Adverse Drug Reaction Classification With Deep Neural Networks

Conference or Workshop Item

How to cite:

© 2016 The Author
Version: Version of Record

Link(s) to article on publisher’s website:

Copyright and Moral Rights for the articles on this site are retained by the individual authors and/or other copyright owners. For more information on Open Research Online’s data policy on reuse of materials please consult the policies page.

oro.open.ac.uk
Adverse Drug Reaction Classification With Deep Neural Networks

Trung Huynh\textsuperscript{1,3}, Yulan He\textsuperscript{2}, Alistair Willis\textsuperscript{1} and Stefan Rüger\textsuperscript{1}
\textsuperscript{1}Knowledge Media Institute, Open University, UK
\textsuperscript{2}Systems Analytics Research Institute, Aston University, UK
\textsuperscript{3}Google UK

trunghlt@gmail.com, y.he@cantab.net
{alistair.willis, stefan.rueger}@open.ac.uk

Abstract

We study the problem of detecting sentences describing adverse drug reactions (ADRs) and frame
the problem as binary classification. We investigate different neural network (NN) architectures
for ADR classification. In particular, we propose two new neural network models, Convolu-
tional Recurrent Neural Network (CRNN) by concatenating convolutional neural networks with
recurrent neural networks, and Convolutional Neural Network with Attention (CNNA) by adding
attention weights into convolutional neural networks. We evaluate various NN architectures on
a Twitter dataset containing informal language and an Adverse Drug Effects (ADE) dataset con-
structed by sampling from MEDLINE case reports. Experimental results show that all the NN
architectures outperform the traditional maximum entropy classifiers trained from \( n \)-grams with
different weighting strategies considerably on both datasets. On the Twitter dataset, all the NN
architectures perform similarly. But on the ADE dataset, CNN performs better than other more
complex CNN variants. Nevertheless, CNNA allows the visualisation of attention weights of
words when making classification decisions and hence is more appropriate for the extraction of
word subsequences describing ADRs.

1 Introduction

Adverse Drug Reactions (ADRs) are potentially very dangerous to patients and are amongst the top
causes of morbidity and mortality (Pirmohamed et al., 2004). Many ADRs are hard to discover as
they happen to certain groups of people in certain conditions and they may take a long time to expose.
Healthcare providers conduct clinical trials to discover ADRs before selling the products but normally
are limited in numbers. Thus, post-market drug safety monitoring is required to help discover ADRs
after the drugs are sold on the market. In the United States, Spontaneous Reporting Systems (SRs) is
the official channel supported by the Food and Drug Administration. However these system are typically
under-reported and many ADRs are not recorded in the systems. Recently unstructured data such as
medical reports (Gurulingappa et al., 2012a) or social network data (Ginn et al., 2014; Nikfarjam et al., 2015) have been used to detect content that contains ADRs. Case reports
published in the scientific biomedical literature are abundant and generated rapidly. Social networks are
another source of redundant data with unstructured format. While an individual tweet or facebook status
that contains ADRs may not be clinically useful, a large volume of these data can expose serious or
unknown consequences.

Common approaches to detect content with ADRs used Support Vector Machines (SVMs), Random
Forest, Maximum Entropy classifiers with heavily hand-engineered features (Rastegar-Mojarad et al.,
2016) [Sarker et al., 2016] [Zhang et al., 2016]. These features normally include \( n \)-grams with different
weighting schemes. When used with unigrams, these approaches suffer from the fact that their models
do not take in account the interaction between terms and their orders. This problem can partially be
solved by using bi-grams or trigrams. However this leads to the number of features exploding, and the
models are thus easily overfitted. Meanwhile neural networks with pre-trained word representations have

This work is licenced under a Creative Commons Attribution 4.0 International Licence. Licence details: \url{http://creativecommons.org/licenses/by/4.0/}
had some successes in other text classification tasks (Kalchbrenner et al., 2014; Kim, 2014; Zhou et al., 2015; Yang et al., 2016). Word representations that are typically pre-trained with unlabelled data are matrices that can be used to project words into a dense low-dimensional space (typically from 50 to 300 dimensions). These neural networks often contain convolutional filters or recurrent connections that compute weighted sums of words and their contexts.

In this paper, we train word embeddings and use them as parameters to different neural network architectures to classify documents to whether they contain ADR content. We show that even without engineered features our neural networks with word embeddings outperform maximum-entropy classifiers with different weighting schemes for n-gram features.

