Document Classification Performance of Naive Bayes, SVM, Deep Averaging Network, and BiLSTM on Reddit Texts

Rina Battulga ub343, Sylvie Chen xc1188, Joanna Zhang jz3026, Zaiyun Lin zl1836

1 ABSTRACT

In this study, we used corpora from social media Reddit to compare document classification systems. Our systems have two linear systems: Support Vector Machine (SVM) and Naive Bayes (NB), and two Neural Networks: Deep Averaging Network (DAN) and Bi-directional LSTM (BiLSTM). TF-IDF word vectorization and Naive Bayes classification is used as the base system to compare the accuracy, precision, recall, and F1-scores generated from the confusion matrices of the algorithms. BiLSTM and DAN outperformed Naive Bayes by over 10% on F1-score and recall, while SVM performed slightly lower than our base. Precision scores for all systems did not differ significantly from each other. We also include error analysis to explain some outstanding trends across the four systems.

2 INTRODUCTION

Supervised document classification is a field where documents are classified based on a given set of matching labels. In recent years, supervised document classification algorithms with the aid of natural language processing have gained awareness for social media. In this paper, we aim to implement 4 supervised document classification algorithms to classify posts on Reddit, a social media website, into 9 predefined categories and compare their performances. The four systems are:

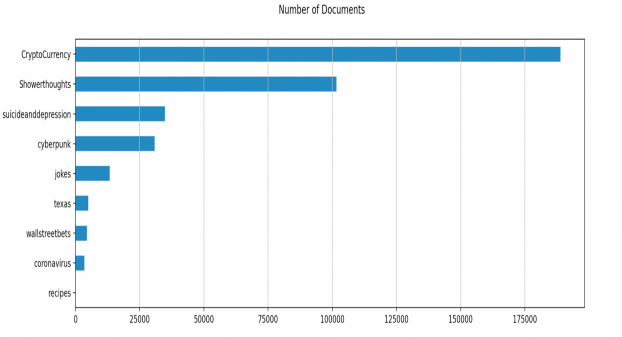
1) Naive Bayes
2) Support Vector Machine
3) Deep Averaging Network
4) Bi-directional LSTM

We will compare results and analyze their functionalities and performances. We aim to point out the advantages and disadvantages of each system based on confusion matrix, yield accuracy, precision, recall and f-score.

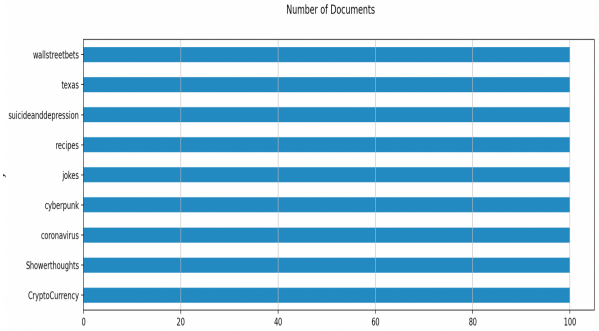
3 DATA PREPERATION

3.1 Data Distribution

We obtain 9 different categories of subreddit data from Kaggle. We process the text in a CSV format. The first column contains text from a subreddit post, and the second column holds the post's corresponding category. Each post is matched to only one label. This dataset contains over 3 million entries, which is a reasonable size to train, validate, and test the systems. We randomly partitioned our dataset into training (80%), validation (10%), and testing (10%) sets. It is important to note that the original dataset manifests a skewed distribution (Graph 3.1.1). Thus, we constructed an additional test set with a balanced distribution – 500 documents per category (Graph 3.1.2).



(Graph 3.1.1: Unbalanced Data Distribution)



(Graph 3.1.2: Balanced Data Distribution)

3.2 Document Processing

We prepare our data in 4 steps: tokenizing, lowercasing, removing stop words, and lemmatizing. First, we tokenized words with NLTK's RegexpTokenizer. Second, we lowercase all characters. Then, we remove stopwords from NLTK's 'English' stopwords library. Lastly, we convert words to their roots with NLTK's WordNetLemmatizer.

3.3 Document Representation And Feature Extraction

Vectorization is the last step for preprocessing and we use two approaches. For Naive Bayes and Support Vector Machine, we employ the TfidfVectorizer from sklearn's feature_extraction library to turn the data into Term Frequency - Inverse Document Frequency, i.e. TF-IDF, vectors. TF-IDF vectors are known for its term-weighting scheme which reflects a document's epitome words based on Bayes Formula.

