Exploring Sarcasm Detection on News Headlines

Zilin Wang
The Ohio State University
Computer Science & Engineering
wang.10455@osu.edu

Xinyue Li
The Ohio State University
Computer Science & Engineering
li.8066@osu.edu

Abstract

The use of sarcasm has always been a great obstacle to the success of sentiment analysis models. To achieve human-level accuracies on these models, it is a pivotal next step to lay emphasis on teaching them to recognize sarcasm. In this report, we first propose two sarcasm detection models that are trained to detect sarcasm with merely the headlines. Then, we compare the performance of our models with state-of-the-art models as well as each other. After careful analysis, we propose a more novel model that not only considers the news headlines but also the according main articles. The best result we get on the development set is 81.53% accuracy.

1 Introduction

It is observed that people are used to utilizing sarcasm and expressing their true inner thoughts with opposite words. Also, Sarcastic sentences often have apparent contradictions. For example, in the sentence "I love the pain present in the breakups" from (Poria et al., 2017), the word "love" has a potentially conflicting meaning against the word "pain". Thus, it is no doubt that a kind of sarcasm is used here. Although current sentiment analysis models are powerful, they are highly likely to make mistakes on these sarcastic sentences that require deeper and subtler semantic understandings. Thus, we try to train computers to understand the deeper semantic meaning and recognize potential sarcasm. Specifically, the two models used are N-gram and Recurrent Neural Network fed with multiple word embeddings.

2 Related Work

In the field of sarcasm detection, there are some related works of high quality. (Ptácek et al., 2014) and (Joshi et al., 2015) focused on the traditional n-gram based approach with sentiment feature. They trained their model based on the same three datasets. Two of the datasets are in English and the other dataset is in Czech. (Ptácek et al., 2014) utilized a set of features that are not dependent on any given language, such as word-shape patterns and punctuation features. (Joshi et al., 2015) proposed to harness the context incongruity based features. Similar to (Ptácek et al., 2014), they used punctuation features as well. More recently, (Poria et al., 2017) compared the performance of using merely deep convolutional neural networks for classification and another model that extracts features from tht fully-connected layer of the convolutional neural network and feeds them to a support vector machine for classification. Table 1 contains the results of previously mentioned works.

3 Data and Tools

We use a dataset from Kaggle, called *News Headlines Dataset For Sarcasm Detection*. There are approximately 30,000 samples and each of them consists of three value fields: news headline, the link to the main article and a label indicating whether the headline is sarcastic or not. For the first part, we only use the headlines themselves. The library we use is Tensorflow.keras, and we train our model on our own GPU. By fully utilizing GPU, our training time per epoch has decreased from 50 seconds to 4 seconds, which helps increase our training efficiency greatly.

4 Preprocessing

Before implementing the model, we do some data preprocessing. Given that our data is the news headlines, there are no spelling errors, and it's less sparse than other common datasets that collect data from Twitter. During preprocessing, we remove URLs, stopwords and part of punctuations. We

Work	Dataset1	Dataset2	Dataset3	D3 on D1
(Ptácek et al., 2014)	94.66%	92.37%	63.37%	53.02%
(Joshi et al., 2015)	65.05%	62.37%	60.80%	47.32%
(Poria et al., 2017)	97.71%	94.80%	93.30%	76.78%

Table 1: Results of sarcasm detection works from other researchers. Legenda: D3 on D1 is the result of model trained on dataset3 but evaluated on dataset1

keep some of the punctuations because they can be helpful for classification, especially the double quotes. (Ptácek et al., 2014) and (Joshi et al., 2015) also used punctuation-based features in their model.

Our vocabulary size of the training set is 27321, including an additional count for any unknown word that we may encounter in development set. The maximum headline length in the training set is 48. To set up the experiment, we use a random order to split data into 70% for training and the left 30% for development.

5 N-gram with Logistic Regression

The first model that comes up in our mind is the simple and the most intuitive model in binary classification problems – logistic regression classifier with n-gram features. We choose to mainly use unigram features and only consider using bigram features when the first word of the bigram is a strong negation word, such as "not" and "ain't". This feature space is powerful enough to capture deep semantic meanings and easy to train. Specifically, we build an indexer with all the training news headlines, by mapping every word present in the headlines to different non-negative integers. The length of the weight vector in this case is equal to the vocabulary size(length of indexer) plus one(bias term). And the feature vector of each headline is of the same length with weight vector. Since the number of features in a single sentence is almost negligible, feature vectors are sparse. Thus, we take advantage of their sparsity by using CSR matrix provided in one of python's library SCiPy.

A challenge we encounter while training this model is overfitting. After training for 20 epochs, the model achieves nearly 100% accuracy on training set examples. However, the classification accuracy on the separate development set is only a little over 70%. In other words, although the empirical risk of our model is ideal, the excess risk is too high. To solve this problem, we append a L2-norm regularization term to our model's loss function, converting this empirical risk minimization approach to the regularized loss minimization approach. By doing this, our model's complexity is reduced, and thus overfitting is prevented. As a result, the accuracy on the training set drops to 81.74%, but the development set accuracy increases to 76.23%.

