Humor Computational Detection Learning Project With Machine Learning Models

Leonard Shi (bs4297), and Kexin Deng (kd2474)

***Abstract*—Our project aims to use one-liners collected from the Web using an automatic bootstrapping process to humor computational detection. We use binary classification to determine whether the oneliner is humor or not. In terms of baseline models, we used SVM, logistic regression and decision tree. After several experial experiments, we discovered several more advanced RNN models that better met with our expectation. The three RNN models include Simple RNN layer, LSTM layer, GRU layer on top of embedding layer. Preprocessing tools include Word2Vec and later we used GloVe, and we applied cross-validation and F-measure test to compare and contrast between different models. The aggregated performance of our models turned out pretty interesting. SVM did not yield very good results, but logistic regression performed best among the three base line models. We discovered that Simple RNN layer on top of embedding layer gives the best performance, gaining a accuracy rate of around 95.68%. This result reveals that we could efficiently classify large humor dataset since we are able to tell humors accurately even when we know little about the context of the specific humor or contradicted in our perception of humor.**

***Index Terms*—Humor Detection, SVM, Logistic Regression, Decision Tree, RNN, LSTM, GRU, Word2Vec, GloVe**

I. INTRODUCTION

Humor is an essential element in interpersonal communication. Everyone has different definition of humor, and our perceptions towards humor vary. However, not everyone is born with the ability to detect humor because of their different cultural backgrounds. Given the importance of humor in our everyday life and the increasing importance of computers in our work and entertainment, we believe that studies related to computational humor, changing computers into a creative and motivational tool for human activity. The project makes it easier for people to identify humor, thus boosting communication efficiency and strengthening their bonds with each other by knowing the hidden punchline beneath seeming innocent humor. We approach this final project by introducing a humor detection classifier using machine-learning algorithms that we learned this semester, training the model using a data test (texts and labels), and using the trained model for validating more incoming text and distinguishing the presence of humor.

The significance of the project is to explore how computational humor, since humor isn't merely a way to induce amusement, but also has a positive effect on the mental state. The computational humor changes computers into a creative and motivational tool for human activity.

In this project, we introduce baseline models of SVM, logistic Regression, decision tree, or from the more advanced models like neural network and deep learning through embedding layer, simple RNN and dense layer, we improve the results of the accuracy score significantly. We are especially interested in finding a better model by considering the difference of each models that we learned in class actually work out in the context of real-life problem-solving process.

II. DATASET AND FEATURES

A. Data Acquisition

We have five datasets in this project. Our dataset is scraped from long humor, wiki, proverb, humor, reuter headlines. All of the datasets are developer-friendly and can be obtained from Kaggle. By reading the pickle files, building dataframe from the data, and giving humorous sentence positive class and the rest negative class, we then randomly picking records from negative class, build data frame, shuffle and reset the index to obtain total number of 10502 sample data.

B. Data Features

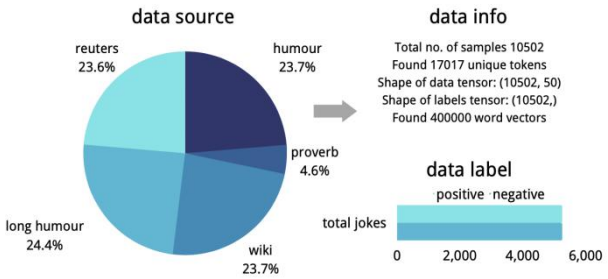

Fig. 1: Data Distribution

The jokes are constructed from the humour record, and the non-jokes are constructed from the proverb, wiki and reuter record. The humor or non-humor form the y labels. After data cleaning and selecting, our main dataset contains jokes and non-jokes of 5251, providing 17017 unique tokens.

C. Data Preprocessing

We processed our data before constructing our machine learning models. First we load the data set by reading the pickle files of data, splitting into positive and negative samples. The regular data cleaning include clearing mistakes and outliers, we find that with the engagement of long humors, low accuracy happens, therefore,

we decided to abandon the long humors, and for other texts during tokenizing, we set the limit of max length to be 50. We expect that the data is already clean enough by removing the long humor one-liners. Thus we preprocess data by creating word vectors using Word2Vec so that that instead of representing as one-hot encoding in high dimensional space, we represent words in a way that similar words get similar word vectors, so that are mapped to nearby points. After building list of sentence vectors, we split the data into training set and testing set, where training set accounts for 80% of data, in total 8401, and testing set accounts for 20% data, in total 2101, with X and Y respectively, where X are the word vectors and Y are the labels of 0 and 1 (0: negative, 1: positive).

