Fracking Sarcasm using Neural Network

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

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

Abstract

Precise semantic representation of a sentence and definitive information extraction are key steps in the accurate processing of sentence meaning, especially for figurative phenomena such as sarcasm, Irony, and metaphor cause literal meanings to be discounted and secondary or extended meanings to be intentionally profiled. Semantic modelling faces a new challenge in social media, because grammatical inaccuracy is commonplace yet many previous state-of-the-art methods exploit grammatical structure. For sarcasm detection over social media content, researchers so far have counted on Bag-of-Words(BOW), N-grams etc. In this paper, we propose a neural network semantic model for the task of sarcasm detection. We also review semantic modelling using Support Vector Machine (SVM) that employs constituency parse-trees fed and labeled with syntactic and semantic information. The proposed neural network model composed of Convolution Neural Network(CNN) and followed by a Long short term memory (LSTM) network and finally a Deep neural network(DNN). The proposed model outperforms state-of-the-art text-based methods for sarcasm detection, yielding an F-score of .92.

1 Introduction

Figurative language, such as metaphor, irony and sarcasm, is a ubiquitous aspect of human communication from ancient religious texts to modern micro-texts. Sarcasm detection, despite being a well-studied phenomenon in cognitive science and linguistics (Gibbs and Clark, 1992; gib, 2007; Kreuz and Glucksberg, 1989; Utsumi, 2000), is still at its infancy as a computational task. Detection is difficult because literal meaning is discounted and secondary or extended meanings are instead intentionally profiled. In social contexts, one's ability to detect sarcasm relies heavily on social cues such as sentiment, belief, and speaker's intention. Sarcasm is mocking and often involves harsh delivery to achieve savage putdowns, even though it can be also crafted more gently as the accretion of politeness and the abatement of hostility around a criticism (Brown and Levinson, 1978; Dews and Winner, 1995). Moreover, sarcasm often couches criticism within a humorous atmosphere (Dews and Winner, 1999). (Riloff et al., 2013) addressed one common form of sarcasm as the juxtaposition of a positive sentiment attached to a negative situation, or vice versa. (Tsur et al., 2010) modeled sarcasm via a composition of linguistic elements, such as specific surface features about a product, frequent words, and punctuation marks. (González-Ibánez et al., 2011) views sarcasm as a conformation of lexical and pragmatic factors such as emoticons and profile references in social media. Most research approaches toward the automatic detection of sarcasm are text-based and consider sarcasm to be as a function of contrasting conditions or lexical clues. Such approaches extract definitive lexical cues as features, where the linguistic scale of features is stretched from words to phrases to provide richer contexts for analysis. Lexical feature cues may yield good results, yet without a precise semantic representation of a sentence, which is key for determining the intended gist of a sentence, robust automatic sarcasm

Proceedings of NAACL-HLT 2016, pages 161–169,
San Diego, California, June 12-17, 2016.

detection will remain a difficult challenge to realize. Accurate semantic modelling of context becomes obligatory for automatic sarcasm detection if social cues and extended meaning are to be grasped.

Encouraging an immediate and very social use of language, social media platforms such as Twitter[1] are rich sources of texts for Natural Language Processing (NLP). Social micro-texts are dense in figurative language, and are useful for figurative analysis because of their topicality, ease of access, and the use of self-annotation via hashtag. In Twitter, language is distorted, often plumbing the depths of bad language (Eisenstein, 2013). Yet due to the presence of grammatical errors liberally mixed with social media markers (hashtags, emoticons, profiles), abbreviations, and code switching, these micro-texts are harder to parse, and parsing is the most commonly used method to obtain a semantic representation of a sentence. The accuracy of state-of-the-art constituency parsers over tweets can be significantly lower than that for normal texts, so social media researchers still largely rely on surface level features. With the recent move to artificial neural networks in NLP, ANNs provide an alternative basis for semantic modelling. In this paper, we perform semantic modelling of sentences using neural networks for the task of sarcasm detection. The paper is organized as follows. Section 2 surveys related works, section 3 outlines methods of data collection and data processing, section 4 describes the recursive SVM model, section 5 describes the neural network model, section 6 & 7 outline our experimental setup and experimental analysis respectively, while section 8 presents a simple sarcastic Twitter bot. Finally, section 9 concludes with a short discussion of future work.

