Fake News Classification Using Deep Learning

Christopher Weicong Hong
Illinois Institute of Technology
CS 579 - Fall 2020
whong7@hakw.iit.edu

Abstract

The scope of this project is to build a multinomial classifier that can predict whether a news pair is “unrelated”, “agreed” or “disagreed”. The minimum acceptable performance metric, f1-score, of the model should be higher than random guessing, i.e., the underlying data distribution of the target. To achieve this project objective, multiple different model implementations have been tested and their results have been analyzed. An LSTM model with vocabulary size of 5,000, input sequence length of 256, embedding size of 256, number of LSTM units of 128, unidirectional layer of 1 has the highest f1 score across the classes of the “label”.

1 Introduction

Social media has become one of the most important and convenient tools that allows users to quickly, easily and efficiently engage with others through creating and/or sharing content. For example, businesses leverage social media tools, such as Twitter, Facebook, YouTube, etc., to market their products and services or to engage with their customers. Yet, certain groups with potentially malicious intentions might initiate fake news using social media to undermine the integrity of information. It is estimated that the spread of fake news has a negative impact on the public and our society. This phenomenon offers an opportunity to leverage machine learning techniques to identify patterns in fake news.

Like any other machine learning activity, this project started with defining the problem of interest and metric of success, collecting and preparing data, selecting models, fine-tuning hyperparameters, assessing model performance and reporting final results.

2 Project Scope

The goal of this project is to build a multinomial classifier that can dispatch input instances (news pairs), i.e. two titles, into one of a predefined set of distinct classes (“unrelated”, “agreed” or “disagreed”). The successful classifier should have predictability and reproducibility [1]. If the input comes from the same distribution as the training data, the classifier has to make approximately the same percentage of errors as observed on the held-out data when the model was trained. Plus, a model can be reproduced from the same training data using the same algorithm and values of hyperparameters. The performance metric of success, f1-score, for each class of the “label”, has to be higher than random guessing (distribution of the training data). Based on **Figure 1** [2], the default distribution of the training data is approximately 68% for “unrelated”, 29% for “agreed” and 3% for “disagreed” .

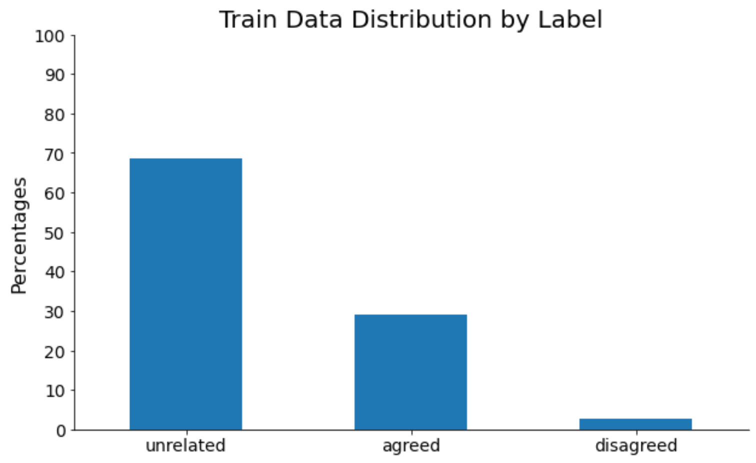
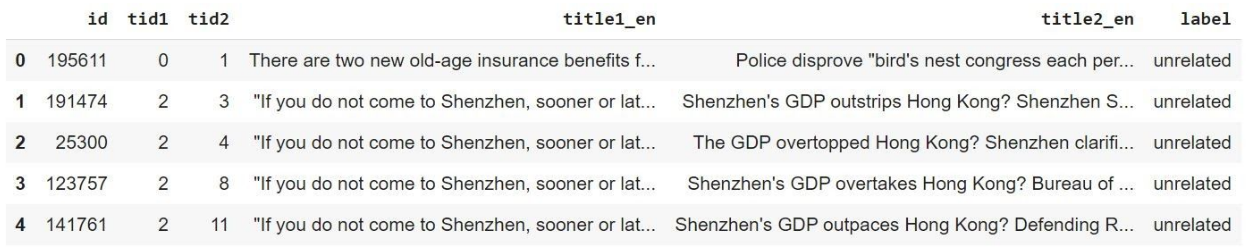


Figure 1

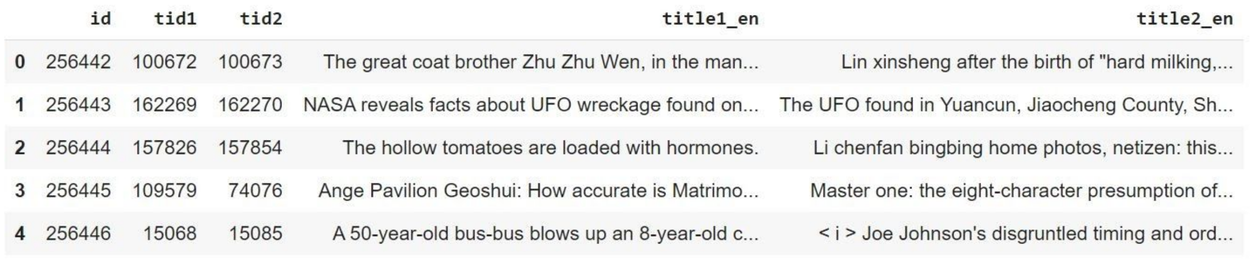
3 Data Collection

The training and test data set were provided in the form of .csv file. There were 256,442 entries and 6 attributes for the training set (**Figure 2**) [3] while 64,100 entries and 5 attributes (excluding the "label") for the test set (**Figure 3**) [3]. There was enough data to train a shallow learning algorithm while it was considered a fairly small data set for deep learning algorithms. The size of the test set was roughly 25% of the training one. Attributed "id" was the case number of each observation, "tid1" and "tid2" were identifiers for "title1_en" and "title2_en" attributes, respectively. Attribute "label" was the target to predict. Data reliability and cost are not the main problem of interest in this project.



	id	tid1	tid2	title1_en	title2_en	label
0	195611	0	1	There are two new old-age insurance benefits f...	Police disprove "bird's nest congress each per...	unrelated
1	191474	2	3	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP outstrips Hong Kong? Shenzhen S...	unrelated
2	25300	2	4	"If you do not come to Shenzhen, sooner or lat...	The GDP overtopped Hong Kong? Shenzhen clarifi...	unrelated
3	123757	2	8	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP overtakes Hong Kong? Bureau of ...	unrelated
4	141761	2	11	"If you do not come to Shenzhen, sooner or lat...	Shenzhen's GDP outpaces Hong Kong? Defending R...	unrelated

Figure 2



	id	tid1	tid2	title1_en	title2_en
0	256442	100672	100673	The great coat brother Zhu Zhu Wen, in the man...	Lin xinsheng after the birth of "hard milking,...
1	256443	162269	162270	NASA reveals facts about UFO wreckage found on...	The UFO found in Yuancun, Jiaocheng County, Sh...
2	256444	157826	157854	The hollow tomatoes are loaded with hormones.	Li chenfan bingbing home photos, netizen: this...
3	256445	109579	74076	Ange Pavilion Geoshui: How accurate is Matrimo...	Master one: the eight-character presumption of...
4	256446	15068	15085	A 50-year-old bus-bus blows up an 8-year-old c...	< i > Joe Johnson's disgruntled timing and ord...

