A Modular Architecture for Unsupervised Sarcasm Generation

Abhijit Mishra, Tarun Tater, Karthik Sankaranarayanan
IBM Research
{abhijimi,ttater24,kartsank}@in.ibm.com

Abstract

In this paper, we propose a novel framework for sarcasm generation; the system takes a literal *negative opinion* as input and translates it into a sarcastic version. Our framework does not require any paired data for training. Sarcasm emanates from *context-incongruity* which becomes apparent as the sentence unfolds. Our framework introduces incongruity into the literal input version through modules that: (a) filter factual content from the input opinion, (b) retrieve incongruous phrases related to the filtered facts and (c) synthesize sarcastic text from the filtered and incongruous phrases. The framework employs reinforced neural sequence to sequence learning and information retrieval and is trained only using unlabeled non-sarcastic and sarcastic opinions. Since no labeled dataset exists for such a task, for evaluation, we manually prepare a benchmark dataset containing literal opinions and their sarcastic paraphrases. Qualitative and quantitative performance analyses on the data reveal our system's superiority over baselines, built using known unsupervised statistical and neural machine translation and style transfer techniques.

1 Introduction

Sarcasm[1] is an intensive, ironic construct that is intended to express contempt or ridicule. It is often linked with intelligence, creativity, and wit, and therefore empowering machines to generate sarcasm is in line with the key goals of *Strong AI*[2]. From the perspective of Natural Language Generation (NLG), sarcasm generation remains an important problem and can prove useful in downstream applications such as conversation systems, recommenders, and online content generators. For instance, in a conversational setting, a more natural and intriguing form of conversation between humans and machines could happen if machines can intermittently generate sarcastic responses, like their human counterparts.

Over the years, a lot of research and development efforts have gone into the problem of detecting sarcasm in text, which aims to classify whether a given text contains sarcasm or not (Joshi et al. (2017b) provide an overview). However, systems for generation of sarcasm have been elusive. This is probably due to the fact that in sarcasm generation both *selection of contents* for sarcastic opinion generation and *surface realization of contents* in natural language form are highly nuanced.

In the broader area of style transformation of texts, most of the existing works have focused narrowly on transformations at lexical and syntax levels, *i.e.,* text simplification (Siddharthan, 2014), text formalization (Jain et al., 2018), sentiment style transfer (Shen et al., 2017; Xu et al., 2018), sentiment flipping (Li et al., 2018) and understanding humor (West and Horvitz, 2019). However, very little work has been done ((Piwek, 2003),(Hovy, 1987)) on incorporating *pragmatics* into generation tasks such as sarcasm. Sarcasm generation offers a rich playground to study this challenge and push the state-of-the-art in text transformation. Moreover, being a pragmatic task, sarcasm construction offers diverse ways to convey the same intent, based on cultural, social and demographic backgrounds. Hence, a supervised treatment of sarcasm generation using paired labeled data (such as parallel sentences) will be highly restrictive. This further motivates the need for exploring unsupervised approaches as the one we propose in this paper.

We make the first attempt towards automatic sarcasm generation where the generation is conditioned on a literal input sentence. For example, the literal opinion *"I hate it when my bus is late."*

[1] https://www.thefreedictionary.com/Sarcasm

[2] https://en.wikipedia.org/wiki/Artificial_general_intelligence

Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, pages 6144–6154, Hong Kong, China, November 3–7, 2019. ©2019 Association for Computational Linguistics

should be transformed into *"Absolutely love waiting for the bus"*. As sarcasm conveys a negative sentiment, our system expects a negative sentiment opinion as input. Out of various possible theories proposed to explain the phenomenon of sarcasm construction (Joshi et al., 2017b), our framework relies on the theory of *context incongruity* (Campbell and Katz, 2012). Context incongruity is prevalent in textual sarcasm (Riloff et al., 2013; Joshi et al., 2015b). The theory presents sarcasm as a contrast between positive sentiment context (*e.g., absolutely loved it*) and negative situational context (*e.g., my bus is late*).

In our framework, translation of literal sentences to sarcastic ones happens in four stages *viz.*, (1) **Sentiment Neutralization**, during which sentiment-bearing words and phrases are filtered from the input, (2) **Positive Sentiment Induction**, where the neutralized input is translated into phrases conveying a strong positive sentiment, (3) **Negative Situation Retrieval**, during which a negative situation related to the input is retrieved, and (4) **Sarcasm Synthesis**, where appropriate sarcastic constructs are formed from the positive sentiment and negative situation phrases gathered in the first three stages.

Training and development of these modules require only three unpaired corpora of positive, negative, and sarcastic opinions. For evaluating the system, we manually prepare a small benchmark dataset which contains a set of literal opinions and their corresponding sarcastic paraphrases. Quantitative evaluation of our system is done using popular translation-evaluation metrics, and document similarity measurement metrics. For qualitative evaluation, we consider the human judgment of sarcastic intensity, fluency, and adequacy of the generated sentences. As baselines, we consider some of our simplistic model variants and existing systems for unsupervised machine translation and style transfer. Our overall observation is that our system often generates sarcasm of better quality than the baselines. The code, data, and resources are available at https://github.com/TarunTater/sarcasm_generation.

