Exploring Document Representations for Adverse Drug Reaction mining in Social Media text

Samuele Garda, University of Potsdam

February 22, 2018

Abstract

Social Media platforms are becoming extensively used for discussing health related issues. Even if Social Media text can be an incredibly rich resource of information for pharmacovigilance, the language used is highly informal and thus challenging to analyze automatically. The aim of this project is to explore the effectiveness of several NLP feature extraction techniques in addressing the task of mining mentions of *Adverse Drug Reactions* in informal text. The methods explored in this work consist in several algorithms widely used in NLP for representing text documents in *dense vectors*. A SVM classifier trained with feature vectors extracted by a simple CNN, fed with word embeddings pre-trained on a corpus of specific domain, outperforms any other system taken into consideration leading to a F1 score of 0.80. These experiments show that with sufficiently large amount of health-related text data it is possible to classify informal text containing mention of *Adverse Drug Reactions* with relatively small effort in feature engineering.

1 Introduction

An *Adverse Drug Reaction* (ADR) is an unexpected and harmful physical event result of taking a medication. The study of ADRs is the concern of the field known as pharmacovigilance. Even though drugs are monitored in clinical trials, due to the various limitations of such testing it is not possible to assess the consequences of the use of a particular drug before it is released. Therefore, post marketing surveillance of ADRs is of utmost importance. There exist spontaneous reporting systems (SRSs) which enable health-care providers and patients to directly submit reports of suspected ADRs such as MedWatch [1] supported by Food and Drug Administration (FDA) agency in United States. However, it is estimated that more than 90% of ADRs are under-reported (Vilhelmsson et al., 2011) limiting the effectiveness of those systems.

With more and more people using social media to discuss health information, not only in specific forums (e.g. AskaPatient [2]), there are millions of messages for instance on Twitter that discuss drugs and their side-effects. These messages contain data on drug usage in much larger sets than any clinical trial will ever have. Hence, text mining can be employed to automatically classify texts or posts that are assertive of ADRs.

In this study some of the most widely-used Natural Language Processing techniques are benchmarked for the ADR mention recognition task. Turning to the detail, the importance of document representation techniques is stressed since they provide a way to create scalable and efficient systems trained on easily accessible data with minimal effort in feature engineering. It is hypothesized that these methods could outperform task specific systems in identifying most ADR mentions. Nonetheless some feature sets characteristic to ADRs recognition, jointly with the main feature sets, are tested as well in order to quantify their contribution to the overall results.

This paper proceeds as follows: Section 2 presents some

[1] www.fda.gov/Safety/MedWatch

[2] www.askapatient.com

of the previous work in this task while in Sections 3 all the methods used are presented in their details. Sections 4 present experiments and the results obtained. Section 5 concludes the study with general insights and possible future work.

2 Related Work

Given that there are standard and extensive ADR lexicons, such as Side Effect Resource (SIDER) (Kuhn et al., 2010), most prior studies focused on exploring existing or customized/expanded lexicons to find ADR mentions in user posts. But this approach has several limitations. First of all user postings are informal and deviate from grammatical rules. Moreover consumers do not normally use technical terms found in medical lexicons and even when correctly identified, matched terms could not refer directly to ADRs but to *Adverse Drug Events* (injury occurring at the time a drug is used, whether or not the drug is identified as a cause of the injury).

(Nikfarjam et al., 2015,) introduced ADRMine, a CRF-based NER system that can recognize ADR-related concepts mentioned in data from DailyStrength [3] and Twitter. The system is trained as well to classify user posts by presence of ADRs. The authors claim that the ADR mining task can be addressed as a sentiment analysis one (mention of ADRs is correlated to negative feelings). Thus in addition to standard task features (ngrams, PoS, ADR lexicon) they implemented a rich feature set based on several sentiment analysis resources and a negation feature handling negated words that could reverse the polarity of a post. Furthermore, they utilized word2vec (Mikolov et al., 2013) to generate 150-dimensional word vectors from Twitter data set and DailyStrength one (more than one million unlabeled user sentences about drugs). Afterwards, the K-means clustering algorithm was performed to group the vectors into 150 clusters. A feature vector then is created with the cluster number for the current token, three preceding and three following tokens. Finally for the classification task a *RBF* kernelized *Support Vector Machine* (SVM) was trained on the feature matrix applying a weighting schema for the classes.

