COMP90049 Report

Anonymous

1 Introduction

This project aims to develop and analyse models for predicting the sentiments of tweets and assess the effectiveness of various machine learning algorithms in predicting tweet sentiment. Meanwhile, research on the relation between unlabelled data and Twitter sentiment classification performance(research Question 1).

The data set used in this project is derived from the report written by (Blodgett et al., 2016). It is given by three feature representations. Each feature representation has four sets of data in the format of pickled Pandas Dataframe that are used for training, developing and testing purposes.

The first "Raw Tweet" representation has the Tweet data in the format of a single string.

The second "TFIDF" representation makes each raw Tweet data string transformed into a 1000 dimension feature vector with each dimension representing the occurrence of the top 1000 words(excluding stop words) in the full raw Tweet data with the highest TFIDF values.

The third "Embedding" representation transformed each raw Tweet data with the Sentence Transformer language model(Reimers and Gurevych, 2019) and turned the raw Tweet string into a 384 dimension-vector representing the meaning of each Tweet with similar Tweets located closer in the vector space.

2 Literature review

2.1 Dialectal Performance Gap

The data set this project used is from the article written by (Blodgett et al., 2016). The main content of this article is: Due to a lack of resources exist for dialectal natural language processes, (Blodgett et al., 2016) conducted a case study in dialect identification, characterisation, and language technology adaptation for the dialectal language of African-American English(AAE) on Twitter and proposed a distantly supervised model to identify African-American English from demographics and geographically located messages.

In detail, they have discovered the existence of a performance gap between African-American-like English and Caucasian English on existing language identification and language dependency parsing tools. Thus, they build their distantly supervised model in a form of an *Ensemble Classifier* using *languid.py* an open-source language identification tool with posterior inference to predict demographic-language proportions for identifying African-American English and standard(Caucasian) English. The model employs demographic correlation with speaker communities and it managed to close the gap between the performance difference on dialectal and standard text for natural language process tools.

The main takeaway of this article is that it raises the awareness of the performance gaps existing between dialectal and African-American English in natural language process tools and made us consider the dialectal difference factor during the conduct of this project.

2.2 semi-supervised learning performance

Training with supervised learning algorithms requires using labelled data only, but labelling data is both costly and time-consuming. In addition, a limited amount of labelled training data could lead to poorly trained classifiers. But according to (Masud et al., 2008), using a semi-supervised algorithm can overcome the need to label data fully and the semi-supervised algorithm can outperform supervised algorithms.

They build a classifier model "SMScluster" using both labelled and unlabelled data and using an ensemble of micro-clusters technique and k nearest neighbour algorithm to build the model. It uses a small amount of labelled data for training and the results show that it outper-

forms classification algorithms that use twenty times more labelled data.

The main takeaway is that supervised learning is not the only way to achieve high accuracy classification, semi-supervised learning can be more efficient under certain circumstances.

2.3 unlabelled data with semi-supervised learning

Furthermore, labelled data can also cause problems. It was found that if only limited labelled data is available for training classification systems, the results show high variability and the performance of the classifier is highly dependent on the reliability of the labelled data(Gabrys and Petrakieva, 2004). but if use selection techniques like selective sampling where it only uses a suitable set of data, then the mean classifier performance and the reduction of classification variance would be highly improved.

3 Methods

3.1 Feature Engineering

The newly engineered features added to the embedding dataset are lexicon rating scores of the raw tweeter sentence on positive sentiment, neutral sentiment and negative sentiment and a compound score that summarised the previous three feature scores to indicate an overall score on sentiment.

These 4 newly added features were calculated using the VADER sentiment analysis tool, which is specifically attuned to sentiment expression in social media. The reason for including the model-calculated sentiment score is because the vector in the embedding data set is already computed with a pretrained language model called Sentence Transformer(Reimers and Gurevych, 2019), the vector generated represents the meaning of each tweet.

But meaning does not always associate with the same sentiments, one sentence can mean different things under different contexts, thus including features directly associated with sentiments was considered helpful in improving the prediction performance of models.

3.2 models

The model used in this project includes: zeroR, Gaussian Naive Bayes, logistic regression, K-means, multi-layer perceptron, stacking classier, boosting and semi-supervised learning.

