Keep Calm and Switch On! Preserving Sentiment and Fluency in Semantic Text Exchange

Steven Y. Feng* **Aaron W. Li*** **Jesse Hoey**

David R. Cheriton School of Computer Science
University of Waterloo
Waterloo, Ontario, Canada
{sy2feng, w89li, jhoey}@uwaterloo.ca

Abstract

In this paper, we present a novel method for measurably adjusting the semantics of text while preserving its sentiment and fluency, a task we call *semantic text exchange*. This is useful for text data augmentation and the semantic correction of text generated by chatbots and virtual assistants. We introduce a pipeline called *SMERTI* that combines entity replacement, similarity masking, and text infilling. We measure our pipeline's success by its *Semantic Text Exchange Score (STES)*: the ability to preserve the original text's sentiment and fluency while adjusting semantic content. We propose to use *masking (replacement) rate threshold* as an adjustable parameter to control the amount of semantic change in the text. Our experiments demonstrate that SMERTI can outperform baseline models on Yelp reviews, Amazon reviews, and news headlines.

1 Introduction

There has been significant research on style transfer, with the goal of changing the style of text while preserving its semantic content. The alternative where semantics are adjusted while keeping style intact, which we call *semantic text exchange (STE)*, has not been investigated to the best of our knowledge. Consider the following example, where the replacement entity defines the new semantic context:

Original Text: *It is **sunny** outside! Ugh, that means I must **wear sunscreen**. I hate being **sweaty** and **sticky all over.***
Replacement Entity: weather = ***rainy***
Desired Text: *It is **rainy** outside! Ugh, that means I must **bring an umbrella**. I hate being **wet** and **having to carry it around.***

The weather within the original text is sunny, whereas the actual weather may be rainy. Not only is the word *sunny* replaced with *rainy*, but the rest of the text's content is changed while preserving its negative sentiment and fluency.

With the rise of natural language processing (NLP) has come an increased demand for massive amounts of text data. Manually collecting and scraping data requires a significant amount of time and effort, and data augmentation techniques for NLP are limited compared to fields such as computer vision. STE can be used for text data augmentation by producing various modifications of a piece of text that differ in semantic content.

Another use of STE is in building emotionally aligned chatbots and virtual assistants. This is useful for reasons such as marketing, overall enjoyment of interaction, and mental health therapy. However, due to limited data with emotional content in specific semantic contexts, the generated text may contain incorrect semantic content. STE can adjust text semantics (e.g. to align with reality or a specific task) while preserving emotions.

One specific example is the development of virtual assistants with adjustable socio-emotional personalities in the effort to construct assistive technologies for persons with cognitive disabilities. Adjusting the emotional delivery of text in subtle ways can have a strong effect on the adoption of the technologies (Robillard et al., 2018). It is challenging to transfer style this subtly due to lack of datasets on specific topics with consistent emotions. Instead, large datasets of emotionally consistent interactions not confined to specific topics exist. Hence, it is effective to generate text with a particular emotion and then adjust its semantics.

We propose a pipeline called SMERTI (pronounced '*smarty*') for STE.[1] Combining entity replacement (ER), similarity masking (SM), and text

*Authors contributed equally

[1]Code for SMERTI (including Google Colab links) can be found at https://github.com/styfeng/SMERTI

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 2701–2711,
Hong Kong, China, November 3–7, 2019.

infilling (TI), SMERTI can modify the semantic content of text. We define a metric called the *Semantic Text Exchange Score (STES)* that evaluates the overall ability of a model to perform STE, and an adjustable parameter *masking (replacement) rate threshold (MRT/RRT)* that can be used to control the amount of semantic change.

We evaluate on three datasets: Yelp and Amazon reviews (He and McAuley, 2016), and Kaggle news headlines (Misra, 2018). We implement three baseline models for comparison: Noun WordNet Semantic Text Exchange Model (NWN-STEM), General WordNet Semantic Text Exchange Model (GWN-STEM), and Word2Vec Semantic Text Exchange Model (W2V-STEM).

We illustrate the STE performance of two SMERTI variations on the datasets, demonstrating outperformance of the baselines and pipeline stability. We also run a human evaluation supporting our results. We analyze the results in detail and investigate relationships between the semantic change, fluency, sentiment, and MRT/RRT. Our major contributions can be summarized as:

- We define a new task called *semantic text exchange (STE)* with increasing importance in NLP applications that modifies text semantics while preserving other aspects such as sentiment.
- We propose a pipeline *SMERTI* capable of multi-word entity replacement and text infilling, and demonstrate its outperformance of baselines.
- We define an evaluation metric for overall performance on semantic text exchange called the *Semantic Text Exchange Score (STES)*.

2 Related Work

2.1 Word and Sentence-level Embeddings

Word2Vec (Mikolov et al., 2013a,b) allows for analogy representation through vector arithmetic. We implement a baseline (W2V-STEM) using this technique. The Universal Sentence Encoder (USE) (Cer et al., 2018) encodes sentences and is trained on a variety of web sources and the Stanford Natural Language Inference corpus (Bowman et al., 2015). Flair embeddings (Akbik et al., 2018) are based on architectures such as BERT (Devlin et al., 2019). We use USE for SMERTI as it is designed for transfer learning and shows higher performance on textual similarity tasks compared to other models (Perone et al., 2018).

