The University of Texas at Austin

Department of Computer Science

Decoding Sentiment: Traditional Models, Neural Networks, and the Art of Feature Engineering

Author:

Zenifer Cheru Veettil

An Assignment submitted for the course:

Case Studies in Machine Learning

December 5, 2023

ABSTRACT

Sentiment analysis is an important topic in natural language processing (NLP). It is widely used in areas such as market research, content moderation, recommendation systems and customer support. In this paper, I present the results from building sentiment analysis models for IMDB movie reviews, which is a well-studied dataset. The objective is to analyze the methodologies, strengths, and limitations of prominent NLP models. They include traditional approaches like naive bayes, logistic regression, and also state-of-the-art deep learning models such as recurrent neural networks (RNNs) using long short-term memory networks (LSTMs) and transformer based models. Through a systematic literature review, I explore the theory and historic evolution of those models. This paper presents results that emphasize the importance of feature engineering. Experiments were conducted and results were analyzed using various techniques such as n-grams, TF-IDF, Bag-of-words with multi-hot encoding, word embeddings, sequence-to-sequence and stopword removals.

1 INTRODUCTION

Sentiment Analysis, also known as opinion mining, is a well-studied topic in NLP. In a data driven world, automated decoding of human emotions from very large corpora of texts becomes critical. As of 2018, the total number of reviews in Amazon.com exceeded 233 million (*Amazon Review Data*, n.d.).

The IMDB Dataset, which contains 50,000 movie reviews from the Internet Movie Review Database, is often referred to as the "Hello World" of NLP due to its ease of use (Geron, 2023). However, it still serves as a powerful benchmark against which new algorithms and architectures are applied and evaluated (*Papers with Code - IMDb Benchmark (Sentiment Analysis)*, n.d.).

Sentiment analysis poses several challenges. Understanding nuanced language expressions, such as sarcasm, irony, and double negation, is easy for humans but difficult for machines. Categorizing sentiment polarity on texts harvested from the internet is fundamentally a complex task in linguistics (Fang & Zhan, 2015).

On top of that, movie and product reviews pose the additional challenge of decoding "thwarted expectations". Reviews may contain words or sequences that are predominantly indicative of a certain polarity, however the reviewer could be trying to convey the opposite polarity. Consider the example, *"This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it can't hold up"*. Here, a prediction that is purely based on the polarity of individual words or bag-of-words is likely to result in a false positive prediction (Pang et al., 2002).

In this paper, I explore models and architectures for building a sentiment analyzer.

My experimentation plan encompasses four distinct methods:

1. Naive Bayes
2. Logistic Regression
3. Recurrent neural networks (RNNs) using Long Short-Term Memory (LSTM)
4. Transformer architecture using LSTM

I experimented with following feature engineering techniques:

1. n-grams.
2. Term Frequency-Inverse Document Frequency (TF-IDF).
3. Bag-of-words using multi-hot encoding.
4. Stopword removal.
5. Lemmatization and stemming.
6. Sequence-to-sequence

The results are summarized and analyzed in detail.

2 RESEARCH BACKGROUND and LITERATURE REVIEW

2.1 Rule-based and Lexicon-based systems

The earliest sentiment analysis models used rule-based and lexicon-based systems. Sentiment lexicons are dictionaries that mapped words and phrases with their associated sentiment, which were labeled as positive, negative, or neutral. Sentiment scores were assigned to words manually by human experts. Rule-based systems used handcrafted rules based on linguistic heuristics and patterns, such as negation handling (e.g., "not bad"), intensifiers (e.g., "very happy"), and sentiment shifters (e.g., "however") (Taboada et al., 2011).

2.2 Statistical models

Statistical approaches, which became popular in the 1980s, resulted in a noticeable improvement in the accuracy. Unlike rule-based approaches, statistical approaches do not rely on rigid and brittle language rules. Instead, they try to learn from the data, making them very flexible about their predictions (Marcus, 1995). Statistical techniques continue to be relevant for reasons including its interpretability and ability to work with less training data.

2.3 Machine Learning with Feature-Based Approaches

By the mid 2000s, the trend was shifting towards data driven approaches over rules and algorithm based approaches. Models such as naive bayes and support vector machines, in combination with manually engineered features, were trained on labeled datasets to learn and predict sentiment. Researchers at Google pointed out the remarkable potential of unlabeled data, which is abundantly available compared to labeled data. They argued that leveraging this vast pool of unlabeled data can lead to superior performance compared to models trained with less data that rely on rules or more complex algorithms (Halevy et al., 2009).

2.3.1 Bayes' Theorem

In a binary classification problem with two classes C_1 and C_2, we can use the Bayes' theorem to derive the conditional probability of a particular data point x belonging to class C_1 as follows:

$$P(C_1 | x) = \frac{P(x | C_1) P(C_1)}{P(x | C_1) P(C_1) + P(x | C_2) P(C_2)}$$

This formula tells us the probability of x belonging to class C_1 considering the prior of probability of C_1 and the probability of observing x in class C_1. The denominator is a normalization factor ensuring that the probabilities sum to 1.

