SI630 Project Proposal

Version 1.0

Xia Yan
Jianshu Li

Abstract

We're working on the problem: Irony detection in English tweets. The main tasks are detect and categorize the type of irony. The model we use is based on Bi-LSTM with pre-trained embeddings.. And our model do perform quite well with high f1 scores, 0.70 for multiple classification and for 0.92 binary classification. Those scores are very close to the top tevels in the SemEval2018 Task3, This implicates the importance of pre-processing of strings since our model are simpler but still get close scores.

1 Introduction

The tremendous increase in using social media stimulate the using of irony. However, traditional methods of natural language processing struggles in achieve high performance in detecting irony. Our project targets at detecting the irony on Twitter and further categorize them in different types at a high performance and also trying to generate some ironic tweets automatically. Currently, lots of other group are still using models like SVM, CNN and etc, whose performances are limited when it comes to classify sentence of different lenghts. In this project, we referenced some previous works and try to apply Bi-LSTM with pre-trained embedings to meet our target. And we are using the output from the last unmasked timestamp, which may improve our performance further.

Some social media companies like Twitter, Reddit or IMDB and their users might might be interested in our project to detect and to delete irony on the internet. Furthermore, some linguistic organizations might modify our models to automatically generate ironic or other kind of sentences.

Our score in multiple classification task is close to the top ones in the competetion in 2018 and our score in binary classification is even higher. The results implies that Bi-LSTM model is reasonable and suitable for irony detecting. However, since our f1 score is binary task is surprisingly high, maybe there're some unfair causes.

2 Problem Definition

This problem has two main tasks. Task A is to detect whether a tweet is ironic or not. The input would be tweet strings and output would be labels '0' or '1' which indicate not-ironic and ironic respectively.

Task B will further give outputs categorized by the irony type with labels from '0' to '3', representing i) verbal irony realized through a polarity contrast, ii) verbal irony without such a polarity contrast (i.e., other verbal irony), iii) descriptions of situational irony, iv) non-irony.

And we also managed to write a modified model which can automatically generate ironic tweets.

3 Data

Our problem is to detect and categorize the irony tweets. We got the all the data from Github of given by SemEval-2018 task 3. The datasets is composed with three parts, the training datasets, the test datasets and also the examples called goldtest and also a very small dataset trial for both task A and B. Note that all those data are originally obtained from twitter by detecting the hashtags #irony, #sarcasm and #not
Here's an example from Task A's gold test set:

Label: 1

Irony detection in Tweets	(Van Hee et al., 2018)	https://competitions.codalab.org/competitions/17468

Table 1: We choose the eighth task, Irony detection in Tweets, as our project's topic.

Tweet: Exam na jud. Merry christmas. ||#Sarcasm|#KillUsSlowly

And one from the gold test set for Task B:

Label: 3
Tweet: If you wanna look like a badass, have drama on social media #not

Actually the tweet content of Task A and B are identical for both training and test with different label methods. The traing data has 3834 tweets and the test one has 784. And the distribution of classifications is listed in table 2.

	0	1	2	3
Train A	1923	1911		
Train B	1923	1390	316	205
Test A	473	311		
Test B	473	164	85	62

Table 2: The distribution of the classification of data.

Some of the most frequent features are hashtags(#), urls(http://), users(@) and emojis. I would say that the data is not that clean since people sometimes use repeated characters or strangely capitalized words in tweets. And there're so many hashtags, urls and etc which are almost unique. So we do some pre-process to clean the dataset up.

4 Related Work

The competition requires papers from every competitor to introduce how they do the classification. We found 4 works from teams who ranked high in the leaderboard. There are two teams using the Long Short-Term Memory (LSTM). THU_NGN (Wu et al., 2018) built "a system based on a densely connected LSTM network with multi-task learning strategy". NTUA-SLP (Baziotis et al., 2018) initialized with word2vec (Tomas Mikolov, 2013) word embedding, pretrained on English tweets and then used deep-learning system (Bi-LSTM). They will train the data both on the character-based and sentence-based.

There are also other methods other than LSTM. For example, NIHRIO team (Rohanian et al., 2018) used MLP models. They compared the two models and thought that MLP is the best without the ironic hashtags. To test it, they tokenize the sentences and we follow some of the methods used. Team WLV (Vu et al., 2018) applied supervised system using an ensemble soft voting classifier with "logistic regression (LR) and support vector machine (SVM) as component models". They "define[d] sentiment patterns of Rise (R), Fall (F), and Stable (S) on a word-by-word basis and encode[d] this information in a vector representing the number of S, R, F, RF, and FR patterns."

5 Methodology

We follow the previous works and use LSTM as our main models. We use the pretrained 310 dimension word vectors provided by team NTUA-SLP to see how effective the word vector works.

And we also use the package ekphrasis(Baziotis et al., 2017) to do pre-processing for tweets. Strings represent urls, emails, users and etc will be replaced by '¡url¿', '¡hashtag¿', '¡/hashtag¿', '¡user¿' and etc. Besides, those repeated words will be repalced by tag ¡repeated¿

The model we use for classification is based Bi-LSTM with one linear layer. We choose MultiLabelSoftMarginLoss as the loss function of Task B and CrossEntropyLoss as the loss function of Task A.