The reason for using these models is that the research question "Does unlabelled data improve Twitter sentiment classification" proposed that different levels of supervision were used in different algorithms, so it is a good idea to utilise the different amounts of supervision and combine labelled and unlabelled training data.

The first set of models implemented was the supervised learning models. Because more knowledge about the supervised learning model was taught during the lectures especially Naive Bayes and Logistic regression.

The first model implemented was ZeroR. It is a supervised model that acts as a baseline machine learning model. It is simply looking at the labels in the training data set, calculating the prior probability and selecting the majority label as the prediction label/class.

The second algorithm implemented was another supervised learning model-Gaussian Naive Bayes. The Gaussian Naive Bayes model is a linear probabilistic generative classifier and it uses Bayes rules to model the P(x,y) using likelihood$P(x|y)$ and prior p(y) with the use of the maximum likelihood principle to find the best parameters. It has 3 assumptions: feature independent, instance independent and distribution in train data is the same in testing data. The reason why this algorithm was chosen is that it is well known and a simple algorithm for classification tasks in machine learning.

The third algorithm implemented was also a supervised learning model-Logistic regression. The logistic regression model is a liner probabilistic discriminative classifier, it directly predicts the $P(y|x)$ and iteratively optimised the parameter. The assumption it has is that it assumed feature correlation. The reason why this algorithm was chosen is that it is similar to Naive Bayes but it can typically perform better than Naive Bayes due to fewer assumptions and a more direct production method.

The fourth algorithm implemented was Kmeans, It is an unsupervised, exclusive, deterministic, partitioning, batch clustering learning model. It typically uses the mean of the points in the cluster as a centroid and assigns each instance to the closest centroid, then computes the new centroids in the new cluster, and repeats until the centroid does not change. It is relatively efficient and can be extended to hierarchical clustering but it only works for homogeneous(convex)datasets and it tends to coverage to local optimal and needs to choose k in advance. This algorithm is chosen because it acts

as an unsupervised algorithm for comparison.

For the multi-layer perceptron neural network model, it was built with the hyperbolic tangent function activation and stochastic gradient descent solver with 3 hidden layers each with a size 4. The activation, solver and hidden layer size were chosen based on an iterative process that tested all the combinations of activation, solver and hidden layer size. The reason to choose this is that it is a universal approximator and it automatically learns features.

For the stacking classifier, I simply combine the Gaussian Naive Bayes and Logistic regression model using the stacking classifier function in sklearn. The reason for using this is simply wanted to achieve higher performance.

For the boosting classier, it uses different weights on training data and uses multiple weak models to make a stronger model. The reason for using this is that it reduces bias and can improve performance.

For semi-supervised learning, I combined the unlabelled embedding data with the labelled embedding data and trained the model using Naive Bayes, logistic regression and multi-layer perceptron separately with confidence thresholds 65%, 75%(default), 85%, 90%.

3.3 Evaluation matix

The evaluation metrics used include: mean accuracy, precision, recall and F1 score. Precision is how often the model is correct when predicting the interested label and Recall is the proportion of truly interested instances correctly identified.F1 score balanced precision and recall, 1 means precision and recall are equally important. The reason for using this matrix is that using accuracy, precision, recall and F1 score is a common approach in evaluating the performance of a model.

4 Results

models	Accuracy	Pricision	Recall	F1
ZeroR(emb)	0.5	-	-	-
ZeroR(tfidf)	0.5	-	-	-
Gaussian NB(emb)	0.6147	0.6147	0.6181	0.6175
Gaussian NB(tfidf)	0.6452	0.6452	0.6452	0.6452
LR(emb)	0.6982	0.6982	0.6985	0.6983
LR(tfidf)	0.6787	0.6787	0.6787	0.6787
Kmeans(emb)	0.0210	0.578	0.5927	0.5954
Kmeans(tfidf)	0.020	0.572	0.5981	0.60
MLPerceptron(emb)	0.6997	0.6997	0.7	0.7
stacking(emb)	0.6995	0.6985	0.6999	0.6996
boosting(emb)	0.6957	-	-	-

