IronyMagnet at SemEval-2018 Task 3: A Siamese Network for Irony Detection in Social Media

Aniruddha Ghosh
University College Dublin
Dublin, Ireland
aniruddha.ghosh@ucdconnect.ie

Tony Veale
University College Dublin
Dublin, Ireland
tony.veale@ucd.ie

Abstract

This paper describes our system, entitled IronyMagnet, for the 3rd Task of the SemEval 2018 workshop, "Irony Detection in English Tweets". In Task 1, irony classification task has been considered as a binary classification task. Now for the first time, finer categories of irony are considered as part of a shared task. In task 2, three types of irony are considered; "Irony by contrast" - ironic instances where evaluative expression portrays inverse polarity (positive, negative) of the literal proposition; "Situational irony" - ironic instances where output of a situation do not comply with its expectation; "Other verbal irony" - instances where ironic intent does not rely on polarity contrast or unexpected outcome. We proposed a Siamese neural network for irony detection, which is consisted of two subnetworks, each containing a long short term memory layer (LSTM) and an embedding layer initialized with vectors from Glove word embedding[1]. The system achieved a f-score of 0.72, and 0.50 in task 1, and task 2 respectively.

1 Introduction

Irony is one of the most prominent and pervasive figures of speech in human communication, dating back to ancient religious texts to modern microtexts. According to literary scholars (Grice, 1978; Lakoff, 1993), irony has been defined as a trope where the speaker intends to communicate a contradictory situation or the opposite meaning of what is literally said. It adopts a subtle technique where incongruity is used to suggest a distinction between reality and expectation in order to produce a humorous or emphatic effect on the listener. Irony poses an important challenge not only from a linguistic perspective but also from a cognitive one. Even without a solid understanding of irony, one can still recognize and produce ironic utterances from as early as childhood (Harris and Pexman, 2003). Such capabilities are often associated with one's ability to correctly infer others' communicative intentions and perspectives towards a given situation. Psychological theories, such as "echoic reminder theory" (Kreuz and Glucksberg, 1989), "allusion pretense theory" (Kumon-Nakamura et al., 1995), and "implicit display theory" (Utsumi, 2000), confirm that cues for understanding ironic intent are not restricted to language. Ironic intent involves several other aspects including, but not limited to, the context of an utterance, the world's perception and familiarity between the listener and the speaker, and psychological dimensions. Thus, as a purely text classification task, the irony detection task poses a significant challenge for computational linguists. Computational approaches focus on identifying the subtle incongruity between different parts of the text. Often, an ironic statement starts with an overtly positive attitude ("Yay I love") and ends up in an disappointment ("working on my birthday") or a negative attitude/statement ("another outage in less than 8 hours.") followed by an appreciation ("Keep up the good work!") or an incident ("I asked God to protect me from my enemies") resulting in a completely unexpected output ("shortly after I started losing friends").

Due to the limited scope of expression in social media such as Twitter[2], authors often end up lacing their statements with ironic cue words ('Yay') or social media specific features such as hashtags ('#irony') to make their ironic intent more obvious for the reader. Following this intuition, most of the attempts were made using probabilistic classification models which relied on textual cues such as lexical indicators like punctuation symbols (e.g., '?'), interjections (e.g., "gee" or "gosh"), and intensifiers (Kreuz and Caucci, 2007); the juxtaposition of positive sentiment and

[1] https://nlp.stanford.edu/projects/glove/

[2] www.Twitter.com

negative situations (Riloff et al., 2013); discriminative N-grams like 'yay!' or 'great!' or "oh really" or "yeah right" (Tsur et al., 2010; Lukin and Walker, 2013); social media markers like hashtags (Davidov et al., 2010); emoticon usage (González-Ibánez et al., 2011); and topics associated with irony (e.g. schools, dentists, church life, public transport, the weather). Carvalho et al. (2009) exploited text patterns in comments on articles of online newspapers to detect ironic statements, while Van Hee et al. (2016) developed a irony detection model using support vector machine (SVM) with a combination of lexical, syntactic, sentiment, and semantic (Word2Vec embedding) features. In recent times, multiple research attempts, founded on variants of the deep neural network built on top of word embeddings, showed a significant improvement over traditional methods over several natural language processing (NLP) tasks. A few representative works in this direction for detecting sarcasm, a demeaning variant of irony, especially in the colloquial form, are based on Convolutional Neural Networks(CNN) (Mishra et al., 2017), Recurrent Neural Networks (RNN) (Zhang et al., 2016) and a combination of CNNs and RNNs (Ghosh and Veale, 2016). Siamese networks (Bromley et al., 1994), widely used in image classification, have displayed a good performance over sentence similarity or document similarity tasks. A Siamese architecture contains two identical sub-networks, which are trained with two different inputs to distinguish the difference among them. Since in irony, different parts of a sentence can be incongruous with each other, we adopted the Siamese architecture to detect such incongruity between different sections of a sentence. In this implementation, each of the subnetworks consists of an embedding layer and a LSTM layer. We slice each input sentence into two fragments and feed them to the subnetworks. The output representations of subnetworks are then compared to distinguish if the two fragments are incongruous or not i.e. a sentence is considered as non-ironic if two fragments of a sentence are semantically non-incongruous, otherwise it is ironic.

