SMM4H Task 2: The Automatic Classification of Tweets Mentioning Adverse Events

Team: Emmi Carr, Angel Ka Yan Chu, Siddharth Madapoosi, and Tasha Torchon

Abstract

As part of the Social Media Mining for Health Applications (SMM4H) Shared Task 2020, our team participated in task 2, the automatic classification of tweets that mention adverse events associated with medication use. Our general approach was to implement a variety of preprocessing steps, feature extraction methods and machine learning as well as deep learning techniques such as Support Vector Machine (SVM), Random Forest and Neural Networks in order to be able to compare various approaches to identify those most suitable for this task. Based on our results, CNN and linear SVM were the best-performing models with F1-scores of 0.5300 and 0.4976, respectively. Moving forward, future steps include further tuning the hyperparameters and experimenting with different combinations of layers for the CNN model and exploring methods for handling the imbalanced dataset including oversampling, undersampling and generating synthetic data.

Introduction

As part of the Social Media Mining for Health Applications (SMM4H) Shared Task 2020, our team participated in task 2, the automatic classification of tweets that mention adverse events associated with medication use. As such, this was a binary classification task with the goal to classify tweets as 1 if an adverse event of a medication is mentioned or 0 in the case of no adverse event mentioned. In order to be successful in doing so, one of the main challenges is to correctly distinguish between reasons why someone is taking the medication versus adverse events as a result of the medication use. In total, the training data was composed of 25,672 tweets (2,374 “positive” tweets; 23,298 “negative” tweets). This was then further split into a train set and a validation set with a 80-20% split. As the test data has not yet been released for this task, we used the train set for training purposes and the validation set to test our models. Since the data is unbalanced with only about 9% of tweets mentioning adverse events, we used the F1-score as the metric of primary importance during evaluation. This aligns with SMM4H’s evaluation metric of choice.

Data Cleaning and Preprocessing

For our models, we employed different methods to make the text more consistent and eliminate extraneous elements. Beyond removing URLs and punctuation, we normalized user mentions by replacing them with “username” and applied the emoji library in Python to substitute the images with synonymous descriptive text. These changes did not seem to improve predictive

ability significantly, however. There was, at most, a 2% difference between the F1 scores of models trained with the original and preprocessed text.

We also experimented with replacing tweet references to brand-name drugs found in RxTerms with the word “drug.” RxTerms is a drug interface terminology derived from RxNorm that includes 99% of the generic and brand names of the most commonly prescribed drugs in the United States, as well as their route of administration, strength, and dose form (Fung, McDonald, & Bray, 2008; RxTerms, 2020). Approximately 6.5% of all words in all tweets in our training data were brand name drugs and these were replaced by the word “drug.” However, the validation F1 scores of models built on drug-normalized data were not significantly different from non-normalized data and were occasionally lower. This observation may be due to an underlying lack of normalization of the actual references to brand name drugs in tweets, such as “pepto” for “pepto bismol” as well as references to generic, rather than brand-name drugs. Therefore, the final models shown in **Table 1** did not incorporate brand name-replacement with “drug” during data preprocessing.

Additionally, we experimented with opinion mining using sentiment analysis to investigate the correlation between sentiment strength and class. There are various methods to perform sentiment analysis, such as 1) machine learning based approach and 2) rule-based, lexicon-based approach using predefined lexical dictionaries. Generally speaking, as the latter approach is faster in execution, requires less computational power, and no prior training is needed, we proceeded with a lexicon-based approach using a manually constructed lexicon named “bing” in the tidytext R package (Hu & Liu, 2004**;** Liske, 2018). This lexicon dataset includes 6786 positive and negative connotations, categorizing predefined words in a binary fashion. The *bing* lexicon was joined with our tweets to assess the positive and negative sentiment of each word. The number of occurrences of words being labelled as positive or negative was recorded as a positive score and a negative score. Furthermore, in order to calculate a net sentiment score for each tweet, we subtracted the negative score from the positive score.