(Dai et al., 2016) implemented a similar system to the previous one (PoS, sentiment features, polarity cues). In addition this study applied a normalization step via a spell checker (Hunspell [4]) configured to use, beyond standard English dictionaries, an ADR dictionary [5] released by (Nikfarjam et al., 2015,) and a dictionary containing drug terms collected from the data set itself. Moreover topic modeling was employed to extract topic weights for each tweet and then set as features. For the classification task they used a linear SVM adjusting the class weights.

(Huynh et al., 2016) have explored different Deep Learning approaches such as CNN (Convolutional Neural Network) and CNNA (Convolutional Neural Network with Attention) by adding attention weights into convolutional neural networks. Both the architectures use a convolutional window of size 5 and randomly initialize the embeddings layer trained together with other parameters. Early stopping was applied: the training process is halted after 5 epochs (forward/backward-propagation step) presenting no improvement.

3 Methods

This section present in more details the algorithms used in this study to represent documents in a dense vector format. These dense vectors are then fed to a supervised machine learning algorithm in order to asses their effectiveness.

3.1 LSA

Latent Semantic Analysis (LSA) (Deerwester et al., 1990) is a method for constructing representation of documents by the contextual-usage meaning of words, widely employed across several NLP application. Firstly this algorithm generates a document by term matrix with the raw frequencies of words, called the *bag of words model* (BOW). Then it applies a weighting scheme to obtain more consistent scores for the words (e.g. *tf-idf*).

LSA employs the *Singular Value Decomposition* (SVD) algorithm in order to achieve a dense representation of the documents. SVD factorizes any real matrix in the product

[3]www.dailystrength.org
[4]hunspell.github.io
[5]diego.asu.edu/downloads/publications/ADRMine/ADR_lexicon.tsv

of three matrices:

$$SVD(A) = U_{m\times m} \times \Sigma_{m\times n} \times V_{n\times n} \tag{1}$$

The intuition behind LSA is that Σ contains the *latent dimensions weights* (topics weights) present in the original matrix, while U represent the dimensionality reduced document-term matrix. Thus the dot product of U and Σ is taken in order to construct the rank lowered vector space, taking into account the *latent dimensions*. Nonetheless estimating the complete SVD decomposition is computationally expensive. Hence, since the values obtained in Σ are ordered from the largest to the smallest, it is a common practice to take the first k values of the topic weights matrix (*Truncated SVD*) before computing the dot product.

Moreover Truncated SVD can be seen as a noisy reduction procedure. This is because it smooths the semantic space capturing "latent dimensions" that generalize over sparser surface dimensions, preserving the variance that best explain the distribution of the data in the new reconstructed vector space.

Since each occurrence of a word is treated as having the same meaning due to the word being represented as a single point in space, a BOW model is not able to capture particular linguistic phenomena such as polysemy. What have been done in this study, as suggested by (Abedi et al., 2014), to partly overcome these limitations is feeding LSA with PoS tagged documents and taking into account bigrams and trigrams frequencies as well.

3.2 paragraph2vec

(Le and Mikolov, 2014) following the intuitions behind the generation of context-predicting word embeddings (Mikolov et al., 2013) created an algorithm able to construct document embeddings. They presented two different methods: *Paragraph Vector Distributional Memory* (PV-DM) and *Paragraph Vector Distributed Bag of Words* (PV-DBOW). The model used for the experiments is the one described below, i.e. PV-DM.

This algorithm basically consists in a shallow neural network (one hidden layer). The input is defined by a matrix of one-hot encoded words $I^{w}_{V\times V}$ and a matrix of one hot encoded documents $I^{d}_{D\times D}$, where V is the size of the vocabulary and D is the number of documents. The document tokens can considered as a words. They act as a memory that remembers what is missing from the current context. For every document and for every word in the document, given a sliding window of size n, n preceding and following word vectors and the document vector are linearly transformed respectively via a weight matrix $W^{w}_{V\times N}$ and $W^{d}_{D\times N}$, where N is the desired dimensionality of the document embeddings (hidden units). The document vector is shared across all contexts generated from the same document but not across documents. The weight matrix for word vectors W^{w} , however, is shared across paragraphs, i.e. the vector for "weak" is the same for all paragraphs. Then the resulting word vectors and the paragraph vector are either averaged or concatenated. Concatenating the vectors result in a much larger, slower and more data-hungry model, but which is proved to perform better than averaging.