2 Dataset Preparation & Resources

The train and test dataset consists of 3834 tweets and 784 tweets, respectively. The train dataset for Task 1 is balanced but in task 2, the prominent category was "irony by polarity contrast" (table 1).

2.1 Data Normalization

The distorted language, use of abbreviation, and high number of one-off words, prevents a model from being robust. Thus, each tweet is preprocessed, normalized, and cleaned with the following criteria.

1. To emphasize the ironic effect, an author often uses repetition of a character or a word. A set of regular expressions is used to normalize the word ("loooong" to "long") and the word is replaced with a word from WordNet[3] which has the lowest minimum edit distance[4] between them. The repeated word sequence is split using a regular expression into multiple words("YAYYAYYAYYAYYAY" to "YAY YAY YAY YAY YAY").

2. Since this is a text based classification task, each link from the tweets is discarded.

3. The limited scope of social media incentivizes users to exploit different features of social media such as hashtags and emoticons more creatively and efficiently to express an opinion. Hashtags are often used by authors to emphasize key parts or themes in their texts or to convey their attitude or feeling towards the subject. This can provide a toehold for NLP techniques in order to infer the intended sense behind an ironic statement. Thus, each hashtag is processed and split into words. For example, "#TheReasonForTheSeason" is converted in "The Reason For The Season".

4. Emojis have played a significant role in expressing the hidden feelings of a person not evident in plain text. Each individual emoji is replaced with their official name[5].

5. The high number of one-off words increases the vocabulary size of a network. Due to the sparsity in the dataset because of one-off words, the model finds it difficult to train. Thus, all non-valid English words, according to WordNet dictionary, occurring less than twice in the entire dataset are removed.

6. Neural network models are data hungry and their success often relies on the size of the

[3]https://wordnet.princeton.edu/

[4]https://en.wikipedia.org/wiki/Edit_distance

[5]https://unicode.org/emoji/charts/full-emoji-list.html

train dataset. Since the train dataset is relatively small, it can not extract the distinctive, robust features for detecting irony. Thus, the dataset is extended by replacing the overtly positive and negative words with "positive" and "negative" respectively using the Sentiwordnet[6].

7. In order to capture the incongruity between topics, the dataset is further extended by replacing words with its category type extracted from WordNet.

2.2 Building Train Dataset

Since a Siamese network expects two inputs and performs a comparison between the generated weights, each tweet is split into two parts. A number of training examples are generated by splitting the tweets where each split contains a minimum r number of words. Consider as an input a tweet containing n words with label y. The tweet will be split into $(n - 2 \times r + 1)$ combinations. In this experiment, the value of r is chosen as 3. For example, "Working on Boxing Day is so fun" will produce the following combinations:

1. ("Working on Boxing", "Day is so fun")
2. ("Working on Boxing Day", "is so fun")

For training purposes, the train dataset is split with a 90%-10% split ratio.

3 Siamese Network

3.1 Input Layer

Each tweet, with length n, is converted to a vector by replacing a word with its index value in the dictionary $s \in \Re^{1 \times n}$. To resolve different lengths of input, each tweet is either padded or truncated to convert into a vector of size $s \in \Re^{1 \times l}$ where l is the maximum allowed length. The maximum allowed length for the input vector for each sub-network is set to the average length of the tweets. The input vector is fed to an embedding layer which converts each word into a distributional vector of dimension D. Thus, the input tweet matrix is converted $s \in \Re^{l \times D}$. The embedding layer is initialized with the embeddings extracted from Glove word embedding. We freeze the embedding layer to keep the general meaning of a word intact.

