Unsupervised Controllable Text Formalization

Parag Jain, Abhijit Mishra, Amar Prakash Azad, Karthik Sankaranarayanan
IBM Research
{pajain34,abhijimi,amarazad,kartsank}@in.ibm.com

Abstract

We propose a novel framework for controllable natural language transformation. Realizing that the requirement of parallel corpus is practically unsustainable for controllable generation tasks, an unsupervised training scheme is introduced. The crux of the framework is a deep neural encoder-decoder that is reinforced with text-transformation knowledge through auxiliary modules (called *scorers*). These scorers, based on off-the-shelf language processing tools, decide the learning scheme of the encoder-decoder based on its actions. We apply this framework for the text-transformation task of formalizing an input text by improving its readability grade; the degree of required formalization can be controlled by the user at run-time. Experiments on public datasets demonstrate the efficacy of our model towards: (a) transforming a given text to a more formal style, and (b) varying the amount of formalness in the output text based on the specified input control. Our code and datasets are released for academic use.

1 Introduction

Automatic text style-transformation is one of the key goals of *text-to-text* natural language generation (NLG) research and most existing systems for such tasks are either supervised (*e.g.,* variants of *Seq2Seq* neural models (Sutskever, Vinyals, and Le 2014; Bahdanau, Cho, and Bengio 2014), or Statistical Machine Translation models (Koehn 2009)) or template/rule based (Gatt and Reiter 2009). Supervised NLG requires large-scale parallel corpora for training, which is a major impediment in scaling to diverse use-cases. For example, in the context of commercial dialog systems alone, there are several scenarios where a system's answer (which may be coming from a database (Jain et al. 2018)) needs to be transformed either for its *tone* (politeness, excitedness, *etc.*), or its level of *formality* (casual, formal, *etc.* based on the user's personality), or for its *complexity* (simplifying linguistic or domain-specific terminology such as in legal or medical domains). As such requirements and use-cases keep growing, it is practically unsustainable to obtain large scale parallel corpora for each such text transformation task.

From a scientific perspective, a supervised treatment of all such tasks using several parallel corpora seeks to learn both the language transformation (while preserving semantics), as well as the style transformation, simultaneously for each task. We make 3 key observations with respect to this: (i) since the preservation of language semantics is necessary for transformation, whereas only the attribute or style of the text needs to be changed, it should be possible to decouple these two aspects, (ii) it should be cheaper to independently *verify* these aspects at the output (with well-understood NLP techniques) than to *specify* the required transformation for each input text with its output example, and (iii) it should be possible to *control* the degree or magnitude of the intended attribute (readability level, politeness level, *etc.*) required at the output. These observations motivate us to seek an unsupervised approach to such a gamut of text transformation tasks and underlie the proposed NLG framework.

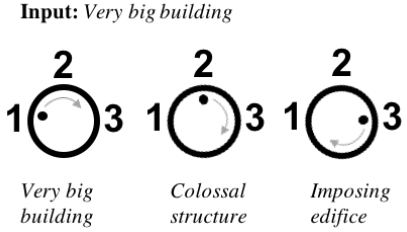

Figure 1: Controllable formalization showing an anecdotal snippet and its expected variations w.r.t. input controls

Our proposed framework relies only on unlabeled texts for initialization and an ensemble of *off-the-shelf* language processing modules. We test our framework for the task of *formalizing the input text*, where the readability of the output text is improved while preserving its meaning (refer Figure 1 for an example). The degree of formalization required may be decided by the user during run-time and is provided as a control input to the system. This task is chosen due to its relevance in many NLG applications such as formal conversation generation, email response composition, or summary document generation in compliance and regulatory domains *etc*. Moreover, such a standalone system can provide assistance to professional writers, the same way Computer Assisted Translation (CAT) systems currently assist human translators. This paves the way for cost- and time-effective solutions for textual content creation.

Our framework is based on an encoder-decoder module (Bahdanau, Cho, and Bengio 2014), which is pre-trained with unlabeled texts. **The decoder additionally takes user-specified control as input**. Further, knowledge of the required text-transformation is acquired through auxiliary modules (called *scorers*) which decide the learning scheme of the encoder-decoder based on its actions. These scorers are based on readily available natural language processing (NLP) tools which can produce scores indicating: (a) how formal the generated text is, (b) whether the generated text is fluent, and most importantly, (c) whether the generated text carries similar semantics as the input. This framework is trained in multiple iterations, where each iteration is comprised of two phases of **(i) exploration** and **(ii) exploitation**. In the exploration phase, the decoder randomly samples candidate texts for given inputs, and with the help of scorers, automatically produces training data for controllable generation. In the exploitation phase, the encoder-decoder is re-trained with the examples thus generated.

For experiments, we prepare a mixture of unlabeled informal texts with low readability grade. Our NLP tools measure readability, adequacy and fluency. With this setup, we observe that (a) the system generated transformed texts are more formal than the input, and (b) their degree of formalness conforms to the input controls provided. The efficacy of our system is demonstrated through both qualitative and quantitative evaluations, in terms of human judgments and various NLG evaluation metrics. We also show the system's effectiveness for another relevant task of text complexification (reversed simplification). The source code and dataset are publicly available.