The obtained vector is then linearly transformed via another weight matrix : $W'_{N\times V}$ for averaging or $W'_{N*windowsize+N\times V}$ for concatenating. This is done because in the final layer a softmax function is applied which computes a multinomial distribution for the missing word. The training objective is that the output distribution should be as close as possible to the original word one-hot representation.

Since the softmax function for an output word y_w is defined as

$$softmax(x) = \frac{e^{y_w}}{\sum_i e^{y_i}} \tag{2}$$

where y_i is the output probability for each word i, it would be necessary to compute $n = V - y_w$ times the softmax for each word, making the model too computationally expensive. In order to overcome this limitation the authors proposed two optimization tricks : *Negative Sampling* and *Hierarchical Softmax*. The first one samples from a probability distribution n *corrupted* words and uses them for computing the softmax. The second one constructs a binary tree with the whole vocabulary where the words are the leaves and for each node, it explicitly represents the relative probabilities of its child nodes. This define a random walk that assigns probabilities to words which are used in the softmax. The model is trained via stochastic gradient descent and the gradients are obtained with backpropagation.

The authors claim that document embeddings can address some of the key weaknesses of bag-of-words models:

they inherit the semantics property of word2vec vectors (e.g. "France" -"Paris" + "Italy" = "Rome") and they take into consideration the word order, at least in a small context.

3.3 SSWE

(Tang et al., 2014) have developed three different algorithms to generate word embeddings that take into account the polarity of the sentence in which the words appear: *Sentiment Specific Word Embedding* (SSWE). The model used in this work and the one demonstrated to be the most effective by their experiments is the the one they call $SSWE_u$.

This algorithm is an extension based on the C&W neural network model by (Collobert et al., 2011). C&W given an ngram such as "cat chills on a mat" replaces the center word with a random word w^r and derives a corrupted ngram "cat chills RANDOM a mat". The training objective is that the original ngram is expected to obtain a higher context score than the corrupted ngram.

$SSWE_u$ extends this idea by capturing the sentiment information of sentences as well as the contexts of words. It takes as input a set of sentences with their sentiment polarities and via a sliding window creates ngrams and a unique vector for each word (word vectors are shared across sentences). After that in the second layer is performed a linear transformation (weight matrix W_1), while in the third layer a non-linear one via a weight matrix to which is applied $hTanh$ function defined as:

$$hTanh(x) = \begin{cases} -1 \text{ if } x < -1 \\ x \text{ if } -1 \leq x \leq 1 \\ 1 \text{ if } x > 1 \end{cases} \tag{3}$$

The output is passed to the final layer where $SSWE_u$ for each ngram predicts a two-dimensional vector. The two scalars in the output f_0^u and f_1^u stand for language model score and sentiment score of the input ngram. The same process is repeated for each corrupted ngram generated as in C&W model.

The training objectives of the model are that each original ngram should obtain a higher language model score than the correspondent corrupted ngram $f_0^u(t^r)$, and that its sentiment score should be more consistent with the gold polarity annotation of sentence than the corrupted one $f_1^u(t^r)$. In order to do so the loss function of $SSWE_u$ is defined as a linear combination of two hinge losses:

$$loss_u(t, t^r) = \alpha \cdot loss_{cw}(t, t^r) + (1 - \alpha) \cdot loss_{us}(t, t^r) \tag{4}$$

where α functions as a weight between the two parts. The loss for the syntactic structure is defined as

$$loss_{cw}(t, t^r) = max(0, 1 - f^{cw}(t) + f^{cw}(t^r)) \tag{5}$$

while the one for the sentiment polarity is

$$loss_{us}(t, t^r) = max(0, 1 - \delta_s(t) f_1^u + \delta_s f_1^u(t^r)) \tag{6}$$

where δ can take the following values:

$$\delta_s(t) = \begin{cases} 1 & \text{if t is positive} \\ -1 & \text{if t is negative} \end{cases} \tag{7}$$

This model was trained via backpropagation on 10M tweets crawled selected by 5M with positive emoticons and 5M with negative ones. The generated word embeddings (publicly available [6]) are proved to be an effective feature set for sentiment analysis tasks. The authors using a linear SVM and the $SSWE_u$ features, modeled via different convolutional layers, were able to outperform the top model of the Twitter sentiment classification benchmark dataset in SemEval 2013.

