Sentiment Analysis

Training a Sentiment Classifier on the Amazon Reviews Dataset

Khaled Hechmi – 793085
Gian Carlo Milanese – 848629

Goal of the Project

Training a classifier on the Amazon Reviews Dataset that predicts positive/negative sentiment and apply it to unseen reviews and to tweets.

Understand if it is possible to train a classifier on a dataset of reviews that is also efficient and effective in performing sentiment analysis on tweets.

Motivation

- Datasets of labelled tweets for training purposes are difficult to find or expensive to create
- The Amazon Reviews Dataset provides a large collection of opinions on several different categories of products, each labelled with a score that is helpful for assigning a basic sentiment.

Applications

- A company could be able to quickly understand how new products are perceived on Twitter by applying the classifier to tweets or to replies to tweets presenting the product.
- A good classifier could be used in different domains where, like Twitter, it is too expensive to build a valid training set: some examples are finding the sentiments of text coming from other social media, such as YouTube comments or discussions in specialized online communities.

Data

- Amazon Reviews from the categories Automotive, Cell Phones and Accessories, Grocery and Gourmet, Video Games with 1-5 scores
- Twitter replies to a tweet by Samsung Mobile presenting the Galaxy Note 10 and 10+, manually labelled as positive, negative or neutral
- Tweets containing the hashtag #CES2020 (unlabelled)
- Sanders Analytics' twitter sentiment corpus consisting of 5113 hand-classified tweets (positive, negative, neutral), each regarding a topic among Apple, Google, Microsoft and Twitter

Pre-processing: Cleaning and Normalization

“<div id="video-block-..." class="a-section a-spacing-small a-spacing-top-mini video-block"></div><input type="hidden" name="" value="..." class="video-url"><input type="hidden" name="" value="https://images-na.ssl-images-amazon.com/images/I/....png" class="video-slate-img-url"> This video will show you this after market cable. It works great and it is not cheap construction either. Works on my iphone 4 and ipad. Charges great. It is the same size as the OEM cable from about which is about 3ft. Quality I would say they are about the same.”

- Lowercase
- Delete URLs and HTML tags

“this video will show you this after market cable. it works great and it is not cheap construction either. works on my iphone 4 and ipad. charges great. it is the same size as the oem cable from about which is about 3ft. quality i would say they are about the same.”

Pre-processing: Cleaning and Normalization

- Replace emoticons with associated sentiment
- Translate slang and acronyms
- Expand contractions
- Spelling correction
- Some special characters and punctuation

“Gr8 product :)) <3”

“Terible qualtiy case, IMO. Cheap-looking. Woudln't recommend :-(”

“great product good good”

“terrible quality case in my opinion cheap-looking would not recommend bad”

Pre-processing: Tokenization and Stop-Words

- Tokenization
- Removal of stop-words
  - Words that represents a negation of the following term are not considered stop-words

“terrible quality case in my opinion cheap-looking would not recommend bad”

['terrible', 'quality', 'case', 'opinion', 'cheap-looking', 'not', 'recommend', 'bad']

Pre-processing: Stemming

Different stemming implementations were tested:

- Porter Stemmer by NLTK
- Lancaster Stemmer by NLTK
- Porter Stemmer by PyStemmer

['terrible', 'quality', 'case', 'opinion', 'cheap-looking', 'not', 'recommend', 'bad']

Porter

['terribl', 'qualiti', 'case', 'opinion', 'cheap-look', 'not', 'recommend', 'bad']

Lancaster

['terr', 'qual', 'cas', 'opin', 'cheap-looking', 'not', 'recommend', 'bad']

Pre-processing: Additional Cleaning for Tweets

Besides the previous operations:

- Removing mentions
- Removing “RT” at the beginning of retweets

Text Representation

TF-IDF Vectorization of pre-processed data:

- Without stemming
- After NLTK’s Porter stemmer
- After NLTK’s Lancaster stemmer
- After PyStemmer’s Porter stemmer

Doc2Vec was also considered, but was discarded due to poor performance and long training times.

Text Representation

TF-IDF Vectorization of pre-processed data:

- Minimum document frequency: 5 documents;
- Keep first 50000 features to reduce dimensionality.