III. EXPLANATION OF METHODS USED

A. *Baseline Models*

1) SVM: Here we thought about SVM because it is a supervised learning method and can be applied in binary classification. We applied SVM on the x data set, and observed how it predicts the y label. In this method, we used the kernel of linear, polynomial and Radial Basis Kernel. Based on sklearn's accuracy score, the linear kernel performs better, followed by rbf and then polynomial. Fig. 2 from "Support vector regression based residual control charts" provides a better understanding of the methods, from left to right (a) Linear (b) Polynomial (c) Gaussian RBF and (d) Exponential RBF.

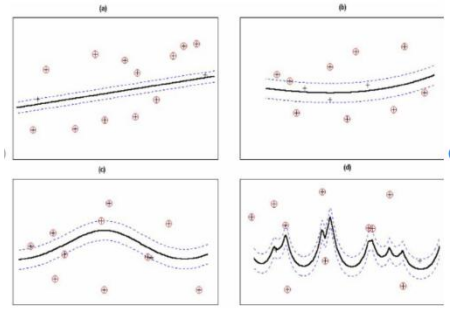

Fig. 2: Different SVM model comparison [8]

We analyzed the pros and cons while using this method. From the accuracy score, we deduced the advantage and disadvantage of SVM in this project. With high dimensional spaces, SVM shows a clear margin and separator. However, SVM performs a bit slower compared with the other methods, i.e., logistic regression and decision tree, as we assume it may be significantly impacted when dealing with large amount of data. Moreover, the model may also be impacted if considering noise implications.

2) Logistic Regression: We thought about Logistic Regression model because the most common logistic regression models have a binary outcome like true/false, our classification is also humor/non-humor. By applying logistic regression we get the best result among the base line models, reaching an accuracy score of around 87%, although vary from time to time based on the classification of training and testing set.

3) Decision Tree: Decision Tree can be used to solve both classification and regression problems, thus can also be used to solve this binary classification problem. We used sklearn's DecisionTreeClassifier and the result is slightly less satisfying than the previous two models, around 81%. We think that the result may because of decision tree's main drawback which is that it generally leads to overfitting of the data, so the model is useful only to its training data set, and this phenomenon slightly impact testing or validation data sets.

Table 1 shows the accuracy score and accuracy range of the three models in terms of cross-validation and gives a indication of their relative performance.

We deduced that Logistic Regression > SVM > Decision Tree.

Model	SVM	Logistic Regression	Decision Tree
Score	0.839	0.879	0.816
Accuracy	0.836 +/- 0.018	0.881 +/- 0.010	0.815 +/- 0.012

TABLE I: Baseline Models

B. *Advance models of Neural Network*

1. *From Word2Vec to GloVe*

In the previous data prepossessing method, we used Word2Vec, which relies on local information of word. Parameters for training on new languages cannot be shared. Thus, Word2Vec only compares word-word relationship, which is not ideal for humor detection and classification because not only words contain humor, but also combinations of the expression contain humor. Thus, considering word pair to word pair relationship while constructing the vectors is more meaningful compared to constructing from word-word relationships.

GloVe stands for Global Vectors for word representation. It is an unsupervised learning algorithm developed by researchers at Stanford University aiming to generate word embeddings by aggregating global word co-occurrence matrices from a given corpus [2]. The modified pre-processing method GloVe is easier to parallelise compared to Word2Vec and require shorter training time. (Fig. 3 shows the transition).

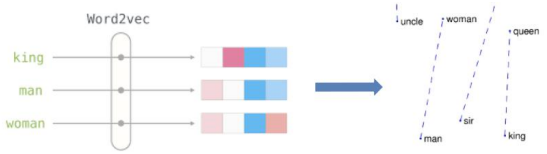

Fig. 2: Word2Vec & GloVe comparison [2]

2. *Embedding layer*

In this process, we used embedding layer which is a common tool in Keras in Natural Language Processing. We use embedding because it allows translating high-dimensional vectors into relatively low-dimensional space. Embeddings make it easier to do machine learning on large inputs like sparse vectors representing words in our task, learned and reused across models [3].

