Building effective text generation systems requires three critical components: content selection, text planning, and surface realization, and traditionally they are tackled as separate problems. Recent all-in-one style neural generation models have made impressive progress, yet they often produce outputs that are incoherent and unfaithful to the input. To address these issues, we present an end-to-end trained two-step generation model, where a sentence-level content planner first decides on the keyphrases to cover as well as a desired language style, followed by a surface realization decoder that generates relevant and coherent text. For experiments, we consider three tasks from domains with diverse topics and varying language styles: persuasive argument construction from Reddit, paragraph generation for normal and simple versions of Wikipedia, and abstract generation for scientific articles. Automatic evaluation shows that our system can significantly outperform competitive comparisons. Human judges further rate our system generated text as more fluent and correct, compared to the generations by its variants that do not consider language style.

1 Introduction

Automatic text generation is a long-standing challenging task, as it needs to solve at least three major problems: (1) content selection (“what to say”), identifying pertinent information to present, (2) text planning (“when to say what”), arranging content into ordered sentences, and (3) surface realization (“how to say it”), deciding words and syntactic structures that deliver a coherent output based on given discourse goals (McKeown, 1985). Traditional text generation systems often handle each component separately, thus requiring extensive effort on data acquisition and system engineering (Reiter and Dale, 2000). Recent progress has been made by developing end-to-end trained neural models (Rush et al., 2015; Yu et al., 2018; Fan et al., 2018), which naturally excel at producing fluent text. Nonetheless, limitations of model structures and training objectives make them suffer from low interpretability and substandard generations which are often incoherent and unfaithful to the input material (See et al., 2017; Wiseman et al., 2017; Li et al., 2017).

To address the problems, we believe it is imperative for neural models to gain adequate control on content planning (i.e., content selection and ordering) to produce coherent output, especially for long text generation. We further argue that, in order to achieve desired discourse goals, it is beneficial to enable style-controlled surface realization by explicitly modeling and specifying proper linguistic styles. Consider the task of producing counter-arguments to the topic “US should cut off foreign aid completely”. A sample argument in Figure 1 demonstrates how human selects...
A few years ago, the US cut financial aid to Uganda... make homosexuality a crime punishable by death

Figure 2: Overview of our framework. The LSTM content planning decoder (§ 3.2) first identifies a set of keyphrases from the memory bank conditional on previous selection history, based on which, a style is specified. During surface realization, the hidden states of the planning decoder and the predicted style encoding are fed into the realizer, which generates the final output (§ 3.3). Best viewed in color.

We thus present an end-to-end trained neural text generation framework that includes the modeling of traditional generation components, to promote the control of content and linguistic style of the produced text. Our model performs sentence-level content planning for information selection and ordering, and style-controlled surface realization to produce the final generation. We focus on conditional text generation problems (Lebret et al., 2016; Colin et al., 2016; Dušek et al., 2018): As shown in Figure 2, the input to our model consists of a topic statement and a set of keyphrases. The output is a relevant and coherent paragraph to reflect the salient points from the input. We utilize two separate decoders: for each sentence, (1) a planning decoder selects relevant keyphrases and a desired style conditional on previous selections, and (2) a realization decoder produces the text in the specified style.

We demonstrate the effectiveness of our framework on three challenging datasets with diverse topics and varying linguistic styles: persuasive argument generation on Reddit ChangeMyView (Hua and Wang, 2018); introduction paragraph generation on a newly collected dataset from Wikipedia and its simple version; and scientific paper abstract generation on AGENDA dataset (Koncel-Kedziorski et al., 2019).

Experimental results on all three datasets show that our models that consider content planning and style selection achieve significantly better BLEU, ROUGE, and METEOR scores than non-trivial comparisons that do not consider such information. Human judges also rate our model generations as more fluent and correct compared to the outputs produced by its variants without style modeling.

