J Healthc Inform Res (2018) 2:25–43
https://doi.org/10.1007/s41666-018-0018-9

RESEARCH ARTICLE

Detecting Adverse Drug Reactions on Twitter with Convolutional Neural Networks and Word Embedding Features

Aaron J. Masino[1] · **Daniel Forsyth**[1] · **Alexander G. Fiks**[1]

Received: 12 May 2017 / Revised: 27 February 2018 / Accepted: 26 March 2018 /
Published online: 12 April 2018
© Springer International Publishing AG, part of Springer Nature 2018

Abstract Motivated by limitations of adverse drug reaction (ADR) detection in clinical trials and passive post-market drug safety surveillance systems, a number of researchers have examined social media data as a potential ADR information source. Twitter is a particularly attractive platform because it has a large, diverse user community. Two challenges faced in applying Twitter data are that ADR descriptions are infrequent relative to the overall number of user posts and human review of all posts is impractical. To address these challenges, we framed the ADR detection problem as a binary classification task, where our objective was to develop a computational method that can classify user posts, known as *tweets*, relative to the presence of an ADR description. We developed a convolutional neural network model (ConvNet) that processes tweets as represented by word vectors created using unsupervised learning on large datasets. The ConvNet model achieved an F1-score of 0.46 and sensitivity of 0.78 for tweet ADR classification on the test dataset, compared to 0.37 F1-score and 0.33 sensitivity obtained by two baseline support vector machine (SVM) models that incorporated word embedding, n-gram, and lexicon features. We attribute the superior ConvNet model performance to its ability to process arbitrary length inputs, which allows it to evaluate every word embedding in a given tweet and make better use of their semantic content as compared to the SVM models which require a fixed length, aggregated embedding input. The results presented demonstrate the feasibility of detection of infrequent ADR mentions in large-scale media data.

✉ Aaron J. Masino
masinoa@email.chop.edu

[1] Department of Biomedical and Health Informatics, The Children's Hospital of Philadelphia, Philadelphia, PA 19146, USA

Keywords Twitter · Social media · Neural networks · Machine learning · Pharmacovigilance

1 Introduction

A number of recent research studies have examined social media data as a potential resource to identify adverse drug reactions (ADRs) for post-market drug safety surveillance [1–10]. The motivation to use social media data is driven by limitations in the identification of ADRs in clinical trials and other pharmacovigilance methods. Clinical trials, which precede a drug's market release, are constrained by time, cost, and ethical concerns (e.g., testing on children) and therefore cannot identify all ADRs for a given medication, particularly rare ADRs [11–13]. Post-market passive reporting systems such as the Federal Drug Administration's Adverse Event Reporting System (FAERS) are known to be limited by delayed, biased, and under reporting of events [11, 14]. In contrast, social media data is frequently updated by large numbers of users who often describe their health status, including the occurrence of drug-related events, and may therefore provide a means for identifying previously unknown ADRs earlier than passive reporting system. Twitter is of particular interest because it serves a large, demographically diverse user base and allows free, programmatic access to user posts. Additionally, previous research has demonstrated a significant correlation ($p < 0.001$) between ADRs described on Twitter and those reported in FAERS [6] indicating that user ADR descriptions are a reliable data source for continued investigation of known ADRs and to prompt further investigation of previously unknown ADRs. As with any dataset, there are inherent limitations of Twitter data for the ADR detection task. In particular, individuals from various population segments (e.g., senior citizens) may be unlikely to use a social media platform. Individuals suffering from certain debilitating conditions (e.g., Alzheimer's) or privacy sensitive conditions (e.g., AIDS) may not be able or willing to report ADRs on a public platform. For these reasons, we view ADR identification via Twitter or other social media data as a hypothesis generation method that can supplement existing pharmacovigilance tools (e.g., FAERS). ADRs detected via social media data require subsequent validation via accepted methods such as electronic health record (EHR) analysis or clinical trials. Although it is possible to initiate ADR detection using these standard methods, both EHR data analysis and clinical trials are more time-consuming and costly than a potential automated analysis of Twitter data. Hence, it is potentially beneficial to use ADRs identified in Twitter to inform these analyses.

Although there is great potential, the scale of social media data, both in terms of the amount of data and the frequency with which it is updated, requires automated methods to identify and aggregate ADR descriptions. The earliest automation approaches utilized predefined drug and ADR lexicons and string matching methods [1–4, 6]. Lexicon methods are subject to a number of limitations. In particular, they are unable to differentiate adverse reactions from indications (the reason the drug was used). More importantly, ADR lexicon matching in social media is extremely difficult due to the highly variable linguistic characteristics employed in social media such as the use of informal, non-clinical event descriptions (e.g., "feeling like crap") that include frequent misspellings, irregular grammar (e.g., "dis aderall got me sweatin"), non-standard abbreviations (e.g., "lol"), and symbols (e.g., emoticons) that convey semantic information. More recently, researchers have treated ADR detection in social media as a machine learning problem to address these concerns. In this paradigm, the problem is most often framed as a sequence labeling task where the input text is tokenized (i.e., divided into its constituent parts, or tokens, that usually correspond to individual words) and each token is individually labeled relative to whether or not it is part of an ADR. For example, in the input "my Adderall is giving me stomach pain," the tokens "stomach" and "pain" should be labeled as belonging to an ADR, while the remaining tokens should be labeled as not belonging to an ADR. The most successful ADR sequence labeling efforts have employed conditional random field models [10] and very recently recurrent neural networks [15]. These models perform the ADR sequence labeling task very well when trained and evaluated on input text that is known to contain an ADR. However, they incur high false positive rates when processing input text that does not contain an ADR. This is a significant challenge to ADR detection in social media because most inputs do not describe an ADR.

We propose that a fully automated system to detect ADRs in social media sources such as Twitter may be constructed as a three-part pipeline: a classifier to identify inputs that contain an ADR, a sequence labeler to identify the specific tokens describing the ADR, and finally an aggregator that maps the ADR tokens to formal ontology concepts to support analysis. The first component in the pipeline is a classifier that determines if the input contains an ADR, without any determination of the specific tokens that compose the ADR description. The purpose of this component is to filter most of the inputs that do not contain an ADR description to address the high false positive errors made by the sequence labeling component. Previous work to develop such a classifier has relied on standard machine learning methods such as Naïve Bayes and support vector machines (SVMs) [16]. These methods are limited by the inability to process arbitrary length inputs, such as text. Therefore, it is necessary to perform feature engineering to create a fixed length representation of the input text that can be processed by the model. The feature engineering process is time-consuming and resource intensive and generally leads to information loss that limits performance.

Here, we investigate a convolutional neural network (CNN) model designed to classify Twitter user posts, known as *tweets*, relative to the presence of an ADR and that could serve as the first component in an ADR detection pipeline. CNN models have been used in a variety of applications since their development more than twenty years ago [17]; however, to our knowledge, this is the first time they have been used to detect adverse drug events in free text, such as social media. Our CNN model closely follows known sentence classification models based on CNN architectures [18, 19]. Importantly, the model includes a pooling mechanism that allows it to process arbitrary length inputs, which is highly advantageous for text classification. Input tweets are represented by a matrix whose rows are real-valued vectors, known as a word embeddings, that are a continuous representation of the word in the tweet. The word embeddings are formed from large, unlabeled datasets and capture latent semantic information that has been shown to be useful in many natural language processing tasks [20, 21]. As the embedding representations are the only model input, we avoid task-specific feature engineering. We evaluate the performance of our CNN model on the ADR classification task for Twitter data and compare to SVMs using both engineered features and an averaged word embedding input.

