applied sciences

MDPI

Article

Hybrid Negation: Enhancing Sentiment Analysis for Complex Sentences

Miftahul Qorib [1,*] and Paul Cotae [2]

[1] Department of Computer Science and Information Technology, University of the District of Columbia, Washington, DC 20008, USA
[2] Department of Electrical and Computer Engineering, University of the District of Columbia, Washington, DC 20008, USA; pcotae@udc.edu
* Correspondence: miftahul.qorib@udc.edu

Abstract

Numerous valuable information is available on the Internet, and many individuals rely on mass media as their primary source of information. Various views, comments, expressions, and opinions on social networks have been a tremendous source of information. Harvesting free, resourceful information through social media makes text mining a powerful tool for analyzing public opinions on various issues across diverse social networks. Various research projects have implemented text sentiment analysis through machine and deep learning approaches. Social media text often expresses sentiment through complex syntax and negation (e.g., implicit and double negation and nested clauses), which many classifiers mishandle. We propose hybrid negation, a clause-aware approach that combines (i) explicit/implicit/double-negation rules, (ii) dependency-based scope detection, (iii) a TextBlob back-off for phrase polarity, and (iv) an MLP-learned clause-weighting module that aggregates clause-level scores. Across 156,539 tweets (three-class sentiment), we evaluate six negation strategies and 228 model configurations with and without SMOTE (applied strictly within training folds). Hybrid Negation achieves 98.582% accuracy, 98.196% precision, 98.189% recall, and 98.193% F1 with BERT, outperforming rule-only and antonym/synonym baselines. Ablations show each component contributes to the model's performance, with dependency scope and double negations offering the largest gains. Per-class results, confidence intervals, and paired tests with multiple-comparison control confirm statistically significant improvements. We release code and preprocessing scripts to support reproducibility.

Keywords: hybrid negation; negation handling; sentiment analysis; machine learning; deep learning; social media

Check for updates

Academic Editors: Antonio Fernández-Caballero, Affan Yasin, Javed Ali Khan and Lijie Wen

Received: 27 October 2025
Revised: 26 December 2025
Accepted: 15 January 2026
Published: 19 January 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Nowadays, the Internet has become the primary source of information for almost everything [1], and approximately 87% of its users utilize it as a research tool [2]. The Web offers a variety of thoughts, comments, and views, and online users constantly publish reviews on blogs and social networks, which generates content at an impressive speed [3]. Social media platforms have been an enormous source of knowledge, characterized by the rapid spread of information, where many personalities convey their views through social networks [1,4]. In 2024, for instance, Twitter had more than 500 million monthly active users, with over 200 million daily tweets on a wide range of topics [5]. Text mining has been

a powerful tool for extracting valuable information while working on social networks [6]. Text mining has become a viable tool for analyzing public opinions on various issues across social media outlets [7].

Sentiment analysis using deep learning and machine learning approaches has been widely employed by researchers in various projects, including movie reviews [8,9], fake tweet detection [10,11], medicine intake detection [12], and others [13,14].

Text sentiment analysis practices are generally categorized into statistical techniques, lexicon-based methods, and hybrid processes [15]. Prior to performing sentiment analysis, researchers perform text processing, including removing URLs and punctuation, tokenization, normalization, removing stop words, and stemming and or lemmatization [16–20]. The impact of text preprocessing on sentiment analysis significantly improves model accuracy [20,21] by removing unnecessary noise, such as punctuation, stop words, and irrelevant characters.

There are 179 words in the NLTK English stop words list, including the word "not" and its contraction forms [22]. Removing stop words during text sentiment analysis can impact the sentiment value. Consider the following negative sentiment review: "The product was not good". After removing stop words, the term becomes "product good", a positive sentiment.

It is critical to perform a text sentiment analysis with negation handling. It might help to accurately compute a sentence's sentiment by correctly identifying and addressing negative words, such as "not" or "no", These negative words can completely flip the polarity of a sentence, which significantly affects the overall sentiment classification if not adequately handled. Ignoring negation might lead to misjudging negative sentiment as positive and vice versa. Improperly handling negation in text sentiment analysis might lead to incorrect classification [23].

Inverting the polarity of the negated term [3,24] is a typical way to handle the negation, which sometimes neglects the scope and impact of negation [25,26]. Other techniques combine the scope of negation with enhanced XLNet to improve the effectiveness of negation handling [27]. About 30% of reviews are implicit sentiments [28], so considering the implicit component combined with the rule-based might enhance the negation in text sentiment analysis [29].

Simple or traditional rule-based negation systems typically fail to capture contextual dependencies within sentence structures [30]. It may encounter difficulties in handling the inherent complexity of human language and nuanced expressions [31]. Accurate negation handling is critical, as it can significantly impact sentiment interpretation and classification result [32], while failure to handle negation correctly can distort polarity and cause systematic errors in sentiment classification [24]. Although various methods have been introduced, effectively managing negation continues to be an obstacle in text-based sentiment analysis [33–35]. This study introduces a hybrid negation approach (integrating multiple components) to address the persistent challenge of handling negation in complex sentences. Furthermore, the hybrid negation approach is expected to outperform single-method techniques, delivering more accurate and reliable sentiment predictions for complex textual data.

This work is organized as follows: Section 2 reviews the existing literature on negation handling, Section 3 covers the research method, and Section 4 explains the results. Furthermore, Sections 5–7 confer discussions, contributions, and conclusions, respectively. Future work is presented in Section 8.

2. Literature Review

In the literature, most negation handling methods involve determining the polarity of sentences [3,24,36,37]. Several proposed methods exist to detect statements with negation using static windows or punctuation marks [34], which utilize the "_NEG" suffix [25,26]. The other methods used are BERT memorization [38] and applying similarity and antonym to the negation words [34]. Previous studies have only employed neural network detection for negation detection [29] and grouped negation based on classes [39]. Table 1 presents previous research projects that have worked on negation handling for sentiment analysis.

Table 1. Overview of negation handling.