Table 1: Evaluation matrix

models	Accuracy	Pricision	Recall	F1
LR(emb,top 25%)	0.6842	0.6842	0.6846	0.6844
LR(emb,50%)	0.688	0.688	0.6882	0.6881
LR(emb,75%)	0.693	0.693	0.6934	0.6931

Table 2: Evaluation matrix after feature selection

models	Accuracy
Gaussian NB(emb extra)	0.62025
LR(emb extra)	0.69175
extra features alone	0.5695

Table 3: Evaluation matrix after feature Engineering

models	Accuracy	Pricision	Recall	F1
Gaussian NB(emb,75%)	0.5882	0.5882	0.5982	0.5990
Gaussian NB(emb,85%)	0.588	0.588	0.5981	0.5988
LR(emb,65%)	0.7015	0.7015	0.7023	0.7018
LR(emb,75%)	0.7027	0.7027	0.7033	0.7029
LR(emb,90%)	0.697	0.6970	0.6971	0.6970

Table 4: Evaluation matrix for semi models

5 Discussion

From just using a single algorithm as a prediction model, we gained the result in (Table 1). The ZeroR model shows an accuracy of 0.5, meaning that the sentiment label in the dev file is evenly distributed, with 50% positive and 50% negative. Gaussian Naive Bayes and Logistic Regression are both linear classifiers, they both compute $P(Y \mid x)$ to predict class. They both have parameters to fit the problem and use the maximum likelihood principle to maximise

it. Also, they both require input instances in the format of pre-defined features. The difference is that Naive Bayes is a generative model means that it tried to model the actual distribution of a class, whereas Logistic regression is a discriminative model which tries to model the decision boundary between classes. They also calculate different things, Logistic regression calculates $P(Y \mid x)$ directly whereas Naive Bayes calculates P(x,y) using the Bayes rule. Logistic regression's performance is better here than Naive Bayes because it does not require conditional independence assumption. The reason why k-means has low accuracy but high precision and recall is that the dev dataset is not homogeneous, the k-means can't handle non-convex clusters or clusters of different densities. For stacking and boosting, stacking is a way to combine predicting models but the accuracy does not outperform the models involved. The reason can be that they are all trying to predict the same thing, which could introduce more noise to the system.

For feature selection results (Table 2), all performances did not exceed the original performance. This shows that the numbers in the embedding vector help reduce the bias and increase the prediction performance.

For feature engineering (Table 3), Gaussian Naive Bayes improved accuracy with the new data set that has the extra feature. This is because adding more features increased the $P(Y \mid x)$ and reduced P(x), thus increasing the $P(Y \mid x)$which improved the prediction. But the dataset reduced the performance of Logistic Regression. The reason for this is that more feature means adding more parameters to the model and this made the learning more difficult which result in decreasing the performance.

For semi-supervised learning (Table 4), the model using logistic regression with a 75% confidence threshold achieved a better prediction performance than previous models. The reason is that the semi-supervised model not only uses the label probability generated from the training data, it also utilises the prediction result for the unlabelled data. Then, it combines the labelled data and the pseudo labelled new data as a new training set. This not only increased the train data size but also combines both the descriptive and predictive results together which made the prediction more accurate.

6 Conclusions

In conclusion, this project has examined the performance of machine learning models including zeroR, Gaussian Naive Bayes, logistic regression, K-means, multi-layer perceptron, stacking classier, boosting and semi-supervised learning. Furthermore, we discovered that semi-supervised learning can perform better than supervised learning which means unlabelled data can improve Twitter sentiment classification if the classification algorithms can utilise and take advantage of both the unlabelled data and labelled data.

References

Blodgett, S. L., Green, L., and O'Connor, B. (2016). Demographic dialectal variation in social media: A case study of african-american english. *arXiv preprint arXiv:1608.08868*.

Gabrys, B. and Petrakieva, L. (2004). Combining labelled and unlabelled data in the design of pattern classification systems. *International journal of approximate reasoning*, 35(3):251–273.

Masud, M. M., Gao, J., Khan, L., Han, J., and Thuraisingham, B. (2008). A practical approach to classify evolving data streams: Training with limited amount of labeled data. In *2008 Eighth IEEE International Conference on Data Mining*, pages 929–934. IEEE.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.