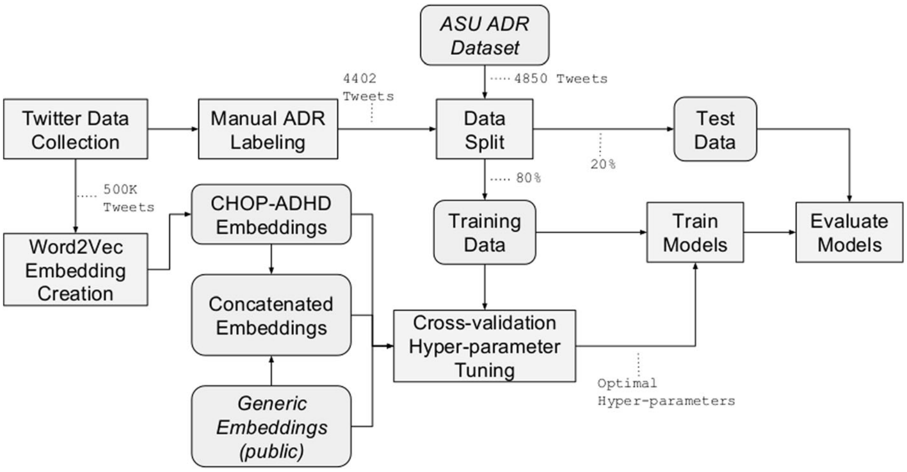

Fig. 1 Experimental workflow from data collection to model evaluation

2 Materials and Methods

This study was determined to exempt from IRB review by The Children's Hospital of Philadelphia (Fig. 1).

2.1 Data Collection and Labeling

We combined two Twitter datasets in our study. We collected the first dataset, denoted as the *CHOP ADHD Dataset*, between May 1st and December 31st, 2015, using the Twitter streaming API. We collected user posts (called tweets) containing at least one search keyword corresponding to a list of 44 brand and generic drug names of medications commonly used to treat attention deficit hyperactivity disorder (ADHD). To account for user misspellings, we also included phonetically similar versions of each drug name using the approach described by Pimpalkhute et al. [22] The final search keyword list contained 166 terms. We removed all *retweets* (user posts that are re-posted by another user) and any tweet containing a URL as they are mainly advertisements. From these, we randomly sampled 4402 tweets,[1] which were labeled as positive or negative for containing a primary or secondary report of a specific adverse drug reaction. Two reviewers labeled each tweet. For event positives, the words in the text span containing the event were also labeled. Tweets that contained an adverse event description that was not associated with a drug were labeled as ADR negative. Two authors independently labeled all tweets. We performed consensus reconciliation of discordant assessment until 100% agreement was achieved. Of the labeled tweets in the *CHOP ADHD Dataset*, 278 (6.32%) were identified as containing an ADR.

The second dataset, denoted as the *ASU ADR Dataset*, was previously published and made publicly available [10]. The *ASU ADR Dataset* contains tweets with at least one keyword corresponding to a list of 74 brand and generic drugs from the IMS Health's

[1] We originally sampled 5000 tweets. However, we found that one of our search terms, liquado, a phonetic misspelling of the drug LiquADD, is also a common misspelling of licuado, a blended beverage similar to a smoothie. We therefore eliminated these samples.

Top 100 drugs by volume for the year 2013. Per the Twitter developer agreement, authors may publish tweet IDs but may *not* publish the text for tweets obtained from the Twitter APIs. The authors of the *ASU ADR Dataset* released annotations and tweet IDs for 7574 tweets, of which only 4850 were available through the Twitter API at the time of this study (Twitter does not store tweets indefinitely). Of the 4850 annotated tweets, 559 (11.51%) were labeled as containing an ADR and 4291 (88.49%) were labeled as containing no ADR.

We combined the *CHOP ADHD Dataset* and the *ASU ADR Dataset* to form our complete dataset for this study. We split the complete dataset into stratified training (80%) and test (20%) sets. The training set contains 7401 tweets with 669 (9%) containing an ADR and 6732 (91%) labeled as containing no ADR. The test set contains 1851 tweets with 168 (9%) labeled as containing an ADR and 1683 (91%) labeled as no ADR. The labeled ADHD dataset is available at https://github.com/chop-dbhi/twitter-adr-blstm.

2.2 ConvNet Model Development

We developed our *convolutional neural network* (*ConvNet*) model based on previously published models for sentence classification [18, 19]. The model is called convolutional because it applies an approximation to the mathematical convolution operator using a collection of kernels (two-dimensional numerical matrices) that represent local features. The outputs of the kernels are subsequently used to classify the input tweet relative to the presence of an ADR. We implemented and trained the ConvNet model using version 0.8.0 of the TensorFlow library [23]. The source code for our ConvNet model implementation and training is available at https://github.com/chop-dbhi/twitter-adr-convnet.

2.2.1 Word Embedding Features

Prior to model development, it is necessary to derive a numerical representation of each tweet to use as model input. For the ConvNet model, we utilized word embedding representations to create the input. A word embedding is a vector of real values (i.e., decimal numbers) that represent a given word. The word embeddings are maintained in an M X N embedding matrix where M is the dimensionality of the word representation (i.e., the number of decimal values used to represent the word) and N is the number of words in the vocabulary. Note that M is the same for all words. Each column of the embedding matrix corresponds to a single word. To create an input for a tweet, we first tokenize the tweet (i.e., divide it into its constituent tokens where each token is usually a word) to create a token list that maintains the order the tokens appear in the tweet. We then form a $D \times M$ matrix representation of the tweet, where D is the number of tokens (words) in the tweet. Thus, each row in the matrix represents a single word in the tweet. The row order is determined by the word order in the tweet, i.e., the j^{th} word in the tweet corresponds to the j^{th} row. The row values are copied from the column in the embedding matrix that corresponds to the word. If a particular word is not in the embedding matrix, then it is ignored. This process creates variable length inputs (i.e., D is not the same for all tweets). Most machine learning models require fixed length inputs. The ConvNet model can, in principle, accept variable length inputs because the pooling layers (described below) effectively collapse arbitrary length inputs to a fixed size. However, for implementation convenience, we converted all training tweets to a fixed length equal to the number of tokens in the longest tweet in the training set by adding <*PAD*> tokens to tweets shorter than the longest. In the same manner, we converted all test tweets to a fixed length equal to the number of tokens in the longest test tweet. Note, the <*PAD*> token is represented by a zero vector of length M. The effect of such zero padding is model dependent. In our ConvNet model, zero padding only serves to inform the model that a given word is near the sentence boundary. Consider that we first convolve a given filter across the input windows (as determined by the filter size and stride), including possible padding. This yields W outputs for the given kernel, where W is the number of windows. We then apply max pooling, i.e., we take the maximum of the W values and zero, denoted as w_{max}. The minimum possible value of w_{max} is therefore zero. Assuming a given kernel is convolved with a region containing zero padding and noting that the convolution of an arbitrary kernel with a zero vector is equal to zero, there are two possible outcomes for this process: (1) w_{max} is greater than zero indicating the convolution is positive for some region of the input which is necessarily *not* composed of only padding values or (2) w_{max} is exactly zero indicating that the convolution is non-positive for all regions of the input. In the first case, the only potential impact of the zero padding is to cancel the effect of the portion of the kernel that is convolved with the padding. This provides the intended effect of informing the system of sentence boundaries. In second case, the zero padding effectively clips the negative region of the convolution output to be zero. However, because we apply rectified linear activation units (which evaluate to zero for all non-positive inputs) to the w_{max} values, there is no net effect in this second case.

