Taro Watanabe for agreeing to be on my thesis committee and helping me through the proposal, the defense, and the final version of the thesis.

I am thankful to Dr. Hang Li, who generously served as a mentor at ByteDance. I cannot thank him enough for discussing with me and supporting the life in China despite his hectic schedule. His passion and foresight are also always inspiring. The experience of working with him is one of the greatest fortunes of my Ph.D. life.

I also have to thank many other people who kindly served as my mentors during my research stays and have been incredibly generous with their time and insights. From AIST: Yusuke Miyao, Naoaki Okazaki, Ichiro Kobayashi, Hiroshi Noji. From ByteDance: Chao Qiao. From Megagon Labs: Wang-Chiew Tan, Xiaolan Wang, Yoshihiko Suhara, Stefanos Angelidis. From Amazon: Xing Fan,

Xiangcheng Kong, Hosseini Roya.

I am grateful to the administrative staff, Masami Kaneko, who has helped many paper works.

I would also like to thank the Japan Society for the Promotion of Science that supported my research.

Finally, I would like to thank my family, who allowed me to go to graduate school. My debt to them is unbounded.

# 1 Introduction

> *"The fundamental problem of communication is that of reproducing at one point, either exactly or approximately, a message selected at another point."*
>
> —Claude Shannon *(1948)*

Language production is an expression of the fundamental human ability to communicate with each other. We exchange ideas, knowledge, opinions, and other information in the form of language every day. Especially in recent years, information technology and the development of the global Internet has fostered such communication with seemingly anyone, anytime, and anywhere, breaking down limits in an unprecedented way.

In the field of artificial intelligence (AI), the problem of automating the language production, called natural language generation (NLG), is one of the crucial research areas, and it has been vigorously studied since the dawn of AI. This problem poses critical questions in developing general-purpose AI, such as those representing non-linguistic forms of information in forms that machines can understand and translating them into forms humans can understand. From an application standpoint, NLG systems can also be used to assist daily documentation tasks. The democratization of information has made it possible for people to acquire and consume a great variety of information, and the need to obtain this information in the form of language continues to grow. Accordingly, NLG systems have continued to show promise as a tool to support such document creation.

Despite these advantages, a long-term study of NLG systems is yet to be conducted, and in terms of potential real-world demand, the practical use of these systems is still limited. They tend to eventually generate sentences inconsistent with their input and have more difficulty processing as input length increases. In the following section, we will describe NLG research that has been conducted

to date. In particular, we will discuss two NLG research problems: data-to-text generation (Section 1.1.1), and text-to-text generation (Section 1.1.2). We will discuss tasks and approaches that have been undertaken so far, the accomplishments of existing systems, and practical challenges. The main goal of this thesis is to explore methods to improve data-to-text generation systems and text editing systems (Section 1.2).

## 1.1 A Brief Review of Text Generation

### 1.1.1 Data-to-Text Generations

A data-to-text generation system is designed to automatically generate text from non-linguistic data. These systems typically need to first select what information should be described in the output text, next, decide how to refer to that information, and finally generate adequate, coherent, and fluent text based on the selected data in the determined order. Most existing data-to-text generation systems fall into one of the two subcategories: pipeline architectures and end-to-end architectures, which we will discuss below.

#### Pipeline Approaches

The most widely investigated approach to NLG is the pipeline approach, which decomposes an NLG problem into several subproblems and then resolves each task independently. The NLG problem of transforming input data into output text in the pipeline approach requires solving the following three subproblems: *content selection*, *text planning*, and *surface realization*.

The content selection problem is that of determining the information that should be included in the output text. For example, consider automatically generating a summary of a basketball game from a database of game statistics. The database contains a wide range of information about the game, including the number of points scored by each team, the number of wins and losses, and statistics about each player's performance in that game. The game summary, however, mentions only some of that vital information in the database. A content selection model aims to identify a subset of the database to be mentioned. However,

manually building such a content selection rule is generally considered difficult.

To overcome this issue, content selection models tend to rely on statistical methods [111, 27, 5, 16, 68, 59]. Duboue and McKeown [27] address the statistical approach to content selection by solving the classification problem of whether each input data is mentioned in the text, and Barzilay and Lapata [5] proposed to extend the model to select the most likely combination of data based on the co-occurrence pattern of each input data. More generally, researchers have also considered automatically obtaining correspondences between data and text, also known as grounded language acquisition. Liang et al. [68] introduce a generative model that aims to derive text segmentation, fact identification, and alignment. Koncel-Kedziorski et al. [59], in contrast, present a discriminative model to identify the most likely segmentation and alignment.

Once the content information has been determined, the content order needs to be determined next. The text planning problem is that of deciding how the data selected in the prior stage should be mentioned in the output text. In the example of the basketball game summary, first, the game results are provided, followed by information on the players; subsequently, the schedule for the next game, and so forth are provided. The text planning task is attempted in the data-to-text task as well as in multi-document summarization [73, 7, 65, 6, 86, 32]. Barzilay and Lee [6] have built a generative model, referred to as a content model, representing topics and transitions between topics. Bollegala et al. [9], however, propose a bottom-up method to obtain the best arrangement of sentences.

Surface realization is the final task in NLG problems: forming natural text from the content and order information obtained in the former stages. The most straightforward approach is to create templates [98, 74], but this is not scalable because significant human resources are required to create linguistically varying template texts (although templates are produced automatically from data in recent attempts [123, 93, 30]). The predominant approach to surface realization in recent years has been a statistical approach [120, 119, 91, 79]. Puduppully et al. [91] and Moryossef et al. [79], for example, proposed a neural network-based text generation model from selected contents.

### End-to-End Approach

End-to-end NLG methods are intended to generate text from data directly. This end-to-end model allows for generation considering past decisions. Angeli et al. [2] proposed a unified framework that repeatedly selects data records and their corresponding templates. Konstas and Lapata [61] defined a probabilistic context-free grammar that can jointly consider the structure of data and text.

Recent progress in neural network-based language generations has led to the production of fluent texts that are indistinguishable from human-written texts. In this line of research, end-to-end neural NLG models have been actively studied [60, 75, 55, 66, 122, 80, 84, 69, 92, 130, 135, 104, 115] with the use of encoder-decoder architectures [110, 18, 3, 71, 113]. A neural encoder-decoder can easily handle any modality of input, and data-to-text is no exception. Typically, the encoder maps each input data record in an unordered fashion [132], and the decoder generates the text by referring to its input representation. Mei et al. [75], for example, built a data-to-text model extending an attention mechanism by introducing a pre-selector and a refiner module in addition to a standard selector. Liu et al. [69] proposed a structure-aware sequence-to-sequence model encoding the structure of a table and content information for the data-to-text generation task.

Owing to these advances in encoder-decoder architectures, significant progress has been made in data-to-text generation in recent years. However, this progress is often limited to cases where the output is at most a few sentences long or there is no redundancy in the input data [33, 85]. For example, if a model is intended to generate long texts, it often generates the same content repeatedly [48, 118]. Alternatively, if a model aims to handle redundant data, it may refer to erroneous data, resulting in the model generating inconsistent text [112, 24, 62, 63]. Moreover, this is often only recognized as a trivial error in the automatic evaluation metric, like BLEU [88], so the automatic textual evaluation metric often has a high score even though the model generates inconsistent text. This thesis attempts to address these issues.

### 1.1.2 Text-to-Text Generations

Text-to-text generation is a NLG task that encompasses a wider range of fields than a data-to-text generation, including machine translation [14], summarization [70], and others. The critical difference between text-to-text and data-to-text is concerning whether the input data are ordered. As this ordering information is important in a text-to-text task, order-sensitive encoding, such as recurrent neural networks [29, 47], or positional encoding [113, 102], is needed. Otherwise, an end-to-end encoder-decoder model can be applied to the text-to-text task directly. Therefore, this chapter focuses on the text-to-text task, especially the *text editing* task, rather than the technical details of the text-to-text model.

#### Text Editing

Text editing is to change the input text into the desired format by adding, deleting, or rewriting a text's content. In NLG, text editing tasks are tackled in the form of paraphrase generations [25], text style transfer [49, 103], grammatical error correction [83], automatic post editing [58, 105, 128], and text simplification [50, 124], and most of them can be performed naturally as text-to-text problems [67, 136, 137]. For example, Li et al. [67] introduce the deep reinforcement learning technique that combines the generator and the evaluator to train the paraphrase generation model. Zhao et al. [136] integrate paraphrasing rules with the Transformer model for text simplification. Zhao et al. [137] propose a method for English grammar correction using a Transformer and copy mechanism.

In contrast, text editing by humans can change the amount of information in text, not only through paraphrasing but also by introducing new content based on facts or removing content that contradicts with facts [126, 127, 128]. For instance, Yang et al. [128] investigated editor intentions in Wikipedia revisions, and they found that updating and introducing facts accounted for about 23% of all edits, while paraphrasing and simplification accounted for 15%. This paper addresses the issue and introduces a new text editing task to fill the gaps.

## 1.2 Thesis Outline

This dissertation addresses problems involved in data-to-text generation and text editing systems. First, we explore the construction of a data-to-text generation system that can generate document-scale text from redundant data records [51].

Second, we introduce a new text editing task, referred to as fact-based text editing, in which the goal is to revise a given document to better describe facts in a knowledge base (e.g., several triples) [52]. Furthermore, we develop a more accurate, efficient, and interpretable model for fact-based text editing compared to the standard encoder-decoder model.

We begin with background knowledge of neural sequence generation, memory networks, transition systems, and evaluation metrics in Chapter 2. We discuss the abovementioned problems for data-to-text generation in Chapter 3 and text editing in Chapter 4. We conclude this dissertation and discuss future directions for natural language generation research in Chapter 5.

### 1.2.1 Data-to-Document Generation

A data-to-text generation system should be able to select salient information and generate fluent and faithful text. However, if input data are too large to find the most salient parts, automated systems may select less critical information or generate text inconsistent with the input data. In our opinion, the problem is the inability to capture saliency transitions in the generative process. For example, as mentioned in the text planning part of Section 1.1.1, it is natural that the saliency of input data changes during the text generation process, and once that data is referred to, the saliency of that data becomes less in subsequent process. Thus, we need a model that finds salient parts, tracks their transitions, and expresses information faithful to the input.

We propose to address this by introducing a memory architecture for an encoder-decoder model that stores each input data's saliency and updates each input state during generation. Moreover, it was found that in the document-level generation task, the saliency of input data can vary for different writers. Once we integrated this information into the model, the model selected more salient data. This approach is based on the paper, Iso et al. [51], and will be detailed in Chapter 3.

### 1.2.2 Fact-based Text Editing

Text editing tasks involve many kinds of intentions, including paraphrasing and simplification, among others. However, the current natural language processing field mainly focuses on text-to-text editing that does not cover addition, deletion, or updates to semantic information.

To make the text editing system more practical, we introduce a new task, *Fact-based Text Editing*, including fact addition, deletion, and updates. This task can be considered as a data&text-to-text generation task. Researchers have developed methods to deal with the problem of retrieving other texts as templates [42, 41, 90]. The difference between the retrieval-based approach and fact-based text editing is that the former focuses on table-to-text generation based on other texts, while the latter focused on text-to-text generation based on structured data. This study further proposes a model that can solve some of the key problems, allowing fact-based text editing to be more interpretable, efficient, and accurate. This chapter is based on the paper, Iso et al. [52], and will be detailed in Chapter 4.

# 2 Background

This chapter provides detailed background information on NLG, including neural sequence generation models, memory networks, transition systems, and evaluation metrics.

## 2.1 Neural Sequence Generations

Neural sequence generation is a current de facto standard framework in NLG for generating a target sequence from source inputs using neural networks. For example, the framework includes machine translation, which translates a text in one source language to another target language, and summarization, which takes a document as input and generates a summary.

We describe autoregressive language models, which form the basis of neural sequence generation, and introduce encoder-decoder architectures as an extension of these language models. Then, we present attention mechanisms and copy mechanisms as a further extension of the encoder-decoder architectures.

### 2.1.1 Autoregressive Language Models

Language modeling is a task of modeling a probability distribution over a sequence of words, which can be formally described as

$$p(\mathbf{y}) = \prod_{t=1}^{|y|} p(y_t \mid y_1, \ldots, y_{t-1}),$$

where $\mathbf{y} = (y_1, \ldots, y_{|\mathbf{y}|})$ is a sequence of words, and $p(y_t \mid y_1, \ldots, y_{t-1})$ denotes the conditional probability of word $y_t$ given all preceding words, $y_1, \ldots, y_{t-1}$. The goal of language modeling is to obtain a model that can analyze the validity of

an arbitrary sequence of words as a probability, which has a crucial role even for "generating" a valid sequence of words [101].

One approach to estimate this conditional probability $p(y_t \mid y_1, \ldots, y_{t-1})$ is from relative frequency counts, which counts the number of sequence $y_1, \ldots, y_{t-1}$ appearing on the training corpus and counts the number of the sequence followed by $y_t$ as follows.

$$p(y_t \mid y_1, \ldots, y_{t-1}) = \frac{C(y_1, \ldots, y_{t-1}, y_t)}{C(y_1, \ldots, y_{t-1})}.$$

However, this approach naturally faces the sparsity problem, implying that the conditional distribution $p(y_t \mid y_1, \ldots, y_{t-1})$ cannot be accurately estimated because most sequences of words rarely appear in the corpus. For this reason, $N$-gram language models that only count the history of $N$ words instead of considering the entire history of the sequence and the associated smoothing methods have been developed so far [57, 17, 99, 108, 8, 11, 43, 15].

Autoregressive language models provide an intelligent solution to this problem, in the form of an autoregressive neural network transforming an arbitrary length of sequence, e.g., $(y_1, \ldots, y_{t-1})$ into a fixed sized vector $\boldsymbol{h}_{t-1}$, and then estimating the probability of the next word $y_t$ from this vector

$$\begin{aligned} p(y_t \mid y_1, \ldots, y_{t-1}) &= p(y_t \mid \boldsymbol{h}_{t-1}) \\ \boldsymbol{h}_{t-1} &= \phi(y_1, \ldots, y_{t-1}) \\ p(y_t \mid \boldsymbol{h}_{t-1}) &= \frac{\exp(\boldsymbol{h}_{t-1} \cdot \boldsymbol{w}_{y_t})}{\sum_{v \in \mathcal{V}} \exp(\boldsymbol{h}_{t-1} \cdot \boldsymbol{w}_v)}, \end{aligned}$$

where $\phi$ is the autoregressive map function, $\mathcal{V}$ is the vocabulary set, and $\boldsymbol{w}_v$ is the trainable weight for word $v \in \mathcal{V}$.

Mikolov et al. [78] first proposed an autoregressive language model using a recurrent neural network (RNN) [29], and Zaremba et al. [133] used long short-term memory (LSTM) [47, 38] with dropout regularization [107] to improve the autoregressive mapping, and many other variants of the autoregressive language model using LSTM have been proposed [31, 36, 77, 76]. Recently, Dai et al. [22] proposed an autoregressive language model that only uses an attention mechanism (described in Section 2.1.3) without using RNN.

### 2.1.2 Encoder-Decoder Models

The encoder-decoder model is a conditional sequence generation model $p(\mathbf{y} \mid \mathbf{x})$, which encodes a source input $\mathbf{x}$ and then decodes a target sequence $\mathbf{y}$ from this encoded representation [54, 18, 110, 3, 113]. This model can be considered as an extension of a language model that is a product of conditional distribution $p(y_t \mid y_1, \ldots, y_{t-1}, \mathbf{x})$ predicting the next target word $y_t$ based on the preceding words, $y_1, \ldots, y_{t-1}$, and source input $\mathbf{x}$

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p(y_t \mid y_1, \ldots, y_{t-1}, \mathbf{x}).$$

The most basic form of the encoder-decoder model [54, 110] is one that encodes the source input $x$ into a single feature vector $\boldsymbol{h}_{\mathbf{x}}$ with a mapping function $\theta$ and then generates the target sequence based on all preceding words $y_1, \ldots, y_{t-1}$ and the feature vector $\boldsymbol{h}_{\mathbf{x}}$

$$\boldsymbol{h}_{\mathbf{x}} = \theta(x_1, \ldots, x_{|\mathbf{x}|})$$
$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p(y_t \mid y_1, \ldots, y_{t-1}, \boldsymbol{h}_{\mathbf{x}}).$$

### 2.1.3 Attention Mechanisms

The basic form of the encoder-decoder model utilizes a fixed-length feature vector $\boldsymbol{h}_{\mathbf{x}}$ obtained by one-shot feature mapping in an integrated generation process. The fundamental problem with this approach is that it becomes increasingly difficult to map all information to a single feature vector $\boldsymbol{h}_{\mathbf{x}}$.

One powerful approach to address this issue is attention mechanism [3, 71, 87, 113, 132]. The attention mechanism can focus on a different subset of source information at each timestep.

More formally, the attention module calculates attention weights $\boldsymbol{\alpha}_{t-1} = \{\alpha_{t-1,i}\}_{i=1}^{|\mathbf{x}|}$ to aggregate the source input information into the single *context vector* $\boldsymbol{c}_{t-1}$ at

$t-1$

$$\alpha_{t-1,i} = \frac{\exp(\gamma(\boldsymbol{h}_{t-1}, \boldsymbol{h}_i))}{\sum_{j=1}^{|\mathbf{x}|} \exp(\gamma(\boldsymbol{h}_{t-1}, \boldsymbol{h}_j))}$$

$$\boldsymbol{c}_{t-1} = \sum_{i=1}^{|\mathbf{x}|} \alpha_{t-1,i} \boldsymbol{h}_i$$

$$p(\mathbf{y} \mid \mathbf{x}) = \prod_{t=1}^{|\mathbf{y}|} p(y_t \mid y_1, \ldots, y_{t-1}, \boldsymbol{c}_{t-1}),$$

where $\gamma$ is a mapping function, $\boldsymbol{h}_{t-1}$ is the hidden state of decoder at time $t-1$, and $\boldsymbol{c}_{t-1}$ is a context vector that aggregates the source hidden states with attention weight $\boldsymbol{\alpha}_{t-1}$.

### 2.1.4 Copy Mechanisms

An intrinsic drawback of the neural sequence generation model is the inability to generate out-of-vocabulary words, which are not included in the list of pre-defined vocabulary [72]. A copy mechanism [39, 40, 100] is one sophisticated solution that directly *copies* the word from the source input.

Technically, the encoder-decoder model with the copy module computes the probability distribution of the next word with the convex combination of estimated word distribution and copy probability of source words as follows.

$$p_{\text{COPY}} = \frac{1}{1 + \exp\left(-\psi(\mathbf{y}_{t<}, \mathbf{x})\right)}$$

$$p_{\text{GEN}}(y_t \mid \boldsymbol{h}_{t-1}) = \frac{\exp(\upsilon(\boldsymbol{h}_{t-1}, \boldsymbol{w}_{y_t}))}{\sum_{v \in \mathcal{V}} \exp(\upsilon(\boldsymbol{h}_{t-1}, \boldsymbol{w}_v))}$$

$$p(y_t \mid y_1, \ldots, y_{t-1}, x) = (1 - p_{\text{COPY}}) * p_{\text{GEN}}(y_t \mid \boldsymbol{h}_{t-1}) + p_{\text{COPY}} * \sum_{i:x_i=y_t} \delta_{t,x_i}.$$

$\psi$ and $\upsilon$ are mapping functions, $p_{\text{COPY}} \in [0, 1]$ is the probability of switching between generation and copy probability, and $\delta_{t,i}$ is the copy probability for source word $x_i$, where $\sum_{i=1}^{|\mathbf{x}|} \delta_{t,x_i} = 1$.

## 2.2 Memory Networks

Memory networks [121, 109, 37, 44] are modules that maintain and update representations of memory components. These memory network modules have been

applied to natural language understanding tasks to maintain a track of entity states [46, 10].

Recently, entity tracking has become popular for generating coherent text in NLG fields as well [55, 53, 129, 20]. Kiddon et al. [55] proposed a neural checklist model updating pre-defined item states, and Ji et al. [53] proposed an entity representation for the language model that updated entity tracking states when an entity was introduced. Herein, we describe a versatile yet straightforward framework for memory-augmented language modeling, the *reference-aware language model* [129].

**Reference-aware language models**

The reference-aware language model introduces a new random variable $\mathbf{z} = (z_1, \ldots, z_{|\mathbf{y}|})$, controlling the decision of which input to generate output from at each timestep $t$, as follows.

$$\begin{aligned} p(\mathbf{y}, \mathbf{z}) &= \prod_{t=1}^{|\mathbf{y}|} p(y_t, z_t \mid \mathbf{y}_{<t}, \mathbf{z}_{<t}) \\ &= \prod_{t=1}^{|\mathbf{y}|} p(y_t \mid \mathbf{y}_{<t}, z_t) p(z_t \mid \mathbf{y}_{<t}), \end{aligned}$$

where $\mathbf{z}_{<t} = (z_1, \ldots, z_{t-1})$ represents the preceding decision histories.

For example, in document-level language modeling, it is typically challenging to remember all previously appearing entity states, leading to low language model quality. The reference-aware language model maintains a list of entities that appeared in past timesteps $< t$ in memory and determines whether the next word $y_t$ is an entity in the list. If an entity is referenced, the memory is updated depending on whether it has appeared before. If an entity has never appeared before, this entity information is newly allocated to memory. Otherwise, the memory of that entity is updated using the language model's state at that time.

Notably, this model selects the salient entity state, whereas our model extends this entity tracking module for data-to-text generation tasks. The entity tracking module selects the salient entity, and the appropriate attribute in each timestep updates its state and generates coherent summaries from the selected data record.

