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

Anonymous

Abstract

In this paper, we present a novel method for measurably adjusting the semantics of text, while preserving its sentiment and fluency, called semantic text exchange. This is different from the usual style transfer where semantics are kept and sentiment is adjusted. We introduce a three-stage pipeline called SMERTI that combines entity replacement, similarity masking, and text infilling. We define our pipeline’s success by its Semantic Text Exchange Score (STES): the ability to preserve the original text’s sentiment and fluency while swapping targeted entities and adjusting semantic content. We propose to use masking (replacement) rate as an adjustable parameter to control the amount of semantic change between input and output. Our experiments show that SMERTI can outperform other models on Yelp reviews, Amazon reviews, and news headlines.

1 Introduction

Sentence embeddings have opened up the potential for many tasks in NLP. With the growth of social media and chatbots, one task that has surfaced with increasing importance is the ability to independently adjust semantics (aspects of the text related to meaning including its content) and style (e.g. its sentiment or personality).

There has been significant research on the task of style transfer (Huang and Belongie, 2017; Logeswaran et al., 2018), with the goal of changing the sentiment or style of text while keeping its semantic content unchanged. However, the alternative where the semantics are adjusted while keeping other elements such as style and sentiment intact, which we call semantic text exchange (STE), has not been investigated to the best of our knowledge. We propose a three-stage pipeline called SMERTI (pronounced ‘smarty’). Combining entity replacement (ER), similarity masking (SM), and text infilling (TI), SMERTI can modify the semantic content of text. Consider the following example, in which 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 original text is of a negative sentiment, but 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 is changed as well.

Our long-term goal is the development of virtual assistants with adjustable socio-emotional personalities. This task arises with importance in the ongoing effort to construct assistive technologies for persons with cognitive disabilities such as Alzheimer’s and related dementias, or autism spectrum disorders. In these situations, adjusting the emotional delivery of text in subtle ways can have a significant effect on the adoption of the technologies (Robillard et al., 2018). It is a significant challenge to transfer style at this level of subtlety, primarily because of the lack of datasets of interactions on specific topics or for specific tasks with consistent emotional personality. Instead, coming at this from the other direction opens up novel avenues for development, as large and varied datasets of personality and sentiment consistent interactions exist, but not confined to specific topics. Therefore, training a model to generate text with high fluency and a particular sentiment, and then adjusting semantics to fit a topic or task, is an important step towards this longer term goal.
To assist in determining the success of the pipeline, we define a new metric called the Semantic Text Exchange Score, or STES, that indicates the ability of the pipeline to generate modified text of the correct semantic content while preserving the fluency and sentiment of the original text.

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: Algorithm 2 from Yao et al. (2017) which we call the Noun WordNet Semantic Text Exchange Model (NWN-STEM), a variation of this algorithm to handle other parts-of-speech which we call the General WordNet Semantic Text Exchange Model (GWN-STEM), and a Word2Vec (Mikolov et al., 2013a,b) model using Gensim (Reháček and Sojka, 2011) which we call the Word2Vec Semantic Text Exchange Model (W2V-STEM).

We illustrate the STE performance of two SMERTI variations on all three datasets, demonstrating outperformance of the baselines and overall pipeline stability, and analyze the results in detail. We investigate relationships between the semantic content, fluency, sentiment, and overall STES by model, dataset, part-of-speech, and amount of text masked and/or replaced. Our contributions can be summarized as follows:

- We define a new task called semantic text exchange with increasing importance in NLP applications, focused on modifying the semantics and content within text while preserving other aspects such as style and sentiment.
- We propose and evaluate a three-stage pipeline for the semantic text exchange task, SMERTI, that can perform multiple-word entity replacement and variable length text filling, and demonstrate its outperformance of the baseline models.
- We define an evaluation metric to evaluate overall performance on semantic text exchange called the Semantic Text Exchange Score (STES) that takes into account various aspects of the resulting text indicative of strong performance.

All code for SMERTI will be released upon publication, and can be found in the supplementary material.