2 Challenges in Sarcasm Generation

Generation of sarcasm, unlike other language generation tasks, is highly nuanced. If we reconsider the example in the introductory section, the output sentence is sarcastic as it presents an unusual situation where the opinion holder has liked the rather boring act of *waiting for a bus*. The unusualness (and hence, the sarcasm) arises from two implicitly opposing (*incongruous*) contexts: *love* and *waiting for the bus*. Such a form of sarcasm, based on the *context incongruity* theory (Campbell and Katz, 2012), is more common in text than other forms such as prepositional, embedded or illocutionary sarcasm (Camp, 2012). For any textual sarcasm generator, figuring out contextually incongruous phrases will be as difficult as generating a fluent sentence. Moreover, most of the existing language generators are known to work on large scale literal/non-sarcastic texts (*e.g.,* language models trained on Wikipedia articles), and are agnostic of the possible collocations of contextually incongruous phrases (Joshi et al., 2017a). We try to overcome these challenges through our modular system design, discussed as follows.

3 System Architecture

The overall system architecture is presented in Figure 1. For development of the modules three corpora are needed: (a) a corpus of positive sentiment sentences ($\boldsymbol{\mathcal{P}}$), (b) a corpus of negative sentiment sentences ($\boldsymbol{\mathcal{N}}$), and (c) a corpus of sarcastic sentences ($\boldsymbol{\mathcal{S}}$). The framework performs transformation of literal text into sarcastic ones in four stages as given below:

3.1 Sentiment Neutralization

The neutralization module is designed to filter sentiment information out from the input text. For example, the input *hate when my bus is late* should be filtered to produce a neutral statement like *bus is late*. Neutralization is performed by a neural sentiment classification module. Each word in the input sentence of length N ($\boldsymbol{x} = \{x_1, x_2, ..., x_N\}$) (in *one-hot* representation, padded wherever necessary) is transformed into K-dimensional embedding. The embeddings are then passed through a layer of recurrent units such as Long Short Term Memory (LSTM)(Hochreiter and Schmidhuber, 1997). The output of the LSTM layers are then sent to a *self-attention* layer before passing through a `softmax` based classifier. The classifier is trained with the supervision from sentiment positive/negative labels using corpora $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{N}}$.

During testing, for a given input of length N, the self attention vector $\boldsymbol{\alpha} = \alpha_1, \alpha_2, ..., \alpha_N$ is first extracted (details skipped for brevity, refer Xu

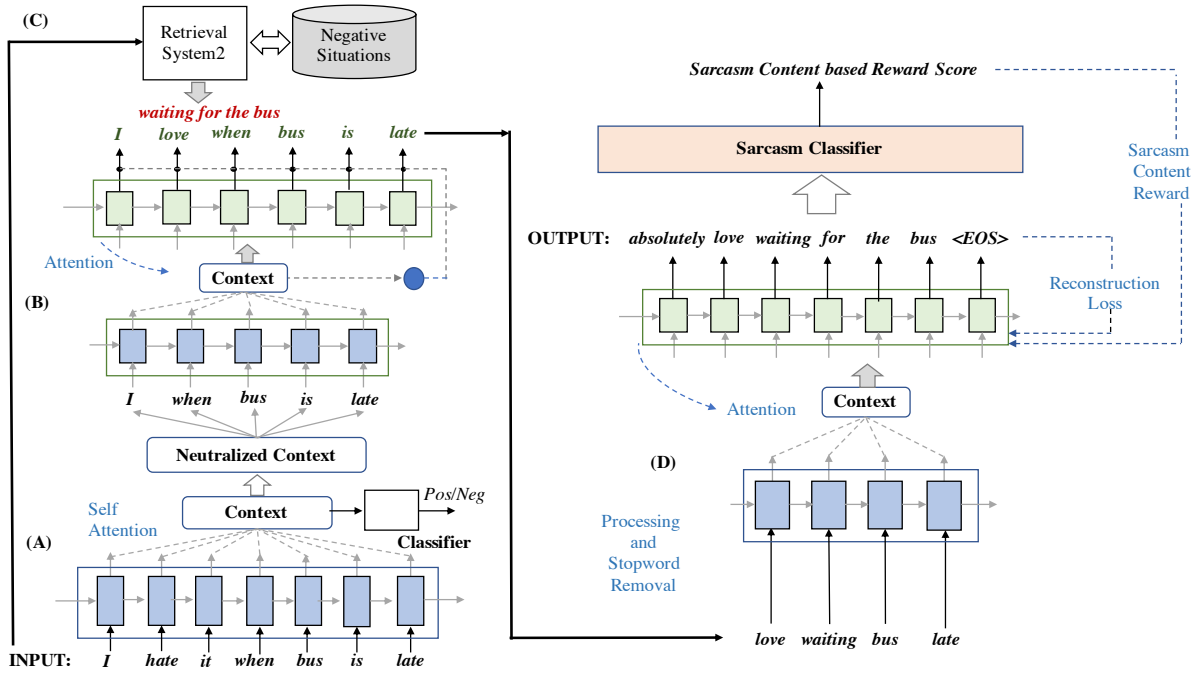

Figure 1: System Architecture. Input to the system are literal sentences. The blue and green boxed represent layers of embedding and LSTMs. The components are (A) Sentiment Neutralizer (B) Positive Sentiment Inducer (C) Negative Situation Retriever , and (D) Sarcasm Synthesizer.