In our adverse drug effects classification task, we applied sentiment analysis to extract the sentiment strength represented by a quantifiable, net sentiment score. With the scores, we enriched the feature space. Along with the preprocessed tweets, the sentiment analysis features were fitted into SVM and Random Forest models. Setting the term frequency at 2 helped reduce the number of features from 27,457 to 15,787, successfully retaining 57.8% of the original features. F1 score, precision, recall, and accuracy metrics of the SVM model with term frequency of 2 and sentiment analysis are shown in **Table 1** used to compare with the SVM models without sentiment analysis. It shows that adding sentiment analysis features did not improve the F1 score, when compared to the SVM model with full text. The added features did not boost the overall metrics; one reason might be due to the nature of the *bing* lexicon, as it was first created in 2004 from customers’ online reviews. Another possible lexicon that can be integrated is the NRC lexicon (Moohammad & Turney, 2013), since the authors built the classifiers to detect the sentiment for Twitter.

With respect to two types of models that we built (Linear SVM and LogitBoost), we also experimented with reducing the number of features by term frequency. In order to reduce the number of features, tweets were first tokenized to individual words while preserving URLs using the tidytext R package and the number of tweets in which each word was found was counted (Silge & Robinson, 2016). Linear SVM and LogitBoost models were generated on the resulting data. Two thresholds for term frequency, 3 and 5, were tested for their impact on F1 score out-of-sample compared to a linear SVM built on all words found in all tweets. Thresholding term frequency at 3 produced a training dataset with 7,898 individual features, while thresholding term frequency at 5 produced a training dataset with 4,752 features. Models in **Table 1** built using this feature selection scheme are labeled with (TF = *n*) under the model description column with *n* representing the frequency threshold.

Convolutional Neural Network (CNN) Model Description

One of the models built was a Convolutional Neural Network (CNN). The structure included an embedding layer, a convolutional layer, a global max pooling layer to help reduce the dimensionality of the input representation, two dropout layers and two dense layers. The full structure is shown in the appendix. We experimented with two word embedding approaches (Brownlee, 2017; Godin; 2019; Janakiev, n.d.). The first were manually created embeddings using the data from SMM4H task 1 and the task 2 training data using Word2Vec. The second were Word2Vec Twitter Embeddings, which proved to be more robust and result in better performance compared to the manually created embeddings. The results of these approaches can be seen in **Table 1** below. The optimizer used for our model was Adam and the loss function was f1_loss, a custom loss function found online (Haltuf, 2018). As loss gives us an indication of how many errors our model is making, by using f1_loss, we were able to optimize the model to achieve a greater f1 score rather than purely the highest accuracy value. When fitting the model, callbacks were used to stop training once the lowest 'val_loss' value was achieved (Brownlee, 2018). When doing so, a patience factor of 10 was applied so that the model trained for 10 additional iterations and then stopped if a lower loss value was not achieved during that time. By using such an approach, it accounts for variation in how the model trains each time it is run and avoids the dilemma of having to set a strict value for the number of epochs, or iterations. With this model, we also experimented with adding a Bidirectional LSTM Layer as LSTM has shown promise for achieving good performance with text classification tasks. The results comparing the model with and without the LSTM layer are shown in **Table 1** below.

Description & Comparison of Runs

The 11 different models that we built as well as their F1 scores, precisions, recalls, and accuracies on the validation set are shown in **Table 1** below, ranked by F1 score in descending order.

Run	Model Description	F1-Score	Precision	Recall	Accuracy
1	CNN, Twitter Word2Vec embeddings	**0.5300**	0.5283	0.5316	0.9129
2	CNN+Bidirectional LSTM layer, Twitter Word2Vec embeddings	0.5175	0.4684	**0.5781**	0.9005
3	Linear SVM with squared hinge loss	0.4976	0.5819	0.4346	0.9190
4	CNN, manually created Word2Vec embeddings	0.4932	0.4568	0.5359	0.8983
5	SVM with linear kernel (Full Text)	0.4584	0.7441	0.3312	**0.9277**
6	SVM with linear kernel (TF = 3)	0.4147	0.4116	0.4178	0.8909
7	SVM with linear kernel (TF = 2 and SA)	0.4103	0.415	0.4059	0.8839
8	SVM with linear kernel (TF = 5)	0.4043	0.5	0.3966	0.8918
9	Decision Trees	0.3509	0.3379	0.3650	0.8753
10	LogitBoost (TF = 3)	0.1198	0.4286	0.0696	0.9053
11	Random Forest	0.1107	**0.8750**	0.0591	0.9123

Table 1: Description of Runs. Models are sorted in descending order by F1 score, model performance metric of choice. The highest values of F1, precision, recall, and accuracy are bolded. TF = term frequency; SA = sentiment analysis

