Irony Detection on Online Social Media

Team Name: *A team has no name*
Team member: *Vidhyasri Ganapathi, Rahul Pandey, and Arjun Mudumbi Srinivasan*

AIT - 690 - Natural Language Processing
Professor: Dr. Ozlem Uzuner
George Mason University, VA.

With increasing usage of social media, the language has been evolving in accordance. The restrictions and conditions imposed by social media have caused the language to evolve in a manner in which users can express their thoughts in short words. The evolved language also allowed users to mock anything in a subtle and abstract manner by using sarcasm and irony. Sarcasm or Irony is defined as the statement given out which has an intended meaning exactly opposite to what it originally defines. Sarcasm and irony depend on many factors like the way it is uttered/written, the mood of the user, intention of the user, the situation of utterance/writing etc. and many more. Because of such factors involving, it is very difficult for a computer to detect an underlying sarcasm in a text. Irony detection has a large potential for various applications in the domain of text mining such as author profiling, detecting online harassment, and sentiment analysis.

However, the ironic statements cannot be processed and analyzed in the same way as normal statements. For example, the tweet “Love this weather #not” is ironic, but a similar tweet “Hate this weather #not happy” is non-ironic. Different methods are proposed to recognize the irony in texts. Their method can significantly improve classification performance without relying on text processing. However, existing methods rely heavily on the hashtags accompanying the tweet. For example, tweets with #irony or #sarcasm hashtags are very likely to be ironic. Therefore, models may focus on these hashtags rather than the contextual information.

Also, in recent years, many works have been done with the advancement of deep learning. These systems use the word and character embeddings for feature generation (e.g. Amir et al., 2016; Ghosh and Veale, 2016)).

We are proposing two classification model. The first model will classify a single tweet message to determine whether irony is used or not. While the second model will classify the given tweet to the 3 irony types (i.e. verbal irony by means of a polarity clash, situational irony, or another type of verbal irony, see further) or is not ironic.

We have used Twitter data provided by the SemEval 2018 Task 3[1] for building the training model and also validating that with the gold standard test-set. Once we have built a robust training model, we will use this model to evaluate 2015 Reddit Comment dataset, which we have acquired from Kaggle competition[2].

RELATED WORK

Previous work on irony detection mostly applied supervised machine learning mainly exploiting lexical features. Other features often include punctuation mark/interjection counts (e.g Davidov et al., 2010), sentiment lexicon scores (Bouazizi and Ohtsuki, 2016), emoji (e.g. Gonzalez-Ebanez et al., 2011), writing style, emotional scenarios, part of speech patterns (e.g. Reyes et al., 2013), and so on.

Existing methods to detect irony are mainly based on rules or machine learning techniques (Joshi et al., 2017). Rules-based methods usually depend to identify irony (Khattri et al., 2015; Maynard and Greenwood, 2014). However, these methods cannot detect contextual irony. Traditional machine learning based methods such as SVM (Desai and Dave, 2016) are also effective in this task, but they heavily depend on the text processing done prior to the modelling (Barbieri et al., 2014). Recently, deep learning techniques are successfully applied to this task. For example, Ghosh et al. (2016) propose to use a CNN-LSTM model to classify the ironic and non-ironic tweets.

The state of the art result for both classifications is given by Wu et al. (Wu et al., 2018). For the binary classification, they have built a constrained (i.e. only the provided training data were used) system and yielded an F1 score of 70.5%. Their architecture consists of densely connected LSTMs based on (pre-trained) word embeddings, sentiment features using the Affective Tweet package and syntactic features (e.g. PoS-tag features + sentence embedding features). Hypothesizing that the presence of a certain irony hashtag correlates with the type of irony that is used, they constructed a multi-task modelable to predict simultaneously

1) the missing irony hashtag,

2) whether a tweet is ironic or not, and

3) which fine-grained type of irony is used in a tweet.

Similarly, for multiclass classification, they used a slightly altered (i.e. ensemble LR models and concatenated word embeddings instead of averaged) model. They have got the best result for binary classification of irony and in the top 3 for multiclass classification of types of ironies. Still, 0.7 F1 score is low and thus we realize that we have to use some other mechanism to represent these short text messages to capture the higher level semantics like ironies.