et al. (2018)). We then inverse and discretize the self attention vector as follows:

$$\hat{\alpha_i} = \begin{cases} 0, & \text{if } \alpha_i > \mu + 2\sigma \\ 1, & \text{otherwise} \end{cases} \quad (1)$$

where α_i is the attention weight for the i^{th} word, μ and σ are the mean and standard deviation for the attention vector. For each word in the input, if the discretized attention weight is 0, it is filtered out.

The motivation behind such a design is that if the classifier is trained well, the attention weights will represent the contribution of each word to the overall sentiment decision (Xu et al., 2018). Sentiment bearing words will receive higher attention whereas neutral words will have lower attention weights. It is worth noting that neutralization can be done in several other ways such as filtering based on a sentiment lexicon. However, such operations would require additional resources such as sentiment dictionary, sense disambiguation tools, whereas the neural classification based filtering can only work with binary sentiment labeled data. Also, for computing word-level sentiment contributions, recent techniques (such as gradient-based methods (Sundararajan et al., 2017)) can be used. For simplicity, we use attention based filtering.

3.2 Positive Sentiment Induction

Once the neutralized output is extracted, it is passed to the positive sentiment induction module which transforms it into a positive sentiment sentence. For this, we use a traditional sequence to sequence pipeline (Bahdanau et al., 2014) with attention and copy mechanisms (Gulcehre et al., 2016). The input is a set of words coming from the neutralization module. These are transformed into embeddings and are then encoded with the help of LSTM layers. The decoder attends over the encoded output and produces the output tokens based on the attended vector and the previously generated tokens. This is a standard technique, typically used in neural machine translation.

As the output from the system is expected to be positive in sentiment, for training the framework, we use only a set of positive sentences from $\mathcal{P}$. Each sentence in the data is filtered using the neutralization module. The filtered version, and the original positive sentence are used as $\langle source, target \rangle$ pairs.

3.3 Negative Situation Retrieval

Negative situations present in sarcastic opinions are typically extrinsic and are loosely related to the semantics of its literal version. Hence, a sequence to sequence module analogous to Section

5-grams: getting up for school facts, getting yelled at by people, trying to schedule my classes, feeling like every single person, walking to class in pouring, making people who already hate, working on my last day, spending countless hours at doctors, getting overdraft statements in mail
4-grams: talking about world politics, stuck in a generation, sitting in class wondering, canceled at short notice, distancing myself from certain, wipe my own tears
3-grams: born not breathing, paid to sleep, scared those faces, taking a shower, starting your monday, accused of everything, worrying about someone, fight jealousy arguments, license to trill, awarded literature prize
2-grams: scratching itchy, looking chair, getting hiv, shot first, collecting death, lost respect
1-gram: canceled, sleeping, trying, buying, stapling

Table 1: Example negative situations extracted using bootstrapping technique (Riloff et al., 2013)

3.2 may not be very useful. Moreover, for sarcasm generation, for a certain topic, it is safe to assume that there can be a finite set of negative situations. From this set appropriate situation phrases can be *"retrieved"* depending on the given input. Thus, finding out appropriate negative situations boils down to two sub-problems of (a) preparing a finite set of negative situations, and (b) setting up the negative situation retrieval process. We discuss each of these two steps below.

3.3.1 Building *Negative Situation* Gazetteer

This is a one-time process and is done using the unsupervised bootstrapping technique similar to Riloff et al. (2013). For each sentence in the sarcasm corpus $\boldsymbol{\mathcal{S}}$, a candidate negative situation phrase is extracted. A candidate negative situation phrase is a word n-gram ($n \leq 5$) that follows a positive sentiment phrase in a sarcastic sentence [3]. After the candidates for a positive phrase are obtained, their Part of Speech tags are extracted with the help of a POS tagger. Specific patterns of n-gram are then obtained using the POS tags. This ensures that the phrases extracted are mostly *verb phrases*, *noun-phrases*, and *to-infinitive* verb phrases that describe situations. In our setting we use 30 predefined POS n-gram patterns following Riloff et al. (2013).

Once the candidate negative situation phrases are extracted, they are filtered based on a scoring function as given below:

$$score_i = \frac{\#ns_i \; in \; \boldsymbol{\mathcal{S}}}{\#ns_i \; in \; \boldsymbol{\mathcal{S}}, \boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{N}}} \qquad (2)$$

where ns_i is the i^{th} negative situation extracted for a certain positive phrase.