2.3.2 Naive Bayes

The Naive Bayes model is a proven classification model that is based on Bayes' theorem (Lewis, 1998). The model assumes that every pair of features (words and phrases) are conditionally independent for a given class. This assumption is rarely true in any non-trivial domain, hence the name "naive" (*1.9. Naive Bayes*, n.d.). Despite its simplified assumptions, the model has been proven to work well in practice, especially for document classification tasks. There are theoretical reasons for its surprisingly high accuracy. Even though features are rarely conditionally independent in the real-world, the effects of these dependencies frequently cancel out with each other, making the model optimal in many cases (Zhang, 2004).

In a naive bayes based sentiment classifier, probability of a given document d belonging to a class C_1 is given by

$$P(C_1|d) = \frac{P(C_1)P(d|C_1)}{P(d)}$$

By assuming conditional independence of each pair of words for a given class, $P(d|C_1)$ can be rewritten as

$P(d|C_1) = \Pi_{i=1}^{n} P(w_i|d)$ where $w_1, w_2, \ldots\ldots, w_n$ are words belonging to d

2.3.3 Logistic regression

Logistic regression is one of the most fundamental machine learning models. Given that it is well-suited for binary classification tasks, it is a natural candidate for performing sentiment analysis (F. E. Harrell, 2015). Even though it dates back to the 19th century (Cramer, 2004), it is considered as a basic building block of neural networks.

Bayes theorem from 2.3.1 can be rewritten as below

$P(C_1|\, x) = \frac{1}{1+e^{-z}}$ where $z = ln(\frac{P(x \mid C_1)\, P(C_1)}{P(x \mid C_2)\, P(C_2)})$ is the log likelihood ratio.

The above function is called a logistic function, or a sigmoid function. Larger the value of z, higher the probability that given data belongs to C_1. The final classification is done based on a predefined threshold (e.g. 0.5). We can express z as a linear combination of parameter Θ to be learned and input vector x as $\Theta^T x$. Then the sigmoid function can be written as

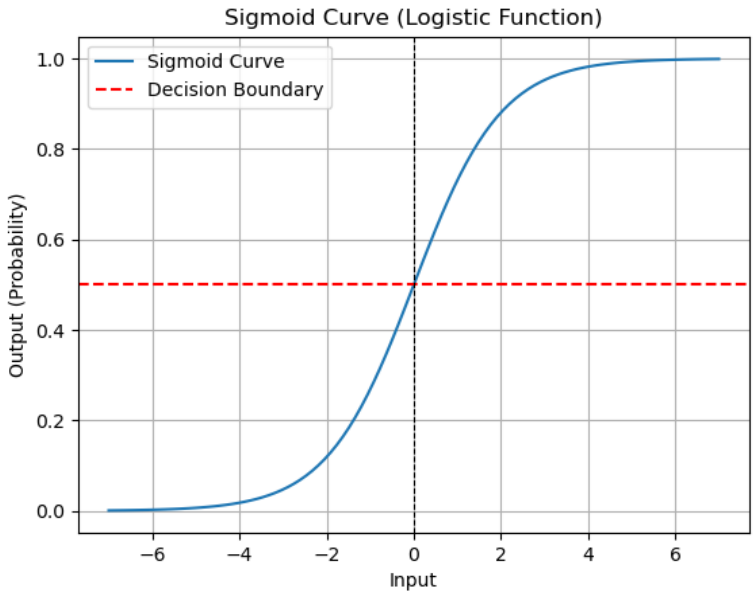
$$\hat{p} = h_\theta(x) = \sigma(\Theta^T x)$$

The cost function of a single training instance is given by

$$c(\Theta) = -ln(\hat{p}) \text{ if } y = 1 \text{ and } c(\Theta) = -ln(1 - \hat{p}) \text{ if } y = 0$$

The goal of the training is to learn parameter vector Θ so that the model assigns high probabilities for positive data points ($y = 1$) and low probabilities for negative data points ($y = 0$). In other words, if the linear combination of input vector and learned parameter, as given by $\Theta^T x$, is positive and large, then the prediction is $y = 1$. When $z = \Theta^T x$ is really large, $-z$ is really small and e^{-z} is close to 0, making the output of the sigmoid function very close to 1. Therefore the cost function makes sense because when prediction is positive, then $\hat{p}$ is close to 1 and $c(\Theta)$ is close to 0. Similarly, when prediction is wrong, $\hat{p}$ approaches 0, and $c(\Theta)$ becomes really large.

The range of the sigmoid function is between 0 and 1, as shown in the below sample diagram.