For Deep Averaging Network and Bi-directional LSTM, we convert the data to pre-trained Word2Vec vectors. With gensim, we load pretrained vectors from GoogleNews-vectors-negative300.bin. Each numericalized word vector has a dimension of 300. For DAN, we get the average vector for each sentence. For BiLSTM, we keep the first 20 words of a sentence, so that a sentence has the dimension [20,300]. We chose 20 because it is the average sentence length in the training set. If a sentence length is shorter than 20, the remaining values are set as 0.

4 METHOD

4.1 Naive Bayes (Base System)

We use Naive Bayes as a baseline system for its straightforward structure and reliability.

The base system is an advanced variant of the bag-of-words model using TF-IDF for text representation. Each word in the corpus has a feature and each text is represented by a vector of the same length of these vocabularies of "bag-of-words". TF-IDF allows the value of a word to increase proportionally to its count, but inversely proportional to the frequency of the word in the whole dataset.

The Naive Bayes algorithm is a conditional probabilistic classifier based on the Bayes Theorem. It is implemented on the processed text. The theorem, $P(A|B) = \frac{P(B|A) \times P(A)}{P(B)}$, describes the probability of an event, A, occurring based on a prior outcome, B, already occurred. The algorithm assumes all predictors are independent and uses them to calculate the probability of each subreddit category. NB algorithm then uses the variables mentioned above to classify text to the category with the highest probability.

For implementation, we use sklearn's naive_bayes. This method works

best with large datasets and is known for handling multi class prediction as well as text classification.

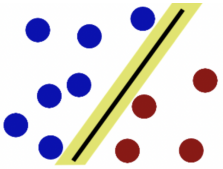
4.2 Support Vector Machine

The first system we use to compare against the baseline is the Support Vector Machine (SVM) optimized with the Stochastic Gradient Descent Classifier. SVM is a linear model that takes pre-classified documents and outputs a hyperplane $y(x) = w^T x + b$ that separates documents from one category to another, where it maximizes the distance from any data point x_0 to the hyperplane (Graph 4.2.1).

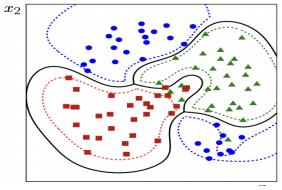
$$\# \arg\max_{w,b} \left\{ \frac{1}{||w||} \min_{n} \left[y_i (w^T x_i + b) \right] \right\}$$

(Graph 4.2.1 : distance maximizing function)

It's easiest to think of a hyperplane in 2D, i.e. 2 pre-trained categories, a line that separates data (Graph 4.2.2). This line is designed to maximize the yellow margin between the blue dot and red dot. When dimension increases with the number of categories, the line will elevate higher space (Graph 4.2.3).



(Graph 4.2.2: hyperplane in 2D is the black line in the middle)



(Graph 4.2.3: hyperplane is the black line that separates 3categories, (Gerrit))

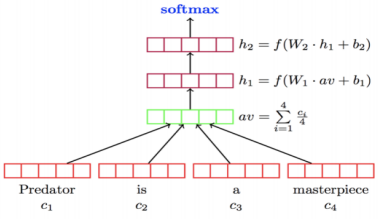
For the implementation of SVM, we utilize sklearn's Stochastic Gradient Descent Classifier (SGD), an optimization algorithm within which SVM is embedded. SDG classifier is known for upscaling of large datasets and sparse data sets which is useful in our case.

4.3 Deep Averaging Network

Deep Averaging Network is a simple Neural Network with high accuracy. DAN accepts data embedded by models like Word2Vec. The neural network takes the sum or the average of the embedding vector and transfers it through one or multiple feed-forward layers, commonly known as Multi-Layer Perceptrons (MLPs). After each hidden layer, the output is passed into a ReLU activation function.

When data reaches the final layer of the Network, the system classifies documents with a nonlinear sigmoid activation function (Shervin et al., 2020). This allows the neural network to map the input data to a higher dimensional space to assign a decision boundary.

Our implementation of DAN is illustrated in Graph 4.3.1. (Iyyer et al.). The input data goes through three Linear layers from PyTorch's nn Module. The first linear layer maps the input size of 300 to 120, and activates the output with the ReLU function. The second linear layer maps the dimension from 120 to 84, again activated by ReLU at the end. The third linear layer maps the 84 dimension to 9, which is the output dimension.



(Graph 4.3.1: Deep Averaging Network)

4.4 Bi-directional LSTM

The third algorithm we use for comparison is the regularized Bi-directional LSTM model (BiLSTM) proposed by Adhikari et al. in "Rethinking Complex Neural Network Architectures for Document Classification". This is a Recurrent Neural Network model that yields high accuracy results on document classification with rather simple architectures.

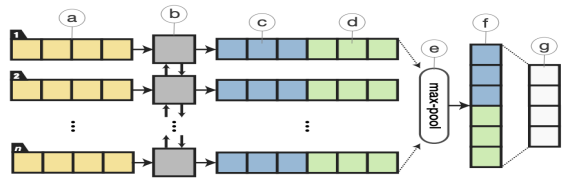


Figure 1: Illustration of the model architecture, where the labels are the following: (a) input word embeddings (b) BiLSTM (c, d) concatenated forward $h^f_{1:n}$ and backward $h^b_{1:n}$ hidden features (e) max-pooling over time (f) document feature vector (g) softmax or sigmoid output.

(Graph 4.4.1: illustration of our implementation of the BiLSTM algorithm)

In the implementation of the algorithm, we first used a weight-dropped BiLSTM layer from PyTorch's nn module. Weigh-dropped LSTM is a regularization technique within LSTMs. It comprises eight input–hidden and hidden–hidden weight matrices in total. Within this, Merity et al. (2018) regularizes the four hidden–hidden matrices with DropConnect (Wan et al., 2013). The operation is applied only once per sequence, using the same dropout mask across multiple timesteps. Conveniently, this feature allows practitioners to use fast, out-of-the-box LSTM implementations without affecting the RNN formulation or training performance. The weight-drop wrapper takes a dropout rate of 0.1, and the wrapped BiLSTM layer is a single bidirectional layer that maps the input data to a hidden dimension of 512. In addition, BiLSTM layer output is activated by ReLU and is max-pooled across time. The layer is then followed by a dropout layer with dropout rate of 0.5. After that, we use two linear layers to reduce the bidirectional dimension from 1024 to 512, and from 512 to the output dimension of 9, which is the number of target classes. Note that we do not use the embedding dropout technique as mentioned in Adhikari et al., primarily because our data has been embedded before feeding into the model. As a remedy, we remove stop words and invalid text and labels to ensure the model is robust and not relying on a small set of input words for prediction.

5 EVALUATION

For evaluation, we implement a confusion matrix to measure the performance of each system based on the macro-average of accuracy, precision, recall and F1-score. We generate all 4 measurement scores for each category and take the average sum.

In 5.1, we will first talk about why we choose a macro-average rather than micro-average for all 4 metrics. In 5.2, we will compare the results of the four systems. In 5.3, we discuss some other factors of our systems and how they affect our results. Namely, we talk about Word2Vec vs. tf-idf, the impact of data distribution on results, SVM vs. Neural Networks, and error analysis on specific classes.

5.1 Marco-Average vs. Micro-Average

Note that we chose macro-average, rather than micro-average for all above measurement scores.

The macro-average calculates the metric independently for each class and then takes the average. This formula treats each class equally. However, a micro-average takes different weights of multi-classes into account to compute the average metric. To illustrate, if a data is comprised of:

Class A: 1 TP and 1 FP

Class B: 10 TP and 90 FP

Then:

Precision(macro-avg) = (0.5+0.1)/2 = 0.3

Precision(micro-avg) = (1+10)/(2+100) = 0.1

We see how micro-average precision doesn't reflect the proportion of input data, which yields a relatively inaccurate score than the macro-average. Therefore, because we have an unbalanced dataset, the macro-average yields a more reliable result since it normalizes the weight of the skewed data.

5.2 Summary of Evaluation Results

Macro-Avg					
UNBALANCED DATA					
SYSTEMS	Accuracy	Error rate	Precision	Recall	F1-score
NB	0.81	0.19	0.9	0.59	0.67
SVM	0.963	0.037	0.911	0.558	0.692
DAN	0.960	0.040	0.830	0.710	0.760
BiLSTM	0.970	0.030	0.880	0.770	0.820
BALANCED DATA					
SYSTEMS	Accuracy	Error rate	Precision	Recall	F1-score
NB	0.906	0.094	0.809	0.577	0.673
SVM	0.902	0.098	0.798	0.558	0.657
DAN	0.930	0.070	0.820	0.700	0.750
BiLSTM	0.950	0.050	0.860	0.770	0.810

(Graph 5.2.1: summary of macro-average of 4 systems: NB, SVM, DAN, BiLSTM)

UNBALANCED DATA					
SYSTEMS	Accuracy	Error rate	Precision	Recall	F1-score
NB	0.81	0.19	0.9	0.59	0.67
SVM	18.84%	-80.30%	1.24%	-5.37%	3.34%
DAN	18.52%	-78.95%	-7.78%	20.34%	13.43%
BiLSTM	19.75%	-84.21%	-2.22%	30.51%	22.39%
BALANCED DATA					
SYSTEMS	Accuracy	Error rate	Precision	Recall	F1-score
NB	0.906	0.094	0.809	0.577	0.673
SVM	-0.46%	4.46%	-1.39%	-3.28%	-2.50%
DAN	2.66%	-25.59%	1.34%	21.39%	11.37%
BiLSTM	4.87%	-46.85%	6.28%	33.53%	20.28%

(Graph 5.2.2: in percentage growth)

For Graph 5.2.1 and 5.2.2, we used NB as our base system and below we compare SVM, DAN, BiLSTM vertically.

For the unbalanced test set, all 3 systems' accuracy is more than 18% greater when compared to NB; and with error rates around 80% lower than the base, the errors of the three systems are relatively small.

In terms of precision, SVM is slightly better than NB, while DAN, BiLSTM are 7% and 2% less than NB, respectively. Since precision = TP/(all Positive) means the percentage of all true correct of all correct results, the data classified by SVM has a better chance of being truely correct.

That said, DAN and BiLSTM covers a significantly larger range of correct answers compared to SVM by outperforming NB by 20% and 30% for recall while SVM underperforms NB by 5%. This potentially shows the limitation of SVM and NB's TF-IDF word vectorization technique to DAN and BiLSTM's Word2Vec. Moreover, the low recall score for SVM might be explained by the model's linearity. Since the model is linear, the hyperplane can fit less accurately when it comes to higher dimensions, compared to the multilayer-models in DAN and BiLSTM.

The high performance of BiLSTM and DAN is primarily rooted in its ability to learn from semantic contexts. Both systems are capable of learning order dependence in sequence prediction problems through multi-layer networks. This property is especially useful in social media posts, where context matters the most.

In addition, the bidirectional aspect of LSTM allows the network to learn from both the previous and backward contexts which further improves the results with more semantic information.

However, DAN loses information when taking the average vector of the sentence, which explains why it slightly underperforms BiLSTM with respect to average accuracy (18% to 29%) and recall (20% vs. 30%), F1-score (12% to 21%) with the unbalanced data.

The pattern on the unbalanced dataset also applies to the balanced dataset. However, the only difference in the balanced set is that accuracy for all 4 systems are relatively the same and error rate increases by 3% .

5.2.1 SVM performance

For both balanced and unbalanced datasets, SVM and NB perform relatively the same – with around 90% macro-average accuracy and 80% precision as shown in Graph 5.2.1. However, for the balanced data, NB's

F1-score 0.673 slightly outperforms SVM's F1-score 0.657. Yet for the unbalanced data, the situation is reversed – SVM's F1-score 0.692 outperforms NB's F1-score 0.670. This is possibly a result of the fact that our training data is skewed.

unbalanced diff(NB-SVM)	Accuracy	Error rate	Precision	Recall	F1-score
coronavirus	-0.112	0.112	-0.171	0.534	0.385
CryptoCurrency	-0.012	0.012	-0.019	0.003	-0.010
cyberpunk	0.041	-0.041	0.068	-0.280	-0.318
jokes	-0.028	0.028	-0.006	0.007	0.006
recipes	-0.024	0.024	-0.149	0.129	0.132
Showerthoughts	0.092	-0.092	0.188	-0.020	0.077
suicideanddepression	-0.006	0.006	-0.033	-0.033	-0.033
texas	0.000	0.000	0.011	-0.036	-0.039
wallstreetbets	-0.001	0.001	0.039	-0.065	-0.027
TOTAL (Macro)	-0.005	0.005	-0.008	0.027	0.018
TOTAL (Micro)			-0.025	-0.025	-0.025

(Graph 5.2.1.1: SVM-NB difference unbalanced data)

balanced diff(NB-SVM)	Accuracy	Error rate	Precision	Recall	F1-score
coronavirus	-0.117	0.117	-0.664	0.410	-0.034
CryptoCurrency	-0.009	0.009	0.003	0.090	0.016
cyberpunk	-0.010	0.010	0.170	-0.170	-0.170
jokes	0.006	-0.006	-0.103	0.120	0.097
recipes	0.005	-0.005	-0.038	0.050	0.064
Showerthoughts	0.179	-0.179	0.675	-0.090	0.378
suicideanddepression	0.004	-0.004	0.058	-0.050	0.004
texas	-0.010	0.010	0.000	-0.090	-0.099
wallstreetbets	-0.011	0.011	0.000	-0.100	-0.058
TOTAL (Macro)	0.004	-0.004	0.011	0.019	0.017
TOTAL (Micro)	0.000	0.000	0.019	0.019	
*if positive, NB performed better					

(Graph 5.2.1.1: SVM-NB difference balanced data)

When we zoom in to see the break-down across the categories, both algorithms performed similarly for all categories except for the "coronavirus" class – NB outperformed SVM by 41% for recall and SVM outperformed NB by 64% for accuracy in the same category. The case applies similarly to an unbalanced data set.

Thus, with similar metrics and compensations, we can only conclude that SVM performs relatively similarly to NB, with one suffering more loss in recall and the other in precision.

5.2.2 DAN performance

DAN	Accuracy	Error rate	Precision	Recall	F1-score
coronavirus	0.994	0.006	0.892	0.439	0.589
CryptoCurrency	0.885	0.115	0.866	0.908	0.886
cyberpunk	0.951	0.049	0.851	0.479	0.613
jokes	0.976	0.024	0.716	0.494	0.584
recipes	1.000	0.000	0.888	0.936	0.912
Showerthoughts	0.887	0.113	0.750	0.865	0.803
suicideanddepression	0.972	0.028	0.890	0.784	0.834
texas	0.992	0.008	0.802	0.559	0.659
wallstreetbets	0.997	0.003	0.852	0.891	0.871
Total(macro avg)	0.960	0.040	0.830	0.710	0.760
Total(micro avg)			0.830	0.830	0.830

DAN	Accuracy	Error rate	Precision	Recall	F1-score
coronavirus	0.943	0.057	1.000	0.490	0.658
CryptoCurrency	0.859	0.141	0.435	0.900	0.586
cyberpunk	0.939	0.061	0.895	0.510	0.650
jokes	0.926	0.074	0.867	0.390	0.538
recipes	0.992	0.008	1.000	0.930	0.964
Showerthoughts	0.834	0.166	0.385	0.820	0.524
suicideanddepression	0.963	0.037	0.885	0.770	0.824
texas	0.944	0.056	0.946	0.530	0.679
wallstreetbets	0.992	0.008	1.000	0.930	0.964
Total(macro avg)	0.930	0.070	0.820	0.700	0.750
Total(micro avg)			0.700	0.700	0.700

(Graph 5.2.2.1 result of DAN. Upper: unbalanced test set; lower: balanced test set)

For both balanced and unbalanced test sets, DAN outperformed NB on the macro-average F1-score by 15%, recall by 25%, and precision by 3%, as shown in Graph 5.2.1. The F1-score of each category all outperformed our baseline. Hence we conclude that DAN has a low false positive rate, but a relatively high false negative rate. For some categories, the algorithm struggled and gave a far below-average F1-score, such as "showerthoughts" at 0.524 and "jokes" at 0.538 in the balanced test set. This is mostly due to the fact that the subject of sentences in these categories does not fall into any specific domain. Humans can easily tell if a sentence is a "joke" or a "showerthoughts" by intuition. However, algorithms can only perceive the objective meaning of the sentence, failing to understand the semantic humour within the sentence. The ability to recognize such a high-level semantic pattern is currently beyond the ability of any natural language processing system.

5.2.3 BiLSTM performance

On the unbalanced dataset, BiLSTM outperforms NB on every metric except for precision. As seen in Graph 5.2.1, for precision, Naive Bayes is at 0.9, and BiLSTM is at 0.88. The 0.02 difference is not very significant, but

BiLSTM is also outperformed in precision by SVM at 0.911. Aside from precision, BiLSTM is able to outperform all other algorithms for the rest of the metrics.

On the balanced dataset, BiLSTM outperforms Naive Bayes on every metric. It is also the best performing algorithm on every measurement score. The outstanding performance of BiLSTM, as previously mentioned, comes from its ability to extract order dependence from the contexts.

5.3 Other Factors

5.3.1 TF-IDF vs. Word2Vec

Our systems use 2 different vectorization methods – TF-IDF and Word2Vec. Specifically, NB and SVM use TF-IDF, and DAN and BiLSTM employ Word2Vec.

TF-IDF is the product of term frequency and inverse document frequency. It is a probabilistic measure that reflects how relative a word is to a class. TF-IDF performs well when the document is long, since more word frequencies can be trained to establish the probabilistic relevance of each word to document more accurately. Rather than a probabilistic construct, Word2Vec gives semantic word-embeddings through a shallow, 2 layered neural network. It considers literary contexts and syntactic patterns when grouping words that indicate semantic similarity.

In our result, we speculate Word2Vec to outperform TF-IDF because reddit posts are relatively short. TF-IDF trains better on larger documents. This fact is distinct when both algorithms that use TF-IDF (NB & SVM) yield a macro-average recall of around 0.57 versus 0.70 from Word2Vec algorithms (DAN & BiLSTM) in Graph 5.2.1.

However, this is an assumption that requires further testing. Horizontal comparisons for the algorithms should be conducted by implementing Word2Vec on NB and SVM, and TF-IDF on Dan & BiLSTM.

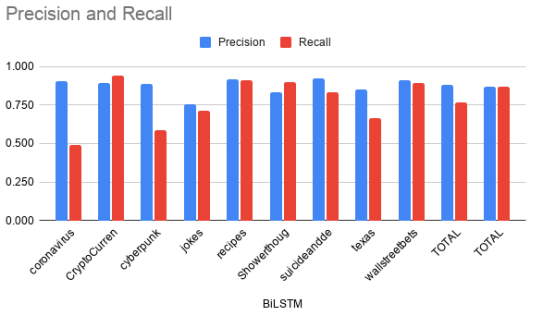
5.3.2 The Impact of Data Distribution on Result

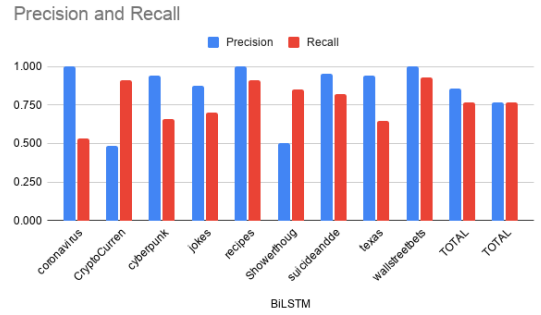
Because the training data had a skewed distribution shown in Graph 3.1.1, the average recall scores for both the skewed test and balanced test sets are all lower than their corresponding average precision scores. In addition, the average error rates across systems are higher in the balanced set than the unbalanced – average error rate of balanced data set is 0.078, 47% higher than error rate of unbalanced data at 0.053 (Graph 5.2.1).

BALANCED DATA					
SYSTEMS	Accuracy	Error rate	Precision	Recall	F1-score
NB	0.906	0.094	0.809	0.577	0.673
SVM	0.902	0.098	0.798	0.558	0.657
DAN	0.930	0.070	0.820	0.700	0.750
BiLSTM	0.950	0.050	0.860	0.770	0.810
Unbalanced Difference					
SYSTEMS	Accuracy	Error rate	Precision	Recall	F1-score
NB	11.84%	-50.49%	-10.09%	-2.26%	0.51%
SVM	-6.32%	162.57%	-12.43%	-0.10%	-5.17%
DAN	-3.12%	75.00%	-1.20%	-1.41%	-1.32%
BiLSTM	-2.06%	66.67%	-2.27%	0.00%	-1.22%

(Graph 5.3.2: Unbalanced data as a percentage with respect to balanced data. formula = (balanced - unbalanced)/balanced)

Since recall is the measure of correctly identified true positives, the models only performed decently well on the categories with the most data in the training set; this includes “cryptocurrency” and “suicideanddepression” as seen in Graph 5.3.2.2.



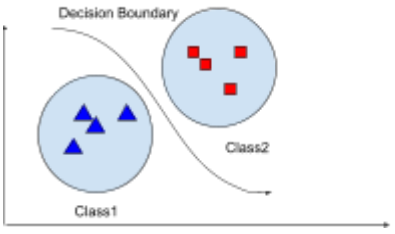


(Graph 5.3.2.2: Precision and Recall comparison for unbalanced and balanced BiLSTM system)

Because the other six categories had much fewer data points available during training, recall for those were noticeably lower for all algorithms. On the other hand, precision calculates the ratio between the true positives and all positives that exist in the corpus; each score for the categories as well as the average for the unbalanced and balanced sets for the four models were much higher.

5.3.3 SVM vs. Neural Networks

To allocate a decision boundary, SVM and Neural Network will map the input data to a higher-dimensional space. Kernel tricks are used in SVM, the linear model; while non-linear activation functions are used in Neural Networks.



With different methods, both types of algorithms can approximate nonlinear decision functions. However, the Neural Network outperforms the SVM because of a variety of techniques, such as:

1. Multilayer: increases the network's learning power
2. Batch normalization: allows the model to learn the best scale and mean for each layer's training data
3. Efficient Optimizers: measure backpropagation signals, which aids the net in changing neuron weights across all layers
4. Dropout: helped networks avoid overfitting

5.3.4 Error Analysis on Specific Classes

1) The “recipes” class

UNBALANCED DATA						
		Accuracy	Error rate	Precision	Recall	F1-score
DAN	recipes	1.000	0.000	0.888	0.936	0.912
SVM	recipes	1.000	0.000	1.000	0.227	0.370
BiLSTM	recipes	1.000	0.000	0.917	0.909	0.913
NB	recipes	0.911	0.089	1.000	0.200	0.333
BALANCED DATA						
		Accuracy	Error rate	Precision	Recall	F1-score
DAN	recipes	0.992	0.008	1.000	0.930	0.964
SVM	recipes	0.911	0.089	1.000	0.200	0.333
BiLSTM	recipes	0.990	0.010	1.000	0.910	0.953
NB	recipes	0.916	0.084	0.962	0.250	0.397

(Graph 5.3.4.1: “recipe” class on both unbalanced and balanced data)

We notice one prominent trend across all of our algorithms: the number of samples for the “recipes” class consists only 0.029% of all the test entries. The accuracy is close to or exactly 1, while error rate gears toward 0 for this category. Precision scores across all 4 systems are relatively high around 90%-100%.

After looking into the recipe posts, we noticed many of the texts contain the actual word “recipe” in it. This word is a strong indicator for this particular classification. For posts that do not contain the keyword, they tend to contain terms that fall into food categories (e.g. “vanilla”, “nutmeg”) and verbs for cooking (e.g. “preheat”, “cook”). The vocabulary for this classification is quite specific to activities related to cuisines, making them strong indicators for the “recipes” class as well.

All systems resulted in high precision scores. However, only DAN and BiLSTM had high recall scores, and NB and SVM scored low (around 0.2). This is likely due to the limited number of training data points for the “recipes” category for NB and SVM to use TF-IDF to obtain frequency of the words. DAN and BiLSTM’s Word2Vec vectorization

method does a better job at handling this situation because it extracts the semantic meaning of the words.

2) The “coronavirus” class

BALANCED DATA		
coronavirus	Precision	Recall
NB	1	0.22
SVM	1	0.33
DAN	1	0.49
BiLSTM	1	0.53

On the balanced test set, all algorithms get 100% for precision, but score significantly lower for recall. While DAN and BiLSTM score higher than the other two, all four systems have a relatively low recall score. This means that the systems are confident with classifying text to “coronavirus”, however, in ambiguous circumstances when the situation is unclear, it fails to categorize posts in “coronavirus”.

Like “recipe”, “coronavirus” posts also contain strong indicators – proper nouns of the virus, vaccines, scientific terms and names of organizations, which explains high accuracy.

Similarly, recall is low for “coronavirus”. We suspect this low recall might stem from the casual style of writing that is prevalent in social media texts – some users omit the subjects in their posts. For example, people write “clinical trials" instead of “clinical trials for coronavirus vaccines”, which could explain the low recall. Another potential factor for the low recall score is deleted posts. There is a fair number of data with “comment removed” in place of the original text. This adds noise to the data, resulting in low recall.

6 CONCLUSION

We implement 4 supervised document classification systems:

1) Naive Bayes (NB)
2) Support Vector Machine (SVM)
3) Deep Averaging Network(DAN)
4) Bi-directional LSTM (BiLSTM)

Among the four, we treat NB as a base system for its high accuracy and speed. We first separated unbalanced data into training, developing and testing data. Then, we constructed a balanced test set to evaluate the 3 systems.

For our data preparation, we tokenized, lowercased, remove-stop-words, and lemmatized, using nltk’s library. For NB and SVM, we represented document features using TF-IDF. While for DAN and BiLSTM, we utilized Word2Vec to extract word vectors under semantic classification.

For the evaluation process, we computed the confusion matrix and calculated accuracy, precision, recall, and F1-score across each category. Due to our skewed training data, we took the macro-average of each metric to compare results.

Through careful evaluation, SVM performs similarly to our base system, in both balanced and unbalanced data. BiLSTM and DAN, however, significantly outperformed NB (and SVM) in all macro-averaged measurements scores. Furthermore, BiLSTM performed better than DAN for all metrics. We infer 3 potential reasons for above conclusions:

1) BiLSTM and DAN utilizes Word2Vec which takes into account the semantic meaning of the texts. This is significant in terms of analyzing the social media’s sentiment-heavy texts. On the other hand, SVM and NB only employ TF-IDF which is a probabilistic vector representation that only performs well under long texts which our Reddit data lacks. However, this result needs further horizontal testing.

2) SVM is a linear model which performs worse in multi-class high-dimension space. And this is no problem for DAN and BiLSTM as their multi-layers are designed to take in high-dimensional data.

3) BiLSTM takes account of the order of documents that it trains on previous and backward contexts. This

explains its 10% more recall and precision from DAN, which only takes the average of the vector.

Eventually we also delve into some other factors such as the impact of skewed data distribution on our results. We realize that the heavier-weighted classes containing the majority of the documents tend to have better recall. This implies that the more documents are trained under a category, the more positive results the system generates. We also specify the difference between SVM and neural networks based on our result, and analyze factors in posts of certain categories that contribute to especially high or low evaluation results.

7 REFERENCE

Adhikari, Ashutosh, et al. "Rethinking Complex Neural Network Architectures for Document Classification." *ACL Anthology*, June 2019, www.aclweb.org/anthology/N19-1408/.

Burg, G. V. D. and P. Groenen. "GenSVM: A Generalized Multiclass Support Vector Machine." J. Mach. Learn. Res. 17 (2016): 225:1-225:42.

https://jmlr.org/papers/v17/14-526.html

Iyyer, M., Manjunatha, V., Boyd-Graber, J., & III, H. (n.d.). Deep unordered composition rivals syntactic methods for text classification. Retrieved April 06, 2021, from https://www.aclweb.org/anthology/P15-1162/

Stephen Merity, Nitish Shirish Keskar, and Richard Socher. 2018. Regularizing and optimizing LSTM language models. In International Conference on Learning Representations. https://arxiv.org/abs/1708.02182

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2020. Deep Learning Based Text Classification: A Comprehensive Review. 1, 1 (January 2020), 43 pages. https://arxiv.org/abs/2004.03705

Li Wan, Matthew Zeiler, Sixin Zhang, Yann Le Cun, and Rob Fergus. 2013. Regularization of neural networks using dropconnect. In International Conference on Machine Learning, pages 1058–1066. http://proceedings.mlr.press/v28/wan13.html

Dataset:
www.kaggle.com/gpreda/reddit-wallstreetsbets-posts.
www.kaggle.com/cuddlefish/reddit-rjokes.
https://www.kaggle.com/nickreinerink/reddit-rcryptocurrency

https://www.kaggle.com/nikhileswarkomati/suicide-watch
https://www.kaggle.com/jaedebug/rtexas-reddit-posts (title)
https://www.kaggle.com/maxspunnring/reddit-rdankmemes-top-1000-memes (title)
https://www.kaggle.com/xhlulu/covid19-vaccine-news-reddit-discussions
https://www.kaggle.com/matheusdalbuquerque/rcyberpunkgame-posts?select=Posts.csv
https://www.kaggle.com/vishxl/showerthoughts (title)
https://www.kaggle.com/viktorarsovski/randroiddev-data-20152019