2 Related Work

Content selection and text planning are critical components in traditional text generation systems (Reiter and Dale, 2000). Early approaches separately construct each module and mainly rely on hand-crafted rules based on discourse theory (Scott and de Souza, 1990; Hovy, 1993) and expert knowledge (Reiter et al., 2000), or train statistical classifiers with rich features (Duboue and McKeown, 2003; Barzilay and Lapata, 2005). Advances in neural generation models have alleviated human efforts on system engineering, by combining all components into an end-to-end trained conditional text generation framework (Mei et al., 2016; Wiseman et al., 2017). However, without
proper planning and control (Rambow and Korel-
sky, 1992; Stone and Doran, 1997; Walker et al.,
2001), the outputs are often found to be incoherent
and hallucinating. Recent work (Moryossef et al.,
2019) separates content selection from the neu-
ral generation process and shows improved gen-
eration quality. However, their method requires
an exhaustive search for content ordering and is
therefore hard to generalize and scale. In this
work, we improve the content selection by incor-
porating past selection history and directly feed-
ing the predicted language style into the realiza-
tion module.

Our work is also inline with concept-to-
text generation, where sentences are produced
from structured representations, such as database
records (Konstas and Lapata, 2013; Lebret et al.,
2016; Wiseman et al., 2017; Moryossef et al.,
2019), knowledge base items (Luan et al.,
2018; Koncel-Kedzioriski et al., 2019), and AMR
graphs (Konstas et al., 2017; Song et al., 2018;
Koncel-Kedzioriski et al., 2019). Shared tasks
such as WebNLG (Colin et al., 2016) and E2E
NLG challenges (Duˇsek et al., 2019) have been
designed to evaluate single sentence planning and
realization from the given structured inputs with
a small set of fixed attribute types. Planning for
multiple sentences in the same paragraph is nev-
evertheless much less studied; it poses extra chal-
enges for generating coherent long text, which
is addressed in this work. Moreover, structured
inputs are only available in a limited number of
domains (Tanaka-Ishii et al., 1998; Chen and
Mooney, 2008; Belz, 2008; Liang et al., 2009;
Chisholm et al., 2017). The emerging trend is to
explore less structured data (Kiddon et al., 2016;
Fan et al., 2018; Martin et al., 2018). In our work,
keyphrases are used as input to our generation sys-
tem, which offers flexibility for concept representa-
tion and generalizability to broader domains.

3 Model

Our model tackles conditional text generation
tasks where the input is comprised of two major
parts: (1) a topic statement, \( x = \{x_i\} \), which can
be an argument, the title of a Wikipedia article, or
a scientific paper title, and (2) a keyphrase mem-
ory bank, \( \mathcal{M} \), containing a list of talking points,
which plays a critical role in content planning and
style selection. We aim to produce a sequence of
words, \( y = \{y_t\} \), to comprise the output,
which can be a counter-argument, a paragraph as
in Wikipedia articles, or a paper abstract.

3.1 Input Encoding

The input text \( x \) is encoded via a bidirectional
LSTM (biLSTM), with its last hidden state used
as the initial states for both content planning de-
coder and surface realization decoder. To en-
code keyphrases in the memory bank \( \mathcal{M} \), each
keyphrase is first converted into a vector \( e_k \)
by summing up all its words’ embeddings from
GloVe (Pennington et al., 2014). A biLSTM-based
keyphrase reader, with hidden states \( h^e_{k} \), is
used to encode all keyphrases in \( \mathcal{M} \). We also insert en-
tries of \(<\text{START}>\) and \(<\text{END}>\) into \( \mathcal{M} \) to facilitate
learning to start and finish selection.

3.2 Sentence-Level Content Planning and
Style Specification

Content Planning: Context-Aware Keyphrase
Selection. Our content planner selects a set
of keyphrases from the memory bank \( \mathcal{M} \) for
each sentence, indexed with \( j \), conditional on
keyphrases that have been selected in previous
sentences, allowing topical coherence and content
repetition avoidance. The decisions are denoted
as a selection vector \( v_j \in \mathbb{R}^{|
\mathcal{M}|} \), with each di-

mension \( v_{j,k} \in \{0, 1\} \), indicating whether the \( k \)-
th phrase is selected for the \( j \)-th sentence gen-
eration. Starting with a \(<\text{START}>\) tag as the input
for the first step, our planner predicts \( v_1 \) for the
first sentence, and recurrently makes predictions
per sentence until \(<\text{END}>\) is selected, as depicted
in Figure 2.