No.	Title	Authors	Data	Model Classifier	Negation Handling	Class	Conclusion
1	A Study of BERT's Processing of Negations to Determine Sentiment	Giorgia Nidia Carranza Tejada, Johannes C. Scholtes and Gerasimos Spanakis	29,175 tweets	BERT	BERT memorization	Positive and negative	The highest precision of 88%, recall of 89%, and F1-score of 89% [38]
2	A Negation Handling Technique for Sentiment Analysis	Claudia Diamantini, Alex Mircoli, and Domenico Potena	597 tweets	Word Sense Disambiguation (WSD)	Switch class/polarity score with dependency parse	Positive, neutral, and negative	The best accuracy is 67%, with an F1-score of 72% [3]
3	Deep Learning Approach for Negation Handling in Sentiment Analysis	Prakash Kumar Singh and Sanchita Paul	5520 sentences	CRF, SVM, RNN, LSTM + CRF, LSTM, and BiLSTM	Neural network detection	Negation and non-negation	BiLSTM achieved the highest F1-score of 93.09% [29]
4	Effect of Negation in Sentences on Sentiment Analysis and Polarity Detection	Partha Mukherjee, Youakim Badr, Shreyesh Doppalapudi, Satish M. Srinivasan, Raghvinder S. Sangwan, and Rahul Sharma	75,000 reviews	RNN, ANN, SVM, NB, and SentiWordNet	_NEG suffix and scope of negation	Positive and negative	ANN achieved the highest accuracy of 95.67% [25]
5	Effective Negation Handling Approach for Sentiment Classification Using a Synsets in the WordNet Lexical Database	Utkarsh Lal and Priya Kamath	50,000 reviews	Logistic Regression and Naïve Bayes with BoW, TFIDF, and Word Embedding	Similarity and antonym	Positive and negative	Logistic regression with word embedding with lemmatization and negation handling achieved the highest accuracy of 91.79% [34]
6	Enhanced Twitter Sentiment Analysis Using Hybrid Approach and by Accounting Local Contextual Semantic	Itisha Gupta and Nisheeth Joshi	12,597 tweets	RBF-SVM	Switch class/polarity score for implicit and explicit negations	Positive, neutral, and negative	Hybrid RBF-SVM achieved the best accuracy of 58.67% [24]
7	Feature-Based Twitter Sentiment Analysis With Improved Negation Handling	Itisha Gupta and Nisheeth Joshi	11,216 tweets	SVM, DT, and NB	_NEG suffix and scope of negation	Positive, neutral, and negative	The SVM model classifier achieved the highest F1-score of 69.5% [26]
8	Naïve Bayes with Negation Handling for Sentiment Analysis of Twitter Data	Lobna H. Kamal, Gerard T. McKee, and Nermin Abdelhakim Othman	700,000 tweets	NB	Switch class/ polarity score	Positive and negative	Naïve Bayes model classifier with negation handling achieved the best accuracy of 77.57% [36]
9	Negation Handling in Sentiment Analysis at Sentence Level	Umar Farooq, Hasan Mansoor, Antoine Nongaillard, and Yacine Qadir Ouzrout, Muhammad Abdul	1000 review sites	WSB, SW, HB, and MP	Scope, diminisher, and morphological negation	Negation and non-negation	Negation identification at sentence level achieved the highest accuracy of 83.3% [39]
10	Negation Handling for Sentiment Classification	Kranti Vithal Ghag and Ketan Shah	2000 movie reviews	RTFSC, ARTSC, Delta TFIDF, SentiTFIDF, and Traditional classifier	Switch class/ polarity score	Positive and negative	SentiTFIDF achieved the highest accuracy of 77.3% [37]

Older models, such as BiLSTM, underachieve in complex sentences commonly due to their sequential nature which struggles with very long-range dependencies and complex relationships [40]. Secondly, BiLSTM is composed of a back and forward LSTM, so for extremely long sequences, maintaining and updating the hidden state becomes computational and memory intensive [41].

The research projects [29,39] did not perform sentiment analysis. They classified text documents into negation or non-negation to identify whether expressions were negated. The study in [29] employed neural-network-based detection to sense both explicit and implicit negation using both rule-based and implicit cues. Using a BiLSTM, the project achieved the highest F1-score of 93.09%, while [39] applied scope, diminisher, and morphological negation. Diminisher negation applied a 0.2 multiplier, while morphological (implicit negation) multiplied by −1. This study attained the highest accuracy of 83.3%.

A research study [38] focused on BERT memorization to detect and understand negation using specific training data. The goal was to gain a better understanding of how to identify the source of BERT's errors to improve sentiment analysis. The project achieved the highest precision of 88%, recall of 89%, and F1-score of 89%.

The work study in [34] implemented rule-based negation, similarity (synonym), and antonyms of negated expressions. This technique uses *synsets* from the NLTK library, providing a more straightforward way to find antonyms or synonyms of negated words in the *WordNet* lexical database rather than a complex training model. This technique could be the most cost-effective in handling negation. Applying logistic regression with word embeddings, lemmatization, and negation, the project earned an accuracy of 91.79%.

The following projects [36,37] applied a rule-based approach to switch the polarity or sentiment of the expressions: the work study in [36] implemented naïve Bayes with negation handling, which gained the best accuracy of 77.57%, while Ref. [37] achieved the highest accuracy of 77.3% using SentiTFIDF.

Another negation-handling project [3] implemented a rule-based approach using a dependency parse tree to enhance the negation technique. Its goal was to analyze grammatical relations by type of dependency, such as *nsubj*, *aux*, *neg*, *dobj*, and *cc*. Using word sense disambiguation (WSD), the project achieved a 67% accuracy and an F1-score of 72%.

Work studies [25,26] implemented _NEG to identify negated terms and employed the scope of negation to measure the impact of word relations on the final sentiment analysis. The project in [25] achieved the highest accuracy of 95.67% by utilizing an ANN. On the other hand, the work study in [26] applied _NEG identification and double negation with scope, achieving the highest F1-score of 69.5% using the SVM classifier.

Next, Ref. [24] used identification of explicit and implicit negations with the SentiWordNet (SWN) lexicon. This method has improved sentiment analysis by 2–6% compared to the traditional method. Using a hybrid RBF-SVM, the project achieved the best accuracy of 58.67%.