We evaluated three versions of word embeddings. The first, denoted as the *Generic Embeddings*, is a published dataset developed from 400 million non-domain specific tweets for which each word is represented by 400 numerical values [24]. We developed a second set of word embeddings, denoted as the *ADHD Embeddings* using approximately 500,000 unlabeled tweets from our *CHOP ADHD Dataset*. We generated the word embeddings using the skip-gram model [25] architecture as implemented in the Gensim Python library [26]. The word embedding dimension was set to 250 as determined through a cross-validation grid search in which we considered values of between 50 and 400 in increments of 50. Finally, we considered a concatenation of these models in which each word is represented by 650 values where the first 400 values for the word are from the first dataset and remaining 250 values are from the second dataset. For words present in only one of the two datasets, a (250 or 400) zero vector was concatenated to the available (400 or 250) values from the other dataset. The concatenated word embeddings can be viewed as a multi-channel input similar to the different color channels in an image file. In the case of word embeddings, the multiple channels correspond to possibly different semantics for a given word as described by previous investigators [18].

2.2.2 Convolutional Neural Network Model Architecture

Our ConvNet model produces a binary label that indicates the presence or absence of an ADR in the input tweet. The input tweet is first converted to a $D \times M$ matrix where each row corresponds to a word representation of length M as described above. The model includes K kernels, where each kernel is represented by a $s_\kappa \times M$ matrix where $s_\kappa \leq D$, noting that s_κ may vary for $\kappa \in [1, K]$. Each kernel is *convolved* with the tweet input to form a vector h_i. The convolution process involves computing the dot product of the kernel with an aligned element in a window of the input matrix. As the kernel has fewer rows than the input, this process is repeated for each window of the input that is the same size as the kernel. The number of windows depends on the kernel size, s_κ, the stride, s, which is the number of rows the kernel is advanced along the input before the next convolution, and the presence of padding (the addition of zeros around the border of the input matrix). In this work, we use a stride of one, meaning that the filter is applied to a moving window that is stepped forward one row at a time. We do not apply padding (commonly called *valid padding*). The resultant output of a given kernel convolved with the input is a vector, $\boldsymbol{c}_\kappa$, of length $D - s_\kappa + 1$. To each element of this vector, we add a bias term and then apply a rectified linear activation function which has been shown to simplify neural network training [27, 28]. A max pooling layer is then applied in which the maximum value of each $\boldsymbol{c}_\kappa$ is retained, yielding an output vector, **h,** of length K, the number of kernels. This vector is finally passed to a fully connected softmax layer to compute the class probabilities, i.e., the probability of the tweet containing or not an ADR mention. The architecture is illustrated in Fig. 2. The inputs, output equations, and parameters to be learned are detailed in Table 1.

2.2.3 Training

We trained the ConvNet model using back-propagation and batch stochastic gradient descent to estimate the gradient of the cross-entropy error function and update the model parameters (i.e., the kernel weights and the weights in the softmax layer). The iterative process of error derivative estimation and parameter updates was conducted over several epochs (i.e., one training epoch is the error derivative and weight updates done per batch for all batches derived from the entire training set so that for multiple

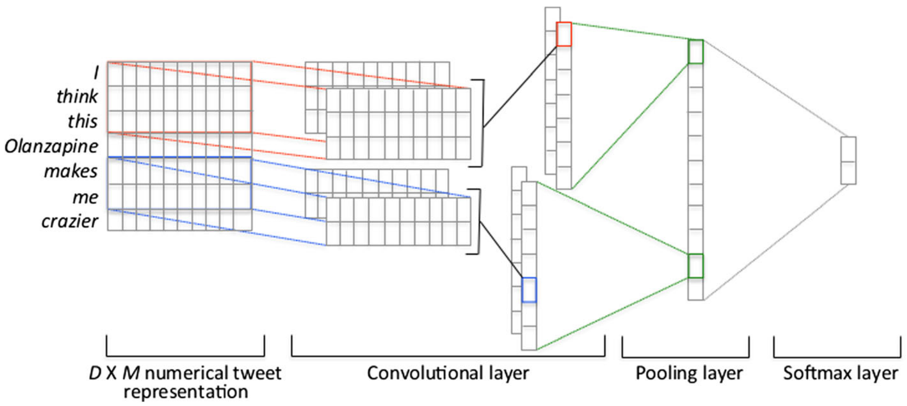

Fig. 2 ConvNet architecture. The input tweet is represented as a $D \times M$ matrix, where M is the length of the word vector representations. A collection of K kernels of size $k_i \times M$ (k_i equal to 2 and 3 in this illustration with multiple filters for each filter width) are convolved with the input to produce K outputs each of size $D - k_i + 1$. In the max pooling layer, the maximum of each convolution is collected into a single vector which is the input to a fully connected softmax layer used to compute the log-probability of the presence of an ADR in the input tweet

Table 1 ConvNet input, output, and parameters by layer. In all cases, κ indicates the kernel index. Note there are K output vectors, $\boldsymbol{c}_\kappa$, and weight matrices, $\boldsymbol{W}_\kappa$, for the convolution layer. The max function yields K values, one for each output vector, $\boldsymbol{c}_\kappa$. The parameters D, M, and s_κ indicate the number of words in the input tweet, the word embedding length, and the size of the κ^{th} kernel, respectively. The function, f, in the convolution layer is a non-linear activation function (a rectified linear function in this study)

Layer	Input	Output	Parameters to be learned
Convolution	$\boldsymbol{x} \in \mathbb{R}^{D \times M}$	$\boldsymbol{c}_\kappa = f(\boldsymbol{W}_\kappa \cdot x_{i:i+s_\kappa} + b_\kappa)$ where $i = 1, \cdots, D - s_\kappa + 1$ b_κ is the κ^{th} element of $\boldsymbol{b}$	$\boldsymbol{W}_\kappa \in \mathbb{R}^{s_\kappa \times D}$ $\boldsymbol{b} \in \mathbb{R}^{\boldsymbol{K}}$
Max Pooling	$\boldsymbol{c}_\kappa$	$\boldsymbol{h} = max\, \boldsymbol{c}_\kappa$	None
Softmax	$\boldsymbol{h}$	$\boldsymbol{C} = \boldsymbol{W}_{sm} \cdot \boldsymbol{h} + \boldsymbol{b_{sm}}$	$\boldsymbol{W}_{sm} \in \mathbb{R}^{2 \times K}$ $\boldsymbol{b} \in \mathbb{R}^{\boldsymbol{2}}$

epochs, the entire training dataset is processed multiple times). We applied dropout regularization [29, 30] as a means for controlling model variance (i.e., overfitting). For the ConvNet model, we implemented this by first computing the convolution and performing the max pooling operation to generate the input vector of size K to the fully connected softmax layer. Prior to computing the softmax layer output, we generated a uniform random value in the range zero to one for each of the K elements in the input vector. If the random value was above the *dropout keep probability*, p, the input element was set to zero. Note, this is the same as removing the convolution kernel that corresponds to that element for the given training sample. Additionally, we used oversampling of positive cases (i.e., samples containing an ADR) to address class imbalance during training. Specifically, we formed each batch by randomly selecting half of the batch from the available ADR positive samples and half of the batch from the negative samples so that each training batch was balanced.