The rest of the paper is organised as follows. Section 2 discusses related work on ADR detection from text and briefly describes word embeddings. Various neural network architectures including two new models, Convolutional Recurrent Neural Networks (CRNN) and Convolutional Neural Network with Attention (CNNA), are presented in Section 3. Experimental setup and results are discussed in Section 4 and 5 respectively. Finally, Section 6 concludes the paper.

2 Related Work

2.1 ADR Detection from Text

Natural Language Processing (NLP) approaches have been used to detect ADRs and their relations from Electronic Health Records (EHR) (Wang et al., 2009; Friedman, 2009) and clinical reports (Aramaki et al., 2010; Gurulingappa and Fluck, 2011). Both EHRs and clinical reports have several advantages over plain text or social network data such as they contain more complete records of patients’ medical history, treatments, conditions. Leaman and Wojtulewicz (2010) are ones of the first to attempt to extract ADRs from text and social networks. They generated a golden data set for DailyStrength[^1], a social network where its users share health-related struggles and successes with each other, and lexicons created from UMLS Metathesaurus[^2], SIDER [Kuhn et al., 2010] and The Canada Drug Adverse Reaction Database[^3]. Their data set contains a total of 6,890 comment records. Their approach is rather straightforward, which is to use direct matches of terms in their built lexicons against terms tokenised from the comments. They reported a precision of 78.3%, a recall of 69.9% and an F-score of 73.9%. Further work that focused on exploring existing or expanded lexicons to find ADRs can be found at (Benton et al., 2011; Harpaz et al., 2012; Gurulingappa et al., 2012b; Yates and Goharian, 2013; Liu and Chen, 2013). Lexicon-based approaches are limited in the number of drugs studied or the number of target ADRs. Nikfarjam and Gonzalez (2011) introduces a rule-based approach on the same DailyStrength data set. Though it does not perform as well as the lexicon-based approach, it can detect expressions not included in the lexicons.

With the emergence of annotated data, there have been more machine-learning based approaches to ADRs detection. Gurulingappa et al. (2011) used Decision Trees, Maximum Entropy and SVMs with many engineered features. They obtained an F-score of 77% for ADR class with ADE data set. Sarker and Gonzalez (2015) used SVMs with different feature sets from combined data sets (ADE, Twitter and DailyStrength). They observed that combining Twitter with ADE data sets or DailyStrength with Twitter data sets help improving their performances. Nikfarjam et al. (2015) used Conditional Random Fields to simultaneously detect ADRs and the condition for which the patient is taking the drug. In addition to traditional features, they introduced embedding clusters features trained with word2vec and k-means clustering. Rastegar-Mojarrad et al. (2016) and Zhang et al. (2016) used ensemble models that combine decision trees (Random Forest) or different classifiers with various features.

Overall, approaches to ADR detection have been limited with shallow models and heavily engineered features. There has been a lack of an end-to-end approach that relies on redundancy of unannotated and annotated data.

2.2 Word Embeddings

Most of deep neural networks in NLP utilise an embedding that projects each unique word into a dense lower-dimensional space (typically from 50 to 300 dimensions) and use it as the input of the network. An embedding is a matrix $\mathbb{R}^{v \times s}$, where $v \in \mathbb{R}$ is the size of vocabulary and $s \in \mathbb{R}$ is the number of dimensions in the low dimensional space. These embeddings are normally trained from unlabelled text that are usually redundant in huge amounts from sources like Wikipedia or CommonCrawl. The embeddings are usually trained in a fashion so that the dot product of vectors of a word and its neighbour word preserves the words’ point-wise mutual information (PMI) (Mikolov et al., 2013; Pennington et al., 2014; Levy and Goldberg, 2014; Shazeer et al., 2016).

After being trained, these vectors can be used to look for word synonyms by looking for words with their vectors closest to the searched word’s vector. They can also be used to answer certain types of questions like “what is to Italy like Paris to France?” by looking for words with vectors that is closest to vector $\vec{Paris} - \vec{France} + \vec{Italy} = \vec{Rome}$ (Mikolov et al., 2013). By representing words using these vectors, the model captures derived information from co-occurrences of the contained words from the unsupervised pre-training. Additionally using lower dimensional vector space also helps reduce overfitting. A tokenised sentence or document with their tokens projected by an embedding becomes a dense matrix that can then be fed as an input into a neural network.