Figure 3

4 Data Preprocessing

4.1 Data Partitioning

The training data was splitted into training and validation sets since the test set was separate from the original training one (**Figure 4**) [1].

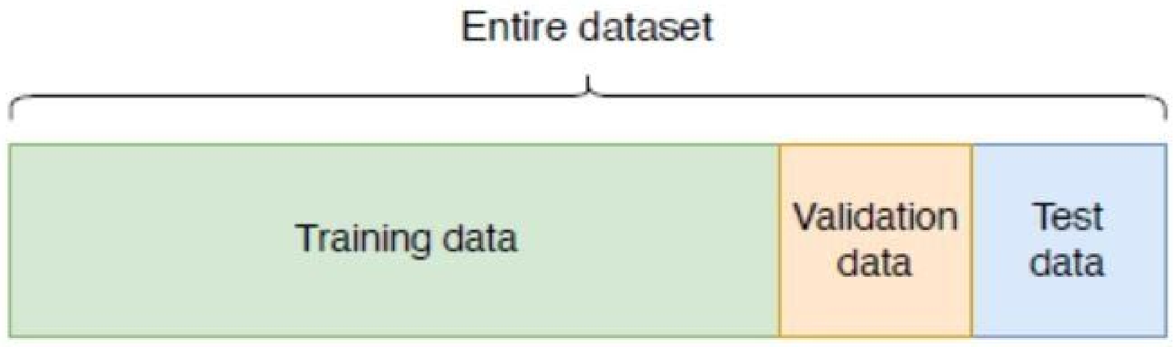


Figure 4: Adapted from Burkov, A.

The training set was used to train models and to select the best one from those. The validation set was used to find the best values of hyperparameters of a selected model. The test set was used to report final submission in this project.

To avoid data leakage, sampling and evaluation bias, the data partitioning satisfied the following conditions [1]:

- **Condition 1:** Split was applied to raw training data before everything else to avoid data leakage.
- **Condition 2:** Training data was randomly shuffled before the split [4].
- **Condition 3:** Training and validation sets followed the same distribution [4], 68%/29%/3% **(Figure 5)** [2], using stratified sampling after split. Plus, the validation set had roughly the same size as the test set.

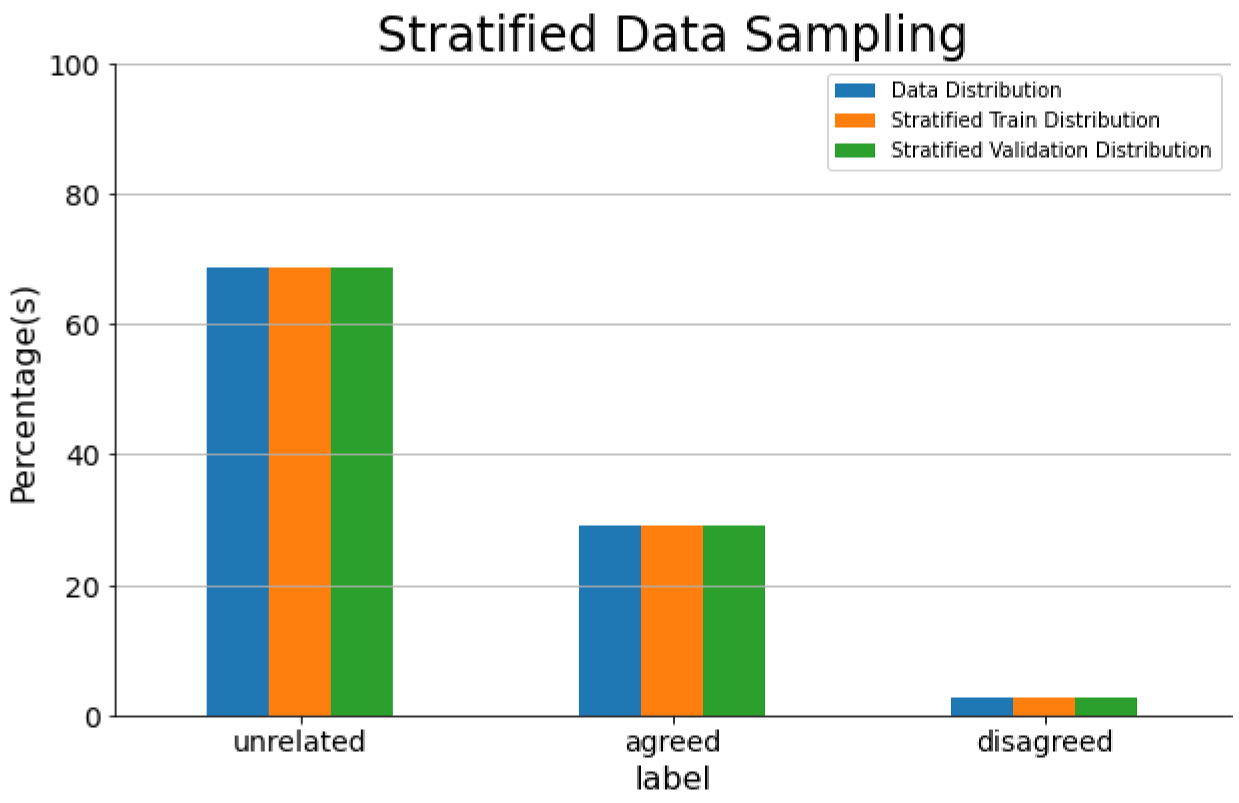


Figure 5

4.2 Data Wrangling

Data content and structural issues were diagnosed first. The enrited data set was complete (no missing records), valid (no constraint violations), accurate (no incorrect records) and consistent (values of columns adhered to the same format except “ttile1_en” and “title2_en”). The data came in a tidy form, each variable formed a column (except “ttile1_en” and “title2_en”), each observation formed a row and each type of observational unit formed a table.

To prepare for feature engineering, “titl1_en” and “title2_en” were normalized using NLP [5] techniques, including lowercasing, tokenization, lemmatization, stop word, punctuation and unwanted token removal. Below was the result of one of the values of “title1_en” before and after normalization:

- **Before:** “A few characteristics, let a pregnant mother know the difference between having a boy and having a baby girl!”
- **After:** “characteristic let pregnant mother know difference boy baby girl”

No matter the lengths, in terms of words, of the normalized and the unnormalized titles exhibited a long-tail problem **(Figure 6)** [2].

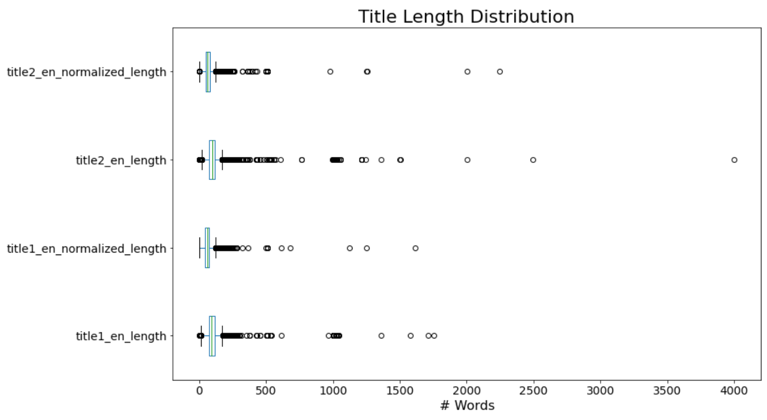


Figure 6

The typical length of either normalized “titl1_en” or “title2_en” was much less than the unnormalized one **(Figure 7)** [2].