3. Methodology

The experimental design in this project is illustrated in Figure 1. Before performing negation handling, we preprocessed text documents (remove retweets, URLs, punctuation, numbers, and alphabetical characters) to ensure precise and reliable analysis. We expanded contractions to their full forms to facilitate negation detection. After cleaning up the dataset, we applied six negation handling methods (TextBlob (version: 0.19.0), Negex (negspacy version: 1.0.4) with SpacyTextBlob (version: 5.0.0), Antonym–synonym, Antonym–synonym with a second rule-based, zero-shot, and Hybrid). We applied SMOTE (imblearn version: 0.14.1, shown in Appendix B) to mitigate class imbalance and developed 228 models consisting of 114 without SMOTE (control) and 114 with SMOTE (treatment) to classify tweets into positive, negative, and neutral categories. Using the *Wilcoxon Signed Rank Test*, we confirmed that SMOTE improved model performance in this research project. A detailed explanation of the proposed research study is presented in the following sections.

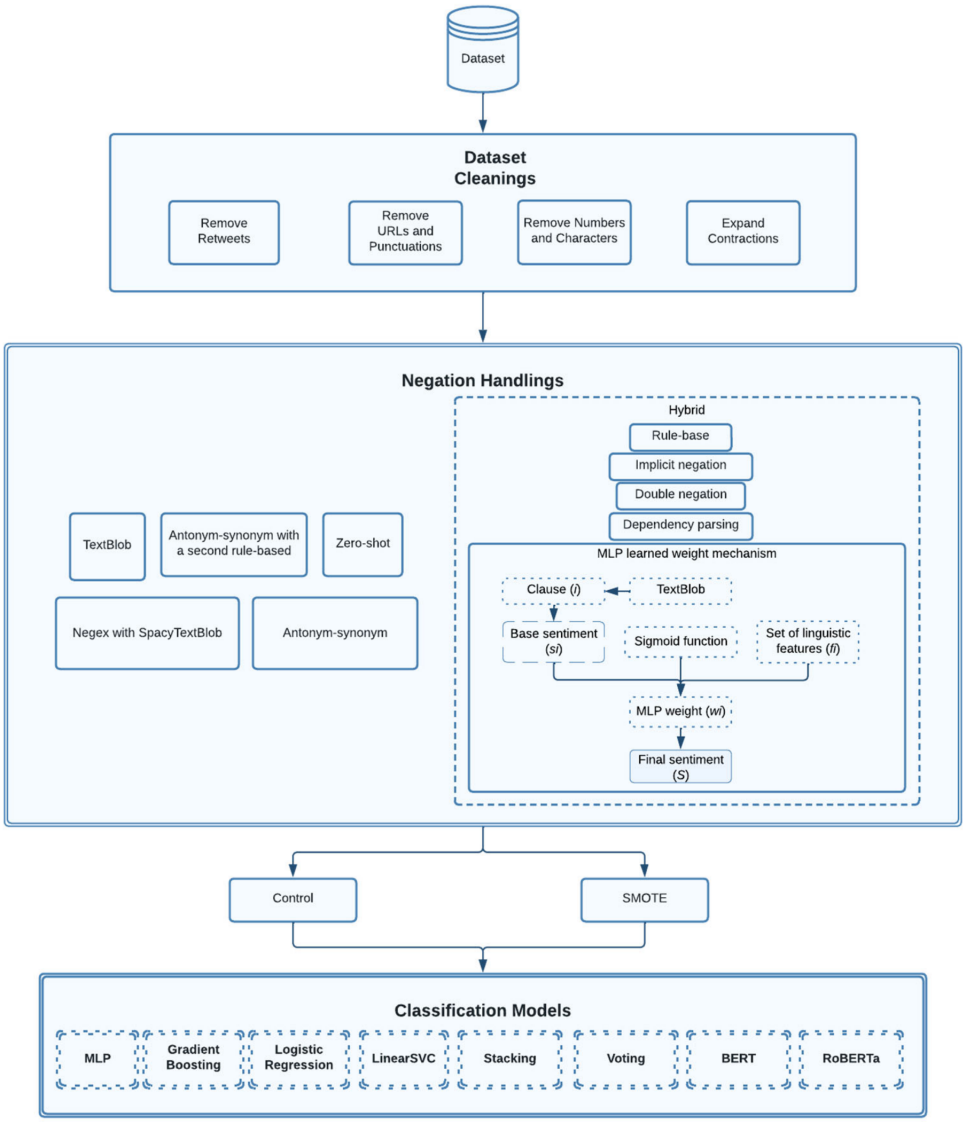


Figure 1. Design experiment.

3.1. Dataset

We used the dataset from [42], which is available to the public [43], to handle negation in text sentiment analysis. It consists of one hundred fifty-six thousand, five hundred thirty-nine public tweets. The public tweets were collected daily from 26 September 2021 to 27 March 2022.

3.2. Data Cleaning

Text preprocessing ensures the dataset is accurate, appropriate, and consistent, which leads to better model performance [44–46]. In this process, we removed retweets to eliminate duplicated content. Additionally, we removed unnecessary information, including URLs, punctuation, numbers, and irrelevant characters to minimize noise that could influence analytical accuracy. Lastly, we expanded contractions into their extended forms, such as the word "weren't" into the form of "were not", to ensure that negated sentences were easily detected.

3.3. Negation Handlings

We implemented six distinct negation handling techniques to identify the most effective approach for sentiment analysis. The negation handling techniques applied in

this study include TextBlob, Negex with SpacyTextblob, antonym–synonym, antonym–synonym with second rule-based, zero-shot, and a Hybrid approach.

