Social Media Mining for Health Monitoring

Abishek Dalvi (adalvi@buffalo.edu)
Navpinder Singh (navpinde@buffalo.edu)
Prithvisagar Rao (psrao@buffalo.edu)

1 Introduction

With many adults using social media to discuss health information, researchers have begun div ing into this resource to monitor or detect health conditions on a population level. Twitter, specifically, has flourished to several hundred million users and could present a rich information source for the detection of serious medical conditions, like adverse drug reactions (ADRs). Adverse drug reactions (ADRs) are defined as "injuries resulting from medical drug use". Though there are many reporting tools and social support networks to self-report, they reflect less than 10% of the adverse effect occurrences. They are also associated with a substantial time lag before such events are officially published.
The challenge with Twitter data is the lack of structure in tweets. The highly unstructured, noisy, real-world, and substantially creative language expressions contain many abbreviations, some of which are unique to Twitter. They contain frequently misspelled words, colloquialisms, idioms, and metaphors that make automatic processing more difficult. For the task of mining for ADR mentions, these issues compound an already challenging problem.
The goal of this project is to develop an end-to-end system complete with a website to detect tweets mentioning an ADR, map the extracted colloquial mentions of ADRs in the tweets to standard concept IDs in the MedDRA vocabulary (lower level terms) and highlight the text span of the ADR mention in the fetched tweets.

2 Objective

The project is divided into the following four tasks:

Task 1: Automatic classification of adverse effects mentions in tweets
Classify the tweets based on mentions of Adverse Drug Reactions(ADRs) from those that do not. This is a binary classification problem.

Task 2: Extraction of Adverse Effect mentions
To identify the text-span of the reported ADRs and distinguish ADRs from similar non-ADR expressions.

Task 3: Normalization of adverse drug reaction mentions (ADR)
The objective is to detect tweets mentioning an ADR and to map the extracted colloquial mentions of ADRs in the tweets to standard concept IDs in the MedDRA vocabulary (lower level terms).

Task 4: Live Demonstration of an end-to-end system
Task 4 requires building a website demonstrating an end-to-end system of ADR mentions in the tweets and highlighting the text span of ADR mentions/indications, annotation indicating ADR and mapping the extracted ADR to the corresponding MedDRA term.

3 Literature Survey

Using machine learning and deep learning methods to find mentions of adverse drug reactions in social media:
SMM4H every year mentions some particular tasks to be handled. This paper tackles automatic classifications of adverse effects mentioned in tweets and extraction of adverse effect mentions.

In task automatic classifications of adverse effects the goal is a binary classification problem. The designed system for this sub-task should be able to distinguish tweets reporting an Adverse Effect (AE) from those that do not.
In the second task called Extraction of Adverse Effect mentions. This task includes identifying the text span of the reported ADRs and distinguishing ADRs from similar non-ADR expressions. ADRs are multi-token, descriptive, expressions, so this subtask requires advanced Named Entity Recognition (NER) approaches.
A results comparisons between various techniques is shown in this paper such as SVM, CNN, Conditional Random Fields and CRF+W2V

Recurrent Neural Network for Text Classification with Multi-Task Learning:
The paper introduces new methods for text classification using Uniform-layer in LSTM, Coupled Layers of LSTM and Shared Layers in LSTM. These models are then compared with existing techniques like TREE-LSTM, Dynamic Convolutional Neural Network and Matrix-Vector Recursive Neural Network. From the experimental results, they concluded that the differences among them are the mechanisms of sharing information among the several tasks. Therefore they inferred that the models can improve the performances of a group of related tasks by exploring common features.

BioBERT: a pre-trained biomedical language representation model for biomedical text mining:
BioBERT is a domain-specific language representation model pre-trained on large-scale biomedical corpora(PubMed abstracts and PMC full-text articles). With almost the same architecture across tasks, BioBERT largely outperforms BERT and previous state-of-the-art models in a variety of biomedical text mining tasks when pre-trained on biomedical corpora.