2 Related work

Semantic modelling of sentence meaning is a well-researched topic in NLP. Due to 'bad language' in Twitter and a noticeable drop of accuracy for start-of-the-art constituency parsers on tweets, the semantic modelling of tweets has captured the attention of researchers. To build a semantic representation of a sentence in various NLP tasks such as sentiment analysis, researchers have used syntactic structure to compose a total representation as a function of the word-vector representation of a sentence's parts. (Nakagawa et al., 2010) describes a Tree-CRF classifier which uses a data-driven dependency parser, maltparser[2], to obtain a parse tree for a sentence, and whose composition function uses the head-modifier relations of the parse tree. (Mitchell and Lapata, 2010) and (Mitchell and Lapata, 2008) defined the composition function of a sentence by algebraic operations over word meaning vectors to obtain sentence meaning vectors. (Guevara, 2010) and (Malakasiotis, 2011) formulated their composition function using a set of specific syntactic relations or specific word categories (Baroni and Zamparelli, 2010). (Socher et al., 2011) proposed a structured recursive neural network based on the convolutional operation, while (Kalchbrenner et al., 2014) proposed a convolution neural network (CNN) with dynamic k-max pooling, considering max pooling as function of input length. For sarcasm detection, due to the complexity of the task and the somewhat poorer accuracy of start-of-the-art constituency parsers on tweets, researchers have considered surface level lexical and syntactic cues as legitimate features. Kreuz and Caucci (Kreuz and Caucci, 2007) explored the role of lexical indicators, such as interjections (e.g., "gee" or "gosh"), punctuation symbols (e.g., '?'), intensifiers, and other linguistic markers for e.g. non-veridicality and hyperbole, in recognizing sarcasm in narratives. Tsur (Tsur et al., 2010) noted the occurrence of "yay!" or "great!" as a recurring aspect of sarcastic patterns in Amazon product reviews. Davidov (Davidov et al., 2010) examined the effectiveness of social media indicators such as hashtags to identify sarcasm. Lukin (Lukin and Walker, 2013) proposed a potential bootstrapping method for sarcasm classification in social dialogue to expand lexical N-gram cues related to sarcasm (e.g. "oh really", "no way", etc.) as well as lexico-syntactic patterns. Riloff (Riloff et al., 2013) and Liebrecht (Liebrecht et al., 2013) applied N-grams features to a classifier for English and Dutch tweets and observed that some topics recur frequently in sarcastic tweets, such as schools, dentists, church life, public transport, the weather and so on.

[1] https://twitter.com

[2] http://www.maltparser.org/

In this paper, we investigate the usefulness of neural-network-based semantic modelling for sarcasm detection. We propose a neural network model for semantic modelling in tweets that combines Deep Neural Networks (DNNs) with time-convolution and Long Short-Term Memory (LSTM). The proposed model is compared to a recursive Support Vector Machine (SVM) model based on constituency parse trees.

3 Dataset

Twitter provides functionality to users to summarize their intention via hashtags. Using a user's self-declaration of sarcasm as a retrieval cue, #sarcasm, we have crawled the Twittersphere. Since this simple heuristic misses those uses of sarcasm that lack an explicit mention of #sarcasm, we used LSA-based approach to extend the list of indicative hashtags (e.g.to include #sarcastic, #yeahright etc.). We also harvested tweets from user profiles with a strong bias toward sincerity or (for professional wits) sarcasm. To build our sarcastic data set we aggregated all tweets containing one or more positive markers of sarcasm, but removed such markers from the tweets, while tweets which did not contain any positive markers of sarcasm were considered non-sarcastic. The training dataset of 39K tweets is evenly balanced containing 18k sarcastic data and 21K non-sarcastic data. As a test set, we have created a dataset of 2000 tweets annotated by an internal team of researchers. For purposes of comparison, we also used two different publicly available sarcasm datasets.

Social media contains many interesting elements such as hashtags, profile references and emoticons. Due to the size limitation of tweets, users exploit these elements to provide contextual information. To tightly focus our research question, we did not include sarcasm from the larger conversational context and thus dropped all profile information from the input text. As users often use multi-worded hashtags to add an additional sarcastic dimension to a tweet. we used a hashtag splitter to split these phrasal tags and appended their words to the text.

For the recursive-SVM, we used the Stanford constituency parser[3] for parsing tweets. In order to extract maximum information from the parse tree, we used both a pre-processing and a post-processing method which are described below.