2.4 Deep Neural Networks (DNN)

Neural Networks are machine learning models that are loosely based on the inner workings of the human brain. They were first proposed by Frank Rosenblatt in the early 1950s (Rosenblatt, 1958), but fell out of favor during the AI Winter in the subsequent decades. Some of the criticism of Rosenblatt's model, called a perceptron, was its inability to solve the XOR problem (Minsky & Papert, 2017).

In the 2010s, the advances in computer hardware (e.g. GPUs) combined with widespread availability of data, finally paved the way for DNNs to be successful in practice. DNNs have shown significant success in NLP tasks, outperforming traditional methods in many areas.

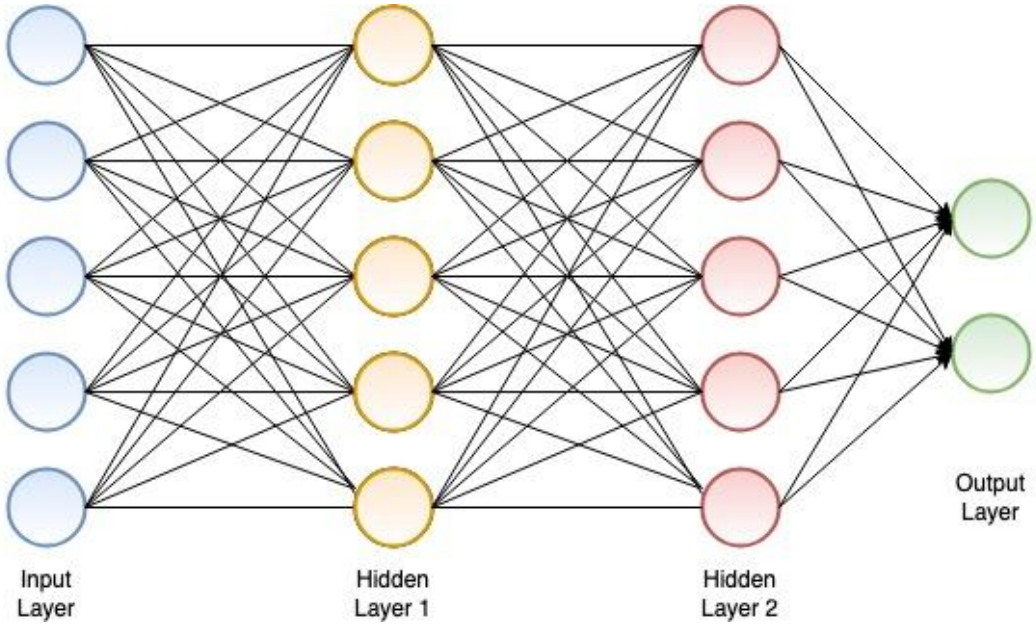
Neural networks are able to learn complex hierarchical representations of input data through representation learning. They can automatically learn and extract features from raw text, capturing the nuances of sentiment. They also minimize the need to manually apply preprocessing, feature engineering and dimension reduction.

2.4.1 Deep Learning Architecture

A sample neural network architecture with an input layer, two hidden layers and an output layer is shown below. Input layer and hidden layers have 5 neurons, each of which are connected to all other neurons in the next layer. Neurons from the final hidden layer are connected to each neuron of the output layer. A neuron has an associated weight which is the parameter that needs to be learned.

A neural network like this is capable of learning complex representations in texts by taking linear combinations of the weight of each neuron with the outputs from the layer before. The output from the final layer is passed through a special function such as the sigmoid function from 2.3.2 inorder to derive a probabilistic model. This is called a forward pass.

After each forward pass, weights are adjusted based on the losses computed by the cost function. This is called backpropagation. It takes several cycles before which the training process is able to learn the parameters (Rumelhart et al., 1986).



2.4.2 Recurrent Neural Networks (RNN)

Handling sequential data requires retaining information from distant past time steps. RNNs are neural networks that are specially designed to address this challenge (*The Unreasonable Effectiveness of Recurrent Neural Networks*, n.d.). Unlike plain NNs, RNNs can handle sequences of tokens or vectors.

The recurrent connection in RNNs is implemented using hidden states that capture and store information across different time steps. During each time step, the network processes an input, updates its hidden state based on the current input and the hidden state, and produces an output. This helps to create a form of memory, while capturing context and dependencies within sequential data.

However, RNNs face challenges in learning long-range dependencies due to the vanishing or exploding gradient problem.

2.4.3 Long Short Term Memory (LSTM)

LSTMs were introduced to handle the limitations of original RNNs (Hochreiter & Schmidhuber, 1997). LSTMs introduced a memory cell and a set of gates, which includes input gates, forget gates, and output gates to selectively update, retain, forget, or output information from the memory cell. This architecture handles the vanishing gradient problem, allowing them to learn and remember patterns over longer distances in sequential data (Gers et al., 2000).

2.4.4 Sequence to sequence architecture

Despite their flexibility and power, LSTMs as such can only handle cases where the dimensionality of the inputs and outputs is known and fixed. Sequence to sequence architecture using multilayered LSTMs was proposed to address these shortcomings (Sutskever et al., 2014).

