SOCIAL MEDIA MINING FOR HEALTH MONITORING

Team 7
Ajit Bhat (abhat3)
Mahitha Mereddy (mahitham)
Nikita Sawant (nsawant)

Abstract

With the rise of social media in the last decade, it has become more and more important to be able to understand such lingo to exact information for various reasons. The problem statement is to analyze tweets for health monitoring purposes. Specifically, to classify tweet into two classes – if the tweet contains a drug mention, is it causing an adverse reaction or not and if there happens to be an adverse reaction then extract such a phrase and normalize it to the standard MEDDRA ID. The state-of-the-art models mostly use a variation of the BiLSTM model to capture the context of a sentence and then model that to meaningful observations. We propose a separate architecture for each of the above explained tasks. Data from social media proves to be difficult to model given the nature of words being used in shorthand and not following the conventional syntax/semantic rules of a language.

1 Introduction

Pharmacovigilance is the practice of monitoring the effects of medical drugs after they have been licensed for use, especially in order to identify and evaluate previously unreported adverse reactions. People discuss their health-related experiences including the use of prescription drugs, side effects, and social media treatments, making social networks unique and robust sources of health, drug, and treatment information. Some researches have focused on extracting specific mentions of adverse reactions (and their lexical variants) and identifying associations between particular drugs and adverse reactions. While most of the past approaches are based on lexicons, recent approaches have employed supervised extraction learning techniques. Following the extraction, co-occurrence metrics have been applied for quantifying adverse-drug reaction associations. We discuss various models for the classification, extraction and normalization of adverse drug effects based on twitter data.

2 Background and Related Work

Dataset

The Mendeley dataset is composed of public tweets downloaded using the official streaming API provided by Twitter and made available in accordance with Twitter's data use policy. There are 20,544 tweets in the training set and 5,134 tweets in the test set with a heavy imbalance of classes. The training set has only 1903 ADR tweets and 18641 non-ADR tweets. The test set has only 474 ADR tweets and 4660 non-ADR tweets. The following are the fields in the dataset:

- tweet_id – Unique identifier for the tweet
- user_id – Unique identifier for the user
- class – 0 for a non-ADR tweet and 1 for an ADR tweet
- tweet – The actual tweet from the user

2.1 Automatic classification of tweets mentioning an ADR

The first task is the binary classification of tweets to predict if a tweet contains an ADR or not. In the past, feature engineering and the training of simple classifiers such as Naive Bayes, linear regression models or support vector machines (SVMs) addressed the problem of text classification. Due to their potential to achieve high precision with less need for engineered features, text classification has benefited greatly from the recent resurgence of deep-learning architectures.

Word embeddings are a type of word representation that allows words with similar meaning to have a similar representation. Word embeddings have been observed to perform significantly better than POS tags, word counts and other features, especially with neural network architectures like Convolutional Neural Networks, Recurrent Neural Networks, Long Short-Term Memory and Gated Recurrent Units. The state-of-the-art for this particular task was achieved using a Bi-Directional LSTM architecture with Conditional Random Fields (CRF).

2.2 Extraction of adverse drug mentions

After classifying a tweet as ADR/non-ADR, we proceed to extract the adverse drug mentions in the tweet concerned. While text extraction is too abstract, we specifically try to model it as a Named Entity Recognition (NER) which is necessarily a multiclass classification problem where the text we are interested in belongs to a set of entity classes (for ex. Person, Organization, Location, etc.). We divide the problem into ADR and non-ADR classes, effectively converting it to a two-class classification.

There has been extensive research in the domain of NER and text classification. Some of the state-of-the-art models include SpaCy, Stanford NER tagger, NER by Allen NLP. The core intuition is to understand the context of a word in a given sentence given its neighbors, either by a window function or the entire sentence as a context to model the outcome. This is achieved by memory-conserving models such as Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), Conditional Random Fields (CRF) which all aim to learn context of a word based on its history of occurrences and its relevance in a given sentence. We use the same motivation to develop a BiLSTM-CRF based model with Fasttext embeddings as word weights based on the vocabulary of our dataset.