3.1 Recursive-SVM Data Processing

Constituency parse trees offer a syntactic model of a sentence which can form a strong basis for semantic modelling. In order to use Stanford constituency parser here, the tweets were first pre-processed by removing social media markers such as profile references, retweets and hashtags. As a tweet may contain multiple sentences, each is split into sentences using the Standford sentence splitter, parsed separately and then stitched back together with a sentence tag (S). Hashtags are dense annotations offered by users of their own texts, and their scope generally applies to the entire content of a tweet. Thus we restored back Hashtags into parse tree by attaching them to the root node of the parse tree of the tweet with a tag (HT). Let's consider the following tweet as example,

```
I love when people start rumors about
me.  #not
```

Hashtag #not is attached to root of parse tree using Part-of-speech tag (HT) (Figure 1).

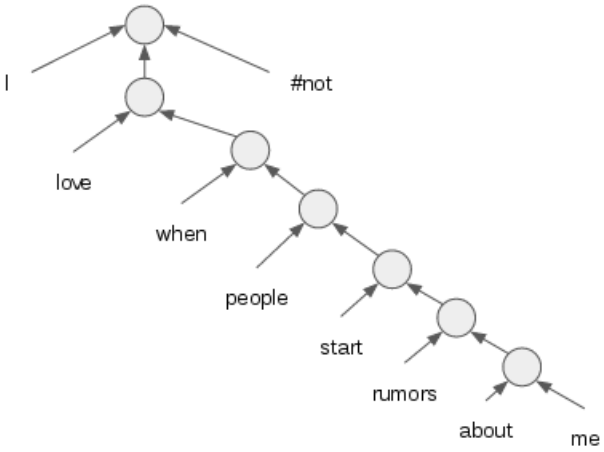

Figure 1: parse tree with Hashtag

4 Recursive SVM

We now define a recursive-SVM model. Consider a subjective sentence (S) containing n phrases with m words in total. w_l, b_l and pos_l denote the surface

[3] http://nlp.stanford.edu/software/lex-parser.shtml

Feature Type	Feature
Node	w_i
Node	$w_i pos_i$
Node	$w_i pos_i b_i$
Node+Edge	$w_i..w_j pos_i..pos_j b_i..pos_j$
Node+Edge	$w_i..w_j pos_i..pos_j b_i..pos_j c_i{+}1..c_j$
Node+Edge	$w_i..w_j pos_i..pos_j b_i..pos_j c_i + 1..c_j o_i + 1..o_j$

Table 1: recursive SVM features

form, root form and part-of-speech respectively of l^{th} word of S, while n_i denotes the i^{th} node and p_i, h_i, and o_i denote phrase, head node and offensive word-marker of the i^{th} node respectively. The 0^{th} node is the root node, while s_i and sa_i denote the predicted values of sentiment polarity and sarcastic polarity of the constituency subtrees whose root is the i^{th} node, ($s_i \in +1, 0$, $sa_i \in +1, 0$). Table 1 shows training vectors ($x_i \in \Re^n$, i = 0, .. , n) where y_i = 1, 0 is the label for the i^{th} node. As the number of parameters is larger than the number of instances, dual-based solvers offer the best fit for this problem. Through grid-search, the optimum penalty value (C) is determined and set to 1000 and 2000 for sentiment and sarcasm detection respectively. The stopping tolerance value was set to -0.0001. Among the variation of different loss functions, L2-regularized L1-loss and L2-loss function yielded the best results.

5 Neural network

Semantic modelling of sentence meaning using neural networks has been a target of attention in the social media community. Neural network architectures, such as CNN, DNN, RNN, and Recursive Neural Networks (RecNN) have shown excellent capabilities for modelling complex word composition in a sentence. A sarcastic text can be considered elementally as a sequence of text signals or word combinations. RNN is a perfect fit for modelling temporal text signals as it includes a temporal memory component, which allows the model to store the temporal contextual information directly in the model. It can aggregate the entire sequence into a temporal context that is free of explicit size constraints. Among the many implementations of RNNs, LSTMs are easy to train and do not suffer from vanishing or exploding gradients while performing back propagation through time. LSTM has the capability to remember long distance temporal dependencies. Moreover, as they performs temporal text modelling over input features, higher level modelling can distinguish factors of linguistic variation within the input. CNNs can also capture temporal text sequence through convolutional filters. CNNs reduce frequency variation and convolutional filters connect a subset of the feature space that is shared across the entire input (Chan and Lane, 2015). (Dos Santos et al., 2015) have shown that CNNs can directly capture temporal text patterns for shorter texts, yet in longer texts, where temporal text patterns may span across 15 to 20 words, CNNs must rely on higher-level fully connected layers to model long distance dependencies as the maximum convolutional filter width for a text is 5 (Figure 2).