Formally, we utilize a sentence-level LSTM \( f \),
which consumes the summation embedding of se-
lected keyphrases, \( m_j \), to produce a hidden state
\( s_j \) for the \( j \)-th sentence step:

\[
    s_j = f(s_{j-1}, m_j) \tag{1}
\]

\[
    m_j = \sum_{k=1}^{|\mathcal{M}|} v_{j,k} h^e_k \tag{2}
\]

where \( v_{j,k} \in \{0, 1\} \) is the selection decision for
the \( k \)-th keyphrase in the \( j \)-th sentence.

Our recent work (Hua et al., 2019) utilizes a
similar formulation for sentence representations.
However, the prediction of \( v_{j+1} \) is estimated by a
bilinear product between \( h^e_k \) and \( s_j \), which is ag-
nostic to what have been selected so far. While
in reality, content selection for a new sentence
should depend on previous selections. For in-
stance, keyphrases that have already been utilized many times are less likely to be picked again; topically related concepts tend to be mentioned closely. We therefore propose a vector $q_j$ that keeps track of what keyphrases have been selected up to the $j$-th sentence:

$$q_j = (\sum_{r=0}^{j} v_r)^T \times E \quad (3)$$

where $E = [h_1^c, h_2^c, \ldots, h_{|M|}^c]^T \in \mathbb{R}^{|M| \times H}$ is the matrix of keyphrase representations, $H$ is the hidden dimension of the keyphrase reader LSTM.

Then $v_{j+1}$ is calculated in an attentive manner with $q_j$ as the attention query:

$$P(v_{j+1,k} = 1|v_{1:j}) = \sigma(w^c_{s} s_j + q_j W^c h_k^c) \quad (4)$$

where $\sigma$ is the sigmoid function, and $w_s$, $W^c$, and $W^{**}$ are trainable parameters throughout the paper. Bias terms are all omitted for simplicity.

As part of the learning objective, we utilize the binary cross-entropy loss with the gold-standard selection $v^*_j$ as criterion over the set $D$:

$$\mathcal{L}_{sel} = - \sum_{(x,y) \in D} \sum_{j=1}^{J} \sum_{k=1}^{|M|} \log(P(v^*_j,k))) \quad (5)$$

**Style Specification.** As discussed in § 1, depending on the content (represented as selected keyphrases in our model), humans often choose different language styles adapted for different discourse goals. Our model characterizes such stylistic variations by assigning a categorical style type $t_j$ for each sentence, which is predicted as follows:

$$t_j = \text{softmax}(w^s_{s} (\text{tanh}(W^{s}\theta; s_j))) \quad (6)$$

$t_j$ is the estimated distribution over all types. We select the one with the highest probability and use a one-hot encoding vector, $t_j$, as the input to our realization decoder (§ 3.3). The estimated distributions $t_j$ are compared against the gold-standard labels $t^*_j$ to calculate the cross-entropy loss $\mathcal{L}_{style}$:

$$\mathcal{L}_{style} = - \sum_{(x,y) \in D} \sum_{j=1}^{J} t^*_j \log t_j \quad (7)$$