[6]http://sentiwordnet.isti.cnr.it/

task	0	1	2	3
1	1923	1911	-	-
2	1923	1390	316	205

Table 1 Train Data Statistics

3.2 LSTM Layer

Among the variants of RNN networks, LSTM has demonstrated the power of semantic modelling by efficiently handling long term dependencies (Hochreiter and Schmidhuber, 1997) by defining each memory cell with a set of gates $\Re^d$, where d is the memory dimension of hidden states of LSTM. It does not suffer from a vanishing or exploding gradient problem while performing back propagation through time. There are three gates, which are functions of x_t and h_{t-1}: input gate i_t, forget gate f_t, and output gate o_t. The gates jointly decide whether the memory update mechanism will occur or not. Equation (2) and (1) denotes the amount of information to discard and what to store in memory. Equation (4) denotes the output of a cell c_t. Equation (3), (5), and (6) denotes the input activation, cell state, and output vector of a LSTM cell respectively.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \tag{1}$$

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \tag{2}$$

$$q_t = sigmoid(W_q[h_{t-1}, x_t] + b_q) \tag{3}$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \tag{4}$$

$$c_t = f_t \odot c_{t-1} + i_t \odot q_t \tag{5}$$

$$h_t = o_t \odot sigmoid(c_t) \tag{6}$$

Due to the small dataset size, extra attention has been paid in order to prevent the network from overfitting. A recurrent Dropout is used between each time step of an input. Each LSTM layer outputs a weight matrix $s \in \Re^{l \times m}$ (m = number of hidden memory units), which is passed to a Dropout layer.

3.3 Subtract Layer

The generated weights of each LSTM layer carries the conceptual representation of its input. Intuitively, the weight difference between the output of two LSTM layers should signify conceptual representations as either incongruous to each other or not. A subtract layer is used to calculate the weight difference between two sub networks.

The subtract layer produced an output matrix $s \in \Re^{l \times m}$, which is passed as an input to a fully connected layer.

3.4 Fully Connected Layer

The fully connected layer produces a higher order feature set, based on the weight matrix obtained from the LSTM layer, which is easily separable into different classes. At the end, a Softmax layer is added on top of the fully connected layer.

4 Experimental Setup

Success with a neural network model largely depends on the apt input and optimal hyper-parameters settings. After investigating different combinations of hyper-parameters, the optimal setting is obtained for each layer of the network. The LSTM has 32 hidden memory units with a *sigmoid* activation function and recurrent dropout ratio of 0.5. The fully-connected layer consisted of 16 hidden memory units and uses *ReLu* as the activation function. Both of the layers are initialized with *Xavier normal initializer* (Glorot and Bengio, 2010). As an optimizer function, Adam optimization is used with a learning rate set to 0.001, while categorical cross-entropy is chosen as a loss function. The code is developed using the *keras*[7] library.

5 Results

For both of the tasks, our model is compared with the state-of-the-art composite neural network model (Ghosh and Veale, 2016). Each model is trained with the same datasets. For the composite model, the entire tweet is fed as an input instead of just fragments of the tweet.

6 Output Analysis

In both tasks, the Siamese network outperforms the composite neural network model. The composite neural network model, trained with original tweets, is only able to capture certain ironic utterances where a similar pattern is encountered in the training dataset. The model responds well to obvious ironic markers such as “Ohh” in figure 2. Without the obvious ironic markers, the model mis-classifies the statement as non-ironic. Whereas the Siamese network classifies the statement correctly even without the obvious ironic marker, the subtract layer captures the incongruity between two concepts in an ironic statement. However, the Siamese network fails to detect ironic statements with dropped negations. For example, the network could not figure out the ironic intent behind the following statement “Hey there! Nice to see you Minnesota/ND Winter Weather”, the model has no information about “Minnesota/ND Winter Weather”.