2 Related Work

Unsupervised NLG has always been more challenging due to the fact that: (i) the output space is more complex and structured, making unsupervised learning more difficult, and (ii) metrics for evaluating NLG systems without reference output text are elusive. Recently, Artetxe et al. (2017) and Lample, Denoyer, and Ranzato (2017) have proposed architectures for unsupervised language translation with unsupervised autoencoder based lexicon induction techniques. This approach primarily focuses on cross-lingual transformation and requires multiple unlabeled corpora from different languages. It can not trivially be extended to our setup to achieve a controllable text-transformation goal within a single language, and further there is no notion of control in language translation. A notable work by Hu et al. (2017) discusses controllable generation; the system takes control parameters like sentiment, tense *etc.*, and generates random sentences conforming to the controls. However, this system, unlike ours, does not transform a given input text.

The work that is most relevant to ours is by (Mueller, Gifford, and Jaakkola 2017) which jointly trains a VAE and an outcome prediction module to correct an input such that the output has a higher expected outcome. We use this work for comparison by configuring their system to perform formal style transformation. This model, however, does not take as input an external control parameter which ours does.

Some other relevant works are on sentiment and attribute based unsupervised style transfer (Ficler and Goldberg 2017; Shen et al. 2017; 2017), semi-supervised transfer through back translation using a translation corpora (Prabhumoye et al. 2018), formal-informal text classification using linguistic feature (Sheika and Inkpen 2012), politeness analysis (Danescu-Niculescu-Mizil et al. 2013), polite-conversation generation (Niu and Bansal 2018) using encoder-decoder models, but these do not perform **controllable** text-transformation. Similarly other relevant generation frameworks for formal-text generation by Sheikha and Inkpen (2011) and paraphrase generation (Wubben, Van Den Bosch, and Krahmer 2010; Prakash et al. 2016) are either template based or supervised, and are not controllable. Language generation systems such as Li et al.; Yu et al. (2017; 2017) which incorporate NLP based scorers are unsupervised but suffer from convergence problems while training, as also pointed out by Hu et al. (2017). To the best of our knowledge, our work is the first to approach the task of *unsupervised controllable text transformation*.

3 System Overview

Our framework is designed to take text (or sentence, at this moment) and a set of control parameters. At this moment we consider only one input control *i.e.*, degree of formalization. For such a task, a neural encoder-decoder is a natural choice. In our setting, the encoder-decoder takes a control value apart from the input text. To train this module without supervision, additional components are employed. Figure 2 depicts an overview of the system and the learning scheme.

3.1 Encoder and Controllable-Decoder

Our framework builds on the encoder-attend-decoder architecture of (Bahdanau, Cho, and Bengio 2014). The working principle of this is well-known, and is thus skipped for brevity. A vital change that we apply to the traditional architecture is the inclusion of an additional input to the decoder - the control input. The control input is passed through an embedding layer of d dimensions.

Given an input sentence $\boldsymbol{X}$ comprising N words, and a d dimensional control vector $\boldsymbol{c}$, the encoder first produces a hidden representation $\boldsymbol{H}$, using stacked layers of GRUs (Cho et al. 2014). The decoder generates one word at each time step, by: (a) *attending* to the hidden representation building a context vector, (b) concatenating the context vector with the control vector and the current hidden state of the decoder, and (c) producing a conditional probability distribution from the resultant output through a non-linear function. To summarize, for the M word output sentence $\boldsymbol{Y} = (\boldsymbol{y_1}, \boldsymbol{y_2}, \cdots, \boldsymbol{y_M})$, the conditional probability term for each word y_i can be computed as,

$$\begin{aligned} p(\boldsymbol{y_i}|\{\boldsymbol{y_1}, \boldsymbol{y_2}, \cdots, \boldsymbol{y_{i-1}}\}, \boldsymbol{z_i}, \boldsymbol{c}) \\ = g(\boldsymbol{E y_{i-1}}, \boldsymbol{c}, \boldsymbol{s_i}, \boldsymbol{z_i}) \end{aligned} \tag{1}$$

where g is the `softmax` function that outputs the probability distribution of $\boldsymbol{y_i}$; $\boldsymbol{E y_{i-1}}$ is the embedding of the previously generated word; $\boldsymbol{c}$ is the control vector; $\boldsymbol{z_i}$ is the vector obtained by attending over $\boldsymbol{H}$; $\boldsymbol{s_i}$ is the hidden state of the decoder GRU at time-step i.