3 Methods

In this section, we introduce a number of neural network architectures and propose two new models, Convolutional Recurrent Neural Networks (CRNN) and Convolutional Neural Network with Attention (CNNA).

3.1 Convolutional Neural Network (CNN)

Deep Convolutional Neural Networks (CNNs) are recently extensively used in many computer vision (Alex Krizhevsky et al., 2012; Szegedy et al., 2014; Simonyan and Zisserman, 2014; He et al., 2015) and NLP tasks. In NLP, CNNs (Figure 1a) were previously used successfully in sentence classification and sentiment analysis (Collobert et al., 2011; Kim, 2014; Zhou et al., 2015). The network starts with a convolutional layer with Rectified Linear Units (RLUs) (Glorot et al., 2011). A RLU takes an input and returns the original input if it is larger than 0, otherwise, it returns 0. The convolutional filters normally have the same width as the word vectors, thus, produce 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 feedforward layer to an output layer with either sigmoid (Equation 3) or softmax (Equation 4) activations depending on whether the classification is binary or multinomial. The mathematical formulations for different layers of the CNN are:

$$
l_{1i}^k = \max\{(W_1^k \ast X)_{1i}, 0\},$$  
$$l_{2k} = \max_i(l_{1i}^k).$$  

If it is binary classification, we set

$$l_3 = \frac{1}{1 + \exp(-W_3^T l_2 - b_3)},$$  

or, otherwise, if it is multinomial classification

$$l_{3i} = \frac{\exp(W_3^T l_2 + b_3)_i}{\sum_j \exp(W_3^T l_2 + b_3)_j}.$$  

Here, $X \in \mathbb{R}^{d \times s}$ is the input matrix after the projection, $d \in \mathbb{N}$ is the document length, $s \in \mathbb{N}$ is the word vector length, $\ast$ denotes convolution, $W_1^T \in \mathbb{R}^{h \times e}$, $W_3 \in \mathbb{R}^{k \times 1}$ are the neural network weights, $b_3 \in \mathbb{R}$ is the bias term, $h \in \mathbb{N}$ is the convolutional filter height and $k \in \mathbb{M}$ is the number of convolutional filters.

Source code is available at https://github.com/trunghlt/AdverseDrugReaction
3.2 Recurrent Convolutional Neural Network (RCNN)

Another architecture that has achieved comparable results in sentence classification task is Recurrent Convolutional Neural Network (RCNN) (Zhou et al., 2015). The RCNN (Figure 1b) also starts with a convolutional layer like the CNN but followed by a recurrent layer rather than a max pooling layer. The convolutional filters have the same width as the embedding and are applied in the manner that the outputs have the same number of rows as the input. We also use the Rectified Linear function as the activation function for the convolutional layer. For the recurrent layer, at time step $t$, the recurrent node takes the input from the outputs produced by all the convolutional filters at row $t$ and previous values at time step $t - 1$. For activation, we use Gated Recurrent Units (Cho et al., 2014). Finally the nodes at the last time step are fully connected to a single node with a sigmoid activation to produce binary classification:

$$l^k_{t_{11}} = \max\{(W^k_1 \ast X)_{t_{11}}, 0\},$$  \hspace{1cm} (5)

$$l^2_{t_j} = \text{gru}(l^1_{t_{11}}),$$  \hspace{1cm} (6)

$$l^3 = \frac{1}{1 + \exp(-W^3_3 l^2_{t_a} - b_3)},$$  \hspace{1cm} (7)

where $\text{gru}(X): \mathbb{R}^{d \times k} \rightarrow \mathbb{R}^{d \times r}$ denotes Gated Recurrent Unit (GRU) with input $X$, $r \in \mathbb{R}$ is the size of the output of the RNN, $t \in \mathbb{R}$ denotes a time step that is equivalent to the order of the window that produces the values from convolutional filters.

GRUs are recurrent units which have additional gating units. The gating units modulate the flow
of information inside the unit. The activation $h^t_j$ of a GRU at time $t$ is a linear interpolation between previous activations:

$$h^t_j = (1-z^t_j)h^t_{j-1} + z^t_j\tilde{h}^t_j,$$

where $z^t_j$ acts as a gate which decides how much the unit updates its content and it is computed by $z^t_j = \sigma(W^z_j x^t + U^z_j h^t_{t-1})^j$, while $\tilde{h}^t_j$ is a candidate activation, computed similarly to traditional recurrent unit, $\tilde{h}^t_j = \tanh (W x^t + U (r^t \odot h^t_{t-1}))^j$, where $r^t$ is a reset gate and $\odot$ is an element-wise multiplication. These reset gates can be computed similarly to the update gate $r^t_j = \sigma(W^r_j x^t + U^r_j h^t_{t-1})^j$.