3.3.1. TextBlob

Figure 2 depicts the TextBlob negation handling process, where detection of negation terms such as "not", "n't", "never", or "no" triggers polarity inversion by multiplying the sentiment score by -1, effectively reversing its orientation from positive to negative or vice versa.

$$ns = s \times -1 \quad (1)$$

where ns is negation sentiment, and s is the original sentiment score. When negation is detected, the original sentiment score is multiplied by negative one. The new polarity score would switch its original score from positive to negative and vice versa.

```
START

FUNCTION analyze_sentiment(text):
    // 1. Create a TextBlob object from the input text.
    blob = TextBlob(text)
    // 2. Get the default sentiment tuple from TextBlob.
    // The sentiment object has `polarity` (float in [-1, 1]) and `subjectivity` (float in [0, 1]) attributes.
    sentiment = blob.sentiment
    // 3. Define a list of negation words.
    negation_words = ["not", "n't", "never", "no"]
    // 4. Check if the TextBlob's words contain any negation words.
    FOR EACH word IN blob.words:
        IF word.lower() is in negation_words:
            // If a negation word is found, invert the polarity score.
            // The `_replace` method is used to create a new sentiment object with the updated polarity.
            sentiment = sentiment._replace(polarity = sentiment.polarity * -1)
            // Break the loop after the first negation word is found.
            BREAK
    // 5. Return the (potentially modified) sentiment object.
    RETURN sentiment
END FUNCTION

END
```

Figure 2. TextBlob negation pseudocode.

3.3.2. Negex with SpacyTextBlob

The subsequent approach integrated Negex and SpacyTextBlob to improve negation detection and sentiment computation (Figure 3). Negex was employed to accurately identify negations in complex sentences, reducing false positives that could arise from misclassification. This approach helps narrow the scope of negation words [47]. After identifying negations, we computed its sentiment using SpacyTextBlob. When negation terms were detected, sentiment scores were inverted by multiplying the original value by -1 switching polarity from positive to negative or vice versa.

3.3.3. Antonym–Synonym-Only

We employed antonym and synonym substitution to handle negation, as illustrated in Figure 4. If a negated term is detected, the subsequent phrase is replaced with its antonym to reverse the intended meaning. Take the following expression: "*The scenery in the previous location is not stunning*" became "*The scenery in the previous location is terrible*". If an antonym was available in *WordNet*, it was retrieved and used to replace the phrase; otherwise, the first antonym from the list was applied. When a synonym from *synsets* was suitable for a negated expression, it replaced the original term to convey the intended meaning. If neither an antonym nor synonym was found, the original word was retained.

```
START

FUNCTION negex_sentiment(doc):
  // 1. Initialize an empty list to store sentiment results for each tweet
  SET results = []

  // 2. Define a list of common negation terms
  SET negation_terms = ["not", "no", "never", "n't"]

  // 3. Iterate over each sentence in the input document
  FOR EACH sentence IN doc.sents:
    // Perform standard sentiment analysis on the entire sentence
    // TextBlob.sentiment() returns a tuple (polarity, subjectivity)
    SET blob_sentiment = doc._.blob.polarity

    // 4. Initialize negation flag
    SET is_negated = FALSE

    // 5. Check for the presence of any negation term within the sentence
    FOR EACH token IN sentence:
      IF token.lower() IN negation_terms:
        SET is_negated = TRUE
        BREAK // Exit inner loop once a negation is found

    // 6. Adjust sentiment based on negation detection
    IF is_negated:
      // 7. Invert the polarity score if a negation was found
      SET corrected_polarity = blob_sentiment.polarity * -1
    ELSE:
      // 8. Otherwise, use the original polarity
      SET corrected_polarity = blob_sentiment.polarity

    // 9. Determine the final sentiment label based on the corrected polarity
    IF corrected_polarity > 0:
      SET final_sentiment = "positive"
    ELSE IF corrected_polarity < 0:
      SET final_sentiment = "negative"
    ELSE:
      SET final_sentiment = "neutral"

    // 10. Append the final sentiment label to the results list
    APPEND final_sentiment TO results

  // 11. Return the list of sentiments for each tweet/sentence
  RETURN results
END FUNCTION

END
```

Figure 3. Negex with SpacyTextBlob negation pseudocode.

3.3.4. Antonym–Synonym with Second Rule-Based

The fourth negation we experienced was almost like the third one (antonym–synonym-only). The difference was that we applied another rule-based on terms that have no antonym or synonym. If an antonym–synonym pair was not found, we reapplied a rule-based approach by multiplying the sentiment score by −1. The purpose of this step is to accommodate negated terms that have no antonym–synonym.

```
START

FUNCTION handle_negation(text):
    // 1. Define a list of words that indicate negation
    negation_cues = [ "not", "no", "never", "n't" ]

    // 2. Split the input text into a list of individual words
    words = split the text by spaces

    // 3. Initialize a flag to track if the previous word was a negation cue
    is_negated_word = FALSE

    // 4. Initialize an empty list to store the modified words
    result_list = []

    // 5. Loop through each word and its index in the words list
    FOR EACH word AT index `i` IN words:
        // 6. Check if the current word is a negation cue
        IF word IS IN negation_cues:
            // 7. Set the flag to indicate negation is active
            is_negated_word = TRUE

        // 8. If the negation flag is active, process the current word
        ELSE IF is_negated_word IS TRUE:
            // 9. Find synonyms for the current word using a lexical database (e.g., wordnet)
            synonyms = get synonyms for word

            // 10. Check if any synonyms were found
            IF synonyms EXIST:
                // 11. Initialize an empty list to store antonyms
                antonyms = []

                // 12. Loop through each synonym (synset) found
                FOR EACH syn IN synonyms:
                    // 13. Get all the lemma names for the current synonym
                    FOR EACH lemma IN syn's lemmas:
                        // 14. If the lemma has an antonym, add it to the antonym list
                        IF lemma has an antonym:
                            antonyms = antonyms + lemma's first antonym

                // 15. Check if any antonyms were successfully found
                IF antonyms EXIST:
                    // 16. Append the first antonym found to the result list
                    append antonyms[0] to result_list
                ELSE:
                    // 17. If no antonym is found, prepend "not_" to the word
                    append ("not_" + word) to result_list
            ELSE:
                // 18. If no synonyms (and thus no antonyms) are found, prepend "not_" to the word
                append ("not_" + word) to result_list

            // 19. After processing the word, turn off the negation flag
            is_negated_word = FALSE

        // 20. If the current word is not a negation cue and is not negated, add it unchanged
        ELSE:
            append word to result_list

    // 21. Join the words in the result list back into a single string
    RETURN join the words in result_list with spaces
END FUNCTION

END
```

Figure 4. Antonyms–synonyms-only negation pseudocode.