2.3 Normalization of adverse drug reaction mentions

The objective of the task 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). MedDRA (Medical Dictionary for Regulatory Activities) is the standard nomenclature for monitoring medical products, and includes diseases, disorders, signs, symptoms, adverse events or adverse drug reactions.

Pre-trained word embeddings like Word2Vec and Glove along with neural architectures using word embeddings pre-trained with the Bidirectional Encoder Representations from Transformers (BERT) gave the best result for this task. However the accuracy for this task was overall low so human inspection is still required to make use of the data extracted automatically.

3 Implementation

We follow the architecture in figure 1 which clearly delineates the separation of tasks and their respective components.

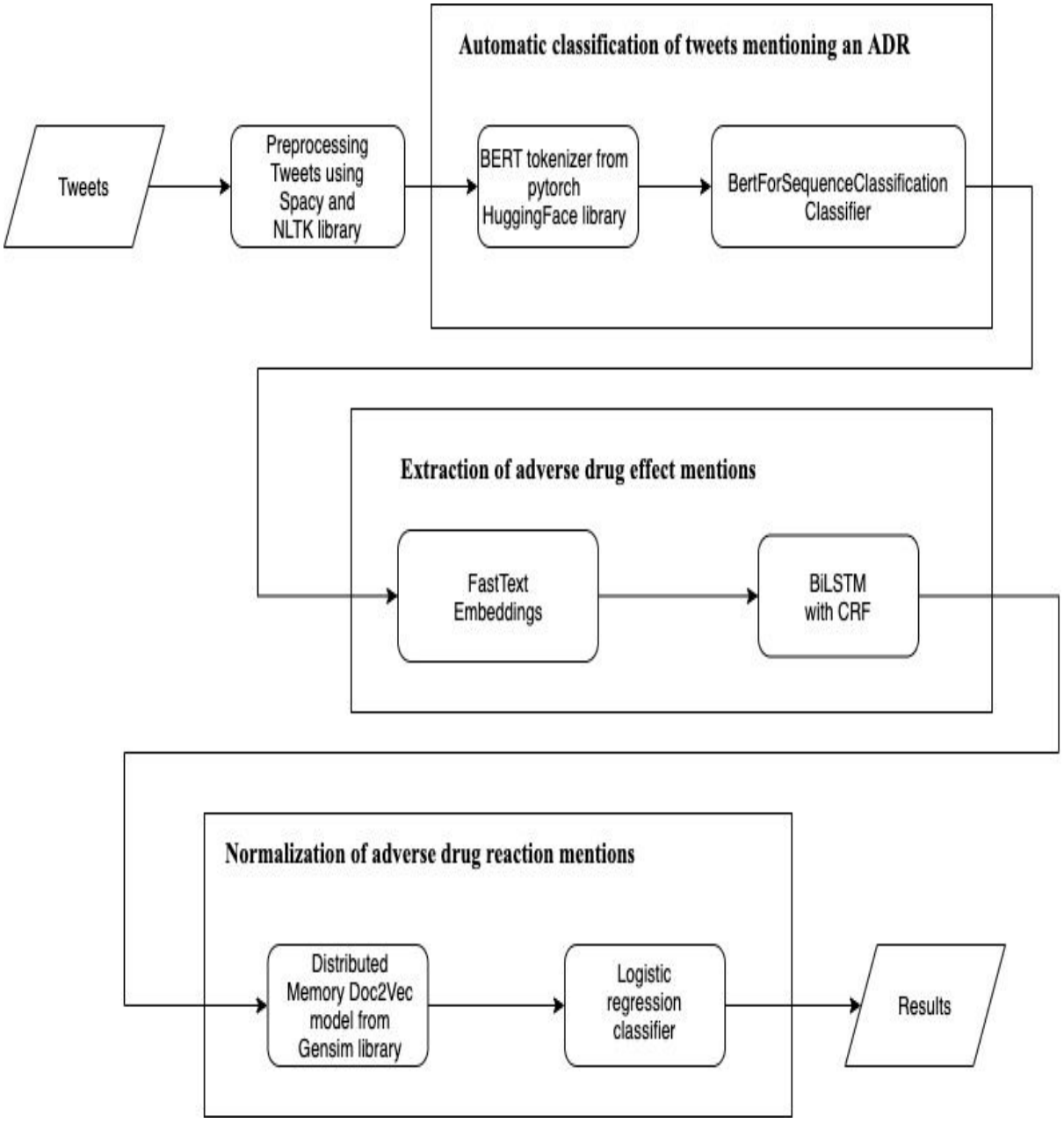


fig1. Pipeline diagram

Data preprocessing - The tweets are processed and cleaned in the following manner

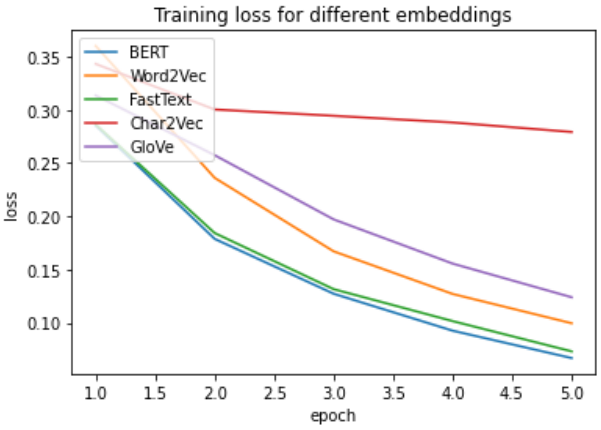
1. Conversion to lowercase
2. Accented character conversion to unicode
3. Removal of the # from hashtags
4. Removal of punctuations
5. Removal of URLs
6. Removal of emojis and emoticons
7. Conversion of contractions to their full form
8. Conversion of slang words and acronyms to their full forms
9. Lemmatization

We did not remove the stop words as we believe they are important in maintaining the dependencies between the words and would make for better word representations that perform well with the models we have chosen.

3.1 Task1- Automatic classification of tweets mentioning an ADR

We experimented with a few models after developing the baseline and one model that gave good results was a neural network architecture as follows:

- Embedding layer with BERT embeddings
- Bi-LSTM layer with 50 units
- Self attention layer with sigmoid activation
- Bi-LSTM layer with 50 units
- Self attention layer with sigmoid attention
- 2 layers of GRU with 50 units each



With the above architecture, we got an F-score of 0.51 with BERT embeddings.