APPROACH

For our approach, we will start with the base model that will create Bag of Words features for input tweets and train a machine learning model like SVM, Naive Bayes, Logistic Regression, and other boosting algorithms. For data preprocessing, we are doing data cleaning of tweet messages that include replacing all URLs with "_url_", replacing mentions(@user_name) by "_mentions_", expanding the contracting word (e.g. ain't -> am not), replacing numbers with

"_num_", etc. Finally, our proposed approach will be an efficient representation of the tweet messages using CNN as feature codes and then using the fully connected layer of CNN as a representation feature for the tweet messages. Then finally, we will train the linear classifier with these features to create a robust model. Our hypothesis is that CNN can capture higher-level semantics of words in the message to detect irony. Once our model is built, we will predict the ironies on Reddit comments and do several statistical analysis on them and visualize the results.

The reason we have chosen Reddit comments for evaluation despite the training done on Twitter data is:

1. Reddit and Twitter both are the part of Online Social Media
2. Reddit and Twitter user base are similar
3. They both have a short text representation
4. Reddit comments tend to have more ironic texts as compared with tweets.

DATASET OVERVIEW

Irony detection had 2 tasks:

1. **TaskA:** To classify if the tweets were ironic or not:
2. **TaskB:** To categorize the tweets into the type of irony

This is the overall distribution of the data is as follows. All data are tweet data.

Class	Sub Class	# Instances
Ironic	Verbal irony by means of a polarity contrast	1728
	Other types of verbal irony	267
	Situational irony	401
Not Ironic	Not ironic	2396

It is divided into 3834 training and 784 testing data respectively.

ANALYSIS

We observe that all the tweets were unique since it was part of the training data. Also, we observe there were only 35 instances that had the "RT" word i.e. Retweet Identifier. We thought of keeping it as it may contain information for detecting irony. However, there were 856 instances of URLs (http://, https://), which is a significantly higher amount. Hence, we converted every URLs to "_url_" token identifier to represent all of them so that it will have a sign of whether the URLs are present in the tweet or not. Before the models were implemented we performed a certain analysis which might felt will be very critical in terms of the O/P of the models

Determination of Key Important factors for models:

We wanted to see the factors(here the words present in the tweets) which were responsible for the model performance. As such in the initial stages, we took the TF-IDF with a ngram_range of (1,2) and applied it to the model. Once the bag of word model has been generated we applied scikit-learn's recursive feature elimination for determining the most important features in the model. This lead to the following two outcomes

Presence of stop words:

The most important key features in the tweets using the RFE[3] method was found to stop words like 'not', 'okaaaay','aaaaaand'. These words played a crucial role in determining whether the tweet was ironic or not. For eg: 'I am going to win the lottery #not' is an ironic statement which is based on the sarcasm caused by '#not' here. With the regular data cleansing method like removing the stop words, the word 'not' will be removed which will make the tweet lose the core

sarcasm. As such we decided not to remove any stop words or lemmatize/stem words in order to maintain the core sarcasm and thus will be able to model to learn the sarcasm well

Bigram models:

The key features also revealed that the most important features were all unigrams except for few. As such we reduced the range of the ngram_range to (1,1) while applying the models. This not only reduced the TF-IDF model size but also significantly helped in improving the computation time.

Retweets and Mentions:

There was a suggestion about analyzing the retweets and mentions to improve the accuracy and remove the redundancy. The original dataset already cleaned the redundant retweets and we cleaned the mentions while removing punctuation and others

Task 1 (Whether the tweets was Ironic or not):

As a baseline solution, a bag of words model has been developed. This has been developed using scikit-learn's TF-IDF vectorizer. The TF-IDF vectorizer converts the tweets into one hot encoded vectors with penalized weights. The vectorizer converted would return a sparse matrix of shape 3834*42000. As such the max_features was set to 200 so that the TF-IDF would return the 200 most frequently used vectors and their weights for each tweet. Now with the sparse matrix of 3834*200 as the training set, simple models like SVM, KNN, Decision trees, and Logistic Regression were implemented. The kernel of the SVM was set to linear and the K value for KNN was 1 so as to be simple. Once the model has been trained we had tested it against the gold standard test data to determine the performance. No cross-validation or parametric tuning was performed to maintain simplicity. The accuracy of the Model turned out as follows.

S: No	Model	Accuracy
1	**SVM**	**63.26 %**
2	Logistic Regression	62.12 %
3	K nearest neighbors	53.69 %
4	Decision Tree Classifier	56.63 %

Of all the models we saw that the SVM performed the best. As such that baseline model accuracy would be that of the SVM model (63.26%)

Task 2 (Categorizing ironic tweets):

For task 2, a similar approach was performed for the data. A TF-IDF has been developed and the simple models have been applied. The accuracy of the different models was as follows.