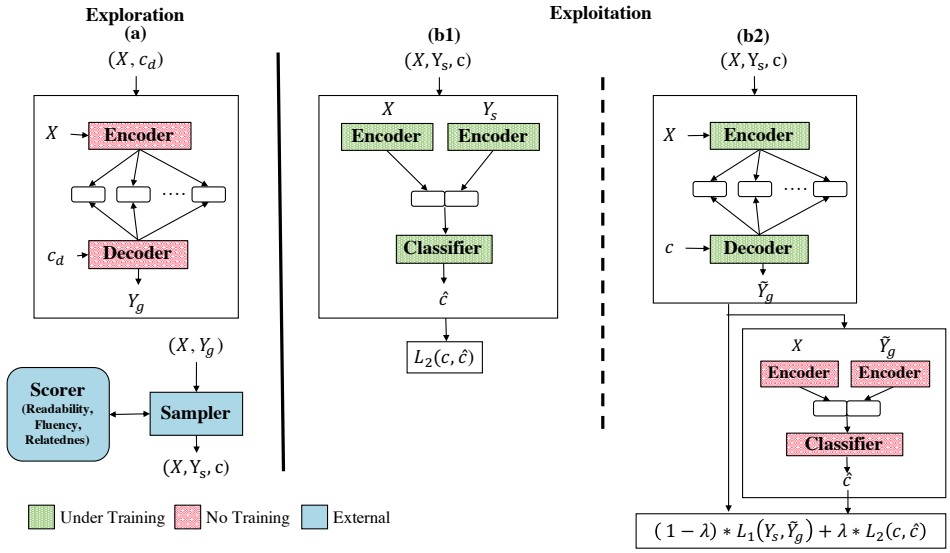

Figure 2: System architecture and phases of training

3.2 Sampler and Scorer

To train the encoder-decoder framework in an unsupervised manner (*i.e.* without true labels $\boldsymbol{Y}$ being available), instead of taking the most likely output of the decoder, the output probability distribution is passed through a *sampler* that samples K different samples from the decoder. Sampling is necessary for the **exploration** step in the training process (discussed later in Section 4.2). Given an input text $\boldsymbol{X}$, the decoder can produce output $\boldsymbol{Y_g}$. The role of the sampler is to take $\boldsymbol{Y_g}$ and produce K variants of $\boldsymbol{Y_g}$ by applying slight variations. Once K samples are obtained, the best sample ($\boldsymbol{Y_s}$) that maximizes the weighted score (produced by the scorer module), is retained for further processing.

$$\boldsymbol{Y_s} = \underset{\boldsymbol{Y}}{\operatorname{argmax}}\{G(\boldsymbol{X}, \boldsymbol{Y}) | \boldsymbol{Y} \in \{\boldsymbol{Y_g}, Sample_K(\boldsymbol{Y_g})\}\} \tag{2}$$

where $G(.)$ is the scorer function and $Sample_K(.)$ is the sampler function that produces K outputs.

The scorer ($G(.)$) comprises of multiple individual scoring modules that decide various aspects of the generated text. In our setting, the scorer intends to measure: (a) to what extent the generated text is semantically related to the input, (b) how fluent the generated text is, and, (c) how formal the text is, in terms of readability grade. These aspects are measured as follows:

Semantic Relatedness: Semantic relatedness between the source ($\boldsymbol{X}$) and generated/sampled text ($\boldsymbol{Y}$) (denoted as r_s) indicate how well the meaning is preserved during transformation.

$$r_s(\boldsymbol{X}, \boldsymbol{Y}) = docsim(\boldsymbol{X}, \boldsymbol{Y})$$

where $docsim(.)$ can be any document similarity measure. We use a simplistic embedding based similarity measure where a cosine similarity between averaged embedding vectors of $\boldsymbol{X}$ and $\boldsymbol{Y}$ is considered as similarity[1].

Fluency: Fluency (denoted as r_f) or how structurally well constructed is the target text ($\boldsymbol{Y}$), is measured by using a language model.

$$r_f(\boldsymbol{Y}) = p(\boldsymbol{y_1}, \boldsymbol{y_2}, \cdots, \boldsymbol{y_M})$$

which can be broken down into a product of conditional probability terms. For estimating the conditional probabilities, N-gram (Brown et al. 1992) or neural language models (Bengio et al. 2003) can be used. For our experiments, we trained a 4-gram back-off model with KenLM (Heafield 2011) and Europarl *monolingual* English corpus (Koehn 2005), which contains mixed domain texts.

Readability Grade: Readability grade score (denoted as r_d) indicates the grade level that must be acquired to understand a given text. Typically a higher grade is indicative of higher linguistic complexity. Out of several metrics for readability grade scoring available, we chose the *Flesch-Kincaid* readability index (Kincaid et al. 1975), a simple yet popular readability metric. The metric relies on number of words in a sentence, and average number of syllables per word; it thus assesses the *lexical* complexity of the text.

The cumulative scoring function $G(.)$ is a linear combination of the individual scores[2] discussed above.

$$G(\boldsymbol{X}, \boldsymbol{Y}) = \beta_s r_s(\boldsymbol{X}, \boldsymbol{Y}) + \beta_f r_f(\boldsymbol{Y}) + \beta_d r_d(\boldsymbol{Y}) \tag{3}$$

where weights β_s, β_f, β_d are empirically decided.

3.3 Sampling Strategy:

At each time step i during generation, instead of choosing the most likely word from $\boldsymbol{y_i}$, one of its synonyms is randomly picked. The synonym of the word is decided by looking up a synonym list[3], and picking one synonym randomly with a uniform probability of

[1] We use Spacy's (spacy.io) document similarity function. Similarity between texts are computed after removing stopwords.

[2] scores normalized between [0,1].

[3] We extract synonym lexicon using English WordNet.