[7]http://keras.io/

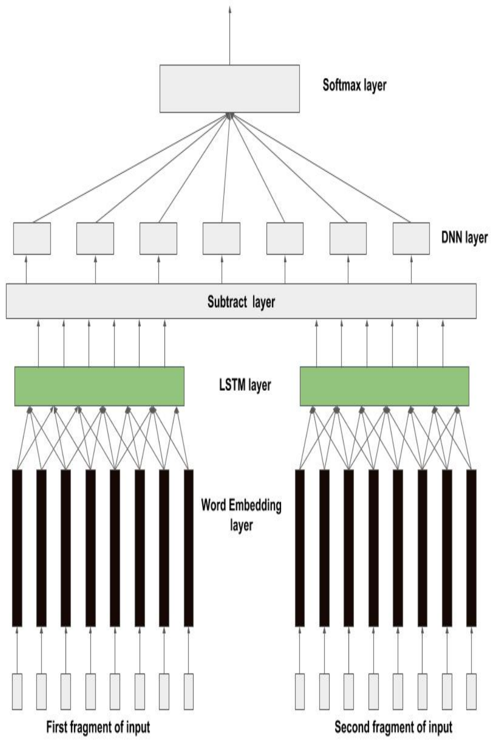

Figure 1 Siamese Neural network

Task	Model	P	R	F1
Task1	Ghosh and Veale (2016)⋆	0.6449	0.4437	0.5257
	Siamese network	0.7878	0.6688	0.7234
Task2	Ghosh and Veale (2016)	0.4099	0.4187	0.3988
	Siamese network⋆	0.5768	0.5044	0.5074

Table 2 Experiment Results; ⋆submissions considered in final standing

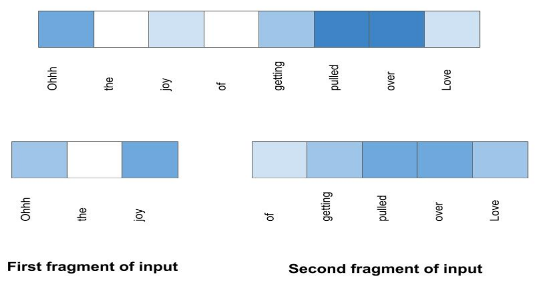

Figure 2 Output Vector of LSTM Layer

7 Conclusion and Future Works

We introduced *IronyMagnet*, a Siamese neural network model that is capable of separating ironic statements from non-ironic statements, as well as at a fine-grained level. Even with small datasets, the Siamese neural network is able to robustly capture the incongruity between two concepts. However, the system lacks the sophistication of understanding incongruity at a pragmatic level. Take, for example, "Whatever happened to the Guano Apes? Did they ever make it "Big in Japan"?". In this example, none of the models are able to establish the incongruity between "Guano Apes" and "Big in Japan". Also, it can not correctly detect if the incongruous elements are located very close to one another within the tweet. However, since the Siamese network is able to correctly classify simple cases of irony, we can therefore hypothesize that the Siamese network can be a stepping stone towards determining contrastive figurative languages. In the future, we would like to extend our model by incorporating an attention network and other psychological stimuli which pertain to Irony.

References

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. 1994. Signature verification using a" siamese" time delay neural network. In *Advances in Neural Information Processing Systems*, pages 737–744.

Paula Carvalho, Luís Sarmento, Mário J Silva, and Eugénio De Oliveira. 2009. Clues for detecting irony in user-generated contents: oh...!! it's so easy;-. In *Proceedings of the 1st international CIKM workshop on Topic-sentiment analysis for mass opinion*, pages 53–56. ACM.

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.

Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

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.

H Paul Grice. 1978. Further notes on logic and conversation. *1978*, 1:13–128.

Melanie Harris and Penny M Pexman. 2003. Children's perceptions of the social functions of verbal irony. *Discourse Processes*, 36(3):147–165.

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

Roger J. Kreuz and Gina M. Caucci. 2007. Lexical influences on the perception of sarcasm. In *Proceedings of the Workshop on Computational Approaches to Figurative Language*.

Roger J Kreuz and Sam Glucksberg. 1989. How to be sarcastic: The echoic reminder theory of verbal irony. *Journal of Experimental Psychology: General*.

Sachi Kumon-Nakamura, Sam Glucksberg, and Mary Brown. 1995. How about another piece of pie: The allusional pretense theory of discourse irony. *Journal of Experimental Psychology: General*, 124(1):3.

George Lakoff. 1993. The contemporary theory of metaphor.

Stephanie Lukin and Marilyn Walker. 2013. Really? well. apparently bootstrapping improves the performance of sarcasm and nastiness classifiers for online dialogue. In *Proceedings of the Workshop on Language Analysis in Social Media*, pages 30–40.

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.

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 *EMNLP*, volume 13, pages 704–714.

Oren Tsur, Dmitry Davidov, and Ari Rappoport. 2010. Icwsm-a great catchy name: Semi-supervised recognition of sarcastic sentences in online product reviews. In *ICWSM*.

A. Utsumi. 2000. Verbal irony as implicit display of ironic environment: Distinguishing ironic utterances from nonirony. *Journal of Pragmatics*.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2016. Monday mornings are my fave:)# not exploring the automatic recognition of irony in english tweets. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2730–2739.

Meishan Zhang, Yue Zhang, and Guohong Fu. 2016. Tweet sarcasm detection using deep neural network. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2449–2460.