3.3.5. Zero-Shot

The fifth method for handling negation was zero-shot classification, as shown in Figure 5. We applied pre-trained model "*zero-shot-classification*" using "*facebook/bart-laarge-mnli*" model. This classification model was initially applied in image processing to predict unseen pictures [48] and can be applied directly to predict new data without training [49,50].

Recently, this model classifier has been adapted to various natural language processing (NLP) tasks, including document classification [51], text classification [52], entity recognition [53], and relation extraction [54].

```
START

FUNCTION classify_with_negation(text, candidate_labels, hypothesis_template = "This text is about {}."):
  // Step 1: Perform initial zero-shot classification using a pre-trained model.
  // The classifier uses the hypothesis template to predict likelihoods for each candidate label.
  // `multi_label=False` means it assumes a single label is the most likely.
  classifier = load_pipeline(task="zero-shot-classification", model="facebook/bart-large-mnli")
  results = classifier(text, candidate_labels, hypothesis_template, multi_label=FALSE)
  // Step 2: Process the input text with spaCy for dependency parsing.
  // This is used to find negation dependencies later.
  nlp = load_spacy_model("en_core_web_sm")
  doc = nlp(text)
  // Step 3: Find all tokens that have a "neg" dependency (negation).
  negated_tokens = [token for token in doc where token.dependency is "neg"]
  // Step 4: Initialize a new results dictionary to store the enhanced scores.
  enhanced_results = new Dictionary
  enhanced_results.labels = new List
  enhanced_results.scores = new List
  // Step 5: Iterate through the initial classification results.
  FOR each label, score in zip(results.labels, results.scores):
    // Assume no negation effect by default.
    negation_effect = 1.0
    // Step 6: Check for any negation that might apply to this label.
    FOR each neg_token in negated_tokens:
      // Extract keywords from the label.
      label_keywords = split_string(label, " ")
      // Check if any label keyword is affected by the negation token.
      FOR each keyword in label_keywords:
        keyword_token = find_token_in_doc_by_text(doc, keyword)
        // Check if the negation token's head is an ancestor of the keyword token,
        // or if the negation token's head is the keyword token itself.
        // This confirms the keyword is within the scope of the negation.
        IF keyword_token is not NULL AND (neg_token.head is ancestor of keyword_token OR neg_token.head is keyword_token):
          negation_effect = -1.0 // Invert the score effect
          BREAK // Exit inner loop once a negation is found for this label
      IF negation_effect is -1.0:
        BREAK // Exit outer loop once a negation is found for this label

    // Step 7: Apply the negation effect and add to the enhanced results.
    enhanced_results.labels.add(label)
    enhanced_results.scores.add(score * negation_effect)

  // Step 8: Return the final, enhanced classification results.
  RETURN enhanced_results
END FUNCTION

END
```

Figure 5. Zero-shot negation pseudocode.

3.3.6. Hybrid Negation

The last negation technique we implemented was hybrid negation. This method consists of rule-based, implicit, double negations, dependency parsing to analyze nested clauses, MLP-learned weighting, and TextBlob integration. Figure 1 displays the sentiment computation flow chart using hybrid negation method.

Dependency Parsing

The code uses spaCy model "*en_core_web_sm*" (pre-trained English pipelines) to build a dependency tree to map the grammatical relationships between words to identify the sentence structure and clause boundaries. Figure 6 displays the pseudocode to set up a dependency tree. The depth of a token within this tree helps detect nested sub clauses, a key feature of complex sentences. The grammatical scope for a token can be established by routing its location within the dependency tree.

Figure 7 displays dependency relations from the following sentence: "*Although the food was delicious, the service was slow and inefficient*". The word "*was*" acts as the head (governor) of the words "*service*, *slow*", and "*inefficient*", while "*and*" is the coordinating conjunction for words "*slow*" and "*inefficient*". "*Service*" is the subject dependent of "*was*", while "*slow*" and "*inefficient*" are adjectival predicate dependents. The entire first clause ("*Although the food was delicious*") acts as an adverbial clause modifier (*advcl* label), and the second occurrence

of "*was*" serves as the head verb of the main clause. So, the word "*was*" (in the second clause) is the ultimate head (root) of the entire sentence's dependency structure.

```
START

// 1. Load the spaCy "en_core_web_sm" model and assign it to the `nlp` variable.
SET nlp TO spaCy.load("en_core_web_sm")

// 2. Define a function `get_sentence_tree` that takes a `sentence` as input.
FUNCTION get_sentence_tree(sentence):
  // 3. Process the input sentence using the loaded spaCy NLP model.
  // This produces a `Doc` object which contains a dependency parse tree.
  SET doc TO nlp(sentence)

  // 4. Return the processed Doc object, which represents the dependency tree.
  RETURN doc
```

Figure 6. Setting up a dependency parse tree.

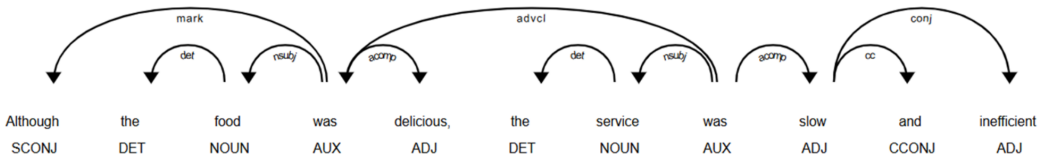


Figure 7. Sample visual representation dependency tree.

Rule-Based, Implicit, and Double Negation

Explicit negation has a direct or asserted negative meaning, while implicit negation has a negative meaning carried from presupposition or pragmatic inference rather than what is literally stated [55]. The explicit negation identifies words like "*not*", "*never*", "*n't*", and "*no*" as negated expressions, while implicit negation relies on context or specific words that imply denial or absence, for instance, "*barely*", "*hardly*", or "*seldom*". For example, "*He seldom talks about the past*" implies "*He does not talk about the pass*". Double negation uses two negative terms to form a statement, such as "*not in-significant*" or "*not hardly interesting*". Setting up rule-based, implicit, and double negation in this project was explained in Figure 8. In Figure 9, explicit negation will switch a positive sentiment to a negative or vice versa by multiplying the sentiment score with a negative one, while implicit negation will partially switch the sentiment by multiplying with -0.7. Furthermore, double negation will neutralize each other's negation, so two negative words will cancel each other's. For example, an expression "*not in-significant*" can be translated as "*significant*".