$1/total\ number\ of\ synonyms$. A sentence is formed by concatenating the words thus chosen. Since large number of combinations of synonyms are possible, one iteration of sampling is stopped when K sample sentences are extracted.

Note that our sampler is highly lexicalized, *i.e.,* it generates only lexical variations. The reason behind such as choice is two-fold: (a) majority of the existing scorers for readability and document similarity assessment emphasize on the lexical aspects, and (b) generic/data-independent paraphrasers that transform text at syntax, semantic levels are elusive. However, the sampler plays an external role and our system can be bolstered with better samplers as the state-of-the-art progresses. For instance, for obtaining more diverse samples, conditional VAEs (Sohn, Lee, and Yan 2015), trained on unlabeled texts can be considered.

3.4 Control Determination

The sampler and scorer result in generation of new example output for $(\boldsymbol{X}, \boldsymbol{Y_s})$. However, for training the encoder-decoder (see *Exploitation*, in Section 4.3), the system needs data in the form of $(\boldsymbol{X}, \boldsymbol{Y_s}, c)$. The control value $(\boldsymbol{c})$ for the newly generated example is determined as follows.

$$c = \begin{cases} 1, \text{ if } c_r < \zeta_1 \\ 2, \text{ if } \zeta_1 < c_r < \zeta_2 \\ 3, \text{ if } c_r > \zeta_2, \end{cases} \tag{4}$$

where $c_r = r_d(Y_s)/r_d(X)$.

In our setup we allow only one control that is based on readability. For other tasks and multiple controls, appropriate criteria can be defined within the same framework.

3.5 Control Predictor

The encoder-decoder during training is coupled with a classification module or *Control Predictor*. The role of the control predictor is to predict the expected control value of any input output pair $(\boldsymbol{X}, \boldsymbol{Y})$. The control predictor constitutes a pair of encoders on top of embedding layers that get hidden representations $\boldsymbol{h_x}$, $\boldsymbol{h_y}$ from both $\boldsymbol{X}$ and $\boldsymbol{Y}$ respectively. The hidden representations are then concatenated $\boldsymbol{h} = [\boldsymbol{h_x}; \boldsymbol{h_y}]$ and passed to a fully connected neural network with `ReLU` activation to get an intermediate hidden representation $\boldsymbol{h_f}$. To obtain a probability vector we perform a linear transform over $\boldsymbol{h_f}$ and normalize it using softmax, which is given as:

$$\boldsymbol{h_f} = \texttt{relu}\left(\boldsymbol{W_h h}\right) \tag{5}$$

$$\boldsymbol{h_c} = \texttt{softmax}\left(\boldsymbol{W_c h_f}\right) \tag{6}$$

During training, the control predictor takes $< \boldsymbol{X}, \boldsymbol{Y_s}, \boldsymbol{c} >$ instances generated by the sampler and scorer module. It predicts control $\hat{\boldsymbol{c}}$, suffers a *cross-entropy* based loss $\mathcal{L}_2(\boldsymbol{c}, \hat{\boldsymbol{c}})$. The losses are back-propagated to the embedding layers. Section 4.4 motivates the need of this module.

4 Training Objective and Process

Training is carried out in three phases: (1) *pre-training* (2) *exploration* and (3) *exploitation*. While phase (1) is carried out once, the system undergoes phases (2) and (3) for several iterations. Figure 2 gives an overview of the training process.

4.1 Pre-training

The encoder-decoder is first pretrained as an autoencoder with unlabeled data, it learns to predict X' from X such that $X' \sim X$. Since, no control input is available during this phase, the control input is neglected. Pretraining ensures better initialization of the encoder and decoder parameters.

4.2 Exploration

Since labeled data is not directly available for training, the exploration phase helps synthesize instances of input text, output text and appropriate control ($< \boldsymbol{X}, \boldsymbol{Y}, \boldsymbol{c} >$). The system is not trained during the process and is only allowed to predict output $\boldsymbol{Y_g}$ and "explore" for other samples that are variations of $\boldsymbol{Y_g}$. With the help of the sampler, the system generates K sample outputs and the scorer selects the sample $\boldsymbol{Y_s}$ that maximizes the score. If none of the samples obtain a better score than the default output $\boldsymbol{Y_g}$, no new instance is augmented for that particular X in that particular iteration of exploration. If the generated output is same as the input, that instance is discarded. The possible control value for the generated sample $\boldsymbol{Y_s}$ is computed following Equation 4.

As the encoder-decoder is pretrained, in the first iteration of exploration, the model predicts $\boldsymbol{Y_g}$ same as input $\boldsymbol{X}$. Since the sampled sentences selected by the scorer will always have better score than $\boldsymbol{Y_g}$ (else it will not be selected), the first iteration of exploration ensures that the output side of the synthesized data is different and with a better cumulative score than the input.

4.3 Exploitation

In this phase, the encoder-decoder are trained using the data generated during the exploration phase. This process is carried out in two stages as mentioned below:

Training of Control Predictor: The Control Predictor Module is trained with $\boldsymbol{X}, \boldsymbol{Y}$ taken as input and $\boldsymbol{c}$ as output. It undergoes training in a standard classification setup where batches of labeled data are fed in multiple iterations and the loss is minimized. Once training gets over, the predictor is plugged into the encoder-decoder network (where it becomes non-trainable) to predict the control for the generated sentences. Both $\boldsymbol{X}, \boldsymbol{Y}$ are provided as input since control values signify the **relative** variation of $\boldsymbol{Y}$ w.r.t. $\boldsymbol{X}$.