2.2 Text Infilling

Text infilling is the task of filling in missing parts of sentences called masks. MaskGAN (Fedus et al., 2018) is restricted to a single word per mask token, while SMERTI is capable of variable length infilling for more flexible output. Zhu et al. (2019) uses a transformer-based architecture. They fill in random masks, while SMERTI fills in masks guided by semantic similarity, resulting in more natural infilling and fulfillment of the STE task.

2.3 Style and Sentiment Transfer

Notable works in style/sentiment transfer include (Shen et al., 2017; Fu et al., 2018; Li et al., 2018; Xu et al., 2018). They attempt to learn latent representations of various text aspects such as its context and attributes, or separate style from content and encode them into hidden representations. They then use an RNN decoder to generate a new sentence given a targeted sentiment attribute.

2.4 Review Generation

Hovy (2016) generates fake reviews from scratch using language models. (Lipton et al., 2015; Dong et al., 2017; Juuti et al., 2018) generate reviews from scratch given auxiliary information (e.g. the item category and star rating). Yao et al. (2017) generates reviews using RNNs with two components: generation from scratch and review customization (Algorithm 2 in Yao et al. (2017)). They define review customization as modifying the generated review to fit a new topic or context, such as from a Japanese restaurant to an Italian one. They condition on a keyword identifying the desired context, and replace similar nouns with others using WordNet (Miller, 1995). They require a *"reference dataset"* (required to be *"on topic"*; easy enough for restaurant reviews, but less so for arbitrary conversational agents). As noted by Juuti et al. (2018), the method of Yao et al. (2017) may also replace words independently of context. We implement their review customization algorithm (NWN-STEM) and a modified version (GWN-STEM) as baseline models.

3 SMERTI

3.1 Overview

The task is to transform a corpus C of lines of text S_i and associated replacement entities RE_i : $C = \{(S_1, RE_1), (S_2, RE_2), \ldots, (S_n, RE_n)\}$ to a modified corpus $\hat{C} = \{\hat{S}_1, \hat{S}_2, \ldots, \hat{S}_n\}$, where

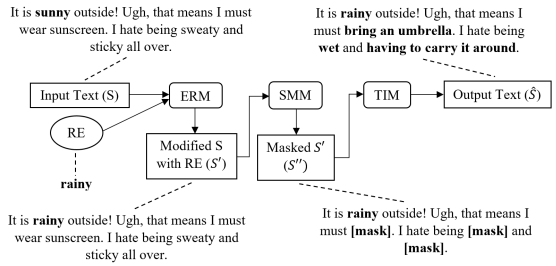


Figure 1: Overall architecture and example, showing the three modules: Entity Replacement (ERM), Similarity Masking (SMM), and Text Infilling (TIM)

$\hat{S}_i$ are the original text lines S_i replaced with RE_i and overall semantics adjusted. SMERTI consists of the following modules, shown in Figure 1:

1. Entity Replacement Module (ERM): Identify which word(s) within the original text are best replaced with the RE, which we call the Original Entity (OE). We replace OE in S with RE. We call this modified text S'.
2. Similarity Masking Module (SMM): Identify words/phrases in S' similar to OE and replace them with a *[mask]*. Group adjacent *[mask]*s into a single one so we can fill a variable length of text into each. We call this masked text S''.
3. Text Infilling Module (TIM): Fill in *[mask]* tokens with text that better suits the RE. This will modify semantics in the rest of the text. This final output text is called $\hat{S}$.

3.2 Entity Replacement Module (ERM)

For entity replacement, we use a combination of the Universal Sentence Encoder (Cer et al., 2018) and Stanford Parser (Chen and Manning, 2014).

Stanford Parser

The Stanford Parser is a constituency parser that determines the grammatical structure of sentences, including phrases and part-of-speech (POS) labelling. By feeding our RE through the parser, we are able to determine its parse-tree. Iterating through the parse-tree and its sub-trees, we can obtain a list of constituent tags for the RE. We then feed our input text S through the parser, and through a similar process, we can obtain a list of leaves (where leaves under a single label are concatenated) that are equal or similar to any of the RE constituent tags. This generates a list of entities having the same (or similar) grammatical structure as the RE, and are likely candidates for the OE. We then feed these entities along with the RE into the Universal Sentence Encoder (USE).

Universal Sentence Encoder (USE)

The USE is a sentence-level embedding model that comes with a deep averaging network (DAN) and transformer model (Cer et al., 2018). We choose the transformer model as these embeddings take context into account, and the exact same word/phrase will have a different embedding depending on its context and surrounding words.

We compute the semantic similarity between two embeddings u and v: $sim(u, v)$, using the angular (cosine) distance, defined as: $\cos(\theta_{u,v}) = (u \cdot v)/(||u||||v||)$, such that $sim(u, v) = 1 - \frac{1}{\pi} arccos(\cos(\theta_{u,v}))$. Results are in $[0, 1]$, with higher values representing greater similarity.

Using USE and the above equation, we can identify words/phrases within the input text S which are most similar to RE. To assist with this, we use the Stanford Parser as described above to obtain a list of candidate entities. In the rare case that this list is empty, we feed in each word of S into USE, and identify which word is the most similar to RE. We then replace the most similar entity or word (OE) with the RE and generate S'.