## 2.3 Transition-based Architectures

A transition-based model consists of a set of configurations and transition actions. The model incrementally selects a transition action based on the configurations' latest state until it reaches a terminal configuration. Transition-based systems decompose complex problems into incremental action selection problems that can be solved efficiently. More formally, these models predict a sequence of actions $\mathbf{a} = (a_1, \ldots, a_{|\mathbf{a}|})$ given an input sequence $\mathbf{x}$ as follows

$$p(\mathbf{a} \mid \mathbf{x}) = \prod_{t=1}^{|\mathbf{a}|} p(a_t \mid \xi(c_t)),$$

where $c_t$ is a configuration at time $t$, and $\xi$ is a mapping function of configuration.

The transition-based system has been applied to a variety of tasks, including dependency parsing [117, 56], constituency parsing [116, 21], word segmentation and part-of-speech tagging [134], named entity recognition [64], and sentence compression [1]. This thesis provides technical details of a transition-based system with stacked long short-term memory [28], which is a simple but highly flexible model.

### Stack LSTM

This stacked LSTM is a component of the transition system used in Ballesteros et al. [4], which is augmented with a stack pointer pointing to the latest state. Our transition-based dependency parsing system with stacked LSTM has one buffer of words to be parsed $\boldsymbol{b}$ and two stacks, one for the partially structured syntactic elements $\boldsymbol{s}$ and another for the action history $\boldsymbol{a}$, and it also has SHIFT, REDUCE-RIGHT, and REDUCE-LEFT actions. The model predicts an action $a_t$ at each timestep $t$ by referring to stack pointers for each configuration and using the corresponding LSTM state as feature representations

$$\xi(c_t) = \tau\left(\boldsymbol{W}_\xi[\boldsymbol{s}_t; \boldsymbol{b}_t; \boldsymbol{a}_t] + \boldsymbol{b}_\xi\right),$$

where $\boldsymbol{s}_t, \boldsymbol{b}_t, \boldsymbol{a}_t$ are the latest states at action step $t$, $\tau$ is a nonlinear function, e.g. hyperbolic tangent (tanh) or rectified linear unit [81], and $\boldsymbol{W}_\xi, \boldsymbol{b}_\xi$ are trainable weights. If the model selects the SHIFT action, the latest state is popped from the

buffer fed to the stack. For REDUCE actions, the model composes a dependency tree fragment with a recursive neural network [106, 45].

## 2.4 Evaluating Text Generation Systems

Along with architecture, evaluating the generated text's quality plays an important role in the text generation systems. Once a model has been improved, we need to measure how the improvement contributes to the quality of the generated text. Notably, without efficient text evaluation methods, good text generation models cannot be built.

However, unlike other natural language processing problems such as text classification, there is no single correct answer to the text generation problem. For example, in machine translation, there may be many possible correct answers when translating one sentence into another language. However, the manual evaluation also costs considerable time and money and significantly slows the development and improvement cycles of text generation systems.

This section describes automatic evaluation metrics quantifying the quality of the generated text, such as BLEU and SARI scores and extractive evaluation metrics.

### 2.4.1 Evaluation via *N*-gram Matching

#### BLEU

BLEU stands for bilingual evaluation understudy, which is one of the most well-known methods for assessing the quality of machine-generated text [88]. It was developed for machine translation tasks and has recently been used in other text generation tasks.

To measure the quality of the generated text using the BLEU score, we prepare a reference text and then evaluate the generated text's quality by $N$-gram matching between the generated and reference texts. More precisely, this approach counts $N$-gram matches between the generated text $\mathbf{y}_{\texttt{gen}}$ and the reference text $\mathbf{y}_{\texttt{ref}}$ and then measures $N$-gram precision, i.e., the ratio of correct $N$-grams compared to the sum of generated $N$-grams as follows.

$$\textsc{Prec}_N = \frac{\sum_{\mathcal{G}_N, \mathcal{R}_N \in \mathcal{Y}_N^{\mathtt{gen}} \times \mathcal{Y}_N^{\mathtt{ref}}} |\mathcal{G}_N \bigcap \mathcal{R}_N|}{\sum_{\mathcal{G}_N \in \mathcal{Y}_N^{\mathtt{gen}}} |\mathcal{G}_N|},$$

where $\mathcal{G}_N$ and $\mathcal{R}_N$ are bags (or multisets) of $N$-grams for the generated and reference texts, and $\mathcal{Y}_N^{\mathtt{gen}}$ and $\mathcal{Y}_N^{\mathtt{ref}}$ are sets of the bags $\mathcal{G}_N$ and $\mathcal{R}_N$, respectively.

The drawback of directly using this $\textsc{Prec}_N$ score is— it results in inaccurately higher scores if the generated text is short. To prevent this problem, the BLEU score is defined by combining the $\textsc{Prec}_N$ score with the brevity penalty (BP), which penalizes the generated text if it is shorter than the reference text as follows.

$$\textsc{Bleu}\text{-}N = \mathrm{BP} * \exp \sum_{n=1}^{N} \lambda_n \log \textsc{Prec}_n$$
$$\mathrm{BP} = \min \left(1, \exp \left(1 - \frac{\sum_{\mathcal{G} \in \mathcal{Y}^{\mathtt{gen}}} \sum_{\mathbf{y}_{\mathtt{gen}} \in \mathcal{G}} |\mathbf{y}_{\mathtt{gen}}|}{\sum_{\mathcal{R} \in \mathcal{Y}^{\mathtt{ref}}} \sum_{\mathbf{y}_{\mathtt{ref}} \in \mathcal{R}} |\mathbf{y}_{\mathtt{ref}}|}\right)\right),$$

where $\lambda_N$ is the weight for the $\textsc{Prec}_N$. In most cases, the Bleu-N score with $N = 4$ and $\lambda_n = \frac{1}{4}$ for all $n \in \{1, \ldots, 4\}$ is used as Bleu score, and this study uses this configuration unless otherwise mentioned.

### SARI

Sari stands for system output against references and against the input sentence [125]. The Sari score measures the ability of Keep, Add, and Delete by the system. The most significant difference between this and the Bleu score is that BLEU just uses the reference text $\mathbf{y}_{\mathtt{ref}}$, while Sari score uses input text $\mathbf{x}$ as well as reference text $\mathbf{y}_{\mathtt{ref}}$ to evaluate the output text $\mathbf{y}_{\mathtt{gen}}$.

The Sari score is formally defined as:

$$\textsc{Sari}\text{-}N = \sum_{op \in \{\textsc{Keep}, \textsc{Add}, \textsc{Delete}\}} \lambda^{op} * \text{F1-}N^{op}$$
$$\textsc{Prec}\text{-}N^{op} = \frac{1}{N} \sum_{n=1}^{N} \textsc{Prec}_n^{op}$$
$$\textsc{Rec}\text{-}N^{op} = \frac{1}{N} \sum_{n=1}^{N} \textsc{Rec}_n^{op}$$
$$\text{F1-}N^{op} = \frac{2 * \textsc{Prec}\text{-}N^{op} * \textsc{Rec}\text{-}N^{op}}{\textsc{Prec}\text{-}N^{op} + \textsc{Rec}\text{-}N^{op}},$$

where $\lambda^{op}$ is the weight of each operation*. The SARI score can be represented as the weighted sum of F1-$N$ scores for KEEP, ADD, and DELETE operations. Hereafter, we assume that the $N = 4$ and $\lambda^{op} = \frac{1}{|\{\text{KEEP,ADD,DELETE}\}|} = \frac{1}{3}$ for all operations unless otherwise noted.

The SARI score obtains the Precision and Recall scores in different ways for each operation. First, the KEEP score presents a reward if the system correctly *keeps* text that appears in both the input and reference sentences defined as:

$$\text{PREC}_N^{\text{KEEP}} = \frac{|\mathcal{I}_N \bigcap \mathcal{G}_N \bigcap \mathcal{R}_N|}{|\mathcal{I}_N \bigcap \mathcal{G}_N|}$$
$$\text{REC}_N^{\text{KEEP}} = \frac{|\mathcal{I}_N \bigcap \mathcal{G}_N \bigcap \mathcal{R}_N|}{|\mathcal{I}_N \bigcap \mathcal{R}_N|},$$

where $\mathcal{I}_N$ is the bag of $N$-grams for the input text $\mathbf{x}$.

However, the ADD score is used to present a reward if the system correctly *adds* text that appears in the reference text but does not appear in the input text, and the DELETE score is used to present a reward if the system correctly *deletes* text that appears in the input text but not in the reference text, calculated as follows.

$$\text{PREC}_N^{\text{ADD}} = \frac{|(\mathcal{G}_N \backslash \mathcal{I}_N) \bigcap \mathcal{R}_N|}{|\mathcal{G}_N \backslash \mathcal{I}_N|}$$
$$\text{REC}_N^{\text{ADD}} = \frac{|(\mathcal{G}_N \backslash \mathcal{I}_N) \bigcap \mathcal{R}_N|}{|\mathcal{R}_N \backslash \mathcal{I}_N|}$$
$$\text{PREC}_N^{\text{DELETE}} = \frac{|(\mathcal{I}_N \backslash \mathcal{G}_N) \bigcap (\mathcal{I}_N \backslash \mathcal{R}_N)|}{|\mathcal{I}_N \backslash \mathcal{G}_N|}$$
$$\text{REC}_N^{\text{DELETE}} = \frac{|(\mathcal{I}_N \backslash \mathcal{G}_N) \bigcap (\mathcal{I}_N \backslash \mathcal{R}_N)|}{|\mathcal{I}_N \backslash \mathcal{R}_N|}.$$

### 2.4.2 Evaluation via Information Extraction

As mentioned above, $N$-gram matching-based text evaluation metrics are often used because they can be evaluated by simply preparing a reference text. However, in the context of data-to-text generation, the adequacy of whether the generated text is consistent with the input data is also an important issue, but it is generally difficult to measure with $N$-gram matching metrics.

*In the paper in which the SARI score was first proposed by Xu et al. [125], the PREC-$N^{\text{DELETE}}$ score was used instead of the F1-$N^{\text{DELETE}}$ , but in this study, the F1-$N$ score is used in all operations to calculate SARI-$N$ score.

| Player | Pts | Reb | Ast | Blk | Stl | Min | ... |
|---|---|---|---|---|---|---|---|
| **LeBron James** | 24 | 5 | 4 | 2 | 2 | 25 | ... |
| | | | ... | | | | |

| Ref. | | Bleu | RG | CS | CO |
|---|---|---|---|---|---|
| Sys A | **LeBron James** posted 20 points , five rebounds , four assists , one steal and one block . | **48.59** | 40.00 | 44.44 | **40.00** |
| Sys B | **LeBron James** scored 24 points , four assists , five rebounds , one steal in 25 minutes . | 34.82 | **80.00** | **68.57** | **40.00** |

Table 2.1: An example of unsuccessful evaluation by $N$-gram matching, Bleu score in this case.

The example shown in Table 2.1 indicates that the text generated by system A has a high Bleu score, even though it includes considerable information contradictory to the input data.

In contrast, the text generated by system B produces text consistent with the input data but uses a different vocabulary or data that do not appear in the reference text. In this case, the Bleu score decreases, although it contains very little inconsistent text with the input data.

Therefore, evaluation metrics that consider consistency with the input data are becoming more critical, especially in data-to-text generation tasks [122, 24]. In this study, we explain the extractive evaluation metrics proposed by Wiseman et al. [122] using an external information extractor to obtain entity-value pairs and their types from the text to evaluate the adequacy. More concretely, the information extractor retrieves sequences of triples from texts. For example in the reference shown in Table 2.1, the sequence of retrieved triples becomes ((LeBron James, FIRST_NAME, 24), ..., (LeBron James, STL, 2)). The same extraction is applied to the generated text and the reference, and then the following metrics are computed by comparing with the input data or between the extracted sequences.

### Relation Generation

Relation generation (RG) is a measure of the faithfulness of the generated text to the input data. To examine whether the system is able to generate text consistent with the input data, it compares the triples extracted from the generated text with the input data as follows.

$$\textsc{Prec}^{\text{RG}} = \frac{\sum_{\mathcal{T}_{\texttt{gen}}, \mathcal{X} \in \mathcal{S}_{\texttt{gen}} \times \mathcal{D}} |\mathcal{T}_{\texttt{gen}} \bigcap \mathcal{X}|}{\sum_{\mathcal{T}_{\texttt{gen}} \in \mathcal{S}_{\texttt{gen}}} |\mathcal{T}_{\texttt{gen}}|},$$

where $\mathcal{X}$ is the set of input data, $\mathcal{D}$ is the set of input datasets, $\mathcal{T}_{\texttt{gen}}$ is the sequence of extracted triples from the generated text, and $\mathcal{S}_{\texttt{gen}}$ is the set of these sequences.

In the example of system B in Table 2.1, four out of five extracted data points are consistent with the input data. Thus, the precision of the RG score becomes 80.00.

### Content Selection

Content selection (CS) measures whether the system generates essential information by comparing the input information and the reference text. Technically, it calculates the precision, recall, and f1 score of the relations extracted from the generated text against those of the reference text as follows.

$$\begin{aligned}
\textsc{Prec}^{\text{CS}} &= \frac{\sum_{\mathcal{T}_{\texttt{gen}}, \mathcal{T}_{\texttt{ref}} \in \mathcal{S}_{\texttt{gen}} \times \mathcal{S}_{\texttt{ref}}} |\mathcal{T}_{\texttt{gen}} \bigcap \mathcal{T}_{\texttt{ref}}|}{\sum_{\mathcal{T}_{\texttt{gen}} \in \mathcal{S}_{\texttt{gen}}} |\mathcal{T}_{\texttt{gen}}|} \\
\textsc{Rec}^{\text{CS}} &= \frac{\sum_{\mathcal{T}_{\texttt{gen}}, \mathcal{T}_{\texttt{ref}} \in \mathcal{S}_{\texttt{gen}} \times \mathcal{S}_{\texttt{ref}}} |\mathcal{T}_{\texttt{gen}} \bigcap \mathcal{T}_{\texttt{ref}}|}{\sum_{\mathcal{T}_{\texttt{ref}} \in \mathcal{S}_{\texttt{ref}}} |\mathcal{T}_{\texttt{ref}}|} \\
\text{F1}^{\text{CS}} &= \frac{2 * \textsc{Prec}^{\text{CS}} * \textsc{Rec}^{\text{CS}}}{\textsc{Prec}^{\text{CS}} + \textsc{Rec}^{\text{CS}}}.
\end{aligned}$$

For example, if the information about points, assists, and rebounds is correctly generated in the output text by system B in Table 2.1, the precision, recall, and F1 scores for content selection become 60.00, 80.00, 68.57, respectively.

### Content Ordering

Content ordering (CS) measures whether the data generated are used in the correct order. To be more precise, it compares the sequences of triples extracted

from the generated text and the reference text using the normalized Damerau-Levenshtein Distance (DLD) [12] as.

$$CO = \frac{1}{|\mathcal{S}|} \sum_{\mathcal{T}_{\texttt{gen}}, \mathcal{T}_{\texttt{ref}} \in \mathcal{S}_{\texttt{gen}} \times \mathcal{S}_{\texttt{ref}}} \text{normalized DLD}(\mathcal{T}_{\texttt{gen}}, \mathcal{T}_{\texttt{ref}}),$$

where $|\mathcal{S}| = |\mathcal{S}_{\texttt{gen}}| = |\mathcal{S}_{\texttt{ref}}|$ is the total number of sets. The content ordering scores are better when the system describes the input data in the same order as the reference text.

# 3 Data-to-Document Generations

Although encoder-decoder models offer the ability to generate high-quality text for data-to-text generation [75, 66, 69], generating a long and coherent summary from large-scale data remains a challenge [122]. One challenge is that the input data are too large for a naive encoder-decoder model for finding their salient part, i.e., to determine which part of the data should be mentioned. Furthermore, the salient part moves as data are delineated by summary. For example, when generating a summary of a basketball game (Table 3.1 (b)) from the box score (Table 3.1 (a)), the input contains numerous data records regarding the game: e.g., *Jordan Clarkson scored 18 points*. Existing models often refer to the same data record multiple times [91]. However, these models may mention an incorrect data record, e.g., *Kawhi Leonard added 19 points* when the summary should mention *LaMarcus Aldridge*, who scored 19 points. Thus, we need a model that finds salient parts, tracks their transitions, and expresses information faithful to the input.

This chapter proposes a novel data-to-text generation model with two modules, one for saliency tracking and another for text generation. The tracking module tracks saliency in the input data so that when the module detects a saliency transition, the tracking module selects a new data record* and updates its state. The text generation module generates a summary conditioned on the current tracking state. The model proposed here could be considered to be a neural network-based variant of an end-to-end data-to-text generation model selecting input information and maintaining decision histories while generating the report [2]. A significant difference from the conventional model is using a re-

*We use "data record" and "relation" interchangeably.

current neural network. Our model employs a recurrent neural network-based saliency tracking module for maintaining decision histories as continuous vectors instead of explicitly retaining all decision history.

Moreover, some writer-specific patterns and characteristics, i.e., how data records are selected to be referred to, how data records are represented as text, e.g., the order of data records and the word usages, are also issues. Thus, writer information was added to the model.

The experimental results demonstrate that, even without writer information, our proposed model achieves the best performance among all previous models across all evaluation metrics with a 91.98% precision of relation generation, 43.31 F1 scores of content selection, 21.56% normalized Damerau-Levenshtein Distance (DLD) of content ordering, and a BLEU score of 15.74. It was also confirmed that adding writer information further improved performance.

## 3.1 Data

### 3.1.1 Data Sources

In this study, we use the box-score and summary pairs of an NBA game dataset. One of the most famous datasets is RotoWire dataset, which is introduced by Wiseman et al. [122]. The summary in this dataset as shown in Table 3.1b contains numerical information such as the win/loss of each team (e.g., (Knicks, Win, 16)), and the points and rebounds scores by each player, (e.g., (Carmelo Anthony, Reb, 11)), stored as a form of the box-score shown in Table 3.1a. Note that most of the records for box-score were not used in summary, and the alignment between input records and the output text is not given explicitly[†]. In other words, the main problem in this task is to select the salient information from the box-score and transform them into a consistent summary.

However, we found an issue: in this dataset, some NBA games had two summaries, one of which was sometimes in the training data, and the other was in the testing or validation data. Such summaries are similar to each other, though

[†]As described in Section 3.3.1, it is possible to obtain noisy alignments between the input box-score and the output summary using an information extractor.

| Team | H/V | Win | Loss | Pts | Reb | Ast | Fg_Pct | Fg3_Pct | ... |
|---|---|---|---|---|---|---|---|---|---|
| Knicks | H | **16** | **19** | **104** | 46 | 26 | 45 | 46 | ... |
| Bucks | V | **18** | **16** | **105** | 42 | 20 | 47 | 32 | ... |

| Player | H/V | Pts | Reb | Ast | Blk | Stl | Min | City | ... |
|---|---|---|---|---|---|---|---|---|---|
| Carmelo Anthony | H | **30** | **11** | **7** | 0 | **2** | 37 | New York | ... |
| Derrick Rose | H | **15** | **3** | **4** | 0 | 1 | 33 | New York | ... |
| Courtney Lee | H | **11** | **2** | **3** | 1 | 1 | 38 | New York | ... |
| Giannis Antetokounmpo | V | **27** | **13** | **4** | **3** | 1 | 39 | Milwaukee | ... |
| Greg Monroe | V | **18** | **9** | **4** | 1 | **3** | 31 | Milwaukee | ... |
| Jabari Parker | V | **15** | **4** | **3** | 0 | 1 | 37 | Milwaukee | ... |
| Malcolm Brogdon | V | **12** | **6** | **8** | 0 | 0 | 38 | Milwaukee | ... |
| Mirza Teletovic | V | **13** | 1 | 0 | 0 | 0 | 21 | Milwaukee | ... |
| John Henson | V | 2 | 2 | 0 | 0 | 0 | 14 | Milwaukee | ... |
| ... | ... | ... | ... | ... | ... | ... | | | |

(a) Box score: Top contingency table shows number of wins and losses and summary of each game. Bottom table shows statistics of each player such as points scored (Player's Pts), and total rebounds (Player's Reb).

The **Milwaukee Bucks** defeated the **New York Knicks**, **105**-**104**, at Madison Square Garden on Wednesday. The **Knicks** (**16**-**19**) checked in to Wednesday's contest looking to snap a five-game losing streak and heading into the fourth quarter, they looked like they were well on their way to that goal. ... **Antetokounmpo** led the Bucks with **27** points, **13** rebounds, **four** assists, a steal and **three** blocks, his second consecutive double-double. **Greg Monroe** actually checked in as the second-leading scorer and did so in his customary bench role, posting **18** points, along with **nine** boards, **four** assists, **three** steals and a block. **Jabari Parker** contributed **15** points, **four** rebounds, **three** assists and a steal. **Malcolm Brogdon** went for **12** points, **eight** assists and **six** rebounds. **Mirza Teletovic** was productive in a reserve role as well, generating **13** points and a rebound. ... **Courtney Lee** checked in with **11** points, **three** assists, **two** rebounds, a steal and a block. ... The Bucks and Knicks face off once again in the second game of the home-and-home series, with the meeting taking place Friday night in Milwaukee.

(b) NBA basketball game summary: Each summary consists of victory or defeat of the game and highlights of valuable players.

Table 3.1: Example of input and output data: task defines box score (3.1a) used for input and summary of game (3.1b) used as output. Extracted entities are shown in **bold face**. Extracted values are shown in green.

not identical. Additionally, Wang [114] found that summaries in the RotoWire dataset had only about 60% of their content grounded in the box-scores.

### 3.1.2 Data Creation Procedure

To conduct more reliable experiments, we use two datasets, RotoWire-Modified and RotoWire-FG, instead of the original RotoWire dataset.

The RotoWire-Modified dataset is a cleaned version of the original RotoWire dataset. The script provided by Wiseman et al. [122] was run, which was used for crawling the RotoWire website for NBA game summaries. The script collected approximately 78% of the summaries in the original dataset; the remaining summaries were removed. The box-scores associated with the collected summaries were also collected. It was observed that some of the box-scores differed from the original RotoWire dataset.

