Evolution of Sentiment Prediction Algorithms: A Comparison Study of Common Approaches

Maryam Sajedinia, *Mat. 1088597*
Project Report Database & Algorithms Course
maryam.sajedinia@edu.unito.it

Abstract

The automatic prediction of the sentiment polarity of a text has various applications in understanding human emotions, opinions, and attitudes expressed in text. In this project, we developed and evaluated the common approaches and algorithms used for sentiment analysis. Using a dataset of 205k product reviews with sentiment labels, we trained a Recurrent Neural Network (RNN) and experimented with different hidden units (vanilla, LSTM, and GRU), as well as the bi-directionality and other hyperparameters of the model. We then fine-tuned a pre-trained transformers-based model (BERT), and compared these data-driven approaches with two baselines as Random and Lexion-Based.

1 Introduction

With the advancements in science and technology, the application domain of computers was expanded as they acquired the capability to perform various jobs to assist their human counterparts in fulfilling their objectives more accurately and consistently. Acquiring knowledge from natural language sources is one of these tasks that has been an active field of research since the early days of developing intelligent systems, attracting countless efforts to reach several breakthroughs varying from processing the textual source to obtaining information. Sentiment analysis is the natural language processing task that automatically extracts the writer's original thoughts from written text (Roccabruna et al., 2022).

In this project, we studied the task of sentiment analysis for product review comments[1]. We used a dataset of 205k product reviews including positive, negative, and neutral polarities for 104 different types of products[2]. We first developed two baselines that do not depend on the data and are supposed to manifest the minimum performance. The baselines are *Random* and *Lexicon-based*. The first model randomly predicts a sentiment polarity for a given sentence. The second model predicts a sentiment polarity based on a pre-defined dictionary of words and tokens (lexicons) that are labeled with their polarity score beforehand by field experts. In the next step, we developed a Recurrent Neural Network as a simple neural network architecture, representing a data-driven approach, and experimented with different hidden units as Vanilla, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU). We experimented with the bi-directionality property of the models as well as hyper-parameter optimizations. As the last model, we fine-tuned the BERT architecture as a transformer-based model. Lastly, we evaluated and compared the performance of the developed models as well as their computation complexity.

The contributions of this project can be summarised as follows:

- We developed two baselines for the task of sentiment prediction as *Random* and *Lexicon-based*.
- We developed a Recurrent Neural Network as a data-driven approach, and experimented with different hidden units as Vanilla, Long Short-Term Memory (LSTM), and Gated Recurrent Unit (GRU).
- Using our dataset, we fine-tuned the BERT architecture as a transformer-based model and evaluated its performance.
- We compared the performance of the models for automatic sentiment prediction as well as their computation complexity.

[1] For this project, we studied two online courses of "Machine Learning Introduction" and "Deep Learning AI" on Coursera platform, and got inspired from the models and codes of Sentiment Analysis with PyTorch repository.

[2] Flipkart Product reviews with sentiment Dataset. Kaggle Link

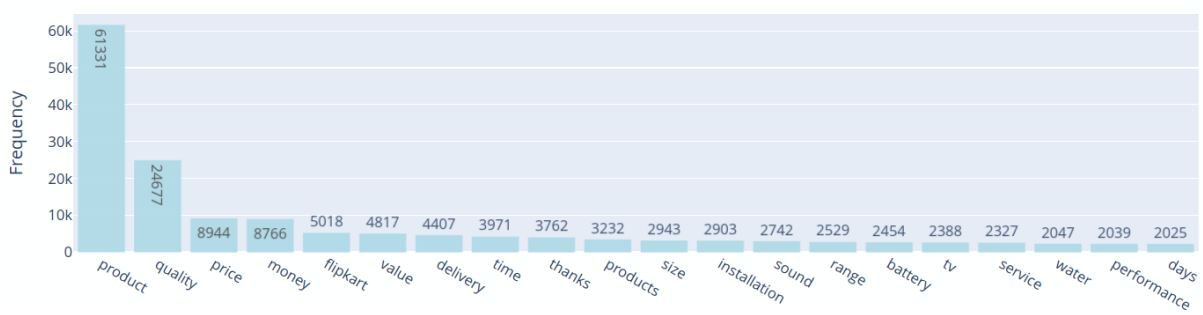

Figure 1: The top-20 frequent nouns in the reviews. The plot is obtained by parsing each review according to its linguistic dependencies and calculating the frequency of the tokens with NOUN part-of-speech tag.