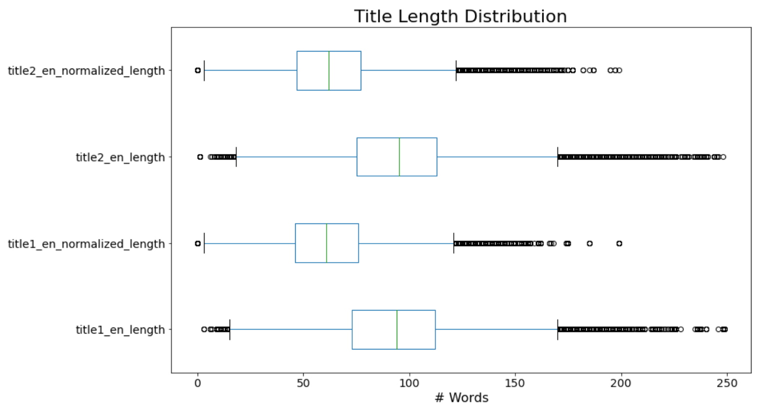


Figure 7

The majority of the lengths of the aforementioned ones were approximately normally distributed (**Figure 8**) [2].

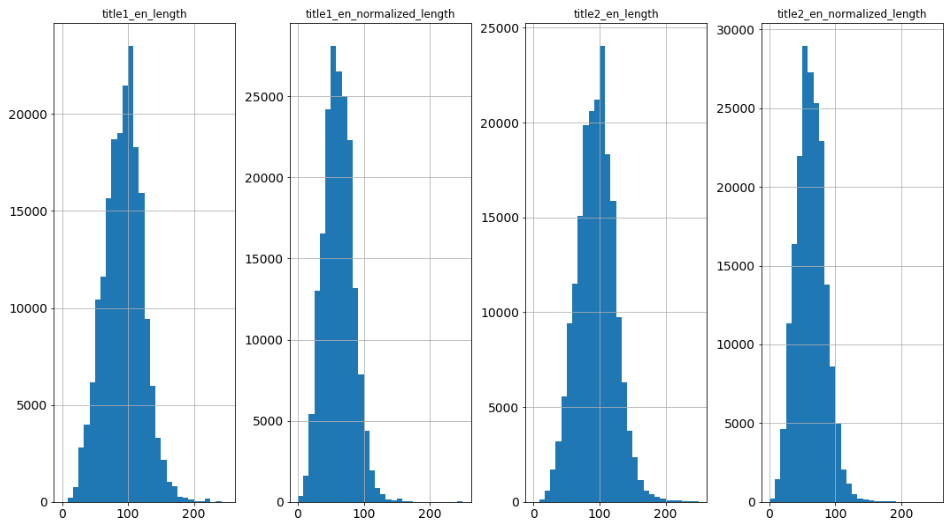


Figure 8

The total length of the aforementioned ones typically ranged from 50 to 200 words (**Figure 9**) [2]. One of the advantages of text normalization, in practice, is that it reduces the vocabulary size without affecting the predictability of the text yet improving the computational efficiency.

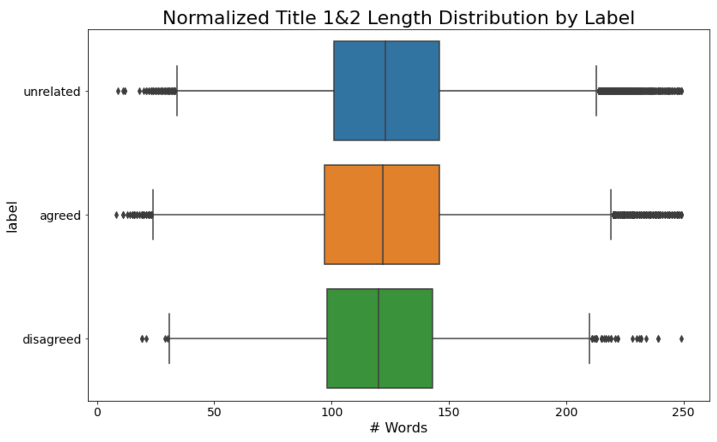


Figure 9

Upon finishing text normalization, normalized “title1_en” and “title2_en” were concatenated to form a meta “title” for feature engineering. **Figure 10** [2] showed some of the most mentioned key words [6] in the aforementioned ones, such as “weight”, “loss”, “white”, ect..

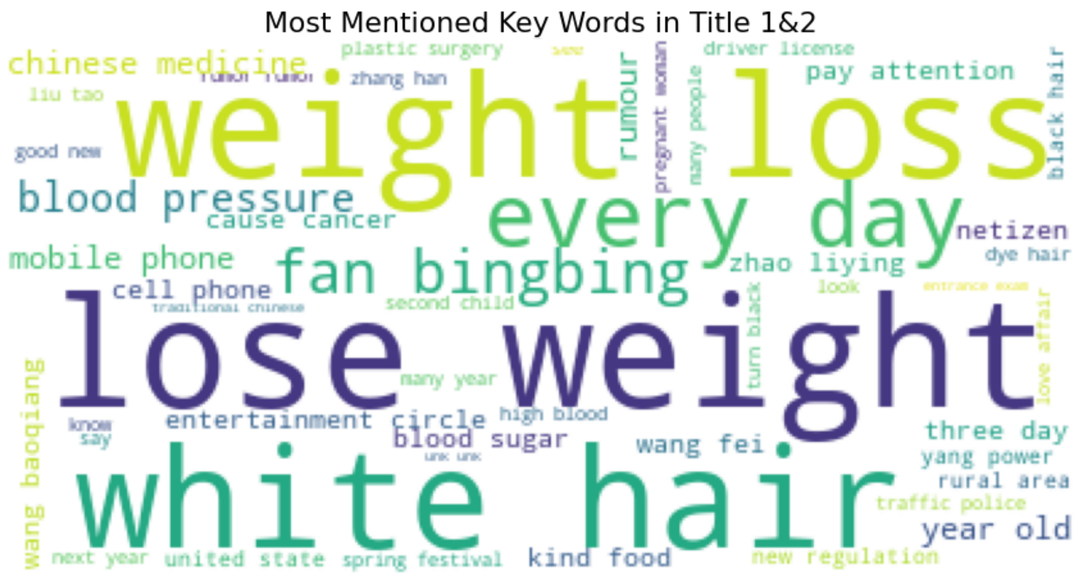


Figure 10

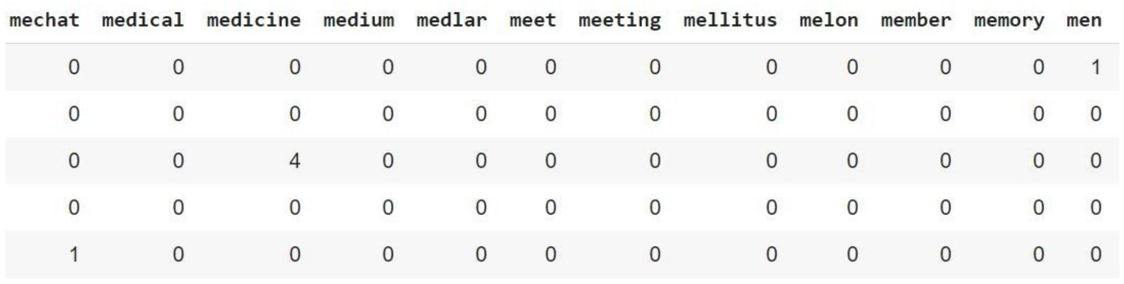
5 Feature Engineering

“After data collection and preparation, feature engineering is the second most important activity in machine learning” [1]. Plus, machine learning algorithms can only apply to feature vectors only.