The idea behind gated flows is to enable information further in the past to be propagated to the current unit with fewer time steps. With fewer time steps, the error gradient is passed by back-propagation more efficiently due to the propagated gradient is less prone to vanishing or exploding. (Cho et al., 2014) show that GRUs have better performance than traditional tanh and comparable performance to Long Short-Term Memory (LSTM) units.

### 3.3 Convolutional Recurrent Neural Network (CRNN)

Inspired by RCNN, we introduce a new architecture called Convolutional Recurrent Neural Network (Figure 1c) that stacks a convolutional layer on top of a recurrent layer, which is opposite to a RCNN. The intuition behind this is that the recurrent layer can capture the global contexts before information passed to the convolutional layer. The convolution and max-pooling layers replace the traditional average over hidden features or only hidden features at the last word in the sentence. We use GRUs for the recurrent layers and RLUs for the convolutional layer:

$$l_1 = \text{gru}(X_i *),$$

$$l_2 = \max\{ (W_k^2 \ast l_1)_{i_1}, 0 \},$$

$$l_3 = \frac{1}{1 + \exp(-W_3^t l_2 - b_3)}.$$  

### 3.4 Convolutional Neural Network with Attention (CNNA)

Inspired by the works from (Bahdanau et al., 2015; Hermann et al., 2015; Rush et al., 2015; Rocktäschel et al., 2016; Yang et al., 2016) which use the attention mechanism that the generation of outputs at each consecutive time step is conditioned on different subsets of the input, we introduce a new architecture built on top of the CNN with additional attention mechanism (Figure 1d). The addition is one-filter convolutional layer on top of the direct outputs from the first convolutional layer. The outputs of this convolutional layer are normalised with softmax function so that they can have a sum of 1, which we call attention weights. These attention weights are then multiplied with the outputs from the first convolution (dot product). The outputs of this dot product are forward connected to a perceptron for binary classification.

The advantage of introducing the attention mechanism is that we can use these attention weights to extract words that the model mainly uses for the prediction. In practice, we found it very interesting and helpful to see which words are more weighted in the model’s decisions (see Figure 2 in Section 5).

Even though getting more popular, attention mechanism has been mostly applied with recurrent neural networks (Bahdanau et al., 2015; Hermann et al., 2015; Rush et al., 2015; Rocktäschel et al., 2016; Yang et al., 2016). There are recently some works that incorporate attention mechanism with CNNs (Yin et al., 2016; Yin et al., 2016). In (Yin et al., 2016), attention weights are computed differently by taking the dot product between the representation of the input query and the sentences in question-answer tasks. In (Yin et al., 2016), even though called attention, the attention layers behave more like feature maps than traditional attention weights (multiplied with features) and are computed by matching two feature maps.

### 4 Experimental Setup

#### 4.1 Datasets

We use two datasets for the evaluation of various neural network architectures. The first one is a Twitter dataset (Sarker et al., 2016) published for a shared task in Pacific Symposium on Biocomputing, Hawaii,
2016. The tweets associated with the data were collected using generic and brand names of the drugs, and also their possible phonetic misspellings. The tweets were annotated for presence of ADRs. In the shared task, 70% (7,575) of the original data set is shared for training and the rest of the data is used for evaluation. Owing to Twitter’s data terms and conditions, only the tweet ids are contained in the original file. At the time of this experiment, we could download only 5,108 tweets (with 557 tweets with ADR descriptions) as many tweets are no longer accessible. Due to the difference in the size of the experimental data set, we can not compare our results directly with the previously reported baselines. Thus we reuse the codes published by (Zhang et al., 2016) that perform classification with the various algorithms (see Section 4.2 for further details).

The second dataset, the ADE (adverse drug effect) corpus, was created by (Gurulingappa et al., 2012b) by sampling from MEDLINE case reports [7]. Each case report provides important information about symptoms, signs, diagnosis, treatment and follow-up of individual patients. The ADE corpus contains 2,972 documents with 20,967 sentences. Out of which, 4,272 sentences are annotated with names and relationships between drugs, adverse effects and dosages.

For both datasets, we use 10-stratified-fold cross-validation and report precision, recall and F-scores of various methods.