Training of Encoder-Decoder: The encoder-decoder framework trains with source $\boldsymbol{X}$, control $\boldsymbol{c}$ as input and target output $\boldsymbol{Y_s}$. For each instance, the model predicts output $\boldsymbol{Y_g}$. The control predictor then has to predict the control category of the output ($\boldsymbol{Y_g}$). However, obtaining $\boldsymbol{Y_g}$ involves finding out the most likely words in the conditional probability distribution (Equation 1) using `argmax` operation. This makes the overall loss (Equation 7) non-differentiable.

To avoid this, we approximate Y_g as follows:

$$Y_g \approx \tilde{Y_g} = \{\tilde{y}_1, \tilde{y}_2, ..., \tilde{y}_M\}$$

where,

$$\tilde{y}_i \sim \texttt{softmax}(s_i/\tau)$$

where, s_i is the unnormalized decoder hidden representation (logits) and $\tau > 0$ is the *temperature*. Setting the temperature to close to zero approximates the output to one-hot representation, typically obtained through `argmax`. This, however, ensures differentiability unlike `argmax`.

The modified output $\tilde{\boldsymbol{Y}}_{\boldsymbol{g}}$ is input to the Control Predictor which predicts the appropriate control $\hat{\boldsymbol{c}}$ pertaining to $\boldsymbol{X}$ and $\boldsymbol{Y}_{\boldsymbol{g}}$. Based on the predicted output and control, a composite loss is calculated, as follows.

$$\begin{aligned}\mathcal{L}(\boldsymbol{Y}_{\boldsymbol{s}}, \boldsymbol{Y}_{\boldsymbol{g}}, \boldsymbol{c}, \hat{\boldsymbol{c}}) = \lambda \times \mathcal{L}_2(\boldsymbol{c}, \hat{\boldsymbol{c}}) \\ +(1-\lambda) \times \mathcal{L}_1(\boldsymbol{Y}_{\boldsymbol{s}}, \tilde{\boldsymbol{Y}}_{\boldsymbol{g}})\end{aligned} \tag{7}$$

where $\lambda \in [0, 1]$ is a hyper parameter and the functions $\mathcal{L}_1(.)$ and $\mathcal{L}_2(.)$ are *cross-entropy* based loss functions, popularly used in sequence-to-sequence and classification tasks.

4.4 Why Control Predictor?

In our setting for transformation, token level entropy faces the following issues: (i) Token-level loss averaged over the generated words equally penalizes all words; but in controllable transformation some words (for instance *content* words) play a more important role than others (*e.g. stop-words*), (ii) Cross entropy alone does not adequately capture how the generated sentences are related to the controls passed. To alleviate these, we employ a predictor that computes an additional loss between predicted controls of the generated sentence and the reference controls.

5 Experiment Setup

We now describe the dataset, architectural choices, and hyperparameters selected. Our dataset contains **unlabeled** text which are simple and informal in nature. It comprises of 14432 informally written sentences collected from *Enron Email Corpus*[4], *Corpus of Late Modern English Prose*[5], *non-spam emails from Spam Dataset*[6], and *essays for kids*[7]. The data is split into a train:valid:test split of $80\% : 12\% : 8\%$. The vocabulary size of the dataset is 95775 words. The source code and dataset are available at https://github.com/parajain/uctf.

We used a bidirectional GRU based encoder with 2 layers; the forward and backward hidden representations from the encoder are concatenated. The embedding matrix with a word embedding size of 300 was shared between the encoder and decoder. For encoder, 2 layers of GRUs were used and the hidden dimension size was set to 250. The decoder had 2 layers with hidden dimension size set to 500 . The weights related to language model, sentence similarity and readability scorers, given in Equation 3 are empirically set to 0.6, 0.2 and 0.2 respectively. Note that, Equation 3, by design, allows choosing different combinations of transformation parameters. These parameters may be tuned differently for different practical settings. For example, a low weight on semantic relatedness will be less restrictive on "relatedness" and can generate text with higher readability grade which is suitable for a domain such as creative writing. On the other hand for a domain such as legal where relatedness is more important, the corresponding weight can be set to a higher value.

During loss computation (Equation 7) we tried different values of parameter $\lambda \in [0.1, 0.3, 0.5]$ and settled with 0.1. The ζ_1 and ζ_2 in equation 4 were set to 1.05 and 1.1 respectively; these threshold values were incremented by 5% for each category to suit practical purposes. The temperature parameter τ was set to 0.001; we did not use annealing. For both encoder-decoder and control predictor, we employed the Adam optimizer with the learning rate set to 0.001. The system underwent pretraining once, followed by a *maximum* of 20 cycles of exploration-exploitation. For each input, we chose max $K = 100$ samples. Exploitation occurs for 20 epochs with early stopping enabled. After each cycle, the model was saved and we chose the model which provided the best average score over a held-out set prepared only for model selection. During testing, for each input sentence in the test-data, control level values of 2 and 3 were chosen.

