SENTIMENT ANALYSIS OF MOVIE REVIEWS USING MACHINE LEARNING: PROJECT PROPOSAL

Julia Mitroi
York University

March 29, 2019

Abstract

Sentiment analysis is the automated process of extracting an opinion (or "sentiment") about a given subject from written or spoken language. It can be used to analyze sentiments of diverse datasets ranging from corporate surveys to movie reviews. Using sentiment analysis systems, businesses can automate operations, get actionable insights, and understand customer needs and attitudes towards their brand or products, overall making business processes more efficient and profitable. In the film industry, the quality of online reviews may influence consumer purchase decisions and daily box office performance. Movie reviews sentiment analysis can provide insights that can determine marketing strategy, and improve movie campaign success, product messaging, and customer service. The goal of this project is to explore and apply supervised machine learning techniques used in sentiment classification for movie review analysis, and to assess the benefit of using those techniques for this kind of problem. Several state-of-the-art algorithms will be implemented, and their performance (prediction accuracy) will be evaluated and compared to baseline algorithms, to select the best-performing model. The project is based on the Kaggle data science competition named "Sentiment Analysis on Movie Reviews" that uses a Rotten Tomatoes movie reviews dataset.

1. Introduction

Sentiment analysis is one of the most popular applications of Natural Language Processing (NLP). It builds systems that automatically identify and extract opinions, attitudes, and feelings (or "sentiments") from natural language [1]. It can be used to analyze sentiments of diverse datasets ranging from corporate surveys to movie reviews. Indeed, in a world where 2.5 quintillion bytes of data are created every day, sentiment analysis has become a key tool of analysis and research for making sense of data, as it is usually beyond human capacity and time to manually process such large amounts of information. Text content is available on review sites, blogs, and social media, or in corporate databases, and sentiment analysis can be used to extract information in the proper context from the vast ocean of content. This has many practical applications, including social media monitoring, user review analysis, marketing analysis, customer service, political views analysis, product analytics, and financial prediction.

One of the most important elements for a business is being in touch with its customer base. It is vital for companies to know what consumers or clients think of new and established products or services, recent initiatives, and customer service offerings. Sentiment analysis is one way to accomplish this necessary task, and is applicable in almost every business and social domain because opinions are central to most human activities and behaviors. In the film industry, the implication is that the quality and volume of movie ratings and reviews reflect movie quality (a movie being the product being evaluated), and that the ratings of online reviews influence consumer purchase decisions and have an effect on daily box office performance (i.e., revenues) [2]. Movie reviews sentiment analysis – the topic of this paper and project – can thus be an excellent source of information, providing insights that can determine marketing strategy; improve movie campaign success, product messaging, and customer service; test business Key Performance Indicators; and generate leads. If done correctly, sentiment analysis can improve a com-

pany's bottom line. Furthermore, insights and methods from sentiment analysis in a given domain (e.g., movie reviews) can be transferred to other domains, such as workforce analytics to predict the sentiment of employee survey data, which the author of this paper plans to achieve in a separate project.

There are two major approaches in the field of sentiment analysis research: Supervised machine learning (including deep learning) and unsupervised lexicon-based approaches. The first approach typically requires a pre-labeled training dataset. The simplest type of algorithm (lexicon-based) would use a dictionary to look up which words or phrases indicate which sentiment they depict. This kind of dictionary- or knowledge-based approach works to some degree; however, advanced algorithms that use machine learning can capture sentiment nuances, for instance the fact that "bloody" may be a positive indicator if used in the phrase "bloody excellent". Machine learning algorithms learn from large datasets, often already labeled by humans as to what constitutes positive and negative, or finer-grained levels of positive and negative sentiment. Recent developments in machine learning have led to emerging approaches that could be applied in new ways to sentiment analysis to improve sentiment classification accuracy. The business objective of this project is to explore and apply state-of-the-art machine learning techniques used for sentiment classification of movie reviews, and to quantify the benefit of using those techniques for this kind of problem.