4 Dataset

The details on the dataset for the three tasks are as follows:

Task 1 Dataset:
For task 1 of automatic classification of adverse effects mentions in tweet, the given data set contains 25,672 tweets as training data with 2,374 positive and 23,298 negative data. Positive data is the data with ADR mentions and negative are without any ADR mentions. Each tweet contains the tweet id, user id of the user who tweeted the tweet and binary annotation indicating the presence or absence of ADRs. We are provided with approximately 5,000 tweets as evaluation data.

Task 2 Dataset:
For task 2 of extraction of adverse effect mentions we need to identify the text span of the reported ADRs and distinguish ADRs from similar non-ADR expressions. We have been provided with 2,367 training dataset consisting of 1,212 positive and 1,155 negative tweets. Evaluation dataset consists of 1,000 tweets. Each tweet contains the tweet id, the start index and the end index of the span containing ADR, the annotation indicating if the tweet contains ADR or not and the ADR text covered by the span indices in the tweet.

Task 3 Dataset:
Task 3 of the project is normalization of extracted ADR mentions to MedDRA preferred term identifiers. Since this is an extension of Task 2 we are given the same dataset.

5 Model

Task 1:

Preprocessing:
For working with the pre-trained BERT, we need to use the tokenizer provided by the BERT model. BERT has a specific, fixed vocabulary and handles out-of-vocabulary words in a particular way.
The “vocab.txt” file provides the tokenizer vocabulary for the BioBERT model.

Formatting:
We format the input tweets in a specific way for the BERT model.

- Special Tokens:
  - At the end of every sentence we append the [SEP] token.
  - [CLS] token in prepend to the beginning of every sentence. The final hidden state corresponding to this classification token is used as the aggregate sequence representation for classification tasks.
- All sentences must be padded or truncated to a single, fixed length. Padding is done with a special [PAD] token.
- We find the maximum length in the training corpus and pass it to the tokenizer.encode_plus function. This function combines the following steps:
  - Split the sentence into tokens.
  - Add the special [CLS] and [SEP] tokens.
  - Map the tokens to their IDs.
  - Pad or truncate all sentences to the same length.
  - Create the attention masks which explicitly differentiate real tokens from [PAD] tokens.

Architecture:
BERT (Bidirectional Encoder Representations from Transformers) is a transformer-based method of learning language representations. It is a bidirectional transformer pre-trained model developed using a combination of two tasks namely: masked language modeling objective and next sentence prediction on a large corpus.
BioBERT is a biomedical language representation model designed for biomedical text mining tasks. BioBERT is trained on the PubMed and PMC Pre-training corpora using the BERT pre-trained weights. We fine-tune this model for classification of tweets containing ADR mentions. Fine-tuning of the BERT model is done for quicker development and obtaining best results using less amount of data.
In our implementation, we have made use of the Hugging Face library's Pytorch interface for working with BERT. The library also includes pre-built modifications of pre-trained transformer models for a variety of specific tasks. We have used the BertForSequenceClassification class in this implementation. The Hugging Face pytorch implementation includes a set of interfaces that are built on top of the trained BERT model. Each of them have different top layers and output types designed for specific NLP tasks. As mentioned, we have used BertForSequenceClassification which is a normal BERT model with an added single linear layer on top for classification that we will use as a sentence classifier.
We load the BioBERT pretrained model in the following way:

```
from transformers import BertModel, BertTokenizer, BertConfig, BertForSequenceClassification
config = BertConfig.from_json_file('/content/gdrive/My Drive/Biobert/bert_config.json')
model= BertForSequenceClassification.from_pretrained('/content/gdrive/My Drive/Biobert/model.ckpt-1000000.index'
                                                     ,from_tf=True, config=config)
model.cuda()
```