The scoring function returns a real value indicating the exclusiveness of the negative situation w.r.t the sarcastic sentences. If the score exceeds a threshold (*i.e.,* $p > 0.5$), the candidate phrase is added to the gazetteer. Once all the possible negative situation phrases are extracted, each phrase is used to extract more positive sentiment phrases similarly as above. This process of positive phrase and negative situation extraction is repeated until no new phrases are found. Table 1 shows some example negative situation phrases extracted from our dataset.

3.3.2 Retrieval Process

The idea is to find negative situations relevant to the input sentence. We implement an information retrieval system based on PyLucene. All the negative situations from the gazetteer (Sec. 3.3.1) are first indexed. The input sentence is considered as the query for which the most relevant negative situation is retrieved from the indexed list. The factors involved in PyLucene's retrieval algorithm include *tf-idf*, number of matching terms in the query sentence and the retrieved sentence, and importance measure of a term according to the total number of terms in the search.

Once the positive sentiment and negative situations are generated for the input sentence, they undergo a post-processing step where stopwords and redundant words are removed and given as input to the sarcasm synthesis module.

3.4 Sarcasm Synthesis

The sarcasm synthesis module is a sequence-to-sequence network that expects a set of keywords related to positive sentiment and negative situation phrases. For training this module, the sarcasm corpus $\boldsymbol{\mathcal{S}}$ is used. To prepare the input, we implement a keyword extraction technique based on POS tagging. Sentences in $\boldsymbol{\mathcal{S}}$ are POS-tagged and then stopwords are removed, and then based on the POS tags noun, verb, adjective and adverbs are retained. This way, the input keywords to the system would somewhat be similar to the keywords expected in real time scenario.

[3] the word "love" is considered as the seed positive sentiment phrase to begin the bootstrapping procedure

The module follows an encode-attend-decode style architecture like the positive sentiment induction module, but with a different learning objective. Keywords in the input (in *one-hot* representation, padded wherever necessary) are transformed into a sequence of embeddings and then encoded by layers of LSTMs, which produces a hidden representations for each input word. The decoder, consisting of a single layer of LSTMs stacked on the top of a decoder embedding layer, attend over the encoded hidden representations and generate target words. In general, for T training instances of keywords and sarcastic texts, $\{x^i, y^i\}_{i=1}^{T}$, the training objective is to maximize the likelihood of a target y^i given x^i, which is similar to minimizing the *cross entropy* between the target distribution and the predicted output distribution. For training the neural network, the cross-entropy loss is back propagated. In other words, the gradient of the negative cross-entropy loss is considered to update the model parameters. The gradient is given by:

$$\nabla_\theta L(\theta) = \nabla_\theta \sum_{i=1}^{T} y^i log\ P_{M_\theta}(\hat{y^i}|x^i) \quad (3)$$

where L is the loss function and M_θ is the translation system with parameter θ. $\hat{y^i}$ is the predicted sentence. In our setting, where the input is not a proper sentence, the problem with the above objective is that it does not strongly enforce the decoder to learn and produce sarcastic output. We speculate that minimizing the token-level cross entropy loss in Eq. 3 may help produce an output that is grammatically correct but not sarcastic enough. For instance, the decoder may incur insignificant cross-entropy loss after generating a sentence like *absolutely loved it*, as this sentence has considerable overlap with the reference sarcastic text that provides supervision.

One possible idea to tackle such problems is to employ a sarcasm scorer that can determine the sarcasm content in the generated output, and use the scores given by the sarcasm scorer for better training of the generator. However, the sarcasm scorer may be external to the sequence-to-sequence pipeline, and the scoring function may not be differentiable with respect to the model M_θ. For this, we apply reinforcement learning which considers sarcasm content score as a form of *reward* and use it to fine-tune the learning process. For learning, the *policy gradient theorem* (Williams, 1992) is used. The system is trained under a modified learning objective *i.e.,* to maximize the expected reward score for a set of produced candidate sentences. The generator, operating with a policy of $P_{M_\theta}(\hat{y^i}|x^i)$, producing an output $\hat{y^i}$ with an *expected* reward score computed using a scorer, will thus have the following gradient:

$$\begin{aligned} \nabla_\theta RL(\theta) &= \nabla_\theta \mathbb{E}_{\hat{y_i} \sim P_{M_\theta}(\hat{y^i}|x^i)}[R(\hat{y^i})] \\ &= \mathbb{E}[R(\hat{y^i}) \nabla_\theta log(P_{M_\theta}(\hat{y^i}|x^i))] \end{aligned} \quad (4)$$

where RL is the modified learning objective which has to be maximized and R is a reward function that is computed using an external scorer. In practice, the expected reward is computed by (a) sampling candidate outputs from the policy P_{M_θ}, (b) computing the reward score for each candidate and (c) averaging the rewards so obtained. In typical RL settings, the learner is typically initialized to a random policy distribution. However, in our case, since some supervision is already available in the form of target sarcastic sentences, we pretrain the model with the loss minimization objective given in Eq. 3 and then fine-tune the model based on the policy gradient scheme following Eq. 4. Thus, the learner gets initialized with a better policy distribution.