6 Proposed Neural Network Model

Other than the traditional n-gram model with logistic regression, we build a neural network model to do more non-linear classification. We train a word embedding model for our dataset that can minimize the loss of semantics during feature extraction and generation of vector representation from headlines. We also utilize some popular word-embedding models, such as word2vec and glove(Pennington et al., 2014), for performance comparison. After data preprocessing, a news headline would be converted to a list containing selected tokens from the original headline string. To make it machine-readable, the list of tokens would be encoded to a list of non-negative integers by mapping each token to the index of it in the vocabulary. After encoded, we pad the vector by appending extra zeros to the end, so that is has the same length as the longest vector. These extra zeros will not influence the model's performance since the model will learn the triviality of them after enough training. Then, the vectors are ready to be fed into our neural network model as features.

We build the neural network model by the Sequential model from Tensorflow. We choose to use a sequential model because it is featured by a linear stack of layers. In other words, each layer has only one input vector or matrix and only on output, which is exactly the case in our model. The first layer we utilize is the embedding layer. What this embedding layer does is to convert the vector representation of each headline into a two dimensional matrix representation. This is mainly done by the word embedding model mentioned in the previous paragraph. The parameters of the selected word embedding model would be passed into this embedding layer as weights. We will discuss more about the word embedding model in the following paragraphs. By the word embedding model, each integer in the input vector of a headline will be converted to a calculated vector representation. Thus, the vector representation of a headline will become a 2 dimensional matrix after this layer.

Next, the matrix representation will be passed into a bidirectional LSTM layer and a GRU layer. GRU, short for gated recurrent unit, is a newer version of LSTM, short for long short-term memory. We set both layers to have 200 hidden units. Initially, we only use the bidirectional LSTM layer. The bidirectional GRU layer is added as an effort to further improve performance and as an experiment. One benefit of these two layers is that they help to solve the gradient vanishing problem. The longest headline in the dataset has almost 50 words, and normal recurrent neural networks are highly likely to "forget" about the information in the beginning. In other words, the earlier layer would not learn due to the shrinked gradient. The cell state and forget state in LSTM and hidden state in GRU help our model to focus on important words, increasing the chance to recognize contradictions or other clues for sarcasm. The reason why we use bidirectional versions of them is to preserve information from both past and future and thus understand further semantic meanings. To prevent from overfitting our model, we set dropout rates of 0.3 in both layers.

At the end, we adopt a deeply connected Dense layer for the final classification. The labels provided in the dataset are either 0 for non-sarcastic or 1 for sarcastic. Thus, we choose to use the binary cross entropy loss as our loss function. Although cross entropy loss is convex for a multilayer neural network as a whole, it is not guaranteed to be convex in the middle layers. Hence, we use Adam optimizer to update our parameters because of its attractive benefits even with non-convex problems.

6.1 Word2Vec

We first use a publicly available word embedding model called word2vec. It is pre-trained on 100 billion words from Google News. The embedding dimension we use is 300, meaning every word is evaluated from 300 aspects and there are 300 integers in the vector representation of each word. We form a weight matrix with a shape of (vocabulary size, embedding dimension) from the word2vec model. Then, we pass this matrix into the embedding layer of our neural network model as weights. Since words' meanings often change in different contexts, we believe that the word2vec model can not accurately reflect the true meanings of every word in our context, even though the word2vec model is well-trained. Hence, we set this weight matrix as trainable so that it can be fine-tuned to better fit our dataset.

6.2 GloVe

In addition to word2vec, we also use the GloVe(Pennington et al., 2014) model. It is also a publicly available model proposed by researchers in stanford and it is pre-trained on aggregated global word-word co-occurrence matrix. Compared to word2vec, GloVe puts more emphasis on the co-occurrence statistics between each word rather than representing a word with different features. We believe these numbers are not likely to change much, so we treat the weight matrix passed into the embedding layer statically.

6.3 Word Embedding Trained by Ourselves

To get a word embedding model that best fits our dataset, we do an experiment where no pre-trained weight matrix is passed into the embedding layer. Instead, the weights of the embedding layer is initialized randomly and is also trained during training time along with other parameters in the whole model. After trying multiple different embedding dimensions, we come to the conclusion that 200 is more than enough to achieve optimal performance.

7 Results

In this section, we present our results of all models we utilize and compare them with state-of-the-art works published by other researchers. Then, we put our results back into the bigger picture and discuss their implications. Figure 1 provides visual comparison between each results.

Before we present our results, we would like to provide some clarifications on our development set accuracy. Since our dataset was collected from news headlines, all examples in this dataset only share one similarity that they are all from formal news reports. In other words, they are not guaranteed to share the same topic in content. After analyzing most of the examples in this dataset, we verify this fact. As a consequence, our models do not learn extra background knowledge from the dataset. So, we mainly compare the development set accuracy with the accuracy (Poria et al., 2017) got on the model trained on dataset3 and evaluated on dataset1. But since their dataset is collected from twitter replies, they are expected to be noisier than ours. Thanks to editors of the news presses, it is easier for a model to achieve better performance on our dataset than (Poria et al., 2017)'s.