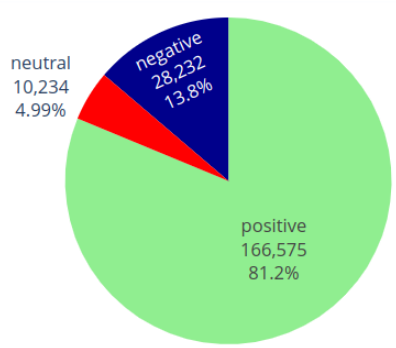

Figure 2: The ground-truth polarity distribution of the comments in the dataset. The majority of the comments are labeled as positive.

2 Data

The dataset used in this project consists of 205053 product review comments, collected from flipkart.com about 104 different products. For each review, the dataset includes Product_Name, Product_Price, User_Rating, Review_Title, Review_Content, and Sentiment_Polarity. There are types of products vary among electronics items, clothing for men, women, and kids, home decor items, etc. There are 12 samples in the dataset that do not have a Review_Content field (the field is empty). After removing these samples, we obtained a complete dataset of 205041 samples.

2.1 Data Analysis

The remaining dataset of 205041 reviews includes 3 sentiment polarities. The majority of the reviews (81.2% =166,575 reviews) are positive, while 13.8% (=28,232 reviews) are negative, and the remaining 5% (=10,234 reviews) are labeled as neutral. Figure 2 represents the polarity distribution in the data set.

We calculated the length of the reviews in our dataset as a measure of sentence complexity to predict the sentiment polarity. Using spaCy[3] toolkit we tokenized each review sentence and obtained the frequency of the token-wise length of the reviews. As plotted in Figure 3, while the review length varies from 1 to 115 tokens, the majority of the reviews have less than 15 tokens.

In the next step, we calculated the vocabulary of the comments as well as term-frequency of each token. Figure 1 presents the top 20 most frequent nouns in the comments. To obtain this Figure, we used spaCy toolkit to parse each review based on its linguistic dependencies. We then collected all tokens in the review sentences and that were labeled as NOUN and counted their frequency. As it is shown in Figure 1, the most frequent tokens are regarding the product quality, its price, the delivery time, and its performance in descending order.

2.2 Data Preparation

Appropriate pre-processing of any dataset noticeably improves the sentiment analysis process. During the data preparation phase, the initial step involved merging the Review_Title and Review_Content columns to create a comprehensive text corpus. As a result, we obtained a dataset of 205041 pairs of (Review, Sentiment_Polarity).

Most of the baseline models, such as *Lexion-Based*, often lack a neutral sentiment label. Therefore, to reduce the task complexity and to achieve a fair performance comparison among models, we selected the samples with Neutral sentiment polarity and saved our data in two different versions, one which contains data with a neutral label and the other without a neutral label. We then focused the rest of the project on the subset with Positive and Negative labels. Subsequently, this corpus was