We conducted training in two phases. The first phase was used to select model hyperparameters such as the number and size of the kernels, the number of training epochs, and the dropout keep probability. We performed a grid search over the parameter combinations for a range of variables and evaluated each combination using a *k-fold* cross-validation, with $k = 5$. In all, we evaluated 972 parameter combinations as detailed in Table 2. Once the optimal parameters were identified, we trained the model on the entire training set. The complete training process required approximately 270 h on a compute server with 72 2.3 GHz Intel Xeon CPUs and 1 TB RAM.

2.3 Baseline Models

We implemented two baseline methods for comparison with the ConvNet model: a support vector machine (SVM) classifier using n-gram and lexicon features and an SVM using word embedding features. As with the ConvNet model, it was necessary to address class imbalance during training. Unlike the ConvNet model, the SVM training procedure does not use batches with multiple epochs over the training set so that we could not balance the training batches as was done when training the ConvNet model. Thus, we applied a bootstrap method. Specifically, we trained each of the SVM models with the available, unbalanced training data using a grid search and *k-fold* cross-

Table 2 Hyperparameters evaluated for ConvNet model tuning. The full cross-product of all variables was evaluated, resulting in 972 parameter combinations

Hyperparameter	Range evaluated
Filter width sets	{2,3,4}, {3,4,5}, {2,3,4,5}
Maps per filter width	25, 50, 100
Dropout keep probability	0.35, 0.5, 0.65
Number of training epochs	10, 20, 30,
Ratio of positive to negative samples in each batch	0.099 (no over sampling), 0.6, 1, 1.5,
Word embeddings	Generic, CHOP ADHD, concatenated

validation ($k = 5$) to determine the optimal model tuning parameters. We then took random samples of 5000 tweets from our unlabeled data (unlabeled tweets collected when forming *CHOPADHD Dataset*) and obtained predictions from each SVM model. Tweets predicted as ADR positive were added to the training data. We repeated this process iteratively until cross-validation accuracy no longer increased (five iterations) yielding a new training set with 21% ADR positive samples (up from 9%). The final SVM models were trained with this boosted training set and evaluated on the held out test set.

2.3.1 SVM with N-gram and Lexicon Features (SVM-1)

Our first baseline model, denoted as SVM-1, is a support vector machine (SVM) classifier with input that includes four feature types inspired by previously published work on ADR detection in social media text [16]. The first feature set is a collection of n-gram features (specifically unigram, bigram, and trigram) associated with each tweet. We derived these features by identifying all unigrams, bigrams, and trigrams in the training data to form an overall n-gram dictionary of size 158,835 terms. For a given input tweet, a vector represents the tweet in which the value at the index of a specific n-gram feature is set to one if that n-gram is present in the tweet and zero if it is not present. The remaining features were based on matching concepts in a lexicon. The lexicon contains 13,014 concepts and is a combination of a previously published ADR lexicon [10] and selected concepts from the Consumer Health Vocabulary[2] (CHF) that corresponded to adverse events that are associated with commonly used ADHD medications as derived from the NICHQ Vanderbilt Assessment Follow-up—PARENT informant form. These features include a binary indication of whether an ADR was present in the tweet, the number of lexicon hits and the sum of the hit similarity match scores per tweet (produced by the Apache Lucene[3] software package), where the latter two features were standardized to have zero mean and unit variance. To identify lexicon matches, we created an Apache Lucene index from the ADR lexicon concept terms and queried the index for concept matches within each tweet, i.e., each tweet was used as a query to the index. The result of each index query, one per tweet, is a rank ordered list of lexicon concepts that correspond to ADRs where the ranking is determined by the

[2] http://consumerhealthvocab.org
[3] https://lucene.apache.org/

Lucene scoring function. We used the *DefaultSimilarity* option in Lucene which is a variant of Tf-Idf scoring. Although a complete description of the hit similarity computation is outside the scope of this work, we note that, in addition to several other factors, it is a function of the term frequency, inverse document frequency, and the number of terms in the index that were found in the tweet.[4] Our additional lexicon terms and model implementation are available at https://github.com/chop-dbhi/twitter-adr-lexicon [15].

2.3.2 SVM with Word Embedding Features (SVM-2)

The second baseline model, denoted as SVM-2, also used a SVM method, but with only word embeddings as input features. We considered both the *Generic Embeddings* and the *ADHD Embeddings* described previously. A challenge to using word embedding inputs for an SVM model is that the SVM model requires a fixed length input vector. One approach to address this issue would be to set all tweets to some fixed length, *say N*, either by padding or cropping each tweet to the appropriate length. This would produce input vectors of size *N* times the length of the word embedding dimension. Assuming *N* is at least 10, this would yield input vectors with O(1000) values per tweet for our model. Unlike the ConvNet model, there is no parameter sharing in the SVM model, and therefore such a large input vector would likely result in severe overfitting. Instead, we addressed this issue using an input vector with dimension equal to that of a single word embedding. To generate the input vector representation for a given tweet, we first initialized a vector of zeros of the same length as a single word embedding (400, 250, and 650 for the *Generic Embeddings*, the *ADHD Embeddings*, and the concatenated embeddings, respectively). We then iterated through each word in the tweet. If a word embedding was available for the word, it was added element wise to the current input vector. The tweet input vector was then divided by the number of words for which an embedding existed. As a final model, we considered an input tweet representation that combined the *Generic Embeddings* and the *ADHD Embeddings*. In this model, a 650 dimensional vector represented the tweet. For each word in the tweet, if the word had an embedding available in both embedding sources, the two vectors were concatenated and added to the current tweet vector. If a word was found in one embedding source but not the other, the available vector was concatenated with a zero vector the length of the missing vector. As before, the final vector representation averaged over the number of words in the tweet for which at least one embedding was available.

3 Results

We evaluated 243 hyperparameter combinations through fivefold cross-validation. The optimal parameters for the ConvNet model architecture were *Generic Embeddings*, 30 training epochs, kernel sizes of 2, 3, and 4 (kernel size refers to the number of words spanned by the kernel), 50 kernels per kernel size, and a dropout keep probability of

[4] A complete description of Lucene scoring is available at http://www.lucenetutorial.com/advanced-topics/scoring.html.