The dataset contains 3,738 instances (i.e., pairs of a summary and box-scores). However, the four shortest summaries were an announcement regarding the postponement of a match. Thus four instances such as these were deleted, and 3,734 instances were left in the dataset. We followed Wiseman et al. [122] to split our dataset into training, development, and test data. Finally, the sizes of the RotoWire-Modified 's training, development, and test datasets were respectively 2705, 532, and 497. On average, each summary had 384 tokens and 644 data records. Each match had only one summary in the dataset.

Information on the writer for each summary was also collected, and the dataset contained 31 different writers. The most prolific writer in the dataset wrote 607 summaries. Some writers wrote less than ten summaries; on average, each writer wrote 117 summaries. The statistics of the number of relations and the length of summary for each writer were also calculated. As shown in Figure 3.1, each writer had a remarkable tendency for the number of retrieved relations and the length of the summary.

The RotoWire-FG dataset Wang [114] introduced is an enlarged and purified RotoWire dataset. They enlarged the dataset by crawling the summaries of 50% more games than the original RotoWire dataset, and then, they purified the collected summaries to be more grounded. More precisely, by retaining the sentences, if any numerical score was associated with the box score. Table 3.2

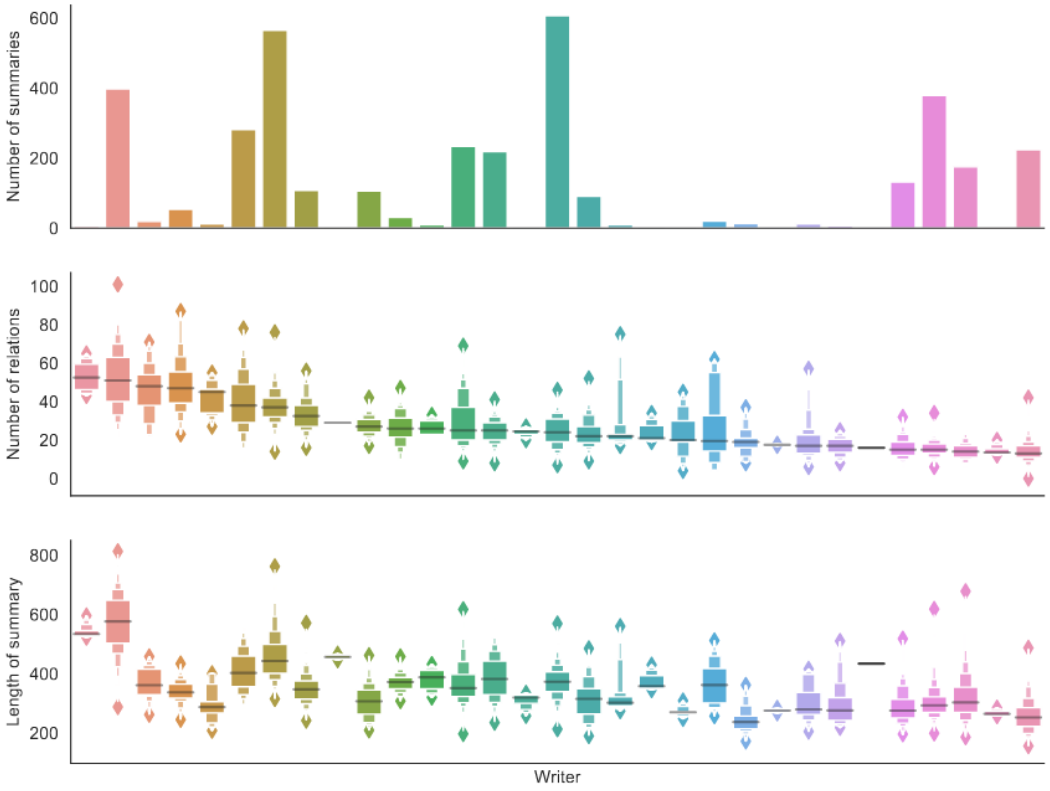



Figure 3.1: Bar chart for the number of summaries for each writer, and boxplots for the number of relations and the length of summary for each writer. Horizontal axes are sorted by the median number of relations retrieved by the information extractor.

gives the statistics of the datasets.

## 3.2 Model

### 3.2.1 Preliminaries

The aim of the data-to-text generation task is to generate a summary $y_{1:T} = (y_1, \ldots, y_T)$ given a set of records $\boldsymbol{x} = \{r_i\}_{i=1}^{|\boldsymbol{x}|}$. The most straightforward approach to constructing a data-to-text generation model is to use an attentional encoder-decoder model with a copy mechanism, $p(y_{1:T} \mid \boldsymbol{x})$ [122], but this model is known

| | RotoWire-Modified | | | RotoWire-FG | | |
|---|---|---|---|---|---|---|
| | Train | Valid | Test | Train | Valid | Test |
| $\#\mathcal{D}$ | 2,705 | 532 | 497 | 5,340 | 1,147 | 1,148 |
| $\#\mathcal{W}$ | 1.0M | 207k | 196k | 1.8M | 397k | 399k |

Table 3.2: Statistics of RotoWire-Modified and RotoWire-FG, where $\#\mathcal{D}$ is the number of instances, and $\#\mathcal{W}$ is the total numbers of words in the summaries.

to encounter a text hallucination problem, producing factually incorrect text by referencing incorrect data.

A well-known approach to mitigate this problem, the two-stage data-to-text generation model proposed by Puduppully et al. [91] decomposes the model into content selection and planning modules $p(z_{1:|z|} \mid \boldsymbol{x})$ and a text generation module $p(y_{1:T} \mid \boldsymbol{x}, z_{1:|z|})$, where $z_{1:|z|}$ is the latent variables for content selection and planning. This approach improved the adequacy performance by a large margin compared to the attentional encoder-decoder, but there remains room for improvement compared to the template-based model.

This study introduces a new data-to-text model that keeps track of a salient entity while generating the summary. The most crucial difference is that our model dynamically updates the input states on every timestep, while previous models use fixed input representations once encoded throughout generations.

### 3.2.2 Saliency-Aware Text Generation

At the core of this model is a neural language model with a hidden state $\boldsymbol{h}^{\text{LM}} \in \mathbb{R}^h$ to generate a summary $y_{1:T}$ given a set of data records $\boldsymbol{x}$, where $h$ is the size of hidden states. This model has another hidden state $\boldsymbol{h}^{\text{Ent}} \in \mathbb{R}^h$, which is used to remember the data records that have been referred to. $\boldsymbol{h}^{\text{Ent}}$ is also used to update $\boldsymbol{h}^{\text{LM}}$, meaning that the referred data records affect the text generation.

Our model decides whether to refer to $\boldsymbol{x}$, which data record $r \in \boldsymbol{x}$ to mention, and how to express a number. The selected data record is used to update $\boldsymbol{h}^{\text{Ent}}$.

| $t$ | 199 | 200 | 201 | 202 | 203 | 204 | 205 | 206 | 207 | 208 | 209 |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $Y_t$ | Jabari | Parker | contributed | 15 | points | , | four | rebounds | , | three | assists |
| $Z_t$ | 1 | 1 | 0 | 1 | 0 | 0 | 1 | 0 | 0 | 1 | 0 |
| $E_t$ | Jabari Parker | Jabari Parker | - | Jabari Parker | - | - | Jabari Parker | - | - | Jabari Parker | - |
| $A_t$ | First Name | Last Name | - | Player Pts | - | - | Player Reb | - | - | Player Ast | - |
| $N_t$ | - | - | - | 0 | - | - | 1 | - | - | 1 | - |

Table 3.3: A running example of our model's generation process. At every time step $t$, the model predicts each random variable. The model firstly determines if data records are to be referred to ($Z_t = 1$) or not ($Z_t = 0$). If random variable $Z_t = 1$, model selects entity $E_t$, its attribute $A_t$, and binary variables $N_t$ if needed. For example, at $t = 202$, the model predicts random variable $Z_{202} = 1$ and then selects the entity **Jabari Parker** and its attribute **Player Pts**. Given these values, the model outputs token **15** from the selected data record.

Formally, we use four variables.

1. $Z_t$: A binary variable determining whether the model refers to input $\boldsymbol{x}$ at time step $t$ ($Z_t = 1$).

2. $E_t$: At each time step $t$, this variable indicates the salient entity (e.g., Hawks, LeBron James).

3. $A_t$: At each time step $t$, this variable indicates the salient attribute to be mentioned (e.g., Pts).

4. $N_t$: If attribute $A_t$ of the salient entity $E_t$ is a numeric attribute, this variable determines if a value in the data records should be output in Arabic numerals (e.g., 50) ($N_t = 1$) or in English words (e.g., five) ($N_t = 0$).

Our model predicts these random variables at each time step $t$ through its summary generation process to keep track of the salient entity. A running example of our model is shown in Table 3.3. For a concise description, we omit the condition for each probability notation. <BoS> and <EoS> represent "beginning of the summary" and "end of the summary," respectively.

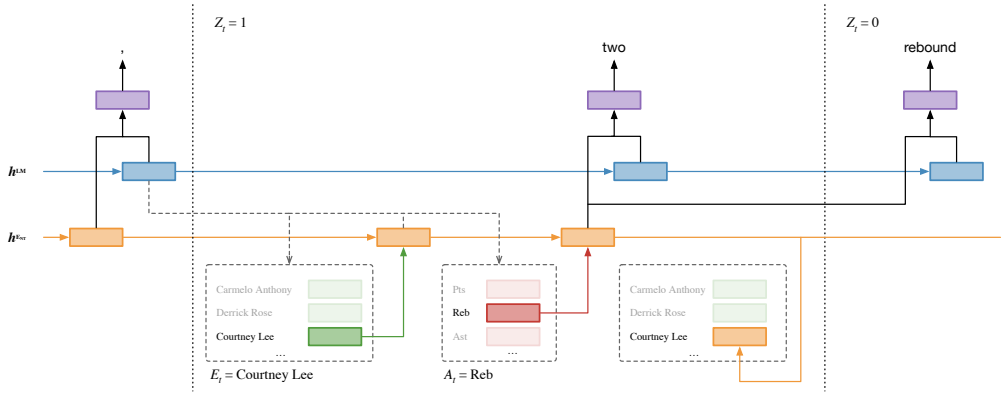



Figure 3.2: Illustration of generation process consisting of hidden states of language modeling, entity tracking, and dynamic entity representations as external memory.

In the following, we explain the model's initialization procedure, predict these random variables, and generate a summary. An illustration of our proposed model is shown in Figure 3.2.

Before discussing our method, we describe the notations used. Let $\mathcal{E}$ and $\mathcal{A}$ denote the sets of entities and attributes, respectively. Each record $r \in \boldsymbol{x}$ consists of entity $e \in \mathcal{E}$, attribute $a \in \mathcal{A}$, and its value $\boldsymbol{x}[e, a]$, and is therefore represented as $r = (e, a, \boldsymbol{x}[e, a])$. For example, the box-score in Table 3.1 has a record $r$ such that $e = \textsc{Anthony Davis}, a = \textsc{Pts}$, and $\boldsymbol{x}[e, a] = 20$.

## Initialization

Let $\boldsymbol{r} \in \mathbb{R}^d$ denote the embedding of data record $r \in \boldsymbol{x}$, where $d$ is the dimension of the embeddings. Let $\bar{\boldsymbol{e}}$ denote the embedding of entity $e$. Note that $\bar{\boldsymbol{e}}$ depends on the set of data records, i.e., it depends on the game. We also use $\boldsymbol{e}$ for the static embedding of entity $e$, which does not depend on the game.

Given the embedding of entity $\boldsymbol{e}$, attribute $\boldsymbol{a}$, and its value $\boldsymbol{v}$, we use the concatenation layer to combine the information from these vectors to produce the embedding of each data record $(e, a, v)$, denoted as $\boldsymbol{r}_{e,a,v}$ as follows.

$$\boldsymbol{r}_{e,a,v} = \tanh\left(\boldsymbol{e} + \boldsymbol{a} + \boldsymbol{v} + \boldsymbol{b}^{\mathrm{R}}\right), \tag{3.1}$$

where $\boldsymbol{b}^{\mathrm{R}} \in \mathbb{R}^d$ denotes a bias vector.[‡]

We obtain a game-dependent entity embedding $\bar{\boldsymbol{e}}$ in the set of data records $\boldsymbol{x}$, by summing all the data-record embeddings transformed by a matrix as follows

$$\bar{\boldsymbol{e}} = \tanh\left(\sum_{a \in \mathcal{A}} \boldsymbol{W}_a^{\mathrm{A}} \boldsymbol{r}_{e,a,\boldsymbol{x}[e,a]} + \boldsymbol{b}_a^{A}\right), \tag{3.2}$$

where $\boldsymbol{W}_a^{\mathrm{A}} \in \mathbb{R}^{d \times d}$ is a weight matrix, and $\boldsymbol{b}_a^{\mathrm{A}} \in \mathbb{R}^d$ is a bias vector for attribute $a$, respectively. As $\bar{\boldsymbol{e}}$ depends on the game as above, $\bar{\boldsymbol{e}}$ is supposed to represent how entity $e$ played in the game.

To initialize the hidden state of each module, we use embeddings of <SoD> for $\boldsymbol{h}^{\mathrm{LM}}$ and averaged embeddings of $\bar{\boldsymbol{e}}$ for $\boldsymbol{h}^{\mathrm{ENT}}$.

## Saliency Transition

Generally, the saliency of text changes during text generation. We assume that the saliency is represented as the entity and its attribute being discussed in our work. Therefore, we propose a model that refers to a data record at each timestep and transits to another as the text is processed.

To determine whether to transit to another data record at time $t$, the model calculates the following probability

$$p(Z_t = 1 \mid \boldsymbol{h}_{t-1}^{\mathrm{LM}}, \boldsymbol{h}_{t-1}^{\mathrm{ENT}}) = \sigma(\boldsymbol{w}_z^{\top}(\boldsymbol{h}_{t-1}^{\mathrm{LM}} \oplus \boldsymbol{h}_{t-1}^{\mathrm{ENT}}) + b_z), \tag{3.3}$$

where $\boldsymbol{w}_z \in \mathbb{R}^{2d}$ is a weight vector, $\oplus$ indicates the concatenation of vectors, and $\sigma(\cdot)$ is the sigmoid function. If $p(Z_t = 1 \mid \boldsymbol{h}_{t-1}^{\mathrm{LM}}, \boldsymbol{h}_{t-1}^{\mathrm{ENT}})$ is high, the model transits to another data record.

When the model decides to transit to another data record, it determines which entity and attribute to refer to and generates the next word. However, if the model decides not to, it generates the next word without updating the tracking states $\boldsymbol{h}_t^{\mathrm{ENT}} = \boldsymbol{h}_{t-1}^{\mathrm{ENT}}$.

## Selection and Tracking

When the model refers to a data record ($Z_t = 1$), it selects an entity and corresponding attribute. It also tracks the saliency by placing the information on the

[‡]We also add embedding vectors representing whether the entity is a home or away team.

selected entity and attribute into the tracking vector $\boldsymbol{h}^{\text{ENT}}$. The model selects the subject entity and updates the entity states if it changes.

Specifically, the model first calculates the probability of selecting an entity $e_t$ that is referred at time step $t$ as follows.

$$p(E_t = e \mid \boldsymbol{h}^{\text{LM}}_{t-1}, \boldsymbol{h}^{\text{ENT}}_{t-1}) \propto \begin{cases} \exp\left(\boldsymbol{h}^{\text{ENT}\top}_{s} \boldsymbol{W}^{\text{OLD}} \boldsymbol{h}^{\text{LM}}_{t-1}\right) & \text{if } e \in \mathcal{E}_{t-1} \\ \exp\left(\bar{\boldsymbol{e}}^{\top} \boldsymbol{W}^{\text{NEW}} \boldsymbol{h}^{\text{LM}}_{t-1}\right) & \text{otherwise} \end{cases}, \tag{3.4}$$

where $\boldsymbol{W}^{\text{OLD}} \in \mathbb{R}^{h \times h}$ and $\boldsymbol{W}^{\text{NEW}} \in \mathbb{R}^{d \times h}$ are weight matrices, $\mathcal{E}_{t-1}$ is the set of entities that have already been referred to by time step $t$, and $s$ is defined as $s = \max\{s : s \leq t-1, e = e_s\}$, indicating the time step when this entity was last mentioned.

The model selects the most probable entity as the next salient entity and updates the set of entities that have appeared ($\mathcal{E}_t = \mathcal{E}_{t-1} \cup \{e_t\}$).

If the salient entity changes ($e_t \neq e_{t-1}$), the model updates the hidden state of the tracking model $\boldsymbol{h}^{\text{ENT}}$ using a recurrent neural network with a gated recurrent unit (GRU) [19]:

$$\boldsymbol{h}^{\text{ENT}'}_{t} = \begin{cases} \boldsymbol{h}^{\text{ENT}}_{t-1} & \text{if } e_t = e_{t-1} \\ \text{GRU}^{\text{E}}(\bar{\boldsymbol{e}}, \boldsymbol{h}^{\text{ENT}}_{t-1}) & \text{else if } e_t \notin \mathcal{E}_{t-1} \\ \text{GRU}^{\text{E}}(\boldsymbol{W}^{\text{S}} \boldsymbol{h}^{\text{ENT}} + \boldsymbol{b}^{\text{S}}, \boldsymbol{h}^{\text{ENT}}_{t-1}) & \text{otherwise,} \end{cases} \tag{3.5}$$

where $\boldsymbol{h}^{\text{ENT}'}_{t}$ is the intermediate entity tracking representation, $\boldsymbol{W}^{\text{S}} \in \mathbb{R}^{d \times h}$ is a weight matrix, and $\boldsymbol{b}^{\text{S}} \in \mathbb{R}^{d}$ is a bias vector. Notably, if the selected entity at time step $t$, $e_t$, is identical to the previously selected entity $e_{t-1}$, the hidden state of the tracking model is not updated. If the selected entity $e_t$ is new ($e_t \notin \mathcal{E}_{t-1}$), the hidden state of the tracking model is updated with the embedding $\bar{\boldsymbol{e}}$ of entity $e_t$ as input. In contrast, if entity $e_t$ has already appeared in the past ($e_t \in \mathcal{E}_{t-1}$) but is not identical to the previous ($e_t \neq e_{t-1}$), we use $\boldsymbol{h}^{\text{ENT}}_{s}$ (i.e., the tracking state when this entity last appeared) to fully exploit the local history of this entity.

Given the updated hidden state of the tracking model $\boldsymbol{h}^{\text{ENT}'}_{t}$, we next select the attribute of the salient entity at time $t$, $a_t$, using the following probability.

$$\begin{aligned} &p(A_t = a \mid e_t, \boldsymbol{h}^{\text{LM}}_{t-1}, \boldsymbol{h}^{\text{ENT}'}_{t}) \\ &\propto \exp\left(\boldsymbol{r}^{\top}_{e_t, a, \boldsymbol{x}[e_t, a]} \boldsymbol{W}^{\text{ATTR}} (\boldsymbol{h}^{\text{LM}}_{t-1} \oplus \boldsymbol{h}^{\text{ENT}'}_{t})\right), \end{aligned} \tag{3.6}$$

where $\boldsymbol{W}^{\text{ATTR}} \in \mathbb{R}^{d \times 2h}$ is a weight matrix. After selecting $a_t$, i.e., the most probable attribute of the salient entity, the tracking model updates the state $\boldsymbol{h}_t^{\text{ENT}}$ with the embedding of the data record $\boldsymbol{r}_{e_t,a_t,\boldsymbol{x}[e_t,a_t]}$ as given below.

$$\boldsymbol{h}_t^{\text{ENT}} = \text{GRU}^{\text{A}}(\boldsymbol{r}_{e_t,a_t,\boldsymbol{x}[e_t,a_t]}, \boldsymbol{h}_t^{\text{ENT}'}). \tag{3.7}$$

Note that we used different GRU units ($\text{GRU}^{\text{E}}$ and $\text{GRU}^{\text{A}}$) to update the tracking model because they accept different types of embeddings for the input.

As described in Fig. 4.1, embeddings of previously selected entities could be seen as being overwritten by the most recently referred states of entity tracking modules $\boldsymbol{h}_s^{\text{ENT}}$. Hence, from this point of view, we can consider each entity's embeddings as external memories updated by the tracking module states $\boldsymbol{h}_s$.

## Summary Generation

Given the two hidden states, one for language model $\boldsymbol{h}_{t-1}^{\text{LM}}$ and the other for tracking model $\boldsymbol{h}_t^{\text{ENT}}$, the model generates the next word $y_t$. We also incorporate a copy mechanism that copies the value of the salient data record $\boldsymbol{x}[e_t, a_t]$.

If the model refers to a data record ($Z_t = 1$), it directly copies the value of the data record $\boldsymbol{x}[e_t, a_t]$. However, the values of numerical attributes can be expressed in at least two different styles: Arabic numerals (e.g., *14*) and English words (e.g., *fourteen*). A straightforward solution would be to follow the basic English rule that uses English words for single digits and Arabic numerals for double digits. However, in real descriptions, we found many irregular usages that did not conform to the basic English rule. Thus, we decide which one to use by the following probability.

$$p(N_t = 1 \mid \boldsymbol{h}_{t-1}^{\text{LM}}, \boldsymbol{h}_t^{\text{ENT}}) = \sigma(\boldsymbol{w}^{\text{N}\top}(\boldsymbol{h}_{t-1}^{\text{LM}} \oplus \boldsymbol{h}_t^{\text{ENT}}) + b^{\text{N}}), \tag{3.8}$$

where $\boldsymbol{w}^{\text{N}} \in \mathbb{R}^{2h}$ is a weight vector. The model then updates the hidden states of the language model as follows.

$$\boldsymbol{h}'_t = \tanh\left(\boldsymbol{W}^{\text{H}}(\boldsymbol{h}_{t-1}^{\text{LM}} \oplus \boldsymbol{h}_t^{\text{ENT}}) + \boldsymbol{b}^{\text{H}}\right), \tag{3.9}$$

where $\boldsymbol{W}^{\text{H}} \in \mathbb{R}^{h \times 2h}$ is a weight matrix, and $\boldsymbol{b}^{\text{H}} \in \mathbb{R}^h$ is a bias vector.