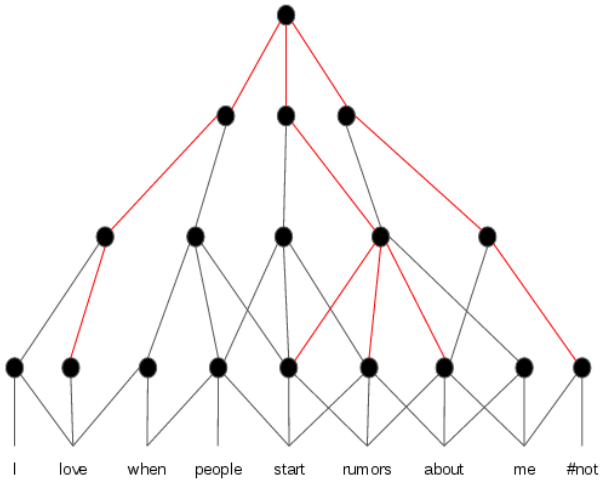

Figure 2: Sentence modelling with CNN

Another major limitation of CNNs is the fixed convolutional filter width, which is not suitable for different lengths of temporal text patterns and cannot always resolve dependencies properly. Obtaining the optimal filter size is expensive and corpus-dependent, while LSTM operates without a fixed context window size. LSTM's performance can be improved by providing better features. Following the proposal of (Vincent et al., 2008), it can be beneficial to exploit a CNN's ability to reduce frequency variation and map input features into composite robust features and using it as an input to a LSTM network. DNNs are appropriate for mapping features into a more separable space. A fully connected

DNN, added on top of an LSTM network, can provide better classification by mapping between output and hidden variables by transforming features into an output space. In the following section we define our proposed network in detail.

5.1 Input layer

Consider a tweet as input containing n words. The tweet is converted into a vector by replacing each word with its dictionary index $s \in \Re^{1\times n}$. To resolve different lengths of input, the tweet vector is padded and the tweet is converted into matrix $s \in \Re^{1\times l}$, where l is the maximum length of tweets in the input corpus. The input vector is fed to the embedding layer which converts each word into a distributional vector of dimension D. Thus the input tweet matrix is converted to $s \in \Re^{l\times D}$.

5.2 Convolutional network

The aim of a convolution network is to reduce frequency variation through convolutional filters and extracting discriminating word sequences as a composite feature map for the LSTM layer. The convolution operation maps the input matrix $s \in \Re^{l\times D}$ into $c \in \Re^{|s|+m-1}$ using a convolutional filter $k \in \Re^{D\times m}$. Each component is computed as follows:

$$c_i = (s * k)_i = \sum_{k,j} (S_{:,i-m+1:i} \otimes F)_{kj} \quad (1)$$

Convolution filter, which has the same dimension D of the input matrix, which slides along the column dimension of the input matrix, performing an element wise product between a column slice *s* and a filter matrix *k* producing a vector component c_i and summed to create a feature map $c \in R^{1(|s|m+1)}$. f filters create a feature map $C \in R^{f(|s|m+1)}$. We chose *Sigmoid* for non-linearity. Initially we passed the output of the convolutional network through a *pooling* layer and *max-pooling* is used with size 2 and 3. Later, we discarded the *max-pooling* layer and fed the LSTM network with all of the composite features to judge sarcasm, which improved the performance of the model.

5.3 LSTM

RNN has demonstrated the power of semantic modelling quite efficiently by incorporating feedback cycles in the network architecture. RNN networks include a temporal memory component, which allows the model to store the temporal contextual information directly in the model. At each time step, it considers the current input x_t and hidden state h_{t-1}. Thus the RNN is unable to plot long term dependencies if the gap between two time steps becomes too large. (Hochreiter and Schmidhuber, 1997) introduced LSTM, which is able to plot long term dependencies by defining each memory cell with a set of gates $\Re^d$, where d is the memory dimension of hidden state of LSTM, and it does not suffer from vanishing or exploding gradient while performing back propagation through time. LSTM contains 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 on the memory update mechanism. Equation (3) and (2) denote the amount of information to be discarded or to be stored from and to store in memory. Equation (5) denotes the output of the cell c_t.