3.4 CNN

Once word embeddings are computed, since the task requires to work at a document level, there is the necessity to model them in order to get a document representation, or more technically: to learn their compositionality. Several techniques have been explored (Mitchell and Lapata, 2011) for this kind of task and among them (Collobert et al., 2011) have demonstrated that *Convolutional Neural Network* (CNN) is an efficient method. Moreover (Kim, 2014) with a CNN outperformed state of the art systems in NLP tasks such as sentiment analysis and question classification.

The pre-trained word embebeddings of which it is wanted to learn the compositionality of are loaded in the CNN initial layer called the *embedding layer*. More specifically the CNN is fed with zero padded sentences (in order

[6] ir.hit.edu.cn/ dytang/paper/sswe/embedding-results.zip

to normalize the sentence length) with each word correlated to its own pre-trained embedding. The words in the data set for which a pre-trained embedding is not available are initialized as vectors of zeros. For instance, if the maximum length of the sentence in the data set is n all the sentences are zero padded to be of that size. Given word embeddings of dimension $1 \times m$, every sentence in input is defined by a matrix of $n \times m$. It is possible to set these embeddings as model parameters which will be fine-tuned during the training phase.

Afterwards a set of *feature maps* are extracted (*convolutional layer*). Each *feature map* is a vector obtained by applying a one dimensional convolutional filter to the input. The filter is represented as a matrix of $n \times m$ dimensionality, where n is equal to the embeddings dimension and m is the *region size* (size of the kernel). The values of the filter matrix are updated during the training phase. The *region size* of the convolutional filter can be intuitively compared to using bigrams and trigrims in a BOW model, since the filter by sliding over the input matrix is learning location specific features for each sentence. To the resulting *feature maps* it is applied a non-linear transformation, commonly via *Rectified Linear Unit*, defined as

$$ReLU(x) = max(0, x) \tag{8}$$

Then a pooling operation is performed, usually a maximum pooling (*pooling layer*), in order to extract the most significant features and to obtain a fixed size output matrix. A sliding window of size n (pool size) is passed over the feature map vectors in order to extract the maximum values within the window. Finally the features extracted are flattened in a single vector in order to be passed to the classification layer. The final layer consist in a softmax function as in (2) representing the output classes.

It has been noticed that these models tend to overfit quickly. Thus is a common practice to address this issue by applying the *Dropout* regularization technique. Dropout prevents co-adaptation of hidden units by randomly dropping out - i.e., setting to zero - a proportion p of the hidden units during training. The whole network is trained via forward/back-propagation using the *categorical cross entropy* as loss function.

It must be said that such models are used themselves as complete systems for classification and not just as method to learn words compositionality. Namely in the architecture proposed by (Kim, 2014) before the classification layer a classical fully connected layer with 100 hidden units is applied making such a model a complete deep learning system for classification. Nonetheless since the final layer of a CNN is just a classification step it is possible to use CNNs just as feature extractors. After having been trained as for a normal classification task to tune the filter weights, the document representation is obtained with a one step of forward propagation stopped at the pooling layer. (Tang, 2013) using SVM with feature extracted by a CNN obtained significant gains on popular deep learning datasets for image classification.

One of the key property of CNNs, which made them to be extensively used in Computer Vision is (local) *Compositionality.* Each filter composes a local patch of lower-level features into higher-level representation. It makes intuitive sense that edges are built from pixels, shapes from edges, and more complex objects from shapes. In the NLP case - even if without a clear interpretation as in Computer Vision - this is a key aspect since words compose with each other e.g. an adjective modifying a noun.

4 Experimental setup

Data set

The data set released by the PSB SMM shared task (Ginn et al., 2014) was used in order to assess the performance of all the feature extraction methods. A pharmacology expert wrote a list of 81 drug names and their misspellings were automatically generated. Afterwards the list was used to crawl tweets via the Twitter streaming API. A total of 7574 annotated tweets were made available, which contain binary annotations, ADR and non-ADR. At the moment of download only 4714 tweets were still on-line with just 507 of them containing ADR mentions.