The project is based on the Kaggle data science competition named "Sentiment Analysis on Movie Reviews" that uses the Rotten Tomatoes movie reviews dataset. The competition was inspired by the work of Socher et al. (2013) [3] who used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the corpus. The challenge is to label phrases on a scale of the five sentiment values, or classes: negative, somewhat negative, neutral, somewhat positive, and positive. Factors such as sentence negation, sarcasm, slang words, misspellings, terseness, and language ambiguity make the task particularly challenging. To solve the Kaggle movie reviews challenge and select a best-performing model, I will attempt a number of supervised machine learning models, as well as deep learning algorithms, including Support Vector Machine, Random Forest, Multinomial Naïve Bayes, Long Short-term Memory (LSTM) Recurrent Neural Networks (RNN), and bidirectional Long Short-term Memory (bi-LSTM) RNN.

2. Related Work

The first academic studies measuring public opinion were conducted during and after WWII, and their motivation was highly political in nature [4]. The outbreak of modern sentiment analysis happened in the mid-2000s, and focused on product reviews available on the Web, with the year 2001 marking the beginning of widespread awareness of the research problems and opportunities that sentiment analysis and opinion mining raise; since then there has been a large volume of papers published on the topic (Pang and Lee, 2008) [1]. Factors behind this proliferation include: the rise of machine learning methods in NLP and information retrieval; the availability of datasets for machine learning algorithms to be trained on, due to the blossoming of the World Wide Web and, specifically, the development of review-aggregation web-sites; and realization of the fascinating intellectual challenges and commercial and intelligence applications that sentiment analysis offers.

Sentiment analysis has achieved a good amount of progress over the past 10 years, including the proposal of new methods and the creation of benchmark datasets [5]. The first approaches in this field of research (which were lexicon-based) depended on the use of words at a symbolic level (unigrams, bigrams, bag-of-words features), where generalizing to new words was difficult. State-of-the-art sentiment analysis methods are based on supervised machine learning, which have long surpassed the sentiment prediction accuracy of lexicon-based methods. The most common

current approach is to use pre-trained word embeddings in combination with a supervised classifier [5]. In this framework, the word embedding algorithm acts as a feature extractor for classification. RNN variants, such as the LSTM network [6] or the Gated Recurrent Units [7], are alternatives of a feed-forward network that includes a memory state capable of learning long distance dependencies. In various forms, they have proven useful for text classification tasks [8]; [9]. Socher et al. (2013) [3] and Tai et al. (2015) [8] used GloVe vectors in combination with a RNN. As their dataset was annotated for sentiment at each node of a parse tree, they trained and tested on these annotated phrases. Both Socher et al. (2013) [3] and Tai et al. (2015) [8] also proposed several RNNs that were able to take better advantage of the labeled nodes, and which achieved better results than standard RNNs. For example, Socher et al. (2013) [3] introduced the Recursive Neural Tensor Network; its accuracy of predicting fine-grained sentiment labels for all phrases reached 80.7%, an improvement of 9.7% over bag of features baselines; and it was the only model that could accurately capture the effects of negation and its scope at various tree levels for both positive and negative phrases. However, these RNN-based models require annotated parse trees, which are not necessarily available for all datasets (but are available for the dataset used in the current project). In their paper titled "Assessing State-of-the-Art Sentiment Models on State-of-the-Art Sentiment Datasets", Barnes et al. (2017) found that both LSTMs and Bi-LSTMs are particularly good at fine-grained sentiment tasks (i. e., data analysis with more than two class labels, such as the Rotten Tomatoes movie reviews dataset).

Convolutional Neural Networks (CNN) have also proven effective for text classification. CNNs are generally used in computer vision, however they have recently been applied to various NLP tasks and the results were promising [10]; [11]. Kim (2014) [10] used skipgram vectors as input to a variety of CNNs and tested on seven datasets. The best performing setup across datasets was a single layer CNN which updates the original skipgram vectors during training. Overall, the approaches noted in this section of the paper currently achieve state-of-the-art results across multiple datasets [5].