$$i_t = \sigma(W_i[h_{t-1}, x_t] + b_i) \quad (2)$$

$$f_t = \sigma(W_f[h_{t-1}, x_t] + b_f) \quad (3)$$

$$q_t = tanh(W_q[h_{t-1}, x_t] + b_q) \quad (4)$$

$$o_t = \sigma(W_o[h_{t-1}, x_t] + b_o) \quad (5)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot q_t \quad (6)$$

$$h_t = o_t \odot tanh(c_t) \quad (7)$$

5.4 Deep Neural Network Layer

The output of LSTM layer is passed to a fully connected DNN layer, which produces a higher order feature set based on the LSTM output, which is easily separable for the desired number of classes. Finally a softmax layer is added on top of the DNN layer. Training of network is performed by minimizing the binary cross-entropy error. For parameter optimization, we have used ADAM (Kingma and Ba, 2014) with the learning rate set to 0.001.

6 Experiment

To evaluate both models, we have tested rigorously with different experimental setups. For the recursive SVM, we employed different sets of feature combinations mentioned in table 1. In the neural network model, we opted for a word embedding dimension

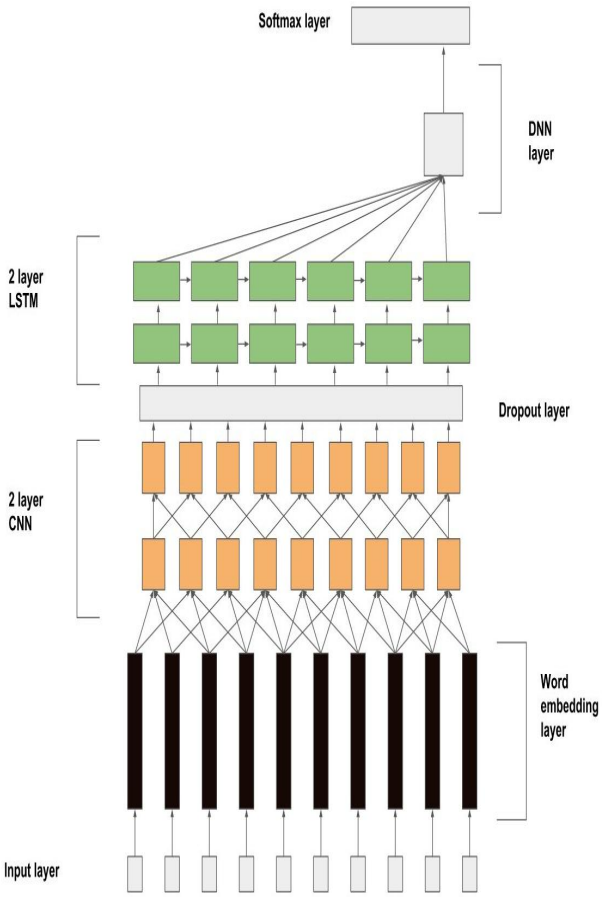

Figure 3: Neural network

set to 256. We tested our model with different settings of the hyperparameters for CNN (number of filter, filter size), LSTM (hidden memory dimension, dropout ratio), and DNN (number of hidden memory units (HMU)). Initially we passed the output of CNN via a *maxdropout* layer, with *maxpooling* size 2 and 3, to the LSTM, but later we dropped the maxpooling layer, which improved the performance by 2%.

In our experiment, apart from the combination of CNN, LSTM, and DNN, we observed the performance for each of the neural networks individually. The CNN network is investigated by varying the number of filters and the filter widths, set to 64, 128, 256 and 2, 3 respectively. For the LSTM network, the number of memory units is varied from 64 to 256. *Sigmoid* is chosen as activation function for both networks. We used Gaussian initialization scaled by the fan-in and the fan-out for the embedding layer and Gaussian initialization scaled by the fan-in for the CNN, the LSTM, and the DNN layer as initial probability distribution. The code was implemented using *keras*[4] library.