S: No	Model	Accuracy
1	**SVM**	**64.92 %**
2	Logistic Regression	62.62 %
3	K nearest neighbors	57.90 %
4	Decision Tree Classifier	51.40 %

As such the SVM outperformed all the basic simple classifiers. We choose that as the baseline

Error analysis for the baseline :

We checked out the confusion matrix for both the models and the results are as below.

Confusion matrix for task 1

Confusion Matrix	**1**	**0**
1	330	140
0	148	163

Confusion matrix for task 2:

Confusion matrix	1	2	3	4
1	413	60	0	0
2	87	77	0	0
3	70	15	0	0
4	58	4	0	0

As we can see that the in task B the model fails to capture any element from 3 or 4 (due to its linear boundary) yet it outperforms the other classifier. This might be due to

1. A higher number of 1's predicted by the SVM. As such we may even infer that the test set might contain a very high number of 1
2. The baseline SVM model is highly biased and might lead to overfitting for unknown foreseen data like the reddit data

Also for task A. There are an equal number of type 1 and type 2 errors. These can say that the model is still yet to be improved

Interesting Observation on the erroneous data

1. Contracting words
    a. We have observed that many words that are misclassified contain contracting words (I've, You'd, We'll, don't) that need to be expanded to get more relevant

words. We have observed in more than 40 instances out of 276 misclassification in our test data. Some of the examples from the misclassifications are:

Original	Predicted	Text_raw
not_ironic	ironic	The fact that this time next week I'll be getting my wisdom teeth out 😖🐵 #chipmunk
ironic	not_ironic	Don't know if I can handle this crazy 11-2 shift. It's so long I might pass out on the job! #sarcasm
ironic	not_ironic	Decided I'd go to bed "early" for once. I've been laying here since 2 am. Yet the total amount of sleep I've had all week is 10 hrs. #irony

b. We have also observed that #tags contains meaningful information whether the tweet is ironic or not and sometimes these hashtags contains multiple words, which make no senses till it gets separated (e.g. #SoCute #)

Original	Predicted	Text_raw
ironic	not_ironic	My entire career has been devoted to being a BOSS in beating corruption, but I get beat in a corrupt election #Irony #SoCloseTilden #1Vote
ironic	not_ironic	A Puppet at the #PTI rally wanting to make #NayaPakistan #irony http://t.co/3kAny6qhKr
ironic	not_ironic	My favorite thing is waking up at 4 and driving 20 minutes in the snow just to see this bitch ass #Not #PaneraProTeam http://t.co/qebtDSVoRT

c. Our vocabulary size of the dataset is 11,372 in our 3834 tweet training document, which is very less to cover the semantics of each vocab. That means, most of the word appears only once in the tweet so their frequency vector doesn’t

give enough information to capture the intent of the tweet whether it is ironic or not.

Based on this analysis, we have proposed our new approach that will build up on our baseline and will try to improve by tackling all our error analysis.

Gold Standard baseline:

The gold standard baseline for task A and task B are as follows

- **Task 1 : Accuracy :73.47 %**
- **Task 2 : Accuracy : 73.21 %**

We will evaluate our proposed approach with this Gold Standard baselines to see how well our proposed approach will work.

PROPOSED APPROACH- 1

For this approach, we have done following changes:

1. We have created a contraction dict to expand all the contracting words like I've, You'd, We'll, etc. Some of the examples are as follows:
    a. {"ain\'t": "am not", "can\'t\'ve": "cannot have", "\'cause": "because", "could\'ve": "could have", "didn\'t": "did not", "hadn\'t\'ve": "had not have", "he\'s": "he is", "how\'d": "how did", "I\'d": "I would", "it\'ll\'ve": "it will have", "it\'s": "it is", "let\'s": "let us", "ma\'am": "madam", "needn\'t": "need not", "shan\'t\'ve": "shall not have", "weren\'t": "were not", "where\'d": "where did", "won\'t\'ve": "will not have", "you\'re": "you are", "you\'ve": "you have"}