Validation approach

In all the following experiments a linear SVM classifier is used in order to test the various feature sets.

The linear SVM is a linear classifier composed by a loss function and a regularizer. This algorithm initializes randomly its own parameters (a vector) called the decision function. In the training process, given a loss function estimating the error in the predictions, the parameters of the model are updated in order to predict the most consistent class label for each training example. Several techniques can be used for computing the parameters update. The regularizer component instead with its own parameter (called C in SVM) prevent the model to learn complex models that do not generalize well. The peculiarity of SVM is that in the its computation it finds the hyperplane with the greatest margin that separates the most possible samples with respect to the class labels.

Since the labels in the data set are unbalanced a class weighting schema is applied, defined as:

$$weight(c_i) = \frac{N}{C \times freq(c_i)}, \forall c_i \in C \tag{9}$$

where N is the total number of samples and C is the number of classes. In this way, in the case of SVM, the regularization parameter C is adjusted with the class weight in order to weaken the effect of the major class in the predictions.

For each feature matrix precision, recall and F1 score are computed applying a 10 fold stratified cross validation.

Baselines

Below are reported the results of two existing approaches taken as baselines. The systems are described in Section 2.

	P	R	F1
(Nikfarjam et al., 2015,)	0.73	0.72	0.72
(Huynh et al., 2016)	0.47	0.57	0.51

Table 1: *Results of baseline approaches.*

The first one is the approach of (Nikfarjam et al., 2015,). The code provided by the authors [7] was modified in order to retrieve the model's scores with 4714 tweets applying the same validation approach used in this work. The second, is the result of the CNN by (Huynh et al., 2016) with the same validation procedure and roughly the same amount of data (5108 tweets) as input.

4.1 Experiment 1

In the first experiment is assessed the quality of the document representation techniques presented in Section 3. The tweets were tokenized and PoS tagged via TweetNLP (Owoputi et al., 2013). All the model parameters mentioned are set with values that are commonly suggested by the correspondent literature.

LSA is implemented with scikit-learn toolkit (Pedregosa et al., 2011), initialized with PoS tagged words, ngram range of size 3 and asked to preserve 300 latent dimensions. The word counts (no frequency cut off was applied) are then weighted via TF-IDF. For PV-DM model was used gensim (Řehůřek and Sojka, 2010) with the following parameters: sliding window of size 3, no frequency cut off and averaging vector scheme, document embeddings of size 300, 50 words sampled for Negative Sampling and 20 epochs for training.

Two different CNNs are implemented with keras (Chollet et al., 2015): one initialized with the word embeddings provided by (Nikfarjam et al., 2015,) the other with the SSWEs by Tang et al. (2014). Both the architectures have 32 feature maps for region size 2 and 3, a dropout rate after the embedding layer of 0.25 and of 0.50 before the final layer and a max pooling step with pool size equal to 2. They are trained with batches of 64 data points each and both do not fine tune the word embeddings in the training process. The number of epochs is 3 since with a very small amount of data these models overfit quickly even when applying regularization. The document representations are obtained as explained in Section 3.4. For CNN-SSWE an averaging pooling layer is added, in addition to the maximum pooling layer, in order to reproduce document embeddings in a way as similar as possible to (Tang et al., 2014).

	P	R	F1
CNN-ADR	0.635108	0.692456	**0.653732**
CNN-SSWE	0.555319	0.579560	0.561000
Doc2Vec	0.569710	0.668166	0.543439
LSA	0.603410	**0.715273**	0.609552

Table 2: *Results first experiment: SVM scores for dense document representation.*

SVM trained with the document representations obtained by the CNN model initialized with word embeddings of health related text is the best performing model. Comparing the result with the CNN by Huynh et al. (2016) it can be claimed that word2vec is particularly effective in

[7] bitbucket.org/asarker/adrbinaryclassifier

capturing the semantics of words even in domain specific and informal text.

Unexpectedly the classifier trained with document representations extracted with SSWEs is outclassed by the one trained with LSA document vectors. Given the proved effectiveness of such embeddings in sentiment analysis, this might suggest that ADR mention mining can not properly be modeled as being similar to that kind of task. More insights are drawn in the second experiment were sentiment analysis features are integrated in the system.