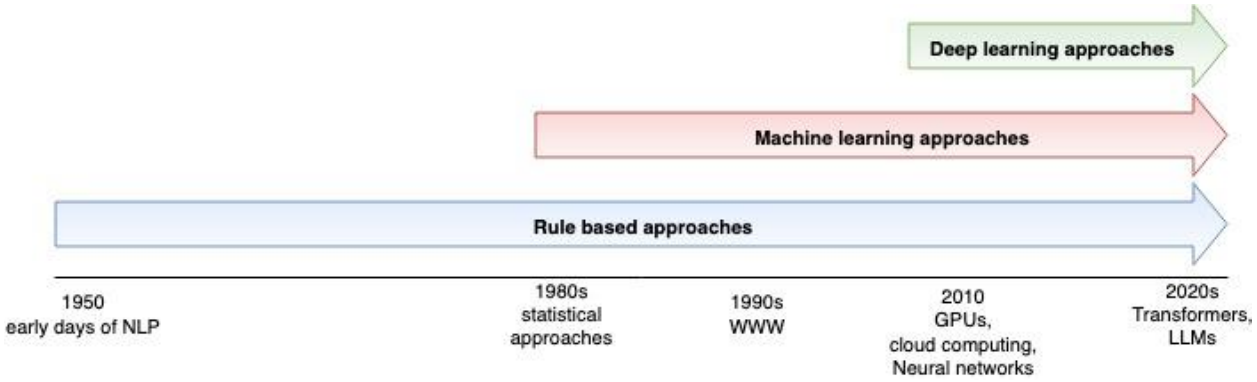
2.4.5 The Transformer architecture

Attention mechanisms were introduced to address the limitations of traditional sequence-to-sequence models, especially in tasks such as machine translation where the input and output sequences may not have a one-to-one correspondence (Bahdanau et al., 2016). However, attention mechanisms used to be mostly used in conjunction with RNNs. The purpose of self-attention is to make the learning context-aware using attention scores that capture the relationship between a given token and all the other tokens in the input.

In 2017, a novel architecture called transformer architecture was proposed that shunned recurrence and instead relied entirely on attention mechanisms. Self-attention mechanism, also known as scaled dot-product attention, helps the model to compute the attention scores based on the importance of different words in a sequence when processing each word. The model is able to capture relationships and dependencies between words, helping to understand the context and semantic meanings. Improved parallelization helps translation based models trains faster than traditional seq2seq models (Vaswani et al., 2023).

Transformer is a sequence-to-sequence model which has an encoder and a decoder. We can use the encoder part for text classification, given that it’s a very generic module.

2.4.6 Evolution of NLP



The above diagram captures the evolution of NLP during the past decades. In practice, NLP often involves a combination of both rule-based and data-driven methods. For example, pre-processing steps like tokenization and stemming may still be rule-based. The choice between different approaches depends on the specific NLP task, available data, and the desired level of performance.

2.5 Feature engineering

Feature engineering is often a decisive factor in the success of machine learning models. Reducing noise from the raw input data allows the model to generalize better, while also avoiding the curse of dimensionality which can result in overfitting. The general trend in machine learning has been to move away from handcrafted feature engineering towards representation learning, whereby features are automatically deducted from raw data.

However, the inherent structure of natural language makes a wide range of feature engineering techniques possible.

2.5.1 Bag of Words

In Bag-of-words (BoW) technique, documents are decomposed into an unordered set of words, mapping each word to its frequency. This results in the creation of a vocabulary of unique words across all documents in the dataset. This way, text data is transformed into a sparse, high-dimensional feature space. One of the drawbacks of BoW is that it overlooks semantic relationships and context (Duncan & Zhang, 2015).

2.5.2 TF-IDF

TF-IDF, which stands for Term Frequency-Inverse, allows us to scale down the relative weight of terms that occur very frequently in the given corpus. The rationale is the assumption that those terms that are more common across documents might be less informative than the terms that appear relatively rarer. (*Sklearn.Feature_extraction.Text.TfidfTransformer*, n.d.).

The TF-IDF score is calculated for each term in a document and is based on two components:

1. Term Frequency (TF): It measures how often a term appears in a document. The more frequent a term is in a document, the higher its TF score.

$$TF(t, d) = \frac{Number\ of\ times\ t\ appears\ in\ document\ D}{Total\ number\ of\ terms\ in\ document\ D}$$

2. Inverse Document Frequency (IDF): It measures the importance of a term across a collection of documents. If a term is rare and appears in only a few documents, it is considered more valuable.

$$IDF(t, D) = log(\frac{Total\ number\ of\ documents\ in\ the\ corpus\ D}{Number\ of\ documents\ containing\ t})$$

The TF-IDF score is then obtained by multiplying TF and IDF:

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D)$$

A higher TF-IDF score for a term in a document increases the significance of that term to that document in the context of the entire corpus.

