Sentiment analysis using supervised machine learning

Dinesh Kannan
College of Computer and Information Sciences (CCIS)
Northeastern University
kannan.d@husky.neu.edu

Abstract

In this paper, we aim to predict the emotion of a piece of text, classifying them as a '*positive'* or a *'negative'* emotion. We are also going to use some supervised learning algorithms and compare the effectiveness of the predictions between the algorithms on the same data set. The dataset used here is same as the one used for [1], amalgamating text reviews from www.rottentomatoes.com and plot summaries from www.imdb.com. The classifiers used here are a Perceptron classifier, a basic Naïve-Bayes classifier and a Naïve-Bayes classifier that uses Information Gain for feature selection.

Introduction

The open internet holds a lot of text that contains opinions of people. Extracting the sentiment behind a piece of text lets us gain an idea of the public opinion about a topic.

Businesses use sentiment analysis to determine their marketing strategy, test business kpis, improve customer service etc. A common interest has always been to find out if an opinion is '*positive'* or '*negative'*. From a computational perspective, this is a classification task which takes in a piece of text as input and outputs a label, either '*positive'* or '*negative'* denoting the emotion of the text. However, sentiment analysis is quite difficult because of the complexity of the English language. People have their own style of phrasing and expressing. For example, the sentences '*Suicide squad is a bad movie'* and '*Good fellas is not bad'* convey opposite emotion. The latter sentence must be categorized as a '*positive'* emotion though it does not contain any positive words like 'good', 'better', 'best' etc. It might be a good idea to include word n-grams in our model for classification, but we still don't know the value of n and it can get complex and computationally intensive as n grows.

In this paper, we are going to consider model based approaches for sentiment analysis and compare the accuracies of those approaches.

Dataset

Comments/opinions about a topic can be found in many open internet forums. Since we are using supervised learning algorithms, it is useful if we have a pre-classified dataset into '*positive'* and '*negative'* emotion texts.

In this paper, we are using a dataset that appears in [1] for our train-dev-test tasks. This dataset combines user opinions/reviews from www.rottentomatoes.com and plot summaries from the Internet Movie Database (IMDb) www.imdb.com.

The dataset consists of three sub-directories, *dev, test* and *train*. The *test* directory consists of a collection of .txt files of text. The other two directories consist of two sub-directories *'pos'* and '*neg',* containing positive and negative reviews respectively. Training is done with '*train'* and *'dev'* sub-directories, since it is pre-classified. We finally test our trained model with files in '*test'* sub-directory.

Background

In this section, we provide necessary background for

1. Primitive Naïve-Bayes approach
2. Information Gain
3. Perceptron classifier approach
4. Add-one / Laplace smoothing
5. Precision / Recall

Naïve-Bayes classifier

Naïve-Bayes is a simple probabilistic classifier based on Bayes theorem. Naïve-Bayes classifier assumes that there is strong independence between the chosen features. Naïve-Bayes classifier is captured by the equation below,

$$p(C_k|\mathbf{x}) = \frac{p(C_k)\ p(\mathbf{x}|C_k)}{p(\mathbf{x})}$$

This is a basic re-arrangement of a Bayes theorem equation where we estimate the probability we don't know using the conditional probability we already know.

In the training phase (described later), we build a model, i.e. the probability of the occurrence of features given a label. We later use that model to predict in our dev and test phase. For prediction, we estimate the probability of a label given the probability of features.

Information Gain

Information Gain is one of the popular feature selectors used in machine learning, especially for text classification. Information Gain is inspired from Information Theory principles. As the name suggests, it gives us the idea of how much information is gained about a label when a feature is observed. For example, observing the word "bad" suggests that it most probably belongs to the *negative* class. This way, we can effectively predict using a few features that is more informative. This is very useful for text classification because we don't want to consider all the words (like "the", "a", etc.) as features but only the informative words (like "good", "terrible", etc.).

Information Gain derives from the notion of Entropy which is a measure of expected information in a probabilistic distribution. So roughly, information gain measures the change in the entropy when a feature occurs.

This is represented by the equation below,

$$IG(T,a) = H(T) - \sum_{v \in vals(a)} \frac{|\{\mathbf{x} \in T | x_a = v\}|}{|T|} \cdot H(\{\mathbf{x} \in T | x_a = v\})$$

T – Training examples
a – feature
H () – Entropy

This paper, for simplicity assumes a multiple-Bernoulli sample event space. Hence, it can be formulated as,

$$\begin{aligned} IG(w) &= H(C) - H(C|w) \\ &= -\sum_{c \in \mathcal{C}} P(c) \log P(c) + \sum_{w \in \{0,1\}} P(w) \sum_{c \in \mathcal{C}} P(c|w) \log P(c|w) \end{aligned}$$

H (C) – Entropy
H (C | w) – Conditional entropy

Perceptron classifier

The perceptron classifier is a linear classifier, which takes in a vector of features and maps them to a binary value (0 or 1). The classifier has a weight vector that stores the weight of each feature and multiplies them with the input feature vector.