An example of this entity replacement process is in Figure 2. Two parse-trees are shown: for RE (a) and S (b) and (c). Figure 2(d) is a semantic similarity heat-map generated from the USE embeddings of the candidate OEs and RE, where values are similarity scores in the range $[0, 1]$.

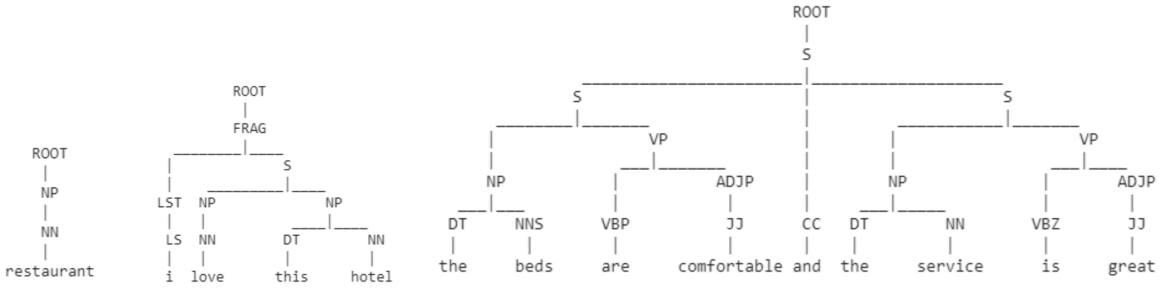
As seen in Figure 2(d), we calculate semantic similarities between RE and entities within S which have *noun* constituency tags. Looking at the row for our RE *restaurant*, the most similar entity (excluding itself) is *hotel*. We can then generate:

S' = i love this restaurant ! the beds are comfortable and the service is great !

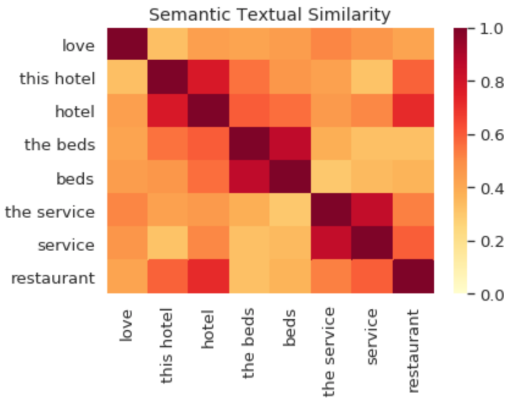
3.3 Similarity Masking Module (SMM)

Next, we mask words similar to OE to generate S'' using USE. We look at semantic similarities between every word in S and OE, along with semantic similarities between OE and the candidate entities determined in the previous ERM step to broaden the range of phrases our module can mask. We ignore RE, OE, and any entities or phrases containing OE (for example, *'this hotel'*).

After determining words similar to the OE (discussed below), we replace each of them with a



(a) (b) (c)



(d)

Figure 2: ERM example with S = *i love this hotel ! the beds are comfortable and the service is great !* and RE = *restaurant* showing (a) Parse tree for RE; (b) and (c) Parse tree for S; (d) Semantic similarity heat map

[mask] token. Next, we replace *[mask]* tokens adjacent to each other with a single *[mask]*.

We set a base similarity threshold (ST) that selects a subset of words to mask. We compare the actual fraction of masked words to the masking rate threshold (MRT), as defined by the user, and increase ST in intervals of 0.05 until the actual masking rate falls below the MRT.[2] Some sample masked outputs (S'') using various MRT-ST combinations for the previous example are shown in Table 1 (more examples in Appendix A).

The MRT is similar to the temperature parameter used to control the "novelty" of generated text in works such as Yao et al. (2017). A high MRT means the user wants to generate text very semantically dissimilar to the original, and may be desired in cases such as creating a lively chatbot or correcting text that is heavily incorrect semantically. A low MRT means the user wants to generate text semantically similar to the original, and may be desired in cases such as text recovery, grammar correction, or correcting a minor semantic error in text. By varying the MRT, various pieces of text that differ semantically in subtle ways can be generated, assisting greatly with text data augmentation. The MRT also affects sentiment and fluency, as we show in Section 6.5.

[2]There are certain cases where two or more outputs for different MRT may be equal. This occurs when a valid ST cannot be found that masks a larger portion of the sentence without going over MRT.

MRT	ST	Masked Outputs
0.2	0.4	*i love this restaurant ! [mask] are comfortable and the [mask] is great !*
0.4	0.3	*i love this restaurant ! [mask] are [mask] and [mask] is great !*
0.6	0.2	*[mask] this restaurant ! [mask] are [mask] and [mask] is great !*
0.8	0.1	*[mask] restaurant ! [mask] and [mask] great !*

Table 1: Masked outputs for different masking rate thresholds (MRT) and base similarity thresholds (ST)

3.4 Text Infilling Module (TIM)

We use two seq2seq models for our TIM: an RNN (recurrent neural network) model (Sutskever et al., 2014) (called SMERTI-RNN), and a transformer model (called SMERTI-Transformer).