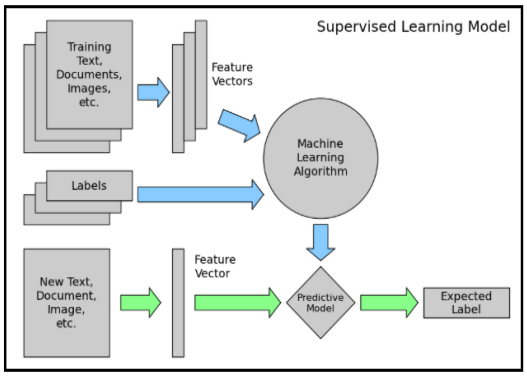

Figure 1: *Text Classification Pipeline*

3. Methodology

The proposed methods for predicting the sentiment of movie reviews in this project are based on supervised machine learning. An end-to-end supervised text classification pipeline is composed of the following components: text input, feature vector (that contains information describing the characteristics of the input data), labels (predefined categories/classes that the model will predict), machine learning algorithm (algorithm through which the model is able to deal with text classification), and predictive model: a model which is trained on the historical dataset, which can perform label predictions. Figure 1 provides a graphical representation of a text classification pipeline (image credit: medium.com).

3.1. Text Pre-processing and Normalization

The Rotten Tomatoes dataset of movie reviews was annotated for 5 levels of sentiment: strong negative, negative, neutral, positive, and strong positive. It is annotated both at the clause level, where each node in a binary tree is given a sentiment score, as well as at sentence level.

As part of the data pre-processing, I will first explore the data through data visualizations, and by calculating basic statistics such as the number of comments/phrases across label categories. I will draw several word clouds with the most representative words for each of the five classes in the training set. A histogram may also help to understand the dataset better.

In this dataset the reviews are already split into sentences, thus sentence segmentation, which is the step before tokenization, does not need to be done; since the dataset is annotated for sentiment at each node of a parse tree, they train and test on these annotated phrases. In preparation for modeling, I will perform data cleaning and normalization, for both the training and test datasets. Words need to be tokenized into numeric format to be passed to supervised machine learning models. Before that, I will also filter out spaces and punctuation marks and convert the data to lower case. After tokenization, I will use lemmatization to further reduce dimensionality. Tokenization is done by splitting apart words whenever there's a space between them. Lemmatizing means grouping together variant forms of the same word (e.g., "walk", "walking", "walks", are reduced to the base form, "walk"). I will not perform stop word removal on this dataset to filter out "stop-words" (e.g., "the", "is", "are"), as RNNs – the main models that I plan to build – are superior at learning context from previously encountered information. For movie reviews, an informal phrase such as "this movie is shit" would have opposite meaning to "this movie is the shit", hence I would like that type of information to be available to the models.

3.2. Feature Selection and Extraction

Machine learning models operate on a numeric feature space, expecting input as a two-dimensional array where rows are instances and columns are features. To perform machine learning on text, our text needs to be transformed into vector representations such that we can apply numeric machine learning. This process is called feature extraction or vectorization, and is an essential first step toward language-aware analysis. The simplest encoding of semantic space is the bag-of-words model (which is lexicon-based), whose primary insight is that meaning and similarity are encoded in vocabulary; however, bag-of-words representations only describe a document in a standalone fashion, not taking into account the context of the corpus.

Recently in NLP, new feature extraction techniques have been applied based on *word embeddings*. With word embedding, words or phrases from the vocabulary are mapped to vectors of real numbers. This makes it possible for words with similar meaning to have a similar representation, which can improve the performance of classifiers. In this project, after cleaning the data, I will vectorize the phrases using n-grams (to differentiate between words such as "good" and "not good", which will help to understand the context better as it will take into consideration both words together, rather than a single word), and I will also use Term Frequency-Inverse Document Frequency (TF-IDF), to determine the correlation between words or text segments versus treating words as discrete symbols as in the traditional bag-of-words model. The TF-IDF vectorizer ensures that the model won't give undue advantage to the stop words (since they are of higher in frequency in a document) and accentuates words that are very relevant to a specific text instance or context. The central insight is that meaning is most likely encoded in the rarer terms in text. For word embedding, I will also use a pre-trained word vector, GloVe, and evaluate it in conjunction with RNNs. Using the knowledge from an external embedding such as GloVe can enhance the precision of the RNN because it integrates new information (lexical and semantic) about the words, an information that has been trained and distilled on a very large corpus of data.