2.5.3 n-grams

N-grams allows us to represent features as pairs, triplets etc., making it possible to capture some of the contextual information that is contained in a sentence. For example, n-gram allows us to capture the word pair “ice cream” in the sentence "I scream for ice cream". Another 2-gram that makes sense to be kept together is "Mr. Altman". A 3-gram will help us to capture a triplet such as “CEO Mr. Altman”.

2.5.4 One-hot encoding

One-hot encoding allows us to represent an entire corpus as a matrix such that each row corresponds to a document, and each column represents a unique term from the entire

vocabulary. Each entry is binary encoded as 0 or 1, indicating the absence or presence of a term in a specific document. As a result, the one-hot encoding captures the categorical nature of term occurrence in each document, providing a sparse and high-dimensional representation of the entire document collection.

2.5.5 Word embeddings

A word embedding is a representation of words in a continuous vector space, where words with similar meanings are mapped to similar vector representations. They are dense and continuous compared to discrete and sparse representations of one-hot encodings. Words with similar meanings have vector representations that are geometrically closer in the vector space. This allows word embeddings to capture semantic relationships and contextual information between words (Bengio et al., 2003). Pre-trained word embeddings, obtained from models trained on large text corpora, can be used as a starting point for various NLP tasks. This form of transfer learning leverages the knowledge embedded in the pre-trained word vectors (Mikolov et al., 2013).

2.5.6 Stopword removal

Stopwords are common words that carry very little semantic information. Examples include ‘a’ which is one of the most common words in a corpus, and ‘but’, ‘now’, ‘cannot’ etc. Removing stopwords reduces the dimension of feature space. However, there is no universal consensus on which words are to be included in the stopwords list (Nothman et al., 2018). Moreover, removal of stopwords can have adverse effects in sentiment analysis because standard stopwords like ‘cannot’, ‘not’ etc. are used as part of double negation.

2.6 NLP pipeline

A typical machine learning pipeline involves data acquisition, data preprocessing, data splitting, training, validation and testing. The output of the process is a model which can be deployed in production and used for predicting unseen data in the real world.

An NLP pipeline might extend the above workflow to normalize the text data by removing html tags, converting to lower cases and stemming and lemmatization using specialized libraries ("Natural Language Processing (NLP) Pipeline," 2023).

3 MATERIALS / DATA / SOURCES

3.1 IMDB Review Dataset

Internet Movie Database (IMDB) is an online database of information related to films and television series that contains millions of fan and critical reviews (*Press Room - IMDb*, n.d.). The IMDB dataset was created by a group of researchers from Stanford (Maas et al., 2011) by selecting 50,000 movie reviews, allowing no more than 30 reviews per movie. The resulting dataset contains an equal number of positive and negative reviews, making it balanced. They considered only highly polarized reviews, excluding neutral reviews altogether. A review was labeled negative when its score was ≤ 4 out of 10, and positive review when its score was ≥ 7 out of 10. The dataset was evenly divided into training and test sets.

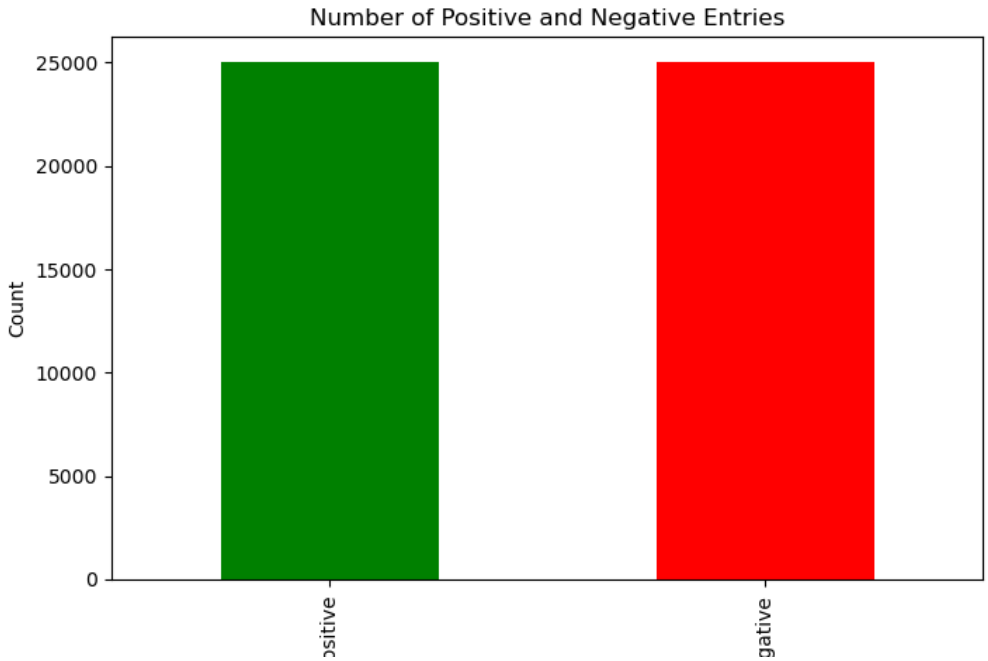
3.2 Exploratory Data Analysis

The original dataset is available under http://ai.stanford.edu/~amaas/data/sentiment/ , but I used the same data in CSV format, available from Kaggle: https://www.kaggle.com/datasets/lakshmi25npathi/imdb-dataset-of-50k-movie-reviews