Bidirectional RNN with Attention

We use a bidirectional variant of the GRU (Cho et al., 2014), and hence two RNNs for the encoder: one reads the input sequence in standard sequential order, and the other is fed this sequence in reverse. The outputs are summed at each time step, giving us the ability to encode information from both past and future context.

The decoder generates the output in a sequential token-by-token manner. To combat information loss, we implement the attention mechanism (Bahdanau et al., 2015). We use a Luong attention layer (Luong et al., 2015) which uses global attention, where all the encoder's hidden states are considered, and use the decoder's current time-step

hidden state to calculate attention weights. We use the dot score function for attention, where h_t is the current target decoder state and $\bar{h}_s$ is all encoder states: $score(h_t, \bar{h}_s) = h_t^T \bar{h}_s$.

Transformer

Our second model makes use of the transformer architecture, and our implementation replicates Vaswani et al. (2017). We use an encoder-decoder structure with a multi-head self-attention token decoder to condition on information from both past and future context. It maps a query and set of key-value pairs to an output. The queries and keys are of dimension d_k, and values of dimension d_v. To compute the attention, we pack a set of queries, keys, and values into matrices Q, K, and V, respectively. The matrix of outputs is computed as:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (1)$$

Multi-head attention allows the model to jointly attend to information from different positions. The decoder can make use of both local and global semantic information while filling in each *[mask]*.

4 Experiment

4.1 Datasets

We train our two TIMs on the three datasets. The Amazon dataset (He and McAuley, 2016) contains over 83 million user reviews on products, with duplicate reviews removed. The Yelp dataset includes over six million user reviews on businesses. The news headlines dataset from Kaggle contains approximately $200,000$ news headlines from 2012 to 2018 obtained from HuffPost (Misra, 2018).

We filter the text to obtain reviews and headlines which are English, do not contain hyperlinks and other obvious noise, and are less than 20 words long. We found that many longer than twenty words ramble on and are too verbose for our purposes. Rather than filtering by individual sentences we keep each text in its entirety so SMERTI can learn to generate multiple sentences at once. We preprocess the text by lowercasing and removing rare/duplicate punctuation and space.

For Amazon and Yelp, we treat reviews greater than three stars as containing positive sentiment, equal to three stars as neutral, and less than three stars as negative. For each training and testing set, we include an equal number of randomly selected positive and negative reviews, and half as many neutral reviews. This is because neutral reviews only occupy one out of five stars compared to positive and negative which occupy two each. Our dataset statistics can be found in Appendix B.

4.2 Experiment Details

To set up our training and testing data for text infilling, we mask the text. We use a tiered masking approach: for each dataset, we randomly mask 15% of the words in one-third of the lines, 30% of the words in another one-third, and 45% in the remaining one-third. These masked texts serve as the inputs, while the original texts serve as the ground-truth. This allows our TIM models to learn relationships between masked words and relationships between masked and unmasked words.

The bidirectional RNN decoder fills in blanks one by one, with the objective of minimizing the cross entropy loss between its output and the ground-truth. We use a hidden size of 500, two layers for the encoder and decoder, teacher-forcing ratio of 1.0, learning rate of 0.0001, dropout of 0.1, batch size of 64, and train for up to 40 epochs.

For the transformer, we use scaled dot-product attention and the same hyperparameters as Vaswani et al. (2017). We use the Adam optimizer (Kingma and Ba, 2014) with $\beta_1 = 0.9, \beta_2 = 0.98$, and $\epsilon = 10^{-9}$. As in Vaswani et al. (2017), we increase the $learning\_rate$ linearly for the first $warmup\_steps$ training steps, and then decrease the $learning\_rate$ proportionally to the inverse square root of the step number. We set $factor = 1$ and use $warmup\_steps = 2000$. We use a batch size of 4096, and we train for up to 40 epochs.

4.3 Baseline Models

We implement three models to benchmark against.[3] First is NWN-STEM (Algorithm 2 from Yao et al. (2017)). We use the training sets as the "reference review sets" to extract similar nouns to the RE (using MIN_{sim} = 0.1). We then replace nouns in the text similar to the RE with nouns extracted from the associated reference review set.

Secondly, we modify NWN-STEM to work for verbs and adjectives[4], and call this GWN-STEM. From the reference review sets, we extract similar nouns, verbs, and adjectives to the RE (using

[3] See Appendix C for more implementation details

[4] WordNet can only work for single words (and not phrases). Also, it turns out that it cannot work for most adjective REs, as discussed in Appendix C

MIN$_{sim}$ = 0.1), where the RE is now not restricted to being a noun. We replace nouns, verbs, and adjectives in the text similar to the RE with those extracted from the associated reference review set.

Lastly, we implement W2V-STEM using Gensim (Řehůřek and Sojka, 2010). We train uni-gram Word2Vec models for single word REs, and four-gram models for phrases. Models are trained on the training sets. We use cosine similarity to determine the most similar word/phrase in the input text to RE, which is the replaced OE. For all other words/phrases, we calculate $w_i' = w_i - w_{OE} + w_{RE}$, where w_i is the original word/phrase's embedding vector, w_{OE} is the OE's, w_{RE} is the RE's, and w_i' is the resulting embedding vector. The replacement word/phrase is w_i''s nearest neighbour. We use similarity thresholds to adjust replacement rates (RR) and produce text under various replacement rate thresholds (RRT).

