SMM4H 2020, Task 3

AUTOMATIC EXTRACTION AND NORMALIZATION OF ADVERSE DRUG EFFECTS IN TWEETS(ENGLISH)

TECHNICAL REPORT

BY: Charlene Hsiao, Jonathan Martindale, Tabassum Nisha, Jingcheng Xiao

CONTENTS

INTRODUCTION

With the fast-growing technological advancements and modernization of lifestyle, social media and its content have become indispensable to millions of users around the globe. Contents on social media particularly of community question/answering/commenting are trending portals of users in search of help and support for any kind of situation.[1] Launched in 2006, the popularity of Twitter has increased rapidly among the population making it the second most important social media after Facebook.[2] Given its key features like short-message type blogs/posts of about 140 characters known as tweets that led to its large scale usage for public discussions during acute events (Burgess 2010).[3]

Twitter serves as a micro-blogging platform that accounts for about millions of users and on average, about 6,000 tweets are tweeted on Twitter per second, corresponding to about 350,000 tweets tweeted per minute which makes it to about 500 million tweets per day and that's around 200 billion tweets per year.[4] Given the ubiquitous nature of Twitter, a large bulk of the population shares and discusses health-related issues. During this time of COVID-19 pandemic, there were over 50 million tweets collected from the inception until March 16, 2020, constituting

roughly 450 GB of raw data accounting for the exchange of information on health issues.[5] People often share a wide variety of information such as pharmaceutical prescriptions, drug allergies, and other health-related issues on Twitter and other disease-specific communities, blogs, microblogs, public news websites, or drug discussion forums.[6]

Stated by Rachel Ginn et al, "Natural language processing from social media text is very challenging for any purpose, given that the text is highly unstructured and informal, and may contain a large number of misspelled words."[6] The Social Media Mining for Health Applications(SMM4H), Task 3, 2020 is an end-to-end task that involves extracting the span of a text containing an adverse effect of medication from tweets that are reported as ADRs, and then mapping the extracted ADRs to a standard concept ID in the MedDRA vocabulary (a clinically validated international medical terminology dictionary).

METHODS

Data Cleaning and pre-processing:

Cleaning: In the raw twitter context, there are many informal expressions, meaningless symbols (like "#" or emoji) and special characters. In order to decrease the noise of the training data, cleaning steps were performed. We first removed all the "@" mentions, URLs and non-standard characters because we think they are not relevant to our task. Then we expanded all the contractions to formalize the text. However, this step may cause some potential problems because, in the given ADR extractions, very few of them had contractions like "I'm". Finally, we changed all the twitter text as well as extracted ADR to lowercase.

Tokenization and lemmatization: The cleaned twitters were tokenized by the tokenizer from NLTK packages. There are several tokenizers available (like word_tokenizer, twitter_tokenizer…), we compared the tokenization performances of the tokenizers mentioned by last year's SMM4H papers and no significant differences were observed. So, we used twitter_tokenizer as it is the most commonly used one for twitter. We also did lemmatization at first, and then realized the lemmatization will cause some problems for twitter labeling because it made the twitter texts unable to exactly match the given extractions. To avoid adding more complexity to our data, we finally chose not to use the lemmatization technique even though it is recommended by some papers [7].

Sequence labeling of the tokenized twitters: In the training data set, no labels were given to the twitters with ADR extractions. So we labeled the tokenized twitter with labels "B"," I", "O". "B" stands for the beginning of the ADR span, "I" in the middle of the ADR span, and "O" is the rest of the twitter words. The provided extraction ADRs were matched in cleaned twitters by the regular expression. The final labeled tokenized twitters were formatted according to the requirement of the NER model.

NER: Named Entity Recognition:

Using the cleaned files, we needed to develop a model that would allow us to predict the tags "B" . "I" and "O" in our validation file. In order to do this we used a deep learning platform Pytorch that is effective for natural language processing as well as the natural language processing library flair.