The model we use for irony generating is based on traditional one direction LSTM. Since we still apply the embedding layer, the loss function is the cosine distance, which is modified from CosineSimilarity to fit the output in the range of [0, 1]. And the scalar loss of a batch is calculated simply using mean.

In the future, we might further modified the generating model. We can reduce the embedding size to accelerate the generating process and choose word with some randomness proportional to the similarity.

6 Evaluation and Results

We'll basically evaluate the models on F1 scores while accuracy will also be compared. We would like to set our goals as 0.60 for task A and 0.4 for Task B, which are better than $2/3$ of the results on the original competition's leader board.

6.1 Baseline

The simplest baseline method is just label with the most frequent labels. Thus the accuracy is 0.60 and the f1 score is only 0.35.

The f1 score of our simple linear Regression model for task A is 0.50, a little bit lower than what we expect but still acceptable since we simply convert tweet to bag of words. It may perform better with pre-trained word embedding.

We also tried NTUA-SLP's baseline model which is based on linear regression. And it gives us a f1 score 0.64 for task A and 0.44 for task B.

6.2 Previous Model

We managed to run the team NTUA-SLP's model. Their model is based on Bi-LSTM with pre-trained embeddings and unweighted average. Their f1 score for TASK A is 0.74 and that for TASK B is 0.54.

6.3 Our Model

Our model is also based on Bi-LSTM and we use the pretrained embeddings of team NTUA-SLP. Though our model is quite simple, it performs surprisingly well. The accuracy and f1 score for TASK A are 0.94 and 0.93. And for TASK B we get accuracy of 0.86 and f1 score of 0.70.

Our performance seems to be way much higher than the all the competitors in the SemEval2018 TASK3, which seems to be not reasonable. I guess that this is because that the test data we're using is much smaller and easier than the real test data in the competition. On the other hand, I don't know how their f1 score is calculated and we're using the micro f1 score for TASK B and binary f1 score for TASK A.

	F1. A	F1. B
Most Freq	0.35	0.35
Our Linear	0.5	
NTUA-SLP Linear	0.64	0.44
NTUA-SLP Bi-LSTM	0.78	0.69
Our Bi-LSTM	0.86	0.70

Table 3: The f1 score of different models.

6.4 Irony Generator

Honestly speaking, our generator doesn't perform well. And I think the main reason is that the training sample is too small. Only 2,222 ironic tweet are used with the train and test datasets combined. On the other hand, since the embeddings we use is too large, whose size is over 80,000, and we have to compute the cosine similarity between the generated embedding with all of them. And this makes the generating process very slow. Besides, we just choose the word with the highest similarity with no randomness, which causes some endless loop. This is one of the generated tweet:

you are that that's the .. #not #not #lol #not #lol #lol #lol #lol

In the future, we may derive a smaller embedding based on the given dataset to improve the generate speed. Also we would try choose from the top similar words randomly with probability proportional to the similarity.

7 Discussion

The problem is focused on the classification and hence we need to build a model to predict the overall probability of being in each label. We use a baseline linear model and a Bi-LSTM model. The method of logistic regression we used receives a relatively lower F1 score, which is 0.5. It is because the system can only find the probability of being in a label for each word regardless of it position and the previous and last words. Therefore, the linear system used by NTUA-SLP will receive a higher score because it considers a char-based and 2-gram word embeddings. This can result in a higher score.

Then, we try a RNN system which uses bi-LSTM. Simply, we linearly combine the hidden layer by using a linear layer. The result of

F1 score is quite high by using our Bi-LSTM, which is 0.86 for task A and 0.70 for task B. This is a relatively unexpected high score possibly due to the development data. We use golden data of the SemEval2018 TASK3 and it is not the final test data for the competition. Hence, an overfitting may occur in our Model. To solve this, we may reference the work form the group of NTUA-SLP and stop beforehand.

Finally, our generation task generate a tweet that human beings can distinguish with normal tweets. One of the reasons is the number of the tweets. For generation, we expect a large corpus instead of the existed thousands of tweets for labeling. Therefore, for future work, we can follow instruction offered by the competition and crab the results from Twitter to get more data.

8 Conclusion

For the whole task, the main work we have done is to choose a suitable model to predict the extent of irony for the tweets. The steps are tokenizing the tweets, transforming the words and then train a model for prediction. For the first step, considering more frequency of the hashtags and urls in tweets. We need to use a certain symbol to represent them similar to representation of the end of statement. Moreove, as the corpus is not large enough, we used a pretrained word embedding to transform the words into vectors instead of improve our word embeddings through the training. Finally, the neural network model outperforms the linear model and receives a higher score. The model is trying to analyze the whole sentence instead of just a combination of words.