TextBlob Integration

As shown in Figure 9, applying TextBlob provides the foundational sentiment polarity for each individual phrase in complex sentences. Next, the original sentiment score will be modified by the rest of the system according to its complexity and clause structure.

MLP (Multilayer Perceptron) Learned Weight Mechanism

An MLP neural network will learn the optimal weighting of sentiment scores from different clauses according to their scopes and structures in nested clauses. Instead of having a uniform inversion, the MLP provides a nuanced approach because it teaches weight sentiment scores according to sentence features and structures. The MLP model was created using input layer with three features (negation count, clause depth, and sentence length) and output layer with one value (the weighing factor). The model learns the possibility that a longer sentence with multiple nested clauses might have a different weight than a simple expression. In Figure 8, we applied a sigmoid function, σ, to ensure

that the weight value is between zero and one. The application of MLP-learned weights (the last step) toward various clause complexities and structures, as shown in Figure 10.

```
CLASS ComplexNegationSentiment:

  // 5. Constructor method to initialize the class instance.
  // It takes a spaCy NLP model as input.
  METHOD __init__(spacy_nlp_model):
    SET self.nlp TO spacy_nlp_model

    // 6. Define rule-based lists for different types of negation cues.
    SET self.explicit_negations TO {'not', 'n\'t', 'no', 'never'}
    SET self.broad_negations TO {'barely', 'hardly', 'seldom'}

    // 7. Build a placeholder Multilayer Perceptron (MLP) model.
    SET self.mlp_model TO self._build_mlp()

  // 8. A private method to simulate building an MLP model.
  METHOD _build_mlp():
    // Define a neural network architecture.
    CREATE model as a new neural network with:
      - INPUT layer with 3 features (e.g., negation count, clause depth, sentence length)
      - OUTPUT layer with 1 value (the weighting factor)
    RETURN model

  // 9. A private method to run the MLP model and get a weighting factor.
  METHOD _run_mlp(features):
    // 10. Simulate converting a list of features into a tensor format.
    CONVERT features_list TO features_tensor

    // 11. Pass the features through the pre-trained MLP model.
    SET weight TO self.mlp_model(features_tensor)

    // 12. Apply a sigmoid function to ensure the weight is between 0 and 1.
    SET weight_sigmoid TO apply sigmoid function TO weight

    // 13. Return the scalar value of the weight.
    RETURN weight_sigmoid as a single number

  // 14. Main method to analyze the sentiment of a given text.
  METHOD analyze_sentiment(text):
    // 15. Process the input text using the spaCy NLP model.
    SET doc TO self.nlp(text)

    // 16. Initialize variables for tracking sentiment and features.
    SET total_polarity TO 0.0
    SET negation_count TO 0

    // 17. Determine the max clause depth and group clauses by depth.
    // The pseudocode captures the intent of finding the max depth.
    SET clause_depths TO empty list
    FOR EACH token IN doc:
      SET depth TO number of ancestors of token
      ADD depth TO clause_depths
```

Figure 8. Set up of a ruled-base, implicit, double, MLP model and nested clause negation.

Figure 1 displays the hybrid negation sentiment computation flow chart. For each clause i, a base sentiment score, s_i, is calculated using the TextBlob approach. The weight of w_i is determined by an MLP (Multilayer Perceptron) using a sigmoid function, σ, s to ensure the weight value is between 0 and 1 for the clause based on a set of linguistic features, f_i. The polarity score computed by TextBlob, the polarity score provided by our rule-based module, and the score supplied by the dependency-parse module are the input vector for the MLP. The purpose of using the MLP is to provide a non-linear learning function that combines the best outputs of the various components to yield the final sentiment score, which serves as the actual learning mechanism for determining when to rely on the

rule-based or dependency-based output. Next, the final sentiment score, S, is calculated as the weighted sum of the individual clause sentiment scores.

$$w_i = \sigma(MLP(f_i)) \tag{2}$$

$$S = \sum_i w_i \cdot s_i \tag{3}$$

```
SET max_depth TO max value in clause_depths IF clause_depths is not empty ELSE 0

// 18. Iterate through each sentence to analyze its sentiment.
FOR EACH sentence IN doc.sents:
  // 19. Get the base sentiment of the sentence using an external tool (TextBlob).
  SET local_polarity TO get sentiment polarity of sentence.text USING TextBlob

  // 20. Initialize boolean flags for negation analysis.
  SET is_negated TO FALSE
  SET double_negated TO FALSE

  // 21. Rule-based negation handling, iterating through tokens in the sentence.
  FOR EACH token IN sentence:
    // 22. Handle explicit negation keywords.
    IF token.lower_ FORM is in self.explicit_negations:
      TOGGLE is_negated (e.g., TRUE becomes FALSE, FALSE becomes TRUE)
      INCREMENT negation_count by 1

    // 23. Handle "broad" or implicit negation keywords.
    IF token.lower_ FORM is in self.broad_negations:
      MULTIPLY local_polarity BY −0.7 (partially invert sentiment)

  // 24. Check for double negation.
  IF negation_count > 1:
    SET double_negated TO TRUE

  // 25. Placeholder for more complex, dependency-based negation analysis.
  // e.g., "The food was not bad, but not good either"
  FOR EACH token IN sentence:
    IF token.dependency IS 'cc' AND token.text.lower_ FORM is in {'but', 'although'}:
      // 26. LOGIC FOR ANALYZING CONJUNCTIONS AND RE-EVALUATING PASS
      CONTINUE

  // 27. Apply rule-based sentiment inversion.
  IF is_negated AND NOT double_negated:
    MULTIPLY local_polarity BY −1.0 (fully invert sentiment)
  ELSE IF double_negated:
    // 28. Negations cancel out, so no change in polarity.
    // 29. MULTIPLY local_polarity BY 1.0 (no change)
    CONTINUE
```

Figure 9. Analyzing sentiment using various negations.