On Pytorch, we used a bidirectional LSTM model and we used GloVE and Word2Vec word

embeddings, then tried multiple different models to identify best results. Some of the models we tried were Ada, RMSProp, Adam, and stochastic gradient descent. What we found was that the best results were from Adam and SGD with a high learning rate. Hyperparameters that we tried tuning were hidden dimensions, the number of word embeddings we used, weight decay, betas, momentum, batch size number, epoch number, learning rate, use_crf, and a few others. The parameters which made the most difference were the learning rate, the batch size. The number of hidden embeddings seems to be correlated with better scores until it reached a certain point We found that the number of epochs largely had no effect past 10 on whether we were able to get better results. In the end, we found that Adam with a hidden dimension of 120, and a batch size of 42 provided the best f scores. The reason Adam provided better results than other models is probably because it offers advantages from both RMSProp and Adagrad. Adaptive moment estimation allows for adaptive learning rates per parameter in a computationally effective way. A couple of models we would also consider trying would be AMSgrad, AdaMax, SGD with Nesterov momentum, and Nadam.

We recognize that better performance could be achieved in a number of other ways. Some additional methods that would be worthwhile to explore include other word embeddings. Additional embeddings worth exploring are stacked and flair embeddings that might make use of the powerful flair library. In addition, our adverse events in the training file were not fully accurately named, and that could have impacted our results as well. Incorporating drop out layers and playing with these layers could also help improve our results.

Normalization:

Mapping the predicted ADRs to get their spans in original twitters: After the labels were predicted to each tokenized word in twitter. We used a dictionary to build the word-label match and extracted all the words with label "B" or "I" to get the predicted ADR words in each twitter. Then the predicted ADR extractions were recognized by regular expression and matched in original twitter. The span of the matched words was generated by the finditer() function in re package.

Matching Identified Features to MedDRA:

After the predicted ADRs were extracted and re-mapped onto each tweet, we needed to match the extraction text to a MedDRA term code. Because MedDRA does not provide, at this time, an API to use the corpus, we chose to use an indirect route to match the extracted text to the correct MedDRA code. Using a dictionary provided by Dr. Vydiswaran that matches common ADRs in MedDRA to UMLS/Metamap Concept Unique Identifiers (CUIs) and a Python package, Quick UMLS[8], that performs fuzzy string matching to find CUIs, we were able to match the extracted text to MedDRA terms and codes. This process involved several steps: First, we needed to run Quick UMLS on the extracted text to find potential matches. This process often returned multiple possible terms and corresponding CUIs. To find the closest match, we specified that Quick UMLS use a combination of length and cosine similarity of the extracted text and any potential matches. This still returned multiple possible matches, so our next step was to see if any of the returned CUIs were included in our UMLS-MedDRA dictionary. Because CUIs were

sorted by similarity and length, we specified that the returned CUI would be the first match found out of all possible CUIs.

At this point, each predicted extraction now has a corresponding best-match CUI, if any were found. We then merged the UMLS-Metamap dictionary onto our data, using CUIs. Because a single CUI can map onto multiple MedDRA codes, however, this left us with many additional rows. To fix this, we used a fuzzy string matcher, FuzzyWuzzy, to rank the similarity of the extracted text to the preferred MedDRA term in the dictionary. The pair with the highest similarity was chosen. To ensure that all rows were included in the final dataset, we re-merged the resulting dataset onto the original validation file from the previous step.

RESULTS

Results for the Named Entity Recognition models:

Below, we have included some of the intermediate results from the NER portion of our code, which describes a number of the different runs and hyperparameters tested in order to optimize the model. All scores shown are the results from the last epoch generated for the validation data.