Figure 1: BioBert Load Module Command

We need to explicitly specify the configuration of the pretrained BioBERT model. The bert_config.json file provides us with the configuration. The index file is the pretrained BioBERT model.
We have used the AdamW optimizer from the HuggingFace library. The other hyperparameter values are as follows:

- Batch size: 64 #need to change
- Learning rate: 2e-5
- Epochs :4

The training summarization can be given as follows:

- Unpack the data inputs and labels
- Load data onto the GPU
- Clear out the gradients calculated in the previous pass

- Feed input through the network using forward pass.
- Backpropagate
- Update the parameters

Results:
We obtained a Positive F1 score of 0.6.

	Precision	Recall	F1 Score
Positive	0.54(+0.18)	0.67(+0.05)	0.60(+0.14)
Negative	0.97(+0.01)	0.94(+0.05)	0.95(+0.03)
Macro			0.78(+0.10)
Micro			0.92(+0.05)
Weighted			0.92(+0.05)

Table 1: Comparison between Final Results and Baseline Model For Task 1

For baseline model we used Multinomial Navies Bayesian using TF-IDF vectors.
Other approaches:

- BERT fine-tuning: Before moving on to BioBERT, we implemented BertForSequenceClassification with BERT tokenizer and weights. After observing promising results and researching and finding about BioBERT we moved to BioBERT pre-trained model.

Task 2:

Preprocessing:
We use the BILOU convention for tagging the preprocessed words (The preprocessing process is same as the previous task)
The Tagging scheme is as follows:-

- B-ADR - 'beginning'
- I-ADR - 'inside'
- L-ADR - 'last'
- O - 'outside'
- U-ADR - 'unit'

Architecture:

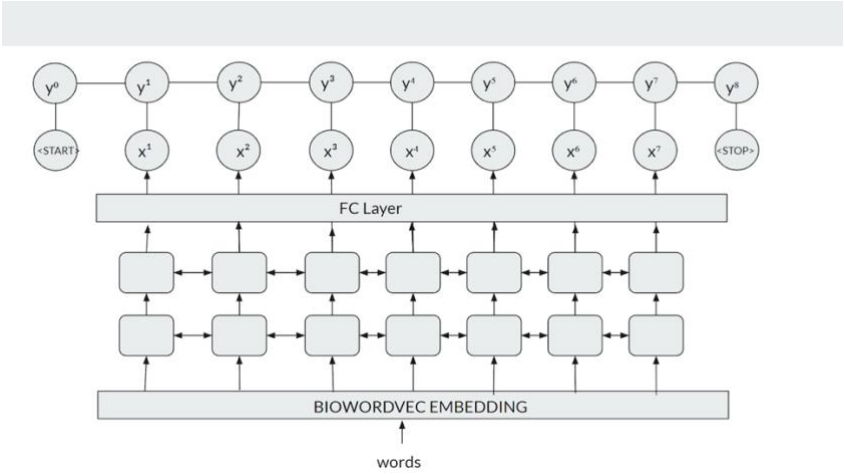

Figure 2: Bio-Word-Vec Embeddings + Bi-Lstm +CRF with constrained transition probabilities

- Bio-Word-Vec Embeddings:
  These are fasttext word embeddings with dimension size of 200 trained on PUBMED+ MIMIC -III and MESH. This model is an extension to the original biowordvec which was trained on a word2vec model. We copy these embeddings of size 200 and vocabulary of roughly 35k to the embedding layer of the Bi-LSTM.
- Bi-LSTM model:
  A 2 layer Bi-LSTM model is used where the output of each timestep is passed through a fully connected layer, whose outputs will be used as input features for the CRF model.
- Conditional Random Fields with constrained transition probabilities:
  In the CRF transition matrix, the transition potentials from certain labels to certain labels are set to -100, so that the labels never transition in a particular sequence.
  The transitions where this assignment is done are:
    - From “B-ADR” to “U-ADR”
    - From “B-ADR” to “O”
    - From “U-ADR” to “I-ADR”
    - From “U-ADR” to “L-ADR”
    - From “I-ADR” to “U-ADR”
    - From “I-ADR” to “B-ADR”
    - From “O” to “I-ADR”
    - From “O” to “L-ADR”

