

report on sentiment analysis

ARTIFICIAL INTELLIGENCE- INT 404



**SUBMITTED TO: SHABNAM MA’AM**

SUBMITTED BY:

DIVYANSHU TIWARI 11802090 ROLL NO. 43

HARSHIT GUPTA 11814354 ROLL NO. 55

ASHISH GAUTAM 11815069 ROLL NO.52

SHUBHAM KUMAR 11813128 ROLL NO. 50

**GITHUB REPOSITORY LINK:**

*https://github.com/deadshot674gam/INT404/blob/master/sentiment\_analyzer.py*

**Sentiment Analysis**

**Introduction**

Sentiment analysis is the task of identifying whether the opinion expressed in a text is positive or negative in general, or about a given topic.

Inception is such a good movie, highly recommends 10/10”, expresses positive sentiment toward the movie, named Inception, which is considered as the topic of this text. The texts that we deal with in this project, express must be either positive or negative sentiment.

We have used Naïve Bayes classifiers over a set of features that were extracted from the texts using techniques taken from the field of natural language processing [NLP]. In this we used sets of manually annotated texts: movie reviews([93mcorpora/movie\_reviews.zip/movie\_reviews.com).

The first set contains reviews about various movies, unlimited in their text length, annotated by humans with their sentiment toward the movie being described.

# Datasets

In this project, we used the following dataset:

1. **Corpus-** **movie-review polarity dataset V 2.0**

This set is a collection of movie-review documents labelled with respect to their overall sentiment polarity (positive or negative). It contains positive and negative processed reviews. The reviews were preprocessed by the dataset editors so that each review is formatted as a plain tokenized text, containing no structured information that can imply on the polarity of the text (e.g. stars rating – 0/5).

The NLTK corpus movie\_reviews dataset has the reviews, and they are already labeled as positive and negative. This means we can train and test with this data.

# Features

Sentiment analysis may be considered as a text categorization problem where the topic of a given textual document is guessed from a set of all possible sentiment labels as in the polarity variation this set contains only the labels positive and negative. We extract various features from the text using NLP based algorithms and inject them into classifiers.

|  |  |  |
| --- | --- | --- |
| **Feature** | **Definition** | **Example** |
| Word | The word itself | hello, can’t, |
| Lemma | The dictionary entry of a word | thought = think, children = child,  books = book |
| Part-­‐of-­‐speech (POS) tag | The part of speech of the word (e.g. Verb, Noun, Adjective), considering *it’s context* | In the text -­‐  “I would like to *book*a ticket”  the POS tag of the word “book” is *Verb* |

# Classifiers

We have used Naïve Bayes classifier. We used Naïve Bayes by treating the distribution of each feature over every class i.e. positive and negative. It’s a simple, fast, and easy classifier which performs well for small datasets. It’s a simple probabilistic classifier based on applying Bayes’ theorem. Bayes’ theorem describes the probability of an event, based on prior knowledge of conditions that might be related to the event.

Train classifier on all instances of the Movie Reviews Dataset.

The Corpus has been pre processed using the default sentence tokenizer and WordPunctTokenizer.

Training and testing , Reviews will be equally split between positive and negative.

NLTK’s corpus is structured in an interesting way:

1.First iterate through the two categories (pos and neg)

2.Each of these categories is just filields, so grab those

3.Then each review is a NLTK class where each item in that class instance is a word.

# Analysis

|  |  |
| --- | --- |
| **Settings name** | **Description** |
| Unigram (U) | Using the words |
| Lemma (L) | Using the lemmas; when a lemma does not exist (e.g. proper name), we use the word itself. |
| Lemma+POS (L+P) | Same as previous, but here every lemma is concatenated with its part-­‐of-­‐speech tag (e.g. book VERB). Subject  words were not considered |
| SelectedLemna (SL) | Only lemmas that their part-­‐of-­‐speech tag is one of the following: adjective, adverb, or verb. |
| SelectedLemma+POS (SL+P) | Here every selected lemma is  concatenated with its part-­‐of-­‐speech tag. Subject words were not considered. |
| LemmasAndPOS (LP) | Using the lemmas and their corresponding part-­‐of-­‐speech  tags as individual features. |
| Lemma+Location  (L+LOC) | Using the lemmas concatenated with their location, i.e.  their distance from the subject. |

# Conclusions

In this project we have investigated the task of sentiment analysis as a classification problem. We have used corpus movie reviews. We have shown the results using word- based features comparing to movie reviews.

**SOURCE CODE**

|  |
| --- |
| from nltk.corpus import movie\_reviews |
|  |

|  |
| --- |
| from nltk.classify import NaiveBayesClassifier |
|  |

|  |
| --- |
| from nltk.classify.util import accuracy as nltk\_accuracy |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| def extract\_features(words): |
|  |

|  |
| --- |
| return dict([(word, True) for word in words]) |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| if \_\_name\_\_=='\_\_main\_\_': |
|  |

|  |
| --- |
|  |
|  |

|  |
| --- |
| fileids\_pos = movie\_reviews.fileids('pos') |
|  |

|  |
| --- |
| fileids\_neg = movie\_reviews.fileids('neg') |
|  |

|  |
| --- |
| features\_pos = [(extract\_features(movie\_reviews.words( |
|  |

|  |
| --- |
| fileids=[f])), 'Positive') for f in fileids\_pos] |
|  |

|  |
| --- |
| features\_neg = [(extract\_features(movie\_reviews.words( |
|  |

|  |
| --- |
| fileids=[f])), 'Negative') for f in fileids\_neg] |
|  |

|  |
| --- |
| threshold = 0.8 |
|  |

|  |
| --- |
| num\_pos = int(threshold \* len(features\_pos)) |
|  |

|  |
| --- |
| num\_neg = int(threshold \* len(features\_neg)) |
|  |

|  |
| --- |
| features\_train = features\_pos[:num\_pos] + features\_neg[:num\_neg] |
|  |

|  |
| --- |
| features\_test = features\_pos[num\_pos:] + features\_neg[num\_neg:] |
|  |

|  |
| --- |
| classifier = NaiveBayesClassifier.train(features\_train) |
|  |

|  |
| --- |
| a="yes" |
|  |

|  |
| --- |
| while a=="yes" or a=="Yes": |
|  |

|  |
| --- |
| print("Enter a string for analysis:") |
|  |

|  |
| --- |
| input\_string=input() |
|  |

|  |
| --- |
| print("\nString predictions:") |
|  |

|  |
| --- |
| print("\nString:", input\_string) |
|  |

|  |
| --- |
| probabilities = classifier.prob\_classify(extract\_features(input\_string.split())) |
|  |

|  |
| --- |
| predicted\_sentiment = probabilities.max() |
|  |

|  |
| --- |
| print("Predicted sentiment:", predicted\_sentiment) |
|  |

|  |
| --- |
| print("Probability:", round(probabilities.prob(predicted\_sentiment), 2)) |
|  |

a=input("Want to check another string:(Enter yes or no)")