Based on F1 scores, the best-performing model across all of our runs was the CNN with Twitter Word2Vec embeddings with an F1 score of 0.5300, while the worst-performing model across all of our runs was the Random Forest model with an F1 score of 0.1107. The CNN model with Twitter Word2Vec embeddings and a Bidirectional LSTM layer had the highest recall at 0.5781, while the linear SVM model without term frequency thresholding had the highest accuracy at 0.9277 and the Random Forest model had the highest precision at 0.8750. The table highlights the importance of using F1-Score as the model performance metric in this scenario given the unbalanced nature of the data. While models with low F1 scores such as the LogitBoost and Random Forest models had accuracies exceeding 90%, their low recall measures indicate that these models were classifying many tweets incorrectly as not reporting adverse events (class 0) when in fact they were. Since the data was heavily unbalanced in favor of class 0, the accuracies of these models were inflated even though they were not predicting adverse events as effectively as the CNNs and linear SVMs which had roughly the same accuracies. The decreasing trend in recall down the table generally aligns with the trend in F1 score and with our expectation that models with higher F1 scores will be more accurate at determining both when a tweet is indeed reporting an adverse event and when it is not.

As seen above, the CNN with Twitter Word2Vec embeddings outperforms the model with manually created Word2Vec embeddings. This can likely be accounted for by the fact that the Twitter Word2Vec embeddings are trained on a much larger set of tweets and are therefore, more robust. Also, we see that the performance of the CNN decreases slightly when a Bidirectional LSTM layer is added. As there is slight variation in the performance metrics across runs due to randomness in the models and how they learn, we decided to rerun our best performing CNN model (the CNN with Twitter Word2Vec embeddings represented by run 1 above) five times and average the results in order to account for this variation. The resulting F1-Scores were [0.5215562565720294, 0.5307443365695793, 0.5216554379210779, 0.5319148936170213, 0.5348595213319459]. Therefore, we get an average F1-Score of **0.5281** across the five runs, which is very close to the result we see in **Table 1**. This indicates that the model is performing pretty consistently across runs.

Among the SVM classification algorithms employed, linear SVC had the highest F1-score. In the Python sci-kit learn library, “LinearSVC” is distinct from “SVC” with a linear kernel because the former allows for changes to linear parameters (StackOverflow, 2017). For instance, LinearSVC uses intercept scaling, which means that the decision boundary is defined in terms of an intercept (StackOverflow, 2017). This is uncharacteristic of the traditional SVM algorithm, which separates data classes based on the maximum margin hyperplane. We left the intercept scaling value at 1. The loss function can also be changed in linearSVC. We set loss to squared hinge because it increased recall by about 10% and had a positive effect to F1 (though at a 16% cost to precision). The sci-kit learn SVM algorithm uses a hinge-loss function. However, squared hinge assigns greater cost to errors and is differentiable all along the curve (Zajano, 2014; Hui, 2017; StackExchange, 2018; Chowdhury, 2019). Tweaking the other parameters of the LinearSVC model did not seem to improve the model much. Therefore, the default values were kept.

In an attempt to improve the model, we also reduced the number of features through term frequency thresholds. This method appeared to decrease the out-of-sample F1 score of the SVM model built on hinge loss by 0.04-0.05, depending on the term frequency threshold. More stringent thresholds generated greater reductions in F1. These findings indicate that infrequent terms in the data may still play an important role in the model’s predictive out-of-sample ability.

Limitations & Future Steps

Overall, this report compares the performance of different approaches for the automatic classification of tweets that mention adverse events associated with medication use. Although we achieved reasonable performance, there are some additional steps we could try implementing. In regard to the CNN, future steps could involve further tuning the hyperparameters of the model and experimenting with additional layers. Also, as we saw that CNN and SVM both performed reasonably well for this task, a pipeline could be developed that integrates these two approaches together. Another option would be to integrate syntactical

features from MetaMap, such as combinations of drug name references and symptoms, to see how these affect the models. Lastly, we could explore methods for handling the imbalanced dataset including oversampling, undersampling and generating synthetic data.