7 Experimental Analysis

In the neural network, success depends on the apt input and the selection of hyperparameters. As we observed that the inclusion of hashtag information in the recursive-SVM method gained a better F-score, we pertained the same input structure for the neural network. Apart from difficulties in training a neural network, enormous training time is another passive obstacle. We observed that compared to stacked LSTM network, the CNN-LSTM network converges faster as CNN reduces frequency variation and produces better composite representation of the input to the LSTM network. Sarcasm detection is considered a complex task, as very subtle contextual information often triggers the sarcastic notion. Thus we noticed that the inclusion of a dropout layer on top of the CNN layer, our model suffered a decrease in performance. In the testing dataset, we observed an interesting example.

```
I don't know about you man but I love
the history homework.
```

With the dropout layer, model identified above mentioned example as non-sarcastic, yet without the dropout layer, our model labeled it as sarcastic. This indicates that the word "man", which functions as an intensifier of sarcasm in this context, was dropped out from the output of the CNN layer. Also we observed that incrementing the filter width of the CNN layer boosted the performance of our model by a small margin. To obtain the apt network size, we have also trained with bigger network sizes and larger filter widths, but no improvement has been observed. Table 2 contains the experimental results over our dataset.

Sarcasm is a very subjective phenomenon. Even for the human annotators, it was quite hard to decide if the speaker was sarcastic or not. It was interesting to observe the performance of our model when human annotators interpreted differently. Since our

[4]http://keras.io/

Model	Feature/Hyper parameter	Precision	Recall	F-score
recursive SVM	BOW + POS	.719	.613	.663
recursive SVM	BOW + POS + Sentiment	.722	.661	.691
recursive SVM	BOW + POS + Sentiment + HT-splitter	.743	.721	.732
CNN + CNN	filter size = 64 + filter width = 2	.838	.857	.847
	filter size = 128 + filter width = 2	.842	.86	.854
	filter size = 256 + filter width = 2	.855	.879	.868
	filter size = 64 + filter width = 3	.839	.854	.847
	filter size = 128 + filter width = 3	.856	.879	.868
	filter size = 256 + filter width = 3	.861	.882	.872
LSTM + LSTM	hidden memory unit = 64	.849	.816	.832
	hidden memory unit = 128	.854	.871	.862
	hidden memory unit = 256	.868	.89	.879
CNN + LSTM + DNN (with dropout)	filter size = 256 + filter width = 2 + HMU = 256	.899	.91	.904
CNN + LSTM + DNN (without dropout)	filter size = 256 + filter width = 2 + HMU = 256	.912	.911	.912
CNN + LSTM + DNN (without dropout)	filter size = 256 + filter width = 3 + HMU = 256	.919	.923	.921

Table 2: Experimental Results

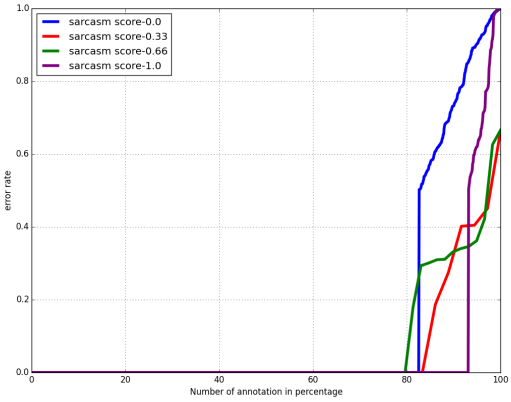

Figure 4: Performance evaluation of model

dataset contains 3 annotations per tweet, we obtained 4 different values for an average sarcasm score from the annotations. We divided the dataset based on the average sarcasm score and observed the performance of the model in each section. From figure 4, we observed that our model performed better for distinct sarcastic data than distinct non-sarcastic data. For dicey examples of sarcasm, where the average sarcasm score is between .7 and .3, our model performed better with non-sarcastic data than sarcastic data.

dataset	Model	P	R	F1
riloff	riloff method	.44	.62	.51
	CNN + LSTM + DNN + filter size = 256 + filter width = 2	.882	.861	.872
	CNN + LSTM + DNN + filter size = 256 + filter width = 3	.883	.879	.881
tsur	SASI	.912	.756	.827
	CNN + LSTM + DNN + filter size = 256 + filter width = 2	.878	.901	.889
	CNN + LSTM + DNN + filter size = 256 + filter width = 3	.894	.912	.901

Table 3: Comparison with other datasets

We evaluated our system with two publicly available datasets (Tsur et al., 2010; Riloff et al., 2013). The results are mentioned in table 3. We observed

that our model has performed with a better f-score than both of the systems, but it has a lower precision value than SASI (Davidov et al., 2010).