3. *F-measure*

We introduces a better validation method than cross validation score, namely the F-measure. The F-measure is calculated as the harmonic mean of precision and recall, giving each the same weighting. It allows a model to be evaluated taking both the precision and recall into account using a single score, which is helpful when describing the performance of the model and in comparing models. By using F-measure, we can better describe the performance of the model and in comparing models.

$$F_1 = \frac{2}{\frac{1}{\text{recall}} \times \frac{1}{\text{precision}}} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$
$$= \frac{\text{tp}}{\text{tp} + \frac{1}{2}(\text{fp} + \text{fn})}$$

Fig. 3: F measure formula

4. About the three RNN models

1) Simple RNN layer on top of embedding layer: We use simple RNN instead of the feed-forward neural network because the latter lacks flexibility and reads data without memorizing data. The simple RNN works well because it doesn't create multiple hidden layers, but only create one layer and loop over it. (fig. 2) The result of the RNN models is as follow. Note that during implementation we used batch size 64, dropout rate 0.1.

Training loss: 0.0795	Training accuracy: 0.9712	Training f-measure: 0.9699
Validation loss: 0.1153	Validation accuracy: 0.9572	Validation f-measure: 0.9568

TABLE II: Simple RNN result

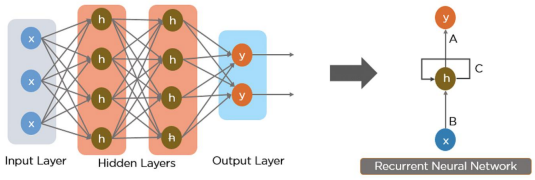

Fig. 2: feed-forward NN & simple RNN comparison [6]

2) LSTM layer on top of embedding layer: LSTM use a series of "gates" which control how the information in a sequence of data comes into, is stored in and leaves the network. The forget gate, input gate and output gate function as their own neural network. [4] (fig. 3) Note that during implementation we used batch size 64, dropout rate 0.1, recurrent drop out rate 0.5.

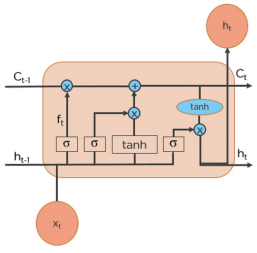

Fig. 3: LSTM layer demonstration[6]

Training loss: 0.0964	Training accuracy: 0.9644	Training f-measure: 0.9630
Validation loss: 0.1155	Validation accuracy: 0.9534	Validation f-measure: 0.9531

TABLE III: LSTM result

3) GRU layer on top of embedding layer: GRU use a gating mechanism in RNN, introduced in 2014 by Kyunghyun Cho et al. The GRU is like a long short-term memory (LSTM) with a forget gate, but has fewer parameters than LSTM, as it lacks an output gate [5]. (fig.4: graphic comparison between RNN, LSTM and GRU layers) Note that during implementation we used batch size 64, activation function ReLu, dropout rate 0.1, recurrent drop out rate 0.5.

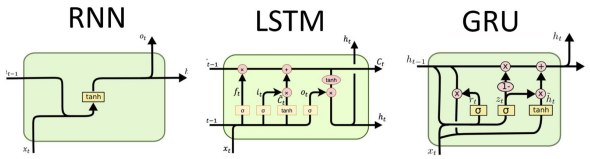

Fig. 4: comparison between RNN, LSTM and GRU layers [7]

Training loss: 0.1134	Training accuracy: 0.9584	Training f-measure: 0.9571
Validation loss: 0.2377	Validation accuracy: 0.9534	Validation f-measure: 0.9522

TABLE II: GRU result

Note: About the Training Process:

Training Process following the three methods is the same, with Optimizer: RMSprop with 0.0001 learning rate, 20 epochs, 32 batch size, validation split: 0.1.

IV. TUNING HYPERPARAMETER OF EPOCH

In this part, we plot the training and validation accuracy, training and validation loss, f-measure result for each method, namely Simple RNN layer, LSTM layer, GRU layer on top of embedding layer. The blue line stands for training loss, and the orange line stands for validation loss, and the below 9 diagrams are our results for tuning hyper-parameter epoch.

