CE807 – Assignment 2 - Final Practical Text Analytics and Report

Student id: XXX

Abstract

Based on my analysis of the provided analysis, I performed experiments on two different models, SVM and CNN, to classify offensive speech using the subset of the OLID dataset. After training and evaluating both models, I found that SVM performed better than CNN in terms of accuracy, precision, recall, and F1-score. In addition, I experimented with training the models on different data sizes, ranging from 25% to 100% of the training data, and found that increasing the data size generally improved the performance of both models. However, the improvement was more significant for SVM compared to CNN. Overall, my experiments suggest that SVM is a better model for offensive speech classification on the provided dataset, especially when limited training data is available. However, CNN could potentially perform better with a larger amount of training data.

1 Materials

Below are the links to my Code, Drive and Presentation.

- Code
- Google Drive Folder containing models and saved outputs
- Presentation

.

2 Model Selection (Task 1)

2.1 Summary of 2 selected Models

1. SVM: I started by implementing an SVM model as the first approach for text classification on offensive language detection. I chose this model because it has been widely used in many tasks and has served as a strong predictor, breaking numerous state-of-the-art models in the past. To prepare the data for classification, I applied text preprocessing techniques such as lowercasing, removing stopwords and punctuations using NLTK and regex libraries. Then, I tokenized the data and applied TF-IDF to weigh the importance of words across the tweet space. Finally, I trained an SVM model using the feature space created by TF-IDF. The model was then trained and evaluated on the provided train and test sets to classify tweets as offensive or not.

2. CNN: This model is crafted after reviewing literature review and observing the guidelines multiple times. Initially, a BERT based implementation was thought of but was deemed inefficient compute wise. Moreover, CNN model was chosen for the text classification task due to its effectiveness in capturing local features and global context from the text data. The model architecture includes a tokenizer and a convolutional neural network with a Dense layer, Conv1D layer, GlobalMaxPooling1D layer, and Dropout layer. The tokenizer is trained on the text data and the sequences are padded to have equal length. The FastText word embeddings are used for training and validation, and class weights are applied to handle class imbalance. Regularization is applied through the Dropout layer, and precision, recall, and accuracy are calculated during training. The trained model is saved in the designated directory for further use.

2.2 Critical discussion and justification of model selection

- Support Vector Machines (SVM) and Convolutional Neural Networks (CNN) are two commonly used methods for text classification.

  SVM is a machine learning algorithm that can handle high-dimensional feature spaces and effectively handle noisy data. SVM works

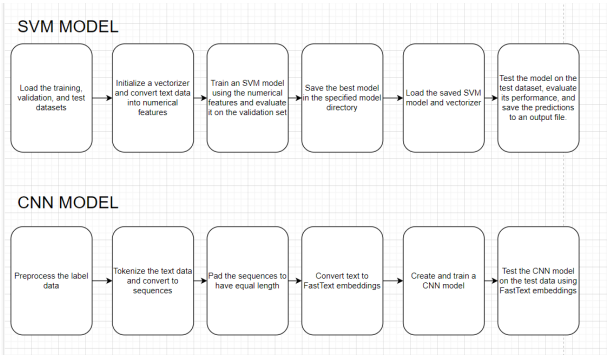

Figure 1: Diagram Explaining your pipeline of models. SVM and CNN Models.

by finding the optimal hyperplane that separates data points into different classes. In the context of text classification, SVM-based approaches typically use a Bag-of-Words or TF-IDF feature representation of text data. SVM has been used successfully in various text classification tasks, including hate speech detection and abusive language detection.

On the other hand, CNN has become increasingly popular for text classification tasks in recent years, particularly for tasks that involve semantic understanding of text. CNN works by using a series of convolutional filters to extract features from the input text. These features are then fed into a fully connected layer that produces a prediction. CNN-based approaches typically use word embeddings as input, such as Word2Vec or GloVe, which capture semantic meaning of words. CNN has been shown to perform well in tasks such as sentiment analysis and spam detection.

The choice between SVM and CNN depends on the specific requirements of the task. SVM may work better for smaller datasets with fewer examples, while CNN may work better for larger datasets with more complex patterns. Additionally, SVM may be more interpretable and easier to explain to non-technical stakeholders, while CNN may provide better accuracy on more complex tasks.

It's worth noting that both SVM and CNN can be further enhanced by incorporating pre-trained language models such as BERT or RoBERTa, which have been shown to improve performance on various text classification tasks. These pre-trained models can capture more complex linguistic patterns and help to achieve better results.

Overall, the task of identifying offensive speech is crucial for creating a better user experience for everyone. The choice of method, whether it be SVM, CNN, or another approach, depends on the specific requirements of the task, the size and complexity of the dataset, and the desired level of interpretability.