For reward calculation, we consider the confidence score of a sarcasm classifier (probability of being sarcastic) trained using $\boldsymbol{\mathcal{S}}$ as positive examples and $\boldsymbol{\mathcal{P}}$ and $\boldsymbol{\mathcal{N}}$ taken as negative examples. For our setting, the classifier is analogous to the classifier used for neutralization. The classifier is based on embedding, LSTM and softmax layers.

Since the input to the system is a list of words, it may seem that the sarcasm synthesis module may not require sequence to sequence learning, and a much simpler approach like bag-of-words to sequence generation could have been used. However, note that the input to the generator is obtained after dropping words during neutralization and later appending the negative situation phrase. The sequentiality is, thus, not completely lost. This makes sequence to sequence model an intuitive choice.

We now explain our experimental setup.

4 Experiment Setup

4.1 Datasets

As stated earlier, our system does not rely on any paired data for training. It requires three corpora of positive sentences, negative sentences, and sarcastic sentences collected independently.

For positive and negative sentiment corpora $\mathcal{P}$ and $\mathcal{N}$, we considered short sentences/snippets from the following well-known sources such as (a) Stanford Sentiment Treebank Dataset, (b) Amazon Product Reviews, (c) Yelp Reviews (d) Sentiment 140 dataset (See Kotzias et al. (2015) for sources). The above datasets primarily contain tweets and short snippets. Tweets are normalized by removing hashtags, usernames, and performing spell checking and lexical normalization using NLTK (Loper and Bird, 2002). We then filtered out sentences with more than 30 words. Approximately $50,000$ sentences from each category are retained. Then, based on the vocabulary overlap with our sarcasm corpus $\mathcal{S}$, $47,827$ sentences are finally retained from each category (total number of instances is 95654).

For the unlabelled sarcasm corpus $\mathcal{S}$, we relied on popular datasets used for sarcasm detection tasks such as the ones by Ghosh and Veale (2016), Riloff et al. (2013), and the Reddit Sarcasm Corpus[4]. Sentences are denoised, spell corrected and normalized. Average sentence length is kept as 30 words. A total number of $306,141$ sentences are thus collected. A common vocabulary of size $20,000$ is extracted (based on frequency) for all the modules from the three corpora. Each corpus is divided into a train-valid-test split of 80%-10%-10%.

4.1.1 Benchmark Dataset for Evaluation

Since no dataset containing paired examples of literal and sarcastic utterances are available, we created a small test-set for evaluating our system. From the test split of the sarcasm corpus $\mathcal{S}$, 250 sentences on diverse topics are selected and are manually translated into literal versions by two linguists. From this, only 203 sentences could be selected by the linguists who mutually decided whether the sentences were sarcastic enough to keep in the test dataset or not.

[4]https://www.kaggle.com/danofer/sarcasm

4.2 Model Configuration

For the neutralization module, the embedding dimension size is set to 128, two layers of LSTMs of hidden dimension of 200 are used. The classifier trains for 10 epochs with a batch size of 32, and achieves a validation accuracy of 96% and training accuracy of 98%.

For positive sentiment induction module, the embedding dimensions for both encoder and decoder are set to 500. Both the encoder and decoder have only one layer of LSTM, with a hidden dimension of 500. The module is built on top of the OpenNMT (Klein et al., 2017) framework. Training happens in $100,000$ iterations and the batch size is set to 64. The positive sentiment induction module, at the end of the training, produces a bigram BLEU (Papineni et al., 2002) score of 62.25%. For bootstrapping negative situations and other purposes, the POS tagger from Spacy[5] is used. The Lucene-IR framework is set up to retrieve negative situations.

The model configuration and training parameters for the sarcasm synthesizer is the same as the positive sentiment induction module. For the RL scheme, for each instance, the expected reward is computed over 100 candidate samples. At the end of the training, the bigram BLEU score on the validation set turns out to be 59.3%. For reward computation, we use a classifier similar to the one used for neutralization. The embedding size for this classifier is 300 and it uses two layers of unidirectional LSTMs with a hidden dimension of 300. It trains with a batch size of 64 and produces a validation accuracy of 78.3%. The probability estimates given by the classifier for any input text are taken as reward scores. For optimization, cross entropy loss criterion is used.

4.3 Evaluation Criteria

Absence of automatic evaluation metrics capable of capturing subtleties of sarcasm makes it difficult to evaluate sarcasm generators. For evaluation, we still use the popular translation and summarization evaluation metrics METEOR (Banerjee and Lavie, 2005) and ROUGE (Lin, 2004). Additionally, to check the semantic relatedness between the input and output, we use Skip-thought sentence similarity metric[6]. Note that using BLEU (Papineni et al., 2002) will be futile here as direct

[5]http://spacy.io

[6]https://github.com/Maluuba/nlg-eval

n-gram overlaps between the predicted and gold-standard sentences are not expected to be significantly higher for such a task. We still include it as an evaluation metric for completion.