  The above assignments force the CRF to follow a sequence that the labelling sequence follows, enforcing the model to learn the syntactic nature of the labels and applying it to the data. This leads to early and better convergence of the model.
  The forward algorithm is performed on the CRF to get the forward score and a negative likelihood loss is calculated between the forward score and the gold standard score.
  This loss is back propagated using adam optimizer and the weights are updated.
  Finally, during testing, to calculate the best sequence of labels conditioned upon the input features of the CRF, Viterbi decode algorithm is used to get the best possible sequence.

Results:

- Strict Evaluation:
  Sequence Accuracy:- 50.46% (+18.68) Accuracy:- 91.11%

	Precision	Recall	F1 Score
I-ADR	0.451(+0.376)	0.302(+0.177)	**0.362(+0.268)**
L-ADR	0.447(+0.348)	0.384(+0.214)	**0.413(+0.288)**
U-ADR	0.465(+0.302)	0.609(+0.498)	**0.527(+0.395)**
B-ADR	0.425(+0.314)	0.367(+0.226)	**0.394(+0.253)**
O	0.954	0.961	0.958
Macro score			0.531(+0.255)
Micro score			0.910(+0.010)
Weighted score			0.909(+0.003)

Table 2: Comparison between Strict Evaluation of

Final Result and Baseline Model For Task 2

Our strict evaluation is the same as the evaluation which was followed in the competition.

The state of the art F1-score is 0.464 while ours is 0.424 (macro average of only the positive labels)

- Relaxed Evaluation:
  Sequence Accuracy:- 50.70%(14.80) Accuracy:- 92.23%

	Precision	Recall	F1 Score
ADR	0.587(+0.292)	0.544(+0.168)	**0.565(+0.234)**
O	0.954	0.961	0.958
Macro score			0.761(+0.124)
Micro score			0.922
Weighted score			0.921

Table 3: Comparison of Relaxed Evaluation between

Final Result and Baseline Model For Task 2

Our relaxed evaluation is stricter than the relaxed evaluation in the competition. The competition uses overlapping concept i.e. the extracted words should be a subset of the true extraction. Our model under this evaluation will produce around the same F1 score as the current state of the art.

For baseline model we used Conditional Random fields.

Future Work:

- Using Bio-Bert or Roberta before instead of the embedding layer would be the next step, which requires a lot of computing power. Also, increasing the number of layers in the LSTM can be done but to avoid vanishing gradient problem from top to bottom, a residual network would be needed.

Task 3:

Architecture:

- Deep Walks:
  From each sample we take the drug, the extraction, the meddra term and the meddraID and consider each of these terms as a node. Then each node is exploded/divided into nodes based on their tokenization.
  E.g.

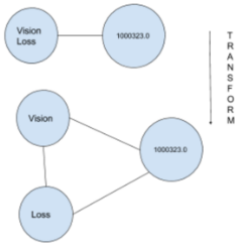

Figure 3: Creation of the Deep Walk Graph

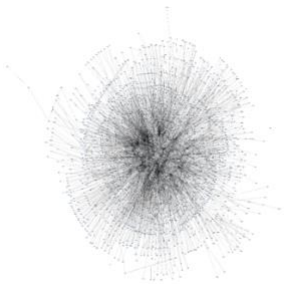

Figure 4: Final Deep Walk Graph

On this graph we perform the Deep Walk Algorithm to get the embeddings of each node. The window size is specified as 4 because the association between words and the meddra-id as tight as possible.
During testing, to handle unknown vocabulary i.e nodes which are not present in the graph, we create a common vocabulary of BioWordVec and Deepwalk vocabulary.
The unknown word is replaced with a word from the common vocabulary such that the word which replaces it should have the minimum cosine distance within the BioWordVec space when compared to the rest of the common vocabulary.
After handling the unknown words, the cosine distance between the extractions and the meddra Ids are found out, and the meddraId with the least distance is assigned to the extraction.
For baseline we used BioWordVec-PubMed word embedding + Gaussian Naive Bayes classfier.