Doc2Vec was also considered, but was discarded due to poor performance and long training time.

Truncated SVD

Truncated SVD was also used for reducing the dimensionality of the TF-IDF Matrix, so that more computationally intensive algorithms could be used.

- Original features: 50 000
- Reduced features: 200

Classification: Task

Binary classification task:

- A review with score $>$ 3 is considered having positive sentiment
- A review with score ≤ 3 is considered having negative sentiment

Motivations:

- The All positive/All critical filter on Amazon
- The classifier will be tested on tweets, where a positive/negative label is more appropriate than a score in the 1 to 5 range.

Classification: Algorithms

Different classification algorithms were evaluated on efficiency, using the 4 different representation:

- Multinomial Naive-Bayes
- Random Forest
- Random Forest on SVD Dataset
- Linear SVC on SVD Dataset
- SVC on SVD Dataset
- Adaboost (20 estimators) on SVD Dataset

Results on Reviews dataset

- Multinomial NB is by far the most efficient algorithm, with low training time (~ 200 ms) and test time (~50 ms)
- Multinomial NB and the "Without stemming" dataset achieve the best accuracy and recall and the second-best precision
- RandomForest has the best precision even though it suffers from overfitting (~ 18 % difference on accuracy)

Classifier	Accuracy	Precision	Recall
MNB, No Stemming	0.84922	0.86423	0.82880

MNB Classifier evaluation on tweets

Recall the data sources:

- CESTweets: 2000 unlabelled tweets containing the hashtag CES2020
- SanTweets: 5113 labelled tweets from the Sanders Analytics twitter sentiment corpus
- SMTweets: 289 labelled replies to a tweet by Samsung Mobile.

Evaluating Tweets: Filtering out Neutral Tweets

The classifiers only predicts positive/negative sentiment, but tweets can also be neutral (for instance, questions). Neutral tweets need to be filtered out:

- We can consider only tweets that are manually labelled Positive or Negative
- We can consider only tweets that are labelled Positive or Negative by an additional classifier that resorts to SentiWordNet, a lexical resource that assigns to each synset of WordNet three sentiment scores: positivity, negativity, and objectivity

SentiWordNet Classifier: Definition

The SWNClassifier takes as input the pre-processed tweets and works as follows:

- Each tweet is split in tokens using NLTK's tokenizer
- A Part of Speech tag is assigned to each token with NLTK's pos_tag
- Each tag is translated to a SentiWordNet tag;
- Each token/tag pair is assigned the positivity and negativity score defined by SentiWordNet
- The positivity and negativity score of a tweet is computed as the sum of positivity and negativity scores of the constituting tokens.

SentiWordNet Classifier: Performance

The SWN Classifier is a very simple algorithm with poor performances. It was tested on the labelled tweets on the task of predicting Sentiment/NoSentiment in tweets:

Dataset	Accuracy	Precision	Recall
SanTweets	0.48680	0.93148	0.37519
SMTweets	0.57093	0.75000	0.15217

NB Classifier evaluation on tweets

The NB Classifier was then evaluated on:

- CESTweets labelled as positive or negative by the SWNClassifier
- SanTweets labelled positive/negative
- SanTweets labelled as positive or negative by the SWNClassifier
- SMTweets labelled positive/negative
- SMTweets labelled as positive or negative by the SWNClassifier

NB Classifier Evaluation: Results

Dataset	Labelling	Accuracy	Precision	Recall
SanTweets	Manual	0.71402	0.78202	0.55299
	SWN	0.58689	0.73879	0.5168
SMTweets	Manual	0.95364	0.83333	0.78947
	SWN	0.62835	0.78846	0.3228
CESTweets	SWN	0.56080	0.81130	0.5142

Conclusions

- Stemming algorithms did not lead to a tangible improvement in accuracy, precision or recall
- Multinomial NB is the most efficient and effective algorithm among those tested
- Model built on Amazon reviews was able to generalise quite well on manually labelled dataset
- SWN does not work well in these domains

Improvements

- Parallelize some pre processing functions (e.g. tokenization, normalisation and spell checking)
- Create a bigger and more robust slang dictionary
- Improve the automatic filtering of neutral tweets in order to input better data to the positive/negative classifier