

Formality Styler: NLP Pipeline for Classifying and Transferring Textual Formality

Yonglin Wang

Brandeis University

yonglinw@brandeis.edu

Abstract

Formality is an important dimension of style, which should be controlled in certain situations to achieve effective communication. In this project, we trained models for classifying and transferring textual formality. The models were then deployed to a web page application to allow potential access for non-technical users.

1 Introduction

Formality is an important dimension in online communication and should be properly controlled to achieve effective communication in certain situations. It would be helpful to build a web-based application that can both classify the formality of a piece of text as formal or informal and suggest a rewrite in the other style.

In this project, we trained a fasttext model for binary formality classification and two Transformer models for rewriting, one for each direction. Then, we deploy the model on a webpage for easy access for non-technical users.¹

2 Related Work

The corpus for this project is the Grammarly’s Yahoo Answers Formality Corpus (Rao and Tetreault, 2018). The corpus has two separate sets of data on Entertainment & Music and Family & Relationship, the two topics on Yahoo Answers that have the most informal sentences. There are a total of 209,124 training entries (104,562 sentence pairs) combining the two topics in the corpus.

The fasttext classifier is based on a shallow neural network classifier with hierarchical soft-max (Joulin et al., 2016) and the transformer model is the modified version of Transformer (Vaswani et al., 2017), using the sequence to sequence toolkit, fairseq (Ott et al., 2019).

Moreover, we also compared the performance of another more specialized style transfer system based on disentangled representation using variational autoencoder and recurrent neural network (John et al., 2018). However, a BLEU-based evaluation revealed that the system, if used as-is, does not perform well on our current task of formality transfer.

3 System Description

3.1 Training the Classifier

We trained several fastText classifier, and found that a setting of using 2-5 character ngrams and word-level bigrams yielded the best performing classifier, with an accuracy of 92% on the test set. However, the size of the model, due to the large amount of n-gram features, reached 2 GB.

In the end, we chose the more space-efficient default setting with no character ngram and word-level unigrams, which gave an accuracy of 88%, which is still acceptable. It is worth noting that the error analysis revealed that many misclassified samples are even hard to classify for humans.

3.2 Training the Rewriter

We used fairseq trained two modified transformers models, namely `transformer_iwslt_de_en`, one for each direction. The training was done on a Titan XP GPU, which took about 2 hours to converge. Note that we did not lowercase or clean the dataset, because misspelling, casing, and irregularities are all important dimensions of formality style transfer.

To see the training and discussion for the VAE rewriting model, which is a modified version of the Tensorflow-based code provided the original authors².

¹Code for this project available at: <https://github.com/yonglin-wang/formality-styler>

²Modified code available at: <https://github.com/yonglin-wang/disentagled-style-rep>

3.3 Evaluation

Table 1 shows that the transformer approach did roughly similar to the baseline, if not slightly better; transformer generates longer text than VAE; moreover, VAE only has the ones digit BLEU score, which is very low. Note that the GYAFC model was evaluated based on a subset of test set (only sentences under topic “Entertainment and Music”), and casing and tokenization method for calculating BLEU was also not specified. Its inclusion in the table is for a rough comparison only.

For a more straightforward performance comparison, Table 2 and Table 3 show some sample outputs from the two systems. We can see that the transformer model tends to handle different forms of informality well, while the VAE model generally gives incomprehensible or completely unreasonable outputs.

3.4 Assembling the Pipeline

It is also important to know that, in a real-world application, most users of this pipeline will not be fluent in the particular technology we used in this project. Therefore, to allow easier access for the users, we deployed the model to a web-based application and enabled the users to access the system from their browser.

The webpage for deploying the model was built using Flask. However, on our local machine, technical difficulties, such as fasttext being unable to install for Python and fairseq not having a Python API for generating output, introduced some complexity to the system. In the end, the fasttext classifier was dockerized and run as a web server to generate predictions, and the interaction with transformer models was realized via spawning subprocesses in Python, sending user input as standard input, and reading and processing standard output for extracting detokenized results.