Columns:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 2 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
 0   review     50000 non-null  object
 1   sentiment  50000 non-null  object
dtypes: object(2)
```

Positive and negative reviews:



Unique words:

Count of unique words in positive reviews: 279534

Count of unique words in negative reviews: 273968

Word cloud after denoising:

4 METHODS

The entire set of experiments were carried out in Python.

4.1 Preprocessing

Dataset was stripped of html tags and special characters, and was split into testing and training sets in a 80:20 ratio.

4.2 Feature engineering

Stopword removal was done using `sklearn` package's `'ENGLISH_STOP_WORDS'` list. Lemmatization experiments were conducted using the `nltk` package. Bag of words were

generated using `CountVectorizer` class and TF-IDF scores were generated using `TfidfVectorizer` class, both from `sklearn`.

4.3 Machine learning models

`MultinomialNB` and `LogisticRegression` classes from `sklearn` were used as models.

4.4 LSTM

Tensorflow and keras frameworks were used for LSTM experiments. `TextVectorization` from keras library was used for preparing data.

5 RESULTS

5.1 Naive Bayes

n-gram range	TF-IDF	Stopword removal	Test accuracy	F1-score	Feature dimension
(1,4)	no	yes	0.8809	0.8809	10,652,745
(1,4)	no	no	0.8991	0.8988	15,147,035
(1,4)	yes	no	0.8817	0.8819	10,652,745
(1,3)	no	yes	0.8814	0.8813	6,554,911
(1,3)	no	no	0.8946	0.8939	7,580,784
(1,3)	yes	yes	0.8824	0.8823	6,554,911
(1,2)	no	yes	0.8903	0.8793	2,595,994
(1,2)	no	no	0.8826	0.8816	2,123,851
(1,2)	yes	yes	0.8823	0.8815	2,595,994
(1,1)	no	yes	0.8560	0.8543	94,562
(1,1)	no	no	0.8492	0.8456	94,873

(1,1)	yes	yes	0.8651	0.8645	94,562

5.2 Logistic regression

n-gram range	TF-IDF	Stopword removal	Test accuracy	F1-score	Feature dimension
(1,4)	no	yes	0.8932	0.8952	10,752,509
(1,4)	no	no	0.9106	0.9120	15,160,052
(1,4)	yes	no	0.889	0.8912	15,160,052
(1,3)	no	yes	0.8944	0.8963	6,532,995
(1,3)	no	no	0.9106	0.9120	7,545,983
(1,3)	yes	no	0.8955	0.8978	7,545,983
(1,2)	no	yes	0.8955	0.8973	2,570,660
(1,2)	no	no	0.9099	0.9112	2,108,652
(1,2)	yes	no	0.9029	0.9047	2,108,652
(1,1)	no	yes	0.8841	0.8855	92,692
(1,1)	no	no	0.8913	0.8930	93,003
(1,1)	yes	no	0.9002	0.9021	93,003

5.3 LSTM

n-gram range	TF-IDF	max_tokens	Word embedding model	Test accuracy
1	no	25000	BoW	0.895

2	yes	25000	BoW	0.898
2	no	25000	BoW	0.904
2	yes	25000	BoW	0.901
3	no	25000	BoW	0.905
3	yes	25000	BoW	0.887
4	no	25000	BoW	0.906
4	yes	25000	BoW	0.843
4	no	50000	BoW	0.905
4	no	70000	BoW	0.913
4	no	100000	BoW	0.910
no	no	20000	Sequence, 128 dimensional self-learned embedding	0.885
no	no	20000	Sequence, 100 dimensional pre-trained Glove embedding	0.876
no	no	20000	Sequence, Transformers (sequence length = 600)	0.888

6 DISCUSSION & CONCLUSION

IMDB dataset offers a balanced dataset to experiment and evaluate various sentiment analysis models and architectures. Even though state-of-the-art text classification methodologies emphasize pretrained models (Devlin et al., 2019), I was able to show that feature engineering plays an important role in achieving good results. I showed that simpler methods like bag-of-words using n-grams can offer superior results than more powerful ones based on TF-IDF. Moreover, a simpler architecture based on logistic regression often outperformed a

sophisticated LSTM model. Stopword removal was found to have contrasting effects in naive bayes and logistic regression models. Stemming and lemmatization was found to have little impact in the accuracy of the model, while significantly increasing the training time. In most cases F1-score was found to be very close to test accuracy. This was expected, given that the dataset is perfectly balanced (Siblini et al., 2020).

Both logistic regression and LSTM achieved similar accuracies, ranging from 0.8841 to 0.9106 for various feature models and configurations. This is to be contrasted with state-of-the-art methods using pre-trained models that have achieved an accuracy of 0.962 (Dai, 2019/2023). In the following sections, I try to interpret the results in detail.

6.1 Performance of Naive Bayes

Naive Bayes showed lower accuracy than logistic regression, ranging from 0.8492 to 0.8991. For BoW, boosting the n-gram range from unigram to bigram gave a clear boost from 0.8651 to 0.8823. Further boosting to trigram and four-grams resulted in accuracy up to 0.8991.