0.5. The cross-validation F1-score had a narrow range of (0.44, 0.5) across all parameter combinations. This suggests the model is reasonably robust to the tuning parameters, at least for the particular task and dataset presented here. For the SVM-2 baseline model, cross-validation revealed that the *ADHD Embedding*-derived inputs provided the best performance.

Using the optimal hyperparameters, we evaluated the ConvNet and SVM models on the test set, which contains 1851 tweets (168 with an ADR mention). Note that items in the test set were never used in training. The parameters learned by the ConvNet model can be affected by the random parameter initialization (i.e., the kernel values and softmax layer weights) and the random selection of tweet batches during training. We addressed this concern by performing ten training and evaluation runs. Specifically, we randomly initialized the learned parameters, trained the model as described previously, and then evaluated on the test set. We repeated this process ten times to obtain statistical measures (mean and confidence intervals) of the performance parameters for the ConvNet model on the test set. The performance results are provided in Table 3. As indicated, the average ConvNet model performance is nearly 25% better in terms of F1-score as compared to the either of the SVM models. We assessed the statistical significance of these F1-score differences by McNemar's test using exact binomial probability calculations [31] for which the maximum p value between the ConvNet model and either of the SVM model predictions over the ten ConvNet training and evaluation runs was 4.6e-13 indicating a clear statistical significance in the model predictions. The superior performance of the ConvNet model relative to the SVM models is attributable to a 133% increase in sensitivity with only a 25% loss in precision. Noting that only 9.1% of the test tweets contain an ADR description, this result suggests that the ConvNet model is better at detecting the few tweets that contain ADR descriptions, while maintaining nearly the same ability as the SVM model to control false positives.

We found that the methods used for addressing class imbalance for both the ConvNet (oversampling) and SVM (boosting) models were of critical importance. For comparison, we evaluated model cross-validation performance with the original unbalanced training datasets. For our ConvNet model, this resulted in randomly selected training batches whose average class distribution is approximately that of the overall imbalanced dataset. The best performance (taken over the grid of the remaining hyperparameters) corresponded to cross-validation average metrics of F1-score, 0.204; precision, 0.748; and sensitivity, 0.119. Similarly, for the SVM-2 model, we evaluated

Table 3 Test set performance metrics. SVM-1 indicates the support vector machine model trained with n-gram 1-hot vectors and lexicon features. SVM-2 indicates the support vector machine model trained with inputs derived from the *ADHD Embeddings*. For the convolutional neural network (ConvNet) model, the mean and 95% confidence intervals over ten training and evaluation runs are shown

Model	Accuracy	F1-score	Precision	Sensitivity
ConvNet	82.0% (77.5–86.5%)	*0.457* (0.419, 0.495)	0.337 (0.288, 0.386)	*0.780* (0.716, 0.845)
SVM-1	*90.3%*	0.366	*0.448*	0.310
SVM-2	89.5%	0.365	0.403	0.333

Values in italics indicate the best performance for a given metric cross-validation performance without boosted samples and obtained average metrics of F1-score, 0.261; precision, 0.692; and sensitivity, 0.161. For both models, the results are significantly lower than the test set performance indicated in Table 3 where oversampling was used. We also evaluated the ConvNet model with the boosted data samples used to train the SVM model. First, we evaluated performance by adding the boosted samples and maintaining the resulting positive to negative class sample ratio of 0.166 in the training batches. This setting is the closest match to the boosted training method used to train the SVM model. We found that the average cross-validation performance was poor compared to the setting in which training batches were nearly balanced, suggesting that the ConvNet model is more sensitive to class imbalance than the SVM models. We also evaluated the ConvNet performance using the boosted samples while balancing the class ratio as before. In this case, the performance showed no statistically significant difference as compared to not using the boosted samples when balancing the training batches.

We also considered an ensemble approach for the ConvNet model, in which the predicted class (ADR present or not) was determined by the majority vote of the ten individually trained models (all models had the same architecture and training samples, but were initialized with different random weights at the start of training). In the event of a tie, the tweet was considered to be negative for an ADR. This produced slightly better results than the average performance of the individual models: 84% accuracy, 0.476 F1-score, 0.338 precision, and 0.804 sensitivity. Using the ensemble model, we can also evaluate the Receiver Operating Characteristic (ROC) curve and the Precision-Recall (PR) curve of the ConvNet model. An ROC curve illustrates the tradeoff between controlling false positives and negatives through the dependence between sensitivity and precision as a function of the classification decision threshold. The ROC area under the curve (AUC) is a point metric that is often used to assess model performance. Similarly, the PR curve illustrates the balance between sensitivity (aka recall) and precision and is often more informative for imbalanced data such as our test set. The ROC and PR curves of the ConvNet and SVM-2 models are shown in Fig. 3 and Fig. 4, respectively. The ROC curves indicate the ensemble ConvNet model is better able to simultaneously control false positives and negatives relative to the SVM models. Further, it may be tuned to better control false positives (i.e., improve precision) without drastically reducing sensitivity by altering the positive decision probability threshold from the default of 0.5. For example, if the decision threshold for the ADR class is set to 0.65 for the ensemble model, the F1-score increases to 0.522.[5] This increase in F1-score results from an increased precision (0.445) but with decreased sensitivity (0.631). This result is also reflected in the PR curve of Fig. 4. This suggests the decision threshold can be adjusted to control precision and sensitivity as warranted by a particular application. Closer analysis of that figure, however, indicates that this tradeoff has significant limitations. First, the only points with precision above 0.6 are those with less than 0.1 recall. Second, the rate of decrease of precision increases as recall increases. In particular, for recall values above 0.7, the precision falls of quickly to below 0.2. There is also a nearly flat region for recall between 0.2 and

[5] The optimal threshold in the given example was selected using the test set. Results are for illustrative purposes only. As with any model tuning parameter, selection of a threshold that yields generalizable performance would require a cross-validation study.

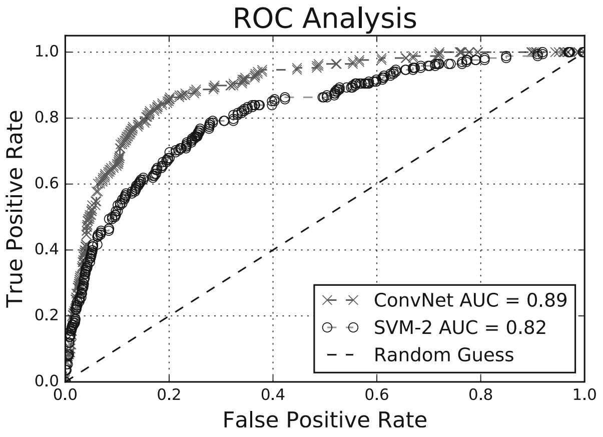

Fig. 3 The Receiver Operating Characteristic (ROC) and ROC area under the curve (AUC) for ConvNet and SVM-2 models. The dashed line is the expected performance of a random binary classifier. AUC values closer to 1.0 indicate high performance with low false positive and false negative rates

0.6 where precision remains above 0.5 which is encouraging as it suggests robustness of the model in this region.