Results:
The results from the deepwalk algorithm vary a lot because deewalk uses randomized walks.
The highest F1 score obtained is of represented by "!"

	Precision	Recall	Macro F1 Score	Micro
Deepwalks	0.275	0.313	**0.193(*!0.295*)**	**0.290(*!0.38*)**
BioSentVec Logistic regression	0.239	0.261	**0.232**	

Table 4: Comparison between Final and Baseline Model For Task 3

Other Approaches:

- Logistic Regression with BioWordVec Embeddings: We used word embeddings for each of the extracted words and calculated the centroid for the extraction. Word embeddings did not capture context and hence gave low results.
- Logistic Regression with Fasttext embeddings: Results similar to BioWordVec embeddings were obtained for this approach.

Future Work:
A combination of deepwalks and logistic regression could be a promising technique to get better results, albeit combining these two approaches by ensemble voting might not give the best results. One decent approach can be mapping the meddraId embeddings which we get from the deep walk algorithm to the BioWordVec space. Doing the mapping task is an unexplored territory but our best bet to do this would be using associations between words. Words in the graph which form an association i.e vision-loss ~ blind, hold true even in the BioWordSpace too. Therefore, vision+loss ~ 1000349.1 holds true in the deep walk embeddings and should also hold true in the BioWordSpace.

6 End to End System:

For end to end system we used Angular 9 for UI and Flask for Backend. We created basic dashboard to see how our models performed on each and every tweet and to test new tweet in the system.

ADR Classification:
The ADR classification tab corresponds to the first task of our project. The left section of the page here displays the tweets from the test dataset. The right section displays the tweets as classified by our model. Blue line on top of the tweet denotes a Non-ADR tweet. Green denotes an ADR tweet.

ADR Extraction:
This tab corresponds to the second task of the project. The red-highlights on the tweet words are the ADR extraction from the dataset. The model predicted ADR extractions are presented below the tweet. The blue-highlighted words in the tweet represent any drug mentions.

ADR Normalization:
This tab presents the results of the third task of the project. The Meddra terms are present below the tweet, where left section contains the Meddra terms from the dataset and the right section contains the Meddra term as predicted by our model.

Charts:
The charts tab gives a visual representation of the data. These charts are developed from the respective data needed by the charts.

Live Test:
In this we tried to find the ADR Extraction from a given query. ADR if present is highlighted in Red Color in the query and corresponding Meddra Term and Code is displayed below.