2 Related Work

Our work relies heavily upon word and sentence-level embeddings to compute semantic similarities between pieces of text. Word2Vec allows for the representation of analogies through vector arithmetic. We implement a baseline model (W2V-STEM) for STE using this technique. The Universal Sentence Encoder (USE) (Cer et al., 2018) is designed specifically for encoding sentences, and is trained on a variety of web sources along with the Stanford Natural Language Inference (SNLI) corpus (Bowman et al., 2015). The more recent Flair embeddings (Akbik et al., 2018) utilize embeddings based on architectures such as BERT (Devlin et al., 2018). We decide to use USE for SMERTI as it is designed for transfer learning and shows higher performance on semantic relatedness and textual similarity tasks compared to many other embedding models as evaluated in Perone et al. (2018).

Another area that has seen research is text infilling, which is the ability to fill 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 and can generate multiple words per mask token for more flexible text output. Zhu et al. (2019) approached the text infilling task using a transformer-based architecture. Unlike their text infilling model which fills in random masks, SMERTI is designed to fill in masks guided by semantic similarity. This results in a more natural replacement of specific parts of sentences and fulfillment of the STE task.

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.

Review generation (Hovy, 2016; Lipton et al., 2015; Dong et al., 2017; Yao et al., 2017; Juuti et al., 2018) comes with many forms. Hovy (2016) generates fake reviews from scratch using language models built using graphical models. Lipton et al. (2015), Dong et al. (2017), and Juuti et al. (2018) generate reviews from scratch given auxiliary information (e.g. a chosen sentiment or topic, such as the item category and star rating). Yao et al. (2017) generates reviews using RNNs. Their fake review generation has two components: generation from scratch and review customization.
(Algorithm 2 in Yao et al. (2017)). They define review customization as modifying the originally generated review to fit a new topic or context, such as changing a review for a Japanese-style restaurant to an Italian-style restaurant. They condition on a keyword that identifies the desired topic, and replace similar nouns with other nouns accordingly by calculating lexical similarity using WordNet (Miller, 1995), a lexical database. However, they require a “reference dataset” (which is required to be “on topic”), easy enough for restaurant reviews, but less so for arbitrary conversational agents). Further, as noted by Juuti et al. (2018), the method of Yao et al. (2017) may require reviews, but less so for arbitrary conversational agents). Further, as noted by Juuti et al. (2018), the method of Yao et al. (2017) may replace words independently of context. We implement their review customization algorithm (NWN-STEM) along with a modified version (GWN-STEM) as baseline models for comparison.

### 3 SMERTI

#### 3.1 Overview

The task is to transform a corpus \( C \) of sentences \( S_i \) and associated replacement entities \( RE_i \) : \( C = \{(S_1, RE_1), (S_2, RE_2),..., (S_n, RE_n)\} \) to a modified corpus \( \hat{C} = \{\hat{S}_1, \hat{S}_2,\ldots, \hat{S}_n\} \), where \( \hat{S}_i \) are the original sentences \( S_i \) replaced with \( RE_i \) and overall semantics adjusted. SMERTI consists of the following steps, shown in Figure 1:

1. **Entity Replacement Module (ERM):** Identify which word(s) within the original text are best replaced with the \( RE \). We call these word(s) the Original Entity \((OE)\). We replace \( OE \) in \( S \) with \( RE \). We call this modified sentence \( S' \).

2. **Similarity Masking Module (SMM):** Identify words and/or phrases in \( S' \) similar to the \( OE \) and replace them with a [mask] token. Group adjacent [mask] tokens into a single [mask] token so we can fill in a variable length of text into each. We call this masked sentence \( S'' \).

3. **Text Infilling Module (TIM):** Fill in the [mask] tokens with text that better suits the \( RE \) being in the now altered text. This will modify the semantics in the rest of the text. This final output sentence 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), a component of the Stanford CoreNLP pipeline (Manning et al., 2014).

### 3.2.1 Stanford Parser

The Stanford Parser is a constituency parser that determines the grammatical structure of sentences, including phrases (groups of words that go together), 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 \( RE \). Then, feeding our input text \( S \) through the parser, we can obtain similar parse-trees for each sentence in \( S \). Iterating through these parse-trees and all their sub-trees, we can obtain a list of leaves (where leaves under a single label are concatenated into a single string) that are equal or similar to any of the \( RE \) constituent tags. This generates a list of entities which are identified as 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) discussed below.