4 Discussion

4.1 Learning Curve Analysis

With any machine learning model, there is concern about overfitting to the data, that is the model performance sensitivity to training data variance. An important factor that influences overfitting is the number of parameters learned in the model relative to the training sample size. Our best performing ConvNet model contains 112,801 learning parameters. This is a relatively large number of parameters as compared to the training sample size. To address this concern, we implemented dropout regularization which is

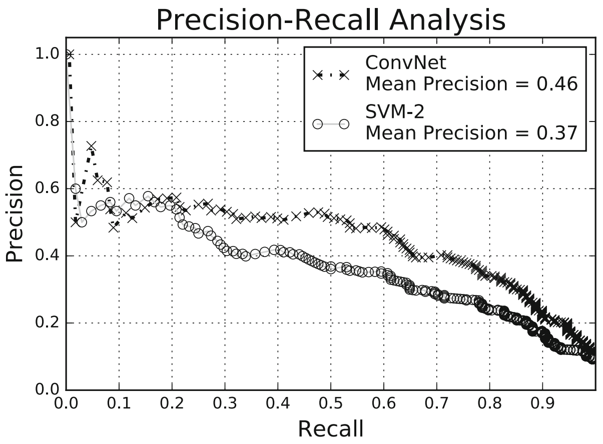

Fig. 4 The Precision-Recall and Mean Precision for ConvNet and SVM-2 models. Average precision values closer to 1.0 indicate better performance

intended to counter potential overfitting. Nonetheless, overfitting remains a concern. To evaluate overfitting in our models, we now consider *learning curve* plots. A learning curve is a plot of a selected performance metric as a function of the number of training samples. Two curves are generated, one each for the training and validation sets. In the absence of bias (insufficient model complexity) and variance (sensitivity to the data sample), both the training and validation curves will approach the optimum metric value as the training set size increases. In the presence of variance, the training curve will approach the optimum value, however the validation curve will not. In the presence of bias, the training and validation curves will fail to approach the optimum value. Learning curves for the ConvNet and SVM-2 models are shown in Fig. 5. For the ConvNet model, the learning curve indicates variance is present as evidenced by the separation between the training and cross-validation curves in the limit of large training set size. Recall that for the ConvNet model, the available training examples are repeated across training epochs. In the learning curve of Fig. 5, the ConvNet (left side) performance does not plateau until after 1000 training batches, at which point it has been exposed to the complete training set nearly nine times (i.e., 1000 batches is 8.7 epochs). Therefore, it is reasonable to infer that variance may be reduced with additional labeled training samples. The learning curve also suggests the presence of bias indicated by the failure of both the training and validation curves to approach the optimal loss value of zero. This is likely due to the fact that the only model inputs are the word embeddings with finite-sized convolution kernels. These features fail to capture information between words that are separated by distances longer than the kernel width, where in this study, the maximum width was four. It is possible that additional features such as part-of-speech tags and lexicon information may reduce bias for this model.

4.2 Error Analysis

The ConvNet model had relatively low precision on test set (see Table 3), implying that false positive errors had the strongest impact on overall model performance. We

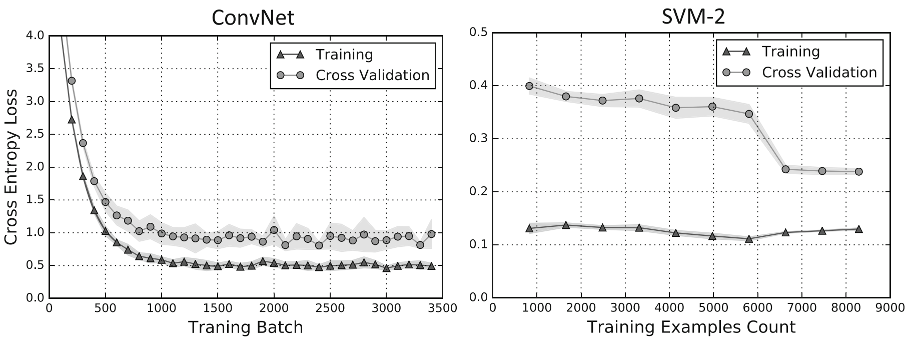

Fig. 5 Learning curves for ConvNet and SVM-2 models. The performance metric, *cross entropy loss*, has an optimal value of zero. For the ConvNet model (left), each training batch contains 64 tweets (115 training batches is approximately one epoch, i.e., 7360 training examples). Note that for the ConvNet model, training samples are repeated across epochs. Performance is evaluated by fivefold cross-validation. The symbols indicate the mean value over the fivefolds and the shaded region indicates one standard deviation. Behavior for the SVM-1 model (not shown) is similar to that of the SVM-2 model

qualitatively examined the ConvNet model false positive errors to better understand the nature of these errors (see Table 4). The most frequent type of false positive error (38.9%) can be attributed to the presence of a symptom description or ADR-related word in the tweet that is not directly attributed to the drug or to a specific individual. Another important and interesting category of false positive error occurs for tweets that contain vague event descriptions (11.1%). The tweets associated with these errors contain descriptions of events that are attributed to the drug but are not easily defined in a clinical sense. Errors due to the misrecognition of indications were also common (10.3). In total, these error sources account for 60% of all false positive errors. In all such cases, it is unlikely that word features alone, as used by the ConvNet model, are sufficient to distinguish from ADR positive cases, again implying that additional features are necessary to improve model performance.

4.3 Limitations

The characteristics of our Twitter dataset represent the most significant limitation of this study. In particular, the labeled tweet dataset is relatively small and therefore the presented results may not generalize to other drug and ADR types described in Twitter posts. It is possible that there are semantic differences between the ADRs mentioned in our dataset and other rare ADRs or with ADRs associated with drugs that were not represented in our data. Similarly, the results may not generalize to other data sources, particularly those with different linguistic characteristics. The unique linguistic style of tweets, driven by the restriction to 140 characters and use of vernacular, strongly differentiate the data from that of other social media sites or from formal sources such as biomedical literature and

Table 4 Author's classification of false positive errors made by the ConvNet model. The **bold text** in the Example column indicates the portion of the tweet that matches the error classification category

False positive classification category	Example	Percent of false positive errors
Symptom description/ ADR-related word	*maybe its just me but i dont think its really "progression" if you **lose weight** from taking adderal or vyvanse.*	38.9
Vague event description	*10 gs of focalin **got me organizing my thoughts** in alphabetical order.*	11.1
Indication	*@MENTION have you tried any? I take Effexor for my **anxiety**, it works really well.*	10.3
Negated reaction	*@MENTION i've cut my son's intake of carbs by half and he seems more awake and **doesn't gain weight** like he used to. methylphenidate slow*	1.2
Drug benefit	*@MENTION not what you think, I don't have money for that. It's my adderal & Paxil **making me happy**.*	2.4
Other	*@MENTION Yes, it is so frustrating. I'm already on the highest dose of Effexor and taking a relatively low dose of Lamictal.*	36.1

clinical notes. To further determine the generalizability of the presented method to data sources with significantly different linguistic styles requires that we first formulate word embeddings derived from the new source (e.g., clinical notes). Following that, it would be necessary to label a data sample relative to the indication of ADRs and then train and evaluate the model. Fortunately, relevant datasets exist (e.g., Multiparameter Intelligent Monitoring in Intensive Care [MIMIC] III [32]), and this is a focus of our continuing research.