References

Barbera, P. SVM, Random Forests, and beyond. (2018, August 9). Retrieved from http://pablobarbera.com/ECPR-SC105/code/12-advanced-sml.html

Brownlee, J. (2017, October 4). How to Use Word Embedding Layers for Deep Learning with Keras. Retrieved from https://machinelearningmastery.com/use-word-embedding-layers-deep-learning-keras/

Brownlee, J. (2018, December 10). Use Early Stopping to Halt the Training of Neural Networks At the Right Time. Retrieved from https://machinelearningmastery.com/how-to-stop-training-deep-neural-networks-at-the-right-time-using-early-stopping/

Chowdhury, K. (2019, December 20). Understanding loss functions : Hinge loss. Retrieved April 17, 2020, from https://medium.com/analytics-vidhya/understanding-loss-functions-hinge-loss-a0ff112b40a1

different results with MEKA vs Scikit-learn! (2018, March 4). Retrieved April 19, 2020, from https://datascience.stackexchange.com/questions/28596/different-results-with-meka-vs-scikit-learn

Fung, K. W., McDonald, C., & Bray, B. E. (2008). RxTerms - a drug interface terminology derived from RxNorm. AMIA ... Annual Symposium proceedings. AMIA Symposium, 2008, 227–231. Retrieved from https://www.ncbi.nlm.nih.gov/pmc/articles/PMC2655997/pdf/amia-0227-s2008.pdf

Godin, F. (2019, August 17). Improving and Interpreting Neural Networks for Word-Level Prediction Tasks in Natural Language Processing. Retrieved from https://github.com/FredericGodin/TwitterEmbeddings

Haltuf, M. (2018, October 19). Best loss function for F1-score metric. Retrieved from https://www.kaggle.com/rejpalcz/best-loss-function-for-f1-score-metric

Hu, M. & Liu, B. (2004, August 22) Mining and Summarizing Customer Reviews. Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD-2004), 2004, 168-177. Retrieved from https://dl.acm.org/doi/10.1145/1014052.1014073

Hui, J. (2017, January 15). Retrieved from https://jhui.github.io/2017/01/15/Machine-learning-regression-and-logistic-regression/

Janakiev, N. (n.d.). Practical Text Classification With Python and Keras. Retrieved from https://realpython.com/python-keras-text-classification/

Kim, R. (2018, February 23). Another Twitter sentiment analysis with Python - Part 11 (CNN Word2Vec). Retrieved from https://towardsdatascience.com/another-twitter-sentiment-analysis-with-python-part-11-cnn-word2vec-41f5e28eda74

Liske, D. (2018, February 2) Lyric Analysis with NLP & Machine Learning with R. Retrieved from https://www.datacamp.com/community/tutorials/R-nlp-machine-learning

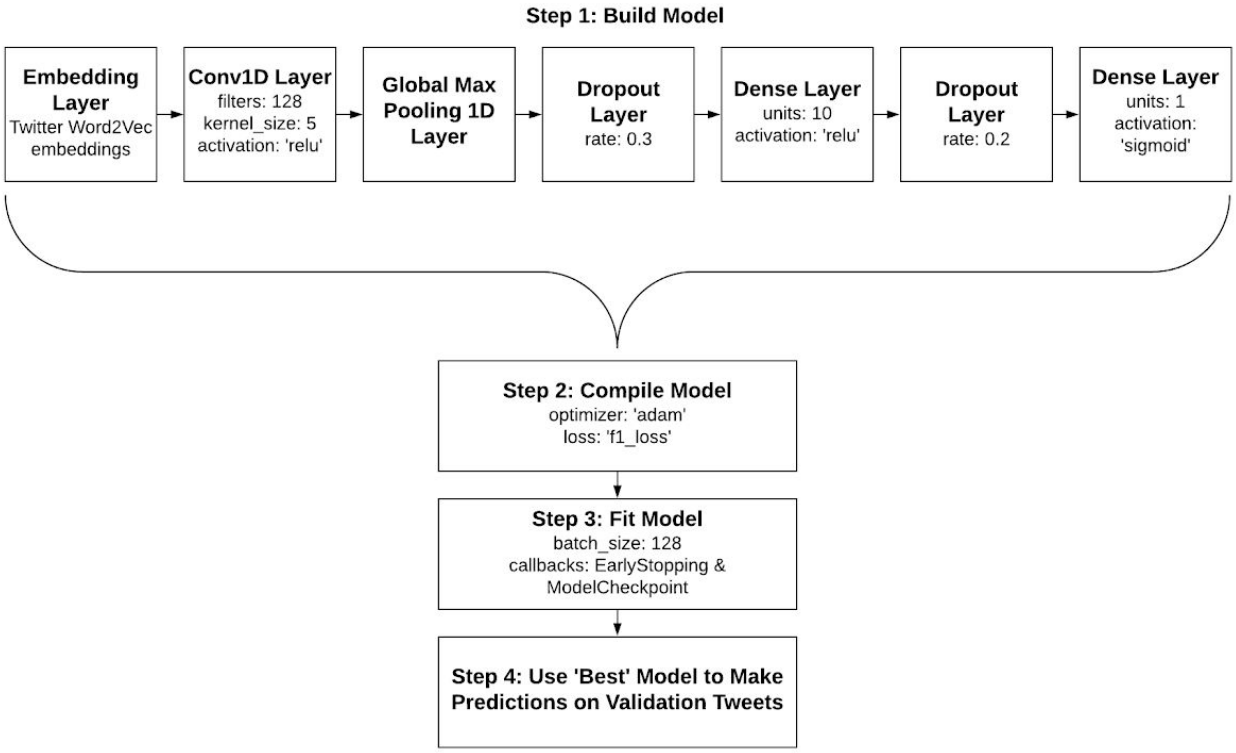
Liske, D. (2018, March 29) Tidy Sentiment Analysis in R. Retrieved from https://www.datacamp.com/community/tutorials/sentiment-analysis-R