We employ an additional metric to judge the percentage of length increment (abbreviated as WL) to see the if the length of the output is generally more than that of the input (for the reference text it is 67%). The notion behind this metric is that sarcasm typically requires more context than its literal version, requiring to have more words present at the target side.

4.3.1 Human Judgement based Evaluation

We also consider human judgment scores indicating whether the generated output is non-sarcastic/sarcastic (0/1 labels), how fluent it is (in a scale of 1-5, 1 being lower), and to what extent it is related to the input (in a scale of 1-5). The relatedness measure is important as the objective of the task is to produce a sarcastic version of the input text without altering the semantics much. For human evaluation, we consider only the 30 sentences randomly picked from the benchmark (test) dataset. Sarcasm is a difficult topic, so we stuck to only two annotators who had a better understanding of the language and socio-cultural diversities.

4.4 Systems for Comparison

For comparison, we consider the following four systems:

1. **SarcasmBot**: This is an open-sourced sarcasm generation chatbot released by Joshi et al. (2015a). The system generates a sarcastic response to an input utterance.

2. **UNMT:** This system is based on Unsupervised Neural Machine Translation technique by Artetxe et al. (2017), which can be extended to any translation task. In our setting, the source and target side are literal and sarcastic utterances, *i.e.* the direction of translation is non-sarcastic to sarcastic.

3. **Monoses:** This is similar to UNMT but based on unsupervised Statistical Machine Translation (Artetxe et al., 2018).

4. **ST:** This is based on the cross alignment technique proposed by Shen et al. (2017), used for the task of sentiment translation.

5. **FLIP:** This is based on heuristics for sentiment reversal. For this, the input sentence is first dependency-parsed. The root verb is determined along with its tense and aspects with the help of its part-of-speech tags[7]. The sentiment of root verb is determined using sentiment lexicon[8]. If the verb has non-zero positive or negative sentiment score, its antonym is found using WordNet. Appropriate tense and aspect form of the antonym is then obtained[9]. The modified antonym replaces the original root verb. Similarly, we replace adjective and adverbs with words carrying opposite sentiment.

For training the above systems (except FLIP), we used $\mathcal{S}$ at one side and a larger version of combined $\mathcal{P}$ and $\mathcal{N}$ containing $558,235$ sentences on the other side, curated from the same sources as mentioned earlier. Apart from this system, we also tested some of our model variants, which are presumably inferior and can be considered as baselines. These are termed as:

1. **SG_NORMAL:** a system with only the sarcasm synthesizer module which takes the input directly (after removing stopwords from the input),

2. **SG_RL:**, same as *SG_NORMAL* but also applies reinforcement learning,

3. **ALL_NORMAL:**, the complete system, with sarcasm synthesizer trained without reinforcement learning strategy.

4. **ALL_RL:**, the complete system with reinforcement learning.

5 Results and Analysis

Tables 2 and 3 present evaluation results. While it was expected that the automatic metrics may not be able to capture the subtleties of sarcasm, the WL measure indicates that a carefully designed modular approach like ours often generates longer sentences with more context. This is also corroborated by the human evaluation where annotators have judged that the output generated from

[7]For parsing and POS-tagging spaCy (http://spacy.io/) is used

[8]https://www.nltk.org/_modules/nltk/sentiment/vader.html

[9]https://www.nodebox.net/code/index.php/Linguistics

System	BLEU	METEOR	ROUGE-L	SkipT	WL
SarcasmBot	0.002	0.012	0.01	0.20	-43%
UNMT	0.17	0.15	0.33	0.45	-1.1%
Monoses	0.12	0.13	0.25	0.42	-3.4%
ST	0.017	0.04	0.11	0.24	-4.3%
FLIP	0.10	0.17	0.34	0.45	0.003%
SG_NORM	0.09	0.13	0.24	0.41	-3.3%
SG_RL	0.09	0.13	0.24	0.40	-4.18
ALL_NORM	0.12	0.13	0.23	0.37	32.3%
ALL_RL	0.11	0.13	0.23	0.37	31.2

Table 2: Evaluation results for our system and various baselines.$SkipT \rightarrow$ skip thought similarity

System	Sarcasm	Fluency	Adequacy
SarcasmBot	86.6%	4.3	1.2
UNMT	20%	3.9	3.8
Monoses	33.3%	3.7	3.5
ST	16%	2.8	2.3
FLIP	20%	4.1	3.8
SG_NORM	56.6%	3.7	3.2
SG_RL	63.3%	3.9	3.6
ALL_NORM	63.3%	3.7	3.9
ALL_RL	73.3%	3.7	3.8

Table 3: Human judgment scores for various systems

our system are more sarcastic than the comparison systems. *SarcasmBot*, being a heuristic driven sarcasm generator produces sarcastic responses but is not related to the input topic. Moreover, it ends up generating only 20 different responses for our entire test dataset making its output redundant and unrelated to the input. Other existing systems such as *UNMT* and *Monoses* converge to autoencoding and end up replicating the input as output. *FLIP*, performs transformations at lexical level, hence achieves better fluency but certainly fails to induce sarcasm in most of the cases.

