Identifying Tweets with Adverse Drug Reactions

COMP90049 - Knowledge Technologies - Project 2
The University of Melbourne

ABSTRACT

In this report, we discuss effectiveness of machine learning algorithms on the problem of tweets containing adverse drug reactions. Firstly, discuss the feature engineering, text normalisation. Then we transition into the analysis phase of the methods, while trying to understand their theoretical differences. We review several machine learning algorithms such as Naive Bayes, Support Vector Machines and Decision Tree. Further, we perform a critical analysis of another study with a similar problem.

KEYWORDS

Classification Techniques; Feature Engineering; Support Vector Machine; Decision Tree; Naive Bayes; Perceptron; Random Forest

1 INTRODUCTION

Capturing of Adverse Drug Reactions (ADR) is of significance importance for multiple reason. The most important being it going unnoticed and thus leading to medical complications. Secondly, this data may not be available elsewhere and offers a valuable perspective. Natural language processing is certainly a challenging task due to the unstructured and informal nature of text. Additionally the complication nature of drugs and their indications is tough to judge. In this paper our focus is on a.set of tweets provided to us. With substantial pre-processing, feature engineering, vector space creation; we attempt to critically evaluate different machine learning algorithms.

2 TEXT NORMALISATION TECHNIQUES

2.1 Text Pruning

This raw text of *train.txt* and *dev.txt* provides us with our first problem. We need to transform our raw text data into a set of features that describe qualitative concepts in a quantitative way, we call this as feature vector space. In our case, that will be a 1/-1 coding strategy: Yes or No. For the this project, we process our features such that they do not include words beginning with # and @ and we are more concerned with English words. We use Gouws et al. [1] for text normalisation.

2.2 Feature Engineering

Before we begin analysing our tokens, we would want to categorise and extract the tokens which we would want to consider as attributes. Our focus is to combine attributes into various n-gram categories, such as 1 Gram, 2 Gram, 3 Gram; defining them as non-gram attributes. We use *tokens.txt* and *dev.txt* for extracting the attributes. We maintain the token in a line-by-line format in $N{-}gramtoken.txt$, where $N = 1, 2, 3$. This is perform in the $n - gram.py$ where we can pass N as a parameter and the script returns a feature vector space of the entire file that is being parsed.

Corpus	Features
train.arff	639KB-3166 tweets
dev.arff	219KB-1076 tweets
test.arff	221 KB-1087 tweets

Table 1: Corpura used for identifying tweets with ADR

Feature subset selection is the process of identifying and removing as many irrelevant and redundant features as possible. This reduces the dimensionality of the data and enables data mining algorithms to operate faster and more effectively. For this we apply **LASSO** (least absolute shrinkage and selection operator) and we make an interesting observation.

In our observation neck, pic, skin, restless and zombie can produce quite a high number of correctly classified instances. To put things in perspective, using the Support vector machine, we received 82.15% correctly classified instances when classifying *dev* keeping *train* as the prepossessing module, while just using particularly these 5 words gives us a staggering, 89.68 % correctly classified instances and 10.316 % incorrectly classified instances. The other parameters such as root mean square error and relative absolute error also take a substantial decrease above 45%.However, it is quite obvious that these 5 features alone may not be sufficient for us to classify the unknown *test* classes and we need some number of instances. However, this observation tells us something about the data i.e. there is a clear bias in the weightage each of the 92 attribute has and perhaps identifying the key attributes can help us in forming new set of attributes.

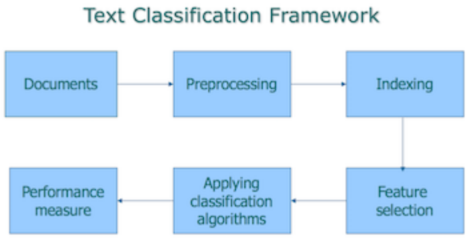
We further explore the 'Select attributes' tab in the *WEKA* explorer, this will allow us to explore rankings the attributes. There are two parameters to observe here *attributeevaluator* and *searchmethod*. We start with exploring the *InfoGainAttributeEval* which evaluates the worth of attribute by measuring the information gain with respect to the class. We use *GreedyStepwise* for the search method, which uses the greedy algorithm for ranking attributes. A threshold greater than 0.0050 is chosen; An overview of the attributes is shown in **Table 2**.

Lastly, we explore synonyms of words and search for approximate matching words using dictionary. Note: we only record those tokens which exists in our twitter files. Using the all the above approaches we get a 3-gram, 48 line token file as our feature selection.

2.3 *Classification* Techniques

When classification is to be performed, we have to seperate data into training and testing sets. Each instance in the training set contains one 'target value' (i.e. the class labels) and several 'attributes' (i.e. the features or observed variables).