2. We have created a regex to expand the hashtags if there is change of case (#SoCute -> so cute). The regex is
    a. expand_hash = lambda i: " ".join([a for a in re.split('([A-Z][a-z]+)', i) if a])
3. We have tried to use word embeddings to represent word. For that, we have used pre trained word embeddings trained on very large corpus to bring the domain knowledge for our small subset of data for better representation of word if they are present at low frequency in our training data. Also, we believe this will help to understand the semantics of word that are not present in training data but are present in test data and are similar to some word in training data.
    a. We have used two pre-trained word embeddings for our experiments:
        i. GloVe: GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space. (Pennington et al., 2014). For our project, we have used a pretrained model of GloVe trained on 27 billion tweets with 200 dimensions.
        ii. Word2Vec: Word2vec is a group of related models that are used to produce word embeddings. These models are shallow, two-layer neural networks that are trained to reconstruct linguistic contexts of words. (Tomas et. al, 2013). We have used Google's pretrained model of word2vec trained on 100 billion News dataset

b. For representing tweet, we have taken the average of their word embeddings if the words are present in the pretrained model vocabulary.

4. After that, we have trained a different classifier for training the data. We have used now more strong classifier like LightGBM, Logistic Regression, and SVM to analyze our result.

Results:

We have analyzed 3 strong classifiers for our result i.e. SVM, Logistic Regression and one boosting learning algorithm LightGBM and tested with two features: an average of GloVe word embedding and average of word2vec word embeddings. We have also added an an update to the text preprocessing as per our initial error analysis.

Task A

Here is the result of accuracy for different models using the word2vec embeddings as the features for the task A

Classifier	**Accuracy**
SVM	54.84%
LR	62.88%
GBM	62.62%
LGBM	**65.43%**

The confusion matrix for the best performing model(Light GBM) is

Confusion matrix	**Ironic**	**Not-Ironic**
Ironic	293	180
Not Ironic	91	220

Accuracy for the different models using the GloVe embeddings as the features for the task A

Classifier	Accuracy
SVM	67.73%
LR	**68.88%**
GBM	66.58%
LGBM	65.43%

The confusion matrix for the best performing model(Logistic Regression) is

Confusion matrix	Ironic	Not Ironic
Ironic	295	178
Not Ironic	66	245

We have observed that we got more than 4% increase in our performance for task A with Logistic Regression model on GloVe Twitter embedding features with expanded hashtags and contracting words.

Task B

Accuracy of the different models using Word2vec embeddings as the features for task B

Classifier	Accuracy
SVM	60.20%
LR	59.56%
GBM	**61.73%**
LGBM	59.94%

Confusion matrix for the best performing model (GBM) for task B

Confusion matrix	1	2	3	4
1	379	91	2	1
2	68	94	2	0
3	58	16	11	0
4	55	6	1	0

Accuracy for the different models using the GloVe embeddings as the features for the task-B

Classifier	Accuracy
SVM	**62.76%**
LR	62.50%
GBM	**62.76%**
LGBM	**62.76%**

Confusion matrix for the best performing model (GBM) (both word2vec and GloVe) for task B

Confusion Matrix	1	2	3	4
1	369	95	9	0
2	55	106	3	0
3	50	25	10	0
4	52	9	1	0

We observed that we have got low performance with task B. We believe that is because of data imbalance in the training with respect to different classes and each class might have similar word distribution that might have confused the classifier and resulted in bad accuracy.

Error Analysis :

- Comparative Analysis of test data between the baseline approach and proposed approach 1
    1. Total test data: 784
    2. Total miss by our baseline bow svm: 276 (35.2%)
    3. Total miss by our new proposed approach: **244 (31.12%)**
    4. Percentage decrease of error = (35.2%-31.12%) **4.08%**

5. Total data that is recalled from the base model: 132
    a. That explains out of 784 tweets in test data, 132 tweets that were misclassified in the baseline approach were classified correctly in our proposed approach.
6. Total new data miss that was classified correctly by the base model: 100
    a. That explains that 100 new tweets were misclassified by the proposed approach were actually predicted correctly with our baseline approach
7. Total common data, which the classified incorrectly by both model: 144
    a. That explains that 144 tweet data in the test set are misclassified in both the baseline and proposed approach.