### 3.3 Style-Controlled Surface Realization

Our surface realization decoder is implemented with an LSTM with state calculation function $g$ to get each hidden state $z_t$ for the $t$-th generated token. To compute $z_t$, we incorporate the content planning decoder hidden state $s_{J(t)}$ for the sentence to be generated, with $J(t)$ as the sentence index, and previously generated token $y_{t-1}$:

$$z_t = g(z_{t-1}, \text{tanh}(W^{w}s_{J(t)} + W^{ww}y_{t-1})) \quad (8)$$

For word prediction, we calculate two attentions, one over the input statement $x$, which produces a context vector $c^w_t$ (Eq. 10), the other over the keyphrase memory bank $M$, which generates $c^e_t$ (Eq. 11). To better reflect the control over word choice by language styles, we directly append the predicted style $t_{J(t)}$ to the context vectors and hidden state $z_t$, to compute the distribution over the vocabulary$^2$:

$$\hat{p}(y_t|y_{1:t-1}) = \text{softmax}(\text{tanh}(W^o[z_t; c^w_t; c^e_t; t_{J(t)}])) \quad (9)$$

$$c^w_t = \sum_{i=1}^{L} \alpha^w_i h_i, \quad \alpha^w_i = \text{softmax}(z_t W^{w} h_i) \quad (10)$$

$$c^e_t = \sum_{k=1}^{|M|} \alpha^e_k h^e_k, \quad \alpha^e_k = \text{softmax}(z_t W^{we} h^e_k) \quad (11)$$

We further adopt a copying mechanism from See et al. (2017) to enable direct reuse of words from the input $x$ and keyphrase bank $M$ to allow out-of-vocabulary words to be included.

### 3.4 Training Objective

We jointly learn to conduct content planning and surface realization by aggregating the losses over (i) word generation: $\mathcal{L}_{gen} = \frac{-1}{|D|} \sum_{i=1}^{T} \log P(y^i_t|x; \theta)$, (ii) keyphrase selection: $\mathcal{L}_{sel}$ (Eq. 5), and (iii) style prediction $\mathcal{L}_{style}$ (Eq. 7):

$$\mathcal{L}(\theta) = \mathcal{L}_{gen}(\theta) + \gamma \cdot \mathcal{L}_{style}(\theta) + \eta \cdot \mathcal{L}_{sel}(\theta) \quad (12)$$

where $\theta$ denotes the trainable parameters. $\gamma$ and $\eta$ are set to 1.0 in our experiments for simplicity.

### 4 Tasks and Datasets

#### 4.1 Task I: Argument Generation

Our first task is to generate a counter-argument for a given statement on a controversial issue. The input keyphrases are extracted from automatically retrieved and reranked passages with queries constructed from the input statement.

---

$^2$The inclusion of style variables is different from our prior style-aware generation model (Hua et al., 2019), where styles are predicted but not encoded for word production.
We reuse the dataset from our previous work (Hua et al., 2019), but annotate with newly designed style scheme. We first briefly summarize the procedures for data collection, keyphrase extraction and selection, and passage reranking; more details can be found in our prior work. Then we describe how to label argument sentences with style types that capture argumentative structures.

The dataset is collected from Reddit /r/ChangeMyView subcommunity, where each thread consists of a multi-paragraph original post (OP), followed by user replies with the intention to change the opinion of the OP user. Each OP is considered as the input, and the root replies awarded with delta ($\Delta$), or with positive karma (# upvotes > # downvotes) are target counter-arguments to be generated. A domain classifier is further adopted to select politics related threads. Since users often have separate arguments in different paragraphs, we treat each paragraph as one target argument by itself. Statistics are shown in Table 1.

Input Keyphrases and Label Construction. To obtain the input keyphrase candidates and their sentence-level selection labels, we first construct queries to retrieve passages from Wikipedia and news articles collected from commoncrawl.org. For training, we construct a query per target argument sentence using its content words for retrieval, and keep top 5 passages per query. For testing, the queries are constructed from the sentences in OP (input statement).

We then extract keyphrases from the retrieved passages based on topic signature words (Lin and Hovy, 2000) calculated over the given OP. These words, together with their related terms from WordNet (Miller, 1994), are used to determine whether a phrase in the passage is a keyphrase. Specifically, a keyphrase is (1) a noun phrase or verb phrase that is shorter than 10 tokens; (2) contains at least one content word; (3) has a topic signature or a Wikipedia title. For each keyphrase candidate, we match them with the sentences in the target counter-argument, and we consider it to be “selected” for the sentence if there is any overlapping content word.