### 3.2.2 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), and we choose the transformer model as sentence embeddings trained on this model take context into account. Therefore, the exact same word or phrase in a different context will have a different embedding depending on surrounding words.

We compute the semantic similarity between two embedding vectors \( u \) and \( v \): \( sim(u, v) \), using the angular (cosine) distance, defined as:

\[
\cos(\theta_{u,v}) = \frac{(u \cdot v)}{(||u|| ||v||)}
\]

such that...
sim(u, v) = 1 − \frac{1}{\pi}arccos(cos(\theta_{u,v})). The results are in the interval [0, 1], with higher values representing greater similarity.

Using USE and the above similarity equation, we can identify the words or phrases within the input text S which are most similar to RE. To assist with this identification, we use the Stanford Parser as described above to obtain a list of candidate entities. In the rare case that the Stanford Parser generates an empty list, 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 overall entity replacement process is in Figure 2. Two parse-trees are shown: for RE (a) and S (b) and (d). Figure 2(c) is a semantic similarity heat-map generated from the USE embedding vectors of the determined candidate OEs and RE, where values on the heat map are semantic similarity scores in the range [0, 1].

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

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

<table>
<thead>
<tr>
<th>MRT</th>
<th>ST</th>
<th>Masked Outputs</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.2</td>
<td>0.4</td>
<td>i love this restaurant ? [mask] are comfortable and the [mask] is great !</td>
</tr>
<tr>
<td>0.4</td>
<td>0.3</td>
<td>i love this restaurant ? [mask] are [mask] and [mask] is great !</td>
</tr>
<tr>
<td>0.6</td>
<td>0.2</td>
<td>i [mask] this restaurant ? [mask] are [mask] and [mask] is great !</td>
</tr>
<tr>
<td>0.8</td>
<td>0.1</td>
<td>[mask] restaurant ? [mask] and [mask] is great !</td>
</tr>
</tbody>
</table>

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

3.3 Similarity Masking Module (SMM)

The next step is to mask words similar to the replaced entity (OE) to generate S″ using USE. We look at semantic similarities between every word in S and OE. We also look at semantic similarities between the OE and the candidate entities list generated in the previous ERM step to broaden the range of phrases our module can mask. We ignore RE, OE itself, and any entities or phrases containing OE (for example, ‘this hotel’).

After determining the words similar to the OE (discussed below), we replace each of them with a [mask] token. Next, we replace any [mask] tokens adjacent to each other with a single [mask] token.

We set a base similarity threshold (ST) that selects a subset of words to mask. We then 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. 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 would like to generate text very dissimilar to the original, and may be desired in cases such as creating a lively chatbot to entertain people. Certainly, a chatbot that repeats similar statements over and over would be very boring. On the other hand, a low MRT means the user would like to generate text similar to the original, and may be desired in cases such as text recovery or grammar correction, where it is desired for the overall semantics to remain similar to the original with slight modifications. The MRT also affects sentiment and fluency, as we show in Section 6.4.

Note that there are certain cases where two or more of the outputs for different MRT may be the same. This occurs when a valid similarity threshold cannot be found that masks a larger portion of the sentence without going over MRT.
3.4 Text Infilling Module (TIM)

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

3.4.1 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., 2014). 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 \( \tilde{h}_s \) is all encoder states: \( \text{score}(h_t, \tilde{h}_s) = h_T^T \tilde{h}_s \).

3.4.2 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:

\[
\text{Attention}(Q, K, V) = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right) V
\]

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 such as books and electronics, with duplicate reviews removed. The Yelp dataset includes over six million user reviews on businesses including restaurants and hotels. These two datasets include text that is strong in sentiment, which is perfect for testing the ability of our pipeline to alter semantics while preserving sentiment. The news headlines dataset from Kaggle contains approximately 200,000 news headlines from 2012 to 2018 obtained from HuffPost (Misra, 2018).

For each dataset, we filter the text to obtain reviews which are English, do not contain hyperlinks and other obvious noise, and lines which are less than 20 words long. This is because we found that many reviews longer than twenty words ramble on and describe unnecessary aspects of the product or business, and are too verbose and long-winded for our purposes. Rather than filtering by individual sentences, we keep each review in its entirety, as we do not wish to limit our pipeline to generating single sentences only. We preprocess the text to ensure everything is lowercase, rare/duplicate punctuation and space is removed.