If the model does not refer to a data record ($Z_t = 0$), it predicts the next word $y_t$ via a probability over words conditioned on the context vector $\boldsymbol{h}'_t$.

$$p(Y_t \mid \boldsymbol{h}'_t) = \text{softmax}(\boldsymbol{W}^{\text{Y}} \boldsymbol{h}'_t), \tag{3.10}$$

where $\boldsymbol{W}^{\text{Y}} \in \mathbb{R}^{|\mathcal{V}| \times h}$ is a weight matrix and $|\mathcal{V}|$ is the size of the vocabulary.

Subsequently, the hidden state of language model $\boldsymbol{h}^{\text{LM}}$ is updated in the same manner.

$$\boldsymbol{h}^{\text{LM}}_t = \text{LSTM}(\boldsymbol{y}_t \oplus \boldsymbol{h}'_t, \boldsymbol{h}^{\text{LM}}_{t-1}), \tag{3.11}$$

where $\boldsymbol{y}_t \in \mathbb{R}^d$ is the embedding of the word generated at time step $t$.[§]

### 3.2.3 Incorporating Writer Information

For the document-level data-to-text generation task, we hypothesize that each writer would select different entity information from the table and describe the summary in their writing style. To test our hypothesis, we also incorporate the writers' information into our model to study how external style information affects the language generation model. Specifically, instead of using Equation (3.9), we concatenate the embedding $\boldsymbol{w}$ of a writer to $\boldsymbol{h}^{\text{LM}}_{t-1} \oplus \boldsymbol{h}^{\text{Ent}}_t$ to construct context vector $\boldsymbol{h}'_t$

$$\boldsymbol{h}'_t = \tanh\left(\boldsymbol{W}'^{\text{H}}(\boldsymbol{h}^{\text{LM}}_{t-1} \oplus \boldsymbol{h}^{\text{Ent}}_t) + \boldsymbol{w} + \boldsymbol{b}'^{\text{H}}\right). \tag{3.12}$$

Because this new context vector $\boldsymbol{h}'_t$ is used to calculate the probability over words in Equation (3.10), the writer information is expected to directly affect word generation, which is regarded as surface realization in terms of traditional text generation. Simultaneously, context vector $\boldsymbol{h}'_t$ enhanced with the writer information is used to obtain $\boldsymbol{h}^{\text{LM}}_t$, the hidden state of the language model, and is further used to select the salient entity and attribute as mentioned. Therefore, in our model, the writer information affects both surface realization and content planning.

[§]In our initial experiment, we observed a word repetition problem when the tracking model was not updated while generating each sentence. To avoid this problem, we also update the tracking model with special trainable vectors $\boldsymbol{v}_{\text{REFRESH}}$ to refresh these states as follows after our model generates a period.$\boldsymbol{h}^{\text{Ent}}_t = \text{Gru}^A(\boldsymbol{v}_{\text{Refresh}}, \boldsymbol{h}^{\text{Ent}}_t)$

### 3.2.4 Training

We applied a fully supervised training maximizing the following log-likelihood.

$$\log p(Y_{1:T}, Z_{1:T}, E_{1:T}, A_{1:T}, N_{1:T} \mid \boldsymbol{x}) = \underbrace{\sum_{t=1}^{T} \log p(Z_t = z_t \mid \boldsymbol{h}_{t-1}^{\mathrm{LM}}, \boldsymbol{h}_{t-1}^{\mathrm{ENT}})}_{\text{Switching objective}}$$
$$+ \underbrace{\sum_{t:Z_t=1} \log p(E_t = e_t \mid \boldsymbol{h}_{t-1}^{\mathrm{LM}}, \boldsymbol{h}_{t-1}^{\mathrm{ENT}})}_{\text{Entity selection objective}} + \underbrace{\sum_{t:Z_t=1} \log p(A_t = a_t \mid e_t, \boldsymbol{h}_{t-1}^{\mathrm{LM}}, \boldsymbol{h}_{t}^{\mathrm{ENT}'})}_{\text{Attribute selection objective}}$$
$$+ \underbrace{\sum_{t:Z_t=1, a_t \text{is num\_attr}} \log p(N_t = n_t \mid \boldsymbol{h}_{t-1}^{\mathrm{LM}}, \boldsymbol{h}_{t}^{\mathrm{ENT}})}_{\text{Number selection objective}} + \underbrace{\sum_{t:Z_t=0} \log p(Y_t = y_t \mid \boldsymbol{h}'_t)}_{\text{Conditional language model objective}}$$

## 3.3 Experiments

### 3.3.1 Settings

We used the RotoWire-Modified and RotoWire-FG datasets in our experiments, as described in Section 3.1. We also used the recently introduced dataset, RotoWire-FG [114], which is an enlarged, enriched, and purified dataset based on the original RotoWire dataset. The most significant difference with the RotoWire-Modified is the inclusion of unsupported sentences by the box-scores. In the RotoWire-FG dataset, all of the numbers written in English words are converted into Arabic numbers. Thus, we do not have to introduce the random variable $N_t$ to determine the selected number's surface realization for this dataset. We used the official split for training, development, and test data, respectively.

As we assume a supervised training approach, we need the annotations of the random variables (i.e., $Z_t$, $E_t$, $A_t$, and $N_t$) in the training data, as shown in Table 3.3. Instead of simple lexical matching with $r \in \boldsymbol{x}$, which is prone to errors in the annotation, we use information extraction systems. For RotoWire-Modified dataset, we used the information extraction model provided by Wiseman et al. [122] and for RotoWire-FG dataset, we used the hard-coded information retrieval system provided by Wang [114]. The former model shows 93.4% for precision and 75.0% for recall, and the latter shows 98% for precision and 95%

for recall. These annotation performances directly affect model performance. Although the model-based system is trained on noisy rule-based annotations, we conjecture that it is more robust to errors because it is trained to minimize the marginalized loss function for ambiguous relations. However, the model only considers sentence-level relation extractions, so it fails to extract relations across sentences.

We set the size of embeddings to 128 and those of the hidden states to 512. All of the parameters were initialized with the Xavier initialization [35]. We set the maximum number of epochs to 30 and chose the model with the highest BLEU score on the development data. The initial learning rate is 2e-3, and AMSGrad was also used for automatically adjusting the learning rate [97]. Our implementation used DyNet [82].

### 3.3.2 Baseline comparison

We compare our model[¶] against the following baseline models.

- ED-CC: this model consists of an attention-based encoder-decoder model with a conditional copy mechanism used by Wiseman et al. [122].
- HIERARCHICAL: this model consists of an encoder with hierarchical attention proposed by Rebuffel et al. [96]
- NCP: this model first predicts the sequence of data records and then generates a summary conditioned on the predicted sequences, proposed by Puduppully et al. [91].
- ENT: this model is contemporary with our proposed model extending the encoder-decoder model with memory modules as proposed by Puduppully et al. [92].

ED-CC model refers to all data records every timestep, while NCP model refers to a subset of all data records, which is predicted in the first stage. Unlike these models, our model uses one tracking vector $\boldsymbol{h}_t^{\text{ENT}}$ tracking the history of the data records during generation. We retrained the baselines on our new dataset. We

[¶]Our code is available from `https://github.com/aistairc/sports-reporter`

also present the performance of GOLD and TEMPLATES summaries. The GOLD summary is identical with the reference summary, and each TEMPLATES summary is generated in the same manner as Wiseman et al. [122].

In the latter half of our experiments, we examine the effect of adding information on writers. In addition to our model enhanced with writer information, we include writer information to the NCP model by Puduppully et al. [91]. Their method consists of two stages corresponding to content planning and surface realization. Therefore, by incorporating writer information in each of the two stages, we can distinguish which part of the writer's model contributes. For NCP model, we attach the writer information in the following three ways:

1. concatenating writer embedding $\boldsymbol{w}$ with the input vector for LSTM in the content planning decoder (stage 1);
2. concatenating writer embedding $\boldsymbol{w}$ with the input vector for LSTM in the text generator (stage 2);
3. using both 1 and 2 above.

For more details about each decoding stage, readers can refer to Puduppully et al. [91].

### 3.3.3 Evaluation Metrics

As evaluation metrics, we use BLEU score [88] and the extractive metrics proposed by Wiseman et al. [122], i.e., relation generation (RG), content selection (CS), and content ordering (CO). The extractive metrics measure how well the relations extracted from the generated summary match the correct relations[‖]:

- RG: the ratio of the correct relations to all the extracted relations, where correct relations refer to those found in the input data records $\boldsymbol{x}$. Furthermore, the average number of extracted relations is reported.

[‖]The model for extracting relation tuples was trained on tuples made from the entities (e.g., team name, city name, player name) and attribute values (e.g., "Lakers", "92") extracted from the summaries, and the corresponding attributes (e.g., "TEAM NAME", "PTS") found in the box- or line-scores. The precision and the recall of this extraction model are respectively 93.4% and 75.0% on the test data.

| Model | RG | | CS | | | CO | Bleu |
|---|---|---|---|---|---|---|---|
| | # | P% | P% | R% | F1% | DLD% | |
| Gold | 27.36 | 93.42 | 100. | 100. | 100. | 100. | 100. |
| Templates | 54.63 | 100. | 31.01 | 58.85 | 40.61 | 17.50 | 8.43 |
| ED-CC | 20.28 | 61.76 | 27.20 | 29.76 | 28.42 | 15.88 | 15.26 |
| Hierarchical | 26.34 | 78.74 | 35.74 | 39.04 | 37.32 | 20.36 | 15.40 |
| ENT | 32.41 | 91.13 | 37.05 | 43.06 | 39.83 | 20.62 | 15.23 |
| NCP | 34.05 | 82.55 | 32.30 | 43.74 | 37.16 | 16.67 | 14.82 |
| NCP+TR | 29.39 | 83.64 | 36.10 | 41.57 | 38.64 | 18.53 | 14.27 |
| Proposed | 31.66 | **91.98** | **40.44** | **46.63** | **43.31** | **21.56** | **15.74** |

Table 3.4: Experimental result on the RotoWire-Modified. Each metric evaluates whether important information (CS) is described accurately (RG) and in correct order (CO). TR stands for table reconstruction.

- CS: precision and recall of the relations extracted from the generated summary against those from the reference summary.
- CO: edit distance measured with normalized Damerau-Levenshtein Distance (DLD) between the sequences of relations extracted from the generated and reference summaries.

### 3.3.4 Results

We first focus on the tracking model's quality and entity representation, using the model without writer information. Then we examine the effect of writer information.

#### Saliency Tracking-Based Model

As shown in Table 3.4, our model outperforms all baselines across all evaluation metrics on RotoWire-Modified. While ENT shows similar performance, our

| Model | RG | | CS | | | CO | Bleu |
|---|---|---|---|---|---|---|---|
| | # | P% | P% | R% | F1% | DLD% | |
| Gold | 29.36 | 95.17 | 100. | 100. | 100. | 100. | 100. |
| Templates† | 51.80 | 98.89 | 23.98 | 43.96 | 31.03 | 10.25 | 12.09 |
| ED-CC† | 30.28 | 82.16 | 35.84 | 38.40 | 37.08 | 18.45 | 20.80 |
| Hierarchical | 34.83 | 88.33 | 38.57 | 47.52 | 42.58 | 18.57 | 21.64 |
| ENT† | 35.69 | 93.72 | 39.04 | 49.29 | 43.57 | 17.50 | 21.23 |
| NCP† | 35.99 | 94.21 | 43.31 | 55.15 | 48.52 | 23.46 | 23.86 |
| NCP+ TR† | 37.49 | 95.70 | 42.90 | 56.91 | 48.92 | **24.47** | **24.41** |
| Proposed | 40.74 | **98.36** | **43.38** | **62.53** | **51.22** | 20.64 | 24.23 |

Table 3.5: Experimental result on the RotoWire-FG. † are taken from Wang [114]. TR stands for table reconstruction.

model achieves noticeable improvement on the F1 score for content selection.

We also show the result on the RotoWire-FG dataset in Table 3.5. Our model outperforms the relation generation and the content selection scores and shows the second-highest performance on the Bleu score. However, the NCP+TR model clearly shows the best performance on the content ordering and Bleu score.

These two experimental results demonstrate the following properties. (1) The entity-centric models, which are the ENT model and our proposed model, perform better than the baselines under the noisy data-to-text setting (RotoWire-Modified). (2) Explicit annotation, used by the NCP model and our proposed model, is beneficial under the clean data-to-text setting (RotoWire-FG). In other words, our proposed model can be perceived as having the best characteristics of both the ENT and the NCP models.


Figure 3.3: Illustrations of static entity embeddings $\boldsymbol{e}$. Players with colored letters are listed in the ranking top 100 players for the 2016-17 NBA season. Only *LeBron James* is in red and the other players in top 100 are in blue. Top-ranked players have similar representations of $\boldsymbol{e}$.

## Qualitative Analysis of Entity Embedding

Our model has the game-dependent entity embedding $\bar{\boldsymbol{e}}$, which depends on the box score for each game in addition the to static entity embedding $\boldsymbol{e}$. Now we analyze the difference between these two types of embeddings.

We present a two-dimensional visualization of both embeddings produced using principal component analysis (PCA) [89]. As shown in Figure 3.3, which is the visualization of static entity embedding $\boldsymbol{e}$, the top-ranked players are located close to one another.

We also present visualizations of game-dependent entity embeddings $\bar{\boldsymbol{e}}$ in Figure 3.4. Although we did not carry out feature engineering specific to the NBA (e.g., whether a player scored double-digits)** for the game-dependent entity embedding $\bar{\boldsymbol{e}}$, the embeddings of the players who performed well for each game have similar representations. Furthermore, changes in the same player's embeddings were observed depending on the box-scores for each game. For instance, *Le-*

**In the NBA, a player who accumulates a *double*-digit score in one of five categories (points, rebounds, assists, steals, and blocked shots) in a game, is regarded as a good player, if a player had a double in two of those five categories, it is referred to as *double-double*.

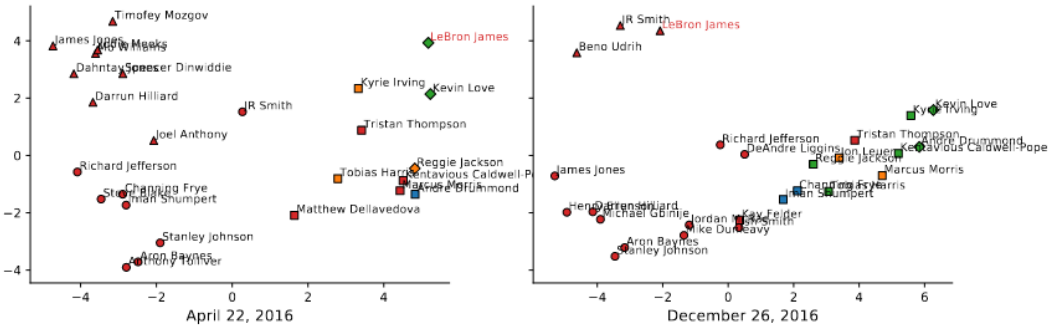



Figure 3.4: Illustrations of game-dependent entity embedding $\bar{\boldsymbol{e}}$. Both left and right figures are for *Cleveland Cavaliers* vs. *Detroit Pistons*, on different dates. LeBron James is in **red letters**. **Entities with orange symbols** appeared only in the reference summary. **Entities with blue symbols** appeared only in the generated summary. **Entities with green symbols** appeared in both the reference and the generated summary. The others are with **red symbols**. □ represents a player who scored double digits, and ◇ represents a player who recorded a double-double. Players with △ did not participate in the game. ○ represents other players.

*Bron James* recorded a double-double in a game on April 22, 2016. Then, the embedding is located close to the embedding of *Kevin Love*, who also scored a double-double. However, he did not participate in the game on December 26, 2016. His embedding for this game became closer to those of other players who also did not participate.

## Duplicate Ratios of Extracted Relations

As Puduppully et al. [91] pointed out, a generated summary may mention the same relation multiple times. Such duplicated relations are not favorable in terms of the brevity of the text.

Figure 3.5 shows the ratios of the generated summaries with duplicate mentions of relations in the development data. While the models by Wiseman et al. [122] and Puduppully et al. [91] respectively showed 36.0% and 15.8% as dupli-

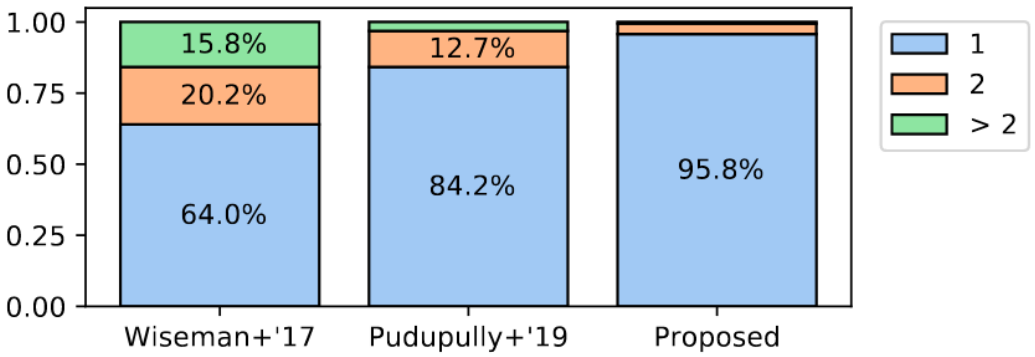



Figure 3.5: Ratios of generated summaries with duplicate mentions of relations. Each label represents a number of duplicated relations within each summary. While Wiseman et al. [122]'s model exhibited 36.0% duplication and Puduppully et al. [91]'s model exhibited 15.8%, our model exhibited only 4.2%.

cate ratios, our model exhibited 4.2%, suggesting that our model dramatically suppressed the generation of redundant relations. We speculate that the tracking model successfully memorized which of the input records were selected in $\boldsymbol{h}_s^{\text{ENT}}$.

## Qualitative Analysis of Output Examples

Table 3.6 shows the generated examples from validation inputs with Puduppully et al. [91]'s model and our model. Although both appear to be fluent, the summary of Puduppully et al. [91]'s model includes erroneous relations colored in orange.

More precisely, for example, the description of Derrick Rose 's relations, "15 points, four assists, three rounds and one steal in 33 minutes.", is also used for other entities (e.g., John Henson and Willy Hernagomez). This is because Puduppully et al. [91] 's model has no tracking module, unlike our model, which mitigates redundant references and, therefore, rarely contains erroneous relations.

However, when complicated expressions such as parallel structures are used, our model also generates erroneous relations, as illustrated by the underlined sentences describing two players who scored the same number of points. For

The **Milwaukee Bucks** defeated the **New York Knicks**, **105-104**, at Madison Square Garden on Wednesday evening. The **Bucks** (**18-16**) have been one of the hottest teams in the league, having won five of their last six games, and they have now won six of their last eight games. The **Knicks** (**16-19**) have now won six of their last six games, as they continue to battle for the eighth and final playoff spot in the Eastern Conference. **Giannis Antetokounmpo** led the way for Milwaukee, as he tallied **27** points, **13** rebounds, **four** assists, **three** blocked shots and **one** steal, in **39** minutes . **Jabari Parker** added **15** points, **four** rebounds, **three** assists, **one** steal and **one** block, and **6**-of-**8** from long range. **John Henson** added **two** points, **two** rebounds, **one** assist, **three** steals and **one** block. **John Henson** was the only other player to score in double digits for the Knicks, with **15** points, **four** assists, **three** rebounds and **one** steal, in **33** minutes. The Bucks were led by **Derrick Rose**, who tallied **15** points, **four** assists, **three** rebounds and **one** steal in **33** minutes. **Willy Hernangomez** started in place of Porzingis and finished with **15** points, **four** assists, **three** rebounds and **one** steal in **33** minutes. **Willy Hernangomez** started in place of Jose Calderon ( knee ) and responded with **one** rebound and **one** block. The Knicks were led by their starting backcourt of **Carmelo Anthony** and **Carmelo Anthony**, but combined for just **13** points on 5-of-16 shooting. The Bucks next head to Philadelphia to take on the Sixers on Friday night, while the Knicks remain home to face the Los Angeles Clippers on Wednesday.

