Literary Style Transfer with Sequence Models and Rejection-Pass Filtering

Daniel Kharitonov
Stanford, CA 94305
dkh@cs.stanford.edu

Yu Fan
Stanford, CA 94305
yuf@stanford.edu

Guneet Kohli
Seattle, WA 98304
gkohli@stanford.edu

Abstract

In this paper, we consider the problem of unsupervised literary writing style transfer in the natural language. We develop a rejection-pass filter architecture for reconstructing the parallel representation of input text using sequence-to-sequence deep network generators, and demonstrate the use of the three-dimensional metric for testing the quality of transfers and searching the hyper-parameter space.

1 Introduction to literary style transfer

The problem of general style transfer has been extensively researched in the last ten years and led to impressive results in applying stylistic cues to still and moving images where content (contours and shapes) is easily separable from style (colors, strokes and spatial transformations). Natural language does not exhibit such separability, and therefore progress in text style transfer was hitherto limited to domains where the parallel text representations already existed.

One example of an environment with parallel representation is translation to foreign languages, where extensive corpus of references is made possible by the work of human interpreters [1]. Another example is phrase formality, where a training set encodes the formal/informal sentence pairs [2]. A third example is modern $\leftrightarrow$ archaic text adaptation, where were the stylized output can be inferred from the specialized dictionaries [3].

Literary style transfer in natural language without the parallel representations, however, remains an unsolved problem. Most researchers working in this field currently concern themselves with narrow tasks – such as sentiment shifts [4], transformation of hand-coded attributes (gender, political affiliation and image captions) [5], or word substitution deciphering [6].

In this paper we permit ourselves to formulate a larger goal and study the effect of literary style transfer in arbitrary texts – that is, the transformation of content from author A into a writing style of author B. We demonstrate possibility of such transfer using state of the art sequence-to-sequence generators constrained for specific outputs, and establish a baseline for further efforts in this area.

1.1 Literary Style

In this paper we will be focusing mostly on examples drawn from fiction, so we need to define the literary style (further called "style") more formally. For purposes of our study, we shall consider "style" to be the same as the "writing style", with a colloquial definition of "being the essential elements of spelling, grammar, and punctuation, including the choice of words, sentence structure, and paragraph structure". It is important to note that while this definition is rooted in personality and language choices of a human writer, for as long as they can be discerned by an expert. In other words, we generally cannot define a literary style without an author, although a "meta-style" can be sometimes defined by averaging the canons of a specific sub-genre.

With that definition, we are speaking of "style transfer" as a process of shifting the properties of some input text (content donor) towards a target style while preserving the content. The "golden standard" for such a shift would be an ability for a snippet of transformed text to be recognized as belonging to the corpus of the style author, while still communicating intentions set forth by the content writer. Note, that this definition excludes long-term features (like story arc), and personal and proper names; in general, we do not expect or require the content input to match style donor in terms of character naming, use of geographical labels, or plot complexity.

1.2 Content

Style transfer involves grafting the writing style from one source onto the content (story) from another. Therefore, a style transfer application features two inputs and one output, and is expected to produce the text that remains clear and approachable for an average human. It is important to observe that the same general content can be expressed with a large variety of linguistic devices (modifiers, metaphors, sentence formations and so on), so the total number of admissible style modifications can be very large. We will refer to texts conveying similar information as "parallel representations", expecting such representations to share the core message, but differ in their expressive ways.

1.3 Evaluation

Since our intention is to provide machine-based style transfer, we need to task ourselves with subject of output evaluation. It follows from our discussion on styling that such score should include the measures of a triplet {style source, narrative fluency, and content equivalence}. Provided that our ultimate goal is a perfect imitation of source style conditioned on story from content source, missing either of the aforementioned factors will not yield a satisfactory result.

For one example, if the output text does not employ the vocabulary and sentence structure of style donor, it will result in the stylistic miss. For another example, if the output employs the style but departs from the content, it will fail to form a parallel representation. For a third example, if the output text successfully fuses the content with style of input sources but violates general language and writing norms, it will result in a poor reading experience. Therefore, to evaluate the quality of style transfer, we need to take all those considerations into account.

2 The scoring metric components

In evaluating results of the literature style transfer, we must consider that the two dimensions of the metric (naturalness and content preservation) are of the satisficing type, while the style is the metric component we optimize for. We leverage the existin research community consensus (such as shown by Mir et. al [12]), to utilize intensity, naturalness and content preservation as key aspects of interest.

We calculate text naturalness by passing transformed text through a neural network adversarially trained to recognize machine-generated text; if the detector admits style donor but rejects the synthesized text, we register a component failure, and report a success otherwise. Currently, a pre-trained LSTM classifier from is used for our purposes, and we are currently exploring training our own adversarial classifier from scratch to improve the results.