- After analyzing the misclassified data statistics, we did into the data and observed the following:
    - Use of #NOT at the end of the sentence
        - It is observed from the misclassified data that the ironic tweets, which has the hashtag #NOT at the end of the tweet are misclassified as non-ironic.
        - It is also observed that, the non-ironic tweets with word 'not' in the middle of the sentences are classified as ironic.
        - That made us think that because of not capturing the order and structure of the tweet text, these errors are caused.
        - Following are some examples for both scenarios:

Original	**Predicted**	**Text**
#NOT at the end of the tweet		
ironic	not_ironic	@RozieBreen @CllrKWakefield @tomriordan @GaryVerity

		@BBCLeedsif we get it right for business the trickle down fairy will do the rest #NOT
ironic	not_ironic	.@ChrisBrett2 @/RBRNetwork1 @GaryBlackal @DiveConLFP I think it's your white skin. Or "racist DNA". He's a geneticist too, did ya know? #not
ironic	not_ironic	Since the tribunal was established 14 years ago, no complaint against the intelligence services has ever been upheld. #trustthesystem #not
ironic	not_ironic	Produce Mobile Apps #not http://t.co/3OV57ZhqcH http://t.co/wX1DbI8W9M
ironic	not_ironic	You decide to go on vacation without me sleep? Reaaaal nice. #considerate #not 😒💤
The word 'not' in the middle of the text		
not_ironic	ironic	Merry Christmas to all those soldiers out there that couldn't make it home! I love you all ❤
not_ironic	ironic	@terry_legg @SR_Duncan @KateOnTheGo It wasn't for one person. It was a business expense.
not_ironic	ironic	At least I don't have to clean the floor
not_ironic	ironic	I'm heading off, but need to say this! "Just because I am concerned for my country, and it's people-Does #NOT make me a racist! #Terrorists
not_ironic	ironic	Access to housing is not a fundamental human right, according to Ontario courts: http://t.co/ykf7vB53RM #onpoli #TOpoli #cdnpoli

- Another issue that could possibly cause misclassification is, if the tweets has sarcasm expressed through images. These images are present when we tried to open the URLs. So, this results in ironic tweets being classified as non-ironic. Some examples are

Original	**Predicted**	**Text_raw**
Tweets with Images		
ironic	not_ironic	@manicsue: You want to see something Irish!!Of all the reasons to be closed 😂😂😂 http://t.co/ePyGvA9pwU #irony
ironic	not_ironic	@ambika1900 @sankrant @CChristineFair @JTMondal @Al_Muwa7id Christian Europe translated Islamic manuscripts #Irony http://t.co/0hXEQig7hM
ironic	not_ironic	@manicsue: You want to see something Irish!!Of all the reasons to be closed 😂😂😂 http://t.co/ePyGvA9pwU #irony
ironic	not_ironic	I may or may not have put @robertashton1 at the top of @swarm_tweets tree soon as he likes Christmas time... #Sarcasm

		http://t.co/OMOURvLfzt
ironic	not_ironic	clashupdate #not http://t.co/1bFOsIwnI5

PROPOSED APPROACH - 2

We saw that the word embeddings as a feature improved the performance of the classifiers and also strong classifiers like ensemble models performed better than the weak classifier. We have achieved **a 4%** increase from our baseline approach in TaskA and got more closer to the existing gold standard accuracy i.e. **73.47%**.

Yet, from the error analysis, we understood that we need a model which would consider the structure of the tweet rather than just that the words present in them. One possible solution to this problem is to get a better representation of the document instead of getting the representation of its individual words. In our second and final approach, we will try to apply an efficient short text representation that utilizes the structure of the short text and also the domain knowledge from the word embedding model. One such approach is using CNN for sentences as a document features and then train these strong classifying model.

PROPOSED APPROACH - 2A

We investigated one approach by extending the word embeddings model by weighted embeddings instead of the regular This approach ensured that the model was just not only dependent on the word meaning but also occurrence of the word too.The weights were determined by using the TF-IDF weights of the words. TF-IDF weights and the embedding vectors have been obtained, the vector is multiplied with the weights and averaged over the count of occurrence to extract the features. Using these features we tried to run a few powerful

classifiers like Logistic Regression, Support Vector machine, LightGBM, etc. The results are reported as follows

Task A

Classifier	Accuracy
SVM	**61.28%**
LR	59.20%
Classifier	Accuracy
SVM	62.76%

This model performed significantly worse than the proposed embeddings approach. We assume that the performance was degraded by using the weights. This suggests that the embeddings were independent of the word count and thus can be ignored. We return to our hypothesis where we believe that the model was dependent on the structure of the document .

PROPOSED APPROACH - 2B - CNN For Sentence Classification

Now for preserving the structure of input documents, we have used the Convolutional Neural Network for Sentence Classification. We have used the following CNN architecture inspired by (Yoon, 2014) and used the last layer as a feature vector for training Logistic Regression Classifier. Fig. 1 shows the architecture of our deep model.