Another concern is the number of words in the training and test sets that do not appear in the word embedding data. We identified 2756 words (32.6% of all words) in the training and test tweets that did not appear in either the *Generic Embeddings* or the *ADHD Embeddings*. This occurs whenever a given word in the training or test data is not in any tweet in the corpus used to form the embeddings. In a cursory analysis, we found that nearly all of the missing words were usernames, hash tagged phrases (e.g., #singlepayernow), or misspellings. It is possible that hash tagged phrases and misspelled words could provide additional information that would have boosted model performance. As a possible area of future research, we hypothesize that it may be possible to reduce the frequency of missing words by attempting to break hash tagged phrases into the constituent words and by incorporating a spell checker for unmatched words, thereby adding more useable information to the inputs.

In addition to the technical difficulties noted above, there are additional limitations associated with Twitter data for ADR analysis. The data provided by the Twitter platform does not include user demographic information such as age, gender, or geographic location. In the data we collected, users did not typically provide such data in their public profile. Additionally, user posts did not typically provide information on disease or comorbidity history. This may be due to the 140 character constraint for Twitter posts or it may reflect a reluctance to share such information. Whatever the reason, the lack of such information prevents analysis that may reveal deeper knowledge of ADRs based on such context. We plan to explore methods that may overcome this limitation with an interactive information system that contacts users to request more detailed information when a potential ADR is detected in Twitter data.

5 Conclusion

Our results indicate that convolutional neural network models that use pretrained word embeddings as the only input features can effectively classify text from social media, specifically Twitter, relative to the presence of an ADR. The results indicate statistically significant superior F1-scores and sensitivity compared to standard baseline methods, which reflects the success of this approach in other natural language processing tasks.

An important aspect to the success of the ConvNet model was the use of word embeddings that encode semantic information derived from a dataset that is much larger than the labeled set used to train the classifier. The semantic information manifests through the formation of word embedding vectors that are close to each other in the embedding space for words that share similar meaning. Importantly, the *Generic Embeddings* were formed from a very large set of tweets that capture much of the lay terminology used to describe adverse events in user tweets. This implies that the ConvNet model may recognize ADR related-words (or phrases) in the test set that were not present in the training set so long as the word embedding representations are similar. This occurs because the kernel parameters learned during training to detect a given ADR word (or phrase) present in the training set produce a similar convolution output for semantically similar words (or phrases) in the test set assuming the words have nearly equivalent embeddings. This result is aligned with similar results in so-called transfer learning [33]. This phenomenon is highly advantageous because such word embeddings can be created using unsupervised learning methods that easily scale to large, unlabeled datasets which are usually readily available at low cost. As these embeddings are the only features used by our model, we avoid requirements for task-specific feature engineering thereby achieving simplicity over systems requiring expert-engineered input features that are typically difficult and expensive to develop. Finally, we contend that the ConvNet model is better able to leverage the semantic information encoded in the word embeddings than other standard models as evidenced by the superior performance relative to an SVM model that also used word embedding inputs. This is likely due to the fact that, through the convolution process combined with pooling, the ConvNet model is able to process arbitrarily sized inputs. This is in contrast to most machine learning models, including SVMs, which require fixed length inputs. In practical terms, when using word embeddings, this constraint forces the SVM input to either be very large (O(10,000) in this study, i.e., the number of words in the largest tweet multiplied by the word embedding length) or to be formed as an aggregation of the embeddings (as with the average embedding in this study). The former is likely to cause severe overfitting (variance) due to the large number of parameters to be learned, which is avoided in the ConvNet model by weight sharing, i.e., the kernel weights are reused across the entire input as opposed to having a parameter for each input dimension. In the latter case of aggregation, there is information loss in the input and reduced model capacity.

As a long-term objective, we seek to create an automated system that aggregates information on the occurrence of ADR reports in large-scale social media data. Such an automated system must identify the specific spans (a group of words) within an input text that describe an ADR and map such spans to known ontology vocabulary terms. Current state-of-the-art methods for identifying the ADR-specific spans, including our recent work [15], perform well when presented with an input text that is known to contain an ADR span; however, performance suffers from high false positive rates (i.e., incorrect labeling of spans as ADRs) when the input includes text without an ADR. The ConvNet results indicate that it is feasible to classify tweets relative to the presence of an ADR at scale with good sensitivity, which is important for recognizing rare ADRs, while maintaining an acceptable specificity. The ConvNet model may therefore serve as the first component in an automated system, wherein the text input is first classified relative to ADR presence, and the text span-labeling model processes only ADR positive tweets. The remaining step of mapping ADR text spans to known ontology terms has received little attention for ADR detection in social media and is an area of exploration for our future work.

Author Contributions AJM and AGF conceived of and designed the study. AJM, DF, and AGF conducted data labeling and review. AJM and DF developed the model. AM and DF implemented the models. AJM and DF analyzed model performance. AJM wrote the manuscript. All authors contributed to the review and revisions of the manuscript. All the authors have seen and approved the final version of the manuscript.FundingThe Leonard David Institute at the University of Pennsylvania supported this work.

Compliance with Ethical Standards

Conflict of Interests AGF received an independent research grant from Pfizer, which provided salary support for his research team for work unrelated to this project.