3 Design and implementation of Classifiers (Task 2)

The only classifier used in the 1st Model is the Support Vector Machine (SVM) classifier, which is implemented using the SVC class from the sklearn.svm module. The SVC class uses the kernel trick to transform the input data into a higher-dimensional space, where it is easier to separate the classes using a linear classifier. The kernel parameter of the SVC class is set to 'linear', which means that the linear kernel is used. The input features are converted into numerical features using a vectorizer, either CountVectorizer or TfidfVectorizer, which are both implemented in the sklearn.feature_extraction.text module. These vectorizers convert the input text into a sparse matrix of numerical features, where each row represents a document (in this case, a tweet) and each column represents a unique word in the corpus. The entries in the matrix represent the frequency of each word in each document (for CountVectorizer) or the tf-idf score of each word in each document (for TfidfVectorizer). The choice of vectorizer and the range of n-grams (in this case, unigrams) is controlled by the vectorizer_type and ngram_range parameters of the train_method1 function.

In the second code, the classifier used is a Convolutional Neural Network (CNN) model. The train_method2() function defines the CNN model using Keras, which includes convolutional layers,

pooling layers, and dense layers and then compiles it with a binary cross-entropy loss function, the Adam optimizer, and three evaluation metrics: accuracy, precision, and recall. The test_method2() function loads the saved CNN model and tokenizer, uses the tokenizer to convert the test data to sequences, pads the sequences to have equal length, and converts the text to FastText embeddings. Then, it predicts the labels for the test set using the loaded CNN model and converts the predicted probabilities to labels. Finally, it evaluates the performance of the model on the test set and saves the predictions to an output file.

Dataset	Total	% OFF	% NOT
Train	12313	33.23	66.77
Valid	860	37.91	72.09
Test	927	33.22	66.78

Table 1: Dataset Details

Model	F1 Score
Model 1	0.7304
Model 2	0.7229
SoA {HASOC 2021}	0.89

Table 2: Model Performance

4 Data Size Effect (Task 3)

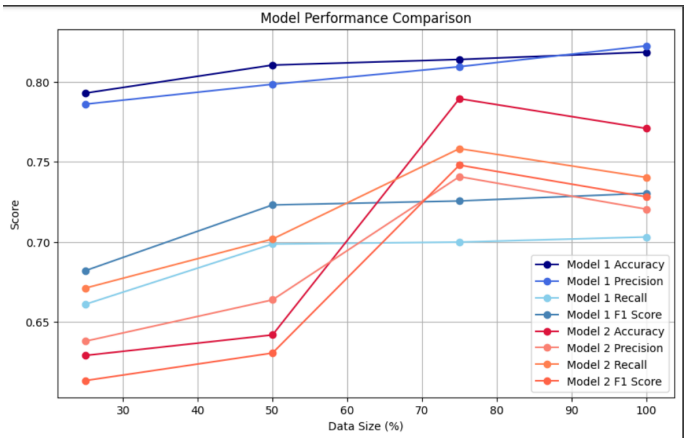

Figure 2: Comparision of Models based on Different data sizes. Accuracy, Precision, Recall and F1 Score.

Model 1, which was a Support Vector Machine (SVM), exhibited an overall trend of improvement in accuracy, precision, recall, and F1 score as the size of the data increased. Specifically, the accuracy increased from 0.7930 to 0.8186, precision increased from 0.7861 to 0.8225, recall increased from 0.6611 to 0.7031, and F1 score increased from 0.6821 to 0.7304. This suggests that Model 1 was able to learn and generalize better with larger data sizes, which may be attributed to SVMs being well-suited for high-dimensional data. The increase in accuracy indicates that Model 1 was able to make more accurate predictions with the larger data sizes, while the increase in precision suggests that Model 1 was able to reduce the number of false positives. The increase in recall means that Model 1 was able to correctly identify more relevant instances in the dataset, while the increase in F1 score indicates that Model 1 was able to achieve a better balance between precision and recall. On the other hand, Model 2, which was a Convolutional Neural Network (CNN), showed a mixed trend in performance metrics as the size of the data increased. Specifically, the accuracy and precision of Model 2 initially increased as the data size increased from 25 to 50, but then decreased as the data size further increased to 75 and 100. In contrast, recall and F1 scores showed a general increasing trend as data size increased. The initial increase in accuracy and precision suggests that Model 2 was able to make more accurate predictions and reduce false positives when trained on a moderate amount of data, but this trend did not continue as the amount of data increased further. The decrease in accuracy and precision with larger data sizes may be due to overfitting, where the model starts to learn the noise in the data and generalize poorly to new instances. In contrast, the increasing trend in recall and F1 score suggests that Model 2 was able to correctly identify more relevant instances in the dataset as the amount of data increased, indicating that Model 2 was able to learn more complex features and patterns with larger data sizes. These findings highlight the importance of understanding the strengths and weaknesses of different machine learning algorithms and their ability to handle varying amounts of data. It also underscores the need to carefully evaluate model performance with respect to different performance metrics, as the optimal trade-off between metrics may differ depending on the problem at hand. Over-

all, these insights can inform the selection and tuning of machine-learning models for future applications in this domain.