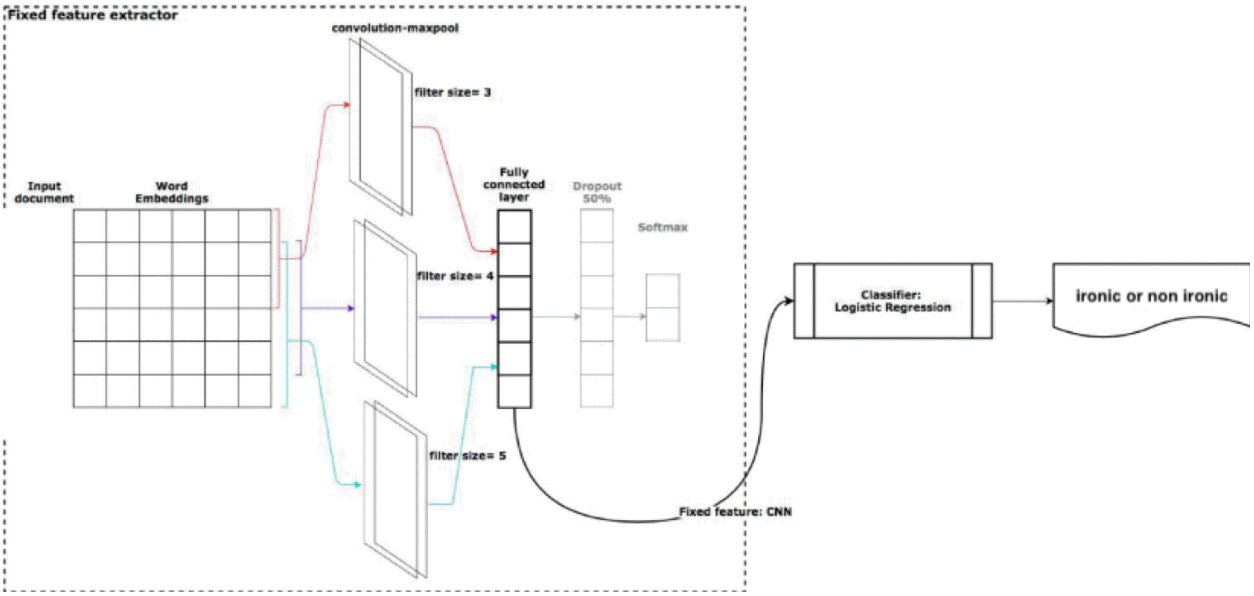

Fig.1. Proposed CNN architecture

For the first layer, we have used GloVe embeddings for the initial word embedding and used tensorflow's embedding_lookup feature to get the embeddings. Then, we have used 3 convolutional maxpool layers, of filter size 3,4, and 5 respectively followed by a fully connected layer, dropout layer with 50% dropout, and finally the softmax layer. We have used ReLu as activation function, Adam Optimizer to reduce cross-entropy loss function. After training, we store the weights and calculate predictions for test data. For each test & train data, we store its fully connected layer output during forward propagation with trained weights. We use these layers value as a fixed feature to represent each test & train data. We then use the train data's fixed feature to train another Logistic Regression classifier and predict the test data using its fixed features. We then compared the results.

Classifier	**Accuracy**
CNN with softmax	35.71%
CNN with LR	**38.76%**
CNN with SVM	37.37%

To our surprise, the CNN based approach performed the worst. We observed that it is due to overfitting since it was giving >99.9% accuracy on training data. We tried early stopping, increasing l2 regularization for non-linearity but we couldn't cross more than 39% in accuracy.

Hence, for our final analysis on Reddit comments, we have used Proposed Approach 1 of Task A i.e. GloVe embeddings with LR classifier to predict Reddit comments and perform our analysis.

ANALYSIS OF REDDIT COMMENTS:

Reddit dataset present in Kaggle was around 31.1 GB in size stored in the SQLite database. It had around **7 billion** comments from different users. We have randomly chosen 20,000 comments to analyze the irony. The new dataset had 20,000 rows and 20 columns. We used the best model (LR- GLoVe embeddings) to see if the comments were sarcastic or not. Following are some analysis done on the Reddit comments.

