Aspect-Based Sentiment Analysis Report

Authors: Chinmay Gangal, Kislaya Singh Kumar

ABSTRACT

Sentiment analysis is a task to determine the positivity, negativity or neutrality of a given *text* based on given *aspect term*. The task assigned to us is to analyse a dataset, containing *text and aspect term information*. This project aims at analysing the sentences/text, understanding various approaches to tackle the task and finally building a classifier capable of determining the sentiment of the provided text/sentence. We examine various text cleaning techniques, machine learning models and discuss their respective merits.

INTRODUCTION:

While sentiment analysis provides fantastic insights, and has a wide range of real-world applications, the overall sentiment of a piece of text won't always pinpoint the root cause of an author's opinion.

Certain types of documents, such as customer feedback or reviews, may contain fine-grained sentiment about different aspects about a product or service that are mentioned in text. For instance, a review about a hotel may contain opinionated sentences about its staff, beds and location. This information can be highly valuable for understanding customers' opinion about a particular service or product.

Aspect-Based Sentiment Analysis makes it easier to identify and determine the sentiment towards specific aspects in text. This project aims at analysing the corpus, understanding various approaches to tackle the task and finally building a classifier capable of determining the sentiment of this corpus. We examine various text cleaning techniques, machine learning models and discuss their respective merits.

Data Details:

Given training data has following columns:
Column A: review sentence id
Column B: review sentence
Column C: aspect term in the sentence
Column D: aspect term location
Column E: sentiment label

Task: Given an aspect term (also called opinion target) in a sentence, predict the sentiment label for the aspect term in the sentence.

TECHNIQUES:

Data Cleaning:

The 'text' and 'aspect term' columns contain a lot of symbols which are detrimental to the task of sentiment analysis. Removal of < > (em tags), _ (underscore), " (quotes) and non-ascii codes (emoji) was essential. Additionally, we explored the conversion of emoticons, which strongly represent an emotion, into specific words to expose the underlying sentiment, but found that it simply adds noise to the corpus. Perhaps this was due to various sarcastic emoticons being used by the subjects.

Features:
We initially tried several features like complete n-gram vectorization of complete text data but came down to following features, as they played most important role in determining the sentiment:

- Distance between aspect term and nearest feature
- Class of adjective
- Positive words count
- Negative words count
- n-gram vectorization of aspect +5 words left & right

Including these features over just taking text as feature not only made our model smarter but also increased Accuracy by a 3-4 %

Bag of Words and n-gram:
We applied the bag of words (count vectorization) technique (later shifted to tfidf vectorizer), which assigns a fixed integer id to each word, which is the count of the number of instances of each word. Thus, the bag of word representation implies that the number of features is equal to the number of words in the extracted +5 words left and right of aspect term.
We can further use N-gram features, which builds upon the 1-gram bag of words to build larger bags. For this project, we attempt to use Bi-gram and Tri-gram features, and found that it significantly augments the performance of machine Learning models. We tried possible combinations of n-gram range from Uni, Bi and Tri gram and found that combination of all three give the best accuracy and score. Therefore, we use Uni + Bi + Tri gram features. Such an approach improves our performance by 2- 2.5 % on average.

Term Frequency Inverse Document Vectorizer on n-gram data:
We shifted from count vectorizer to tfidf vectorizer because count vectorizer just gives us frequency of each term in the document ignoring how important that word is for the sentiment. Tfidf perfectly serves this purpose. Incorporating IDF (Inverse Document Frequency) with TF transformation permits us to normalize the count of N-grams in a uniform manner. It also assists in removal of very frequent words which simply add computational complexity without improving the score. We performed an experiment using just term frequencies and found that inverse document frequency scaling was very important, as our ML models failed to generalize well with just TF. Using TF-IDF, our score improves by approximately 2-3 %.

Chi2 method:
We used chi2 method for choosing best 'k' feature for classification. This proved to be great for the provided data and max increase in accuracy was recoded as 5%, which had great benefit for us. In this, we tried for different values of k (4000, 6000, 8000, 10000, 12000, 15000) and got the best results on 12000.