COMP90049-Knowlege Technologies, University of Melbourne, 2017
2017. 123-4567-24-567/08/06…$15.00
DOI: 10.1145/000000.000000



Naive Bayes Classification Technique: Naive Bayes classifier, looks for differences of this sort by searching through text for words that are either (a) noticeably more likely to occur in A class, or (b) noticeably more likely to occur in B class. When a word is noticeably more likely to occur in one context rather than the other, its occurrence can be diagnostic of whether a new message is in A or B. If we see a single word that's more likely to occur in A than B, that's evidence that the token as a whole is in class A.

Quite evidently, this approach is quite east of implement. Decent results are returned from this technique.

Support Vector Machine: The goal of Support vector machine (SVM for short) is to produce a model (based on the training data) which predicts the target values of the test data given only the test data attributes. SVM finds a linear separating hyperplane with the maximal margin. The instances that are closest to the maximum hyperplane are called support vectors.

The kernel trick that could be used to solve problems with nonlinear decision boundaries. SVM allows you to use multiple different kernels to find nonlinear decision boundaries.

Decision Trees: The root node in decision tree contains all the documents. Each internal node is a subset of documents separated by one attribute. Each node is further labelled with a predicate which can be applied at the parent. Finally, each leaf node is labelled with a class.Through the decision tree we were able get an accuracy of

The confidence is used to compute the pessimistic upper bound of the error rate and is higher for pruning. J48 does not accept values greater then 0.5.

Perceptron(a review): We use sklearn and numpy to develop a perceptron model as it is a binary classifier. We do this on just our tokens and try to understand the relation between testing and training errors. Further we adjust parameters such as the learning rate (eta). We observe that lower the learning rate higher the training accuracy. We receive an accuracy of 88.97% on our training set. The average test error computes to 0.16% and train error is 0.12%. This process is performed on the basis of heldout validation. Perceptron is introduced as an alternative to existing approaches and can be explored further in future studies.

2.4 Comparison and Review

An overview of the attributes is shown in **Table 3** and **Table 4**. Accuracy is computed as in equation (1).

$$Accuracy = \frac{Total\ correct\ predictions}{Number\ of\ test\ predictions} \tag{1}$$

Precision is computed as in equation (2).

Character	Word position-Word
0.01235	50-me
0.01079	48-makes
0.00924	29-feel
0.00883	40-i
0.00836	46-lozenge
0.00688	76-sick
0.00655	67-prozac
0.00638	42-it
0.00621	49-making
0.00619	63-pain
0.00598	61-olanzapine
0.00591	32-gain
0.00587	93-zombie
0.00543	89-weight
0.00510	65-person

Table 2: Select Attributes

$$Precision = \frac{Number\ of\ correct\ predictions}{Total\ number\ of\ predictions} \tag{2}$$

Recall is computed as in equation (3).

$$Recall = \frac{Number\ of\ correct\ words}{Total\ number\ of\ words} \tag{3}$$

F-measure is computed as in equation (4).

$$F - Measure = \frac{2 * Precision * Recall}{Precison + Recall} \tag{4}$$

Naive Bayes. The low accuracy rate can be attributed to the assumption of class conditional independence. As we know that in practical systems, there is a dependency between variables. We were unable to find a way of modelling these dependencies directly. Upon further research it appeared that Bayesian Belief Networks were to be considered for handling there dependencies. Thus, the assumptions outweigh the ease of implementation when it comes to critical comparison with other techniques. On observing the true positive rate i.e., 0.482 we realise that this is quite high. Further the number of Y classes(52) i.e. predicted Y are very close to the N(59) predicted (Data from the confusion matrix). The higher number of Y terms explain the positive relation with a lower value in correctly classified instances. The false positive rate for N class is 0.518, thus reducing the number of correctly classified instances.

Support Vector. We found SVMs as much slower compared to other techniques. The computational cost of SVM was rather high. This can be explained as large quadratic optimisation have to be solved on a large set of variables. This could be explained by the fact that SVMs training is associated with Lagrangian dual instead of primal problem. In our observation, SVM produce very accurate classifiers, the highest among all the techniques discussed. It would be important to note that SVMs generally are less overfitting and robust to noise.