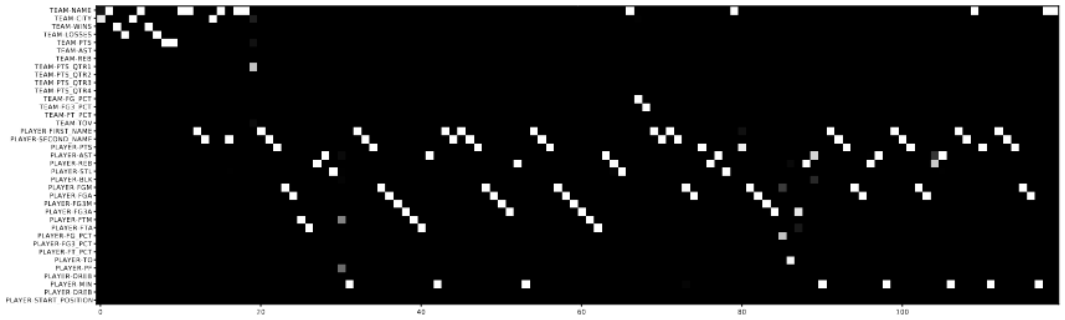
(a) Puduppully et al. [91]

The **Milwaukee Bucks** defeated the **New York Knicks**, **105-104**, at Madison Square Garden on Saturday. The **Bucks** (**18-16**) checked in to Saturday's contest with a well, outscoring the **Knicks** (**16-19**) by a margin of 39-19 in the first quarter. However, New York by just a 25-foot lead at the end of the first quarter, the Bucks were able to pull away, as they outscored the Knicks by a 59-46 margin into the second. 45 points in the third quarter to seal the win for New York with the rest of the starters to seal the win. The Knicks were led by **Giannis Antetokounmpo**, who tallied a game-high **27** points, to go along with **13** rebounds, **four** assists, **three** blocks and a steal. The game was a crucial night for the Bucks' starting five, as the duo was the most effective shooters, as they posted Milwaukee to go on a pair of low low-wise (Carmelo Anthony) and Malcolm Brogdon. **Anthony** added **11** rebounds, **seven** assists and **two** steals to his team-high scoring total. **Jabari Parker** was right behind him with **15** points, **four** rebounds, **three** assists and a block. **Greg Monroe** was next with a bench-leading **18** points, along with **nine** rebounds, **four** assists and **three** steals. **Brogdon** posted **12** points, **eight** assists, **six** rebounds and a steal. **Derrick Rose** and **Courtney Lee** were next with a pair of {**11** / **11**} -point efforts. **Rose** also supplied **four** assists and **three** rebounds, while **Lee** complemented his scoring with **three** assists, a rebound and a steal. **John Henson** and **Mirza Teletovic** were next with a pair of {**two** / **two**} -point efforts. **Teletovic** also registered **13** points, and he added a rebound and an assist. **Jason Terry** supplied **eight** points, **three** rebounds and a pair of steals. The Cavs remain in last place in the Eastern Conference's Atlantic Division. They now head home to face the Toronto Raptors on Saturday night.

(b) Our model

Table 3.6: Example summaries generated with Puduppully et al. [91]'s model (left) and our model (right). Names in **bold face** are salient entities. **Blue numbers** are correct relations derived from input data records but are not observed in reference summary. **Orange numbers** are incorrect relations. **Green numbers** are correct relations mentioned in reference summary.

(a) Transition of probabilities of entity $p(E_t = e)$.



(b) Transition of probabilities of attribute $p(A_t = a)$.

Figure 3.6: Probability sequences for entity and attribute selection.

example, "11-point efforts" is correct for Courtney Lee but not for Derrick Rose. As a direction for future research, it is necessary to develop a method to handle such complicated relations.

We also show the sequences of probabilities of selecting an entity $p(E_t = e_t)$ and an attribute $p(A_t = a_t)$ in Figure 3.6. The model first refers to the game's general information and then starts to look for salient players. Once the model selects a player, it consistently selects the same player until it finishes mentioning this player's important information. After this, the model rarely refers to that entity, suggesting that the model successfully learns to guide the consistent entity reference and maintains its saliency discussed above.

## Use of writer information

We first look at the results of an extension of Puduppully et al. [91]'s model with writer information $\boldsymbol{w}$ in Table 3.7. By adding $\boldsymbol{w}$ to content planning (stage 1),

| Method | RG | | CS | | | CO | BLEU |
|---|---|---|---|---|---|---|---|
| | # | P% | P% | R% | F1% | DLD% | |
| NCP | 34.05 | 82.55 | 32.30 | 43.74 | 37.16 | 16.67 | 14.82 |
| + $\boldsymbol{w}$ in stage 1 | 30.26 | **85.54** | 42.33 | **49.38** | 45.58 | 21.26 | 18.01 |
| + $\boldsymbol{w}$ in stage 2 | 32.42 | 83.35 | 33.28 | 42.92 | 37.49 | 16.73 | 16.57 |
| + $\boldsymbol{w}$ in stage 1 & 2 | 28.16 | 84.09 | **43.63** | 47.75 | **45.60** | **21.96** | **18.57** |
| PROPOSED | 31.66 | 91.98 | 40.44 | 46.63 | 43.31 | 21.56 | 15.74 |
| + $\boldsymbol{w}$ | 29.44 | **93.32** | **51.76** | **55.21** | **53.42** | **24.97** | **20.62** |

Table 3.7: Effects of writer information. $\boldsymbol{w}$ indicates that WRITER embeddings are used. Numbers in **bold** are the largest among the variants of each method.

the method obtained improvements in CS (37.16 to 45.58), CO (16.67 to 21.26), and BLEU score (14.82 to 18.01). By adding $\boldsymbol{w}$ to the component for surface realization (stage 2), the method obtained an improvement in BLEU score (14.82 to 16.57), while the effects on the other metrics were not very significant. By adding $\boldsymbol{w}$ to both stages, the method scored the highest BLEU, while the other metrics were not significantly different from those obtained by adding $\boldsymbol{w}$ to stage 1. This result suggests that writer information contributes to content planning and surface realization when appropriately used, and improvements in content planning lead to much better surface realization performance.

Our model showed improvements in all metrics and showed the best performance by incorporating writer information $\boldsymbol{w}$. As discussed in Section 3.2.3, $\boldsymbol{w}$ is supposed to affect both content planning and surface realization. Our experimental result is consistent with the discussion.

We show generated examples of the PROPOSED model with and without writer information in Table 3.8. The summary generated by the PROPOSED model with writer information shares more similar vocabulary usage with the GOLD summary than the summary generated by the PROPOSED model without writer information.

| | Summary | Bleu |
|---|---|---|
| Gold | The **Denver Nuggets** defeated the host **Orlando Magic**, 121-113, at Amway Center on Saturday. In a game between two struggling teams, Orlando could n't quite keep up in their fourth game in five nights. That was clearly evident late, as the Nuggets ended the game on a 17-5 run to pull off the comeback. While the defense struggled all night, Denver held Orlando to just 18 points in the fourth quarter, while scoring 29 points on their end. The Nuggets offense was on point, as they shot 57 percent from the field and recorded 25 assists in the victory. Rebounding was key as well, with Denver winning that battle, 44-31. Denver (9-15) rode one of their best offensive performances here to an impressive road win. While no player on the Nuggets had more than 21 points, they had six players score at least 14. **Danilo Gallinari** led the team in scoring, tallying 21 points, six rebounds and three assists. **Jusuf Nurkic** scored 17 points and grabbed eight boards on 8-of-11 shooting in just 20 minutes of action. **Kenneth Faried** was huge off the bench, dropping 19 points and 10 rebounds on 9-of-10 from the field. **Nikola Jokic** and **Juancho Hernangomez** each scored {14/14} a piece off the bench. On deck for the Nuggets is a road matchup against the Dallas Mavericks on Monday.. **Orlando** (10-15) had to play this game without Nikola Vucevic and it clearly hurt their defense. **Evan Fournier** did all he could against his former team, as he led the Magic with 24 points, four rebounds and six assists. **Serge Ibaka** was n't far behind, accumulating 22 points, four rebounds and four blocks in the defeat. **Jeff Green** was questionable coming in to this game with an ankle injury, but provided 17 points off the bench. Orlando 's next game is on Tuesday in a road matchup against the Atlanta Hawks. | – |
| Proposed | The **Denver Nuggets** defeated the host **Orlando Magic**, 121-113, at Amway Center on Wednesday. The Nuggets (9-15) checked in to Friday with only seven road wins in 23 tries per game and had a road win behind the **Magic** (10-15). However, their second-highest scorer in Kenneth Faried 's head-high 21 combined attempts from the T-Wolves. He went off to grab a win for two of the Nuggets ' points in the victory, despite shooting 57 percent from the field and 42 percent from three-point range. They did turn up on the defensive side of the ball, turning the ball over two times and a pair of rebounds of an impressive 19 points. **Faried** supplemented his total double-double on the night, tallying 10 boards and dishing out one assist. **Danilo Gallinari** and **Will Barton** were next with a pair of {16/16}-point efforts. **Gallinari** added six rebounds, and three assists, while the backcourt was also a perfect night for the starting shooting night. **Nikola Jokic** continued his strong play in reserve roles, supplying 14 points, six rebounds, six assists and a block. **Jusuf Nurkic** continued his stretch of strong play down low, posting 17 points, eight rebounds, one block and one block. **Evan Fournier** continued his recent stretch of strong play with 24 points, four rebounds, six assists and one steal. **Serge Ibaka** totaled 22 points, four rebounds, one steal and four blocks. **Aaron Gordon** provided 11 points, six boards, three assists and one steal. **Bismack Biyombo** tallied 12 points, six rebounds, four assists and one block. **Barton** added four rebounds and a block, while **Juan Hernangomez** and **Jeff Green** supplied matching {14/14}-point efforts, with the former also hauling in four boards and recording zero boards, and the latter posting four rebounds and one assist. The Nuggets head back home to tangle with the Philadelphia 76ers on Wednesday evening, while the Magic remain home to take on the Magic on Friday evening as well. | 22.03 |
| Proposed + $w$ | The **Denver Nuggets** defeated the host **Orlando Magic**, 121-113, at Amway Center on Wednesday. The Nuggets were expected to win this game easily, but they were n't able to prevail with a win. In fact, the Nuggets led by 10 after the first quarter, outscoring Orlando, 33-14. Defense was key for the Nuggets, as they held the **Magic** to 49 percent from the field and 5-of-28 from three-point range. Los Angeles were able to pull out a huge rebound, as they held the **Magic** to just 18 points in the final quarter. The Nuggets (9-15) have been one of the best teams in the league right now and they simply could n't get enough of a rhythm to win. **Danilo Gallinari** led the team in scoring, as he tallied 21 points, six rebounds and three assists on 5-of-8 shooting. **Kenneth Faried** was second on the team, as he accrued 19 points and 10 rebounds on 9-of-10 shooting off the bench. **Jusuf Nurkic** was second on the team, with his 17 points and eight rebounds on 8-of-11 shooting. **Juan Hernangomez** was a nice spark off the bench, as he amassed 14 points, four rebounds and two assists. On deck for Denver is a home matchup against the Los Angeles Lakers on Wednesday. The **Magic** (10-15) have been one of the surprises of the season and a comeback performance from their worst team. **Evan Fournier** had his best game of the season, as he tallied 24 points and four rebounds on 9-of-19 shooting. **Serge Ibaka** was second on the team, as he accrued 22 points, four rebounds and four blocks. **Aaron Gordon** was a nice spark off the bench, providing 11 points, six rebounds and three assists. Orlando will look to keep rolling on Monday in a home matchup against the Cleveland Cavaliers. | 30.58 |

Table 3.8: Example summaries and their Bleu scores on the sample of the development set. **Blue numbers** are correct relations derived from input data records but are not observed in reference summary. **Orange numbers** are incorrect relations. **Green numbers** are correct relations mentioned in reference summary.

| Player | H/V | Pts | Reb | Ast | Blk | Stl | Min | City | ... |
|---|---|---|---|---|---|---|---|---|---|
| LeBron James | H | 33 | 11 | 11 | 0 | 0 | 33 | Cleveland | ... |
| Will Barton | V | 27 | 4 | 1 | 0 | 2 | 26 | Denver | ... |
| JR Smith | H | 15 | 4 | 5 | 0 | 4 | 29 | Cleveland | ... |
| Channing Frye | H | 14 | 3 | 2 | 1 | 0 | 25 | Cleveland | ... |
| ... | ... | ... | ... | ... | ... | ... | | | |

The **Cleveland Cavaliers** defeated the **Denver Nuggets** , **124** - **91** , at Quicken Loans Arena on Saturday . The Cavaliers were expected to win this game easily and they left no doubt with this result . The second quarter is when Cleveland really turned it on , as they outscored the Nuggets , 33 - 18 , in the second half . The **Nuggets** ( **29** - **42** ) had to play this game without Kevin Love and Kenneth Faried , but they did n't have enough bullets to make up with the rest of the game . **Cleveland** shot **56** percent from the field , while **Denver** shot **40** percent . The assist - to - turnover ratio was decisive as well , with **Cleveland** winning that battle , **52** - **32** . The Nuggets had a tough task here with a win . **Will Barton** led the team with **27** points and **four** rebounds , but he also had **two** steals and zero turnovers . **Jusuf Nurkic** was the only other player in double figures , as he tallied **11** points , **seven** rebounds , **three** assists and **three** blocks . The Nuggets will look to keep up this good form against the Los Angeles Clippers on Monday . The Cavaliers played this game without Kevin Love and Jameer Nelson and they simply didn ' t miss them . ***LeBron James*** *did all he could though , as he accumulated* *33* *points , 11 rebounds and 11 assists in the win .* **Channing Frye** also finished with **14** points and zero shots , while **Kyrie Irving** scored **eight** points in **24** minutes . Up next for the Cavaliers is a matchup against the Los Angeles Lakers on Monday .

The **Cleveland Cavaliers** ( **50** - **20** ) defeated the visiting **Denver Nuggets** ( **29** - **42** ) on Wednesday , **124** - **91** . The **Cavaliers** have now won three straight games and now sit two in a row . However , the team's top advantage is a strong one for the **Cavaliers** , as the team shot **56** percent from the field and hit **11** three - pointers . The Nuggets had a tough night on the glass , going just 5 - 37 on the night . The Cavaliers were able to maintain the lead on the boards , however , thanks in large part to strong offense in the second half of the season . Denver had been out of its previous three games with three games in a row . The Nuggets saw solid performances from their bench , as veteran **Will Barton** leading the team with **27** points . The team shot just 42 percent from the field and dished out 13 - of - 26 three - point attempts . Cleveland's offense was able to get off to a hot start in the second half , allowing the team to pull away with the second half of the season . The Cavaliers , meanwhile , have now lost seven of their last seven , and are now just one game ahead of the Clippers for the top draft seed in the Eastern Conference . The team will face the Clippers on Friday , while the Nuggets will host the Lakers on Wednesday .

Table 3.9: The generated summaries for the same game with different writer embeddings. **Green numbers** are relations that appeared in both summaries. **Orange numbers** are relations that appeared only in the above and **Red numbers** are relations that appeared only in the below.

In addition to the text style, the writer information is useful for choosing similar relations with the GOLD summary. Although the PROPOSED model without writer information mentions a more significant number of relations by generating a more extended summary, the PROPOSED model with writer information mentions a similar number of relations with the GOLD summary. Specifically, the GOLD summary contains the 31 relations and summaries generated by the PROPOSED model with or without writer information contain 46, 34 relations, and shares 20, 24 correct relations with those in the GOLD summary respectively.

We also examined the writer bias effects for the model. In particular, we examined the number of mentions of LeBron James, one of the most famous players in the NBA, by the model with each writer's information. The model generates summaries for all writers for all 36 games in which LeBron James played in the development set. As a result, about 70% of writers mentioned LeBron James more than 30 games. In particular, three of these writers describe LeBron in all games he played.

On the other hand, one of these writers mentioned LeBron James in just 13 out of 36 matches.

Considering the game shown in Figure 3.9 as an example, LeBron James is considered one of the most active players in the game because he scored the most points in this game. In the generated summaries using the different author information, the above summary mentions LeBron James in detail, while the below only mentions another active player, Will Barton.

We can also see that the summaries are written in different writing styles. This example also indicates the stylistic difference using the different writer information in the model.

## 3.4 Conclusion

This chapter proposed a new data-to-text model producing a summary text while tracking the salient information. As a result, our model outperformed the existing models in all evaluation measures on the ROTOWIRE-MODIFIED dataset. We also explored the effects of incorporating writer information into data-to-text models. With writer information, our model successfully generated the highest quality

summaries, scoring 20.62 points on BLEU score.

# 4 Fact-based Text Editing

Automatic editing of text by computers is an important application that can help human writers write better in terms of accuracy, fluency, etc. This task is easier and more practical than an automatic generation of texts from scratch and has been recently attracting attention [128, 131]. In this chapter, we consider a new and specific setting of this problem, referred to as *fact-based text editing*, in which a draft text and several facts (represented in triples) are given, and the system aims to revise the text by adding missing facts and deleting unsupported facts. Table 4.1 gives an example of the task.

As far as we know, no previous work has addressed this problem. In a text-to-text generation, given a text, the system automatically creates another text, where the new text can be a text in another language (machine translation), a summary of the original text (summarization), or a text in better form (text editing). In a table-to-text generation, given a table containing facts in triples, the system automatically composes a text describing the facts. The former is a text-to-text problem, and the latter a table-to-text problem. In comparison, fact-based text editing can be viewed as a 'text&table-to-text' problem.

First, we devise a method for automatically creating a dataset for fact-based text editing. Recently, several table-to-text datasets have been created and released, consisting of pairs of facts and corresponding descriptions. We leverage such data in our method. We first retrieve facts and their descriptions. Next, we consider the descriptions as revised texts and automatically generate draft texts based on the facts using several rules. We build two datasets for fact-based text editing on the basis of WebNLG [33] and RotoWire, consisting of 233k and 37k instances respectively [122] *.

Second, we propose a model for fact-based text editing called FactEditor.

*The datasets are publicly available at `https://github.com/isomap/factedit`

| **Set of triples** |
|---|
| {(**Baymax**, **creator**, **Douncan_Rouleau**), (**Douncan_Rouleau**, **nationality**, **American**), (**Baymax**, **creator**, **Steven_T._Seagle**), (**Steven_T._Seagle**, **nationality**, **American**), (**Baymax**, **series**, **Big_Hero_6**), (**Big_Hero_6**, **starring**, **Scott_Adsit**)} |
| **Draft text** |
| **Baymax** was created by **Duncan_Rouleau**, **a winner of Eagle_Award**. **Baymax** is a character in **Big_Hero_6** . |
| **Revised text** |
| **Baymax** was created by **American** creators **Duncan_Rouleau** and **Steven_T._Seagle** . **Baymax** is a character in **Big_Hero_6** which stars **Scott_Adsit** . |

Table 4.1: Example of fact-based text editing. Facts are represented in triples. The **facts in green** appear in both draft text and triples. The **facts in orange** are present in the draft text, but absent from the triples. The **facts in blue** do not appear in the draft text, but do in the triples. The task of *fact-based text editing* is to edit the draft text on the basis of the triples, by deleting **unsupported facts** and inserting **missing facts** while retaining **supported facts**.

One could employ an encoder-decoder model to perform such a task. Encoder-decoder models implicitly represent the actions for transforming the draft text into a revised text. In contrast, FactEditor explicitly represents the actions for text editing, including `Keep`, `Drop`, and `Gen`, which imply retention, deletion, and generation of word respectively. The model utilizes a buffer for storing the draft text, a stream to store the revised text, and a memory for storing the facts. It also employs a neural network to control the entire editing process. FactEditor also has a lower time complexity than the encoder-decoder model, and thus it can edit a text more efficiently.

Experimental results show that FactEditor outperforms the baseline encoder-decoder approach in terms of fidelity and fluency and shows that FactEditor can perform text editing faster than the encoder-decoder model.

## 4.1 Data

In this section, we describe our method of data creation for fact-based text editing. The method automatically constructs a dataset from an existing table-to-text dataset.

### 4.1.1 Data Sources

There are two benchmark datasets for table-to-text language processing, WebNLG [33][†] and RotoWire[122][‡]. We create two constructed datasets on the basis of these two, referred to as WebEdit and RotoEdit respectively. In these datasets, each instance consists of a table (structured data) and an associated text (unstructured data) describing almost the same content.[§].

For each instance, we consider the table as triples of facts and the associated text as a revised text, and we automatically create a draft text. The set of triples

[†]The data is available at `https://github.com/ThiagoCF05/webnlg`. We utilize version 1.5.

[‡]We utilize the RotoWire-modified data provided by Iso et al. [51] available at `https://github.com/aistairc/rotowire-modified`. The authors also provide an information extractor for processing the data.

[§]In RotoWire, we discard redundant box-scores and unrelated sentences using the information extractor and heuristic rules.

is represented as $\mathcal{T} = \{t\}$. Each triple $t$ consists of subject, predicate, and object, denoted as $t = (subj, pred, obj)$. For simplicity, we refer to the nouns or noun phrases of subject and object simply as entities. The revised text is a sequence of words denoted as $\mathbf{y}$. The draft text is a sequence of words denoted as $\mathbf{x}$.

Given the set of triples $\mathcal{T}$ and the revised text $\mathbf{y}$, we aim to create a draft text $\mathbf{x}$, such that $\mathbf{x}$ is not following $\mathcal{T}$, in contrast to $\mathbf{y}$, and therefore text editing from $\mathbf{x}$ to $\mathbf{y}$ is needed.

### 4.1.2 Data Creation Procedure

Our method first creates templates for all the sets of triples and revised texts and then constructs a draft text for each set of triples and revised text based on their related templates.

#### Creation of templates

For each instance, our method first delexicalizes the entity words in the set of triples $\mathcal{T}$ and the revised text $\mathbf{y}$ to obtain a set of triple templates $\mathcal{T}'$ and a revised template $\mathbf{y}'$. For example, given $\mathcal{T}$ ={(Baymax, voice, Scott_Adsit)} and $\mathbf{y}$ ="Scott_Adsit does the voice for Baymax", it produces the set of triple templates $\mathcal{T}'$ ={(AGENT-1, voice, PATIENT-1)} and the revised template $\mathbf{y}'$ ="AGENT-1 does the voice for PATIENT-1". Our method then collects all the sets of triple templates $\mathcal{T}'$ and revised templates $\mathbf{y}'$ and retains them in a key-value store with $\mathbf{y}'$ being a key and $\mathcal{T}'$ being a value.

#### Creation of draft text

Next, our method constructs a draft text $\mathbf{x}$ using a set of triple templates $\mathcal{T}'$ and a revised template $\mathbf{y}'$. For simplicity, it only considers the use of either *insertion* or *deletion* in text editing, and one can easily extend it to a more complex setting. Note that the process of data creation is the reverse of that of text editing.