3.3. Modeling

3.3.1 The Training and Prediction Processes

Once the feature extraction process is complete, the next step is the training process, the first step of modeling. In this process, the model learns to associate a particular input (i.e., a text) to the corresponding output (tag or label) based on the samples used for training. The feature extractor transfers the text input into a feature vector. Pairs of feature vectors and labels (e.g., positive, negative, neutral) are fed into a machine learning algorithm to generate a model. In the prediction process (which happens after training), the feature extractor is used to transform unseen text inputs into feature vectors. These feature vectors are then fed into the model, which generates predicted labels or tags (again, positive, negative, or neutral; or finer-grained versions of positive and negative).

For this Kaggle project, the data are already split into training and test sets. The goal of modeling is to create a model that generalizes well to new data. The training dataset contains the examples used for learning, and is used to fit the parameters (e.g., weights) of the classifiers used in this study. For this analysis, I will also use 10-fold cross validations on the training set (cross validation is defined below) to evaluate different feature-classifier combinations, including the RNN classifiers. The validation dataset (in this case obtained iteratively through applying cross-validation) provides an unbiased evaluation of a model's fit on the training dataset while tuning the model's hyper-parameters. The test set serves as a proxy for new data; it is the sample of data used to provide an unbiased evaluation of a final model fit on the training dataset.

3.3.2 Classification Algorithms

Classification algorithms find a rule or set of rules to represent data into classes. For sentiment analysis, rule(s) are used to identify sentiment classes. In this project, I will attempt several machine learning algorithms in conjunction with word embeddings. A first classification algorithm that I will build is *Support Vector Machine (SVM)*, which was originally designed for binary classification, but extends well to multi-class classification. An SVM classifies data by determining the optimal hyperplane that separates observations according to their class labels. The vectors (cases) that define the hyperplane are the support vectors. The central concept is to accommodate classes separable by linear and non-linear class boundaries. Another classification technique that I will use is *Random Forest*, which is a supervised learning algorithm that creates a forest and makes it random. The "forest" is a combination of decision trees, most of the time trained by a combination of learning models that enhances the overall result. In other words, random forest builds a number of decision trees and merges them together to get a more accurate and stable prediction. A decision tree works as follows: Each internal node represents a "test" on an attribute/variable (e.g. whether a sentence is positive or not), each branch depicts the outcome of the test, and each leaf node is a class label (decision taken after computing all attributes). Finally, I will also apply a *Multinomial Naïve Bayes* classification algorithm, which is well-suited for discrete data such as movie ratings ranging 1 and 5, as each rating has a certain frequency to represent, and in text learning we have the count of each word to predict the class or label. Multinomial naive Bayes works similar to the Gaussian Naive Bayes, however the features are assumed to be multinomially distributed; in practice, this means that this classifier is commonly used with discrete data.

3.3.3 Modeling With Deep Neural Networks

Deep learning involves a diverse set of algorithms that attempts to imitate how the human brain works by employing artificial neural networks to process data. Deep learning libraries assume a vectorized representation of the data. In this project, I will use *Recurrent*

Neural Networks. RNNs are a class of neural networks where connections between nodes form a directed graph along a temporal sequence. This allows it to exhibit temporal dynamic behavior. Unlike feedforward neural networks, RNNs use their internal state (memory) to process sequences of inputs, which makes them well suited for machine learning problems that involve sequential data such as speech recognition, time series, stock prices, or text processing. As noted in the "Related Work" section, LSTM and bi-LSTM networks are the state-of-the-art for sentiment analysis, and perform particularly well for multi-class sentiment prediction. Based on this background, I will implement LSTM and bi-LSTM RNNs in conjunction with word embeddings. I will use Python's Keras library's LSTM layer to implement this, as well as Keras' bidirectional wrapper for the bi-LSTM network. Depending on the performance of the RNNs, I may also build a *Convolutional Neural Network*. CNNs are a special type of neural networks. Their layers are organized in three dimensions, width, height, and depth; the neurons in one layer do not connect to all the neurons in the next layer (as is the case in regular neural networks) but only to a small region of it; and the final output is reduced to a single vector of probability scores, organized along the depth dimension. In the hidden layers/feature extraction part of the CNN, the network will perform a series of convolutions and pooling operations, during which the features are detected. In the classification part, the fully connected layers will serve as a classifier on top of these extracted features. They will assign a probability for the input being what the algorithm predicts it is. The end goal is to feed new text to the CNN so that it can give a high and accurate probability for the sentiment it predicts.