A sample result page is shown in Figure 1. The user first enter the input text to be rewritten, and specify a rewrite direction. If automatic is chosen, the classifier will be used and its output will be used as the source style. The optional ASCII-folding mechanism, which we will discuss in the following subsection, can also be activated if needed. After the user click on “Rewrite!”, the input will be passed to the corresponding model, and the rewrite (and formality prediction, if automatic) will be displayed in a few seconds. In the example in Fig. 1, we can see that the model correctly predicts that

“Hello, world!” is a formal input (compared to informal languages on the internet) and suggests a rewrite that is much more informal.

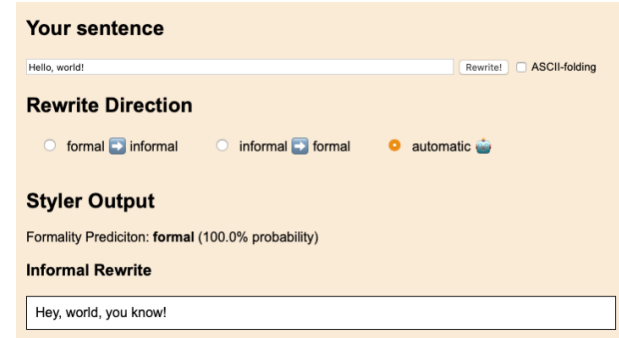


Figure 1: Result page for the input “Hello, world!” from the user.

3.5 ASCII-folding Mechanism

It is also important to know that, in a real-world application, most users of this pipeline will not be fluent in the particular technology we used in this project. Therefore, to allow easier access for the users, we deployed the model to a web-based application and enabled the users to access the system from their browser.

Our model and BPE tokenizer is train on a very clean corpus, in that the input sentences are all ASCII. This means that, even if we assume English-only input from the user, there might be words with diacritics (e.g. Zoë, façade) that will not be recognized by the tokenizer, and the non-ASCII word will be considered as an out-of-vocabulary token and converted into unknown tokens by the tokenizer. For example, “i think her name’s Zoë” will be tokenized into “i think her name ’ s Zo<unk>”, causing the rewriter model to generate unexpected output such as “I think her name is Zoboyfriend.”

Therefore, the restyler addresses this by using a technique similar to ASCII folding filter in Elasticsearch using a Python package called fold-to-ascii. To enable ASCII-folding, the user can check the “ASCII-folding” box on the Web UI; the input string from the user will be first and foremost ASCII-folded before being passed to the model (and the classifier if in automatic mode).

In this way, after ASCII-folding, “i think her name’s Zoë” is first folded into “i think her name’s Zoe”, and then fed to the model, which generates the correct out “I think her name is Zoe.”

Figure 2 shows the result of using ASCII-folding for the input “i think her name’s Zoë,” note that non-

		Informal ->Formal	Formal ->Informal
Transformer	BLEU	68.9	37.5
	BLEU (lc)	71.1	43.0
	Ratio, ref/gen	0.998	1.039
VAE	BLEU (lc)	6.7	4.8
	Ratio, ref/gen	0.834	0.952
GYAFC Best Baseline	BLEU (case-NA)	67.67	NA

Table 1: Test set evaluation metrics comparison on the transformer model, VAE model, and the best performing baseline model proposed in the GYAFC paper.

Input (informal)	Tran. Output (formal)	VAE Output (formal)	Reference (formal)
I LOOOOOVVVVVVVEEE this song SOOO Much!!!!!!	I am very fond of this song.	i love the song	I love the song so much.
I dun think he loves her	I do not think he loves her.	i believe that he is attractive	I do not think he loves her.
The sound of the 2 together, wow I do love the sound lol =)	I love the sound of the two together.	the sound of the sound of the sound of the sound of the sound of the sound	I love the sound of the two together.
I love Tyra, she has a great show, but Oprah is just great!	I love Tyra because she has a great show, but Oprah is great!	i like oprah oprah oprah oprah oprah is oprah	I love Tyra’s show, but I prefer Oprah.
How dare he call her kids - orphans.	How dare he call her kids - orphans.	how can i eat children	How dare he refer to her children as orphans?