Given a pair of $\mathcal{T}'$ and $\mathbf{y}'$, our method retrieves another pair denoted as $\hat{\mathcal{T}}'$ and $\hat{\mathbf{x}}'$, such that $\mathbf{y}'$ and $\hat{\mathbf{x}}'$ have the longest common subsequences. We refer to $\hat{\mathbf{x}}'$ as a reference template. There are two possibilities; $\hat{\mathcal{T}}'$ is a subset or a superset of $\mathcal{T}'$. (We ignore the case in which they are identical.) Our method then manages

| | |
|---|---|
| $\mathbf{y}'$ | AGENT-1 performed as PATIENT-3 on BRIDGE-1 mission that was operated by PATIENT-2 . |
| $\hat{\mathbf{x}}'$ | AGENT-1 served as PATIENT-3 was a crew member of the BRIDGE-1 mission . |
| $\mathbf{x}'$ | AGENT-1 performed as PATIENT-3 on BRIDGE-1 mission . |

(a) Example for insertion. The revised template $\mathbf{y}'$ and the reference template $\hat{\mathbf{x}}'$ share subsequences. The set of triple templates $\mathcal{T}\backslash\hat{\mathcal{T}}$ is {(BRIDGE-1, operator, PATIENT-2)}. Our method removes "that was operated by PATIENT-2" from the revised template $\mathbf{y}'$ to create the draft template $\mathbf{x}'$.

| | |
|---|---|
| $\mathbf{y}'$ | AGENT-1 was created by BRIDGE-1 and PATIENT-2 . |
| $\hat{\mathbf{x}}'$ | The character of AGENT-1 , whose full name is PATIENT-1 , was created by BRIDGE-1 and PATIENT-2 . |
| $\mathbf{x}'$ | AGENT-1 , whose full name is PATIENT-1 , was created by BRIDGE-1 and PATIENT-2 . |

(b) Example for deletion. The revised template $\mathbf{y}'$ and the reference template $\hat{\mathbf{x}}'$ share subsequences. The set of triple templates $\hat{\mathcal{T}}\backslash\mathcal{T}$ is {(AGENT-1, fullName, PATIENT-1)}. Our method copies "whose full name is PATIENT-1" from the reference template $\mathbf{x}'$ to create the draft template $\mathbf{x}'$.

Table 4.2: Examples for insertion and deletion, where **words in green** are matched, **words in gray** are not matched, **words in blue are copied**, and **words in orange** are removed. Best viewed in color.

to change $\mathbf{y}'$ to a draft template denoted as $\mathbf{x}'$ on the basis of the relation between $\mathcal{T}'$ and $\hat{\mathcal{T}}'$. If $\hat{\mathcal{T}}' \subsetneq \mathcal{T}'$, then the draft template $\mathbf{x}'$ created is for insertion, and if $\hat{\mathcal{T}}' \supsetneq \mathcal{T}'$, then the draft template $\mathbf{x}'$ created is for deletion.

For *insertion*, the revised template $\mathbf{y}'$ and the reference template $\hat{\mathbf{x}}'$ share subsequences, and the set of triples $\mathcal{T}\backslash\hat{\mathcal{T}}$ appear in $\mathbf{y}'$ but not in $\hat{\mathbf{x}}'$. Our method keeps the shared subsequences in $\mathbf{y}'$, removes the subsequences in $\mathbf{y}'$ on $\mathcal{T}\backslash\hat{\mathcal{T}}$, and copies the rest of words in $\mathbf{y}'$, to create the draft template $\mathbf{x}'$. Table 4.2a gives an example. The shared subsequences "AGENT-1 performed as PATIENT-3 on BRIDGE-1 mission" are retained. The set of triple templates $\mathcal{T}\backslash\hat{\mathcal{T}}$ is {(BRIDGE-1, operator, PATIENT-2)}. The subsequence "that was operated by PATIENT-2" is removed. Note that the subsequence "served" is not copied because it is not shared by $\mathbf{y}'$ and $\hat{\mathbf{x}}'$.

For *deletion*, the revised template $\mathbf{y}'$ and the reference template $\hat{\mathbf{x}}'$ share subsequences. The set of triples $\hat{\mathcal{T}}\backslash\mathcal{T}$ appear in $\hat{\mathbf{x}}'$ but not in $\mathbf{y}'$. Our method retains the shared subsequences in $\mathbf{y}'$, copies the subsequences in $\hat{\mathbf{x}}'$ about $\hat{\mathcal{T}}\backslash\mathcal{T}$, and copies the rest of words in $\mathbf{y}'$, to create the draft template $\mathbf{x}'$. Table 4.2b

| | WebEdit | | | RotoEdit | | |
|---|---|---|---|---|---|---|
| | Train | Valid | Test | Train | Valid | Test |
| $\#\mathcal{D}$ | 181k | 23k | 29k | 27k | 5.3k | 4.9k |
| $\#\mathcal{W}_{\mathtt{d}}$ | 4.1M | 495k | 624k | 4.7M | 904k | 839k |
| $\#\mathcal{W}_{\mathtt{r}}$ | 4.2M | 525k | 649k | 5.6M | 1.1M | 1.0M |
| $\#\mathcal{S}$ | 403k | 49k | 62k | 209k | 40k | 36k |

Table 4.3: Statistics of WebEdit and RotoEdit, where $\#\mathcal{D}$ is the number of instances, $\#\mathcal{W}_{\mathtt{d}}$ and $\#\mathcal{W}_{\mathtt{s}}$ are the total numbers of words in the draft texts and the revised texts, respectively, and $\#\mathcal{S}$ is total the number of sentences.

gives an example. The subsequences "AGENT-1 was created by BRIDGE-1 and PATIENT-2" are retained. The set of triple templates $\hat{\mathcal{T}} \backslash \mathcal{T}$ is {(AGENT-1, fullName, PATIENT-1)}. The subsequence "whose full name is PATIENT-1" is copied. Note that the subsequence "the character of" is not copied because it is not shared by $\mathbf{y}'$ and $\hat{\mathbf{x}}'$.

After obtaining the draft template, $\mathbf{x}'$, our method lexicalizes it to obtain a draft text $\mathbf{x}$, where the lexicons (entity words) are collected from the corresponding revised text $\mathbf{y}$.

We obtain two datasets with our method, referred to as WebEdit and RotoEdit. Table 4.3 presents the statistics of the datasets.

In the WebEdit data, sometimes entities only appear in the *subj*'s of triples. In such cases, we also make them appear in the *obj*'s. To do so, we introduce an additional triple (`ROOT`, `IsOf`, *subj*) for each *subj*, where `ROOT` is a dummy entity.

## 4.2 Model

In this section, we describe our proposed model for fact-based text editing, referred to as FactEditor.

### 4.2.1 Model Architecture

FactEditor transforms a draft text into a revised text based on given triples. The model consists of three components, a buffer for storing the draft text and its representations, a stream for storing the revised text and its representations, and a memory for storing the triples and their representations, as shown in Figure 4.1.

FactEditor scans the text in the buffer, copies the parts of text from the buffer into the stream if they are described in the triples in the memory, deletes the parts of the text if they are not mentioned in the triples, and inserts new sections of text into the stream which are only present in the triples.

The architecture of FactEditor is inspired by those in sentence parsing Dyer et al. [28], Watanabe and Sumita [116]. The actual processing of FactEditor generates a sequence of words into the stream from the given sequence of words in the buffer and the set of triples in memory. A neural network is employed to control the entire editing process.

### 4.2.2 Neural Network

#### Initialization

FactEditor first initializes the representations of content in the buffer, stream, and memory.

There is a feed-forward network associated with the memory, utilized to create the embeddings of triples. Let $M$ denote the number of triples. The embedding of triple $t_j, j = 1, \cdots, M$ is calculated as

$$\boldsymbol{t}_j = \tanh(\boldsymbol{W}_{\mathtt{t}}[\boldsymbol{e}_j^{\mathtt{subj}}; \boldsymbol{e}_j^{\mathtt{pred}}; \boldsymbol{e}_j^{\mathtt{obj}}] + \boldsymbol{d}_{\mathtt{t}}),$$

where $\boldsymbol{W}_{\mathtt{t}}$ and $\boldsymbol{d}_{\mathtt{t}}$ denote parameters, $\boldsymbol{e}_j^{\mathtt{subj}}, \boldsymbol{e}_j^{\mathtt{pred}}, \boldsymbol{e}_j^{\mathtt{obj}}$ denote the embeddings of subject, predicate, and object of triple $t_j$, and $[\ ;]$ denotes vector concatenation.

There is a bi-directional LSTM associated with the buffer, utilized to create the embeddings of words of draft text. The embeddings are obtained as $\boldsymbol{b} = \text{BiLSTM}(\mathbf{x})$, where $\mathbf{x} = (\mathbf{x}_1, \ldots, \mathbf{x}_N)$ is the list of embeddings of words and $\boldsymbol{b} = (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_N)$ is the list of representations of words, where $N$ denotes the number of words.

An LSTM associated with the stream for representing the hidden states of the stream exists. The first hidden state is initialized as

$$\boldsymbol{s}_1 = \tanh\left(\boldsymbol{W}_{\mathtt{s}}\left[\frac{\sum_{i=1}^{N}\boldsymbol{b}_i}{N};\frac{\sum_{j=1}^{M}\boldsymbol{t}_j}{M}\right] + \boldsymbol{d}_{\mathtt{s}}\right)$$

where $\boldsymbol{W}_{\mathtt{s}}$ and $\boldsymbol{d}_{\mathtt{s}}$ are parameters.

### Action prediction

FactEditor predicts an action at each time $t$ using the LSTM. There are three types of action, namely `Keep`, `Drop`, and `Gen`. First, it composes a context vector $\tilde{\boldsymbol{t}}_t$ of triples at time $t$ using attention

$$\tilde{\boldsymbol{t}}_t = \sum_{j=1}^{M}\alpha_{t,j}\boldsymbol{t}_j$$

where $\alpha_{t,j}$ is a weight calculated as

$$\alpha_{t,j} \propto \exp\left(\boldsymbol{v}_\alpha \cdot \tanh\left(\boldsymbol{W}_\alpha[\boldsymbol{s}_t;\boldsymbol{b}_t;\boldsymbol{t}_j]\right)\right)$$

where $\boldsymbol{v}_\alpha$ and $\boldsymbol{W}_\alpha$ are parameters. Then, it creates the hidden state $\boldsymbol{z}_t$ for action prediction at time $t$

$$\boldsymbol{z}_t = \tanh\left(\boldsymbol{W}_z[\boldsymbol{s}_t;\boldsymbol{b}_t;\tilde{\boldsymbol{t}}_t] + \boldsymbol{d}_z\right)$$

where $\boldsymbol{W}_z$ and $\boldsymbol{d}_z$ are parameters. Next, it calculates the probability of action $a_t$

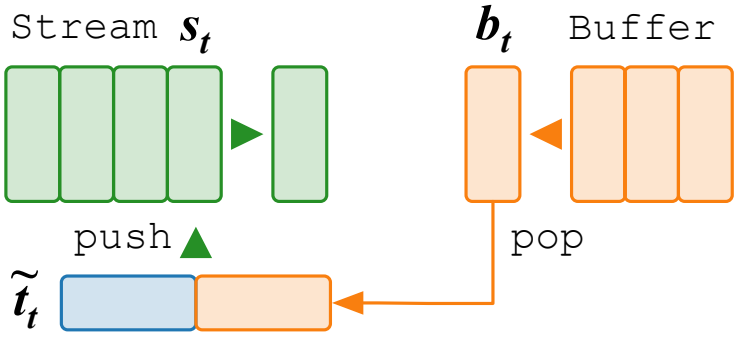
$$p(a_t \mid \boldsymbol{z}_t) = \mathrm{softmax}(\boldsymbol{W}_a\boldsymbol{z}_t)$$

where $\boldsymbol{W}_a$ denotes parameters and chooses the action having the largest probability.

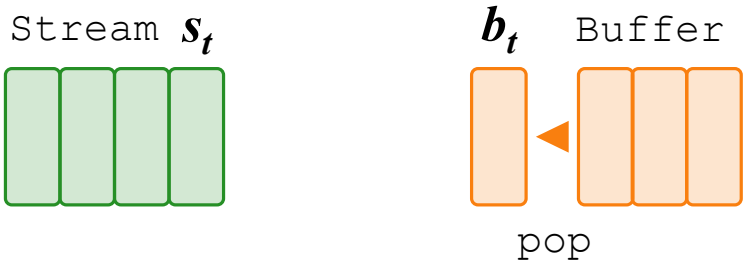
### Action execution

FactEditor takes action based on the prediction result at time $t$.

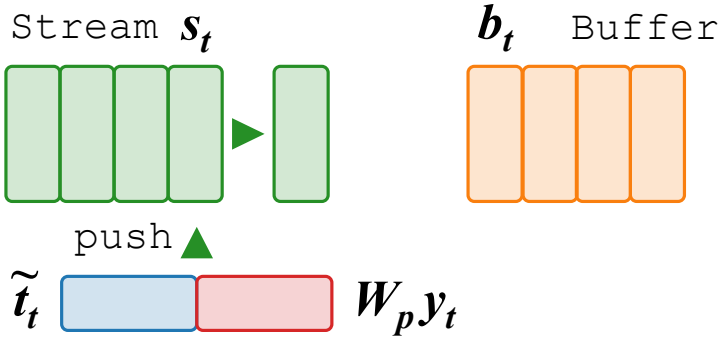
For `Keep` at time $t$, FactEditor pops the top embedding $\boldsymbol{b}_t$ in the buffer, and feeds the combination of the top embedding $\boldsymbol{b}_t$ and the context vector of triples $\tilde{\boldsymbol{t}}_t$ into the stream, as shown in Figure 4.1a. The state of stream is updated with




(a) The `Keep` action, where the top embedding of the buffer $\boldsymbol{b}_t$ is popped and the concatenated vector $[\tilde{\boldsymbol{t}}_t; \boldsymbol{b}_t]$ is pushed into the stream LSTM.




(b) The `Drop` action, where the top embedding of the buffer $\boldsymbol{b}_t$ is popped and the state in the stream is reused at the next time step $t+1$.




(c) The `Gen` action, where the concatenated vector $[\tilde{\boldsymbol{t}}_t; \boldsymbol{W}_p \mathbf{y}_t]$ is pushed into the stream, and the top embedding of the buffer is reused at the next time step $t+1$.

Figure 4.1: Actions of FactEditor.

the LSTM as $\boldsymbol{s}_{t+1} = \text{LSTM}([\tilde{\boldsymbol{t}}_t; \boldsymbol{b}_t], \boldsymbol{s}_t)$. FACTEDITOR also copies the top word in the buffer into the stream.

For `Drop` at time $t$, FACTEDITOR pops the top embedding in the buffer and proceeds to the next state, as shown in Figure 4.1b. The state of stream is updated as $\boldsymbol{s}_{t+1} = \boldsymbol{s}_t$. Note that no word is input into the stream.

For `Gen` at time $t$, FACTEDITOR does not pop the top embedding in the buffer. It feeds the combination of the context vector of triples $\tilde{\boldsymbol{t}}_t$ and the linearly projected embedding of word $w$ into the stream, as shown in Fig. 4.1c. The state of stream is updated with the LSTM as $\boldsymbol{s}_{t+1} = \text{LSTM}([\tilde{\boldsymbol{t}}_t; \boldsymbol{W}_p \mathbf{y}_t], \boldsymbol{s}_t)$, where $\mathbf{y}_t$ is the embedding of the generated word $y_t$, and $\boldsymbol{W}_p$ denotes parameters. In addition, FACTEDITOR copies the generated word $y_t$ into the stream.

FACTEDITOR continues the actions until the buffer becomes empty, and then the text stored in the stream is considered as the revised text.

### Word generation

FACTEDITOR generates a word $y_t$ at time $t$, when the action is GEN,

$$p_{\text{GEN}}(y_t \mid \boldsymbol{z}_t) = \text{softmax}(\boldsymbol{W}_y \boldsymbol{z}_t)$$

where $\boldsymbol{W}_y$ represents the parameters.

To avoid generation of OOV words, FACTEDITOR exploits the copy mechanism. It calculates the probability of copying the object of triple $t_j$

$$\delta_{t,o_j} \propto \exp\left(\boldsymbol{v}_c \cdot \tanh(\boldsymbol{W}_c[\boldsymbol{z}_t; \boldsymbol{t}_j])\right)$$

where $\boldsymbol{v}_c$ and $\boldsymbol{W}_c$ denote parameters, and $o_j$ is the object of triple $t_j$. It also calculates the probability of getting

$$p_{\text{COPY}} = \text{sigmoid}(\boldsymbol{w}_{\text{g}}^{\cdot} \boldsymbol{z}_t + b_{\text{g}})$$

where $\boldsymbol{w}_{\text{g}}$ and $b_{\text{g}}$ are parameters. Finally, it calculates the probability of generating a word $w_t$ through either generation or copying,

$$P(y_t \mid \boldsymbol{z}_t) = (1 - p_{\text{COPY}}) * p_{\text{GEN}}(y_t \mid \boldsymbol{z}_t) + p_{\text{COPY}} * \sum_{j=1:o_j=y_t}^{M} \delta_{t,o_j},$$

where it is assumed that the triples in the memory have the same subject and thus only objects need to be copied.

### 4.2.3 Training

The conditional probability of sequence of actions $\mathbf{a} = (a_1, a_2, \cdots, a_T)$ given the set of triples $\mathcal{T}$, and the sequence of input words $\mathbf{x}$ can be written as

$$P(\mathbf{a} \mid \mathcal{T}, \mathbf{x}) = \prod_{t=1}^{T} P(a_t \mid \boldsymbol{z}_t)$$

where $P(a_t \mid \boldsymbol{z}_t)$ is the conditional probability of action $a_t$ given state $\boldsymbol{z}_t$ at time $t$ and $T$ denotes the number of actions.

The conditional probability of sequence of generated words $\mathbf{y} = (y_1, y_2, \cdots, y_T)$ given the sequence of actions $\boldsymbol{a}$ can be written as

$$P(\mathbf{y} \mid \mathbf{a}) = \prod_{t=1}^{T} P(y_t \mid a_t) \tag{4.1}$$

where $P(y_t \mid a_t)$ is the conditional probability of generated word $y_t$ given action $a_t$ at time $t$, which is calculated as

$$P(y_t \mid a_t) = \begin{cases} P(y_t \mid \boldsymbol{z_t}) & \text{if } a_t = \texttt{Gen} \\ 1 & \text{otherwise} \end{cases} \tag{4.2}$$

Note that not all positions have a generated word. In such a case, $y_t$ is simply a null word.

The training of the model is carried out via supervised learning. The objective of learning is to minimize the negative log-likelihood of $P(\mathbf{a} \mid \mathcal{T}, \mathbf{x})$ and $P(\mathbf{y} \mid \mathbf{a})$

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{t=1}^{T} \{\log P(a_t \mid \boldsymbol{z}_t) + \log P(y_t \mid a_t)\}$$

where $\boldsymbol{\theta}$ denotes the parameters.

A training instance consists of a pair with draft text and revised text, as well as a set of triples, denoted as $\mathbf{x}$, $\mathbf{y}$, and $\mathcal{T}$ respectively. For each instance, our method derives a sequence of actions denoted as $\boldsymbol{a}$, in a similar way as that in [26]. It first finds the longest common sub-sequence between $\mathbf{x}$ and $\mathbf{y}$, and then selects an action of `Keep`, `Drop`, or `Gen` at each position, according to how $\mathbf{y}$ is obtained from $\mathbf{x}$ and $\mathcal{T}$ (cf., Table 4.4). Action `Gen` is preferred to action `Drop` when both are valid.

| | |
|---|---|
| Draft text $\mathbf{x}$ | Bakewell_pudding is Dessert that can be served Warm or cold . |
| Revised text $\mathbf{y}$ | Bakewell_pudding is Dessert that originates from Derbyshire_Dales . |
| Action sequence $\boldsymbol{a}$ | `Keep Keep Keep Keep Gen(originates) Gen(from) Gen(Derbyshire_Dales) Drop Drop Drop Drop Keep` |

Table 4.4: An example of action sequence derived from a draft text and revised text.

### 4.2.4 Time Complexity

The time complexity of inference in FactEditor is $\mathcal{O}(NM)$, where $N$ is the number of words in the buffer, and $M$ is the number of triples. Scanning data in the buffer is of complexity $\mathcal{O}(N)$. The generation of action needs the execution of attention, which is of complexity $\mathcal{O}(M)$. Usually, $N$ is much larger than $M$.

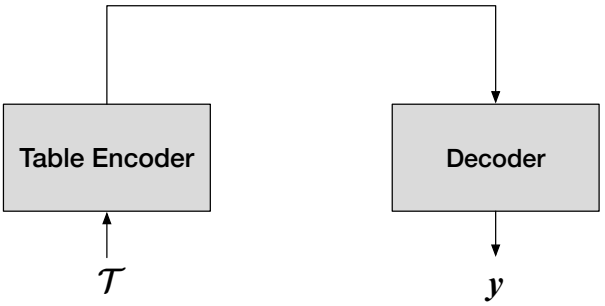
## 4.3 Experiments

We conduct experiments to compare FactEditor and the baselines using the two datasets WebEdit and RotoEdit.

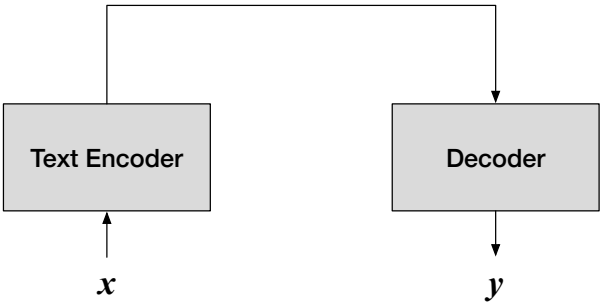
### 4.3.1 Settings

The primary baseline is the encoder-decoder model EncDecEditor, as explained above. We further consider three baselines, No-Editing, Table-to-Text, and Text-to-Text. In No-Editing, the draft text is directly used. In Table-to-Text, a revised text is generated from the triples using encoder-decoder. In Text-to-Text, a revised text is created from the draft text using the encoder-decoder model. Figure 4.2 illustrates the baselines.