Major Python Packages/Libraries used:

pandas	sklearn.feature_selection.chi2, SelectKBest
numpy	sklearn.svm.LinearSVC
scipy	sklearn.neighbors.KNeighborsClassifier
sklearn.metrics.confusion_matrix	nltk.tokenize.word_tokenize
sklearn.feature_extraction.text.TfidfVectorizer	nltk.tokenize.RegexpTokenizer
sklearn.model_selection.StratifiedKFold	nltk.stem.porter import PorterStemmer
sklearn.tree.DecisionTreeClassifier	nltk.stem import WordNetLemmatizer
sklearn.naive_bayes.MultinomialNB	nltk.corpus import stopwords

Classification Methods Experimented:

The following machine learning models were experimented with, and for each of these, various hyperparameters for respective models were tuned for performance improvement:

1. K-nearest neighbors (Params tuned: n_neighbors)
2. Decision Trees (Params tuned: max_depth)
3. Linear SVM (Params tuned: dual)
4. Polynomial SVM (Params tuned: degree, gamma, C)
5. SVM with radial basis function (Params tuned: C, gamma)
6. Naïve Bayes

EVALUATION

The following data reflects the accuracy, precision, recall & F-score (with respect to positive, negative & neutral classes) of various models experimented with on both datasets. These values have been calculated on basis of 10-fold cross validation:

Data 1				
Classifier	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	63.5	63.7 70.5 -	68.4 75.9 27.5	65.97 73.1
K Nearest Neighbors	59	66.8 55.2 46.2	69.7 62.3 29.8	68.2 58.5 36.2
Linear SVM	76.1	89.2 - 84.8	95.1 - 68.6	92.1 75.8
Naïve Bayes	58.4	72.6 55.6 -	51.5 91.7 0	60.3 69.2

Data 2				
Classifier	**Accuracy**	**Precision**	**Recall**	**F1-Score**
Decision Tree	65	55.6 76.6 40.5	34.9 82.7 42.9	42.9 79.5 41.7
K Nearest Neighbors	60.2	64.6 39.1 31.1	91.1 14.1 13.4	75.6 20.7 18.7
Linear SVM	70.4	86.5 - 65.4	95.8 - 34.2	90.9 44.9
Naïve Bayes	60.965	- 60.7 -	3.96 100 0	 75.5

CONCLUSION

After trying several classifiers for training and prediction of sentiment classes based on aspect-term using 10-fold cross validation, it was observed that Linear Support Vector Machine yielded best results out of all. Although polynomial SVM demanded much more time for fitting the training data, it did not show significant improvement in performance over linear SVM.

Thus, we use Linear SVM as our chosen classifier for predicting classes based on the features mentioned above.

We plan to further experiment with various ensemble methods to check for score improvement.

REFERENCES

1. Minqing Hu and Bing Liu. "Mining and Summarizing Customer Reviews." ; Proceedings of the ACM SIGKDD International Conference on Knowledge ; Discovery and Data Mining (KDD-2004), Aug 22-25, 2004, Seattle, ; Washington, USA
2. Bing Liu, Minqing Hu and Junsheng Cheng. "Opinion Observer: Analyzing and Comparing Opinions on the Web." Proceedings of the 14th ; International World Wide Web conference (WWW-2005), May 10-14, ; 2005, Chiba, Japan
3. http://nltk.org/
4. https://www.cs.uic.edu/~liub/FBS/sentiment-analysis.html
5. scikit-learn.org/stable/
6. https://dzone.com/articles/nlp-tutorial-using-python-nltk-simple-examples
7. https://www.sciencedirect.com/science/article/pii/S1877050916320683
8. http://pages.cs.wisc.edu/~jerryzhu/cs769/text_preprocessing.pdf
9. http://text-processing.com/demo/sentiment/
10. https://machinelearningmastery.com/clean-text-machine-learning-python/
11. https://stackoverflow.com/questions/15388831/what-are-all-possible-pos-tags-of-nltk?utm_medium=organic&utm_source=google_rich_qa&utm_campaign=google_rich_qa