Project#7: Sarcasm Generator

Abhinav Tripathi[1], **Chitrak Raj Gupta**[2], **Sunny Kant**[3]
[1]15807023, [2]15917207, [3]15817740
[1]EE, [2]MTH, [3]MTH
{abhinavt, chitrak, sunnyk}@iitk.ac.in

Abstract

In this project, we generate sarcastic remarks on a topic given by the user as input. Sarcasm is a relatively new innovation in NLP and topic-based sarcasm generation is still an unexplored field. We propose a novel four-step based sarcasm generation approach. Given an input, in the first step, we find the general consensus about the topic by retrieving relevant tweets and reddit articles. In the second step, we extract the key adjectives from the retrieved corpus using adjective clustering. In the third step, we generate a simple sentence containing the topic and the key adjective using a predefined template. Finally, in the fourth step, we use the simple sentence as an input to a customized plug and play model to generate a sarcastic comment.

1 Introduction

Sarcasm is defined as the use of irony to mock or convey contempt [Oxford]. Usually, the literal meaning of a sarcastic sentence is the opposite of the idea conveyed. Sarcasm detection and generation have been a complex problem in NLP owing to ambiguity in apparent and intended meaning.

Quite some work has been done in sarcasm detection, but studies related to sarcasm generation are very limited. There are some studies about sarcasm generation like Sarcasm-Bot(Joshi et al., 2015) and (Mishra et al., 2019), but while the former framework generates sarcasm as a response to an input statement, and not a topic, the latter approach generates sarcasm by converting a negative sentiment into its sarcastic counterpart. Topic based sarcasm generation has not been studied in detail, to the best of our knowledge.

We propose a four-step architecture for sarcasm generation: (i) using the input topic to retrieve relevant tweets and reddit articles. (ii) We will use this corpus to select the key adjectives describing the given input topic using similar adjective clustering method. (iii) A simple sentence is generated using the given topic and the retrieved topic using a pre-defined template. (iv) We extend this sentence using a customized plug and play model.

The rest of the paper is organized as follows: in section 2, we formally define our problem. In section 3, we describe the related work done in the field of generating sarcasm. In section 4, we describe in detail our proposed approach. In section 5, we describe the corpus we gather. This is followed by experimental results in section 6, and error analysis methods in section 7. We discuss the individual contribution in section 8 and future work in section 9. We also briefly discuss the questions and suggestions we received during our presentation after conclusion in section 10.

2 Problem Definition

Our problem statement is very simple. Given any topic by the user as input, we generate a sarcastic remark on it.

For example, given an input NETFLIX, we would like to generate comments like *(i) My NETFLIX history is a bunch of movies I have fallen asleep to. (ii) Use the promo code NETFLIX to get 50% off your next exam.* Note that both these sarcastic comments are famous tweets written by humans. Also, they are sarcastic because there is an underlying context in both these tweets. In the former, it is about the NETFLIX content being so boring that a person falls asleep while watching it. The latter one is about how NETFLIX can affect the studies of a student. Hence, we see that the context plays an important role in making a comment sarcastic.

This task is challenging for the following main reason: there has to be a context to express sarcasm. Sarcasm can be expressed in many ways, but at its

very core is an incongruity that indicates irony. For the sarcasm to make sense, both - the person (or machine) that makes the comment and the one who reads it must understand the underlying context.

3 Related Work

To the best of our knowledge, there has not been any work on topic based sarcasm generation. However, (Joshi et al., 2015) algorithmize the different kinds of sarcastic expressions to be given as responses based on the input statement of the user. The model is known as Sarcasm-Bot. This can be modified to be used in chat-bots that can understand user inputs and reply sarcastically. For example, User input: *Why did Greg go home?* Sarcastic response: *Because Greg was in a mood for adventure.* This is done by analyzing the input given by the user to select an appropriate generator and generate a sarcastic response. The summary of this approach is listed in the table below.