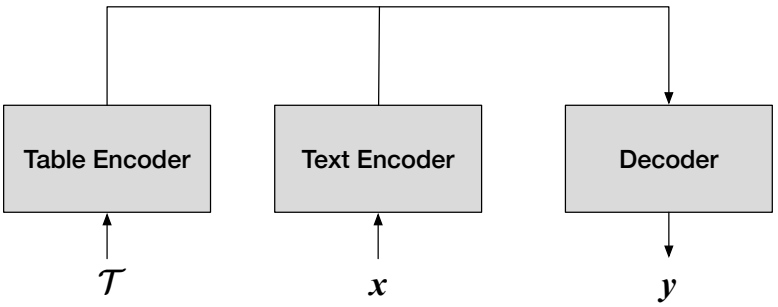
We evaluate the results of texts revised by the models from the viewpoint of fluency and fidelity. We utilize ExactMatch (EM), Bleu [88] and Sari [125]




(a) Table-to-Text




(b) Text-to-Text




(c) EncDecEditor

Figure 4.2: Model architectures of the baselines. All models employ attention and copy mechanisms.

scores[¶] as evaluation metrics for fluency. Further, we utilize precision, recall, and F1 score as evaluation metrics for fidelity. For WebEdit, we extract the entities from the generated text and the reference text and then calculate the precision, recall, and F1 scores. For RotoEdit, we use the information extraction tool provided by Wiseman et al. [122] for calculation of the scores.

For the embeddings of subject and object for both datasets and the embedding of the predicate for RotoEdit, we use the embedding lookup table. For the embedding of the predicate for WebEdit, we first tokenize the predicate, lookup the embeddings of lower-cased words from the table, and use averaged embedding to deal with the OOV problem [79].

We tune the hyperparameters based on the Bleu score on a development set. For WebEdit, we set the sizes of embeddings, buffers, and triples to 300 and the stream's size to 600. For RotoEdit, we set the size of embeddings to 100 and the sizes of buffers, triples, and stream to 200. The initial learning rate is 2e-3, and AMSGrad is used for automatically adjusting the learning rate [97]. Our implementation makes use of AllenNLP [34].

### 4.3.2 Baseline

We consider a baseline method using the encoder-decoder architecture, which takes the set of triples and the draft text as input and generates a revised text. We refer to the method as EncDecEditor. The encoder of EncDecEditor is the same as that of FactEditor. The decoder is the standard attention and copy model, which creates and utilizes a context vector and predicts the next word.

The time complexity of inference in EncDecEditor is $\mathcal{O}(N^2+NM)$ (cf.,Britz et al. [13]). Note that in fact-based text editing, usually $N$ is very large, implying that EncDecEditor is less efficient than FactEditor.

[¶]We use a modified version of SARI where $\beta$ equals 1.0, available at `https://github.com/tensorflow/tensor2tensor/blob/master/tensor2tensor/utils/sari_hook.py`

### 4.3.3 Results

#### Quantitative evaluation

We present the performances of our proposed model FACTEDITOR and the baselines on fact-based text editing in Table 4.5. Several conclusions can be drawn from the results.

First, our proposed model, FACTEDITOR, achieves significantly better performances than the main baseline, ENCDECEDITOR, in terms of almost all measures. In particular, FACTEDITOR obtains significant gains in DELETE scores on both WEBEDIT and ROTOEDIT.

Second, the fact-based text editing models (FACTEDITOR and ENCDECEDITOR) significantly improve upon the other models in terms of fluency scores and achieve similar performances in terms of fidelity scores.

Third, Table-to-Text has higher fidelity scores but lower fluency scores than No-Editing. Text-to-Text has almost the same fluency scores but lower fidelity scores on ROTOEDIT.

We also examined the relationship between the editing difficulty and performance. Specifically, we partitioned the data into buckets based on the edit distance between the input and output and then calculate the SARI score for each.

As shown in Table 4.6, the SARI scores are generally stable, indicating that the overall editing performance does not highly depend on the editing difficulty except for straightforward cases. The Keep score, which indicates whether the draft text is correctly retained, is stable even when the edit distance increases in both datasets. Also, the DELETE score, which indicates the capability to delete inconsistent text with the facts, is stable or increases as the edit distance increases. This indicates that, especially in the WEBEDIT dataset, as the edit distance increases, the number of factually inconsistent parts increases, and the deletion of such parts is more straightforward.

However, the ADD score, which indicates whether new facts are inserted correctly, decreased significantly as the editing distance increased, suggesting that correctly inserting facts remains an essential issue in fact-based text editing.

| Model | Fluency | | | | | | Fidelity | | |
|---|---|---|---|---|---|---|---|---|---|
| | Bleu | Sari | Keep | Add | Delete | EM | P% | R% | F1% |
| **Baselines** | | | | | | | | | |
| No-Editing | 66.67 | 31.51 | 78.62 | 3.91 | 12.02. | 0. | 84.49 | 76.34 | 80.21 |
| Table-to-Text | 33.75 | 43.83 | 51.44 | 27.86 | 52.19 | 5.78 | **98.23** | 83.72 | 90.40 |
| Text-to-Text | 63.61 | 58.73 | 82.62 | 25.77 | 67.80 | 6.22 | 81.93 | 77.16 | 79.48 |
| **Fact-based text editing** | | | | | | | | | |
| EncDecEditor | 71.03 | 69.59 | 89.49 | 43.82 | 75.48 | 20.96 | 98.06 | 87.56 | 92.51 |
| FactEditor | **75.68** | **72.20** | **91.84** | **47.69** | **77.07** | **24.80** | 96.88 | **89.74** | **93.17** |

(a) WebEdit

| Model | Fluency | | | | | | Fidelity | | |
|---|---|---|---|---|---|---|---|---|---|
| | Bleu | Sari | Keep | Add | Delete | EM | P% | R% | F1% |
| **Baselines** | | | | | | | | | |
| No-Editing | 74.95 | 39.59 | 95.72 | 0.05 | 23.01 | 0. | 92.92 | 65.02 | 76.51 |
| Table-to-Text | 24.87 | 23.30 | 39.12 | 14.78 | 16.00 | 0. | 48.01 | 24.28 | 32.33 |
| Text-to-Text | 78.07 | 60.25 | 97.29 | 13.04 | 70.43 | 0.02 | 63.62 | 41.08 | 49.92 |
| **Fact-based text editing** | | | | | | | | | |
| EncDecEditor | 83.36 | 71.46 | 97.69 | **44.02** | 72.69 | 2.49 | 78.80 | 52.21 | 62.81 |
| FactEditor | **84.43** | **74.72** | **98.41** | 41.50 | **84.24** | **2.65** | **78.84** | **52.30** | **63.39** |

(b) RotoEdit

Table 4.5: Performance of FactEditor and baselines on two datasets in terms of fluency and fidelity. EM represents the exact match.

| Edit Distance | [1, 5) | [5, 10) | [10, 15) | [15, ∞) |
|---|---|---|---|---|
| SARI | 77.26 | 76.10 | 75.80 | 76.47 |
| KEEP | 95.51 | 93.74 | 92.76 | 92.48 |
| ADD | 56.36 | 52.94 | 50.66 | 50.49 |
| DELETE | 79.92 | 81.62 | 83.99 | 86.45 |
| Ratio | 18.07 | 43.90 | 29.61 | 8.42 |

(a) WebEdit

| Edit Distance | [1, 25) | [25, 50) | [50, 75) | [75, ∞) |
|---|---|---|---|---|
| SARI | 78.34 | 73.79 | 73.31 | 73.62 |
| KEEP | 98.78 | 98.33 | 98.16 | 98.17 |
| ADD | 50.85 | 39.01 | 37.47 | 38.12 |
| DELETE | 85.38 | 84.03 | 84.31 | 84.57 |
| Ratio | 35.29 | 37.29 | 19.39 | 8.03 |

(b) RotoEdit

Table 4.6: Relationship between editing difficulty and editing performance.

**Qualitative evaluation**

We also manually evaluate 50 randomly sampled revised texts for WEBEDIT. We check whether the revised texts given by FACTEDITOR and ENCDECEDITOR include all the facts. We categorize the factual errors made by the two models. Table 4.7 shows the results. We note that FACTEDITOR covers more facts than ENCDECEDITOR and has less factual errors than ENCDECEDITOR.

FACTEDITOR has a larger number of correct editing cases (CQT) than ENCDECEDITOR for fact-based text editing. In contrast, ENCDECEDITOR often includes a larger number of unnecessary rephrasings (UPARA) than FACTEDITOR.

We considered four types of factual errors: fact repetitions (RPT), fact omis-

| | Covered facts | | Factual errors | | | |
|---|---|---|---|---|---|---|
| | CQT | UPara | Rpt | Ms | USup | DRel |
| EncDecEditor | 14 | 7 | 16 | 21 | 3 | 12 |
| FactEditor | **24** | **4** | **9** | **19** | **1** | **3** |

Table 4.7: Evaluation results on 50 randomly sampled revised texts in WebEdit in terms of numbers of correct editing (CQT), unnecessary paraphrasing (UPara), repetition (Rpt), missing facts (Ms), unsupported facts (USup) and differing relations (DRel)

sions (Ms), unsupported facts (USup), and relation differences (DRel). Both FactEditor and EncDecEditor often fail to insert missing facts (Ms), but rarely insert unsupported facts (USup). EncDecEditor often generates the same facts multiple times (RPT) or facts in different relations (DRel). In contrast, FactEditor seldom makes such errors.

Table 4.8 shows an example of results provided by EncDecEditor and FactEditor. The revised texts of both EncDecEditor and FactEditor appear to be fluent, but that of FactEditor has higher fidelity than that of EncDecEditor. EncDecEditor cannot effectively eliminate the use of an unsupported fact (in orange) appearing in the draft text. In contrast, FactEditor can deal with the problem effectively. In addition, EncDecEditor conducts an unnecessary substitution in the draft text (underlined). FactEditor tends to avoid such unnecessary editing.

### Comparison by oracle action creation method

In this study, we assumed the approach of prioritizing Gen actions when there was ambiguity in the creation of oracle action sequences, and we analyzed the impact of this approach.

Table 4.9 shows the performance of FactEditor for different oracle action creation methods on the development set of WebEdit dataset. The experimental results show that editing performance was similar for both methods, but the overall score is slightly higher when Gen actions are prioritized than when Drop

| | |
|---|---|
| Set of triples | {(**Ardmore_Airport**, **runwayLength**, **1411.0**), (**Ardmore_Airport**, **3rd_runway_SurfaceType**, **Poaceae**), (**Ardmore_Airport**, **operatingOrganisation**, **Civil_Aviation_Authority_of_New_Zealand**), (**Ardmore_Airport**, **elevationAboveTheSeaLevel**, **34.0**), (**Ardmore_Airport**, **runwayName**, **03R/21L**)} |
| Draft text | **Ardmore_Airport** , **ICAO Location Identifier UTAA** . **Ardmore_Airport** 3rd runway is made of **Poaceae** and **Ardmore_Airport** . **03R/21L** is **1411.0** m long and **Ardmore_Airport** is **34.0** above sea level . |
| Revised text | **Ardmore_Airport** is operated by **Civil_Aviation_Authority_of_New_Zealand** . **Ardmore_Airport** 3rd runway is made of **Poaceae** and **Ardmore_Airport** name is **03R/21L** . **03R/21L** is **1411.0** m long and **Ardmore_Airport** is **34.0** above sea level . |
| EncDecEditor | **Ardmore_Airport** , **ICAO Location Identifier UTAA** , is operated by **Civil_Aviation_Authority_of_New_Zealand** . **Ardmore_Airport** 3rd runway is made of **Poaceae** and **Ardmore_Airport** . **03R/21L** is **1411.0** m long and **Ardmore_Airport** is **34.0** m long . |
| FactEditor | **Ardmore_Airport** is operated by **Civil_Aviation_Authority_of_New_Zealand** . **Ardmore_Airport** 3rd runway is made of **Poaceae** and **Ardmore_Airport** . **03R/21L** is **1411.0** m long and **Ardmore_Airport** is **34.0** above sea level . |

Table 4.8: Example of generated revised texts given by EncDecEditor and FactEditor on WebEdit. **Entities in green** appear in both the set of triples and the draft text. **Entities in orange** only appear in the draft text. **Entities in blue** appear in the revised text but not in the draft text.

| | Bleu | Sari | Keep | Add | Delete | EM |
|---|---|---|---|---|---|---|
| Gen first | 77.29 | 76.25 | 93.66 | 52.68 | 82.42 | 25.36 |
| Drop first | 76.22 | 75.12 | 93.45 | 49.90 | 82.02 | 22.35 |

Table 4.9: FactEditor performance with different oracle action creation methods on validation dataset of WebEdit.

| | WebEdit | RotoEdit |
|---|---|---|
| Table-to-Text | **4,083** | **1,834** |
| Text-to-Text | 2,751 | 581 |
| EncDecEditor | 2,487 | 505 |
| FactEditor | <u>3,295</u> | <u>1,412</u> |

Table 4.10: Runtime analysis (# of words/second). Table-to-Text always shows the fastest performance (**Bold-faced**). FactEditor shows the second fastest runtime performance (<u>Underlined</u>).

actions are prioritized.

### Runtime analysis

We conducted a runtime analysis on FactEditor and the baselines in terms of number of processed words per second, on both WebEdit and RotoEdit. Table 4.10 presents the results for all methods when the batch size was 128. Table-to-Text is the fastest, followed by FactEditor. FactEditor is always faster than EncDecEditor, apparently because it has a lower time complexity, as explained in Section 4.2.4. The texts in WebEdit are relatively short, and thus FactEditor and EncDecEditor have similar runtime speeds. In contrast, the texts in RotoEdit are relatively long, and thus FactEditor executes approximately two times faster than EncDecEditor.

## 4.4 Conclusion

In this chapter, we defined a new task referred to as *fact-based text editing* and made two contributions to research on the problem. First, we proposed a data construction method for fact-based text editing and created two datasets. Second, we proposed a model for fact-based text editing, named FactEditor, which performs the task by generating a sequence of actions. Experimental results show that the proposed model FactEditor performs better and faster than the baselines, including an encoder-decoder model.

# 5 Conclusion

In this dissertation, methods for developing natural language generation systems were presented. We discussed our methods to tackle *data-to-document generations* and *fact-based text editing*. In this section, we conclude this thesis and discuss the challenges and suggest future work.

Concretely, in Chapter 3, we developed models that can generate a meaningful and faithful document from massive data. We found that the tracking module successfully memorizes the saliency transition during the generation of a game summary. As a result, our model outperformed the existing models in all evaluation measures on the RotoWire-Modified dataset. We also explored the effects of incorporating writer information into data-to-text models. With writer information, our model successfully generated the highest quality summaries, achieving a BLEU score of 20.62 points.

In Chapter 4, we introduced a new text editing task, referred to as *Fact-based Text Editing*, in which the goal is to revise a given document to describe facts in a knowledge base better. First, we proposed a data construction method for fact-based text editing and created two datasets. Second, we proposed a model for fact-based text editing, named FactEditor, which performs the task by generating a sequence of actions. Experimental results show that the proposed model FactEditor performs better and faster than the baselines, including an encoder-decoder model.

## 5.1 Future work

We conclude this dissertation by discussing the remaining challenges in natural language generation systems.

## Integrating with Transformer architectures

Although we mainly focus on the recurrent neural network-based NLG models in this thesis, a new encoder-decoder architecture, referred to as *Transformer*, has been proposed, which only uses the attention mechanism and often achieves better performance than recurrent neural network based-sequence generation model [113].

However, the model introduced in this thesis includes memory-like architecture in which the saliency of each entity is updated during the generation process, but it is not apparent that such a mechanism is incorporated into the Transformer model. Furthermore, in recent years, pre-trained language models have been developed on top of a Transformer such as bidirectional encoder representations from Transformer (BERT)[23], and the generative pre-trained transformer (GPT)[94, 95]. Thus, it is crucial to develop a transformer model that can generate text while retaining long-term history with a memory, such as the architecture proposed in this thesis.

## Interactive Text Editing

In this thesis, we introduced Fact-based Text Editing, which enables a machine agent to add and delete information from external knowledge. Although our work can be viewed as a first step toward editing text based on facts, a more versatile editing model should be developed in the future. This thesis assumes the presence of triples as sources of information to achieve text editing, but we envision the realization of text editing from unstructured text instruction for the future direction. For example, given a draft text and textual instructions to the model, the model should retrieve facts from a large corpus of documents and edit text based on the retrieved facts.

# Publication List

## Journal paper

1. Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoaki Okazaki and Hiroya Takamura, Learning to Select, Track, and Generate for Data-to-Text, Journal of Natural Language Processing, The Association for Natural Language Processing Vol.27 No.3, 2020

2. Hayate Iso, Shoko Wakamiya, Eiji Aramaki, Conditional Density Estimation of Tweet Location: A Feature-Dependent Approach, Stud Health Technol Inform. 2017;245:408-411. PMID: 29295126.

## International conference

1. Hayate Iso, Chao Qiao, Hang Li, Fact-based Text Editing, Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL) (long), 2020.

2. Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoaki Okazaki, Hiroya Takamura Learning to Select, Track, and Generate for Data-to-Text, Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL) (long), 2019.

3. Hayate Iso, Kaoru Ito, Hiroyuki Nagai, Taro Okahisa and Eiji Aramaki, Parsing Japanese Tweets into Universal Dependencies, Proceedings of Universal Dependencies Workshop (UDW), 2018.

4. Paolo Casani, Hayate Iso, Shoko Wakamiya, Eiji Aramaki, Wisdom in Adversity: A Twitter Study of the Japanese Tsunami, Advances in Social Networks Analysis and Mining (ASONAM), 2018.

5. Hayate Iso, Camille Ruiz, Taichi Murayama, Katsuya Taguchi, Ryo Takeuchi, Hideya Yamamoto, Shoko Wakamiya, Eiji Aramaki, NTCIR13 MedWeb Task: Multi-label Classification of Tweets using an Ensemble of Neural

Networks, Proceedings of NTCIR Conference on Evaluation of Information Access Technologies, (NTCIR), 2017.

6. Hayate Iso, Shoko Wakamiya, Eiji Aramaki, Forecasting word model: Twitter-based influenza surveillance and prediction, Proceedings the 26th International Conference on Computational Linguistics: Technical Papers (COLING) (long), 2016.

# References


[1] Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. Globally normalized transition-based neural networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2442–2452, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1231. URL `https://www.aclweb.org/anthology/P16-1231`.

[2] Gabor Angeli, Percy Liang, and Dan Klein. A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512, Cambridge, MA, October 2010. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D10-1049`.

[3] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *Proceedings of the Third International Conference on Learning Representations*, 2015. URL `https://arxiv.org/pdf/1409.0473.pdf`.

[4] Miguel Ballesteros, Chris Dyer, Yoav Goldberg, and Noah A. Smith. Greedy transition-based dependency parsing with stack LSTMs. *Computational Linguistics*, 43(2):311–347, June 2017. doi: 10.1162/COLI_a_00285. URL `https://www.aclweb.org/anthology/J17-2002`.

[5] Regina Barzilay and Mirella Lapata. Collective content selection for concept-to-text generation. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 331–338, 2005. URL `http://aclweb.org/anthology/H05-1042`.

[6] Regina Barzilay and Lillian Lee. Catching the drift: Probabilistic content models, with applications to generation and summarization. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 113–120, Boston, Massachusetts, USA, May 2 - May 7 2004. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/N04-1015`.

[7] Regina Barzilay, Noemie Elhadad, and Kathleen R. McKeown. Inferring strategies for sentence ordering in multidocument news summarization. *J. Artif. Int. Res.*, 17(1):35–55, August 2002. ISSN 1076-9757.

[8] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

[9] Danushka Bollegala, Naoaki Okazaki, and Mitsuru Ishizuka. A bottom-up approach to sentence ordering for multi-document summarization. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 385–392, Sydney, Australia, July 2006. Association for Computational Linguistics. doi: 10.3115/1220175.1220224. URL `https://www.aclweb.org/anthology/P06-1049`.

[10] Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. Simulating Action Dynamics with Neural Process Networks. In *Proceedings of the Sixth International Conference on Learning Representations*, 2018. URL `https://openreview.net/pdf?id=rJYFzMZC-`.

[11] Thorsten Brants, Ashok C. Popat, Peng Xu, Franz J. Och, and Jeffrey Dean. Large language models in machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 858–867, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D07-1090`.

[12] Eric Brill and Robert C. Moore. An improved error model for noisy channel spelling correction. In *Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics*, pages 286–293, Hong Kong, October 2000. Association for Computational Linguistics. doi: 10.3115/1075218.1075255. URL https://www.aclweb.org/anthology/P00-1037.

[13] Denny Britz, Melody Guan, and Minh-Thang Luong. Efficient attention using a fixed-size memory representation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 392–400, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1040. URL https://www.aclweb.org/anthology/D17-1040.

[14] Peter F Brown, Stephen A Della Pietra, Vincent J Della Pietra, and Robert L Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational linguistics*, 19(2):263–311, 1993.

[15] C. Buck, Kenneth Heafield, and Bas van Ooyen. N-gram counts and language models from the common crawl. In *LREC*, 2014.

[16] David L Chen and Raymond J Mooney. Learning to sportscast: a test of grounded language acquisition. In *Proceedings of the 25th international conference on Machine learning*, pages 128–135, 2008. URL https://icml.cc/Conferences/2008/papers/304.pdf.