SVM map to a high dimensional space, thus having a strong bias in that space. Thus, there is a rich proportion of optimisation in SVM. Under WEKA we found that when 10-fold cross validation is

Type	Correctly classified instances	Precisson	Recall	F-Measure
Zero-R	89.4052%	0.799	0.894	0.844
Naive Bayes	82.1561%	0.866	0.822	0.840
Decision Tree(J48)	89.3123%	0.860	0.893	0.864
Random Forest	89.4052%	0.860	0.894	0.861
Support Vector Machines	89.5911%	0.866	0.896	0.865

Table 3: Performance comparison of classification strategies without new tokens

Type	Correctly classified instances	Precisson	Recall	F-Measure
Zero-R	89.4052%	0.799	0.894	0.844
Naive Bayes	82.1561%	0.866	0.822	0.840
Decision Tree(J48)	89.5911%	0.866	0.896	0.867
Random Forest	89.5911%	0.865	0.896	0.858
Support Vector Machines	90.0558%	0.878	0.901	0.871

Table 4: Performance comparison of classification with new tokens

performed on the Normalized Poly Kernel, we obtain 1161 support vectors of which 86.4% are cached.

Decision Trees. The default confidence value in J48 is 0.25. It seems that there is an inverse relationship between the confidence pruning and tree pruning. Higher the value of confidence pruning, less the tree is pruned. Decision tree tries to make the optimal tree at every step. An observation between the time to train versus the time to test is made. The time for training is substantially higher, perhaps sometimes the highest across all the techniques. However, the testing is almost instantaneous once the model is trained.

The confusion matrix of decision tree reveals the bias in information gain which favours attributes with more level. In the confusion matrix we see large amounts of N predicted where the classifier has a True Positive of 0.986.

We make the observation on the percentage split of decision tree. By default, WEKA maintain this number as 66% and we are able to make 1076 observations. This gives us an accuracy of 88.29%. However, say we increase the percentage split to 90%, then the number of instances are decreased to 317 with an accuracy of 90.85%. This shows us the impact of pruning of the trees and helps us validate our understanding of generating better results with smaller trees.

3 *MINING TWITTER FOR ADVERSE DRUG REACTION MENTIONS:* A CORPUS AND CLASSIFICATION BENCHMARK(RESEARCH REVIEW)

In this section, we review and try to understand the work of Sarker and Gonzalez [3] on the same problem. The corpus was manually annotated and created through extraction of tweets related to 74 drugs, using their brand and generic names, and phonetic misspellings. These were annotated for the presence of ADRs (a binary attribute for each tweet), the location/span of the reaction mentions, and the UMLS concept IDs for the ADR mentions. Further, they extended the drug list to include misspelled drug names. This was critical to obtaining relevant tweets, as drug names are often misspelled in social media. They generated the misspellings through a phonetic spelling filter.

Additionally, the corpus was not only set for the presence or absence of ADR mentions, but also to identify the span of the expressions conveying individual ADRs, and to map them to formal medical terminology. Two annotators were manually annotated the processed tweets: binary annotation of ADRs and full ADR annotations. The result would be for, 'weight gain', 'gained 20 pounds', 'put on too much weight', or 'fat fat fat' would all be annotated to a general concept ID for 'weight gain'.

For the ADR detection, they used NB and SVM classifiers for the binary classification task. In SVM only the was used linear kernel with no attempts at parameter optimisation. Three sub datasets were created changing the proportions of HasADR and NoADR. In dataset1 both the parameters were equal, in the second HasADR was greater than NoADR and in the third NoADR was greater.

Mapping an annotated concept to semantically equivalent UMLS concept IDs for an agreement was needed. Inter-annotator agreement (IAA) using Cohen's kappa was computed for binary annotations. The largest source of error came from the differences in concept IDs chosen. Some term used by the user, such as 'hangover' may or may not be an ADR. Also the span has to be considered.

Performance of accuracy in SVM and NB are compared. Majority labelling is performed in which every test instance to the majority class in the test set. Both classifiers perform better than the majority labelling baseline. An observation that the classifier accuracies increase slightly as the proportion of the majority class increases is made. The study is concluded as a stepping stone into classification performance while maintaining the goal of deep semantic learning on tweets.

REFERENCES

[1] S. Gouws, D. Metzler, C. Cai, and E. Hovy. Contextual bearing on linguistic variation in social media. *Proceedings of the Workshop on Languages in Social Media*, pages 20–29, 2011. URL http://dl.acm.org/citation.cfm?id=2021109.2021113.

[2] G. R, Pimpalkhute, P. Nikfarjam, A. Patki, K. O. Connor, A. Smith, 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. URL https://www.researchgate.net/profile/Abeed_Sarker/publication/280301158_Mining_Twitter_for_adverse_drug_reaction_mentions_a_corpus_and_classification_benchmark/links/56d205b608ae85c8234ae39d.pdf.

[3] A. Sarker and G. Gonzalez. Syntactic normalization of twitter messages. *Portable automatic text classification for adverse drug reaction detection via multi-corpus training. Journal of Biomedical Informatics.*, pages 196–207, 2015. URL https://www.ncbi.nlm.nih.gov/pubmed/25451103.