During test time, we further adopt a stance classifier from Bar-Haim et al. (2017) to produce a stance score for each passage. We retain passages that have a negative stance towards OP, and a greater than 5 stance score. They are further ordered based on the number of overlapping keyphrases with the OP. Top 10 passages are used to construct the input keyphrase bank, and as optional input to our model.

Sentence Style Label Construction. For argument generation, we define three sentence styles based on their argumentative discourse functions (Persing and Ng, 2016; Lippi and Torroni, 2016): CLAIM is a proposition, usually containing one or two talking points, e.g., “I believe foreign aid is a useful bargaining chip”; PREMISE contains supporting arguments with reasoning or examples; FUNCTIONAL is usually a generic statement, e.g., “I understand what you said”. For training, we employ a list of rules extended from the claim detection method by Levy et al. (2018) to automatically construct a style label for each sentence. Statistics are displayed in Table 2, and sample rules are shown below, with the complete list in the Supplementary:

- **CLAIM**: must be shorter than 20 tokens and matches any of the following patterns: (a) i (don’t)? (believe|agree|...); (b) (anyone|all|everyone|nobody...) (should|could|need|must|might|...);
(c) (in my opinion|my view|...)

- **PREMISE**: must be longer than 5 tokens, contains at least one noun or verb content word, and matches any of the following patterns: (a) (for (example|instance)|e.g.); (b) (increase|reduce|improve|...)

- **FUNCTIONAL**: contains fewer than 5 alphabetical words and no noun or verb content word

Paragraphs that only contain FUNCTIONAL sentences are removed from our dataset.

### 4.2 Task II: Paragraph Generation for Normal and Simple Wikipedia

The second task is generating introduction paragraphs for Wikipedia articles. The input consists of a title, a user-specified global style (normal or simple), and a list of keyphrases collected from the gold-standard paragraphs of both normal and simple Wikipedia. During training and testing, the global style is encoded as one extra bit appended to \( m_j \) (Eq. 2).

We construct a new dataset with topically-aligned paragraphs from normal and simple English Wikipedia.\(^4\) For alignment, we consider it a match if two articles share exactly the same title with at most two non-English words. We then extract the first paragraphs from both and filter out the pair if one of the paragraphs is shorter than 10 words or is followed by a table.

**Input Keyphrases and Label Construction.** Similar to argument generation, we extract noun phrases and verb phrases and consider the ones with at least one content word as keyphrase candidates. After de-duplication, there are on average 5.4 and 3.7 keyphrases per sentence for the normal and simple Wikipedia paragraphs, respectively. For each sample, we merge the keyphrases from the aligned paragraphs as the input. The model is then trained to select the appropriate ones conditioned on the global style.

**Sentence Style Label Construction.** We distinguish sentence-level styles based on language complexity, which is approximated by sentence length. The distribution of sentence styles is displayed in Table 2.

### 4.3 Task III: Paper Abstract Generation

We further consider a task of generating abstracts for scientific papers (Ammar et al., 2018), where the input contains a paper title and scientific entities mentioned in the abstract. We use the AGENDA data processed by Koncel-Kedziorski et al. (2019), where entities and their relations in the abstracts are extracted by SciIE (Luan et al., 2018). All entities appearing in the abstract are included in our keyphrase bank. The state-of-the-art system (Koncel-Kedziorski et al., 2019) exploits the scientific entities, their relations, and the relation types. In our setup, we ignore the relation graph, and focus on generating the abstract with only entities and title as the input. Due to the dataset’s relatively uniform language style and smaller size, we do not experiment with our style specification component.

### 5 Experiments

#### 5.1 Implementation Details

For argument generation, we truncate the input and retrieved passages to 500 and 400 words. Passages are optionally appended to OP as our encoder input. The keyphrase bank size is limited to 70 for argument, and 30 for Wikipedia and AGENDA data (based on the average numbers in Table 1), with keyphrases truncated to 10 words.

We use a vocabulary size of 50K for all tasks.