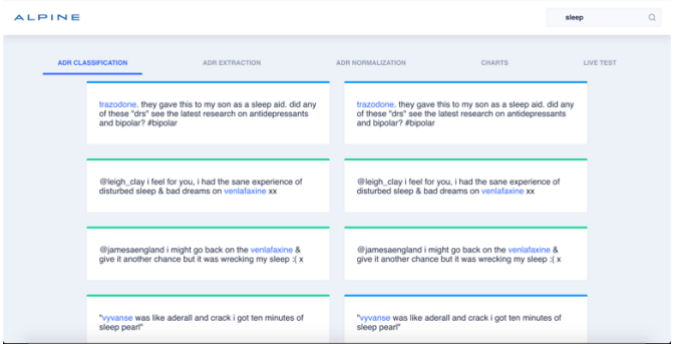

Figure 5: ADR Classification

Figure 6: ADR Extraction

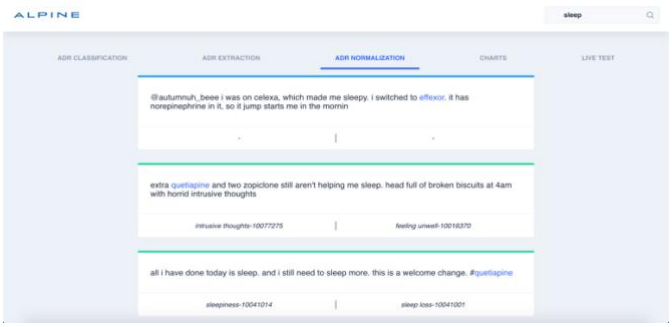

Figure 7: ADR Normalization

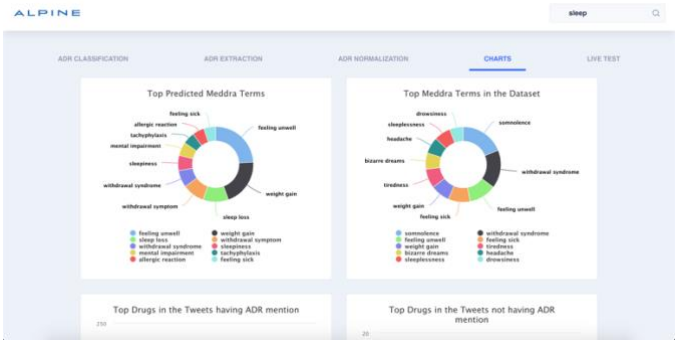

Figure 8: Charts

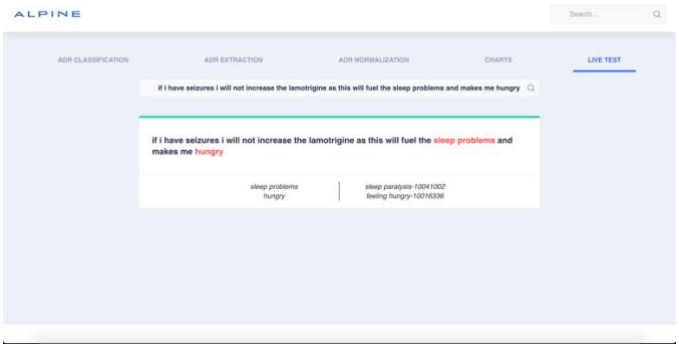

Figure 9: Live Test

7 Contributions:

Abhishek Dalvi:

- Task 2: BiLSTM + CRF (proposed model)
- Task 2: CRF (baseline model)
- Task 3: Deepwalk (proposed model)
- Task 3: BioWordVec Logistic Regression

Navpinder Singh:

- Task 1: Multinomial Naive Bayesian (baseline)
- Task 3: BioSentenceVec Logistic Regression (proposed)
- Frontend Angular web application
- Backend application

Prithvisagar Rao

- Task 1: BioBERT Sentence Classification (proposed)
- Task 3: BioSentenceVec Logistic Regression
- Task 3: Gaussian Naive Bayesian, SVM (baseline)
- Backend application

8 References:

- BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding (Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova)
- BioBERT: a pre-trained biomedical language representation model for biomedical text mining (Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, Jaewoo Kang)
- RoBERTa: A Robustly Optimized BERT Pretraining Approach(Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, Veselin Stoyanov)
- BioWordVec, improving biomedical word embeddings with subword information and MeSH(Yijia Zhang, Qingyu Chen, Zhihao Yang, Hongfei Lin,Zhiyong Lu)
- Bidirectional LSTM-CRF Models for Sequence Tagging(Zhiheng Huang, Wei Xu, Kai Yu)
- DeepWalk: Online Learning of Social Representations(Bryan Perozzi, Rami Al-Rfou, Steven Skiena)
- http://doi.org/10.1093/bioinformatics/btz682
- https://www.ijcai.org/Proceedings/16/Papers/408.pdf
- https://www.aclweb.org/anthology/W19-3216.pdf