5.1 Feature Engineering for Text

Multiple text feature engineering techniques were implemented as following:

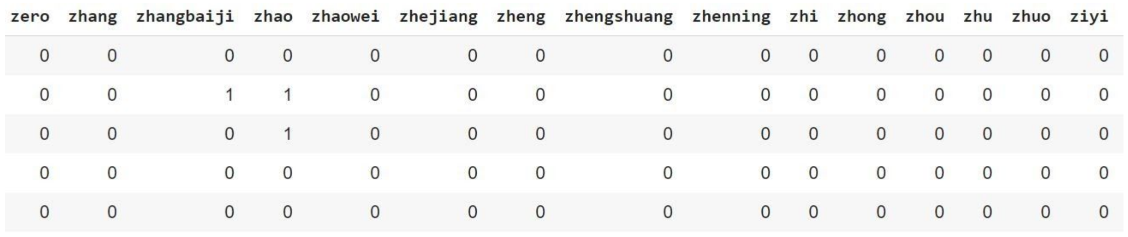
- **Count bag-of-one-gram:** The count of presence of each word in the vocabulary of each news pair was recorded in each cell of each vector (**Figure 11**) [3, 4].



mechat	medical	medicine	medium	medlar	meet	meeting	mellitus	melon	member	memory	men
0	0	0	0	0	0	0	0	0	0	0	1
0	0	0	0	0	0	0	0	0	0	0	0
0	0	4	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0

Figure 11

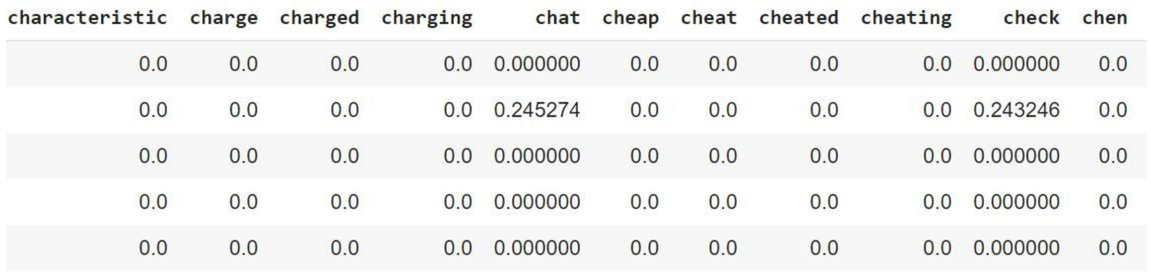
- **Binary bag-of-one-gram:** Same format as count bag-of-words, but the presence of a word was recorded as 1 otherwise 0 (**Figure 12**) [3, 4].



zero	zhang	zhangbaiji	zhao	zhaowei	zhejiang	zheng	zhengshuang	zhenning	zhi	zhong	zhou	zhu	zhuo	ziyi
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figure 12

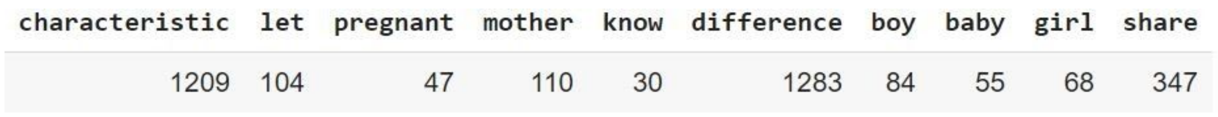
- **TF-IDF-of-one-gram:** The presence of a word was weighed instead of counted (**Figure 13**) [3, 4].



characteristic	charge	charged	charging	chat	cheap	cheat	cheated	cheating	check	chen
0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0
0.0	0.0	0.0	0.0	0.245274	0.0	0.0	0.0	0.0	0.243246	0.0
0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0
0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0
0.0	0.0	0.0	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0

Figure 13

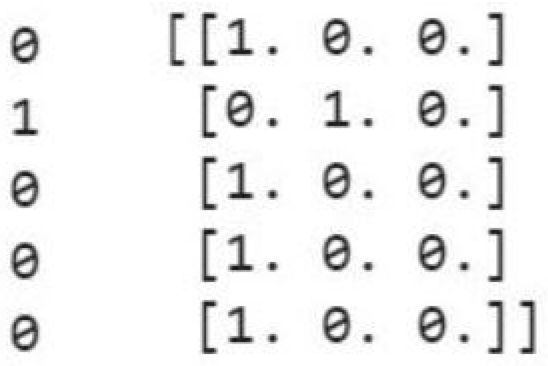
- **Order encoding:** Vocabulary was converted into ordered integers (**Figure 14**) [3, 4]. Same applied to label encoding in shallow learning algorithms.



characteristic	let	pregnant	mother	know	difference	boy	baby	girl	share
1209	104	47	110	30	1283	84	55	68	347

Figure 14

- **One-hot encoding:** Label was converted into one-hot vectors in deep learning algorithms for multinomial classification (**Figure 15**) [7].



```
0    [[1. 0. 0.]
1     [0. 1. 0.]
0     [1. 0. 0.]
0     [1. 0. 0.]
0     [1. 0. 0.]]
```

Figure 15

- **Word Embedding:** An LSTM RNN was used to learn the useful feature from the training data (**Figure 16**) [7].

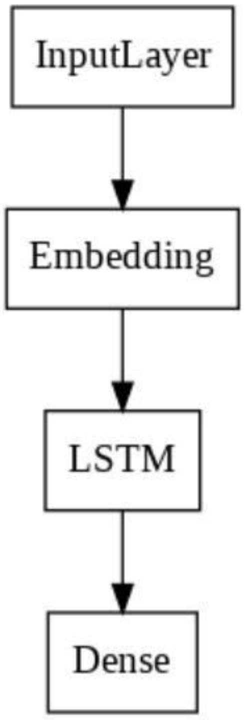


Figure 16

5.2 Feature Engineering for Continuous Features

The hypothesis was that the cosine similarity score of the news pair vectors was a good proxy of their relationship. Thus, cosine similarity score based on binary and TF-IDF vectors were engineered, respectively.

5.3 Feature Engineering for Categorical Features

The hypothesis was that the length of the news pair was associated with the classes of the "label". Thus, features for the length of each news pair were engineered and cutted into bins including [0, 47), [47, 62), [62, 77), [77, 2246] based on the distribution shown in **Figure 7** [3].

6 Feature Selection

Not all features would be equally important. The vocabulary can be very large, while it exhibited long-tailed phenomenon. "[I]f the feature vector is very wide (contains thousands or millions of features), the training time can become prohibitively long. Furthermore, the overall size of the training data can become too large to fit in the RAM of a conventional server. If we could estimate the importance of features, we would keep only the most important ones. That would allow us to save time, fit more examples in memory, and improve the model's quality " [1].