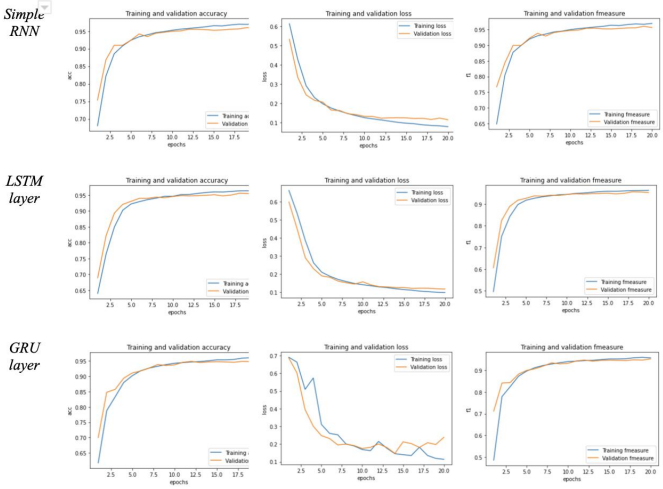

Fig. 5: Tuning hyper-parameter epoch result

V. RESULTS

In our previous evaluation on baseline models including SVM, logistic regression and regression tree, we get the result of SVM perform the best. After adjusting the preprocessing from Word2Vec to GloVe, and explore the three RNN models, including Simple RNN layer on top of embedding layer, LSTM layer on top of embedding layer, GRU layer on top of embedding layer, we applied cross-validation and F-measure test, and find that Simple RNN layer on top of embedding layer works the best.

VI. CONCLUSION AND FUTURE WORK

In this project, we studied introducing a humor detection classifier using machine-learning algorithms that we learned this semester, trained and compared between different models. We went from naively thinking of basic models we learned in class, like SVM, logistic regression and regression tree, to learning about RNN models and explore more about the advantage and limitations in the process of applying the models.

We realize that for any binary classification, we should not only simply select the models suitable for the machine learning task, in our case, binary classification, but should also select the more consistent data pre-processing method, models, and validation score methods. We concluded our humor classification project by realizing the benefits of choosing RNN compared with other models. It is not only reflected in the growth of accuracy, but also a better logical fit for the method of humor judgment.

Looking forward, we can improve our project by adding more humor materials for training, and incorporating humor detection in different languages, augmenting our dataset size, and further testing more advanced model to better understand the role humor play in our machine learning class, life and beyond. With Machine Learning developments, we have more choices in this project in the future, and we will try to improve the accuracy of model using Google's Bidirectional Encoder Representations from Transformer (BERT), ELMo pre-trained word embeddings, and Roberta. We also enjoyed looking at the training data we find, and realize what humor really is, giving us strength and courage in the busy final week and under the extremely harsh pandemic situation. May the force of humor be with you!

REFERENCES

[1] Rada Mihalcea and Carlo Strapparava, Making Computers Laugh: Investigations in Automatic Humor Recognition, in Proceedings of the Joint Conference on Human Language Technology / Empirical Methods in Natural Language Processing (HLT/EMNLP), Vancouver, October, 2005.

[2] Oliveira and Rodrigo, "Humor Detection in Yelp reviews", at Stanford University.

[3]"Embeddings | Machine Learning Crash Course | Google Developers." Google, Machine Learning Crash Course, 2022, https://developers.google.com/machine-learning/crash-course/embeddings/video-lecture.

[4] Dolphin, Rian, "LSTM Networks | A Detailed Explanation", Towards Data Science, October 22, 2020.

https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9.

[5] Saxena, Shipra, "Introduction to Gated Recurrent Unit (GRU)", Analytics Vidhya, March 17, 2021.

https://www.analyticsvidhya.com/blog/2021/03/introduction-to-gated-recurrent-unit-gru/.

[6] Biswal, Avijeet, "Recurrent Neural Network (RNN) Tutorial: Types, Examples, LSTM and More", *Sympli Learn*, Feb.21, 2022.

https://www.simplilearn.com/tutorials/deep-learning-tutorial/rnn.

[7] Lopez, dProgrammer , "RNN, LSTM & GRU", *dprogrammer*, April.6, 2019. http://dprogrammer.org/rnn-lstm-gru.

[8] Gani, Walid, Taleb, Hassen, MOhamed Limam, "Support Vector Regression Based Residual control Charts", Journal of Applied Statistics, vol 37, pp. 309-324.