Detecting Twitter posts with Adverse Drug Reactions using Convolutional Neural Networks

Sarthak Jain, Xun Peng, Byron C. Wallace
Northeastern University, Boston, MA, USA

Abstract We describe our system for Shared Task 1, which involves recognizing tweets containing adverse drug reaction mentions. We used a relatively standard CNN architecture coupled with task-specific features and pre-processing steps to achieve an F-score on the test set of 0.412, placing us as the third best-scoring team and 0.015 points beneath the best score. We observe empirically that the test dataset differs substantially from the training corpus; building models to explicitly account for this shift will likely be important to further improve results.

1 Overview

Briefly, our system comprises a standard CNN architecture [1] augmented with manually crafted features and an embedding projection layer. In this brief note, we describe these components in greater detail.

2 Preprocessing

We remove all characters except for those in the Basic Latin Set (U+0000 - U+007F) and replace @usernames, urls, numbers and dollar symbols with corresponding special tokens. We also add a space before and after all non-alphanumeric characters. We heuristically replace all drug mentions in sentences with a DRUG token using Drugbank database[1] [2]. Finally, we split the sentence into tokens on whitespace and lemmatize each word using nltk [3].

2.1 Features

As mentioned above, we use a CNN architecture, which may be viewed as inducing its own feature map from words. We augment this learned representation with manually crafted indicator features based on prior work by [4][2]. These include features that encode information derived via synset expansions and use of an ADR Lexicon, those capturing SentiWord Score, topic based and structural features, and features based on cluster assignments and change phrases .

3 Models

3.1 Projecting embeddings

Embedding based NLP systems have been shown to benefit from pre-training on large volumes of unlabeled data [5]. While raw documents or texts are usually unannotated, they do contain structure. This can be leveraged to learn word features in an unsupervised fashion. Context is one strong indicator for word similarity; related words tend to occur in similar contexts (i.e., words realize some notion of 'distributional semantics'). A popular approach to capitalizing on this insight involves estimating word embeddings by maximizing the probability 'nearby' words, i.e. those within a given window size, can be predicted based on the induced representation of the target word. This strategy is used, e.g., in the popular skipgram and CBOW architectures [6].

However, if these embeddings are retrained using a small amount of labeled data, this can lead to severe overfitting for the words that are seen in training. Moreover, word vectors corresponding to tokens not present in the small training corpus will not be updated at all. Thus, rather than retraining or fine-tuning embeddings, we introduced a transformation layer to project pre-trained embeddings into a task-specific space. This has the indirect effect that all embeddings in pre-trained space are updated rather than only those in the training set. Instead of hoping the projection directly learns a desired underlying mapping, we explicitly let it fit a residual mapping. This is inspired by the residual layers used in deep convolutional networks [7]. The original mapping is recast into $F(W)+W$. Concretely, we remap words as follows:

[1] https://www.drugbank.ca/

[2] Available at https://bitbucket.org/asarker/adrbinaryclassifier

$$W' = \text{ReLU}(W \cdot P) + W \tag{1}$$

Our intuition here is that it is easier to optimize the residual mapping than to optimize vector representations in the original or source embedding space. In the extreme, if an identity mapping were optimal, it would be easier to push the residual to zero than to fit an identity mapping.

We use two sets of pre-trained word embeddings in this project: (i) Google News Dataset (300 dim, 3M words) [6] and (ii) Twitter Dataset (400 dim, 3M words) [8]. For each word in our vocabulary, we look up a corresponding word embedding in pre-trained word2vec vectors. If the embedding is found, we add it to our system. In the case that a word is not found, we generate a random vector for them if they occur more than 2 times in the dataset. Otherwise, we replace them with a single UNK token.

3.2 Convolutional Neural Network

Convolutional Neural Networks (CNN)s have been extensively used in many computer vision and, more recently, NLP tasks. In NLP, CNNs were previously used successfully in sentence classification and sentiment analysis [1, 9]. CNNs start with a convolutional layer, usually equipped with Rectified Linear Units (ReLUs) as activation units.[3] Convolutional filters normally have the same width as the word vectors, thus producing feature maps with only 1 column. The network is then stacked by a max-pooling layer that picks the maximum element from each column. The last layer is a feed forward layer to an output layer with a softmax activation.

3.3 Feature Only Model

We also train a multilayer perceptron on the feature vectors that performs binary classification. The MLP comprises a single hidden layer and an output layer. We use a softmax activation to produce the output and we minimize cross entropy loss over the training dataset.

3.4 Feature+CNN Model

In this model, we concatenate the features returned by pooling layer of convolution network with the output of the hidden layer of feature network before passing them through the softmax layer.