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 of our training and testing sets, we include an equal number of positive and negative reviews, and half as many neutral reviews, all of which are selected randomly from our filtered reviews. This is because neutral reviews only occupy one out of five stars compared to positive and negative which occupy two each, so we cut its proportion in half. 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 our training and testing data. 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, for each of the training and testing sets. For Amazon and Yelp, we ensure one-third of the reviews for each sentiment is masked with one of these masking rates. These masked reviews serve as the inputs to our TIM models, while the original reviews serve as the ground-truth. Setting up our data in this way allows our TIM models to learn relationships
between masked words and relationships between masked and unmasked words within each line.

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

For the transformer, we use scaled dot-product attention and the same hyperparameters used within the original attention paper by 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 Benchmarks

We implement three models to benchmark against.

First is NWN-STEM (Algorithm 2 from Yao et al. (2017)). We use the training sets as the respective “reference review sets” to extract similar nouns to the RE (using $MIN_{sim} = 0.1$), where the RE acts as the “topic keyword” as indicated in their algorithm. Then, we replace nouns in the text similar to the RE with the noun extracted from the associated reference review set that is most similar to the noun being replaced.

Secondly, we modify NWN-STEM to work for verbs and adjectives, and call this GWN-STEM. From the reference review sets, we extract similar nouns, verbs, and adjectives to the RE (using $MIN_{sim} = 0.1$), 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 the most similar noun, verb, or adjective, respectively, extracted from the associated reference review set.

Lastly, we implement W2V-STEM. For single word REs, we train uni-gram Word2Vec models using Gensim, and for phrases, we train four-gram models (words are grouped into phrases up to four words long), where the models are trained on each dataset’s training set. We use cosine similarity to determine the most similar word/phrase in the input text to the RE, which becomes the replaced OE. Then, 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 the nearest neighbor to $w'_i$. We can use similarity thresholds to adjust replacement rates (RR) (similar to SMERTI’s masking rates) to produce outputs satisfying various RR thresholds (RRT).

5 Evaluation

5.1 Evaluation Setup

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

For each evaluation RE, we choose one-hundred lines from the associated testing set that does not already contain RE. We choose lines with at least five words, as many with less carry little semantic meaning and are unsuitable for our evaluation purposes (e.g. ‘Great!’, ‘It is okay’). For Amazon and Yelp, we choose 50 positive and 50 negative lines per RE. We then 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, where the positive-negative split for each set is half-and-half for Amazon and Yelp. Our final results are averaged metrics over these three sets per dataset per POS.

For SMERTI-Transformer, SMERTI-RNN, and W2V-STEM, we generate four outputs per evaluation line for MRT/RRT of 20%, 40%, 60%, and 80%, which represent upper-bounds on the percentage of the input text that can be masked and/or replaced. Note that NWN-STEM can only evaluate on nouns and GWN-STEM can only evaluate on nouns and verbs and their maximum re-

---

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

3 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)
placement rates are limited. We select MINsim values of 0.075 and 0 for nouns and 0.1 and 0 for verbs, as these result in average output 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, with slight modifications from their word-level formula to character-level ($SLOR_c$). This uses Flair perplexity\(^7\) values from a language model trained on the One Billion Words corpus (Chelba et al., 2013):

\[
SLOR_c(S) = \frac{1}{|S|} \left( \ln(p_M(S)) - \frac{\ln(\prod_{w \in S} p_M(w))}{\sum_{w \in S} |w|} \right)
\]

\[
= -\ln(PPL_S) + \sum_{w \in S} |w| \ln(PPL_W) - \sum_{w \in S} |w|
\]

where $|S|$ is the character length of the input text $S$, $p_M(S)$ is the probability of $S$ under the language model $M$, $|w|$ is the length of the word $w$ within $S$, and $p_M(w)$ is the probability of $w$ under $M$. $PPL_S$ is the character-level perplexity of $S$, and $PPL_W$ is the character-level perplexity of the individual word $w$. SLOR (from hereon we refer to character-level SLOR as simply SLOR) includes perplexity and other aspects of text fluency such as grammaticality. Higher SLOR values represent higher fluency. We rescale the SLOR values in our results to the interval [0, 1] by fitting and normalizing a Gaussian distribution (Appendix F).