**Training Details.** Our models use a two-layer LSTM for both decoders. They all have 512-dimensional hidden states per layer and dropout probabilities (Gal and Ghahramani, 2016) of 0.2 between layers. Wikipedia titles are encoded with the summation of word embeddings due to their short length. The learning process is driven by AdaGrad (Duchi et al., 2011) with 0.15 as the learning rate and 0.1 as the initial accumulator. We clip the gradient norm to a maximum of 2.0. The mini-batch size is set to 64. And the optimal weights are chosen based on the validation loss.

For argument generation, we also pre-train the encoder and the lower layer of realization decoder using language model losses. We collect all the OP posts from the training set, and an extended set of reply paragraphs, which includes additional counter-arguments that have non-negative karma. For Wikipedia, we consider the large collection of 1.9 million unpaired normal English Wikipedia paragraphs to pre-train the model for both normal and simple Wikipedia generation.

**Beam Search Decoding.** For inference, we utilize beam search with a beam size of 5. We disallow the repetition of trigrams, and replace the UNK

\(^4\)We download the dumps of 2019/04/01 for both dataset.
5.2 Baselines and Comparisons

For all three tasks, we consider a SEQ2SEQ with attention baseline (Bahdanau et al., 2015), which encodes the input text and keyphrase bank as a sequence of tokens, and generates the output.

For argument generation, we implement a RETRIEVAL baseline, which returns the highest reranked passage retrieved with OP as the query. We also compare with our prior model (Hua and Wang, 2018), which is a multi-task learning framework to generate both keyphrases and arguments.

For Wikipedia generation, a RETRIEVAL baseline obtains the most similar paragraph from the training set with input title and keyphrases as the query, measured with bigram cosine similarity. We further train a logistic regression model (LOGREGSEL), which takes the summation of word embeddings in a phrase and predicts its inclusion in the output for a normal or simple Wiki paragraph.

For abstract generation, we compare with the state-of-the-art system GRAPHWRITER (Koncel-Kedziorski et al., 2019), which is a transformer model enabled with knowledge graph encoding mechanism to handle both the entities and their structural relations from the input.

We also report results by our model variants to demonstrate the usefulness of content planning and style control: (1) with gold-standard keyphrase selection for each sentence (Oracle Plan.), and (2) without style specification.

6 Results and Analysis

6.1 Automatic Evaluation

We report precision-oriented BLEU (Papineni et al., 2002), recall-oriented ROUGE-L (Lin, 2004) that measures the longest common subsequence, and METEOR (Denkowski and Lavie, 2014), which considers both precision and recall.

Argument Generation. For each input OP, there can be multiple possible counter-arguments. We thus consider the best matched (i.e., highest scored) reference when reporting results in Table 3. Our models yield significantly higher BLEU and ROUGE scores than all comparisons while producing longer arguments than generation-based approaches. Furthermore, among our model variants, oracle content planning further improves the performance, indicating the importance of content selection and ordering. Taking out style specification decreases scores, indicating the influence of style control on generation.

Wikipedia Generation. Results on Wikipedia (Table 4) show similar trends, where our models almost always outperform all comparisons across metrics. The significant performance drop on ablated models without style prediction proves the effectiveness of style usage. Our model, if guided with oracle keyphrase selection per sentence, again achieves the best performance.

We further show the effect of content selection on generation on Wikipedia and abstract data in Figure 3, where we group the test samples into 10 bins based on F1 scores on keyphrase selection. We observe a strong correlation between keyphrase selection and generation performance, e.g., for BLEU, Pearson correlations of 0.95 ($p < 10^{-4}$) and 0.85 ($p < 10^{-2}$) are established for Wikipedia and abstract. For ROUGE, the values are 0.99 ($p < 10^{-8}$) and 0.72 ($p < 10^{-1}$).

Abstract Generation. Lastly, we compare with the state-of-the-art GRAPHWRITER model on AGENDA dataset in Table 5. Although our model does not make use of the relational graph encoding, taking out style selection and ordering. Taking out style specification decreases scores, indicating the influence of style control on generation.