Moreover it is worth noticing that LSA method, even in its oversimplified assumptions, is the second best document representation technique reporting as well the classifier with the highest recall. This propose that for short documents the latent dimensions (embeddings) found by Truncated SVD are an effective method to capture the main characteristics of documents leading to a classifier strongly confident in its predictions.

It is possible to see that paragraph2vec is the worst performing model. That said, it must be mentioned that the model that paragraph2vec authors found to be the most effective is the concatenative one. Nonetheless in this task too few data points were available to feed that particularly data hungry model.

Even if CNN-ADR is outperforming a normal CNN the results show that none of the model performed better than the state of the art system.

Experiment 2

Taking into consideration the two best performing document representation techniques from the previous experiment here the improvement of task specific features to the overall result are assessed.

For each external knowledge resource is performed a search through the data set in order to compute a standard vector representation (all entries are 0). Afterwards frequencies of each component of the feature vector are taken for each tweet.

With respect to the sentiment analysis features, a standard vector of size 317 is extracted via the lexicon [8] provided by (Liu, 2010) . The lexicon consists of 6800 words divided ca. in half by negative and positive words. Since negated words can reverse the polarity of a tweet, e.g. 'not bad', a feature set for handling negated words is adopted. A simple negation list [9] is used to mark every preceding and following negation token as negated, resulting in a standard vector of size 609. These two feature matrices are grouped in one as being established methods for sentiment analysis tasks. A regular expression search is performed with the ADR lexicon by (Nikfarjam et al., 2015,) through the data set generating a standard vector of size 310.

	P	R	F1
CNN-ADR-ADR-LEX	0.653314	0.696127	**0.669801**
CNN-ADR-SENT	0.640729	0.682680	0.655895
CNN-ADR-TOT-EXTRA	0.662099	0.700895	0.677079
LSA-ADR-LEX	0.622110	0.718083	0.641007
LSA-SENT	0.608236	0.696856	0.623334
LSA-TOT-EXTRA	0.628709	0.694747	0.648200

Table 3: *Results experiment 2: SVM scores with best base model and merged task specific features.*

The best features are still the on extracted by CNN with ADR word embeddings. Interestingly the sentiment feature are not improving the overall result. In addition to the findings of the first experiment (CNN-SSWE), this might suggest that ADRs mention mining do not belong to the class of sentiment analysis task. Indeed negative feelings are not always related to ADR but they may also refer to ADE or more general expression: e.g. in the tweet "Cipro is poison and Big Pharma knows it" a brand name of a drug is associated with a negative word as "poison" nonetheless there is no mention of an ADR.

The ADR lexicon features instead are improving the performances of the classifiers. This feature set is adding important information to the task since it provides clear features for the classification algorithm in identifying mentions of ADR more precisely than the sentiment one. Nonetheless it must be said that it is affected by some of the same limitations. For instance in the tweet "Xarelto side effect: 'may cause bleeding, most of which is serious and sometimes leads to death' That sounds fantastic." side effects are mentioned but not experienced by the user. This example show that even tough lexicons approaches can be useful in detecting standard ADR tweets they can

[8]www.cs.uic.edu/ liub/FBS/sentiment-analysis.html

[9]words ending in "n't", not, no, never, nobody, nothing, none, nowhere, neither

also easily lead to misclassification since they can not capture the context in which the expressions appear.

Experiment 3

In this final experiment the performances of the classifier are tested with just the document representations obtained by CNNs. The difference with the previous setting is that, as proposed in (Kim, 2014), the initial embedding layer is fine tuned during the training process. In this way also the words embeddings weights are updated with the training objective of minimizing the error in the prediction of the tweet class. After the training process the document vectors are extracted as before and fed to SVM.

	P	R	F1
CNN-ADR-T	0.787717	0.823639	**0.803004**
CNN-SSWE-T	0.581042	0.610747	0.590376

Table 4: *Results experiment 3: SVM scores for CNN fine tuned features.*

The results show that allowing the model to tune the drug text embeddings for the classification task leads to results outperforming one of the state of the art system for ADR mention mining. Moreover, comparing the results to CNN-SSWE-T and to the baseline CNN the importance of the domain specific word embeddings appears clearly, since just tuning SSWEs does not lead to the same amount of improvement. Finally it can be noticed that there is as well a large amount of improvement in the score for recall, meaning that fewer tweets are misclassified (non ADR tweets marked as ADR).