[3] spaCy Natural Language Processing ToolKit Website

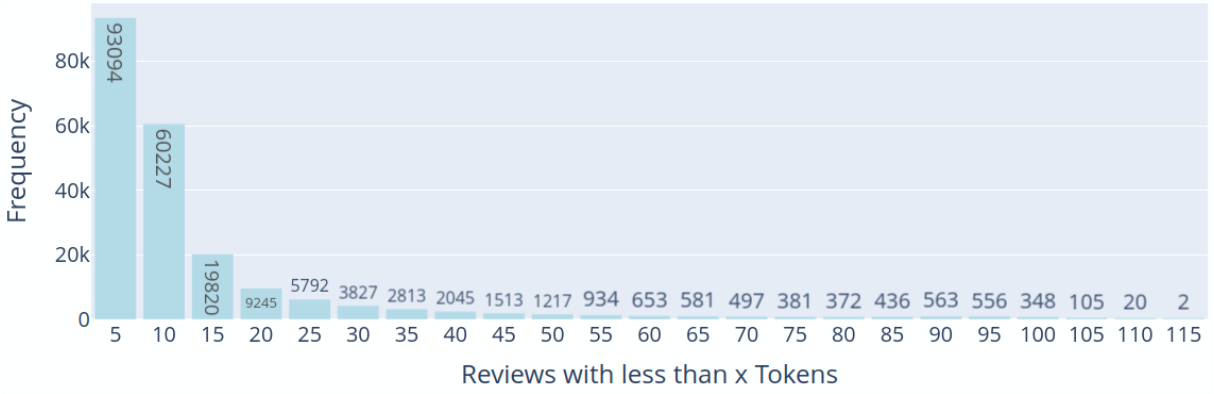

Figure 3: The token-wise length distribution of reviews. The plot is obtained by tokenizing each review using spaCy toolkit. The majority of reviews have less than 15 tokens.

divided into three essential subsets: Specifically, 80% (164,032 instances) were allocated for training data, 10% (20,504 instances) for development data, and an additional 10% (20,504 instances) for testing data.

3 Sentiment Prediction Models

Baseline We developed two non-neural baselines as *Random* and *Lexion-Based* models. The details of these models are as follows:

- **Random**: We implemented a Random predictor as a solid baseline to compare the results of our models with. This baseline was implemented in Python language which for each review sentence, randomly predicts positive or negative sentiment.

- **Lexion-Based**: The second baseline model is Lexicon-Based sentiment prediction. The Lexion-Based model in this project is VADER (Valence Aware Dictionary and sentiment Reasoner) by Hutto and Gilbert (2014)[4]. Lexicon-based models operate on the premise that the sentiment's polarity can be deduced from the frequency and intensity of sentiment-laden words. In the first step, the sentence is tokenized and split lexicons, disregarding stop words and punctuation. Using a valence dictionary, words within the text are categorized as either positive or negative. After labeling each word, a comprehensive sentiment score is calculated by summing the number of positive and negative words and integrating these numerically. Nevertheless, a limitation of the lexicon-based approach is its treatment of text as a bag of words, undermining the interpretation of language in its true context. However, the overall meaning of a text often plays an important role in identifying sentiments not necessarily the individual words or phrases it consists of.

Neural Networks Following the baseline models, we first implemented a Recurrent Neural Network (RNN) as a straightforward neural architecture that signifies a data-driven method. We tested different hidden units, such as Vanilla, LSTM, and GRU. Our experimentation also covered the bi-directional capabilities of the models and tweaks in hyper-parameters.

The descriptions of the RNN models are as follows:

- **Vanilla RNN**: Vanilla RNN is a type of neural network that is used for processing sequential data, where the hidden state at the current time as h_t is determined by the input at the current time step as x_t and the hidden state from the previous time step as h_{t-1}, multiplied by a matrix of weights W we plan to optimize. As a result, we will have $h_t = f_W(h_{t-1}, x_t)$. In the context of sentiment analysis, vanilla RNN can be used to predict the sentiment of a given sentence by processing the sentence word by word, and at each time step, updating the hidden state based on the current input word and the previous hidden state. The final hidden state of the RNN is then fed into a fully connected layer that predicts the sentiment of the sentence (Mousavi, 2019).