Content preservation is calculated as Word Mover's Distance (WMD) on texts with style masking (i.e. placeholders used in place of style words) calculated after training a word2vec model on the entire dataset.

Finally, style transfer intensity is calculated as the earth mover's distance between the input and output style distributions. We train a style discriminator to provide probabilities for authorship attribution; earth mover's distance between the output of the soft-max unit and the input (donor) style distribution concludes our optimizing (and final) metric component. Considering naturalness, content preservation and style transfer intensity together, we effectively have an evaluation score with two binary features and one vector of real numbers.

3 Rejection-Pass filtering for style transfer

Our architecture relies on the ability of generative models to produce samples from training dataset with style z and content c via latent generative distribution: $\hat{x} \sim G(z, c) = \prod p(x_t | x_{t-1..0}, c)$

With generative models, we can formulate the problem of literary text transformation for source s as finding such sequence x_i that embedding $e(s) = c$. More specifically, since a trained generative model already includes style z, we can control for the embedding score by means of repeated sampling admitted on the condition of satisfying some threshold (a hyper-parameter). This process of threshold-based filtering is recurring and self-adjusting: for as long as the generative model produces the output consistent with embeddings of input lexemes the samples will be admitted; a failure to match the score forces the output filter to accept a lexeme from the donor text unchanged and resets the context for generator. This way, the growing distances between the embedding vectors of content in donor text and generator output are periodically realigned.

Complete architecture of a rejection-pass filter is shown in Fig.1:

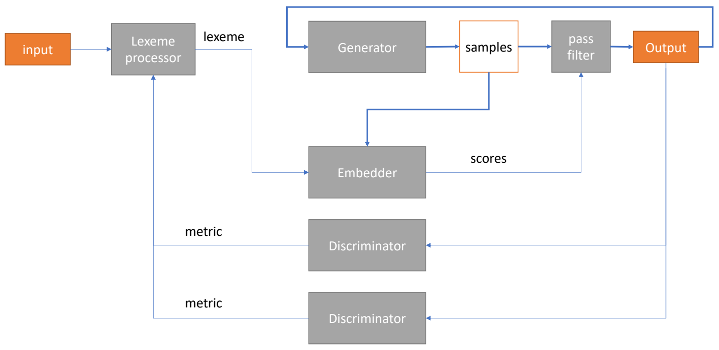

Figure 1: Reconstruction of parallel representation with phrase sampling.

The shown architecture is a variant of a constraint-based generator conditioned on the previously transformed sequence. It bootstraps from forcing the generator to produce a sample that would randomly match the embedding score of the first lexeme; the assumption is that a well-trained generator will have enough contexts to choose from. Two discriminator networks are employed to evaluate the metric and adjust system hyper-parameters.

3.1 RPF component: sequence-to-sequence generator

Our algorithm relies on using a sequence-to-sequence text generator pre-trained on the style source. We employ a Open AI GPT-2 model with 124M or 345M parameters [9].

3.2 RPF component: Lexeme processor

Lexeme processor takes input text and separates it into phrases suitable for phrase embedder. One input lexeme can result in multiple lexemes spun by the generator; such behavior permits the style transfer engine to replicate the sentence structure and punctuation of the style donor. In our model we are using a relatively simple hand-coded processor, but it can also be implemented as neural network optimizing for lexeme features.

3.3 RPS component: Embedder

Phrase embedder calculates the similarity scores for the content input lexeme and the candidate expansions (samples) produced by the generator. Encoder precision significantly depends on the length of the input lexemes; longer phrases stand are both more difficult to encode and stand lower chance to be matched by a sample from generator. We employ Google Sentence Encoder to calculate the lexeme scores.

3.4 RPS component: pass filter

Pass filter is a simple logical gate admitting the best sample that clears embedding threshold. If the sample set fails to clear the filter, the input lexeme is admitted unchanged.

3.5 RPS component: Discriminator: Naturalness

A discriminator for naturalness is a custom neural network designed to detect machine-generated text; we use it to gauge the smoothness of model output. The task of the naturalness detector is conceptually similar to functionality of "fake news" detectors [10, 11], but must take into account the fact that fiction uses significantly more language features relative to Internet news articles, and needs a different training routine.

<final architecture description here>

3.6 RPS component: Discriminator: Style

A discriminator for the style is a custom authorship attribution engine that produces soft-max probabilities for the output text to belong to various authors. The model is pre-trained using texts written by the three authors. After the pre-training, the model is then fixed and takes as its input the generated text, to evaluate if its style is transferred into a target domain. In an ideal case, the discriminator will be "fooled" by the output of the generator and thus assign a high score of the target author to it. We are optimizing our architecture to maximize the score in the direction of matching the style donor. A baseline model for the style discriminator is build based on GloVe word representations, followed by a convolutional layer and two stacked GRU layers.