In another work, (Mishra et al., 2019) use an LSTM based model in order to convert a sentence with negative sentiments into its sarcastic counterpart. This is done in four main stages: (A) Sentiment Neutralizer, (B) Positive Sentiment Inducer, (C) Negative Situation Retriever, and (D) Sarcasm Synthesizer. For example, the sentence: *I hate it when the bus is late* is converted into *I absolutely love waiting for the bus.*

(Petrović and Matthews, 2013) use the template *I like my X like I like my Y, Z* to generate jokes. In these kinds of jokes, *X* and *Y* are nouns, and *Z* is an adjective. *X* and *Y* should be highly dissimilar, but *Z* should occur frequently with both *X* and *Y*. An example can be: *I like my relationships like I like my source, open*.

Even though there is some relevant work in the field of joke and sarcasm generation, there are no studies on topic based sarcasm.

4 Proposed Approach

Given a topic by user as input, our model for sarcasm generation follows following steps to generate sarcasm :

1. **Context Retrieval**: Relevant tweets from twitter and comments from reddit are retrieved using tweepy and praw api. We have applied filtering conditions like language, recent or popular etc. while retrieving the tweets and comments.

2. **Collecting Key Adjectives** On the relevant tweets and comments, we used NLTK library to get POS tagging and POS tree. From that we collect adjectives related to given context. The adjectives are collected in a way that they express the sentiments about the topic. For that we collect adjectives from POS subtree with topic word as root and we collect all such adjectives within a fixed radius from context word in tweets and comments.

   Then we get opposite adjectives from those we got from tweets and comments by POS tagging. To do so, we use WordNet vocabulary and NLTK antonyms library to get opposite adjectives.

   Adjective Clustering: Collecting adjectives is one of the main tasks in the pipeline, hence it is important that we get the most relevant adjectives. We propose to do this by the method of adjective clustering. We use the POS tree to extract all the minimal NPs that contain only the head noun (user input topic). We make the assumption that the adjectives only modify the head noun (Hatzivassiloglou and McKeown, 1993). From the POS tree, we extract word pairs (adj-noun and adj-adj) and use two similarity modules: Firstly, in the case of adj-noun pairs, we know that the adjective modifies the noun. This allows us to know which adjective is the noun related to. Secondly, we use adj-adj pairs in order to find unrelated adjectives. Consider these 3 cases for the adj-adj pair:

   - If both the adjectives are of opposite polarity, then the sentence will become self contradictory. *eg, cold hot*.
   - If both the adjectives represent different scales of the same quality, it would violate the common structure of language. *eg, warm hot*.
   - Thus, we can only have an adj-adj pair if both are unrelated. *eg, light blue*.

   Finally, when we have a bucket of possibly related adjectives, we can find out whether both are similar or not as follows. For example, consider the adjectives *global* and *international* (X and Y). Let us check their occurrences with a pair of nouns *i,j*. For the observations pairs (X_i,X_j) and (Y_i,Y_j), we call the observation **concordant** if both ($X_i > X_j$)

INPUT	INVOKES	EXAMPLE INPUT	EXAMPLE OUTPUT
Offensive Word	Abusive Language Handler	**** ***	I can't believe you said ****. You're really classy.
Yes/No Question	1. Opposite Polarity Verb-Situation Generator 2. Irrealis Sarcasm Generator 3. Hyperbole Generator	1. Do you like Tim? 2. Will he marry me?	1. Hmm, well.. I used to like Tim, just the way I like being stuck in the elevator all night. 2. Only if he is drunk.
Opinion Question	1. Opposite Polarity Person Attribute Generator 2. Opposite Polarity Verb-Situation Generator 3. Hyperbole Generator	1. What do you think about Jim? 3. What do you think about the immigration reforms?	1. I think that Jim is Tom Cruise minus talent. 3. I think they are the best immigration reforms in the history of humankind.
Why Question	Incongruent Reason Generator	Why did Jim miss the date tonight?	Because he loves to play the piano.
Opinion	Sentiment Based Sarcasm Generator	I am excited about my new job!'	Poor you!