Table 4 presents example generations from different systems. It is quite interesting to note that due to the RL, the model tends to produce longer sentences and brings additional context necessary for sarcasm. The fluency is however compromised. A close inspection of the outputs from each module suggests that the overall error committed by the system is due to accumulation of different types of errors, mainly (a) error during neutralization due to inappropriate assignment of weights to the words in the input, (b) dropping of words and/or insertion of spurious words during positive sentiment induction, and (c) error in scoring the sarcasm content in the RL setting. These can be addressed through better hyper-parameter tuning, gathering more training data for training the individual modules (especially the sarcasm synthesizer), and fine-tuning the sarcasm scorer.

Input: worrying because did not finish my homework.
Reference: *did not finish any homework & i still need to shower ! lol ! love stressing out.* **SarcasmBot:** How exciting! $\star rolls\ eyes \star$. **Monoses:** whining because actually finish my homework. **UNMT:** worrying because i did not finish my homework. **ST**: if do not work my mom hurts. **FLIP**: reassuring because did not finish my homework. **SG (NORMAL)**: i am worrying about the worrying of homework. **SG (RL)**: worrying about finish homework. **ALL (NORMAL):** no worrying , i finish doing homework great. **ALL (RL):** worrying about finish homework is great.
Input: swimming lessons are very costly in nyc.
Reference: *you have to sacrifice your first born child for swimming lessons in nyc.* **SarcasmBot:** That is a very useful piece of information! Yawn! **Monoses:** Dry lessons are very costly financially. **UNMT:** swimming lessons are very costly in nyc. **ST:** a will was in bed. **FLIP**: swimming lessons are very costly in nyc. **SG (NORMAL)**: this is so costly to me swimming lessons in nyc. **SG (RL)**: swimming lessons is so costly in nyc. **ALL (NORMAL):** loving the swimming lessons in nyc ch. **ALL (RL):** i am loving the swimming lessons . going to be a very costly in nyc ch.

Table 4: Examples of generated outputs from different systems. *Reference*$\rightarrow$ human written sarcastic text

6 Related Work

As stated earlier, not many systems for sarcasm generation exist today. The closest work to ours is the one by Joshi et al. (2015a) which employs a heuristic driven approach for generating a sarcastic response to an input utterance. Since, the output of the system is a response, the system is not suitable for translating a literal input text into a sarcastic version. Unlike sarcasm generation, sar-

casm detection has been a well-known problem with several available solutions. For this problem, traditional supervised and deep neural network based solutions have been proposed. The supervised approaches rely on: (a) Unigrams and Pragmatic features (González-Ibánez et al., 2011; Barbieri et al., 2014; Joshi et al., 2015b) (b) Stylistic patterns (Davidov et al., 2010) and patterns related to *situational disparity* (Riloff et al., 2013) and (c) Cognitive features extracted from gaze patterns (Mishra et al., 2016, 2017). Recent systems are based on variants of deep neural networks built on the top of embeddings. Deep neural networks based solutions for sarcasm detection include (Ghosh and Veale, 2016) who uses a combination of RNNs and CNNs for sarcasm detection, and (Tay et al., 2018), who propose a variant of CNN for extracting features related to context incongruity.

A few works exist in the domains of irony, pun and humour generation and are summarized by Wallace (2015), Ritchie (2005) and Strapparava et al. (2011) respectively. However, most of these are heuristic driven and, hence, may not be easily scaled to new domains and languages. From the perspective of language style transfer. Shen et al. (2017) propose an unsupervised scheme to learn latent content distribution across different text corpora and use it for sentiment style transfer. Xu et al. (2018) introduce an unsupervised sentiment translation technique through sentiment neutralization and reinforced sequence generation. Zhang et al. (2018) propose a style transfer technique based on unsupervised MT inspired by Artetxe et al. (2017). Artetxe et al. (2018) have recently proposed an unsupervised statistical machine translation scheme. We adopt some of these modules for the task of sarcasm generation. As far as we know, our proposed model is the first of its kind for end-to-end neural sarcasm generation.

7 Conclusion and Future Work

We proposed a first of its kind approach for textual sarcasm generation from literal opinionated texts. We designed a modular framework for extracting facts from the input, generating incongruous positive and negative situational phrases related to the facts, and finally generating sarcastic variations. For evaluation, we prepared a benchmark dataset containing literal opinions and their sarcastic versions. Through qualitative and quantitative analysis of the system's performance on the benchmark dataset, we observed that our system often generates better sarcastic sentences compared to some of our trivial model variants, and unsupervised systems used for machine translation and sentiment style transfer. In the future, we would like to extend this framework for cross-lingual and cross-cultural sarcasm and irony generation.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018. Unsupervised statistical machine translation. *arXiv preprint arXiv:1809.01272*.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv:1409.0473*.