3.4. SMOTE

The initial imbalanced dataset was divided into training and testing sets. Then, a vectorization model was fitted exclusively on the training dataset which would convert the raw text of the training set into a numerical feature space. Next, the SMOTE (synthetic minority over-sampling technique) was applied to the vectorized training data that would create synthetic new minority class samples to address the imbalanced data. After having balanced training data, machine learning models trained the new-adjusted training data. Finally, the trained model evaluated untouched testing dataset to measure model performance.

```
      // 30. Get features for MLP weighting.
      SET features TO a list CONTAINING:
        - negation_count
        - max_depth
        - sentence length

      // 31. Run the MLP model to get a learned weighting factor.
      SET learned_weight TO self._run_mlp(features)

      // 32. Apply the MLP weight to the local polarity and add to total.
      ADD (local_polarity * learned_weight) TO total_polarity

    // 33. Normalize the total polarity by the number of sentences.
    IF there are sentences in doc:
      RETURN total_polarity DIVIDED BY number of sentences
    ELSE:
      RETURN 0.0
END FUNCTION

END
```

Figure 10. Applying an MLP model as a learned weight mechanism.

3.5. Wilcoxon Signed Rank Test

The final step was to apply to the *Wilcoxon Signed Rank Test* to the model performance. The W-Statistic test is a non-parametric statistical test that is useful for ranked or ordinal data to provide an alternative for analyzing paired observations without requiring a normally distributed dataset [56]. A Z-score is computed to perform hypothesis testing using a z-score formula, where n is the number of data, W_+ is the sum of positive signs, W_- is the sum of negative signs, and t ties data (have the same rank).

$$z = \frac{MAX\,(W_+, W_-) - \frac{n(n+1)}{4}}{\sqrt{\frac{n(n+1)(2n+1)}{24} - \frac{t^3 - t}{48}}} \tag{4}$$

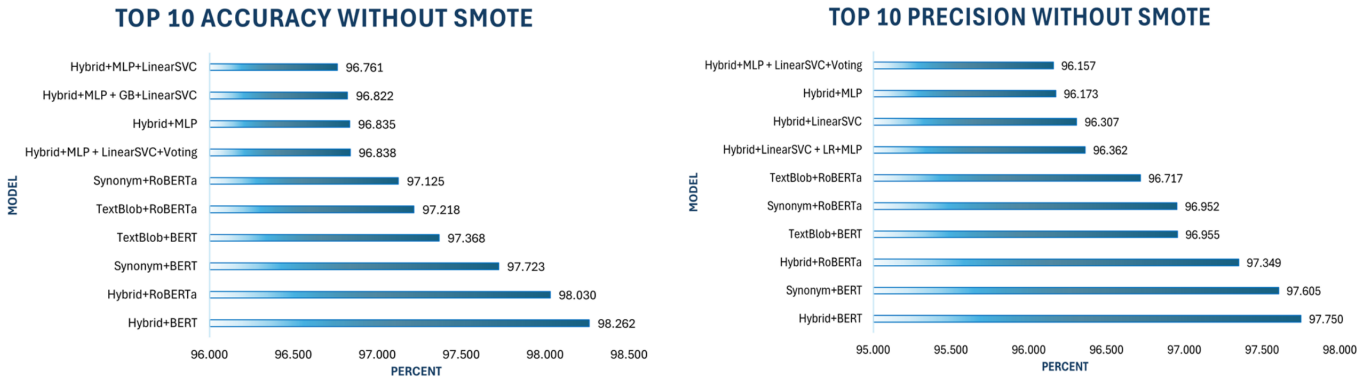
We examined whether there was a statistically significant improvement when we applied the SMOTE imbalance technique on the negation handling techniques. Furthermore, we evaluated whether the hybrid negation performance significantly surpassed the other negation methods.

4. Results

We examined the impact of the SMOTE imbalance technique employing negation handling on text sentiment analysis. We used classification models without the SMOTE as a control group in this study. The following sections present the results of each method (full table can be found in Appendix A).

4.1. Control Classification Model (Without SMOTE)

Figure 11 represents the model performance without applying the SMOTE imbalance technique (serving as a control group). Without applying SMOTE, we achieved the highest accuracy of 98.262% using the hybrid negation technique applying a BERT model classifier as the final prediction. The next highest accuracy (98.030%) was achieved by employing a similar negation handling method (hybrid negation) and applying a RoBERTa classification model for the final prediction. The third-highest accuracy (97.723%) was achieved by applying antonym–synonym with a second rule-based approach using a BERT model classifier.

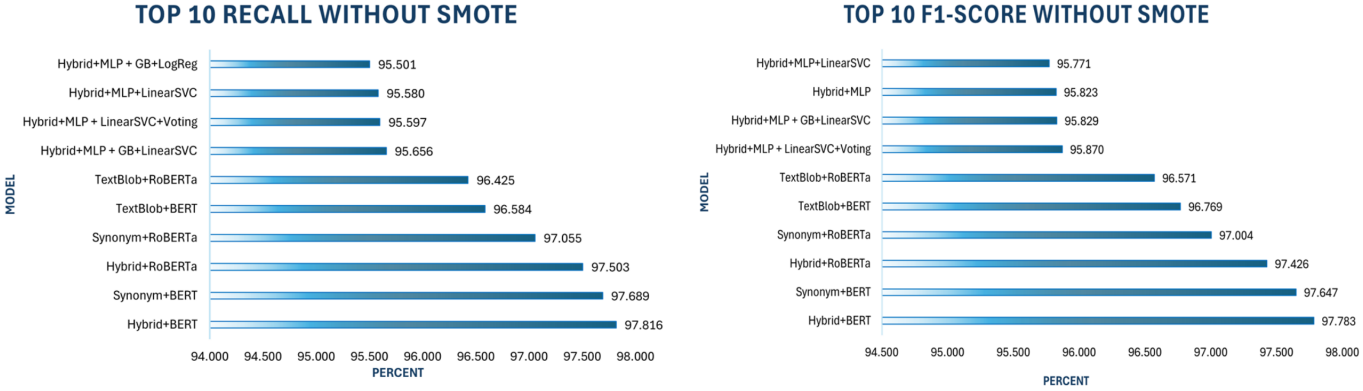


(**a**) (**b**)

Figure 11. Control model performance: (**a**) top 10 model accuracy without SMOTE; (**b**) top 10 model precision without SMOTE.