We do not compare with our recent model in Hua et al. (2019) due to the training data difference caused by our new sentence style scheme. However, the newly proposed model generates arguments with lengths closer to human arguments, benefiting from the improved content planning module.

We calculate F1 by aggregating the selections across all sentences. For argument generation, keyphrases are often paraphrased, making it difficult to calculate F1 reliably, therefore omitted here.

Table 3: Results on argument generation with BLEU (up to bigrams), ROUGE-L, and METEOR (MTR). Best systems without oracle planning are in bold per metric. Our models that are significantly better than all comparisons are marked with * ($p < 0.001$, approximate randomization test (Noreen, 1989)).

<table>
<thead>
<tr>
<th></th>
<th>BLEU</th>
<th>ROUGE</th>
<th>MTR Len.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETRIEVAL</td>
<td>7.81</td>
<td>15.68</td>
<td>10.59</td>
</tr>
<tr>
<td>SEQ2SEQ</td>
<td>3.64</td>
<td>19.00</td>
<td>9.85</td>
</tr>
<tr>
<td>H&amp;W (2018)</td>
<td>5.73</td>
<td>14.44</td>
<td>3.82</td>
</tr>
<tr>
<td>OURS (Oracle Plan.)</td>
<td>16.30*</td>
<td>20.25*</td>
<td>11.61</td>
</tr>
<tr>
<td>OURS</td>
<td>13.19*</td>
<td>20.15*</td>
<td>10.42</td>
</tr>
<tr>
<td>w/o Style</td>
<td>12.61*</td>
<td>20.28*</td>
<td>10.15</td>
</tr>
<tr>
<td>w/o Passage</td>
<td>11.84*</td>
<td>19.90*</td>
<td>9.03</td>
</tr>
</tbody>
</table>

Table 4: Results on Wikipedia Generation (up to bigrams), ROUGE-L, and METEOR (MTR). Best systems without oracle planning are in bold per metric. Our models that are significantly better than all comparisons are marked with * ($p < 0.001$, approximate randomization test (Noreen, 1989)).

<table>
<thead>
<tr>
<th></th>
<th>BLEU</th>
<th>ROUGE</th>
<th>MTR Len.</th>
</tr>
</thead>
<tbody>
<tr>
<td>RETRIEVAL</td>
<td>7.81</td>
<td>15.68</td>
<td>10.59</td>
</tr>
<tr>
<td>SEQ2SEQ</td>
<td>3.64</td>
<td>19.00</td>
<td>9.85</td>
</tr>
<tr>
<td>H&amp;W (2018)</td>
<td>5.73</td>
<td>14.44</td>
<td>3.82</td>
</tr>
<tr>
<td>OURS (Oracle Plan.)</td>
<td>16.30*</td>
<td>20.25*</td>
<td>11.61</td>
</tr>
<tr>
<td>OURS</td>
<td>13.19*</td>
<td>20.15*</td>
<td>10.42</td>
</tr>
<tr>
<td>w/o Style</td>
<td>12.61*</td>
<td>20.28*</td>
<td>10.15</td>
</tr>
<tr>
<td>w/o Passage</td>
<td>11.84*</td>
<td>19.90*</td>
<td>9.03</td>
</tr>
</tbody>
</table>

5"Gold-standard" indicates the keyphrases that have content word overlap with the reference sentence.
Table 4: Results on Wikipedia generation. Best results without oracle planning are in bold. *: Our models that are significantly better than all comparisons (p < 0.001, approximate randomization test).

Table 5: Results on paper abstract generation. Notice that GraphWriter models rich information about relations and relation types among entities, which is not utilized by our model.

Table 6: Human evaluation on argument generation (Upper) and Wikipedia generation (Bottom). Grammaticality (Gram), correctness (Corr), and content richness (Cont) are rated on Likert scale (1 – 5). We mark our model with * to indicate statistically significantly better ratings over the variant without style specification (p < 0.001, approximate randomization test).