Test scenario: The present system is configured to take an input sentence and **three** possible control levels pertaining to the degree of formalization desired by the user. The control levels can be 1, 2 and 3. As depicted in Figure 1 in the introductory section, an input control of 1 would retain the input sentence as is, control 2 would transform it to a *relatively* medium level formal text (termed as *Formalness-Mid*) and control of 3 will transform the input to a text that is even more formal that 2, (termed as *Formalness-High*).

5.1 Research Questions Addressed

Our experiments are designed to address the following research questions

- **RQ1:** Is our method able to produce output that is formal, fluent and adequate? Is the transformation controllable? How well does the output's formalness conform to the input control provided?
- **RQ2:** Does the control predictor have a positive impact on the overall performance of our model?
- **RQ3:** Is iterative exploration-exploitation helpful? Why wouldn't a much simpler strategy of just one round of sampling and training suffice?
- **RQ4:** Does our approach outperform the existing supervised and unsupervised methods for such tasks?

RQ1 is answered by measuring fluency, semantic relatedness and readability of the output text following Section 3.2. To answer **RQ2** and **RQ3**, we prepare two trivial variants of system: (a) CTRLNOPREDICTOR, in which the control predictor module is removed, and (b) CTRLONESHOT where exploration and exploitation cycles are not iteratively carried out. Rather, the system undergoes one round of exploration cycles followed by one exploitation cycle.

For **RQ4**, since the existing approaches/systems have no provisions to take control parameters like ours, comparison is only done based on their transformation capability.

[4]http://bailando.sims.berkeley.edu/enron_email.html

[5]http://ota.ox.ac.uk/desc/2077

[6]http://csmining.org/index.php/spam-assassin-datasets.html

[7]http://www.mykidsway.com/essays/

Mode	CTRL WITHPREDICTOR		CTRL NOPREDICTOR		CTRL ONESHOT		Mueller et al., 2017
Formalness Control	Mid	High	Mid	High	Mid	High	NONE
Readability	0.568	0.583	0.538	0.538	0.554	0.554	0.33
Relatedness	0.72	0.74	0.77	0.77	0.78	0.78	0.05
LM Score	0.34	0.34	0.32	0.32	0.30	0.30	0.16

Table 1: Average test-set scores (normalized between $[0-1]$)

System	BLEU	Relatedness (with input)	Readability
Mueller et al.	5.09	0.41	0.38
Seq2Seq (skyline)	38.37	0.17	0.71
Formalness-Mid (Ours)	21.81	0.58	0.52
Formalness-High (Ours)	21.14	0.57	0.74

Table 2: Comparison of BLEU (%), avg. Semantic Relatedness with input, and avg. normalized readability scores. **The avg. readability for input and reference sentences in test data are (0.41) and (0.50) respectively**

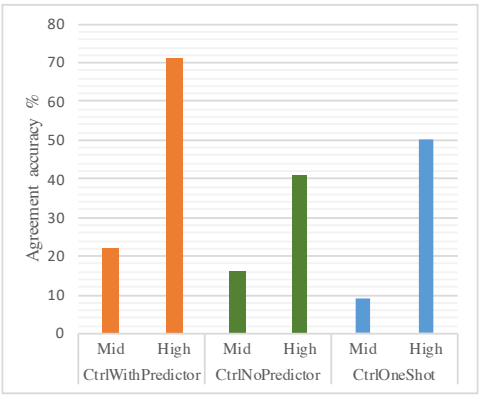

Figure 3: Accuracy (in %) showing the agreement between desired input control and control measured on the output text using Equation 4

- **Comparison with Existing Unsupervised Method:** The most relevant unsupervised transformation system is by Mueller, Gifford, and Jaakkola (2017). This system takes a sequence as input, auto-encodes it and applies minor variation based on expected outcome that is decided by a scorer. We fix readability metric as the scoring function and treat the system as a baseline.
- **Indirect Comparison with Supervised Systems** A similar task to ours is the reverse of text simplification, for which labeled datasets exist. We chose the dataset configuration given in the neural-text simplification work by Nisioi et al. (2017), originally derived from Hwang et al. (2015). We flip the input-output side of the parallel corpus, thereby reversing the training objective. Our system uses only the simplified side of the data for training, with the previously discussed configuration. Additionally, the complete *simple-to-complex* parallel corpus was used to train a neural Seq2Seq generation system (Bahdanau, Cho, and Bengio 2014) using the Marian toolkit (Junczys-Dowmunt et al. 2018) with default configurations[8].

6 Evaluation Results

For our dataset, we first evaluate the output text based on: (a) basic parameters such as fluency, adequacy (semantic relatedness with input) and readability grade, and (b) whether the quality of output comply with the input control parameters or not. A direct comparison between different variants of our own system could be done using these measures.