8 Twitter Bot

In NLP research, building a carefully crafted corpus has always played a crucial role. In recent research, Twitter has been used as an excellent source for various NLP tasks due to its topicality and availability. While sharing previous datasets, due to copyright and privacy concerns, researchers are forced to share only tweet identifiers along with annotations instead of the actual text of each tweet. As a tweet is a perishable commodity and may be deleted, archived or otherwise made inaccessible over time by their original creators, resources are lost in the course of time. Following our idea of retweeting via a dedicated account (@onlinesarcasm) to refrain tweets from perishing without copyright infringement, we have retweeted only detected sarcastic tweets. Regarding the quality assurance of the automated retweets, we observed that a conflict between human annotation and the output of the model is negligible for those tweets predicted with a softmax class probability higher than 0.75.

9 Conclusion & Future work

Sarcasm is a complex phenomenon and it is linguistically and semantically rich. By exploiting the semantic modelling power of the neural network, our model has outperformed existing sarcasm detection systems with a f-score of .92. Even though our model performs very well in sarcasm detection, it still lacks an ability to differentiate sarcasm with similar concepts. As an example, our model classified *"I Just Love Mondays!"* correctly as sarcasm, but it failed to classify *"Thank God It's Monday!"* as sarcasm, even though both are similar at the conceptual level. Feeding the model with word2vec[5] to find similar concepts may not be beneficial, as not every similar concept employs sarcasm. For example, *"Thank God It's Friday!"* is non-sarcastic in nature. For future works, selective use of word2vec can be exploited to improve the model. Also performing a trend analysis from the our twitter bot can also benefit the system to separate the semantic space of sarcasm and non-sarcasm more efficiently.

[5]https://code.google.com/archive/p/word2vec/

Acknowledgments

This research is supported by Science Foundation Ireland (SFI) as a part of the CNGL Centre for Global Intelligent Content at UCD (Grant No: CNGLII, R13645).

References

Marco Baroni and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193. Association for Computational Linguistics.

Penelope Brown and Stephen C Levinson. 1978. Universals in language usage: Politeness phenomena. In *Questions and politeness: Strategies in social interaction*.

William Chan and Ian Lane. 2015. Deep convolutional neural networks for acoustic modeling in low resource languages. In *IEEE International Conference on Acoustics, Speech and Signal Processing*.

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.

Shelly Dews and Ellen Winner. 1995. Muting the meaning a social function of irony. *Metaphor and Symbol*, 10(1):3–19.

Shelly Dews and Ellen Winner. 1999. Obligatory processing of literal and nonliteral meanings in verbal irony. *Journal of pragmatics*, 31(12):1579–1599.

Cıcero Nogueira Dos Santos, Bing Xiang, and Bowen Zhou. 2015. Classifying relations by ranking with convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, volume 1, pages 626–634.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *HLT-NAACL*, pages 359–369.

2007. *Irony in language and thought: A cognitive science reader*. Psychology Press.

Deanna W. Gibbs and Herbert H. Clark. 1992. Coordinating beliefs in conversation. *Journal of Memory and Language*.

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.

Emiliano Guevara. 2010. A regression model of adjective-noun compositionality in distributional semantics. In *Proceedings of the 2010 Workshop on GEometrical Models of Natural Language Semantics*, pages 33–37. Association for Computational Linguistics.

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

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *arXiv preprint arXiv:1404.2188*.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

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*.

CC Liebrecht, FA Kunneman, and APJ van den Bosch. 2013. The perfect solution for detecting sarcasm in tweets #not.

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.

Prodromos Malakasiotis. 2011. *Paraphrase and Textual Entailment Recognition and Generation*. Ph.D. thesis, Ph. D. thesis, Department of Informatics, Athens University of Economics and Business, Greece.

Jeff Mitchell and Mirella Lapata. 2008. Vector-based models of semantic composition. In *ACL*, pages 236–244.

Jeff Mitchell and Mirella Lapata. 2010. Composition in distributional models of semantics. *Cognitive science*, 34(8):1388–1429.

Tetsuji Nakagawa, Kentaro Inui, and Sadao Kurohashi. 2010. Dependency tree-based sentiment classification using crfs with hidden variables. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 786–794. Association for Computational Linguistics.

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*, pages 704–714.

Richard Socher, Jeffrey Pennington, Eric H Huang, Andrew Y Ng, and Christopher D Manning. 2011. Semi-supervised recursive autoencoders for predicting sentiment distributions. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 151–161. Association for Computational Linguistics.

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*.

Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. 2008. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM.