CSE 256_FA24 Project: Sentiment Analysis for HappyDB Dataset

Author - Anandhini Rajendran
anrajendran@ucsd.edu

1 Introduction

In this project, I have performed sentiment analysis on the HappyDB dataset (Asai et al., 2018). The dataset comprises textual data along with happy emotion categories. The objective of this project is to analyze the provided text and classify sentiment categories. The list of tasks:

- Collected and preprocessed dataset: DONE.
- Build and train baseline on the collected dataset and examine its performance: DONE
- Build and train better models than baseline on the dataset: DONE
- ~~Run Bert(Souza and Filho, 2022), VADER(?), and implement (Zanwar et al., 2022)~~: NOT DONE: Failed due to computing capacity and runtime.
- Analyse results using time, accuracy and error prediction data: DONE

2 Related work

This section provides an overview of related research studies in sentiment analysis and emotion classification.

2.1 Sentiment Analysis

The study presented in (Hutto and Gilbert, 2014) introduces a rule-based sentiment analysis method for text collected from social media platforms. In (Souza and Filho, 2022), an experimental study explores different strategies for aggregating features obtained from the BERT output layer for sentiment analysis tasks. The BERT models used in this study are trained on the Brazilian Portuguese corpus. The results show that BERT achieved the highest ROC-AUC values compared to TF-IDF approach. There are a lot of studies involving various non-English languages like (Zhang et al., 2020), (Nguyen et al., 2020) and (Nugroho et al., 2021) which do fine-tuning of BERT for their datasets. For context-sensitive sentiment analysis (Suruthi et al., 2015) presents a neural network which works for ambiguous words. This approach is useful, especially for happyDB-like datasets where there is a subtle difference in positive emotions text.

2.2 Emotion Classification

The study in (Basile et al., 2021) presents a model for emotion classification that doesn't need a large annotated corpus for training. It uses zero-shot and few-shot learning combined in the ensemble model framework. The work by (Abaskohi et al., 2022) explores the challenges of handling imbalanced data for emotion detection, using a Persian text dataset. The authors of (Wang et al., 2020) employ LSTM-based encoders that capture self and inter-speaker dependency, the method helps in modelling the emotional consistency and enables the model to outperform the current state-of-the-art methods on multiple emotion classification datasets. These studies demonstrate the research in emotion classification and approaches to handle data scarcity and context dependencies.

3 Your dataset

HappyDB dataset contains a collection of sentences which is collected by surveying crowd workers with the question: 'what made you happy in a particular time period?' as described in (Asai et al., 2018). The HappyDB dataset contains 100,535 rows of data with the following features: *hmid*, *wid*, *reflection_period*, *original_hm*, *cleaned_hm*, *modified*, *num_sentence*, *ground_truth_category*, and *predicted_category*. The *original_hm* feature contains the textual

data used for sentiment analysis, while the *predicted_category* feature comprises seven categories: *'affection'*, *'exercise'*, *'bonding'*, *'leisure'*, *'achievement'*, *'enjoy_the_moment'*, and *'nature'*. To ensure effective training and performance, the distribution of text across these categories must not be skewed. The Figure.1 shows the distribution of *predicted_category* feature.

To further understand the corpus and distribution of text, I plotted the visual representation of the frequency of words as shown in Figure.2. It's interesting to observe that *affection* category includes words such as 'family', 'happy', 'made', 'dog' etc. which align well with the context for affection. To further analyze the data, I generated N-grams for each category. The plot for *Achievement* category is shown in Figure.3. It's amusing to see how the words *bought* and *getting*, and *new* have a higher frequency than *work*, *job* or *time*.

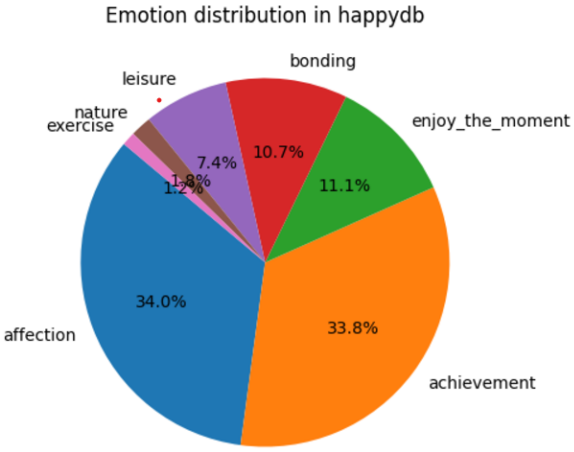

Figure 1: Category Distribution

3.1 Data preprocessing

I performed data preprocessing steps such as lowercasing, removing whitespace, eliminating invalid characters, and other cleaning operations. Interestingly, removing single-letter words and stop words significantly improved the accuracy of the models. Additionally, I tokenized the data to prepare it for further analysis and applied lemmatization to enhance model performance. Implementing these preprocessing techniques resulted in a notable increase in accuracy compared to the baseline.

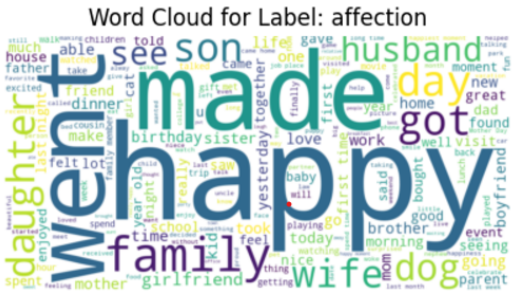

Figure 2: Word cloud

3.2 Data annotation

The dataset was originally labelled in a string format, whereas the models require input labels in a float format. To address this, I used a mapping function to convert the labels to integers and then converted them back to their original format after the prediction step. The string labels were consistent and required no additional cleaning or processing apart from the conversion to float. After processing and annotating the data, I split the dataset into an 80% training set and a 20% testing set for all the models.

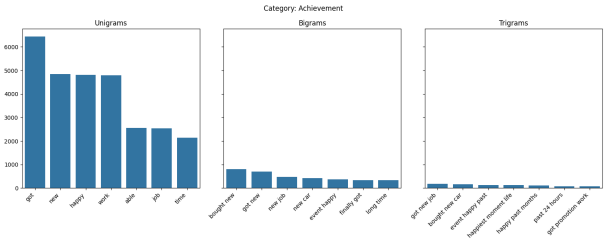

Figure 3: N-grams plot

4 Baselines

The baseline models I used were decision trees and random forests. Decision trees were selected because the data in emotion sentiment analysis is categorical. Random forests, being an ensemble of decision trees, typically provide higher accuracy for such datasets. These models are commonly used for classification tasks due to their robust performance, making them a logical starting point for the baseline. When providing the dataset without feature selection to the random forest model, the code took a long time to execute. Random forests often require feature extraction and dimensionality reduction to improve efficiency. To address this, I applied Principal Component Analysis (PCA) and ran the models on the reduced feature set. For the

random forest model, I used 100 estimators. The baseline runtime results are presented in Table 1.

Additionally, for each model, I printed 15 incorrect predictions to analyze the results in detail. The code for all models has been uploaded to Gradescope. I used basic libraries to implement the models and tuned hyperparameters and layers as needed. All the models were run on my personal computer.

5 My approach

I experimented with popular sentiment analysis models spanning various categories, including simple regression, neural networks, and ensemble methods such as random forests. For many of these models, I used TF-IDF vectorization for feature extraction.

5.1 Regression

I experimented with both logistic regression using a one-vs-rest classifier and linear regression using one-hot encoding (selecting the top k=100 features) from the scikit-learn library. These models were chosen for their simplicity and effectiveness in classification tasks. Surprisingly, they performed on par with neural network-based models. The results for the linear regression model are shown in Table 1. A notable drawback of these models is their inability to capture complex relationships in the data due to their inherent linearity.