9 Other Things We Tried

We have tried the generating the tweets by using the most similar words after the previous words. However, the number of words is too large, which is over 30000 and it takes a long time to predict the following word. This may be improved later by using character based. Also, the result of generating is not good enough as provided. It may be iterated after a few word. In fact, through the training of the generating model, the loss function does not decrease a lot because of the small corpus, which has only about 3600 tweets with at most 68 words. Therefore, each epoch get less data than expected and resulting in a bad result. As we discussed in previous sections, one of the solutions is to crab more data from Twitter and judge them by out classifier. Then, we can choose the irony tweets and generate the data.

10 What You Would Have Done Differently or Next

For the limit of time, there are still some experiments that should have been done in our project, on of which is to test the result of using character-based embedding. As the small corpus, word embedding trained from it may not be ideal and one of other solutions is to use character-based embedding. Then, this can be further used in our generating models. In this way, we can compare the two results and get more conclusions.

Also, the next-step work can be finding more data for generating. Due to the limitation of time we have not tried built a corpus by ourselves and find the result of the generation. The work can be done in the future.

11 Work Plan

We plotted a Gantt Plot in Figure 1 by using Excel to show the timeline of our work through the update of the project.

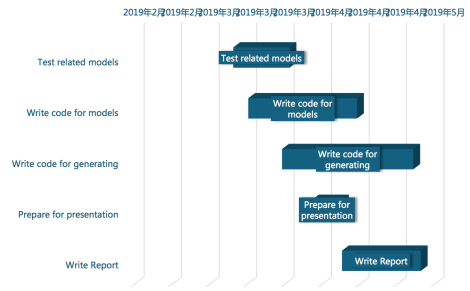

Figure 1: The updated Gantt Graph for our work plan

We planned to start our report two weeks before while we actually started last weekend due to the final exams and projects of other courses. The same thing happenned in the process of generating irony tweets. We should have spent longer time to get more data and try character based to generate a better result. We should have started ealy if given chace to redo the project.

12 Group Effort

We have two persons on this project. From the proposal, we planned to divide works into two parts and each member is responsible for one task from Task A and B. However, the task A and task B are quite similar and do not need to divide into work of two. Therefore, we reassigned the work in which Xia Yan is responsible of building the model for both task A and task B and Jianshu Li is reponsible for building a model for generating the irony tweets, Each one build a complex neural network model and hence we think the overall workload is nearly the same. Also, We two cooperated writing the final report.

Acknowledgments

This project is for SI630 course for School of Information in University of Michigan. Thanks for the instructor Prof.David Jurgens and GSi Heeryung Choi for giving instructions. Thanks for people discussing with us on UMSI EXPO.

References

Christos Baziotis, Nikos Athanasiou, Pinelopi Papalampidi, Athanasia Kolovou, Georgios Paraskevopoulos, Nikolaos Ellinas, and Alexandros Potamianos. 2018. NTUA-SLP at SemEval-2018 Task 3: Tracking Ironic Tweets using Ensembles of Word and Character Level Attentive RNNs. *arXiv preprint arXiv:1804.06659* .

Christos Baziotis, Nikos Pelekis, and Christos Doulkeridis. 2017. Datastories at semeval-2017 task 4: Deep lstm with attention for message-level and topic-based sentiment analysis. In *Proceedings of the 11th international workshop on semantic evaluation (SemEval-2017)*. pages 747–754.

Omid Rohanian, Shiva Taslimipoor, Richard Evans, and Ruslan Mitkov. 2018. WLV at SemEval-2018 Task 3: Dissecting Tweets in Search of Irony. In *Proceedings of The 12th International Workshop on Semantic Evaluation*. pages 553–559.

Ilya Sutskever Kai Chen Greg Corrado Jeffrey Dean Tomas Mikolov. 2013. Distributed representations of words and phrases and their compositionality. *NIPS'13 Proceedings of the 26th International Conference on Neural Information Processing Systems* 2:3111–3119.

Cynthia Van Hee, Els Lefever, and Véronique Hoste. 2018. Semeval-2018 task 3: Irony detection in english tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*. pages 39–50.

Thanh Vu, Dat Quoc Nguyen, Xuan-Son Vu, Dai Quoc Nguyen, Michael Catt, and Michael Trenell. 2018. Nihrio at semeval-2018 task 3: A simple and accurate neural network model for irony detection in twitter. *arXiv preprint arXiv:1804.00520* .

Chuhan Wu, Fangzhao Wu, Sixing Wu, Junxin Liu, Zhigang Yuan, and Yongfeng Huang. 2018. THU_NGN at SemEval-2018 Task 3: Tweet Irony Detection with Densely connected LSTM and Multi-task learning. In *Proceedings of The 12th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, pages 51–56. https://doi.org/10.18653/v1/S18-1006.

A Supplemental Material

Our codes can be found on github https://github.com/Lijs007/si630_final

The dataset can be found on github https://github.com/Cyvhee/SemEval2018-Task3

The pretrained word embedding files can be found on https://drive.google.com/file/d/11zrXc1h_saJsMT6eo0VARKeZuzvK2bU0/view.