For our final model, we have leveraged the power of the HuggingFace transformer library's pre-trained model. The reason we chose this as our final model is because it took much less time to train and fine-tune, performed well on our small dataset and gave better results.

We use the BertTokenizer which encodes the tweets in the format required for the classifier, namely - tokenize the sentence, prepend [CLS} token to the start and [SEP] token at the end of the sentence, map tokens to their IDs, pad or truncate sentence to a fixed length and create attention masks for the sentence. The input IDs, attention masks and labels are passed to the pre-trained BertForSequenceClassification classifier model. This is the normal BERT model with an added single linear layer on top for classification that we will use as a sentence classifier. As we feed input data, the entire pre-trained BERT model and the additional untrained classification layer is trained on our specific task. We use the "bert-large-uncased" pre-trained model which has 24-layers, 1024-hidden nodes, 16-heads, 340M parameters and is trained on lower-cased English text. For the fine-tuning, we used the following hyperparameters:

- Batch size - 32
- Number of epochs - 4
- Learning rate - 2e-5

3.2 Task 2 - Extraction of adverse drug effect mentions

We employ NER techniques to solve this problem. Essentially, we model it as a two-class supervised problem where word in a sentence is marked as an ADR tag if it is part of/is an adverse effect mention, else it is tagged as non-ADR.
For the baseline, we used a combination of SpaCy Dependency Parser along with a custom TextRank model. It gave us a flat accuracy of about 53%, which is quite good considering it was a rule-based method.
For the final model, we tried BiLSTM architectures with Attention and CRF as final layers, and the CRF model produced the best results. The model is described below.
We use the pretrained Fasttext model to learn word embeddings in a n-dimensional space. This ensures that two words with similar meanings share the same vector space and serve as weights to the embedding layer of the model.