Table 1: Algorithmic Sarcastic Responses from SarcasmBot

	X: global	Y: international
i: market	20	27
j: coordination	12	11

Table 2: Adjective Clustering Technique: A Concordant Example

and ($Y_i > Y_j$) or ($X_i < X_j$) and ($Y_i < Y_j$), otherwise they are **discordant**.

Representing the number of concordant examples by C and discordant examples by D, we define similarity as $S = \frac{C-D}{C+D}$. In the end, we will choose a random adjective from the list as all the adjectives will have similar meaning.

A template is used to make a simple sentence from the given topic and the adjective opposite to the chosen one.

3. **Sentence generation with PPLM** After finding out relevant adjectives from our corpus, we need to output a sarcastic message. OpenAI's GPT-2 (Radford et al., 2019) is a transformer based Language model which is capable of generating text based on user input, but it is unconditional, hence hard to control. But training or fine tuning a model as large as GPT-2 is a tremendous and computationally expensive task. Plug and Play Language model (Dathathri et al., 2020) resolves this issue by using a small attribute model on top of large language Models such as GPT-2 to control the generated output. The model takes inspiration from Bayes rule.

$$p(x \mid a) \propto p(a \mid x)p(x) \tag{1}$$

where p(x) is the unconditional Language Model, $p(a \mid x)$ is the attribute model and $p(x \mid a)$ is the controlled Language Model. Instead of using random sampling, which is highly efficient, we use Metropolis-adjusted Langevin Sampler(Roberts and Tweedie, 1996).

In out implementation, instead of using a heavy sarcasm detection module, we use a simple bag of words model, with two bags of word. The first set contains words which are often used to mark sarcasm such as **actually**, **really** etc and quantifiers such as **much**, **most** etc. In the second set, we have nouns which impart a sarcastic sense to adjective such as (fast, snail), (cold, fire) etc. The first set of words are fixed, but the second set is computed for adjective by looking at a large corpus. To select nouns related to an adjective, we look at the metric

$$f_{A,N} = \frac{C(A', N)}{C(N)} \tag{2}$$

Where, A is adjective, N is noun and A' is adjective opposite to A. We choose nouns N which have high values of $f_{A,N}$. This metric is an estimate of likelihood noun N will impart a sense opposite to that imparted by adjective A. A' can be found with wordnet.

5 Corpus/Data Description

For getting adjective-adjective and adjective-noun relations, we look at datasets from multiple domains such as News, reddit, Twitter, Wikipedia etc to find which can work best with our problem.

1. **All the news**: Contains 143,000 news articles from 15 news publications. Hosted on Kaggle.
2. **Sentiment140**: Contains 1.6 million tweets. Hosted on Kaggle.
3. **Sarcasm on Reddit**: Contains 1.3 million annotated comments from redit. Hosted on kaggle.
4. **Wikipedia Summary Dataset** Contains summary of all wikipedia articles extracted as on September 2017. Hosted on github.

Since we are working on Statistical NLP, the models are learned from data.

6 Experiments and Results

To test out Plug and Play Language Model, we used four starting points which were names of place(Iran), sport(Football), person(Trump) and a pronoun(I). The adjectives we handpicked were hostile, bad, stupid, slow. We generated text from these adjectives and then replace them with peaceful, good, smart and fast respectively. We obtained a few interesting results.

1. *Iran is peaceful than* any other country in the world today, but the US has no intention of letting this be so.
2. *Football is better than* any other sport. That's what a lot of people believe. So what?
3. *Trump is smarter* and less likely to be able to say what the right of a nation to protect its right to life.
4. *I am faster than* most people in my first year so I don't know how much faster that actually is.

Here, the blue words are from our bag of word model's word list. Hence, we can see that with a good starting point, our model is able to generate sarcastic responses.
As we are still looking into good datasets for getting adjective-noun relations, to test our method of generating contradictory pairs, we take help from google n-gram viewer and look into the adjective noun pairs.