Stopword removal had an impact only for unigrams, where it resulted in a higher accuracy of full percentage point. However, it had little meaningful impact for higher n-gram ranges. This make sense because removing stopwords are more impactful in smaller dimension space.

Using TF-IDF had the highest impact for the unigram model, boosting accuracy from 0.8492 to 0.8651. However, in higher n-gram ranges, TF-IDF had only negligible impact. This suggests that capturing local information had more impact than term frequency normalization across documents.

6.2 Performance of Logistic Regression

The removal of stopwords resulted in marked deterioration of accuracy, by up to 1.74 percentage points. This may be due to the fact that many common stopwords carry semantic meaning in movie reviews. These include phrases involving double negation such as “not bad” and “couldn’t agree”.

Increasing n-gram range from (1,1) to (1,4) resulted in consistent performance improvement, by more than a full percentage point. Also, applying TF-IDF showed consistent improvement in accuracy.

6.3 Comparison between Logistic regression and Naive Bayes

Outperformance of logistic regression over naive bayes is likely to have resulted from the superior ability of the former model to capture correlation of words within the review. Naive Bayes assume conditional independence of token pairs, which do not hold true in practice. Another key factor is the assumption of the naive bayes model we use for classification is that data follows a Multinomial distribution (*1.9. Naive Bayes*, n.d.). For a relatively smaller dataset such as this one, data could deviate from a multinomial distribution (Rennie et al., 2003).

6.4 Performance of LSTM

LSTM performed poorer than logistic regression when the number of tokens were capped at 25000. However, increasing the token size to around 70000 with four-grams boosted accuracy by 0.1 percentage points above that of logistic regression. This was expected, since neural networks do not generalize well with smaller amounts of data. Increasing token size to 100,000 however resulted in deterioration of accuracy, signaling overfitting. Changing output mode from `multi_hot` to `count` did not have any impact.

Using TF-IDF instead of multi-hot encoding boosted the accuracy for unigrams. However, for higher n-grams, it resulted in deterioration of the accuracy. For 4-grams, TF-IDF resulted in a drop of 0.6 percentage points. 4-grams result in representations with longer, but rarer sequences and those terms may have got boosted at the expense of semantically richer and shorter sequences.

6.5 Bag of Words or Sequence to Sequence?

The decision to use BoW over seq2seq or the other way depends on factors such as average document size and the number of documents in the training set, among other things. I was able to

demonstrate that for a relatively smaller dataset like IMDB, a simpler model like logistic regression could match the performance of neural networks. In neural network models, BoW outperformed the more sophisticated transformer based seq2seq model. Some researchers from Google have suggested a heuristic based on the ratio of number of samples in the corpus and mean length of the documents in the corpus. If this ratio is greater than 1500, then the seq2seq model performs better than a BoW model (*Step 2.5*, n.d.). My experiments on IMDB confirms that.

REFERENCES

1.9. Naive Bayes. (n.d.). Scikit-Learn. Retrieved December 2, 2023, from https://scikit-learn/stable/modules/naive_bayes.html

Amazon review data. (n.d.). Retrieved December 1, 2023, from https://cseweb.ucsd.edu/~jmcauley/datasets/amazon_v2/

Bahdanau, D., Cho, K., & Bengio, Y. (2016). *Neural Machine Translation by Jointly Learning to Align and Translate*.

Bengio, Y., Ducharme, R., Vincent, P., & Janvin, C. (2003). A Neural Probabilistic Language Model. *J. Mach. Learn. Res.*, *3*(null), 1137–1155.

Cramer, J. S. (2004). The early origins of the logit model. *Studies in History and Philosophy of Science Part C: Studies in History and Philosophy of Biological and Biomedical Sciences*, *35*(4), 613–626. https://doi.org/10.1016/j.shpsc.2004.09.003

Dai, Z. (2023). *Zihangdai/xlnet* [Python]. https://github.com/zihangdai/xlnet (Original work published 2019)

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*.

Duncan, B., & Zhang, Y. (2015). Neural networks for sentiment analysis on Twitter. *2015 IEEE 14th International Conference on Cognitive Informatics & Cognitive Computing (ICCI*CC)*, 275–278. https://doi.org/10.1109/ICCI-CC.2015.7259397