Number of ironic/ Non ironic comments given by top reddit users

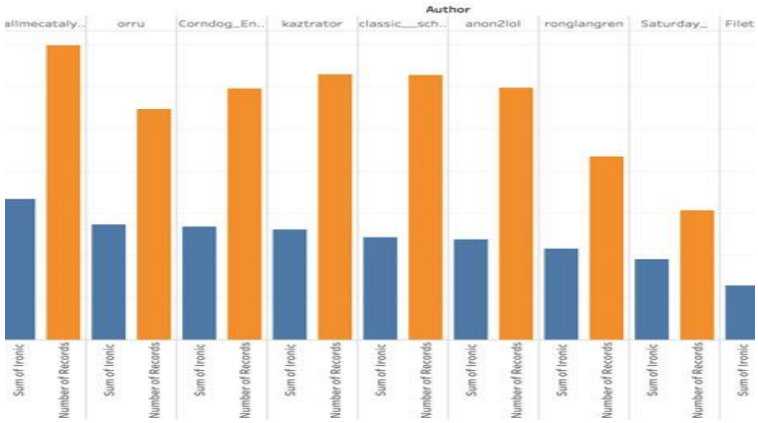

Fig.2

We took the top 10 Reddit commenters and we compared the count of their ironic comments (In graph blue) to their total number of comments(In graph orange). We see that in all the cases, the users only tend to be ironic only half the time and for some people even the lower than half

Number of ironic comments given by most sarcastic users in each section

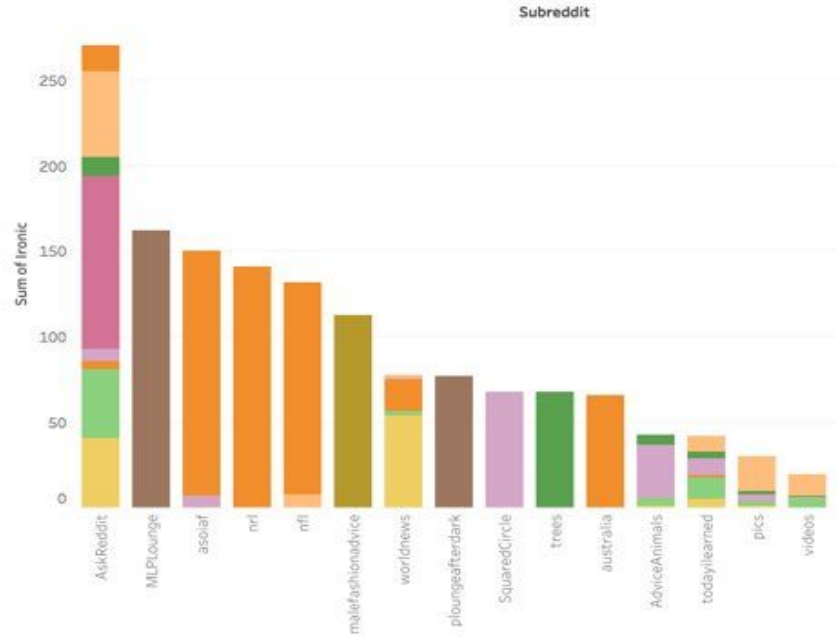

Fig.3

We also took the subreddit sections where the redditors comment. We observe that in generalized topics like 'AskReddit' or simple topics like 'NFL' users tend to be very ironic. This may be because it is easier to create a generalized sarcastic comment rather in a generalized scenario rather than in a specific field where you need to have knowledge of the field to create the sarcasm.