<final architecture description here>

4 Dataset and model hyperparameters

To train and evaluate our initial model, we created three different style datasets compiled from texts in the Project Gutenberg online library and other sources . Specifically, we are using the corpora of texts by Alexander Dumas (8.3MB), Jane Austen (4.1MB) and Vladimir Nabokov (9.8MB).

We employed these datasets almost unchanged (text minus publishing notes and annotations) to train sequence-to-sequence style generator instances (finetune the pre-trained GPT-2 model). In addition, we sliced these texts into 600B chunks, cleaned them off proper and personal names (to regularize on style), and labeled them with author codes to train our style discriminator network. Discriminator training sets are balanced for presence of snippets by every author.

The list of the hyperparameters the RPF system is using (with optimal values) is provided below:

Rejection threshold	minwords	maxwords	seed	nsamples	temperature
0.68	2	7	10	10,000	1.0

Rejection threshold – defines the minimum semantic similarity between phrase embeddings from the content input and samples the filter could accept. Increasing this parameters would force the model to search for better matches at risk of not finding the good candidates.

Minwords, maxwords – are the hyperparameters for lexeme processor that define the minimum and maximum number words in the next phrase for processing. A lexeme preferentially ends with a natural separator (period, comma, or semicolon), but a long input phrase may come without such reasonable boundaries, and taking a very long lexeme would make matching the meaning of such phrase with a sample unlikely.

seed – is the size of the seed phrase (context cue length measured in lexemes) given to the generator. More seed lexemes will result in the deeper context and may produce better samples, but can also restrict the generator to a cue that appears too narrow.

nsamples – the number of samples the generator produces per every new context. Larger numbers allow more candidates for rejection filter to choose from, but require more processing time to generate them and calculate embedding scores for them.

temperature – the generative network hyperparameter that describes a relative likelihood of produced samples. Higher values of temperature produces less likely – i.e. "more creative" – expansions more often.

5 Code

Python notebooks: https://github.com/fy164251/CS230

RPS_AWS_v4.ipynb - generator

Style_Discriminator_Baseline.ipynb - style

Metrics.ipynb - naturalness, style transfer intensity demo

6 Experimental results

Before we move to discussing the RPF model results, it is worth touching on problems of baseline performance and starting points.

6.1 The baseline

6.2 RPF transform examples

6.3 Analysis and open problems

7 Conclusions and future work

This paper outlines an original style transfer algorithm built alongside the three-component new evaluation metric, which sets a new baseline for literary style transfer. The idea of our RPF architecture is straightforward, modular and can be readily improved once better sequence generation and embedding models become available.

Acknowledgments

The authors owe a lot of debt to many researchers contributing to computational natural language field in the last decade. This project would not be possible without Google Colab, Amazon AWS, and OpenAI GPT-2 models.

References

[1] Orch, Franz (April 28, 2006). "Statistical machine translation live". Google Research Blog. Google.

[2] Rao, S., Tetreault J. (2018) Dear Sir or Madam, May I introduce the YAFC Corpus: Corpus, Benchmarks and Metrics for Formality Style Transfer, NAACL-HLT

[3] Jhamtani H., Gangal v., Nyberg E. "Shakespearizing Modern Language Using Copy-Enriched Sequence-to-Sequence Models"

[4] John V., Mou L., Bahuleyan H., Vechtomova O. "Disentangled Representation Learning for Non-Parallel Text Style Transfer"

[5] Sudhakar A., Upadhyay B., Maheswaran A. "Transforming Delete, Retrieve, Generate Approach for Controlled Text Style Transfer"

[6] Lample G., Subramanian S., Denoyer L. "Multi-attribute Text Rewriting"

[7] Yang Z., Hu Z., Dyer C., Xing E., Berg-Kirkpatrick T. "Unsupervised Text Style Transfer using Language Models as Discriminators"

[8] Cera D., Yinfei Y., Sneg0yi K., Nan H., Nicole L., Rhomni J., Noah C., Marco G., Steve Y., Chrs T., Yun-Hsuan S., Brian S., Ray K., "Universal Sentence Encoder", Google Research 2018

[9] Open AI GPT-2. https://github.com/openai/gpt-2

[10] GROVER: A State-of-the-Art Defense against Neural Fake News. https://grover.allenai.org/

[11] Catching a Unicorn with GLTR: A tool to detect automatically generated text http://gltr.io/dist/index.html

[12] Mir R., Felbo B., Obradovich N., Rahwan I. "Evaluating Style Transfer for Text" https://arxiv.org/abs/1904.02295