Table 2: Test set outputs from the transformer (Tran.) and VAE models in informal to formal direction.

Input (formal)	Tran. Output (informal)	VAE Output (informal)	Reference (informal)
I am still fond of both bands today.	I still like both bands today.	i like the bands today	still like em both!
I suggest you let him know that you want to marry him.	tell him you want to marry him.	i don’t know if you want to marry him	Tell him you wanna marry him.
I love all of the songs of Green Day, and particularly Boulevard of Broken Dreams.	I love all the songs of Green Day and especially Boulevard of Broken Dreams.	i love the song i love the song i love the song i love	I love all Green Day songs especially Blvd of broken dreams
Not I; however, I was previously in orchestra.	not me but i was in orchestra.	no i like the olsen twins	Not me, I used to be in orchestra though.
How dare he call her kids - orphans.	How dare he call her kids - orphans.	how can i eat children	How dare he refer to her children as orphans?

Table 3: Test set outputs from the transformer (Tran.) and VAE models in formal to informal direction.

ASCII character in the input field has already been corrected to its folded version.

Figure 2: result after entering “i think her name’s Zoë” to the pipeline.

4 Challenges and Discussion

During exploring the output of the transformer models, we found the following patterns in the negative outputs.

First, while the model handles frequent short-hands such as omg and lol well, newer or less frequent internet acronyms or initialisms, e.g. ttyl, wdyr, cannot be translated properly by the model. This suggests that we might need a more comprehensive and up-to-date model to capture more short-hands in informal languages.

Second, the model is unstable when it comes to character repetition, which is a commonly found form of informal spelling. For example, the example in Figure 3 demonstrates the model’s general incapability of dealing with long repetitions. This may be fixed in via a rule-based system to clean up repetition, under the somewhat untenable assumption that repetition does not entail different meanings.

Figure 3: the outputs of the informal-formal transformer model for varying times of “y” repetition in the word “hey”.

Thirdly, the model tends to fill in unmentioned words or phrases when the input is only a phrase, especially a person name. For example, when the input is “Yonglin Wang”, the model would generate “Yonglin Wang is a good match”, which seems to come from the training data on entries under the topic Family and Relationship.

Moreover, we noticed that the model is sensitive to casing, in that input such as “lol” and “Lol” has different outputs, “I find it amusing” and “Do you know?” respectively. This might be the result of leaving the corpus of mixed casing; to fix this, one can collect more data where different casing of the same word are present, or, include lowercased tokens as part of the input with feature engineering.

In summary, from the challenges discussed above, we can see that, for the task of rewriting online languages, it was equally important to maintain an up-to-date and comprehensive dataset and to cast the problem properly as one of the many available modeling tasks.

On the engineering side, in a less-than-ideal situation such as in our case, it is important to note that when Python API for a program is not available, one can always try running the command line version of the program as a subprocess or as a server, and manage it using subprocess and web server packages in Python.

References

- Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2018. [Disentangled Representation Learning for Non-Parallel Text Style Transfer](#). *arXiv:1808.04339 [cs]*. ArXiv: 1808.04339.
- Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. [Bag of Tricks for Efficient Text Classification](#). *arXiv:1607.01759 [cs]*. ArXiv: 1607.01759.
- Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. [fairseq: A Fast, Extensible Toolkit for Sequence Modeling](#). *arXiv:1904.01038 [cs]*. ArXiv: 1904.01038.
- Sudha Rao and Joel Tetreault. 2018. [Dear Sir or Madam, May I introduce the GYAFC Dataset: Corpus, Benchmarks and Metrics for Formality Style Transfer](#). *arXiv:1803.06535 [cs]*. ArXiv: 1803.06535.
- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. [Attention Is All You Need](#). *arXiv:1706.03762 [cs]*. ArXiv: 1706.03762.