5 Conclusion and Future Work

As hypothesized in the introduction this project has shown that a relatively simple CNN architecture which fine tunes domain specific word embeddings can outperform task specific systems, reducing the effort in feature engineering. Moreover since sentiment analysis features do not really improve the overall result it can be claimed that ADRs mention mining should no be addressed as being similar to that task.

With respect to possible feature work, being social media rich of misspellings due to their informal nature, applying a spell checker for text normalization could lead to an improvement in the results. Regarding the exploration of other architectures to obtain document representations, given their effectiveness, it would be possible to exploit CNNs more consistently with respect to their assumptions about the data. Turning to the detail, it is possible to compute character embeddings, which intuitively speaking can be used as the pixels of the sentences. In this way sentences can be treated as images, making possible to use a CNN architecture in its original formulation (two dimensional convolutions) in order to completely preserve the key property of compositionality.

References

V. Abedi, M. Yeasin, and R. Zand. Empirical study using network of semantically related associations in bridging the knowledge gap. *Journal of Translational Medicine*, 2014.

F. Chollet et al. Keras. `https://github.com/fchollet/keras`, 2015.

R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, pages 2493–2537, 2011.

H. Dai, M. Touray, J. Jonnagaddala, and S. Syed-Abdul. Feature engineering for recognizing adverse drug reactions from twitter posts. *Information (Switzerland)*, 2016.

S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *JOURNAL OF THE AMERICAN SOCIETY FOR INFORMATION SCIENCE*, 41(6):391–407, 1990.

R. Ginn, P. Pimpalkhute, A. Nikfarjam, A. Patki, K. OConnor, A. Sarker, K. Smith, and G. Gonzalez. Mining twitter for adverse drug reaction mentions: a corpus and classification benchmark. In *Proceedings of the fourth workshop on building and evaluating resources for health and biomedical text processing*, 2014.

T. Huynh, Y. He, A. Willis, and S. Rüger. Adverse drug reaction classification with deep neural networks. In *Proceedings of COLING 2016, the 26th International Con-*

ference on Computational Linguistics: Technical Papers, pages 877–887, 2016.

Y. Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

M. Kuhn, M. Campillos, and I. L. et al. A side effect resource to capture phenotypic effects of drugs. *Mol Syst Biol.*, 2010.

Q. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on Machine Learning (ICML-14)*, pages 1188–1196, 2014.

B. Liu. Sentiment analysis and subjectivity. *Handbook of natural language processing*, 2:627–666, 2010.

T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

J. Mitchell and M. Lapata. Composition in distributional models of semantics. *Cognitive Science*, pages 1388–1429, 2011.

A. Nikfarjam, A. Sarker, K. OConnor, R. Ginn, and G. Gonzalez. Pharmacovigilance from social media: Mining adverse drug reaction mentions using sequence labeling with word embedding cluster features. *J. Am. Med. Inform. Assoc.*, 2015,.

O. Owoputi, B. O'Connor, C. Dyer, K. Gimpel, N. Schneider, and N. A. Smith. Improved part-of-speech tagging for online conversational text with word clusters. Association for Computational Linguistics, 2013.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

R. Řehůřek and P. Sojka. Software Framework for Topic Modelling with Large Corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 45–50, Valletta, Malta, May 2010. ELRA. `http://is.muni.cz/publication/884893/en`.

D. Tang, F. Wei, N. Yang, M. Zhou, T. Liu, and B. Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1555–1565, Baltimore, Maryland, June 2014. Association for Computational Linguistics. URL `http://www.aclweb.org/anthology/P14-1146`.

Y. Tang. Deep learning using linear support vector machines. *arXiv preprint arXiv:1306.0239*, 2013.

A. Vilhelmsson, T. Svensson, and A. Meeuwisse. What can we learn from consumer reports on psychiatric adverse drug reactions with antidepressant medication? experiences from reports to a consumer association. *BMC Clin Pharmacol.*, pages 11–16, 2011.