5 Evaluation

5.1 Evaluation Setup

We manually select 10 nouns, 10 verbs, 10 adjectives, and 5 phrases from the top 10% most frequent words/phrases in each test set as our evaluation REs. We filter the verbs and adjectives through a list of sentiment words (Hu and Liu, 2004) to ensure we do not choose REs that would obviously significantly alter the text's sentiment.[5]

For each evaluation RE, we choose one-hundred lines from the corresponding test set that does not already contain RE. We choose lines with at least five words, as many with less carry little semantic meaning (e.g. *'Great!', 'It is okay'*). For Amazon and Yelp, we choose 50 positive and 50 negative lines per RE.[6] We repeat this process three times, resulting in three sets of 1000 lines per dataset per POS (excluding phrases), and three sets of 500 lines per dataset for phrases. Our final results are averaged metrics over these three sets.

For SMERTI-Transformer, SMERTI-RNN, and W2V-STEM, we generate four outputs per text for MRT/RRT of 20%, 40%, 60%, and 80%, which represent upper-bounds on the percentage of the input that can be masked and/or replaced. Note that NWN-STEM and GWN-STEM can only evaluate on limited POS and their maximum replacement rates are limited.[7] We select *MIN$_{sim}$* values of 0.075 and 0 for nouns and 0.1 and 0 for verbs, as these result in replacement rates approximately equal to the actual MR/RR of the other models' outputs for 20% and 40% MRT/RRT, respectively.

5.2 Key Evaluation Metrics

Fluency (SLOR) We use syntactic log-odds ratio (SLOR) (Kann et al., 2018) for sentence level fluency and modify from their word-level formula to character-level ($SLOR_c$). We use Flair perplexity values from a language model trained on the One Billion Words corpus (Chelba et al., 2013):

$$SLOR_c(S) = \frac{1}{|S|}(ln(p_M(S)) - \frac{ln(\prod_{w \in S} p_M(w))}{\sum_{w \in S} |w|} \quad (2)$$

$$= -ln(PPL_s) + \frac{\sum_{w \in S} |w| ln(PPL_W)}{\sum_{w \in S} |w|} \quad (3)$$

where $|S|$ and $|w|$ are the character lengths of the input text S and the word w, respectively, $p_M(S)$ and $p_M(w)$ are the probabilities of S and w under the language model M, respectively, and PPL_S and PPL_w are the character-level perplexities of S and w, respectively. SLOR (from hereon we refer to character-level SLOR as simply SLOR) measures aspects of text fluency such as grammaticality. Higher values represent higher fluency.

We rescale resulting SLOR values to the interval [0,1] by first fitting and normalizing a Gaussian distribution. We then truncate normalized data points outside [-3,3], which shifts approximately 0.69% of total data. Finally, we divide each data point by six and add 0.5 to each result.

Sentiment Preservation Accuracy (SPA) is defined as the percentage of outputs that carry the same sentiment as the input. We use VADER (Hutto and Gilbert, 2014) to evaluate sentiment as positive, negative, or neutral. It handles typos, emojis, and other aspects of online text.

Content Similarity Score (CSS) ranges from 0 to 1 and indicates the semantic similarity between generated text and the RE. A value closer to 1 indicates stronger semantic exchange, as the output is closer in semantic content to the RE. We also use the USE for this due to its design and strong performance as previously mentioned.

5.3 Semantic Text Exchange Score (STES)

We come up with a single score to evaluate overall performance of a model on STE that combines

[5] A list of the chosen REs along with more detailed explanation of how they were selected is in Appendix D

[6] We don't test on neutral reviews as evaluation of accuracy in sentiment is less well defined (i.e. most "neutral" reviews actually carry more positive or negative sentiment)

[7] See Appendix C for explanations

Model	SPA	SLOR	CSS	STES
Input	---	0.5962	0.1166	---
SMERTI-Transformer	0.6606	**0.5255 (-11.86%)**	0.2857 (+145.03%)	0.4337
SMERTI-RNN	0.6574	0.5122 (-14.09%)	**0.2927 (+151.03%)**	**0.4354**
W2V-STEM	0.6667	0.4672 (-21.64%)	0.2851 (+144.51%)	0.4197
GWN-STEM	0.8903	0.4864 (-18.42%)	0.1419 (+21.70%)	0.2934
NWN-STEM	**0.9116**	0.4832 (-18.95%)	0.1335 (+14.49%)	0.2814

Table 2: Overall average results by model (with % changes from the input)

the key evaluation metrics. It uses the harmonic mean, similar to the F_1 score (or F-score) (Chinchor, 1992; Rijsbergen, 1979), and we call it the *Semantic Text Exchange Score (STES)*:

$$STES = \frac{3 * A * B * C}{A * B + A * C + B * C} \quad (4)$$

where A is SPA, B is SLOR, and C is CSS. STES ranges between 0 and 1, with scores closer to 1 representing higher overall performance. Like the F_1 score, STES penalizes models which perform very poorly in one or more metrics, and favors balanced models achieving strong results in all three.