The model uses a BiLSTM-CRF structure and the transition path is found using the Viterbi algorithm on the CRF graph generated. The bidirectional LSTM accepts the learnt word embeddings as inputs and maps them to the hidden layers to learn context between words in a sentence. These weights are then mapped to a tag space so as to prepare them for the CRF layer. The CRF layer consists of probabilities of transitions from a particular word to a tag and probabilities of transition of a tag to another tag. These two probabilities together make up the logic of this layer. The problem is when the mapping of hidden dimensions to tag increases tremendously, then it is computationally expensive to traverse the CRF graph. For this reason, we use the Viterbi algorithm to decode the best path by finding tag transitions and then mapping those transitions to the feature of the sentence to considerably reduce the traversal time.

We run the models with the following values for hyperparameters:

- Embeddings dimension – 100
- Hidden layers – 64
- Loss threshold – 0.1 (when loss drops below 0.1, we stop training)

This configuration yielded the best results. The architecture for the model is shown in fig 2.

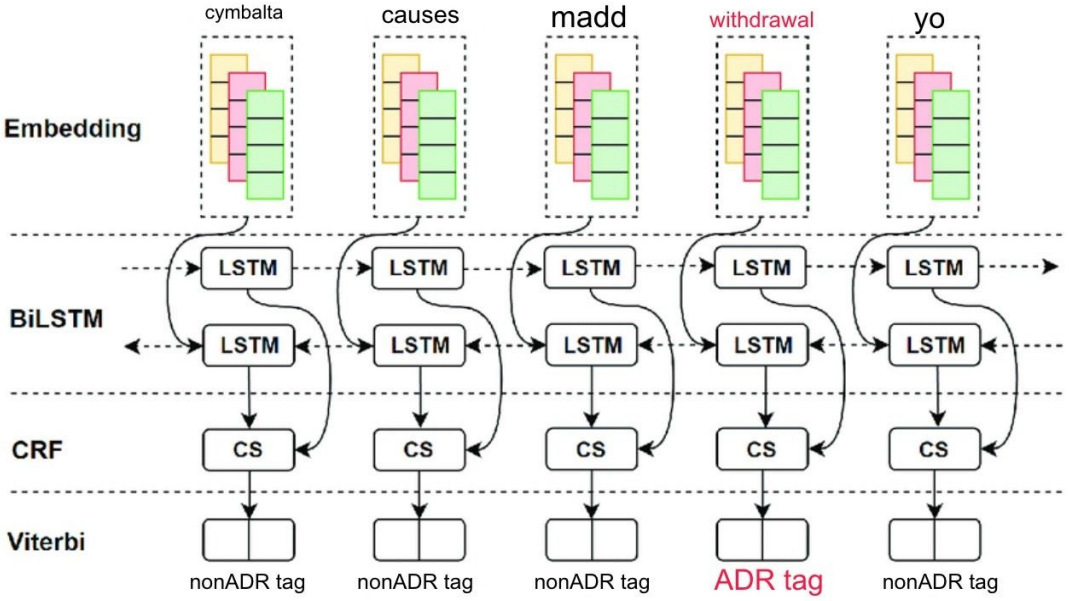


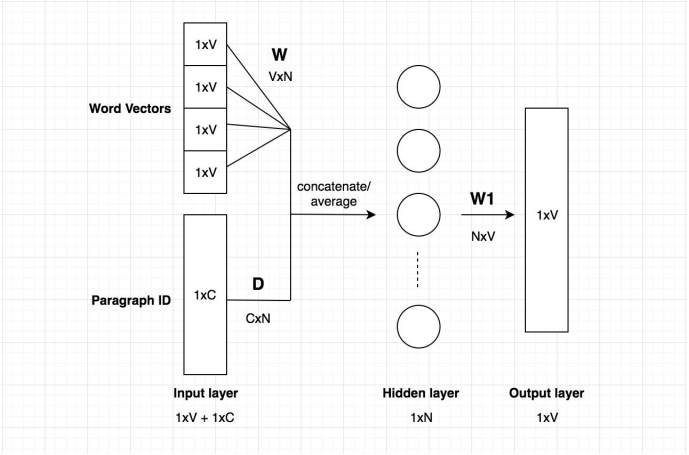
fig2. BiLSTM-CRF architecture with Viterbi decoding