3.3.4 Sentiment Analysis Metrics and Evaluation

As part of the modeling process, performance metrics need to be used to evaluate each classifier, to understand how accurate a sentiment analysis model is. In this project, I will use *k-fold cross-validation* to split the training data into 10 training folds and the same number of testing folds (and shuffle the data in the process) to evaluate different vectorizer-classifier combinations. Cross-validation uses the training folds to train the classifier, and tests it against the testing folds to obtain performance metrics (i.e., precision, recall, and accuracy, defined below). The process is repeated multiple times and an average for each of the metrics is calculated. Compared with other choices of k, 10-fold cross-validation has been accepted as a reliable method that gives a highly accurate estimate of the generalization error of a given model for a variety of algorithms and applications [12]. Hence the choice for k=10 in this study. The mean accuracies from the 10-fold cross validations will be used to determine which model (or feature-classifier combination) achieved the best accuracy for the dataset. After selecting the best feature-classifier pair based on the cross-validation results, I will retrain the feature-classifier on the whole training set, and report the final evaluation on the test set. Cross-validation helps prevent a loss of stability in the model, which can happen if the testing dataset is always the same; for example, we may be overfitting to that testing set, which means we might be adjusting the analysis to a given set of data so much that we might fail to accurately analyze a different set. Cross-validation achieves a balance between underfitting and overfitting. For the deep learning algorithms, I will also apply the early stopping technique, which is used to avoid overtraining deep learning neural network models. Early stopping stops the training when the generalization error on the validation set increases (e.g. loss begins to increase or accuracy begins to decrease). The model at the time that training is stopped is then used and is considered to have good generalization performance.

3.3.5 Precision, Recall, and Accuracy

Precision, recall, and accuracy are standard metrics used to evaluate the performance of a

classifier. *Precision* is the number of correct positive results (sentiment labels, in this project) divided by the number of positive results predicted by the classifier. *Recall* measures how many text sentiments were predicted correctly as belonging to a given category out of all the sentiments that should have been predicted as belonging to the category; in general, the more data we feed our classifiers with, the better recall will be. Classification *accuracy* is the number of correct sentiment predictions made as a ratio of all predictions made. Often, precision and recall are used to measure performance because accuracy alone can give a false sense of achieving high accuracy, especially when the cost of misclassification of the minor class samples are very high. For a difficult task like analyzing sentiment, precision and recall levels are likely to be low at first, and performance is expected to improve as the classifier is fed with more data. In this project, I will use classification accuracy to evaluate the performance of the models, defined as the mean accuracies from the 10-fold cross validations explained above. In their paper titled "A systematic analysis of performance measures for classification tasks", Sokolova and Lapalme (2009) [13] concluded that *average accuracy* provides a reliable classifier evaluation for multi-class text classification.

4. Project Time Frame

4.1. Milestones

March 29: Submit formal Project Proposal for solving the movie reviews sentiment classification problem.
April 12: Begin coding for data exploration, pre-processing, modeling, development and iteration; complete Progress Report.
May 3: Integration and packaging: Finishing development of final solution, working on the report/presentation to accompany the solution.
May 12: Hand in submission package.