Runs	Model and Parameters	Precision	Recall	Accuracy	F1 Score
Run1	**SGD optimizer** Params = batch_size = 32, use_crf = True, embedding_dim = 100 ,relearned_dim = embedding_dim, hidden_dim=50, lr = 0.01, training_epoch =5	B-0.38 I-0.34 O-0.96	**B-0.29** **I-0.23** **O-0.97**	0.93	**B-0.33** **I-0.27** **O-0.97**
Run2	**SGD optimizer** Params = batch_size = 32, use_crf = True, embedding_dim = 100 ,relearned_dim = embedding_dim, hidden_dim=50, lr = 0.1, training_epoch =5	B-0.39 I-0.35 O-0.96	B-0.22 I-0.15 O-0.98	**0.97**	B-0.28 I-0.29 O-0.97
Run3	**SGD optimizer** Params = batch_size = 32, use_crf = True, embedding_dim = 100 ,relearned_dim = embedding_dim, hidden_dim=50, lr = 0.15, training_epoch =5	B-0.34 I-0.21 O-0.96	B-0.27 I-0.35 O-0.95	0.91	B-0.30 I-0.27 O-0.95
Run4	**SGD optimizer** Params = batch_size = 32, use_crf = True, embedding_dim = 100 ,relearned_dim = embedding_dim, hidden_dim=50, lr = 0.12, training_epoch =4	**B-0.42** **I-0.43** **O-0.96**	B-0.28 I-0.16 O-0.98	0.94	**B-0.34** **I-0.23** **O-0.97**
Run 5	**Adam Optimizer** Params = batch_size = 52, use_crf = True, embedding_dim = 100 ,relearned_dim = embedding_dim, hidden_dim=120, lr = 0.001, training_epoch =4	B-0.35 I-0.33 O-0.96	B-0.27 I-0.28 O-0.97	0.93	B-0.30 I-0.20 O-0.96

After performing a number of runs giving the f1-score of each tag below 0.1 and sometimes 0, the above runs showed promise with the SGD Optimizer and Adam Optimizer. The best of the f1-scores were achieved with the SGD Optimizer and Adam Optimizer. The below are results from Run 5, as it was the best model at the time of data analysis.

Overall Performance Of The Model:

We report here the accuracy of our model at three different points, described below. For each one, we report the overall accuracy, precision, recall, and F1-Score for the positive class as well as the weighted average F1-score.

1. If the NER model correctly predicted the presence of an ADR in a given tweet.
2. For all instances where the model predicted the presence of an ADR, how accurate the model was in predicting the exact ADR text.
3. In the subset of all tweets where the predicted extraction text matched the true extraction text, how accurate the model was in converting the extraction text to the correct MedDRA Code.

Measure	**Accuracy**	**Precision**	**Recall**	**F1-Score (Positive Class)**	**F1-Score (Weighted Ave)**
ADR Present	0.59	0.87	0.45	0.58	0.59
Correct Extraction	0.36	1.00	0.26	0.42	0.40
Correct MedDRA Code	0.56	1.00	0.30	0.46	0.52

Precision at each step was quite high, ranging from 0.87 to 1.00. In each case, however, the recall was noticeably lower, resulting in F1-Scores for the positive class to range from 0.42 to 0.58. In the next

section, we discuss some of the potential reasons for this lower recall along with other challenges and limitations our team faced with the project.

LIMITATIONS

In terms of limitations, our results yielded a relatively low f1 score, which we believe can be improved by changes to different portions of the task. Accuracy in the end result was affected by minor issues along different portions of the task. Addressing these challenges would allow us to have significant improvements to the f1 score. We list several of these below:

1. When performing the twitter text cleaning, we didn't use the lemmatization technique because it would complicate the tracing back of adverse drug reaction span to the original twitter.
2. Since we each ran different parts on separate computers, when the predicted results (.csv file) were imported, all the list of the tokenized twitter and predicted labels were recognized by python as strings, which means the "[", "," and citation marks for list were all regarded as part of the string. We didn't find an easy approach to solve this problem, so we cleaned the string and changed it back to a list type. However, these cleaning steps may change the positions of the words in the tweet and add extra inaccuracies to the results of ADR spans.
3. We used regular expressions to find the matched ADR with our predicted extractions in the original tweets, the matching accuracy was highly dependent on how the twitter cleaning process changed the words and the regular expression features we used.
4. One major limitation that we believe negatively influenced our models was the quality of the training and validation data. When our team took a closer look at the individual tweets and how they were coded for ADRs, we found several instances where, from our perspective, tweets that included ADRs were not coded as having them. This means that our NER model would see potentially contradictory information on what was and was not considered an ADR.