6.1 Vocabulary Size

For shallow learning algorithms, the size of the vocabulary was chosen as the maximum length of the new pair combined. For the deep learning approach, vocabulary size was tuned as a hyperparameter.

6.2 Continuous Features

Deviance tests were performed on the association between continuous variables and the categorical target, “label”. **Figure 17** [2] showed that cosine similarity score engineered from binary bag-of-one-gram had the highest McFadden Pseudo Squared score while attributes including “id”, “tid1” and “tid” showed negligible association.

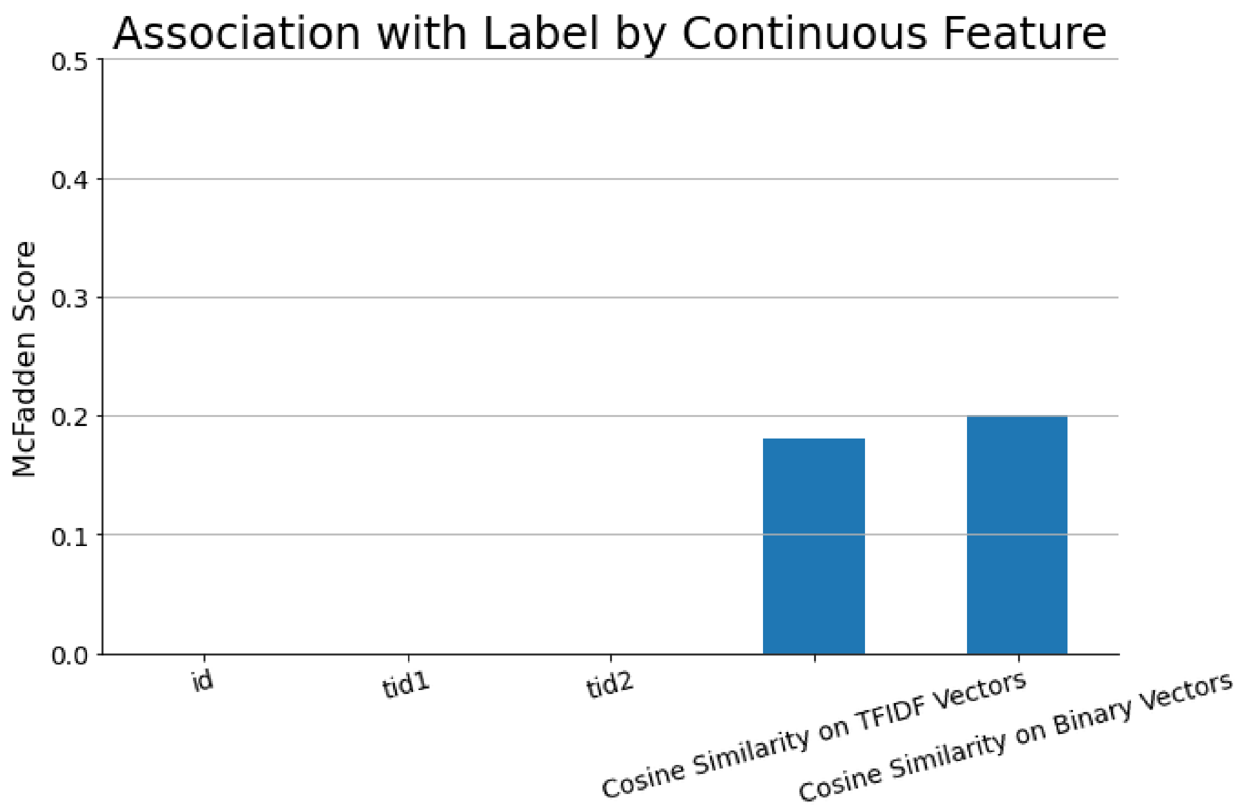


Figure 17

6.3 Categorical (Nominal) Features

Chi-squared tests were performed on the association between categorical variables and the categorical target, “label”. **Figure 18** [2] showed that the length of a title had no association (Cramer’s V score was close to 0) with “label”.

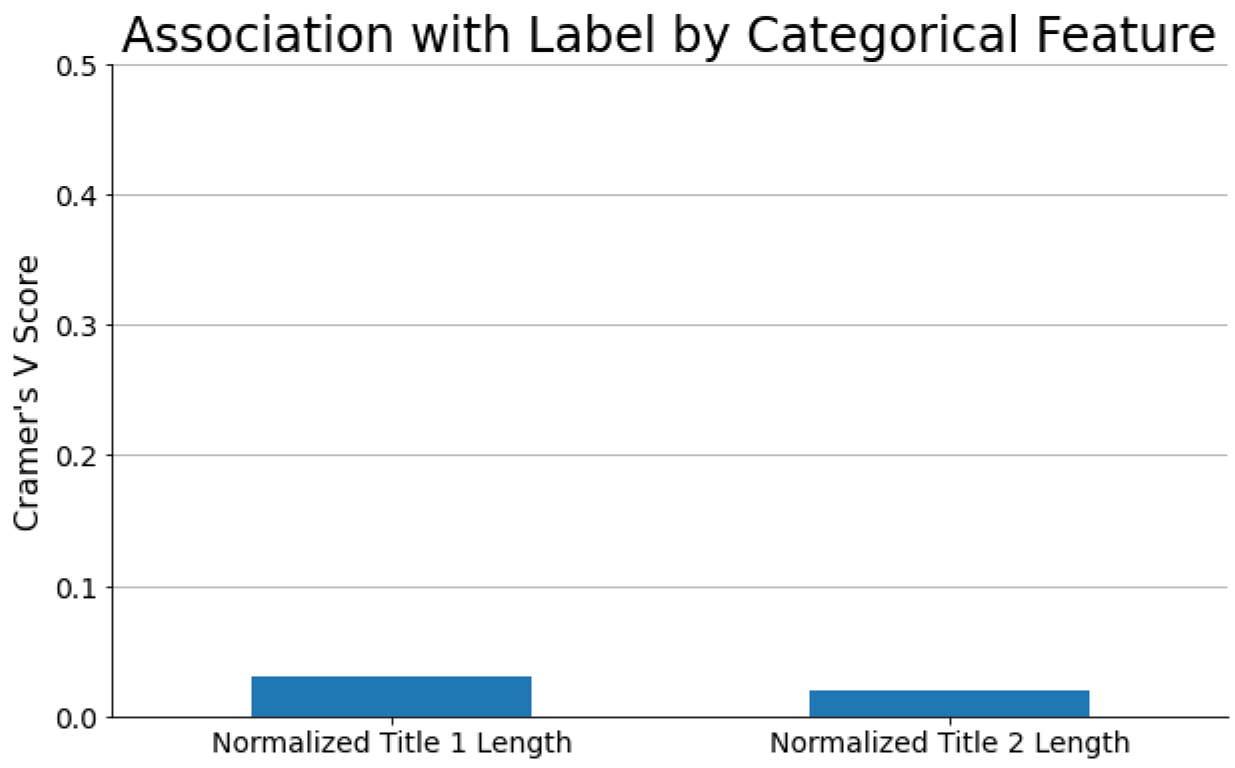


Figure 18

6.4 Vector Features

Wrapper methods, i.e. using machine learning algorithms, were used to select the best vectorization. Multinomial classifiers including **Naive Bayes** [4] and **Logistic Regression** [4] were leveraged to evaluate the f1 score on each class of “label” using **K-fold cross validation** [4] on the training set [4]. From **Figure 19** [2], it was shown that binary bag-of-one-gram had better performance on both aforementioned algorithms. **Logistic Regression** [4] trained on the combination of Binary bag-of-one-gram and its cosine similarity score had the best performance.