	Rabbit	snail	ice	rock	scientist
slow	*5.5*	0.85	**9.2**	3.4	0
fast	0.35	**5.7**	0.3	5.2	0
hot	0	0.71	**7.1**	1.2	5.1
light	0.29	0.22	6.6	**9.2**	0
stupid	0	0	0.03	0.39	**1.1**

Table 3: $f_{A,N}$ score for various adjectives.

From the table, we see that the pairs chosen are (slow, ice), (fast, snail), (hot, ice), (light,rock) and (stupid, scientist). Except for (slow, ice) all other pairs have a sense of contradiction. As fast ice is an often used bigram, it makes the result erroneous, but this problem is unlikely to happen when we use a large corpus along with proper Part of Speech Tags.

7 Error Analysis

Sarcasm is subjective in nature. Hence, it is not possible to have a precise metric to check the level of sarcasm through a machine. We believe that in case of sarcasm, there is no better method when compared to human evaluation. We propose the following three metrics: grammatical correctness, sarcastic nature and comparative evaluation. To evaluate *sarcastic nature*, we will outsource the output produced by our model and ask a large number of English speaking people to rate the text on a scale of 0-2 as (a) neutral: 0, (b) somewhat sarcastic: 1, and (c) sarcastic: 2. For *comparative evaluation*, we will check whether the people are able to identify sarcastic responses from the non-sarcastic responses from the GPT-2 model.

8 Individual Contribution

Abhinav	Adjective Clustering, Collecting Opposite polarity adjectives form WordNet using NLTK
Chitrak	Sentence gereration using GPT2 and PPLM model, Bag of Words model for adjectives
Sunny	Scraping relevant tweets and comments of reddit, POS tagging using NLTK, POS tree to get relevant adjectives of context

9 Future Work

1. Bag of Words Attribute model, although primitive is still quite effective. We aim to find a slightly complex, but not too computationally heavy model to improve the results of PPLM.
2. We will try to get relevant adjectives from tweets or comments from reddit instead of some fixed adjectives to make our model work on different and diverse given topics.

10 Conclusion

We have successfully demonstrated the use of plug and play models to generate sarcasm using a handcrafted starting sentence. Our pipeline is working fine with given topics and handpicked adjectives. In order to complete our end to end model, we have to implement adjective clustering on the mentioned corpora to get relevant adjectives from tweets and comments.

11 Presentation Feedback (Max 2 columns)

The following questions were asked during presentation:

1. Will we be using Knowledge-based models? - So far we have not looked into the same. If time permits, we will see if we can incorporate knowledge based models to get context for sarcasm.

2. Any approach other than Adjective clustering? - Knowledge based or sentiment based sarcasm generation.

3. Are you getting tweets in different languages and how? - Using tweepy api, we can get tweets in specified language.

4. How are we generating simple sentences? - We are using Plug and Play model to generate sentences. We use template based method to generate a simple sentence (or a part of it) which is used as input for Plug and Play model.

5. How are we selecting the adjectives? - We are using POS tree to get adjectives relevant to given topic. We use adjective clustering to bucket the similar adjectives. We get opposite polarity adjectives using NLTK antonyms from Wordnet Vocabulary.

References

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Vasileios Hatzivassiloglou and Kathleen R McKeown. 1993. Towards the automatic identification of adjectival scales: Clustering adjectives according to meaning. In *Proceedings of the 31st annual meeting on Association for Computational Linguistics*, pages 172–182. Association for Computational Linguistics.

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

Abhijit Mishra, Tarun Tater, and Karthik Sankaranarayanan. 2019. A modular architecture for unsupervised sarcasm generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6146–6155.

Saša Petrović and David Matthews. 2013. Unsupervised joke generation from big data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–232.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8).

Gareth O. Roberts and Richard L. Tweedie. 1996. Exponential convergence of langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363.