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.\(^8\)

Content Similarity Score (CSS) ranges from 0 to 1 and determines the semantic textual similarity between the 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 on semantic similarity tasks as previously mentioned. Note that a model that results in output with lower average CSS than the input clearly fails at STE since it alters the text to be more different from the $RE$ rather than more similar.

5.3 Semantic Text Exchange Score (STES)

We come up with a single score to evaluate the overall performance of a model on STE that combines the key evaluation metrics. It uses the harmonic mean, similar to the F-score (Chinchor, 1992; Rijsbergen, 1979), and we call it the Semantic Text Exchange Score (STES):

\[
STES = \frac{3 \times A \times B \times C}{A \times B + A \times C + B \times C}
\]

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 F1 score, STES penalizes models which perform very poorly in one or more of the metrics, and favors balanced models which achieve strong results in all three. Clearly, all three of CSS, SLOR, and SPA are important, and a model which performs very poorly in even one performs poorly overall. Further, STES being a single score makes it easy to understand and use to compare models.

5.4 Evaluation Results

Figure 3 a) is a table showing overall average results per model, and Figure 3 b) is a corresponding graph.\(^9\) Table 2 shows outputs from each model for a Yelp evaluation example.\(^{10}\)

![Table 3](image)

\(\text{Figure 3: Overall average results per model, in (a) table format; (b) graph format.}\)

\(^6\)See Appendix C for explanations

\(^7\)See Appendix E for Flair perplexity details

\(^8\)See Appendix E for VADER details

\(^9\)See Appendix G for tables and graphs of detailed results broken down by POS, dataset, and MRT/RRT

\(^{10}\)See Appendix H for many more examples for various POS and datasets
Input: great food, large portions! my family and i really enjoyed our saturday morning breakfast.
Replacement entity: pizza

<table>
<thead>
<tr>
<th>MRT/RRT</th>
<th>Generated Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>20% SMERTI-Transformer</td>
<td>great pizza, large slices! my family and i really enjoyed our saturday morning lunch.</td>
</tr>
<tr>
<td>40%,60% SMERTI-RNN</td>
<td>great pizza, large slices! service was terrific and i really enjoyed our saturday morning lunch.</td>
</tr>
<tr>
<td>80% W2V-STEM</td>
<td>great pizza, chewy crust! nice ambiance and i really enjoyed it.</td>
</tr>
<tr>
<td>80% GWN-STEM/NWN-STEM</td>
<td>great pizza, large delivery! my family and i really enjoyed our saturday morning place.</td>
</tr>
</tbody>
</table>

Table 2: Generated output text by model for various masking rates on a Yelp evaluation example.

As observed from Table 2 and Appendix H, SMERTI is able to generate high quality output text that is similar to the RE while flowing better than other models’ outputs. It can even replace entire phrases and sentences due to the variable length infilling capabilities of the two TIMs. Note that for nouns, the outputs from GWN-STEM and NWN-STEM are equivalent.11

6 Analysis

6.1 Performance by Model

As seen in Figure 3, 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, likely because they modify little of the input text, and mainly modify only verbs and especially nouns (which are less likely to carry sentiment). However, they achieve by far the lowest CSS, likely in part due to this limited text replacement, and because their algorithm does not guarantee the RE will even appear in the output (as evident from Table 2). Overall, the WordNet models are not very effective at STE.

W2V-STEM achieves the lowest SLOR, especially for higher RRT, as supported by the examples in Table 2 and Appendix H. W2V-STEM and the WordNet models consistently output text that is grammatically incorrect and flows poorly.12 In many examples, certain words are 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.13 The SMERTI variations achieve the highest TTR, while W2V-STEM and NWN-STEM the lowest, especially for higher MRT/RRT.

It is worth noting that while W2V-STEM achieves lower CSS than SMERTI, it performs quite 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 similar to the RE multiple times.