5.4 Automatic Evaluation Results

Table 2 shows overall average results by model.[8] Table 3 shows outputs for a Yelp example.[9]

As observed from Table 3 (see also Appendix F), SMERTI is able to generate high quality output text similar to the RE while flowing better than other models' outputs. It can replace entire phrases and sentences due to its variable length infilling. Note that for nouns, the outputs from GWN-STEM and NWN-STEM are equivalent.[10]

5.5 Human Evaluation Setup

We conduct a human evaluation with eight participants, 6 males and 2 females, that are affiliated project researchers aged 20-39 at the University of Waterloo.[11] We randomly choose one evaluation line for a randomly selected word or phrase for each POS per dataset. The input text and each model's output (for 40% MRT/RRT - chosen as a good middle ground) for each line is presented to participants, resulting in a total of 54 pieces of text, and rated on the following criteria from 1-5:

- *RE Match*: *"How related is the entire text to the concept of [X]", where [X] is a word or phrase (1 - not at all related, 3 - somewhat related, 5 - very related)*. Note here that [X] is a given RE.
- *Fluency*: *"Does the text make sense and flow well?" (1 - not at all, 3 - somewhat, 5 - very)*
- *Sentiment*: *"How do you think the author of the text was feeling?" (1 - very negative, 3 - neutral, 5 - very positive)*

Each participant evaluates every piece of text. They are presented with a single piece of text at a time, with the order of models, POS, and datasets completely randomized.

5.6 Human Evaluation Results

Average human evaluation scores are displayed in Table 4. *Sentiment Preservation* (between 0 and 1) is calculated by comparing the average *Sentiment* rating for each model's output text to the *Sentiment* rating of the input text, and if both are less than 2.5 (negative), between 2.5 and 3.5 inclusive (neutral), or greater than 3.5 (positive), this is counted as a valid case of *Sentiment Preservation*. We repeat this for every evaluation line to calculate the final values per model. Harmonic means of all three metrics (using rescaled 0-1 values of *RE Match* and *Fluency*) are also displayed.

6 Analysis

6.1 Performance by Model

As seen in Table 2, both SMERTI variations achieve higher STES and outperform the other models overall, with the WordNet models performing the worst. SMERTI excels especially on fluency and content similarity. The transformer variation achieves slightly higher SLOR, while the RNN variation achieves slightly higher CSS.

The WordNet models perform strongest in sentiment preservation (SPA), likely because they modify little of the text and only verbs and nouns. They achieve by far the lowest CSS, likely in part due to this limited text replacement. They also do not account for context, and many words (e.g. proper nouns) do not exist in WordNet. Overall, the WordNet models are not very effective at STE.

W2V-STEM achieves the lowest SLOR, especially for higher RRT, as supported by the example in Table 3 (see also Appendix F). W2V-STEM and WordNet models output grammatically incorrect text that flows poorly. In many cases, words are

[8] See Appendix E for tables and graphs of detailed results broken down by POS, dataset, and MRT/RRT

[9] See Appendix F for many more example outputs from each model for various POS and datasets

[10] See Appendix C for explanations

[11] The authors are not part of the human evaluation

Input text: great food , large portions ! my family and i really enjoyed our saturday morning breakfast .
Replacement entity: pizza

MRT/RRT	Generated Output
SMERTI-Transformer	
20%	great pizza , large slices ! my family and i really enjoyed our saturday morning lunch .
40%,60%	great pizza , large slices ! service was terrific and i really enjoyed our saturday morning lunch .
80%	great pizza , chewy crust ! nice ambiance and i really enjoyed it .
SMERTI-RNN	
20%	great pizza , large delivery ! my family and i really enjoyed our saturday morning place .
40%,60%	great pizza , large delivery ! good beer and i really enjoyed our saturday morning place .
80%	great pizza , amazing pizza ! reasonable and i really enjoyed everyone .
W2V-STEM	
20%	great pizza , large portions ! my family and i really enjoyed our saturday morning breakfast .
40%	great pizza , large slices ! my family dough i crust enjoyed our saturday morning breakfast .
60%	awesome pizza , large slices ! my mom dough i crust enjoyed our saturday morning bagel .
80%	awesome pizza , slices slices ! my mom dough we crust liked our sunday morning bagel .
GWN / NWN-STEM	
20%	great food , large stuff ! my family and i really enjoyed our saturday i breakfast .
40%	great food , large stuff ! my i and i really enjoyed our saturday i breakfast .

Table 3: Generated output text by model for various masking rates on a Yelp evaluation example

Model	Input Text	SMERTI-Transformer	SMERTI-RNN	W2V-STEM	GWN-STEM	NWN-STEM
RE Match (1-5)	1.8229	**3.5833**	3.5000	3.4792	2.2500	2.1250
Fluency (1-5)	4.1250	2.8750	2.8230	2.0830	2.5000	**2.9583**
Sentiment Preservation (0-1)	---	0.7500	0.5833	0.6667	0.8333	**1.0000**
Harmonic Mean (0-1)	---	**0.5982**	0.5446	0.4408	0.4245	0.4547

Table 4: Average human evaluation scores by model

Model	Input Text	SMERTI-Transformer	SMERTI-RNN	W2V-STEM	GWN-STEM	NWN-STEM
Avg. TTR	0.9488	0.9429	0.9367	0.9025	0.9042	0.8950

Table 5: Average TTR values by model

repeated multiple times. We analyze the average Type Token Ratio (TTR) values of each model's outputs, which is the ratio of unique divided by total words. As shown in Table 5, the SMERTI variations achieve the highest TTR, while W2V-STEM and NWN-STEM the lowest.