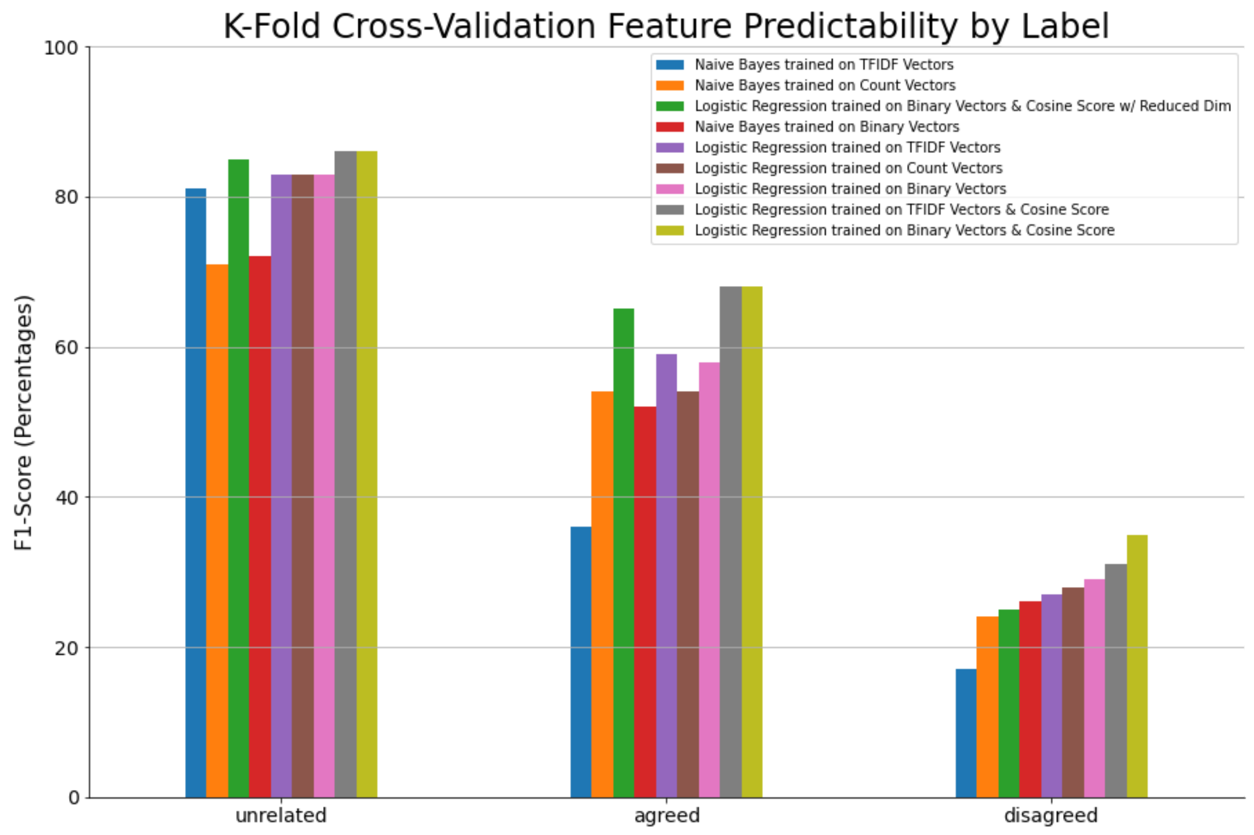


Figure 19

6.5 Vector features with Dimensionality Reduction

The combination of binary bag-of-one-gram and cosine similarity with 70% variances captured, using **PCA** [4], had obvious performance degradation on minority classes "agreed" and "disagreed" (**Figure 17**) [2]. However, the vector with reduced dimensionality from 2,336 to 574 provided feasible computational runtime on later model selection on shallow learning algorithms.

7 Model Selection

Logistic Regression [4] was chosen as a baseline model and provided a reference point for comparison on the chosen features with 70% variances captured. **Random Forest** [4] and **RNN** [7] with one unidirectional **LSTM** layer were trained on the same data set.

Figure 20 [2] showed that **Random Forest** [4] had much better performance than random guessing (**human performance**) while it was worse than linear **Logistic Regression** [4] and **LSTM** model. On the other hand, the **LSTM** model had much higher performance on all classes of "label" than others. This result confirmed the expected one since this model took the order in addition to the presence of words into consideration.