Moh, M., Moh, T. S., Peng Y., & Wu L. (2017). On adverse drug event extractions using twitter sentiment analysis. Netw Model Anal Health Inform Biooinforma, 2017, 6-18. Retrieved from https://link.springer.com/article/10.1007%2Fs13721-017-0159-4

Mohammad, S., Kiritchenko, S., & Zhu, X. (2013, June) NRC-Canada: Building the State-of-the-Art in Sentiment Analysis of Tweets. Retrieved from https://www.aclweb.org/anthology/S13-2053/

RxTerms. (2020, February 6). Retrieved from https://lhncbc.nlm.nih.gov/project/rxterms
Silge, J. & Robinson, D. (2016). Tidytext: Text Mining and Analysis Using Tidy Data. Retrieved from https://cran.r-project.org/web/packages/tidytext/index.html
Silge, J., & Robinson, D. (2020, March 7). Sentiment analysis with tidy data. Retrieved from https://www.tidytextmining.com/sentiment.html
Under what parameters are SVC and LinearSVC in scikit-learn equivalent? (2017, May 23). Retrieved April 17, 2020, from https://stackoverflow.com/questions/33843981/under-what-parameters-are-svc-and-linearsvc-in-scikit-learn-equivalent/33844092
Zajano. Squared versus Hinge Losses: SVC versus RLS. (2014, November 29). Retrieved from https://cvstuff.wordpress.com/2014/11/29/latex-l_1-versus-latex-l_2-loss-a-svm-example/

Appendix



Individual Contributions Table

Team Member	Contributions
Emmi Carr	My main contribution to our team's efforts was the development of the CNN. I researched and experimented with different combinations of layers and hyperparameters in order to tune the model and improve performance. Along this, I experimented with implementing different word embeddings into the model.
Angel Chu	My main contribution to our project was building linear SVM and Random Forest models using R. Prior to the model training, I researched and reviewed previous papers about what people steps previously took in tweet classification. Besides data preprocessing, I performed a lexicon-based sentiment analysis and implemented the results to the SVM model. Besides, I also attempted developing a Naive Bayes model with R, but did not get it to run.

Siddharth Madapoosi	My main contribution to our team's efforts was developing the linear SVMs and LogitBoost models in R with various term frequency thresholds. I also experimented with using RxTerms to replace brand-name drug references in tweets with the word "drug" and measured the impact this had on F1 scores of term frequency-based linear SVMs and LogitBoost models.
Tasha Torchon	My main contribution was data cleaning and running models (linearSVC, SVC, Decision Trees, Random Forest , and others not included in the paper or presentation) using Python. I was able to find a library to replace emojis with equivalent word descriptions. I also found the differences in parameters between linearSVC and SVC with a linear kernel. I attempted to download and use MetaMap for analysis. However, I was unsure of how to automate it for our purposes.