Formally, a perceptron classifier is defined as below.

$$y = \begin{cases} +1 & \text{if } \ w \cdot f(x) \geq 0 \\ -1 & \text{if } \ w \cdot f(x) < 0 \end{cases}$$

In our experiments, a positive value represents the *positive* class and a negative value represents the *negative* class.

Add-one smoothing

Add-one smoothing/Laplace smoothing is a smoothing technique used in this experiment for making calculations simpler. Using this, we don't ever face situations such as dividing by 0, we add one to the denominator so that the calculations are consistent, hence the name add-one smoothing.

It is formally defined as below.

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

Precision/Recall

Precision and Recall are two evaluation metrics used in this paper. They are formally defined as below.

$$\text{Precision} = \frac{tp}{tp + fp}$$

$$\text{Recall} = \frac{tp}{tp + fn}$$

Related work

Text classification can be complicated however it wants to be. Since one can structure their sentences in many ways,

we need a sophisticated algorithm to understand the structure and syntax of sentences and accurately assign a label. It is because of this complexity; sentence classification is very interesting to researchers in the field of Natural Language Processing.

A better/sophisticated approach that could have been used in this paper is using a Convolutional Neural Network (CNN). Due to its effectiveness and popularity, deep learning almost always has a better accuracy. We could have performed several layers of convolutions on sentences followed by max-pooling (for sampling) to arrive at a single class which is our final prediction. We could have also used several layers of neural networks as in a traditional feed-forward network architecture. These approaches are better because deep learning is effective even with a small training dataset.

Project Description

In this paper, we have used three different approaches for the sentiment analysis task.

Firstly, we use the Naïve-Bayes approach (formally presented earlier) to train our model, evaluate accuracy using dev folder and assign scores for positive and negative classes in test folder.

Secondly, we use Information Gain (formally presented earlier) to find out top words that are informative about each class and use only those features for our evaluation and further analysis. This provides a big jump in efficiency because we are getting rid of a lot of non-informative words for our task. This helps to keep our model small and precise.

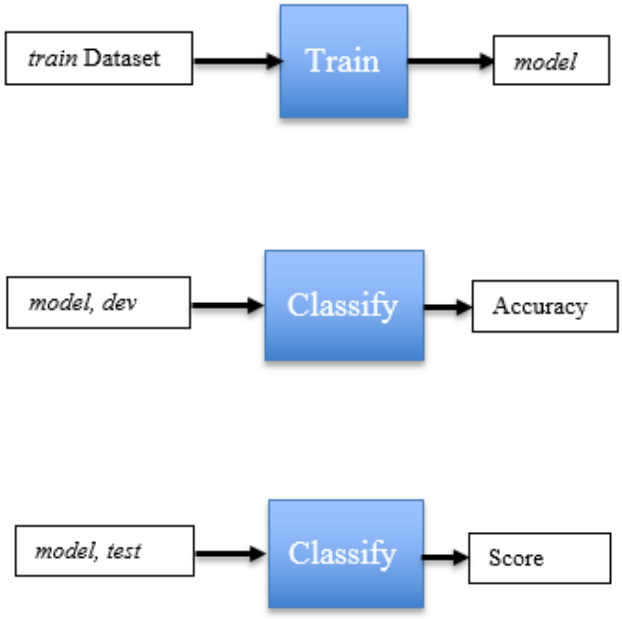
Thirdly, we apply the perceptron classifier (formally presented earlier) to see how text classifier works out for a linear classifier. The training phase assigns a weight to each feature depending on how important the feature is for this task. We finally test this on our dev folder and further evaluate and analyze accuracy as we did for our previous approaches.

We use add-one smoothing to make calculations simple and consistent. Evaluation metrics chosen for our experiments are precision, recall and correctness-proportionality chart.

Experiments

All the three chosen algorithms use the *train* folder in the dataset to learn a *model* from the training set. We then use the learnt *model* to calculate the accuracy of the algorithm using the *dev* folder. We finally test our algorithm on our *test* folder containing files of text reviews. We record the score that our algorithm assigns to *positive* and *negative* labels against each file in the *test* folder.

The entire pipeline of our experimentation is illustrated below,



The performance of algorithms is gauged using several metrics as discussed earlier. They are presented below.

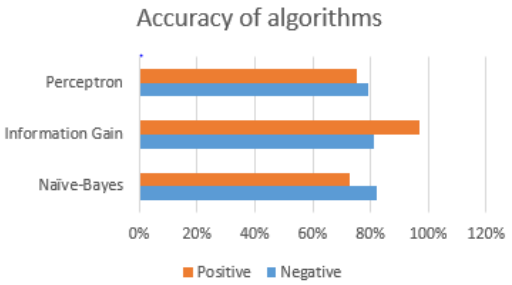
Accuracy of Algorithms

To measure the accuracy of an algorithm, the percentage of correctness to predict each *positive* and *negative* class is observed on the *dev* dataset (because we know the correct label already), i.e., The *positive* and *negative* texts are fed into the algorithms separately with the trained model and the correctness is observed.