5.2 LightGBM Classifier

This classifier achieved the best performance among the models I have tested, considering both time and accuracy. This model uses a tree based learning algorithm. It is known for its speed and accuracy on large datasets. Additionally, It requires less memory and time compared to other gradient boost methods. To further optimize this algorithm, I tuned different hyperparameters like *no_of_estimators*,*learning_rate*,*max_depth*,*num_leaves*, etc. However, the results indicated no further splits with positive gain, suggesting that changing it to a more complex model may cause overfitting. This classifier is implemented using the lightgbm library.

5.3 XGBoost

XGBoost is another gradient-boosting framework known for its strong performance. I used this classifier from the built-in XGBoost library. I experimented with different hyperparameters, including the number of estimators (100, 300), learning rate (0.01, 0.1), and max_depth (5, 7). The best accuracy was achieved with 300 estimators, a learning rate of 0.1, and a max_depth of 5. Increasing the number of estimators improved accuracy but at the cost of increased runtime. Therefore, for real-world use cases, it is important to consider the tradeoff between time and accuracy when selecting the optimal hyperparameters.

5.4 Naive Bayes

I used the Multinomial Naive Bayes classifier from the scikit-learn library. It took only 0.08 seconds to run, and the results are shown in Table 1. This model assumes feature independence, which works well for text data. Although it offers the best runtime, its accuracy is lower compared to more complex models.

5.5 Neural Network Models: RNN,LSTM,BiLSTM,BiGRU

For these models, I performed sequencing and padding using the maximum length from the training data for the text. These models were implemented using the TensorFlow Keras library. I experimented with different values for epochs, batch size (32, 64), layers, etc. More complex networks required significantly more time to run, so I was only able to test the BiLSTM model using GloVe, and not the other modified versions present in the submitted code. The BiGRU model was faster than BiLSTM but still took several hours to run. Therefore, the results shown in Table 1 are limited to the RNN, LSTM, and BiLSTM models using GloVe. LSTM achieved the best accuracy among these models.

5.6 Advanced Models: BERT, VADER, RoBERTa

While running these models, I encountered issues with the torch version on my computer. As a result, I switched to Google Colab, but the models took a very long time to run, and I was unable to complete them. These models will be explored in future work.

6 Results

The graph in Figure 4 shows the runtime for different models. We can see that the Multinomial Naive Bayes model is the fastest, while the LSTM takes the longest to run. The accuracy, RMSE,

Model	Accuracy	RMSE	F1 Score	Runtime (s)
Random Forest	0.7639	1.4250	0.7469	514.49
Decision Tree	0.6128	1.8645	0.6151	41.84
Regression	0.8920	0.7212	0.9248	48.35
LightGBM Classifier	**0.8948**	**0.8568**	**0.8934**	**45.10**
XGBoost (N=300)	0.8649	0.9781	0.8614	245.21
XGBoost (N=100)	0.8215	1.1593	0.8142	95.42
Multinomial Naive Bayes	0.7848	1.5001	0.7710	0.08
BiLSTM with Glove	0.8619	–	–	–
RNN	0.8758	–	–	3496.56
LSTM	**0.8953**	**0.9201**	**0.8945**	**15783.08**

Table 1: Comparison of Model Performance on HappyDB Dataset

and F1 score graph in Figure 5 reveal that LightGBM achieves the highest accuracy and F1 score, as well as the lowest RMSE, making it the best-performing model among those compared. LSTM also performs very well but has a longer training time. The gradient boosting models, LightGBM Classifier and XGBoost show good performance, with accuracies and F1 scores above 0.85. The Decision Tree model has the lowest accuracy and F1 score, as well as the highest RMSE, indicating it is the least effective model for this task. The Bidirectional LSTM and RNN models show competitive accuracy, but they have long training times. Random Forest and Multinomial Naive Bayes show moderate performance, with accuracies around 0.78, with Multinomial Naive Bayes taking the least time to run.