Data %	Total	% OFF	% NOT
25%	3078	33.23	66.97
50%	6156	29.02	71.98
75%	9234	30.09	69.91
100%	12312	28.55	71.45

Table 3: Train Dataset Statistics of Different Size

You need to use Table 5 6 and 4 compare the model's output and provide exciting insights..

5 Summary (Task 4)

The lessons learned from this exploration of SVM and CNN text classification models for offensive language detection are valuable for both understanding the nuances of machine learning algorithms and guiding future research in this area. Some of the key lessons learned include:

Model selection: Different models have unique strengths and weaknesses, making it crucial to choose the appropriate algorithm for the task at hand. In this case, the SVM model performed consistently well with increasing data size, while the CNN model showed a more complex performance pattern.

Data size and overfitting: The relationship between data size and model performance can be intricate. While more data generally helps improve model performance, it may also lead to overfitting in some cases, as observed with the CNN model.

Evaluation metrics: Assessing model performance using diverse metrics is essential, as the ideal balance between accuracy, precision, recall, and F1 score may vary depending on the problem. This investigation showed the importance of considering all of these metrics to understand the true performance of a model.

Data preprocessing: Proper data preprocessing, such as lowercasing, removing stopwords and punctuations, and tokenization, is crucial for preparing the data for classification tasks. This step can significantly impact model performance.

Feature representation: Choosing the right feature representation, like TF-IDF for SVM and FastText embeddings for CNN, can enhance the model's ability to capture word relationships and improve its overall performance.

Experimentation: Testing models with different data sizes and configurations can reveal valuable insights into their performance and potential limitations. This iterative approach can inform model selection and fine-tuning in future applications.

6 Conlusion

In conclusion, I explored two different text classification models, SVM and CNN, for detecting offensive language. My journey led to interesting findings about how these models perform as the size of the training data increased.

The SVM model consistently improved in all metrics—accuracy, precision, recall, and F1 score—as the data size grew, showing its strength when working with high-dimensional data. On the other hand, the CNN model's performance showed a more complex pattern. It started with an increase in accuracy and precision using a moderate amount of data but didn't continue as the data size increased. This surprising result could be due to overfitting, where the model learns noise instead of useful patterns.

However, the CNN model showed an increasing trend in recall and F1 score as data size increased, indicating its ability to capture complex features and patterns with more data.

These findings not only sparked my curiosity but also highlighted the importance of understanding the unique strengths and weaknesses of different machine learning algorithms. They also emphasized the need to evaluate model performance using various metrics, as the ideal balance between them may differ based on the problem. In the end, the insights gained from this investigation can help guide the selection and fine-tuning of machine learning models for future applications in this exciting field.

References

• Wang, Q., Mu, H. (2020). Privacy-Preserving and Lightweight Selective Aggregation with Fault-Tolerance for Edge Computing-Enhanced IoT. Sensors, 20(18), 5088. • Fortuna, P., Nunes, S. (2018). A Survey on Automatic Detection of Hate Speech in Text. ACM Computing Surveys (CSUR), 51(4), 1-30. • Zampieri, M., Malmasi, S., Nakov, P., Rosenthal, S., Farra, N., Kumar, R., ... Mubarak, H. (2019). Predicting the type and target of offen-

Example %	GT	M1(100%)	M2(100%)
Example 1	NOT	NOT	NOT
Example 2	OFF	OFF	NOT
Example 3	NOT	NOT	NOT
Example 4	NOT	NOT	NOT
Example 5	OFF	OFF	NOT

Table 4: Comparing two Model's using 100% data: Examples and model output using Model 1 & 2..

Example %	GT	M1(25%)	M1(50%)	M1(75%)	M1(100%)
Example 1	NOT	NOT	OFF	NOT	NOT
Example 2	OFF	NOT	OFF	OFF	OFF
Example 3	NOT	NOT	NOT	NOT	NOT
Example 4	NOT	OFF	OFF	NOT	NOT
Example 5	OFF	OFF	OFF	OFF	OFF

Table 5: Comparing Model Size: Examples and model output using Model 1 with different Data Size

Example %	GT	M2(25%)	M2(50%)	M2(75%)	M2(100%)
Example 1	NOT	NOT	NOT	NOT	NOT
Example 2	OFF	NOT	NOT	OFF	OFF
Example 3	NOT	NOT	NOT	NOT	NOT
Example 4	NOT	OFF	OFF	NOT	NOT
Example 5	OFF	NOT	NOT	NOT	NOT

Table 6: Comparing Model Size: Sample Examples and model output using Model 2 with different Data Size

sive posts in social media. In Proceedings of the 2019 workshop on semantic evaluation (pp. 99-107). • Impact of SMOTE on Imbalanced Text Features for Toxic Comments Classification Using RVVC Model - IEEE. • Research on Web text classification algorithm based on improved CNN and SVM - IEEE. • HASOC 2021: Hate Speech and Offensive Content Identification in Indo-European Languages" by Mandl et al. (2021)