3.3 Task3 - Normalization of adverse drug reaction mentions

Preprocessing - The input consisted of tweets classified as ADRs and non-ADRs. The dataset contained columns including tweets, MEDDRA code, MEDDRA term, drug name, type, extraction among other things. The extracted features contained special characters, capitalized words, and spelling errors. The following preprocessing of the data was done:

- For the classification task, only data which was classified as ADR was considered.
- As a part of text cleaning the data was converted to lowercase and special characters were removed.
- Using NLTK tokenizer the phrases were split into single tokens.
- Using TaggedDocument from gensim python library extracted features were tagged to their respective labels.

For the baseline we used a Distributed Bag Of Words embedding model from Doc2Vec (PV-DBOW). A Multiclass training algorithm Logistic Regression from sklearn was used for classification.
We tried implementing the LSTM model using BioBERT embeddings. Surprisingly the results were not a significant improvement over our baseline. We tried this approach with varied network structure without any success.
The dataset provided to us was highly unbalanced in the sense that it consisted of 475 classes of which the majority of samples were associated with only around 30 classes. This was affecting the accuracy of the model.
For our final implementation we decided on a Distributed Memory model from Doc2Vec (PV-DM). The Distributed model learns to predict a center word based on the context. In our case a Distributed model combined with a Logistic regression classifier gave the highest accuracy. The class_weight parameter in the Logistic Regression classifier was set to 'balanced' to handle unbalanced data distribution. The "balanced" mode uses the values of y to automatically adjust weights inversely proportional to class frequencies in the input data.
Doc2Vec is an extension of Word2Vec in a way that there is an added another vector - Paragraph ID. Instead of using just words to predict the next word, there is another feature vector, which is document-unique.So, when training the word vectors, the document vector is trained as well, and at the end of training, it holds a numeric representation of the document.



The model above is a Distributed Memory version of Paragraph Vector (PV-DM). It acts as a memory that remembers what is missing from the current context — or as the topic of the paragraph. While the word vectors represent the concept of a word, the document vector intends to represent the concept of a document.

4 Results

4.1 Task1 - Automatic classification of tweets mentioning an ADR

Baseline results

bioBERT + SVM SMOTE oversampling

Classifier	F-score	Precision	Recall	Accuracy
Linear SVM	0.41	0.29	0.73	81%
Random Forest	0.30	0.37	0.26	89%

Ada Boost	0.33	0.23	0.63	77%
Gaussian Naive Bayes	0.26	0.17	0.59	69%
Logistic Regression	0.42	0.29	0.65	81%
Decision Tree	0.24	0.18	0.36	78%
Gradient Boost	0.37	0.26	0.65	79%
KNN	0.22	0.13	0.96	38%

Different embeddings

Embedding	Best F-score	Best classifier
Word2Vec (Continuous bag of words)	0.27	Linear SVM
Word2Vec (Skip-gram)	0.32	Gradient Boost
Doc2Vec	0.32	Gradient Boost
Universal Sentence Encoder	0.38	Linear SVM
BERT	0.41	Logistic Regression
bioBERT	0.42	Logistic Regression

Different oversampling techniques

Oversampling method	Best F-score	Best classifier
Synthetic Minority Oversampling Technique (SMOTE)	0.38	Logistic Regression
Borderline SMOTE	0.40	Logistic Regression
Borderline SMOTE SVM	0.41	Logistic Regression
Adaptive Synthetic Sampling (ADSYN)	0.38	Logistic Regression