Fang, X., & Zhan, J. (2015). Sentiment analysis using product review data. *Journal of Big Data*, *2*(1), 5. https://doi.org/10.1186/s40537-015-0015-2

Geron, A. (2023). In *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: Concepts, tools, and techniques to build intelligent systems (chapter 16, page 587).* (3rd ed.). O'Reilly.

Gers, F. A., Schmidhuber, J., & Cummins, F. (2000). Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, *12*(10), 2451–2471. https://doi.org/10.1162/089976600300015015

Halevy, A., Norvig, P., & Pereira, F. (2009). The Unreasonable Effectiveness of Data. *IEEE Intelligent Systems*, *24*(2), 8–12. https://doi.org/10.1109/MIS.2009.36

Harrell, F. E. (2015). Binary Logistic Regression. In Jr. Harrell Frank E. (Ed.), *Regression Modeling Strategies: With Applications to Linear Models, Logistic and Ordinal Regression, and Survival Analysis* (pp. 219–274). Springer International Publishing. https://doi.org/10.1007/978-3-319-19425-7_10

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Comput.*, *9*(8), 1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. In C. Nédellec & C. Rouveirol (Eds.), *Machine Learning: ECML-98* (pp. 4–15). Springer Berlin Heidelberg.

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning Word Vectors for Sentiment Analysis. *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, 142–150. http://www.aclweb.org/anthology/P11-1015

Marcus, M. (1995). New trends in natural language processing: Statistical natural language processing. *Proceedings of the National Academy of Sciences*, *92*(22), 10052–10059. https://doi.org/10.1073/pnas.92.22.10052

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*.

Minsky, M., & Papert, S. A. (2017). *Perceptrons: An Introduction to Computational Geometry*. The MIT Press. https://doi.org/10.7551/mitpress/11301.001.0001

Natural Language Processing (NLP) Pipeline. (2023, June 1). *GeeksforGeeks*. https://www.geeksforgeeks.org/natural-language-processing-nlp-pipeline/

Nothman, J., Qin, H., & Yurchak, R. (2018). Stop Word Lists in Free Open-source Software Packages. In E. L. Park, M. Hagiwara, D. Milajevs, & L. Tan (Eds.), *Proceedings of Workshop for NLP Open Source Software (NLP-OSS)* (pp. 7–12). Association for Computational Linguistics. https://doi.org/10.18653/v1/W18-2502

Pang, B., Lee, L., & Vaithyanathan, S. (2002). *Thumbs up? Sentiment Classification using Machine Learning Techniques* (arXiv:cs/0205070). arXiv. http://arxiv.org/abs/cs/0205070

Papers with Code—IMDb Benchmark (Sentiment Analysis). (n.d.). Retrieved December 1, 2023, from https://paperswithcode.com/sota/sentiment-analysis-on-imdb

Press Room—IMDb. (n.d.). Retrieved October 25, 2023, from https://www.imdb.com/pressroom/stats

Rennie, J. D., Shih, L., Teevan, J., & Karger, D. R. (2003). Tackling the poor assumptions of naive bayes text classifiers. *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, 616–623.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408. https://doi.org/10.1037/h0042519

Rumelhart, D. E., Hinton, G. E., & Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, *323*(6088), 533–536. https://doi.org/10.1038/323533a0

Siblini, W., Fréry, J., He-Guelton, L., Oblé, F., & Wang, Y.-Q. (2020). Master Your Metrics with Calibration. In M. R. Berthold, A. Feelders, & G. Krempl (Eds.), *Advances in Intelligent*

Data Analysis XVIII (pp. 457–469). Springer International Publishing.

Sklearn.feature_extraction.text.TfidfTransformer. (n.d.). Scikit-Learn. Retrieved November 30, 2023, from https://scikit-learn/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html

Step 2.5: Choose a Model | Machine Learning | Google for Developers. (n.d.). Retrieved December 4, 2023, from https://developers.google.com/machine-learning/guides/text-classification/step-2-5

Sutskever, I., Vinyals, O., & Le, Q. V. (2014). *Sequence to Sequence Learning with Neural Networks*.

Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-Based Methods for Sentiment Analysis. *Computational Linguistics*, *37*(2), 267–307. https://doi.org/10.1162/COLI_a_00049

The Unreasonable Effectiveness of Recurrent Neural Networks. (n.d.). Retrieved December 2, 2023, from https://karpathy.github.io/2015/05/21/rnn-effectiveness/

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2023). *Attention Is All You Need*.

Zhang, H. (2004). The Optimality of Naive Bayes. In *Proceedings of the Seventeenth International Florida Artificial Intelligence Research Society Conference, FLAIRS 2004* (Vol. 2).