- **LSTM**: We used Long Short-Term Memory (LSTM) as the hidden units of our RNN. The

[4] VADER Repository

LSTM unit consists of a cell and three gates of input gate, an output gate, and a forget gate. The gates control the flow of information in the cell and decide whether a new input should be kept and substituted with the previous value, or forgotten so that the old value is retained so that the LSTM network can model long dependencies in the input vector. While vanilla RNN stores only the usual h_t hidden state, LSTM units store an additional cell state c_t as an internal vector. In this way, LSTMs can take into account the long-term dependencies in the input vector (in this work the distant words in our input sentence). In other words, when reaching the end of the sentence sequence in RNN, the influence of the first tokens on the hidden state is mostly faded. As a result, while RNN can output accurate predictions from the recent information, by increasing the sequence length we observe a decrease in the vanilla RNN performance. Meanwhile, LSTM can retain the information for a long period of time in the sequence.

- **GRU**: Gated Recurrent Units (GRU) are another type of gating mechanism in RNN, with an update gate and a reset gate, with fewer parameters than LSTM to optimize as they only have update and reset gates. In these units, the update gate helps the model determine how much of the information from previous time steps needs to be passed forward, while the reset gate is used to decide how much of the past information must be forgotten.

- **BERT**: Lastly, we fine-tuned the BERT model, which is based on the transformer architecture. Transformer architecture has an encoder and decoder stack, while BERT is just an encoder stack of transformer architecture. BERT is a bidirectional model, thus learning information from both the left and the right side of a token's context during the training phase. Compared to RNNs, BERT is a sophisticated architecture, that exploits attention mechanisms.

4 Evaluations

4.1 Implementation Details

Regarding the Vanilla-RNN model, the best performance was achieved by setting the batch size 16, with a hidden dimension of 128, and training the model for 15 epochs. The second model, LSTM, had a batch size of 64 and a hidden dimension of 256, trained for 20 epochs. The third model, GRU, utilized a larger batch size of 128 and a higher hidden dimension of 256, trained for 10 epochs. All RNN models exploited the bi-directionality property. Lastly, BERT shared the same batch size and hidden dimension as the GRU model, and it was also trained for 10 epochs.

Models	*f1*	*Prec.*	*recall*
Random	41%	49%	49%
Lexicon-Based	70%	67%	78%
Vanilla-RNN	78%	74%	84%
GRU	82%	81%	83%
LSTM	94%	95%	92%
BERT	93%	94%	92%

Table 1: The Macro F1, Precision, and Recall of the developed models on the whole test set.

Models	*f1*	*Prec.*	*recall*
Random	55%	70%	49%
Lexicon-Based	85%	88%	83%
Vanilla-RNN	89%	91%	89%
GRU	90%	90%	90%
LSTM	95%	95%	95%
BERT	97%	97%	97%

Table 2: The weighted F1, Precision, and Recall of models on the test set.

4.2 Performance

Few examples of the test set and the model's predictions are presented in Table 3. The results of the model evaluation using macro F1 on the test set are presented in Table 1. The LSTM model outperforms other alternatives by obtaining a high precision and recall. Furthermore, BERT model obtains very similar results to the LSTM as it falls short by only 1 percent of F1. Therefore, to strongly conclude that LSTM outperforms BERT we need to conduct further studies. Nevertheless, by obtaining all the predictions and applying majority voting (the prevailing sentiment polarity based on the majority of model predictions) we obtained an F1 score of 93%, precision of 93%, and recall of 92%.

Since the unbalanced nature of our dataset (there are more positive samples than negative ones), we studied the model performance using weighted scores to gain better insights, Table 2. These met-

Review Comment	GT	Predictions					
		Rand.	*Lex-Based*	*RNN*	*GRU*	*LSTM*	*BERT*
mind-blowing purchase waoo so cute and i m happy ty flipcart	neg	pos	neg	pos	neg	pos	pos
worth every penny good for decoration	pos	neg	pos	pos	pos	pos	pos
pretty good value for money medium hall looking great then assembled	pos	neg	pos	pos	pos	pos	pos
useless product very disappointed too small not stand properly	neg	neg	pos	neg	neg	neg	neg

Table 3: Few examples from the test set with the ground truth polarity (GT) and the predictions of each model. Values in red are the cases that the model predicts wrongly, with respective to ground truth.