4.2 Baselines

For the Twitter dataset, it was reported from the shared task that both the best (Rastergar-Mojarad et al., 2016) and the second best (Zhang et al., 2016) approaches are classifiers with engineered features.

For the Twitter dataset, it was reported from the shared task that both the best (Rastergar-Mojarad et al., 2016) and the second best (Zhang et al., 2016) approaches are classifiers with engineered features.

In order to directly compare our results with the existing approaches, we have reimplemented these classifiers based on the published code by (Zhang et al., 2016) including term-matching classifier based on an ADR lexicon, maximum entropy with n-grams and TFIDF weightings or NB log-count ratio, and maximum entropy with word embeddings. We describe each of these methods below:

- **Term-matching based on an ADR lexicon (TM).** An existing ADR lexicon is directly used for ADR detection. The lexicon contains 13,699 terms describing side effects from COSTART, SIDER, CHV and DIEGO Lab. A document is classified as positive if it contains a term from the lexicon.

- **Maximum-Entropy classifier with n-grams and TFIDF weightings (ME-TFIDF).** For a document $d \in D$, an n-gram $i$ has a weight of

  $$F_i(d) = \begin{cases} 
  (1 + \log(n_i(d))) \times \log \left(1 + \frac{|D| + 1}{|D| - |\{ d \in D \mid n_i(d) > 0 \}| + 1} \right) & \text{if } n_i(d) > 0 \\
  0 & \text{otherwise,}
  \end{cases}$$

  where $n_i(d)$ is the number of times a term $i$ appears in document $d$.

- **Maximum-Entropy classifier with n-grams and NB log-count ratio (ME-NBLCR).** Each n-gram $i$ has a weight of

  $$f_i = \begin{cases} 
  \log \left( \frac{1 + \sum_{y(d) = 1} n_i(d)}{\sum_{i' \in V} (1 + \sum_{y(d) = 1} n_{i'}(d))} \times \frac{\sum_{i' \in V} (1 + \sum_{y(d) = -1} n_{i'}(d))}{1 + \sum_{y(d) = -1} n_{i}(d)} \right) & \text{if } n_i(d) > 0 \\
  0 & \text{otherwise,}
  \end{cases}$$

  where $V$ is a set of all n-grams and $y(d) \in \{1, -1\}$ is the true label of each document.

- **Maximum-Entropy classifier with mean word embeddings (ME-WE).** This method simply uses the average of embeddings of words in each document as their input into a maximum-entropy classifier.

For the ADE dataset, the best performance published is 0.81 in F-score using SVMs trained from a rich set of features including n-grams, UMLS semantic types and concept IDs, synset expansions, polarity indicator features, ADR lexicon matches, and topics, etc. (Sarker and Gonzalez, 2015). However, since our ME-NBLCR outperforms SVMs on ADE, we don’t report the results using SVMs here.

<table>
<thead>
<tr>
<th>Method</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>AUC</th>
<th>Precision</th>
<th>Recall</th>
<th>F1</th>
<th>AUC</th>
</tr>
</thead>
<tbody>
<tr>
<td>TM</td>
<td>0.13</td>
<td>0.89</td>
<td>0.23</td>
<td>0.59</td>
<td>0.30</td>
<td>0.99</td>
<td>0.46</td>
<td>0.53</td>
</tr>
<tr>
<td>ME-TFIDF</td>
<td>0.33</td>
<td>0.70</td>
<td>0.45</td>
<td>0.85</td>
<td>0.74</td>
<td>0.86</td>
<td>0.80</td>
<td>0.94</td>
</tr>
<tr>
<td>ME-NBLCR</td>
<td>0.79</td>
<td>0.14</td>
<td>0.23</td>
<td>0.83</td>
<td>0.91</td>
<td>0.79</td>
<td>0.84</td>
<td>0.95</td>
</tr>
<tr>
<td>ME-WE</td>
<td>0.27</td>
<td>0.73</td>
<td>0.40</td>
<td>0.82</td>
<td>0.48</td>
<td>0.70</td>
<td>0.57</td>
<td>0.76</td>
</tr>
<tr>
<td>CNN</td>
<td>0.47</td>
<td>0.57</td>
<td><strong>0.51</strong></td>
<td><strong>0.88</strong></td>
<td>0.85</td>
<td>0.89</td>
<td><strong>0.87</strong></td>
<td><strong>0.97</strong></td>
</tr>
<tr>
<td>CRNN</td>
<td>0.49</td>
<td>0.55</td>
<td><strong>0.51</strong></td>
<td>0.87</td>
<td>0.82</td>
<td>0.86</td>
<td>0.84</td>
<td>0.96</td>
</tr>
<tr>
<td>RCNN</td>
<td>0.43</td>
<td>0.59</td>
<td>0.49</td>
<td>0.87</td>
<td>0.81</td>
<td>0.89</td>
<td>0.83</td>
<td>0.92</td>
</tr>
<tr>
<td>CNNA</td>
<td>0.40</td>
<td>0.66</td>
<td>0.49</td>
<td>0.87</td>
<td>0.82</td>
<td>0.84</td>
<td>0.83</td>
<td>0.95</td>
</tr>
</tbody>
</table>