3.5 Training Setup

For CNN models, we use filters of size 1-5 with 250 filters for each. We use a dropout layer [10] with $p = 0.5$ for regularization. For feature models, we use a hidden layer of size 300 with $\ell 2$ regularisation at $\lambda = 0.001$ and Relu activation function. For each model, we used l2 regularization for dense softmax layer with $\lambda = 0.001$. The models were trained using Adam Optimiser ($lr = 0.01$) [11]. We used early stopping when validation loss doesn't show absolute improvement of value > 0.01 for 5 epochs. We use the model with least validation loss during training for evaluation. All hyperparameter selection is done on the validation set.

We also used an ensemble of all methods for evaluation. To counter the class imbalance problem, rather than using majority voting, we labeled the data point as positive if at least 2 classifiers labeled it as positive.[4] We also set the threshold for a tweet to be labeled as positive to 0.35. This setting reduced our precision but significantly improved recall.

[3] A ReLU takes an input and returns the original input if it is larger than 0, otherwise, it returns 0.

[4] This performed better than alternative ensembling approaches such as 'stacking' on the dev dataset.

Experiments

3.6 Datasets

The training data consists on 11293 tweets with 9330 negative samples and 965 positive samples. The test data consists of 9961 tweets (9190 negative, 771 positive). We will refer this particular train and test split as the **original** corpus.

Because the training and test datasets were generated by sampling at different times with different drug names, we wanted to see whether the train and test sets reflected different input feature distributions. To assess this, we combined the train and test data and shuffled them together. We then randomly divided the total data into 2 parts with the same distribution of labels as given, i.e., we sampled 9330 negative and 965 positive data points for training and kept remaining for testing. We will call this train/test split as **randomized**.

We use 90/10 split of training data in both sets above for validation purposes.

4 Results

We present the results of the experiments in Table 1.

Model	Original			Randomized		
	P	R	F1	P	R	F1
CNN (goog)	0.44	0.31	0.36	0.55	0.51	0.53
Feature	0.45	0.31	0.37	0.51	0.59	0.55
CNN (goog) + Feature	0.53	0.28	0.36	0.48	0.64	0.55
CNN (tw) + Feature	0.46	0.27	0.33	0.60	0.48	0.53
CNN (goog + EP) + Feature	0.53	0.31	0.39	0.54	0.61	0.58
CNN (tw + EP) + Feature	0.53	0.24	0.34	0.52	0.54	0.53
Ensemble	0.39	0.43	0.41	0.46	0.70	0.56

Table 1: Evaluation Metrics on Original/Randomized Datasets for different models in our system. goog = using Google Embeddings, tw = using Twitter Embeddings, + EP = with embedding projection

We will mention following observations in regards to our results :

- As can be clearly seen from the table, the models consistently and significantly improved when trained on randomised datasets. This suggests that our model could benefit from better normalization of texts as well as the use of external ontologies to map words to concepts like drugs, side effects, and so on. In this work, We do not explore the differences between training and test set underlying this observation in further detail.
- Models with the embedding projection outperform ones with no embedding projection by a small margin.
- All models have better precision than recall in the original train/test split.
- Google Embeddings performed better than twitter embeddings, which we found somewhat surprising.

5 Conclusion

We proposed a hybrid CNN and feature based system to perform binary classification of tweets containing mentions of Adverse Drug Reactions (ADRs). We evaluated our system against the training and test dataset provided as part of the shared task. On using these datasets as they were intended, we achieved an overall F1 score of 0.412. We also observe that combining the given datasets and dividing it into training and test set randomly significantly improved the performance of our classifiers. This suggests a shift in the data distributions. In future work, we will further study the reasons behind this shift and how they can be rectified to generate more precise classifiers.

References

[1] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

[2] Iain Marshall, Joël Kuiper, Edward Banner, and Byron C. Wallace. Automating biomedical evidence synthesis: Robotreviewer. In *Proceedings of ACL 2017, System Demonstrations*, pages 7–12, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[3] Steven Bird, Ewan Klein, and Edward Loper. *Natural Language Processing with Python*. O'Reilly Media, 2009.

[4] Abeed Sarker and Graciela Gonzalez. Portable automatic text classification for adverse drug reaction detection via multi-corpus training. *J. of Biomedical Informatics*, 53(C):196–207, February 2015.

[5] Yoav Goldberg. Neural network methods for natural language processing. *Synthesis Lectures on Human Language Technologies*, 10(1):1–309, 2017.

[6] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[8] Fréderic Godin, Baptist Vandersmissen, Wesley De Neve, and Rik Van de Walle. Multimedia lab@ acl w-nut ner shared task: named entity recognition for twitter microposts using distributed word representations. *ACL-IJCNLP*, 2015:146–153, 2015.

[9] Ye Zhang and Byron Wallace. A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification. *arXiv preprint arXiv:1510.03820*, 2015.

[10] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958, January 2014.

[11] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.