Total upvotes in each subreddit classified by ironic/non-ironic

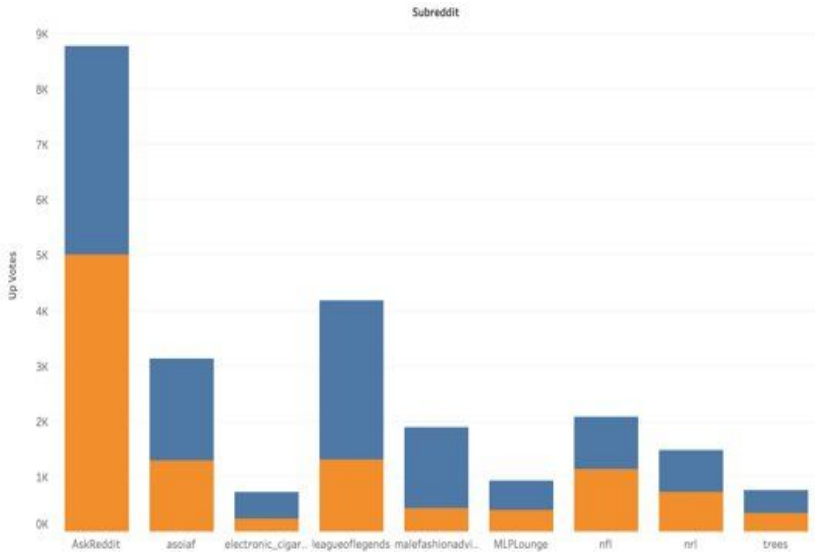

Fig.4

We also observed in a number of upvotes gained by sarcastic comments to number of upvotes gained by non-ironic comments. We see that in the generalized topic, people appreciate sarcasm and irony and thus give high upvotes. In a specified topic, if you are sarcastic, then you are not given many upvotes because of the seriousness of the topic

Word Cloud for Ironic vs non-ironicc comments

Ironic *Non Ironic*

Fig.5

We also checked the Word cloud for sarcastic and non-sarcastic comments. We can clearly observe that the words ending “n’t” are highly sarcastic. As most of the sarcastic comments use won’t, can’t, etc to create the contradiction required for the sarcasm. Words like good also create sarcasm in a certain context.On comment word cloud, we have http as the most frequent word as the user tends to submit a link along with a comment when providing the answer to a serious question.

CONCLUSION

LR with GLoVE embeddings performed the best in Task A with a classification accuracy of 68.88% and F-1 score of 65%. SVM with GLoVE performed the best for Task B with the Classification accuracy of 62% and F-1 score of 61%. The embedded model performed better than the regular Bag of Words models. The simple models performed much better than complex models.

FUTURE RESEARCH

We believe there might be a few methods which can be investigated to improve the sarcasm detection model

1) By using more data to train and test the models on
2) By using more hidden layers in CNN based model (we don't know if the model will perform better or will overfit)
3) By using embeddings specific for the sarcastic comments (None present till now)

REFERENCES

[1] The CodaLab Team. “CodaLab - Competition.”. Retrieved May 3, 2019 (https://competitions.codalab.org/competitions/17468).

[2] The Kaggle Team. “May 2015 Reddit Comments | Kaggle.”. Retrieved May 3, 2019 (https://www.kaggle.com/reddit/reddit-comments-may-2015).

[3] S. Amir, B. C. Wallace, H. Lyu, P. Carvalho, and M. J. Silva. “Modelling Context with User Embeddings for Sarcasm Detection in Social Media,” *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, 2016.

[4] F. Barbieri, H. Saggion, and F. Ronzano, “Modelling Sarcasm in Twitter, a Novel Approach,” *Proceedings of the 5th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2014.

[5] M. Bouazizi and T. Ohtsuki, “Sarcasm Detection in Twitter: ‘All Your Products Are Incredibly Amazing!!!" - Are They Really?,” *2015 IEEE Global Communications Conference (GLOBECOM)*, 2014.

[6] A. Ghosh and D. T. Veale, “Fracking Sarcasm using Neural Network,”

Proceedings of the 7th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, 2016.

[7] A. Joshi, P. Bhattacharyya, and M. J. Carman, "Automatic Sarcasm Detection," *ACM Computing Surveys*, vol. 50, no. 5, pp. 1–22, 2017.

[8] A. Khattri, A. Joshi, P. Bhattacharyya, and M. Carman, "Your Sentiment Precedes You: Using an author's historical tweets to predict sarcasm," *Proceedings of the 6th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, 2015.

[9] A. Reyes, P. Rosso, and T. Veale, "A multidimensional approach for detecting irony in Twitter," *Language Resources and Evaluation*, vol. 47, no. 1, pp. 239–268, 2012.

[10] C. Wu, F. Wu, S. Wu, J. Liu, Z. Yuan, and Y. Huang, "THU_NGN at SemEval-2018 Task 3: Tweet Irony Detection with Densely connected LSTM and Multi-task Learning," *Proceedings of The 12th International Workshop on Semantic Evaluation*, 2018.

[11] J. Pennington, R. Socher, and C. Manning, "Glove: Global Vectors for Word Representation," Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014.

[12] M. Tomas, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. "Distributed

representations of words and phrases and their compositionality." In Advances in neural information processing systems, pp. 3111-3119. 2013.

[13] K. Yoon. "Convolutional neural networks for sentence classification." In arXiv preprint arXiv:1408.5882. 2013.