For the baseline n-gram model with logistic regression, we achieve a highest accuracy of 76.23% on the development set. This accuracy is a bit lower than what (Poria et al., 2017) got for generalizing their model to a new dataset. Due to the less noisy dataset we have, we think it is ideal to be used as a baseline model, but definitely not the final result we expect. We mainly do experiments on the regularization coefficient, meaning the tradeoff between the empirical risk and the excess risk. With 0.01 regularization coefficient, the development set accuracy was only 66%. And with it smaller than 0.001, the overfitting problem shows up again. Thus, we decide to use 0.001 as our regularization coefficient.

The results we get from our proposed neural network is higher. In the experiment where we use a pre-trained word2vec model, we receive 77.45% accuracy on the development set, which is slightly higher than the accuracy of (Poria et al., 2017). But, again, due to the advantages we have in the dataset, we would say that our model is still not better than theirs. As for the experiment where we use pre-trained GloVe as the word embedding model, the accuracy on the development set increases greatly to 81.53%, which is the highest accuracy we get across all models. We believe the reason behind this is that, as we mentioned before, the co-occurrence matrix of words serves as ground truth that would not change much in any context. As a matter of fact, we expect the word embedding model trained by ourselves to achieve the highest accuracy. However, the result we get from this model is only 77.91%. It is approximately the same as what the similar pre-trained word2vec with fine-tuning achieved, indicating that we may have reached the ceiling of this kind of word representation model.

8 Limitations

Our models that use feature-based word embeddings, such as word2vec and the model trained by ourselves, could hardly be improved. In fact, we believe other models that make classifications

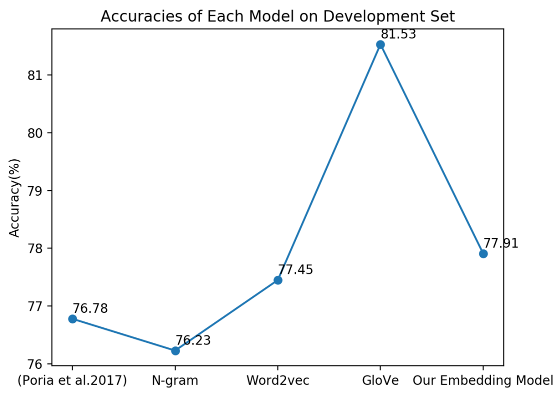

Figure 1: Final results comparison

with merely news headlines are not likely to have significant improvements as well, since sarcasm is usually heavily context-dependent. Here's an example from our dataset: "after careful consideration, bush recommends oil drilling". Without extra political knowledge, there is no way a computer can tell if this is sarcastic or not, not even human. Thus, we need to figure out a new method that allows our model to obtain extra knowledge before making classifications.

9 Future Work

Since the accuracy of sarcasm detection is now at a bottleneck both for us and other researchers, we believe it is necessary to see this problem from another angle. (Bamman and Smith, 2015) tried to detect sarcasm from a Twitter post. What they did was to extract information not presented in a post, such as personal profile information and author previous topics, and include them as features. Inspired by them, we propose to treat sarcasm detection as a question-answering problem rather than a classification problem for our dataset. Apart from news headlines and labels, our dataset provides links to the full news articles as well. For a given news headline, its main article provides important context for the headline. Thus, it could play an important role in detecting sarcasm from headlines.

The language representation model we are planning on using is BERT, proposed by Google AI Language(Devlin et al., 2019). BERT development group (Wolf et al., 2020) provides a great number of powerful pre-trained BERT models that could be easily fine-tuned with just one additional output layer to be generalized to a wide range of other tasks. We have already managed to answer superficial questions given any news main article. What our BERT model does first is to find the sentence segment that is most likely to derive the correct answer. And then infer the correct answer from the sentence segment. However, the problem is that the news articles hardly contain information about whether their headlines are sarcastic or not. As a result, our accuracy achieved by this method is not ideal. But we believe it is a great direction to further improve sarcasm detection rate that is worth future devotion.

10 Conclusion

In this work, we compared different approaches to detect sarcasm in news headlines. First, we implemented an n-gram model as the baseline. Then, we proposed more complex recurrent neural network models that used different word embedding models that are either pre-trained or trained by ourselves. By analyzing the results, we found disadvantages and limitations of regarding sarcasm detection as a classification problem. In future works, we would like to change our viewpoints and try to solve this problem by question-answering approaches.

References

David Bamman and Noah A Smith. 2015. Contextualized sarcasm detection on twitter. In *Ninth international AAAI conference on web and social media*. Citeseer.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding.

Aditya Joshi, Vinita Sharma, and Pushpak Bhattacharyya. 2015. 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)*, pages 757–762, Beijing, China. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Soujanya Poria, Erik Cambria, Devamanyu Hazarika, and Prateek Vij. 2017. A deeper look into sarcastic tweets using deep convolutional neural networks.

Tomás Ptácek, Ivan Habernal, and Jun Hong. 2014. Sarcasm detection on czech and english twitter. pages 213–223.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.