[17] F. ChenStanley and GoodmanJoshua. An empirical study of smoothing techniques for language modeling. *Computer Speech & Language*, 1999.

[18] Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, pages 1724–1734, 2014. URL https://www.aclweb.org/anthology/D14-1179.

[19] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio.

Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[20] Elizabeth Clark, Yangfeng Ji, and Noah A Smith. Neural Text Generation in Stories Using Entity Representations as Context. In *Proceedings of the 16th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2250–2260, 2018. URL `http://aclweb.org/anthology/N18-1204`.

[21] James Cross and Liang Huang. Incremental parsing with minimal features using bi-directional LSTM. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 32–37, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-2006. URL `https://www.aclweb.org/anthology/P16-2006`.

[22] Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2978–2988, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1285. URL `https://www.aclweb.org/anthology/P19-1285`.

[23] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL `https://www.aclweb.org/anthology/N19-1423`.

[24] Bhuwan Dhingra, Manaal Faruqui, Ankur Parikh, Ming-Wei Chang, Dipanjan Das, and William Cohen. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4884–4895, 2019.

[25] William B. Dolan and Chris Brockett. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005. URL https://www.aclweb.org/anthology/I05-5002.

[26] Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3393–3402, Florence, Italy, July 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1331. URL https://www.aclweb.org/anthology/P19-1331.

[27] Pablo Ariel Duboue and Kathleen R. McKeown. Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 Conference on Empirical Methods in Natural Language Processing*, pages 121–128, 2003. URL https://www.aclweb.org/anthology/W03-1016.

[28] Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. Transition-based dependency parsing with stack long short-term memory. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 334–343, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1033. URL https://www.aclweb.org/anthology/P15-1033.

[29] J. Elman. Finding structure in time. *Cogn. Sci.*, 14:179–211, 1990.

[30] Yao Fu, Chuanqi Tan, Bin Bi, Mosha Chen, Yansong Feng, and Alexander M Rush. Latent template induction with gumbel-crfs. In *Advances in Neural Information Processing Systems*, 2020.

[31] Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *Advances in Neural Information Processing Systems*, 2016.

[32] Claire Gardent and Laura Perez-Beltrachini. A statistical, grammar-based approach to microplanning. *Computational Linguistics*, 43(1):1–30, 2017.

[33] Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 179–188, Vancouver, Canada, July 2017. Association for Computational Linguistics. doi: 10.18653/v1/P17-1017. URL `https://www.aclweb.org/anthology/P17-1017`.

[34] Matt Gardner, Joel Grus, Mark Neumann, Oyvind Tafjord, Pradeep Dasigi, Nelson F. Liu, Matthew Peters, Michael Schmitz, and Luke Zettlemoyer. AllenNLP: A deep semantic natural language processing platform. In *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)*, pages 1–6, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-2501. URL `https://www.aclweb.org/anthology/W18-2501`.

[35] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256, 2010. URL `http://proceedings.mlr.press/v9/glorot10a/glorot10a.pdf`.

[36] E. Grave, Armand Joulin, and Nicolas Usunier. Improving neural language models with a continuous cache. *Proceedings of the International Conference on Learning Representations*, abs/1612.04426, 2017.

[37] Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwińska, Sergio Gómez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John Agapiou, et al. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538(7626): 471, 2016. URL `https://www.nature.com/articles/nature20101.pdf`.

[38] Klaus Greff, R. Srivastava, J. Koutník, Bastiaan Steunebrink, and J. Schmidhuber. Lstm: A search space odyssey. *IEEE Transactions on Neural Networks and Learning Systems*, 28:2222–2232, 2017.

[39] Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating Copying Mechanism in Sequence-to-Sequence Learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, volume 1, pages 1631–1640, 2016. URL `http://www.aclweb.org/anthology/P16-1154`.

[40] Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. Pointing the Unknown Words. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 140–149, 2016. URL `http://aclweb.org/anthology/P16-1014`.

[41] Kelvin Guu, Tatsunori B Hashimoto, Yonatan Oren, and Percy Liang. Generating Sentences by Editing Prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450, 2018. URL `https://www.aclweb.org/anthology/Q18-1031`.

[42] Tatsunori B Hashimoto, Kelvin Guu, Yonatan Oren, and Percy S Liang. A retrieve-and-edit framework for predicting structured outputs. In *Advances in Neural Information Processing Systems*, pages 10052–10062, 2018.

[43] Kenneth Heafield, Ivan Pouzyrevsky, Jonathan H. Clark, and Philipp Koehn. Scalable modified Kneser-Ney language model estimation. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 690–696, Sofia, Bulgaria, August 2013. Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/P13-2121`.

[44] Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. Tracking the world state with recurrent entity networks. In *Proceedings of the Fifth International Conference on Learning Representations*, 2017. URL `https://openreview.net/pdf?id=rJTKKKqeg`.

[45] Karl Moritz Hermann and Phil Blunsom. The role of syntax in vector space models of compositional semantics. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 894–904, Sofia, Bulgaria, August 2013. Association for

Computational Linguistics. URL https://www.aclweb.org/anthology/P13-1088.

[46] Luong Hoang, Sam Wiseman, and Alexander Rush. Entity Tracking Improves Cloze-style Reading Comprehension. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1055, 2018. URL http://www.aclweb.org/anthology/D18-1130.

[47] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Computation*, 9:1735–1780, 1997.

[48] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.

[49] Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P. Xing. Toward Controlled Generation of Text. In Doina Precup and Yee Whye Teh, editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1587–1596, International Convention Centre, Sydney, Australia, 06–11 Aug 2017. PMLR. URL http://proceedings.mlr.press/v70/hu17e.html.

[50] Kentaro Inui, Atsushi Fujita, Tetsuro Takahashi, Ryu Iida, and Tomoya Iwakura. Text simplification for reading assistance: A project note. In *Proceedings of the Second International Workshop on Paraphrasing*, pages 9–16, Sapporo, Japan, July 2003. Association for Computational Linguistics. doi: 10.3115/1118984.1118986. URL https://www.aclweb.org/anthology/W03-1602.

[51] Hayate Iso, Yui Uehara, Tatsuya Ishigaki, Hiroshi Noji, Eiji Aramaki, Ichiro Kobayashi, Yusuke Miyao, Naoaki Okazaki, and Hiroya Takamura. Learning to select, track, and generate for data-to-text. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 2102–2113, Florence, Italy, August 2019. URL https://www.aclweb.org/anthology/P19-1202.

[52] Hayate Iso, Chao Qiao, and Hang Li. Fact-based Text Editing. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 171–182, Online, July 2020. URL https://www.aclweb.org/anthology/2020.acl-main.17.

[53] Yangfeng Ji, Chenhao Tan, Sebastian Martschat, Yejin Choi, and Noah A Smith. Dynamic Entity Representations in Neural Language Models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1830–1839, 2017. URL http://aclweb.org/anthology/D17-1195.

[54] Nal Kalchbrenner and Phil Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1176.

[55] Chloé Kiddon, Luke Zettlemoyer, and Yejin Choi. Globally coherent text generation with neural checklist models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 329–339, 2016. URL http://aclweb.org/anthology/D16-1032.

[56] Eliyahu Kiperwasser and Yoav Goldberg. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *Transactions of the Association for Computational Linguistics*, 4:313–327, 2016. doi: 10.1162/tacl_a_00101. URL https://www.aclweb.org/anthology/Q16-1023.

[57] Reinhard Kneser and H. Ney. Improved backing-off for m-gram language modeling. *1995 International Conference on Acoustics, Speech, and Signal Processing*, 1:181–184 vol.1, 1995.

[58] Kevin Knight and Ishwar Chander. Automated Postediting of Documents. In *Proceedings of the AAAI Conference on Artificial Intelligence.*, volume 94, pages 779–784, 1994. URL https://www.aaai.org/Papers/AAAI/1994/AAAI94-119.pdf.

[59] R. Koncel-Kedziorski, Hannaneh Hajishirzi, and Ali Farhadi. Multi-resolution language grounding with weak supervision. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 386–396, Doha, Qatar, October 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1043. URL https://www.aclweb.org/anthology/D14-1043.

[60] Ioannis Konstas and Mirella Lapata. Inducing document plans for concept-to-text generation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1503–1514, Seattle, Washington, USA, October 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1157.

[61] Ioannis Konstas and Mirella Lapata. A global model for concept-to-text generation. *Journal of Artificial Intelligence Research*, 48:305–346, 2013.

[62] Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, 2019.

[63] Wojciech Kryscinski, Bryan McCann, Caiming Xiong, and Richard Socher. Evaluating the factual consistency of abstractive text summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9332–9346, 2020.

[64] Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. Neural architectures for named entity recognition. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 260–270, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1030. URL https://www.aclweb.org/anthology/N16-1030.

[65] Mirella Lapata. Probabilistic text structuring: Experiments with sentence

ordering. In *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics*, pages 545–552, 2003.

[66] Rémi Lebret, David Grangier, and Michael Auli. Neural Text Generation from Structured Data with Application to the Biography Domain. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1203–1213, 2016. URL http://www.aclweb.org/anthology/D16-1128.

[67] Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. Paraphrase Generation with Deep Reinforcement Learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1421. URL https://www.aclweb.org/anthology/D18-1421.

[68] Percy Liang, Michael I Jordan, and Dan Klein. Learning semantic correspondences with less supervision. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 91–99, 2009. URL http://aclweb.org/anthology/P09-1011.

[69] Tianyu Liu, Kexiang Wang, Lei Sha, Baobao Chang, and Zhifang Sui. Table-to-text Generation by Structure-aware Seq2seq Learning. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. URL https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/viewFile/16599/16019.

[70] Hans Peter Luhn. The automatic creation of literature abstracts. *IBM Journal of research and development*, 2(2):159–165, 1958.

[71] Thang Luong, Hieu Pham, and Christopher D Manning. Effective Approaches to Attention-based Neural Machine Translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015. URL http://www.aclweb.org/anthology/D15-1166.

[72] Thang Luong, Ilya Sutskever, Quoc Le, Oriol Vinyals, and Wojciech Zaremba. Addressing the rare word problem in neural machine translation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 11–19, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1002. URL `https://www.aclweb.org/anthology/P15-1002`.

[73] Daniel Marcu. From local to global coherence: a bottom-up approach to text planning. In *Proceedings of the fourteenth national conference on artificial intelligence and ninth conference on Innovative applications of artificial intelligence*, pages 629–635, 1997.

[74] Susan W McRoy, Songsak Channarukul, and Syed S Ali. An augmented template-based approach to text realization. *Natural Language Engineering*, 9(4):381, 2003.

[75] Hongyuan Mei, Mohit Bansal, and Matthew R Walter. What to talk about and how? Selective Generation using LSTMs with Coarse-to-Fine Alignment. In *Proceedings of the 15th Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 720–730, 2016. URL `http://www.aclweb.org/anthology/N16-1086`.

[76] Gábor Melis, Chris Dyer, and P. Blunsom. On the state of the art of evaluation in neural language models. *Proceedings of the International Conference on Learning Representations*, abs/1707.05589, 2018.

[77] Stephen Merity, N. Keskar, and R. Socher. Regularizing and optimizing lstm language models. *Proceedings of the International Conference on Learning Representations*, abs/1708.02182, 2018.

[78] Tomas Mikolov, M. Karafiát, L. Burget, J. Černocký, and S. Khudanpur. Recurrent neural network based language model. In *INTERSPEECH*, 2010.

[79] Amit Moryossef, Yoav Goldberg, and Ido Dagan. Step-by-step: Separating planning from realization in neural data-to-text generation. In *Proceed-*

*ings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2267–2277, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1236. URL https://www.aclweb.org/anthology/N19-1236.

[80] Soichiro Murakami, Akihiko Watanabe, Akira Miyazawa, Keiichi Goshima, Toshihiko Yanase, Hiroya Takamura, and Yusuke Miyao. Learning to generate market comments from stock prices. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1374–1384, 2017. URL http://aclweb.org/anthology/P17-1126.

[81] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, pages 807–814, 2010.

[82] Graham Neubig, Chris Dyer, Yoav Goldberg, Austin Matthews, Waleed Ammar, Antonios Anastasopoulos, Miguel Ballesteros, David Chiang, Daniel Clothiaux, Trevor Cohn, et al. Dynet: The dynamic neural network toolkit. *arXiv preprint arXiv:1701.03980*, 2017.

[83] Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, Raymond Hendy Susanto, and Christopher Bryant. The CoNLL-2014 Shared Task on Grammatical Error Correction. In *Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–14, Baltimore, Maryland, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-1701. URL https://www.aclweb.org/anthology/W14-1701.

[84] Feng Nie, Jinpeng Wang, Jin-Ge Yao, Rong Pan, and Chin-Yew Lin. Operation-guided Neural Networks for High Fidelity Data-To-Text Generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3879–3889, 2018. URL http://aclweb.org/anthology/D18-1422.

[85] Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. The E2E dataset:

New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5525. URL https://www.aclweb.org/anthology/W17-5525.

[86] Naoaki Okazaki, Yutaka Matsuo, and Mitsuru Ishizuka. Improving chronological sentence ordering by precedence relation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 750–756, 2004.

[87] Chris Olah and Shan Carter. Attention and augmented recurrent neural networks. *Distill*, 2016. doi: 10.23915/distill.00001. URL http://distill.pub/2016/augmented-rnns.

[88] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318, 2002. URL http://www.aclweb.org/anthology/P02-1040.

[89] Karl Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[90] Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. Text generation with exemplar-based adaptive decoding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2555–2565, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1263. URL https://www.aclweb.org/anthology/N19-1263.

[91] Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-Text Generation with Content Selection and Planning. In *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence*, 2019. URL https://www.aaai.org/Papers/AAAI/2019/AAAI-PuduppullyR.754.pdf.

[92] Ratish Puduppully, Li Dong, and Mirella Lapata. Data-to-text Generation with Entity Modeling. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2023–2035. Association for Computational Linguistics, 2019. URL `https://www.aclweb.org/anthology/P19-1195`.

[93] Guanghui Qin, Jin-Ge Yao, Xuening Wang, Jinpeng Wang, and Chin-Yew Lin. Learning latent semantic annotations for grounding natural language to structured data. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3761–3771, 2018.

[94] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. *OpenAI blog*, 2018.

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

[96] Clément Rebuffel, Laure Soulier, Geoffrey Scoutheeten, and Patrick Gallinari. A hierarchical model for data-to-text generation. In *European Conference on Information Retrieval*, pages 65–80. Springer, 2020.

[97] Sashank J Reddi, Satyen Kale, and Sanjiv Kumar. On the convergence of adam and beyond. In *Proceedings of the Sixth International Conference on Learning Representations*, 2018. URL `https://openreview.net/pdf?id=ryQu7f-RZ`.

[98] Ehud Reiter, Chris Mellish, and John Levine. Automatic generation of technical documentation. *Applied Artificial Intelligence an International Journal*, 9(3):259–287, 1995.

[99] R. Rosenfeld. Two decades of statistical language modeling: where do we go from here? *Proceedings of the IEEE*, 88:1270–1278, 2000.

[100] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th*

*Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, 2017.

[101] C. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27:379–423, 1948.

[102] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 464–468, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-2074. URL https://www.aclweb.org/anthology/N18-2074.

[103] Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. Style Transfer from Non-Parallel Text by Cross-Alignment. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 6830–6841. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7259-style-transfer-from-non-parallel-text-by-cross-alignment.pdf.

[104] Xiaoyu Shen, Ernie Chang, Hui Su, Cheng Niu, and Dietrich Klakow. Neural data-to-text generation via jointly learning the segmentation and correspondence. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7155–7165, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.641. URL https://www.aclweb.org/anthology/2020.acl-main.641.

[105] Michel Simard, Cyril Goutte, and Pierre Isabelle. Statistical phrase-based post-editing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 508–515, Rochester, New York, April 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/N07-1064.

[106] Richard Socher, Eric Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. *Advances in neural information processing systems*, 24:801–809, 2011.

[107] Nitish Srivastava, Geoffrey E. Hinton, A. Krizhevsky, Ilya Sutskever, and R. Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15:1929–1958, 2014.

[108] Andreas Stolcke. Srilm-an extensible language modeling toolkit. In *Seventh international conference on spoken language processing*, 2002.

[109] Sainbayar Sukhbaatar, Jason Weston, Rob Fergus, et al. End-to-end memory networks. In *Advances in neural information processing systems*, pages 2440–2448, 2015. URL `https://papers.nips.cc/paper/5846-end-to-end-memory-networks.pdf`.

[110] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. URL `http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf`.

[111] Kumiko Tanaka-Ishii, Kôiti Hasida, and Itsuki Noda. Reactive content selection in the generation of real-time soccer commentary. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics*, pages 1282–1288, 1998. URL `http://aclweb.org/anthology/P98-2209`.

[112] Zhaopeng Tu, Yang Liu, Zhengdong Lu, Xiaohua Liu, and Hang Li. Context gates for neural machine translation. *Transactions of the Association for Computational Linguistics*, 5:87–99, 2017. URL `http://www.aclweb.org/anthology/Q17-1007`.

[113] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. V. Luxburg, S. Bengio,

H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc., 2017. URL http://papers.nips.cc/paper/7181-attention-is-all-you-need.pdf.

[114] Hongmin Wang. Revisiting challenges in data-to-text generation with fact grounding. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 311–322, Tokyo, Japan, October–November 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-8639. URL https://www.aclweb.org/anthology/W19-8639.

[115] Zhenyi Wang, Xiaoyang Wang, Bang An, Dong Yu, and Changyou Chen. Towards faithful neural table-to-text generation with content-matching constraints. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1072–1086, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.101. URL https://www.aclweb.org/anthology/2020.acl-main.101.

[116] Taro Watanabe and Eiichiro Sumita. Transition-based neural constituent parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1169–1179, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1113. URL https://www.aclweb.org/anthology/P15-1113.

[117] David Weiss, Chris Alberti, Michael Collins, and Slav Petrov. Structured training for neural network transition-based parsing. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 323–333, Beijing, China, July 2015. Association for Computational Linguistics. doi: 10.3115/v1/P15-1032. URL https://www.aclweb.org/anthology/P15-1032.

[118] Sean Welleck, Ilia Kulikov, Stephen Roller, Emily Dinan, Kyunghyun Cho,

and Jason Weston. Neural text generation with unlikelihood training. In *International Conference on Learning Representations*, 2020.

[119] Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. Stochastic language generation in dialogue using recurrent neural networks with convolutional sentence reranking. In *Proceedings of the 16th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 275–284, Prague, Czech Republic, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-4639. URL https://www.aclweb.org/anthology/W15-4639.

[120] Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1711–1721, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi: 10.18653/v1/D15-1199. URL https://www.aclweb.org/anthology/D15-1199.

[121] Jason Weston, Sumit Chopra, and Antoine Bordes. Memory Networks. In *Proceedings of the Third International Conference on Learning Representations*, 2015. URL https://arxiv.org/pdf/1410.3916.pdf.

[122] Sam Wiseman, Stuart Shieber, and Alexander Rush. Challenges in Data-to-Document Generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2253–2263, 2017. URL http://aclweb.org/anthology/D17-1239.

[123] Sam Wiseman, Stuart Shieber, and Alexander Rush. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1356. URL https://www.aclweb.org/anthology/D18-1356.

[124] Sander Wubben, Antal van den Bosch, and Emiel Krahmer. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th*

*Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1015–1024, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P12-1107.

[125] Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4: 401–415, 2016. doi: 10.1162/tacl_a_00107. URL https://www.aclweb.org/anthology/Q16-1029.

[126] Diyi Yang, Aaron Halfaker, R. Kraut, and E. Hovy. Who did what: Editor role identification in wikipedia. In *The International AAAI Conference on Web and Social Media*, 2016.

[127] Diyi Yang, Aaron Halfaker, Robert Kraut, and Eduard Hovy. Edit categories and editor role identification in Wikipedia. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 1295–1299, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA). URL https://www.aclweb.org/anthology/L16-1206.

[128] Diyi Yang, Aaron Halfaker, Robert Kraut, and Eduard Hovy. Identifying semantic edit intentions from revisions in Wikipedia. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2000–2010, Copenhagen, Denmark, September 2017. Association for Computational Linguistics. doi: 10.18653/v1/D17-1213. URL https://www.aclweb.org/anthology/D17-1213.

[129] Zichao Yang, Phil Blunsom, Chris Dyer, and Wang Ling. Reference-Aware Language Models. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1850–1859, 2017. URL https://www.aclweb.org/anthology/D17-1197.

[130] Rong Ye, Wenxian Shi, Hao Zhou, Zhongyu Wei, and Lei Li. Variational template machine for data-to-text generation. In *International Conference on Learning Representations*, 2020.

[131] Pengcheng Yin, Graham Neubig, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L Gaunt. Learning to Represent Edits. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=BJl6AjC5F7.

[132] Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Russ R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pages 3391–3401, 2017.

[133] W. Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *Proceedings of the Third International Conference on Learning Representations*, 2015. URL https://arxiv.org/abs/1409.2329.

[134] Yue Zhang and Stephen Clark. Joint word segmentation and POS tagging using a single perceptron. In *Proceedings of ACL-08: HLT*, pages 888–896, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P08-1101.

[135] Chao Zhao, Marilyn Walker, and Snigdha Chaturvedi. Bridging the structural gap between encoding and decoding for data-to-text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2481–2491, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.224. URL https://www.aclweb.org/anthology/2020.acl-main.224.

[136] Sanqiang Zhao, Rui Meng, Daqing He, Andi Saptono, and Bambang Parmanto. Integrating Transformer and Paraphrase Rules for Sentence Simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3164–3173, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1355. URL https://www.aclweb.org/anthology/D18-1355.

[137] Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. "Improving Grammatical Error Correction via Pre-Training a Copy-Augmented Architecture with Unlabeled Data". In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and*

*Short Papers)*, pages 156–165, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1014. URL https://www.aclweb.org/anthology/N19-1014.