Observe in Table 1, the readability scores for CTRLWITHPREDICTOR are always better than the average readability grade of the input (**0.54**). This indicates that the transformation obtained by the system has an improved readability grade. The improvement is even more for CTRLWITHPREDICTOR than the other two, which demnostrates the importance of the Control Predictor and the iterative training scheme. The Language Model scores CTRLWITHPREDICTOR is the highest and differs significantly from CTRLNOPREDICTOR. This shows that adding the auxiliary loss from Control Predictor output results in better fluency. Regarding semantic relatedness, we observe that most sentences generated by CTRLONESHOT undergo little or no modification *vis-á-vis* the input. This results in a slightly higher semantic relatedness score. Moreover, in most cases the output obtained from these two systems are indifferent to control values of Mid and $High$, hence the average scores are same for both control levels. For CTRLONESHOT, for the same input text, multiple variants for the same input are generated in one-shot. In the hindsight, using such data may have confused the system during training and it may have learned to ignore the control. Lastly, CTRLNOPREDICTOR hardly incurs any direct loss pertaining to the control values, and thus, becomes somewhat agnostic of the control levels.

Figure 3 indicates how robustly the systems respond to the input control. The agreement accuracy in the figure refers to the percent of test-instances for which the output control (measured using Equation 4) match the intended input control. For all the three systems, for control value Mid, most of the generated text actually are either confused with

[8]convergence attained after 70000 iterations.

Mode	Input Sentence	*Formalness Control-Mid*	*Formalness Control-High*
WithPred	(1) 18 year old who abandoned her child in a hospital later got custody (2) the first sync after upgrading will be slow	(1) 18 year old who unpopulated her kid in a infirmary resultant got custody (2) the first synchronise afterward upgrading will be idle	(1) 18 year old who deserted her tyke in a infirmary resultant got detention (2) the first synchronise afterward upgrading will be laggard
NoPred	(1) 18 year old who abandoned her child in a hospital later got custody (2) the first sync after upgrading will be slow	(1) 18 class old who deserted her child in a infirmary accompanying got detention (2) the introductory synchronise afterward upgrading will be goosy	(1) 18 class old who deserted her child in a infirmary accompanying got detention (2) the introductory synchronise afterward upgrading will be goosy
OneShot	(1) 18 year old who abandoned her child in a hospital later got custody (2) the first sync after upgrading will be slow	(1) 18 yr old who untenanted her tyke in a hospital subsequently got detention (2) the eightieth sync later upgrading bequeath be tedious	(1) 18 class old who deserted her tyke in a hospital subsequently got detention (2) the eightieth sync later upgrading bequeath be tedious

Table 3: Example input and transformed sentences with varying control values and system variants

control value $Default$ (*i.e.*,very similar to input) or control value $High$. This suggests the need for fine-tuning the user-defined parameters in Equation 4. However, note that the highest percentage values are obtained for CTRLWITHPREDICTOR.

6.1 Human Evaluation

We performed an additional experiment where three language experts unaware of the task were given 30 random instances from our test dataset. Each instance consists of the input sentence, the predicted outputs with controls Mid and $High$ (shuffled for ruling out bias). For each instance the experts were asked to rank the output sentences based on their perceived readability. Ideally, the output for Mid should rank lower than the one for $High$. The average agreement between the human-rated rank labels and the default ranking based on input control categories turned out to be **80.2%**, This indicates that the readability of output $High$ is indeed higher than that of Mid.

Table 3 presents a few randomly selected examples for different input sentences and varied control values for all 3 system variants. We can clearly observe that the input sentences are modified according to control values. CTRLWITHPREDICTOR shows increasing readability grade going from Mid to $High$. This is also corroborated by readability scores in Table 1. Further, as expected, the CTRLNOPREDICTOR system is unable to capture the change in control levels due to lack of explicit feedback. The CTRLONESHOT system also does not show much improvement with controls due to lack of transformation knowledge it would have gained if the iterative exploration/exploitation based training had been applied. We believe, our observations above provide positive answers to research questions **RQ1**, **RQ2**, and **RQ3**.

6.2 Comparison with Supervised Systems for Reversed-simplification Task

Table 2 shows how comparable are the generated complex sentences with the reference complex texts for reverse simplification task. For this, BLEU (Papineni et al. 2002) is considered as the metric. Supervised Seq2Seq, as expected serves as a *skyline* and achieves a much better BLEU than ours. Yet the BLEU score of our system is significant, considering that our system is unsupervised and not designed to produce outputs that overlap more with the reference text. It is important to note that the average readability scores for our system variants are better than that of the reference corpora (avg. readability 0.50) and Seq2Seq, which shows the capability of our system in transforming the input to a more formal version. Moreover, the higher semantic relatedness scores indicate that our models are indeed capable of preserving the semantics better.

Regarding comparison with Mueller et al., the crux of their system is a variational autoencoder and an outcome prediction module which revises the input such that the output has a higher expected outcome. The problem with such an approach is that it is restricted to generating from a known distribution of sentences provided in the form of training data. The reduced performance by the system as seen in Tables 1 and 2 is perhaps due to the facts that the training data size is less and the system does not have schemes for iterative training like ours. The above comparison studies provide insights for **RQ4**.