5. Risk Management

As this is an individual project undertaken as an academic exercise to deepen the author's knowledge of NLP and sentiment analysis, there are no foreseen risks, comparable to those that could occur if the project had been undertaken in a corporate setting, or in collaboration with a business partner (e.g., a higher financial cost of the project than anticipated). The dataset is publicly available at the Kaggle website, thus there are no data privacy issues either. If the models that I develop would be adopted by others and implemented in a real-world setting, there could be risks inherent to using sentiment analysis as a decision-making tool. Despite the enthusiasm for sentiment analysis in the research and business fields, many are also quick to highlight its limitations. For example, Rob Metcalf, president of Digital Reasoning, pointed out that sentiment analysis should be paired up with tools that can identify audience, content, and tone, to understand the context of the communication; and that even in the best circumstances, multi-class text classification is usually only $65 - 70\%$ accurate [14]. For example, the use of substitute words or euphemisms to try to hide the true meaning of a communication, like using "baseball" for "financial instrument", are often dependent on context cues. Humor and sarcasm in text are also difficult for automated sentiment analysis platforms to identify and parse, and can confound modeling results, making such results potentially risky and unsuitable for business decision-making.

6. Conclusion

In this paper, I propose a machine learning framework for sentiment analysis of movie reviews. Various feature extraction methods in conjunction with machine learning and deep learning algorithms will be explored. By undertaking the project, my work could contribute to efforts to find best-performing models for sentiment analysis of social media content. If during the course of the project I will encounter un-

foreseen obstacles, or if the performance of the models proposed above turns out to be significantly lower than the industry standard, then the proposed framework might change and different combinations of word embeddings and classification models may be attempted; or I may allocate additional project time and effort to improving the predictive accuracy, or reducing computational run-time, or model complexity. This would be noted in an upcoming Progress Report associated with the project, or in the final report.

References

[1] Pang, B. & Lee, L. (2008). Opinion Mining and Sentiment Analysis. Found. Trends Inf. Retr. 2, 1-2 (January 2008), 1-135. doi:http://dx.doi.org/10.1561/1500000011

[2] Duan, W., Gu, B. & Whinston, A. B. (2008). Do Online Reviews Matter? - an Empirical Investigation of Panel Data. Decision Support Systems, Vol. 45, No. 4, pp. 1007-1016.

[3] Socher, R., Perelygin, A., Wu, j., Chuang, J., Manning, C., Ng, A., & Potts, C. (2013). Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. Conference on Empirical Methods in Natural Language Processing (EMNLP 2013).

[4] Mantylä, M.V., Graziotin, D., & Kuutila, M. (2018). The evolution of sentiment analysis—A review of research topics, venues, and top cited papers. Journal of Computer Science, 27, 16-32.

[5] Barnes, J., Klinger, R., & Schulte im Walde, S. (2017). Assessing State-of-the-Art Sentiment Models on State-of-the-Art Sentiment Datasets. In: Proceedings of the 8th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis, Copenhagen, Denmark, doi:10.18653/v1/W17-5202

[6] Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural computation, 9(8):1735–1780.

[7] Chung, C.J., Gulcehre, C., Cho, K., & Bengio, Y. (2014). Empirical evaluation of gated recurrent neural networks on sequence modeling. CoRR, abs/1412.3555.

[8] Tai, K.S., Socher, R, & Manning, C.D. (2015). Improved semantic representations from tree-structured long short-term memory networks. In Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics (ACL).

[9] Tang, D., Qin, B., Feng, X., & Liu, T. (2016). Effective LSTMs for target-dependent sentiment classification. In Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers.

[10] Kim, Y. (2014). Convolutional neural networks for sentence classification. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP).

[11] Flekova, L. & Gurevych, I. (2016). Supersense embeddings: A unified model for supersense interpretation, prediction, and utilization. In Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL).

[12] Karimi, S., Yin, J., & Baum, J. (2015). Evaluation methods for statistically dependent text. Comput. Linguist. 41, 3 (September 2015), 539-548. doi:http://dx.doi.org/10.1162/COLI_a_00230

[13] Sokolova, M., & Lapalme, G. (2009). A systematic analysis of performance measures for classification tasks. Inf. Process. Manage. 45, 4 (July 2009), 427-437. doi:http://dx.doi.org/10.1016/j.ipm.2009.03.002

[14] Heires, K. (2015). Sentiment Analysis: Are You Feeling Risky? Retrieved from https://digitalreasoning.com/blog/sentiment-analysis-feeling-risky/