Note that while W2V-STEM achieves lower CSS than SMERTI, it performs comparably in this aspect. This is likely due to its vector arithmetic operations algorithm, which replaces each word with one more similar to the RE. This is also supported by the lower TTR, as W2V-STEM frequently outputs the same words multiple times.

6.2 Performance By Model - Human Results

As seen in Table 4, the SMERTI variations outperform all baseline models overall, particularly in *RE Match*. SMERTI-Transformer performs the best, with SMERTI-RNN second. The WordNet models achieve high *Sentiment Preservation*, but much lower on *RE Match*. W2V-STEM achieves comparably high *RE Match*, but lowest *Fluency*.

These results correspond well with our automatic evaluation results in Table 2. We look at the Pearson correlation values between *RE Match*, *Fluency*, and *Sentiment Preservation* with CSS, SLOR, and SPA, respectively. These are 0.9952, 0.9327, and 0.8768, respectively, demonstrating that our automatic metrics are highly effective and correspond well with human ratings.

POS	Nouns	Verbs	Adjectives	Phrases
Input SLOR	0.5971	0.5960	0.5977	0.5939
Input CSS	0.1175	0.1287	0.0935	0.1267
SMERTI SPA	0.7118	0.6586	0.6195	0.6460
SMERTI SLOR	0.5100	0.5078	0.4963	0.5613
SMERTI CSS	0.2982	0.2579	0.2434	0.3572
SMERTI STES	0.4464	0.4073	0.3876	0.4895

Table 6: Input text's avg. SLOR, CSS, and SMERTI's avg. SPA, SLOR, CSS, and STES by POS

6.3 SMERTI's Performance By POS

As seen from Table 6[12] , SMERTI's SPA values are highest for nouns, likely because they typically carry little sentiment, and lowest for adjectives, likely because they typically carry the most.

SLOR is lowest for adjectives and highest for phrases and nouns. Adjectives typically carry less semantic meaning and SMERTI likely has more trouble figuring out how best to infill the text. In contrast, nouns typically carry more, and phrases the most (since they consist of multiple words).

SMERTI's CSS is highest for phrases then nouns, likely due to phrases and nouns carrying

[12]Note that the SMERTI values in Tables 6 to 8 refer to the average between SMERTI-Transformer and SMERTI-RNN

Dataset	Amazon	Yelp	News Headlines
Input SLOR	0.6230	0.5532	0.6123
Input CSS	0.1113	0.1429	0.0955
SMERTI SPA	0.6738	0.7238	0.5793
SMERTI SLOR	0.5406	0.4765	0.5394
SMERTI CSS	0.2778	0.2835	0.3062
SMERTI STES	0.4326	0.4281	0.4381

Table 7: Input text's avg. SLOR, CSS, and SMERTI's avg. SPA, SLOR, CSS, and STES by dataset

MRT / RRT	20%	40%	60%	80%
SMERTI SPA	0.7994	0.7124	0.6202	0.5040
SMERTI SLOR	0.5489	0.5375	0.5197	0.4694
SMERTI CSS	0.2461	0.2557	0.2828	0.3722
SMERTI STES	0.4204	0.4181	0.4240	0.4410
SMERTI BLEU	0.5490	0.3527	0.1752	0.0448

Table 8: SMERTI's avg. SPA, SLOR, CSS, STES, and BLEU by MRT/RRT

more semantic meaning, making it easier to generate semantically similar text. Both SMERTI's and the input text's CSS are lowest for adjectives, likely because they carry little semantic meaning.

Overall, SMERTI appears to be more effective on nouns and phrases than verbs and adjectives.

6.4 SMERTI's Performance By Dataset

As seen in Table 7, SMERTI's SPA is lowest for news headlines. Amazon and Yelp reviews naturally carry stronger sentiment, likely making it easier to generate text with similar sentiment.

Both SMERTI's and the input text's SLOR appear to be lower for Yelp reviews. This may be due to many reasons, such as more typos and emojis within the original reviews, and so forth.

SMERTI's CSS values are slightly higher for news headlines. This may be due to them typically being shorter and carrying more semantic meaning as they are designed to be attention grabbers.

Overall, it seems that using datasets which inherently carry more sentiment will lead to better sentiment preservation. Further, the quality of the dataset's original text, unsurprisingly, influences the ability of SMERTI to generate fluent text.

6.5 SMERTI's Performance By MRT/RRT

From Table 8, it can be seen that as MRT/RRT increases, SMERTI's SPA and SLOR decrease while CSS increases. These relationships are very strong as supported by the Pearson correlation values of -0.9972, -0.9183, and 0.9078, respectively. When SMERTI can alter more text, it has the opportunity to replace more related to sentiment while producing more of semantic similarity to the RE.

Further, SMERTI generates more of the text itself, becoming less similar to the human-written input, resulting in lower fluency. To further demonstrate this, we look at average SMERTI BLEU (Papineni et al., 2002) scores against MRT/RRT, shown in Table 8. BLEU generally indicates how close two pieces of text are in content and structure, with higher values indicating greater similarity. We report our final BLEU scores as the average scores of 1 to 4-grams. As expected, BLEU decreases as MRT/RRT increases, and this relationship is very strong as supported by the Pearson correlation value of -0.9960.

