**PROJECT REPORT:**

**SENTIMENT ANALYSIS USING N-GRAM AND SVM**

**Group Members:**

**Daniel Santoso / 2201756506**

**Boban Nathaniel Seputra / 2201762540**

**Luwis Lim / 2201761771**

**Steven Odolf Yuwono / 2201758045**

1. **INTRODUCTION**

Sentiment relates to the meaning of a word or sequence of words is usually associated with an opinion or emotion. And analysis is the process of looking at data and making inferences; in this case, using machine learning to learn and predict whether a movie review is positive or negative. Sentiment Analysis refers to the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information.

1. **METHOD**
2. **N-Gram**

Statistical language models are the type of models that assign probabilities to the sequence’s words. N-gram is a contiguous sequence of *n*-items from a given sample of text or speech. The items can be letters, words, phonemes, syllables, or base pairs according to the application. The *n*-grams typically are collected from a text or speech corpus. When the items are words, *n*-grams may also be called shingles.

*n*-grams can also be used for sequences of words or almost any type of data. For example, they have been used for extracting features for clustering large sets of satellite earth images and for determining what part of the Earth a particular image came from. They have also been very successful as the first pass in genetic sequence search and in the identification of the species from which short sequences of DNA originated.

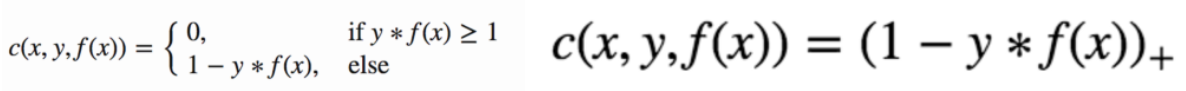
Diagram, schematic

Description automatically generatedConsider an *n*-gram where the units are characters and a text with *t* characters. The space this *n*-gram requires is exponential:

1. **Support Vector Machine**

The objective of the support vector machine algorithm is to find a hyperplane in an N-dimensional space that distinctly classifies the data points. To separate the two classes of data points, there are many possible hyperplanes that could be chosen. Our objective is to find a plane that has the maximum margin, that is the maximum distance between data points of both classes. Maximizing the margin distance provides some reinforcement so that future data points can be classified with more confidence.

In the Support Vector Machine algorithm, we are looking to maximize the margin between the data points and the hyperplane. The loss function helps maximize the margin is hinge loss.



The cost is 0 if the predicted value and the actual value are of the same sign. If they are not, we then calculate the loss value. We also add a regularization parameter the cost function. The objective of the regularization parameter is to balance the margin maximization and loss. After adding the regularization parameter, the cost functions looks as below.

Diagram, schematic

Description automatically generated

Now that we have the loss function, we take partial derivatives with respect to the weights to find the gradients. Using the gradients, we can update our weights.

Text

Description automatically generated

1. **sklearn.feature\_extraction.text.CountVectorizer**

Convert a collection of text documents to a matrix of token counts. This implementation produces a sparse representation of the counts using scipy.sparse.csr\_matrix.

1. **sklearn.model\_selection.train\_test\_split**

Split arrays or matrices into random train and test subsets. Quick utility that wraps input validation and next ( SuffleSplit ().split ( X, y )) and application to input data into a single call for splitting (and optionally subsampling) data in a oneliner.

1. **sklearn.svm.LinearSVC**

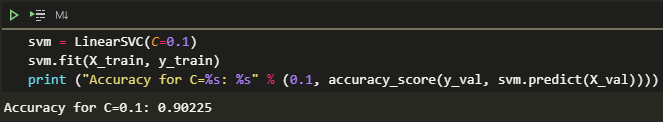
Linear Support Vector Classification is similar to SVC with parameter kernel = ‘linear’, but implemented in terms of liblinear rather than libsvm, so it has more flexibility in the choice of penalties and loss functions and should scale better to large numbers of samples. This class supports both dense and sparse input and the multiclass support is handled according to a one-vs-the-rest scheme.

1. **sklearn.metrics.accuracy\_score**

In multilabel classification, this function computes subset accuracy. The set of labels predicted for a sample must exactly match the corresponding set of labels in y\_true.

1. **RESULTS AND DISCUSSION**

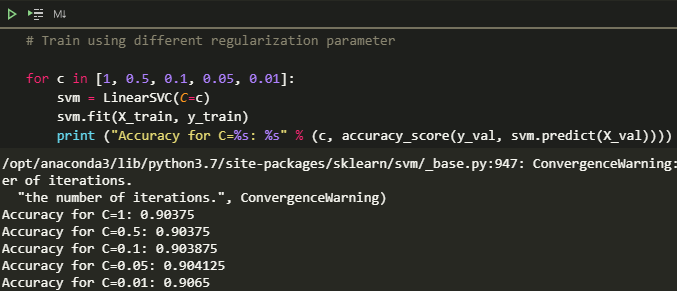
This is the result after training with the dataset that we have.



This is the training section. Before this, we process the data first by removing the stop words ('in', 'of', 'at', 'a', 'the') then vectorize the dataset using n-gram (combination of adjected words or letter of length n that you can find in dataset per text) and split it to make dataset for train and test. Dataset is now ready to be processed using SVM. SVM train the dataset that we processed, and the accuracy is 0.90225 which is good for the algorithm because it means that the algorithm can predict movie review accurately. But we had a speculation that overfitting may be occurred because the accuracy is just too high to be true. Moving onto the test, so the second time we try to use test dataset to test the algorithm. Here is the result:



The result is like the train one, the accuracy that we got is 0.9028 which is surprisingly high algorithm can predict the review with 90% accuracy which is good. Like the train one, we had a speculation that overfitting may be occurred because the accuracy is just too high at this rate. After debating, the conclusion that we had is the algorithm is just too good so there is no overfitting. There is something that we can change to make this algorithm even better than before, that is C (regularization parameter). Basically, C is useful to control fitting parameter. As the magnitudes of the fitting parameters increase, there will be an increasing penalty on the cost function. This penalty is dependent on the squares of the parameters as well as the magnitude of $\lambda$. We tried to play with the C, and this is the result:



As you can see, the accuracy is increasing when we use smaller C. The best Accuracy is 0.9065 with C = 0.01.

1. **CONCLUSION**

The conclusion of our test using sentiment analysis for movie reviews turns out to be good. Because the accuracy that we got is 0.9065 which means that the algorithm can predict 9/10 cases. There are many methods to be used besides N-Gram and SVM, for example BERT is now giving better state-of-the-art results.