5. Since we did not understand the deep learning model in-depth, it was quite difficult tuning it with different hyperparameters. The model gave different results to some of the same hyperparameters gave very different f1 scores which were very surprising.
6. Our final limitation relates to the tools used in the normalization portion of the task. There were issues in multiple steps: first, the Quick UMLS NER component, which was used to translate the extracted ADR text into Metamap terms, struggled to recognize less straightforward ADR extractions from the tweets (e.g., “space cadet” to “spaced out”). Next, using our corpus to match metamap terms and CUIs to MedDRA terms was also imperfect. While our corpus did include all of the MedDRA codes found in the training and validation data, we found that for each MedDRA code there are several CUIs in Metamap. Our corpus did not include all possible CUI to MedDRA Code matches. This is an issue that could be addressed through creation of a more complete corpus.

FUTURE CONSIDERATIONS

For further research on this task, we believe that there are a lot of additional areas of research that would be valuable for a true application of the methods we used in this report. Some of the further areas to consider and research in order to improve the results would be:

1. It would be better if we can mark the position of each word in the original tweets and then trace it back after their labels were predicted so that lemmatization could be used. Also, different regular expression features should be tried to increase the matching accuracy.
2. When the intermediate results were saved, we should try the most appropriate format so the features of columns can be kept (like the list in each cell of dataframe can be recognized as list,

not string). Or we should figure out how to designate the type of each column in dataframe when importing it from the file.

3. As mentioned above, our NER model was likely limited due, in part, to the quality of the data. We believe that the data quality issues lead our model to underpredict adverse events (keeping in mind that our precision for predicting the presence of an ADR was much higher than recall). If we had more time to adjust the model, one major implementation would be to set the model to potentially "over predict" ADRs. This would result in both a higher recall for tweets accurately classified as having ADRs, and it would help us to find instances where an ADR is present in tweets that were not classified as having ADRs.
4. One major limitation mentioned above was the ability of Quick UMLS to accurately map the whole extraction text to the appropriate term in Metamap. There are at least two options to address this. The first would be to expand the Metamap-MedDRA dictionary. While this option may enhance the model slightly, it does not address the core issue, where the extracted ADRs are significantly different from the accepted medical terms used in Metamap and MedDRA. An alternative option that would at least partially address that problem would be to train a corpus of text using packages such as TensorFlow. In this case, we could use the extraction and MedDRA term pairs to train a corpus on which ADRs match with MedDRA terms. This would serve as a stronger semantic match, where the current tools use fuzzy string matching that appears to be based more on lemmatization / word structure than the actual relationship of two words or phrases.

REFERENCES

1. Agichtein, E., Castillo, C., Donato, D., Gionis, A., & Mishne, G. (2008, February). Finding high-quality content in social media. In Proceedings of the 2008 international conference on web search and data mining (pp. 183-194).
2. Bruns, A. (2012). Ad Hoc innovation by users of social networks: The case of Twitter. ZSI Discussion Paper, 16(2012), 1-13.
3. Burgess, J. (2010, June). Remembering Black Saturday: the role of personal communicative ecologies in the mediation of the 2009 Australian bushfires. In ACS Crossroads Conference, Hong Kong.
4. https://www.internetlivestats.com/twitter-statistics/
5. Chen, E., Lerman, K., & Ferrara, E. (2020). Covid-19: The first public coronavirus twitter dataset. arXiv preprint arXiv:2003.07372.
6. Ginn, R., Pimpalkhute, P., Nikfarjam, A., Patki, A., O'Connor, K., Sarker, A., ... & Gonzalez, G. (2014, May). Mining Twitter for adverse drug reaction mentions: a corpus and classification benchmark. In Proceedings of the fourth workshop on building and evaluating resources for health and biomedical text processing (pp. 1-8).
7. Proceedings of the 4th Social Media Mining for Health Applications (#SMM4H) Workshop & Shared Task, pages 107–109
8. Luca Soldaini and Nazli Goharian. "QuickUMLS: a fast, unsupervised approach for medical concept extraction." MedIR Workshop, SIGIR 2016.