The accuracy is presented below,

	Negative	Positive
Naïve-Bayes	82%	73%
Information Gain	81%	97%
Perceptron	79%	75%

The accuracy is illustrated below,



Observations

It suggests that using Information Gain as a feature selector pushes up the accuracy for both the labels. One possible explanation for this is that by using Information Gain we are not making uninformative words contribute to the final score for the labels. For example, if P ("the" | negative) > P ("the" | positive) in our training set, encountering "the" while testing makes the possibility bend more towards the negative class. But it might belong to the positive class.

Accuracy for Perceptron and Naïve-Bayes classifier are almost the same for both the classes. This might be because they consider all the words as features and weights them (by probability in Naïve-Bayes and a trained weight in a perceptron).

Precision

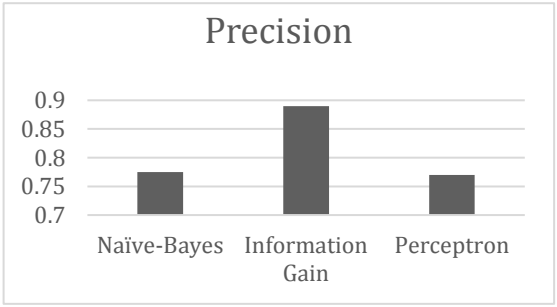
Precision is the proportion of correctly classified training instances when compared to the whole training set. This gives us the overall estimate of correctness for both *positive* and *negative* classes.

Calculating the precision of the results is important because, it considers the overall dataset and estimates correctness for all the labels.

The precision estimate for the considered algorithm is presented below.

APPROACH	PRECISION
Naïve-Bayes	0.775
Information Gain	0.89
Perceptron	0.77

Graphically it is represented as follows.



Observation

Unlike previous metrics, we do not consider the two labels separately. We consider the whole dataset to see how it performs generally.

Here too, Information Gain approach scores more than the other two. The possible explanation to this is like the one provided earlier. Irrespective of the class we consider, we only want the informative features to contribute to the final score. In the other two approaches, un-informative features bring down the final score, leading to wrong prediction in some cases.

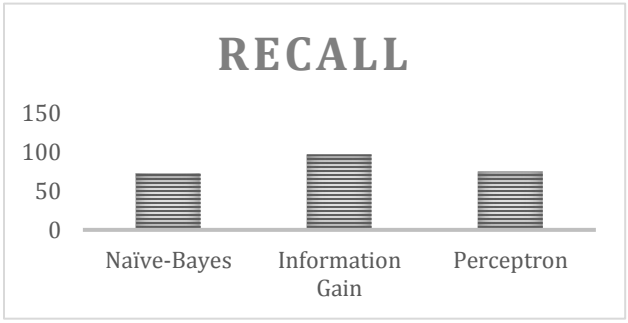
Recall

Recall is the measure of the proportion of relevant classes that was correctly identified in the testing, i.e. the proportion of true positives amongst true positives and false negatives.

The recall estimate is presented below,

ALGORITHM	RECALL
Naïve-Bayes	73
Information Gain	97
Perceptron	75

This is illustrated as,



Observation

We can observe that the proportion of *true positives* classified is more in the Information Gain approach. It also suggests that it is difficult to distinguish a significant portion of instances as positive or negative review. Information Gain breaks the bias in such cases by weighing up important features and promoting to contribute for the final score.

Conclusion

From our experiments, we have learnt that a Perceptron classifier performs like a Naïve-Bayes classifier for text classification tasks. This suggests that there is a difference only in the weighting schemes adopted by both the algorithms. Weight for a perceptron classifier is continually learnt using train-dev datasets and that for a Naïve-Bayes classifier is the simple probability distribution given a label.

However, significant accuracy improvements can be found when we intelligently select features that are important to the task. For our text classification task, we found Information Gain to be a useful feature selector. This also makes sense intuitively because of the sense that uninformative features push up the final score of the classifier in the other approaches.

References

[1] Pang, Bo., and Lee, Lillian. 2004. *A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts.* Proceedings of the *Association for Computational Linguistics*.

[2] Roobaert, D., Karakoulas, G., and Chawla, N. 2003. *Information Gain, Correlation and Support Vector Machines*. NIPS 2003 Feature Selection Challenge

[3] Hu, M., and Liu, B. 2004. *Mining and summarizing customer reviews.* Proceedings of the 10th ACM SIGKDD international on Knowledge discovery and data mining

[4] Pang, B., Lee, L. and Vaithyanathan, S. 2002. *Thumbs up? Sentiment classification of product reviews*. Proceedings of the ACL-02 conference on Empirical methods in NLP

[5] Dave, K., Lawrence, S. and Pennock, D. 2003. *Mining the peanut gallery: Opinion extraction and semantic classification of product reviews.* Proceedings of the 12th international conference on World Wide Web, pp. 519-528