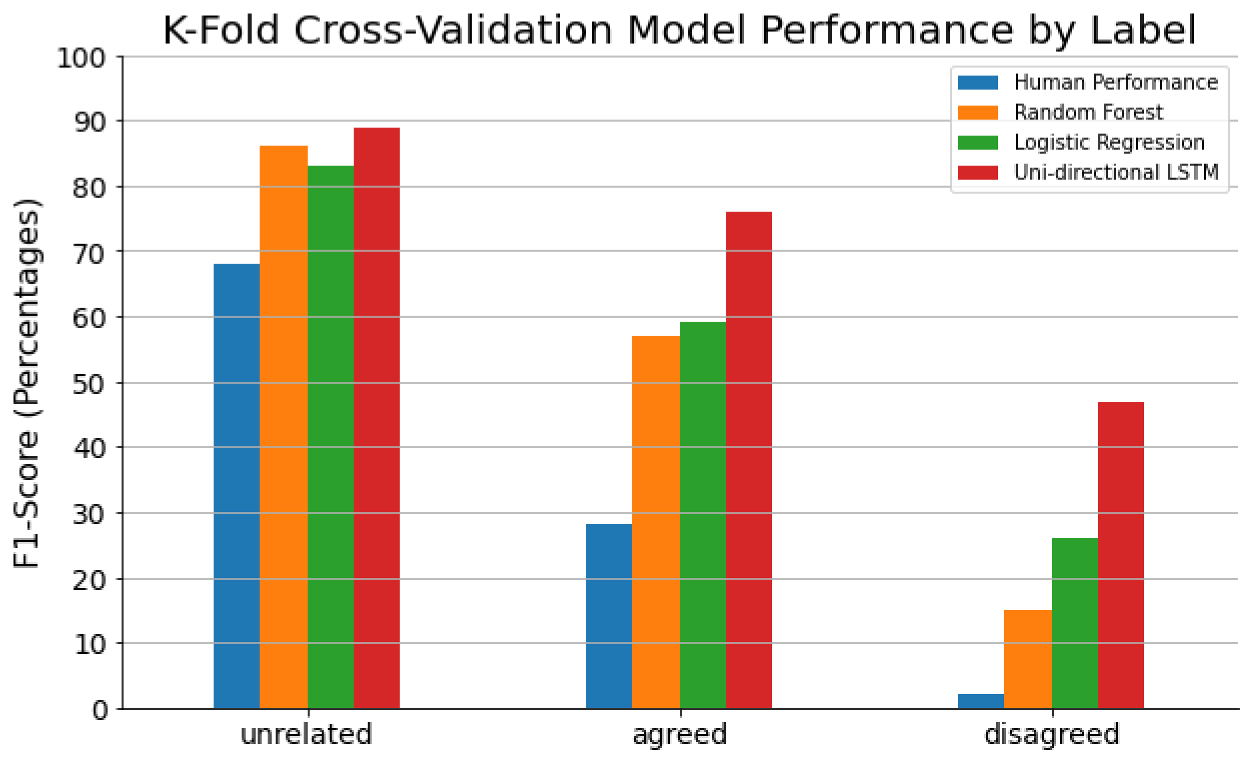


Figure 20

8 Hyperparameter Tuning

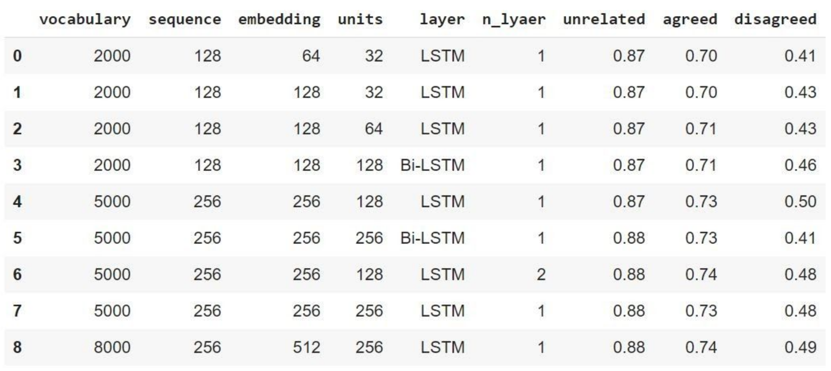
"Hyperparameters play an important role in the model training process. Some hyperparameters influence the speed of training, but the most important hyperparameters control the two [trade offs]: bias-variance and precision-recall. Hyperparameters aren't optimized by the learning algorithm itself" [1].

Upon selecting the **LSTM** model as the final one, a manual random search strategy was applied to find the best hyperparameters. The values of hyperparameters were as follows:

- **Vocabulary size:** 2,000, 5,000 and 8,000;
- **Input sequence length:** 128 and 256;
- **Word embedding size:** 64, 128, 256 and 512;
- **LSTM units:** 32, 64, 128 and 256;
- **LSTM direction:** unidirectional and bidirectional;
- **Number of LSTM layers:** 1 and 2.

The selections of values listed above were based on common practices and recommendations by pioneer researchers [8]. The order of hyperparameters tuning based on previous model performance.

Figure 21 [3] showed that the best set of hyperparameters were with **vocabulary size** of **5,000**, **input sequence length** of **256**, **word embedding size** of **256**, **LSTM units** of **128** and **1 unidirectional LSTM layer**. Selection of the final model took into consideration the bias-variance tradeoff and actual typical sequence length of this training data set.

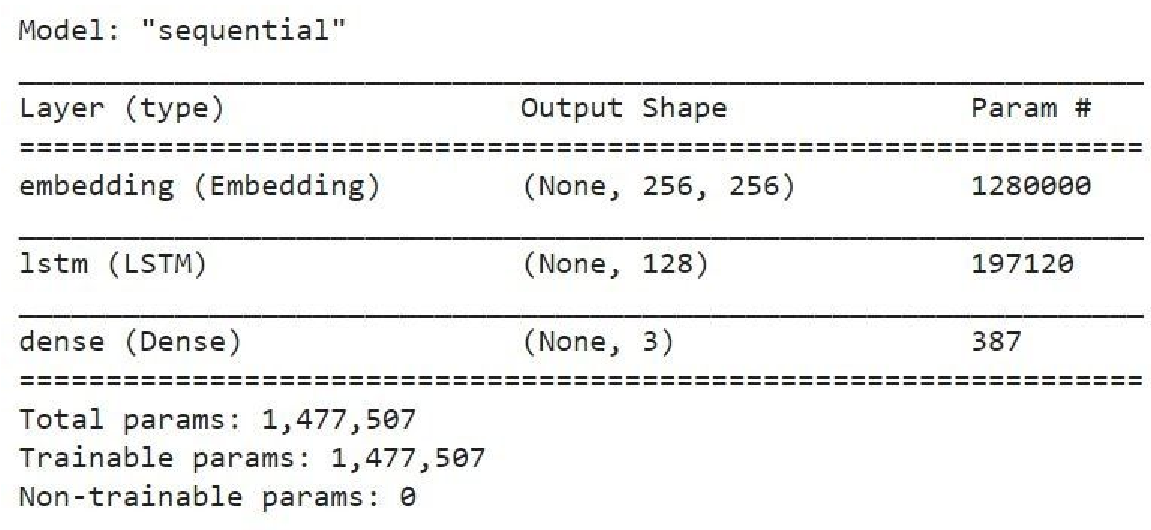


	vocabulary	sequence	embedding	units	layer	n_lyaer	unrelated	agreed	disagreed
0	2000	128	64	32	LSTM	1	0.87	0.70	0.41
1	2000	128	128	32	LSTM	1	0.87	0.70	0.43
2	2000	128	128	64	LSTM	1	0.87	0.71	0.43
3	2000	128	128	128	Bi-LSTM	1	0.87	0.71	0.46
4	5000	256	256	128	LSTM	1	0.87	0.73	0.50
5	5000	256	256	256	Bi-LSTM	1	0.88	0.73	0.41
6	5000	256	256	128	LSTM	2	0.88	0.74	0.48
7	5000	256	256	256	LSTM	1	0.88	0.73	0.48
8	8000	256	512	256	LSTM	1	0.88	0.74	0.49

Figure 21

9 Model Description

Figure 22 [7] was the description of the final model. It contained an **embedding**, **LSTM** and **dense** layer in total. Each sample was 256 by 256 matrix. The LSTM layer was unidirectional and consisted of 218 units. The dense (output) layer contained 3 units because of three classes of "label". **Figure 23** [7] showed the model in a top-down fashion.