Table 1: Adverse drug reaction classification results on the Twitter and ADE datasets.

4.3 Training of Neural Networks

In all the described neural network architectures in Section 3, the training algorithm is Adadelta [Zéiler, 2012] with learning rate of 1.0, decay rate ($\rho$) of 0.95 using library Keras. The embedding is trained together with other parameters. For each fold, we split the training dataset into training and validating sets. The training stops when there is no performance improvement on the validation set after 5 consecutive epochs. The batch size is set as 50. All convolutional window has a size of 5.

5 Results

We compare the precision, recall and F-scores of the positive class (instances labeled as containing the description of adverse drug reactions) of neural network architectures with the baselines in Table 1. Since both the Twitter and ADE datasets contain imbalanced class distribution, we also report the Area Under the ROC Curve (AUC) results. It can be observed that in general, results on the ADE dataset are better than those on the Twitter dataset. This is perhaps not surprising since tweets contain a lot of ill-grammatical sentences and short forms. Simply relying on an ADR lexicon for the detection of ADRs from text gives the worst results. Among the baselines, the best performing method is ME-TFIDF on the Twitter dataset where an F-score of 0.45 and an AUC value of 0.85 are obtained. But on the ADE dataset with more formal language, ME-NBLCR gives superior results compared to ME-TFIDF with an F-score of 0.84 and an AUC value of 0.95. Training MaxEnt from aggregated word embeddings (ME-WE) outperforms the term matching method (TM), but performs worse than both ME-TFIDF and ME-NBLCR.

All the neural network architectures perform similarly on the Twitter dataset and they improve upon the best baseline method ME-TFIDF by 4-6% in F-score and 2-3% in AUC. On the ADE dataset, CNN outperforms other neural network architectures and its performance gain over ME-NBLCR is 7% in F-score and 3% in AUC. Overall, CNN gives the best results although CRNN and CNNA are quite close to CNN in terms of AUC values. It is not very straightforward to explain why CNNs are better than the recurrent architectures in our experiments. Our hypothesis is that as ADR descriptions are composed of short fragments of texts, convolutions with small windows are enough to capture necessary information for ADR classification.

Since CNNA assigns a weight to each word when making classification decision, we show in Figure 2 a visualisation of attention weights of sampled tweets from the Twitter dataset. Words with higher attention weights are highlighted with darker blue colour. We can observe that most of the highlighted words are indeed related to descriptions of adverse drug effects. For example, “neck ache” and “lower back pain” in the fifth tweet and “dry eyed” in the seventh tweet. The above results suggest that although CNNA gives slightly worse results compared to CNN for ADR classification, it presents results in a more interpretable form and could be potentially used for the extraction of word subsequences actually
describing ADRs. As such, CNNA would be a better candidate than CNN for more fine-grained ADR extraction.

6 Conclusion

This paper has explored different neural network (NN) architectures for ADR classification. In particular, it has proposed two new neural network models, Convolutional Recurrent Neural Network (CRNN) and Convolutional Neural Network with Attention (CNNA). Experimental results show that all the NN architectures outperform the traditional Maximum Entropy classifiers trained from n-grams with different weighting strategies considerably on both the Twitter and the ADE datasets. Among NN architectures, no significant differences were observed on the Twitter dataset. But CNN appears to perform better compared to other more complex CNN variants on the ADE dataset. Nevertheless, CNNA allows the visualisation of attention weights of words when making classification decisions and hence is more appropriate for the extraction of word subsequences describing ADRs.

Acknowledgements

YH is partly funded by the EPSRC AMR4AMR project (grant number EP/M02735X/1).

References