It is clear that MRT/RRT represents a trade-off between CSS against SPA and SLOR. It is thus an adjustable parameter that can be used to control the generated text, and balance semantic exchange against fluency and sentiment preservation.

7 Conclusion and Future Work

We introduced the task of semantic text exchange (STE), demonstrated that our pipeline SMERTI performs well on STE, and proposed an STES metric for evaluating overall STE performance. SMERTI outperformed other models and was the most balanced overall. We also showed a trade-off between semantic exchange against fluency and sentiment preservation, which can be controlled by the masking (replacement) rate threshold.

Potential directions for future work include adding specific methods to control sentiment, and fine-tuning SMERTI for preservation of persona or personality. Experimenting with other text infilling models (e.g. fine-tuning BERT (Devlin et al., 2019)) is also an area of exploration. Lastly, our human evaluation is limited in size and a larger and more diverse participant pool is needed.

We conclude by addressing potential ethical misuses of STE, including assisting in the generation of spam and fake-reviews/news. These risks come with any intelligent chatbot work, but we feel that the benefits, including usage in the detection of misuse such as fake-news, greatly outweigh the risks and help progress NLP and AI research.

Acknowledgments

We thank our anonymous reviewers, study participants, and Huawei Technologies Co., Ltd. for financial support.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Samuel R. Bowman, Gabor Angeli, Christopher Potts, and Christopher D. Manning. 2015. A large annotated corpus for learning natural language inference. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 632–642, Lisbon, Portugal. Association for Computational Linguistics.

Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Brian Strope, and Ray Kurzweil. 2018. Universal sentence encoder for English. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 169–174, Brussels, Belgium. Association for Computational Linguistics.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, and Phillipp Koehn. 2013. One billion word benchmark for measuring progress in statistical language modeling. *CoRR*, abs/1312.3005.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 740–750.

Nancy Chinchor. 1992. Muc-4 evaluation metrics. In *Proceedings of the 4th conference on Message understanding*, pages 22–29. Association for Computational Linguistics.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Shaohan Huang, Furu Wei, Mirella Lapata, Ming Zhou, and Ke Xu. 2017. Learning to generate product reviews from attributes. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 623–632.

William Fedus, Ian J. Goodfellow, and Andrew M. Dai. 2018. Maskgan: Better text generation via filling in the _____. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Zhenxin Fu, Xiaoye Tan, Nanyun Peng, Dongyan Zhao, and Rui Yan. 2018. Style transfer in text: Exploration and evaluation. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Ruining He and Julian McAuley. 2016. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517. International World Wide Web Conferences Steering Committee.

Dirk Hovy. 2016. The enemy in your own camp: How well can we detect statistically-generated fake reviews–an adversarial study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, volume 2, pages 351–356.

Minqing Hu and Bing Liu. 2004. Mining and summarizing customer reviews. In *Proceedings of the tenth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 168–177. ACM.

Clayton J Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Eighth international AAAI conference on weblogs and social media*.

Mika Juuti, Bo Sun, Tatsuya Mori, and N Asokan. 2018. Stay on-topic: Generating context-specific fake restaurant reviews. In *European Symposium on Research in Computer Security*, pages 132–151. Springer.

Katharina Kann, Sascha Rothe, and Katja Filippova. 2018. Sentence-level fluency evaluation: References help, but can be spared! In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 313–323, Brussels, Belgium. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *International Conference on Learning Representations*.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: a simple approach to sentiment and style transfer. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1865–1874, New Orleans, Louisiana. Association for Computational Linguistics.

Zachary Chase Lipton, Sharad Vikram, and Julian J. McAuley. 2015. Capturing meaning in product reviews with character-level generative text models. *ArXiv*, abs/1511.03683.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

George A Miller. 1995. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41.

Rishabh Misra. 2018. News headlines dataset for sarcasm detection. https://rishabhmisra.github.io/publications/.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Christian S Perone, Roberto Silveira, and Thomas S Paula. 2018. Evaluation of sentence embeddings in downstream and linguistic probing tasks. *arXiv preprint arXiv:1806.06259*.

Radim Řehůřek and Petr Sojka. 2010. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta. ELRA. http://is.muni.cz/publication/884893/en.

C. J. Van Rijsbergen. 1979. *Information Retrieval*, 2nd edition. Butterworth-Heinemann, Newton, MA, USA.

Julie M Robillard, Ian Cleland, Jesse Hoey, and Chris Nugent. 2018. Ethical adoption: A new imperative in the development of technology for dementia. *Alzheimer's & Dementia*, 14(9):1104–1113.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jingjing Xu, Xu Sun, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 979–988, Melbourne, Australia. Association for Computational Linguistics.

Yuanshun Yao, Bimal Viswanath, Jenna Cryan, Haitao Zheng, and Ben Y Zhao. 2017. Automated crowdturfing attacks and defenses in online review systems. In *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, pages 1143–1158. ACM.

Wanrong Zhu, Zhiting Hu, and Eric P. Xing. 2019. Text infilling. *CoRR*, abs/1901.00158.