Satanjeev Banerjee and Alon Lavie. 2005. Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Francesco Barbieri, Horacio Saggion, and Francesco Ronzano. 2014. Modelling sarcasm in twitter, a novel approach. *ACL 2014*, page 50.

Elisabeth Camp. 2012. Sarcasm, pretense, and the semantics/pragmatics distinction. *Noûs*, 46(4):587–634.

John D Campbell and Albert N Katz. 2012. Are there necessary conditions for inducing a sense of sarcastic irony? *Discourse Processes*, 49(6):459–480.

Dmitry Davidov, Oren Tsur, and Ari Rappoport. 2010. Semi-supervised recognition of sarcastic sentences in twitter and amazon. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 107–116. Association for Computational Linguistics.

Aniruddha Ghosh and Tony Veale. 2016. Fracking sarcasm using neural network. In *Proceedings of the 7th workshop on computational approaches to subjectivity, sentiment and social media analysis*, pages 161–169.

Roberto González-Ibánez, Smaranda Muresan, and Nina Wacholder. 2011. Identifying sarcasm in twitter: a closer look. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 581–586. Association for Computational Linguistics.

Caglar Gulcehre, Sungjin Ahn, Ramesh Nallapati, Bowen Zhou, and Yoshua Bengio. 2016. Pointing the unknown words. *arXiv preprint arXiv:1603.08148*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eduard Hovy. 1987. Generating natural language under pragmatic constraints. *Journal of Pragmatics*, 11(6):689–719.

Parag Jain, Abhijit Mishra, Amar Prakash Azad, and Karthik Sankaranarayanan. 2018. Unsupervised controllable text formalization. *arXiv preprint arXiv:1809.04556*.

Aditya Joshi, Samarth Agrawal, Pushpak Bhattacharyya, and Mark J Carman. 2017a. Expect the unexpected: Harnessing sentence completion for sarcasm detection. In *International Conference of the Pacific Association for Computational Linguistics*, pages 275–287. Springer.

Aditya Joshi, Pushpak Bhattacharyya, and Mark J Carman. 2017b. Automatic sarcasm detection: A survey. *ACM Computing Surveys (CSUR)*, 50(5):73.

Aditya Joshi, Anoop Kunchukuttan, Pushpak Bhattacharyya, and Mark James Carman. 2015a. Sarcasmbot: An open-source sarcasm-generation module for chatbots. In *WISDOM Workshop at KDD*.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015b. Harnessing context incongruity for sarcasm detection. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 757–762.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Dimitrios Kotzias, Misha Denil, Nando De Freitas, and Padhraic Smyth. 2015. From group to individual labels using deep features. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 597–606. ACM.

Juncen Li, Robin Jia, He He, and Percy Liang. 2018. Delete, retrieve, generate: A simple approach to sentiment and style transfer. *arXiv preprint arXiv:1804.06437*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Edward Loper and Steven Bird. 2002. Nltk: the natural language toolkit. *arXiv preprint cs/0205028*.

Abhijit Mishra, Kuntal Dey, and Pushpak Bhattacharyya. 2017. Learning cognitive features from gaze data for sentiment and sarcasm classification using convolutional neural network. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 377–387.

Abhijit Mishra, Diptesh Kanojia, Seema Nagar, Kuntal Dey, and Pushpak Bhattacharyya. 2016. Harnessing cognitive features for sarcasm detection. *ACL 2016*, page 156.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL, 2002*.

Paul Piwek. 2003. A flexible pragmatics-driven language generator for animated agents. *arXiv preprint cs/0312050*.

Ellen Riloff, Ashequl Qadir, Prafulla Surve, Lalindra De Silva, Nathan Gilbert, and Ruihong Huang. 2013. Sarcasm as contrast between a positive sentiment and negative situation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 704–714.

Graeme Ritchie. 2005. Computational mechanisms for pun generation. In *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*.

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.

Advaith Siddharthan. 2014. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298.

Carlo Strapparava, Oliviero Stock, and Rada Mihalcea. 2011. Computational humour. In *Emotion-oriented systems*, pages 609–634. Springer.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 3319–3328. JMLR. org.

Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. 2018. Reasoning with sarcasm by reading in-between. *arXiv preprint arXiv:1805.02856*.

Byron C Wallace. 2015. Computational irony: A survey and new perspectives. *Artificial Intelligence Review*, 43(4):467–483.

Robert West and Eric Horvitz. 2019. Reverse-engineering satire, or" paper on computational humor accepted despite making serious advances". *arXiv preprint arXiv:1901.03253*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Jingjing Xu, Xu Sun, Qi Zeng, Xuancheng Ren, Xiaodong Zhang, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. *arXiv preprint arXiv:1805.05181*.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018. Style transfer as unsupervised machine translation. *arXiv preprint arXiv:1808.07894*.