The Table1 identifies the trade-offs between runtime and performance across the different models tested on the HappyDB dataset. LightGBM has the best balance in terms of both accuracy and runtime, while neural network-based models offer high accuracy but at the cost of significantly longer computation times. Choosing a model for sentiment classification task would depend on the specific requirements of the task, balancing accuracy, speed, and computational resources.

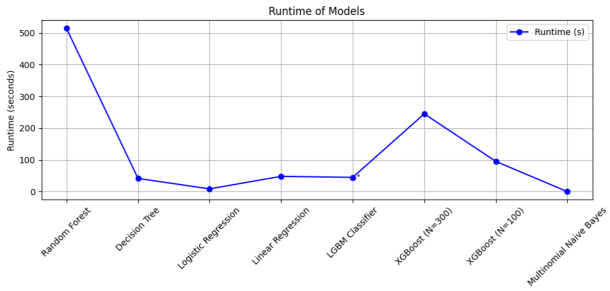

Figure 4: Runtime of models

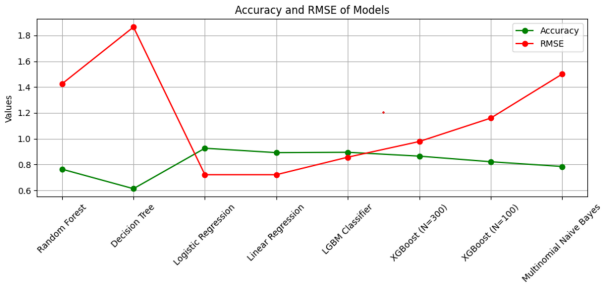

Figure 5: Performance of models

7 Error analysis

To conduct this error analysis, I am printed the first 15 wrong predictions for all the models. The sample example predictions are

- True: achievement, Predicted: affection
- True: leisure, Predicted: achievement
- True: bonding, Predicted: enjoy_the_moment
- True: leisure, Predicted: achievement

Many models confuse the 'achievement' and 'affection' categories, implying these categories may have linguistic similarities. The error predictions for 'enjoy_the_moment' are consistently predicted as 'achievement' across all models, implying these categories have similar words in the text. 'Leisure' is often misclassified as 'achievement' or 'affection', which may be due to the train data distribution as shown in Figure1 or because leisure words are described with language related to personal accomplishments or relationships. This makes the study interesting as it reveals how human happiness varies.

On a model-level analysis, XGBoost tends to overpredict the 'achievement' category. Naive Bayes often predicts 'affection' for other categories suggesting it may be overestimating the prior probability of the 'affection' class. LightGBM has a balanced error distribution but struggles with 'achievement' and 'affection' predictions. LSTM exhibits unique errors, such as predicting 'bonding' for the 'achievement' category. These observations show that the models have difficulty distinguishing very minute differences in positive experiences. The confusion between 'enjoy_the_moment' and 'achievement' could be because present enjoyment can be described as past accomplishments and vice versa.

The future works must include models that capture subtle differences in language and augment train data with challenging examples. These steps help models differentiate between closely related sentiment categories.

8 Future work

For dataset selection, I explored multiple datasets like the IMDB movie sentiment analysis, Twitter sentiment dataset and US airline dataset. It would be interesting to extend the study to different datasets, and this would provide a more comprehensive guide to choosing models for real-world sentiment analysis problems. The other possible direction to this problem is to perform sentiment analysis on rare languages or non-English languages like Persian, demonstrated in the papers (Abaskohi et al., 2022) and (Rasouli and Kiani, 2023). Instead of predicting emotion categories, we can do an emoji prediction as an alternative representation to sentiment (Calefato et al., 2017). Finally, we can apply state-of-the-art models like Vader(Hutto and Gilbert, 2014) and RoBerta(Tan et al., 2022) to compare the performance with the current set of models. The recent paper (Zanwar et al., 2022) also is an interesting approach to replicate. These extensions will help provide more generalization to different sentiment analysis techniques.

9 Conclusion

The project gave good insights into the current NLP problems, different models and their respective advantages and disadvantages. This project also shows the complete pipeline of NLP tasks from dataset selection, and processing to getting results.