Models	*f1*	*Prec.*	*recall*
Random	36%	44%	38%
Lexicon-Based	87%	89%	86%
Vanilla-RNN	73%	72%	74%
GRU	95%	95%	94%
LSTM	83%	83%	83%
BERT	82%	84%	80%

Table 4: F1, Precision, and Recall of the baselines and neural network models when we evaluated them just by reviews with lengths more than 75 up to 110.

rics help us provide a fair evaluation of our model's performance. Using the weighted scores, while LSTM manages to outperform baselines and other simple neural models (W-f1=95%), BERT achieves the highest performance among all models (W-f1=97%).

Since the sequence length is an important factor in this task, in our third analysis we studied the models' performance on long sequences. We calculated the token-wise length of each review using spaCy toolkit and selected samples with longer than 75 tokens. The performance of the models on this subset of the test set is presented in Table 4. Compared to Table 1, while we observe a decrease of performance in Vanilla-RNN, LSTM, and BERT, there is an increase of 13% in the performance of GRU model, outperforming all the other models. This result suggests that GRU manages to model long dependencies better than other alternatives.

4.3 Computation Complexity

Time Complexity There are few studies on the computational complexity of RNNs, with different hidden units, input dimensions, etc. It is well-established that the computational complexity of RNNs depends on several factors, including the architecture of the RNN (such as vanilla, LSTM, or GRU), the sequence length, and the size of the hidden layers (Zhang et al., 2016). However, we can confidently claim that the BERT model has higher complexity with respect to the RNN models. This is due to two reasons; a) while our RNNs have only 1 layer, the BERT model used in this study consisted of 12 layers, 768-hidden units, and 110M parameters.[5]; and b) each layer in the BERT model consists of a Feed-Forward model, and a self-attention model (Vaswani et al., 2017). While the Feed-Forward model has more-or-less the same complexity as the RNN, the complexity of self-attention mechanism is at least quadratic (Keles et al., 2023). Therefore, each layer of the BERT model has a higher complexity than each of our RNN networks.

The same chain of reasoning holds for the space complexity of the models since it is required to load matrices of values of each model to optimize/train, as well as the inference time on the test set.

5 Conclusion

In this project, we studied and evaluated different models for the task of automatic sentiment prediction using a dataset of product reviews. We developed two baselines, RNN models with different hidden units (Vanilla, GRU, LSTM), and fine-tuned the BERT model. We evaluated the models according to macro- and weighted-scores, as well as the influence of the sequence length. Lastly, we presented a short discussion on the computational complexity of the neural models, indicating that the BERT model is the most expensive alternative.

References

Clayton Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. In *Proceedings of the international*

[5] HuggingFace Documentations: Pretrained models

AAAI conference on web and social media, volume 8, pages 216–225.

Feyza Duman Keles, Pruthuvi Mahesakya Wijewardena, and Chinmay Hegde. 2023. On the computational complexity of self-attention. In *International Conference on Algorithmic Learning Theory*, pages 597–619. PMLR.

Seyed Mahed Mousavi. 2019. An entity-centric approach to response selection using unstructured knowledge. In *Master's Degree in Computer Science, Dipartimento di Ingegneria e Scienza dell'Informazione*.

Gabriel Roccabruna, Steve Azzolin, and Giuseppe Riccardi. 2022. Multi-source multi-domain sentiment analysis with bert-based models. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 581–589.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Saizheng Zhang, Yuhuai Wu, Tong Che, Zhouhan Lin, Roland Memisevic, Russ R Salakhutdinov, and Yoshua Bengio. 2016. Architectural complexity measures of recurrent neural networks. *Advances in neural information processing systems*, 29.