6.2 Performance By Part-of-Speech (POS)

As seen from Appendix G, SPA is highest for nouns because they are typically objective and carry little sentiment, and lowest for adjectives, likely because they typically carry the most.14 SLOR is relatively consistent, with slightly lower values for adjectives and higher for phrases. Adjectives typically carry less semantic meaning, and hence the models likely have more trouble figuring out how best to infill/replace the text. Phrases typically carry more (since they consist of multiple words), and may result in the replacement of other entire phrases, leading to higher fluency.

---

11 See Appendix C for explanations

12 See Appendix I for more disadvantages of the Word2Vec and WordNet models compared to SMERTI

13 See Appendix J for TTR values

14 See Appendix K for more on SPA vs. POS
CSS is typically highest for phrases then nouns, and lowest for verbs then adjectives. This is also likely due to phrases and nouns carrying more semantic meaning, making it easier to generate text semantically similar to them. Verbs and especially adjectives are less semantically significant, making it harder to generate similar text.

As expected, STES follows this same pattern, being highest for phrases then nouns, and lowest for verbs then adjectives. Hence, it appears that the STE task is, overall, more effective on nouns and phrases than verbs and adjectives.

6.3 Performance By Dataset

As seen in Appendix G, SPA values are lowest for the news headlines dataset. Amazon and Yelp reviews naturally carry stronger sentiment, likely making it easier to generate text with the same/similar sentiment.

SLOR values (including for input text) appear to be slightly lower for Yelp reviews. This may be due to many reasons, such as more typos and grammatical mistakes within the original reviews, more emojis used, and so forth.

CSS values are relatively similar, with slightly higher values for news headlines. This may be due to news headlines 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, will influence the ability of STE models to generate fluent text. Lastly, stronger content similarity appears easier to achieve on text that is initially shorter and carries more semantic meaning.

6.4 Performance By MRT/RRT

From Appendix G and L, it can be seen that as the MRT/RRT increases, CSS increases, while SPA and SLOR decrease. These relationships are strong as supported by the Pearson correlation values of 0.7264, -0.9171, and -0.6518, respectively. When a model is given the ability to alter more text, it has the opportunity to replace more text related to sentiment, while producing more of semantic similarity to the $RE$.

Further, the models generate more of the text themselves, becoming less similar to the human-written input, resulting in lower fluency. To demonstrate this, we look at average BLEU (Papineni et al., 2002) scores against MRT/RRT. BLEU generally indicates how close two pieces of text are in terms of content and structure, with higher values indicating greater similarity. As expected, BLEU decreases as MRT/RRT increases, and this relationship is extremely strong as supported by the Pearson correlation value of -0.9848.

SMERTI really shines on higher thresholds, as its performance exceeds W2V-STEM on nearly every metric for 60% and 80% MRT/RRT. This is evident from examples in Table 2 and Appendix H. SMERTI has higher SLOR across the board, and this difference increases with MRT/RRT.

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, proposed and demonstrated that our pipeline SMERTI performs well on this task, and proposed an STES metric for evaluating overall performance on this task. Both SMERTI-Transformer and SMERTI-RNN outperformed other models and achieved strong STES scores, and were shown to be the most balanced models overall across all the datasets, parts-of-speech, and rate thresholds.

Additionally, we showed there is a trade-off between semantic exchange against fluency and sentiment preservation, and that this can be controlled by the masking rate or replacement rate threshold.

Potential directions for future work include introducing specific methods within the pipeline to control sentiment in hopes of increasing sentiment preservation further, and fine-tuning SMERTI for preservation of persona or personality. Further, experimenting with other potential text infilling models and architectures such as fine-tuning BERT (Devlin et al., 2018) or Transformer XL (Dai et al., 2019) for longer contexts, and improving the text infilling module to generate more words for higher masking rates, may be potential areas of exploration to improve SMERTI further.

---

See Appendix L for line graphs of CSS, SPA, SLOR, and BLEU against MRT/RRT.

See Appendix M for explanation of Pearson correlation.

---

See Appendix N for BLEU scores.
References


Lajanugen Logeswaran, Honglak Lee, and Samy Bengio. 2018. Content preserving text generation


Radim Rehřek and Petr Sojka. 2011. Gensimstatistical semantics in python. statistical semantics; gensim; Python; LDA; SVD.