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 256, 256)          1280000
_________________________________________________________________
lstm (LSTM)                  (None, 128)               197120
_________________________________________________________________
dense (Dense)                (None, 3)                 387
=================================================================
Total params: 1,477,507
Trainable params: 1,477,507
Non-trainable params: 0
```

Figure 22

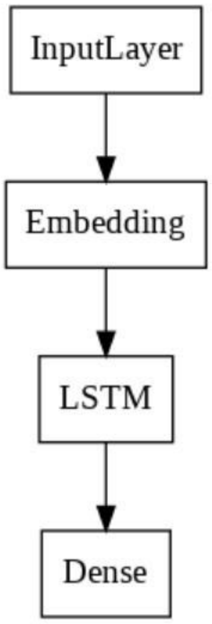


Figure 23

10 Regularization

The number of regularization techniques were large. Due to time and computational complexity, one of the most commonly used techniques, early stopping, was chosen [8].

Figure 24 [2] showed that the loss stopped decreasing after 4 epochs while **Figure 25** [2] showed that the accuracy stopped increasing after 7 epochs. Since the objective was accuracy, the final model was the one trained with 7 epochs.

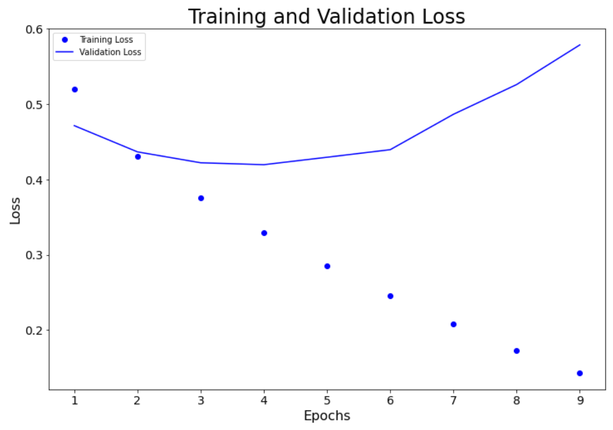


Figure 24

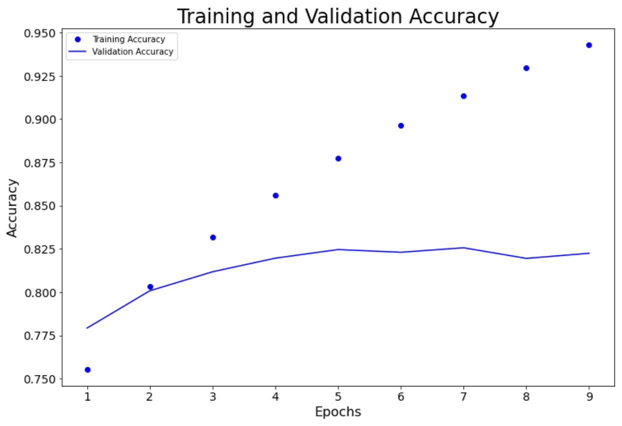


Figure 25

Early stopping [8] requires a validation set which is not fed to the model. The strategy was to continue training the model from the first round of training using all of the data (training set and validation set) for 2 more rounds. The number of two (ceiling(7/4)) chosen here was based on the number of epochs, 7, in the first round training and the validation set size which was roughly 25% of the training one.

11 Results

Figure 26 [2] showed that the held-out f1 score was much higher than the default data distribution (human performance) for each class of "label". The categorization score was approximately **83%**.

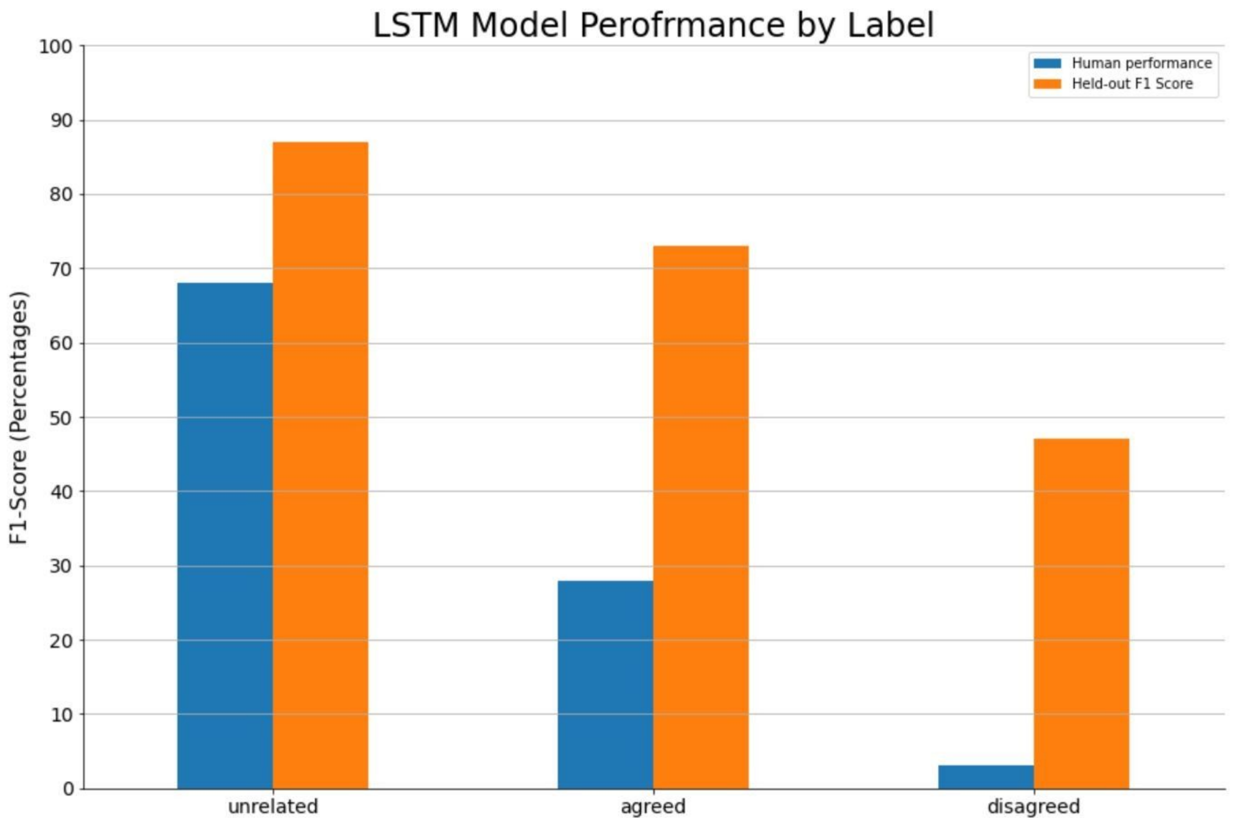


Figure 26

12 Conclusion

In accordance with the above mentioned model performance, the **RNN** with **LSTM** units outperformed shallow learning algorithms such as **Multinomial Naive Bayes**, **Logistic Regression**, and **Random Forest**. Increasing the vocabulary size of more than 5,000 had negligible performance improvement. Furthermore, adding additional **LSTM** layers or using bidirectional **LSTM** layers given the number of 128 **LSTM** units did not do significantly better.

13 Future Directions

Since the majority of the class is “unrelated”, adding more minority classes such as “agreed” and “disagreed” will, usually, improve the general model performance. Furthermore, adding features, including sources (blog, social media, publisher, etc.), topics (politics, education, entertainment, etc.), might help improve feature predictability. Last but not least, tuning the learning rate of the **RNN** model might help decrease the loss on the validation set.

Acknowledgements

Gratitude to Dr. Kai Shu, TA Wei Lan and Hao Ding, who offered such wonderful class experiences.

References

[1] Burkov, A., *Machine Learning Engineering*, http://www.mlebook.com/wiki/doku.php, November, 2020.

[2] J. D. Hunter, *Matplotlib: A 2D Graphics Environment*, Computing in Science & Engineering, vol. 9, no. 3, pp. 90-95, 2007

[3] McKinney, W., & others. *Data structures for statistical computing in python*, In Proceedings of the 9th Python in Science Conference (Vol. 445, pp. 51–56), 2010.

[4] Pedregosa et al., *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830, 2011.

[5] Bird, S., Edward L. & Ewan K., *Natural Language Processing with Python*, O'Reilly Media Inc, 2009.

[6] Mueller, A. *WorldCloud*, https://github.com/amueller/word_cloud, November, 2020

[7] Chollet, F., & others, *Keras*, https://github.com/fchollet/keras, November, 2020

[8] Goodfellow, I, Bengio, Y., & Courville, A., *Deep Learning*, https://www.deeplearningbook.org, November, 2020.