Neural Network from scratch results

Embeddings	F-score	Precision	Recall	Accuracy

Char2Vec	0.50	0.52	0.49	91%
Word2Vec	0.44	0.58	0.36	92%
FastText	0.47	0.57	0.41	92%
GloVe	0.49	0.49	0.50	90%
BERT	0.51	0.49	0.54	91%

Comparison of results

Model	F-score	Precision	Recall
Baseline - bioBERT with Logistic Regression	0.42	0.29	0.76
Intermediate model - biLSTM + Self Attention + GRU with BERT	0.51	0.49	0.54
Final model - bert-large-uncased + BertTokenizer + BertForSequenceClassification	0.62	0.60	0.64
State-of-the-art-BERT + CRF	0.64	0.608	0.69

4.2 Extraction of adverse drug effect mentions

The results for baseline and the final models are depicted in the tables below.

4.2.1 Baseline models

Model	Flat Accuracy (rule-based)
PyTextRank	36.9%
SpaCy Dependency Parser with TextRank (custom)	~ 53 - 54%

4.2.2 Final Model

Model	F1-score (averaged)	Precision (averaged)	Recall (averaged)

BiLSTM-CRF with FastText embeddings modelled as 2-class classification	64.84%	66.20%	63.76

Bounds	**F1-score**	**Precision**	**Recall**
Lower bound (min value recorded - Class ADR)	35.9%	34.42%	37.52%
Higher bound (max value recorded - Class ADR)	44%	42.72%	45.36%
Lower bound (min value recorded - Class NON-ADR)	91.46%	94.4%	88.71%
Higher bound (max value recorded - Class NON-ADR)	95.05%	95.31%	94.81%

The imbalance in results can be attributed to the imbalance in the number of samples in each class.

4.3 Normalization of adverse drug reaction mentions

4.3.1 Baseline Models

Model	**Accuracy**	**F1 score**
PV-DBOW with Logistic Regression Classifier	~35.2%	~37.4%
PV-DBOW with SGD Classifier	~33.9%	~34.5%
PV-DBOW with LinearSVC Classifier	~32.8%	~33.5%

4.3.2 Final Model

Model	**Accuracy**	**F1 score**
PV-DM with Logistic Regression	~41.8%	~42.7%
PV-DM with SGD Classifier	~38.2%	~39.8%
PV-DM with LinearSVC Classifier	~39.8%	~40.5%

5 Scope for future work

We suggest a couple of ways to improve accuracy. We can employ one-class classification models to build an anomaly detection kind of architecture to classify classes more accurately. We can also try k-fold cross validation for consistent results and learning. Word-sense disambiguation is another technique to give better context understanding, but this is quite a daunting task and needs further in-depth research.

Contributions

The tasks were split equally among all 3 individuals.
Ajit Bhat - worked on Task2 which involves SpaCy with TextRank baseline, and BiLSTM-CRF model with Viterbi path traversal for the final model.
Mahitha Mereddy - Worked on Task2 which has the bioBERT with logistic regression baseline and the BertForSequenceClassification classifier with embeddings from the bert-large-uncased model.
Nikita Sawant - Worked on Task 3 which had PV-DBOW with Logistic Regression Classifier as a baseline and PV-DM with Logistic Regression for the final model.

References

https://www.aclweb.org/anthology/W19-3203.pdf - Overview of the Fourth Social Media Mining for Health (#SMM4H) Shared Task at ACL 2019

https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/ - Imbalanced sampling

https://github.com/huggingface/transformers - HuggingFace Transformers Library

https://huggingface.co/transformers/v2.2.0/pretrained_models.html - Pre-trained language models

https://spacy.io/ - SpaCy

https://nlp.stanford.edu/software/CRF-NER.html - Stanford NER Tagger

https://allennlp.org/ - Allen NLP

https://pytorch.org/tutorials/ - Building PyTorch models

https://arxiv.org/abs/1706.03762 - Attention is all you need

https://radimrehurek.com/gensim/models/doc2vec.html - Gensim Doc2Vec paragraph embedding

https://pypi.org/project/biobert-embedding/ - BioBERT Embedding

https://scikit-learn.org/stable/modules/multiclass.html - Multi-class Classifiers