6.2 Human Evaluation

We further ask three proficient English speakers to assess the quality of generated arguments and Wikipedia paragraphs. Human subjects are asked to rate on a scale of 1 (worst) to 5 (best) on grammaticality, correctness of the text (for arguments, the stance is also considered), and content richness (i.e., coverage of relevant points). Detailed guidelines for different ratings are provided to the raters (see Supplementary). For both tasks, we randomly choose 30 samples from the test set; out-

6.3 Further Analysis and Discussions

We further investigate the usage of different styles, and show the top frequent patterns for each argument style from human arguments and our system.

Figure 3: Effect of keyphrase selection (F1 score) on generation performance, measured by BLEU and ROUGE. Positive correlations are observed.

Figure 4: Sample outputs from two variants of our models and a human written text are presented in random order.

According to Krippendorff’s α, the raters achieve substantial agreement on grammaticality and correctness, while the agreement on content richness is only moderate due to its subjectivity. As shown in Table 6, on both tasks, our models with style specification produce more fluent and correct generations, compared to the ones without such information. However, there is still a gap between system generations and human edited text.

We further show sample outputs in Figure 4. The first example is on the topic of abortion, our models that are significantly better than all comparisons (p < 0.001, approximate randomization test).

It also contains fewer repetitions than the seq2seq baseline. For Wikipedia, our model is not only better at controlling the global simplicity style, but also more grammatical and coherent than the seq2seq baseline. We further show sample outputs in Figure 4. The first example is on the topic of abortion, our models that are significantly better than all comparisons (p < 0.001, approximate randomization test).

As shown in Table 6, on both tasks, our models with style specification produce more fluent and correct generations, compared to the ones without such information. However, there is still a gap between system generations and human edited text.

We further show sample outputs in Figure 4. The first example is on the topic of abortion, our model captures the relevant concepts such as “fetuses are not fully developed” and “illegal to kill”. It also contains fewer repetitions than the seq2seq baseline. For Wikipedia, our model is not only better at controlling the global simplicity style, but also more grammatical and coherent than the seq2seq output.

6.3 Further Analysis and Discussions

We further investigate the usage of different styles, and show the top frequent patterns for each argument style from human arguments and our system generation (Table 7). We first calculate the most
frequent 4-grams per style, then extend it with context. We manually cluster and show the representative ones. For both columns, the popular patterns reflect the corresponding discourse functions: CLAIM is more evaluative, PREMISE lists out details, and FUNCTIONAL exhibits argumentative stylistic languages. Interestingly, our model also learns to paraphrase popular patterns, e.g., “have the freedom to” vs. “have the right to”.

For Wikipedia, the sentence style is defined by length. To validate its effect on content selection, we calculate the average number of keyphrases per style type. The results on human written paragraphs are 2.0, 3.8, 5.8, and 9.0 from the simplest to the most complex. A similar trend is observed in our model outputs, which indicates the challenge of content selection in longer sentences.

For future work, improvements are needed in both model design and evaluation. As shown in Figure 4, system arguments appear to overfit on stylistic languages and rarely create novel concepts like humans do. Future work can lead to improved model guidance and training methods, such as reinforcement learning-based explorations, and better evaluation to capture diversity.

7 Conclusion

We present an end-to-end trained neural text generation model that considers sentence-level content planning and style specification to gain better control of text generation quality. Our content planner first identifies salient keyphrases and a proper language style for each sentence, then the realization decoder produces fluent text. We consider three tasks of different domains on persuasive argument generation, paragraph generation for normal and simple versions of Wikipedia, and abstract generation for scientific papers. Experimental results demonstrate the effectiveness of our model, where it obtains significantly better BLEU, ROUGE, and METEOR scores than non-trivial comparisons. Human subjects also rate our model generations as more grammatical and correct when language style is considered.

Acknowledgements

This research is supported in part by National Science Foundation through Grants IIS-1566382 and IIS-1813341, and Nvidia GPU gifts. We are grateful to Rik Koncel-Kedziorski and Hannaneh Hajishirzi for sharing their system outputs. We also thank anonymous reviewers for their valuable suggestions.
References