7 Conclusion and Future Work

We proposed a novel NLG framework for unsupervised controllable text transformation that relies on *off-the-shelf* language processing systems for acquiring transformation

knowledge. Our system is tested for the task of formalizing the input text by improving its readability grade, where the degree of readability grade is controllable. Experiments on general domain datasets demonstrate the goodness of the output transformed texts both quantitatively and qualitatively. The system also learns to give importance to the user-defined control and responds accordingly. A shortcoming of the current approach is that it only performs lexical formalization because of the sampling strategy and choice of Flesch Kincaid readability metric that is highly lexical. Our future agenda involves exploring better sampling techniques for generating complex structural and semantic variants, and also testing the framework for tasks like text simplification and style-transformation.

References

Artetxe, M.; Labaka, G.; Agirre, E.; and Cho, K. 2017. Unsupervised neural machine translation. *arXiv:1710.11041*.

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

Bengio, Y.; Ducharme, R.; Vincent, P.; and Jauvin, C. 2003. A neural probabilistic language model. *JMLR, 2003* 3(Feb):1137–1155.

Brown, P. F.; Desouza, P. V.; Mercer, R. L.; Pietra, V. J. D.; and Lai, J. C. 1992. Class-based n-gram models of natural language. *Computational linguistics*.

Cho, K.; Van Merriënboer, B.; Bahdanau, D.; and Bengio, Y. 2014. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv:1409.1259*.

Danescu-Niculescu-Mizil, C.; Sudhof, M.; Jurafsky, D.; Leskovec, J.; and Potts, C. 2013. A computational approach to politeness with application to social factors. In *ACL, 2013*.

Ficler, J., and Goldberg, Y. 2017. Controlling linguistic style aspects in neural language generation. *arXiv:1707.02633*.

Gatt, A., and Reiter, E. 2009. Simplenlg: A realisation engine for practical applications. In *12th European Workshop on NLG*.

Heafield, K. 2011. Kenlm: Faster and smaller language model queries. In *Sixth Workshop on SMT*.

Hu, Z.; Yang, Z.; Liang, X.; Salakhutdinov, R.; and Xing, E. P. 2017. Toward controlled generation of text. In *International Conference on Machine Learning*.

Hwang, W.; Hajishirzi, H.; Ostendorf, M.; and Wu, W. 2015. Aligning sentences from standard wikipedia to simple wikipedia. In *NAACL-HLT*.

Jain, P.; Laha, A.; Sankaranarayanan, K.; Nema, P.; Khapra, M. M.; and Shetty, S. 2018. A mixed hierarchical attention based encoder-decoder approach for standard table summarization. In *NAACL-HLT*.

Junczys-Dowmunt, M.; Grundkiewicz, R.; Dwojak, T.; Hoang, H.; Heafield, K.; Neckermann, T.; Seide, F.; Germann, U.; Aji, A. F.; Bogoychev, N.; Martins, A. F. T.; and Birch, A. 2018. Marian: Fast neural machine translation in c++. *arXiv:1804.00344*.

Kincaid, J. P.; Fishburne Jr, R. P.; Rogers, R. L.; and Chissom, B. S. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel. Technical report, DTIC Document.

Koehn, P. 2005. Europarl: A parallel corpus for statistical machine translation. In *MT summit*.

Koehn, P. 2009. *Statistical machine translation*. Cambridge University Press.

Lample, G.; Denoyer, L.; and Ranzato, M. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv:1711.00043*.

Li, Z.; Jiang, X.; Shang, L.; and Li, H. 2017. Paraphrase generation with deep reinforcement learning. *arXiv:1711.00279*.

Mueller, J.; Gifford, D.; and Jaakkola, T. 2017. Sequence to better sequence: continuous revision of combinatorial structures. In *ICML, 2017*, 2536–2544.

Nisioi, S.; Štajner, S.; Ponzetto, S. P.; and Dinu, L. P. 2017. Exploring neural text simplification models. In *ACL, 2017*.

Niu, T., and Bansal, M. 2018. Polite dialogue generation without parallel data. *arXiv:1805.03162*.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL, 2002*.

Prabhumoye, S.; Tsvetkov, Y.; Salakhutdinov, R.; and Black, A. W. 2018. Style transfer through back-translation. *arXiv:1804.09000*.

Prakash, A.; Hasan, S. A.; Lee, K.; Datla, V.; Qadir, A.; Liu, J.; and Farri, O. 2016. Neural paraphrase generation with stacked residual lstm networks. *arXiv:1610.03098*.

Sheika, F. A., and Inkpen, D. 2012. Learning to classify documents according to formal and informal style. *Linguistic Issues in Language Technology*.

Sheikha, F. A., and Inkpen, D. 2011. Generation of formal and informal sentences. In *13th European Workshop on NLG*.

Shen, T.; Lei, T.; Barzilay, R.; and Jaakkola, T. 2017. Style transfer from non-parallel text by cross-alignment. In *NIPS, 2017*, 6830–6841.

Sohn, K.; Lee, H.; and Yan, X. 2015. Learning structured output representation using deep conditional generative models. In *NIPS*.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *NIPS*.

Wubben, S.; Van Den Bosch, A.; and Krahmer, E. 2010. Paraphrase generation as monolingual translation: Data and evaluation. In *INLG, 2010*.

Yu, L.; Zhang, W.; Wang, J.; and Yu, Y. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI, 2017*.