The gradient-boosting approaches are highly effective for text classification tasks providing optimal balance between speed and performance. LSTM and its variants are the best for accuracy if there are no time constraints. The results for regression were surprisingly good despite its simplicity. The good accuracy of neural models comes with a cost of time.

To expand in other directions, future work can explore multilingual sentiment analysis, and multi-modalities combined with text e.g. image captions, audio transcripts etc. Another approach to try is to do more feature engineering and create hybrid models to improve performance.

10 Acknowledgements

- I used ChatGPT for proofreading and correcting grammar in the report after I had written it myself.
- I also used ChatGPT as a reference to get the syntax for plotting line graphs for runtime and accuracy.
- For the models, I used the respective library's documentation as a reference.

References

Abaskohi, A., Sabri, N., and Bahrak, B. (2022). Persian emotion detection using parsbert and imbalanced data handling approaches. *arXiv preprint arXiv:2211.08029*.

Asai, A., Evensen, S., Golshan, B., Halevy, A., Li, V., Lopatenko, A., Stepanov, D., Suhara, Y., Tan, W.-C., and Xu, Y. (2018). Happydb: A corpus of 100,000 crowdsourced happy moments. In *Proceedings of LREC 2018*, Miyazaki, Japan. European Language Resources Association (ELRA).

Basile, A., Pérez-Torró, G., and Franco-Salvador, M. (2021). Probabilistic ensembles of zero-and few-shot learning models for emotion classification. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 128–137.

Calefato, F., Lanubile, F., and Novielli, N. (2017). Emotxt: a toolkit for emotion recognition from text. In *2017 seventh international conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)*, pages 79–80. IEEE.

Hutto, C. and Gilbert, E. (2014). Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international AAAI conference on web and social media*, volume 8, pages 216–225.

Nguyen, Q. T., Nguyen, T. L., Luong, N. H., and Ngo, Q. H. (2020). Fine-tuning bert for sentiment analysis of vietnamese reviews. In *2020 7th NAFOSTED conference on information and computer science (NICS)*, pages 302–307. IEEE.

Nugroho, K. S., Sukmadewa, A. Y., Wuswilahaken DW, H., Bachtiar, F. A., and Yudistira, N. (2021). Bert fine-tuning for sentiment analysis on indonesian mobile apps reviews. In *Proceedings of the 6th International Conference on Sustainable Information Engineering and Technology*, pages 258–264.

Rasouli, M. and Kiani, V. (2023). Investigating shallow and deep learning techniques for emotion classification in short persian texts. *Journal of AI and Data Mining*, 11(4):587–598.

Souza, F. D. and Filho, J. B. d. O. e. S. (2022). Bert for sentiment analysis: pre-trained and fine-tuned alternatives. In *International Conference on Computational Processing of the Portuguese Language*, pages 209–218. Springer.

Suruthi, S., Pradeeba, M., Sumaiya, A., and Sheeba, J. (2015). Neural network based context sensitive sentiment analysis. *International Journal of Computer Applications Technology and Research*, 4(3):188–191.

Tan, K. L., Lee, C. P., Anbananthen, K. S. M., and Lim, K. M. (2022). Roberta-lstm: a hybrid model for sentiment analysis with transformer and recurrent neural network. *IEEE Access*, 10:21517–21525.

Wang, Y., Zhang, J., Ma, J., Wang, S., and Xiao, J. (2020). Contextualized emotion recognition in conversation as sequence tagging. In *Proceedings of the 21th annual meeting of the special interest group on discourse and dialogue*, pages 186–195.

Zanwar, S., Wiechmann, D., Qiao, Y., and Kerz, E. (2022). Improving the generalizability of text-based emotion detection by leveraging transformers with psycholinguistic features. *arXiv preprint arXiv:2212.09465*.

Zhang, H., Dong, J., Min, L., and Bi, P. (2020). A bert fine-tuning model for targeted sentiment analysis of chinese online course reviews. *International Journal on Artificial Intelligence Tools*, 29(07n08):2040018.