References

1. Leaman R, Wojtulewicz L, Sullivan R, Skariah A, Yang J, Gonzalez G (2010) Towards Internet-age pharmacovigilance: extracting adverse drug reactions from user posts to health-related social networks. In: Proc. 2010 Work. Biomed. Nat. Lang. Process., Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 117–125. http://dl.acm.org/citation.cfm?id=1869961.1869976
2. Benton A, Ungar L, Hill S, Hennessy S, Mao J, Chung A, Leonard CE, Holmes JH (2011) Identifying potential adverse effects using the web: a new approach to medical hypothesis generation. J Biomed Inform 44:989–996. https://doi.org/10.1016/j.jbi.2011.07.005
3. Yang CC, Jiang L, Yang H, Tang X (2012) Detecting signals of adverse drug reactions from health consumer contributed content in social media. Proc ACM SIGKDD Work Heal Informatics
4. Yates A, Goharian N (2013) ADRTrace: detecting expected and unexpected adverse drug reactions from user reviews on social media sites. In: Serdyukov P, Braslavski P, Kuznetsov S, Kamps J, Rüger S, Agichtein E, Segalovich I, Yilmaz E (Eds) Adv. Inf. Retr. SE - 92, Springer Berlin Heidelberg, pp 816–819. doi:https://doi.org/10.1007/978-3-642-36973-5_92.
5. White RW, Tatonetti NP, Shah NH, Altman RB, Horvitz E (2013) Web-scale pharmacovigilance: listening to signals from the crowd. J Am Med Inform Assoc 20:404–408. https://doi.org/10.1136/amiajnl-2012-001482
6. Freifeld CC, Brownstein JS, Menone CM, Bao W, Filice R, Kass-Hout T, Dasgupta N (2014) Digital drug safety surveillance: monitoring pharmaceutical products in twitter. Drug Saf 37:343–350. https://doi.org/10.1007/s40264-014-0155-x
7. Ginn R, Pimpalkhute P, Nikfarjam A, Patki A, O'Conner K, Sarker A, Gonzalez G (2014) Mining Twitter for adverse drug reaction mentions: a corpus and classification benchmark. Proc Fourth Work Build Eval Resour Heal Biomed Text Process. http://www.nactem.ac.uk/biotxtm2014/papers/Ginnetal.pdf.
8. Liu X, Liu J, Chen H (2014) Identifying adverse drug events from health social media: a case study on heart disease discussion forums. In: Zheng X, Zeng D, Chen H, Zhang Y, Xing C Neill DB (eds) Smart Heal. Int. Conf. ICSH 2014, Beijing, China, July 10–11, 2014. Proc., Springer International Publishing, Cham, pp 25–36. doi:https://doi.org/10.1007/978-3-319-08416-9_3.
9. K. O'Conner, A. Nikfarjam, R. Ginn, P. Pimpalkhute, A. Sarker, K. Smith (2014) Pharmacovigilance on Twitter? Mining tweets for adverse drug reactions. Am Med Informatics Assoc Annu Symp
10. Nikfarjam A, Sarker A, O'Connor K, Ginn R, Gonzalez G (2015) Pharmacovigilance from social media: mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. J Am Med Informatics Assoc. 22:671–681. https://doi.org/10.1093/jamia/ocu041
11. Hakkarainen KM, Hedna K, Petzold M, Hägg S (2012) Percentage of patients with preventable adverse drug reactions and preventability of adverse drug reactions—a meta-analysis. PLoS One 7:e33236 10.1371%2Fjournal.pone.0033236
12. Sultana J, Cutroneo P, Trifirò G (n.d.) Clinical and economic burden of adverse drug reactions. J Pharmacol Pharmacother 73:OP-77 VO-4. doi:https://doi.org/10.4103/0976-500X.120957.
13. Ahmad SR (2003) Adverse drug event monitoring at the food and drug administration. J Gen Intern Med 18:57–60. https://doi.org/10.1046/j.1525-1497.2003.20130.x
14. Lindquist M (2008) VigiBase, the WHO Global ICSR Database System: basic facts. Drug Inf J 42:409–419. https://doi.org/10.1177/009286150804200501

15. Cocos A, Fiks AG, Masino AJ (2017) Deep learning for pharmacovigilance: recurrent neural network architectures for labeling adverse drug reactions in Twitter posts. J Am Med Informatics Assoc 24:813–821. https://doi.org/10.1093/jamia/ocw180
16. Sarker A, Gonzalez G (2015) Portable automatic text classification for adverse drug reaction detection via multi-corpus training. J Biomed Inform 53:196–207. https://doi.org/10.1016/j.jbi.2014.11.002
17. Bengio Y, LeCun Y, Henderson D (1994) Globally trained handwritten word recognizer using spatial representation, convolutional neural networks, and hidden Markov models. Adv Neural Inf Process Syst 937–944
18. Kim Y (2014) Convolutional neural networks for sentence classification. http://arxiv.org/abs/1408.5882 (accessed March 4, 2016)
19. Kalchbrenner N, Grefenstette E, Blunsom P (2014) A convolutional neural network for modelling sentences. http://arxiv.org/abs/1404.2188
20. De Boom C, Van Canneyt S, Demeester T, Dhoedt B (2016) Representation learning for very short texts using weighted word embedding aggregation. doi:https://doi.org/10.1016/j.patrec.2016.06.012
21. Das R, Zaheer M, Dyer C (2015) Gaussian lda for topic models with word embeddings. Proc 53nd Annu Meet Assoc Comput Linguist
22. Pimpalkhute P, Patki A, Nikfarjam A, Gonzalez G (2014) Phonetic spelling filter for keyword selection in drug mention mining from social media. AMIA Summits Transl Sci Proc 2014:90–95 http://www.ncbi.nlm.nih.gov/pmc/articles/PMC4333687/
23. Abadi M, Agarwal A, Barham P, Brevdo E, Chen Z, Citro C, Corrado GS, Davis A, Dean J, Devin M, Ghemawat S, Goodfellow I, Harp A, Irving G, Isard M, Jia Y, Jozefowicz R, Kaiser L, Kudlur M, Levenberg J, Mane D, Monga R, Moore S, Murray D, Olah C, Schuster M, Shlens J, Steiner B, Sutskever I, Talwar K, Tucker P, Vanhoucke V, Vasudevan V, Viegas F, Vinyals O, Warden P, Wattenberg M, Wicke M, Yu Y, Zheng X (2016) TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems, . http://arxiv.org/abs/1603.04467 (accessed April 8, 2016)
24. Godin F, Vandersmissen B, De Neve W, Van de Walle Rik (2015) Multimedia lab@ acl w-nut ner shared task: named entity recognition for twitter microposts using distributed word representations, in: ACL-IJCNLP 2015, : pp. 146–153. http://www.aclweb.org/anthology/W/W15/W15-43.pdf#page=158 (accessed April 11, 2017)
25. T. Mikolov, K. Chen, G. Corrado, J. Dean (2013) Efficient estimation of word representations in vector space. http://arxiv.org/abs/1301.3781
26. Rehurek R, Sojka P (2010) Software framework for topic modelling with large corpora, in: Proc. Lr. 2010 Work. New Challenges NLP Fram pp. 45–50. http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.695.4595 (accessed April 11, 2017).
27. Nair V, Hinton GE (2010) Rectified linear units improve restricted boltzmann machines, in: Proc. 27th Int. Conf. Mach Learn:807–814
28. Dahl GE, Sainath TN, Hinton GE (2013) Improving deep neural networks for LVCSR using rectified linear units and dropout. IEEE Int Conf Acoust Speech Signal Process 2013:8609–8613. https://doi.org/10.1109/ICASSP.2013.6639346.
29. Srivastava N, Hinton G, Krizhevsky A, Sutskever I, Salakhutdinov R (2014) Dropout: a simple way to prevent neural networks from overfitting. J Mach Learn Res 15:1929–1958 http://jmlr.org/papers/v15/srivastava14a.html
30. Hinton GE, Srivastava N, Krizhevsky A, Sutskever I, Salakhutdinov RR (2012) Improving neural networks by preventing co-adaptation of feature detectors. http://arxiv.org/abs/1207.0580 (accessed April 28, 2017)
31. Dietterich TG (1998) Approximate statistical tests for comparing supervised classification learning algorithms. Neural Comput 10:1895–1923. https://doi.org/10.1162/089976698300017197
32. Johnson AEW, Pollard TJ, Shen L, Lehman LH, Feng M, Ghassemi M, Moody B, Szolovits P, Anthony Celi L, Mark RG (2016) MIMIC-III, a freely accessible critical care database. Sci Data 3:160035. https://doi.org/10.1038/sdata.2016.35
33. Pan SJ, Yang Q (2010) A survey on transfer learning. IEEE Trans Knowl Data Eng 22:1345–1359. https://doi.org/10.1109/TKDE.2009.191