Without applying the oversampling technique, we achieved a model precision of 97.750% by utilizing the hybrid negation with the BERT model classifier. The next highest precision (97.605%) was achieved by applying the antonym–synonym method in conjunction with a second rule-based approach and the BERT model classifier. The third-highest precision (97.349%) was accomplished using the hybrid negation technique with the RoBERTa classification model.

Figure 12 shows the top 10 recall (a) and F1-score (b) without applying SMOTE. The application of the hybrid negation using the BERT model classifier achieved the highest recall of 97.816%. The second-highest model recall (97.689%) was achieved using the antonym-synonym with a second rule-based approach and the BERT as the final prediction model. The third-highest model recall (97.503%) was achieved by applying hybrid negation and using RoBERTa as the final prediction model.



(**a**) (**b**)

Figure 12. Model performance: (**a**) top 10 model recall without SMOTE; (**b**) top 10 model F1-score without SMOTE.

Lastly, handling negated expressions using a hybrid method combined with BERT classification achieved the highest F1-score of 97.783%, as shown in Figure 12. The second-highest F1-score (97.647%) was achieved by applying the antonym–synonym negation with a second rule-based approach and BERT as its final prediction model. The third-highest model F1-score of 97.426% was acheived by applying the hybrid negation with RoBERTa as the model classifier.

Table 2 shows the lowest model accuracies and precision without applying SMOTE (control group). The lowest model accuracy (69.059%) was achieved when we used the antonym–synonym with a second rule-based approach using traditional gradient boosting

as a classification model. The next lowest model accuracy was 69.059% when we used the antonym–synonym with only the gradient-boosting model classifier. Lastly, the next lowest model accuracy of 73.307% resulted when we operated TextBlob negation handling and the gradient-boosting model classification.

Table 2. The lowest model accuracies and precisions in the control group.

Lowest Accuracy		Lowest Precision	
Model	**Percent**	**Model**	**Percent**
Synonym + GB	69.059	Zero-Shot + MLP	70.045
Synonym-only + GB	69.059	Zero-Shot + GB	71.147
TextBlob + GB	73.307	Zero-Shot + MLP + LinearSVC + Voting	73.727

The lowest model precision of 70.045% was achieved when we applied a zero-shot negation handling technique using MLP as its final model classifier. The second-lowest model precision (71.147%) was observed when we applied zero-shot using gradient boosting as the classification model. The next lowest model precision (73.727%) occurred when we used the zero-shot negation method applying MLP and LinearSVC as its base model with voting classifier as its final prediction.

Table 3 displays the lowest recall and F1-score. The lowest recall, 51.552%, happened when we used zero-shot negation handling and gradient boosting as its model classifier. The next lowest model recall (58.029%) was observed when we applied the zero-shot negation technique with random forest as the final classification model. Lastly, the next lowest recall, 59.564%, occurred when we utilized zero-shot negation and logistic regression as the classification model.

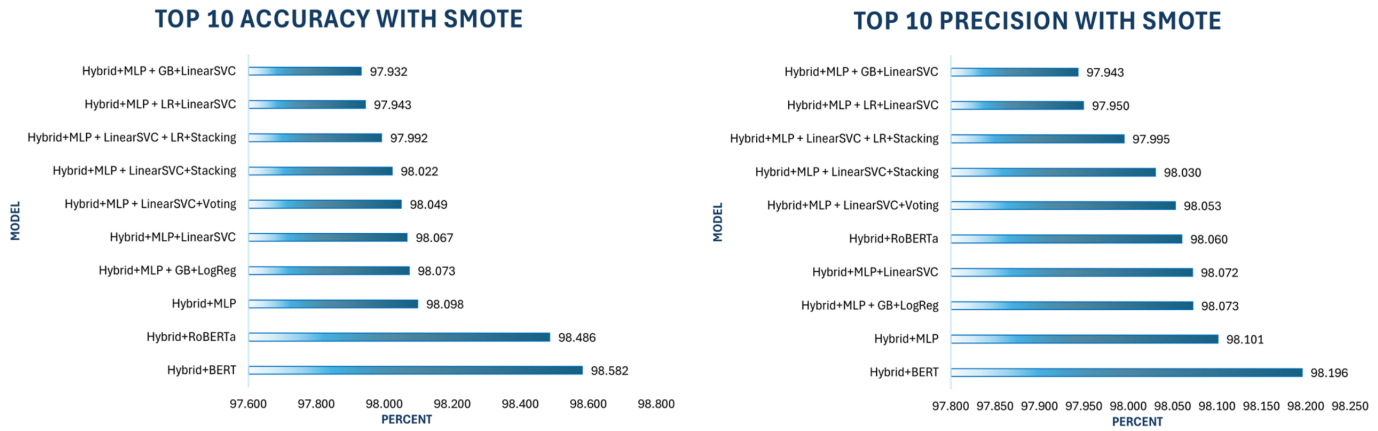
Table 3. The lowest model recalls and F1-scores in the control group.

Lowest Recall		Lowest F1-Score	
Model	**Percent**	**Model**	**Percent**
Zero-Shot + GB	51.552	Zero-Shot + GB	55.531
Zero-Shot + RF	58.029	Zero-Shot + RF	63.278
Zero-Shot + LR	59.564	Zero-Shot + LR	63.453

Furthermore, the lowest F1-score of 55.531% was exhibited by applying a zero-shot negation approach with gradient boosting as the model classifier. The next lowest F1-score (63.278%) was achieved by applying zero-shot negation handling using random forest as its final classification model. The third-lowest F1-score (63.453%) occurred when we applied similar negation method (zero-shot) using Logistic Regression (LR) as the model classifier.

4.2. Classification Model with SMOTE

Figure 13 represents the ten highest model accuracy (a) and model precision (b) classification model performances applying the SMOTE imbalance technique on negation handling techniques. The highest accuracy of 98.582% was achieved when we applied hybrid negation by utilizing a BERT model classifier. The next highest model accuracy (98.486%) was accomplished by applying the same negation handling technique (hybrid method) using a RoBERTa classification model. The third-highest model accuracy (98.098%) was achieved when we applied hybrid approach to handle negation and utilized Multi-Layer Perceptron (MLP) as its model classifier.

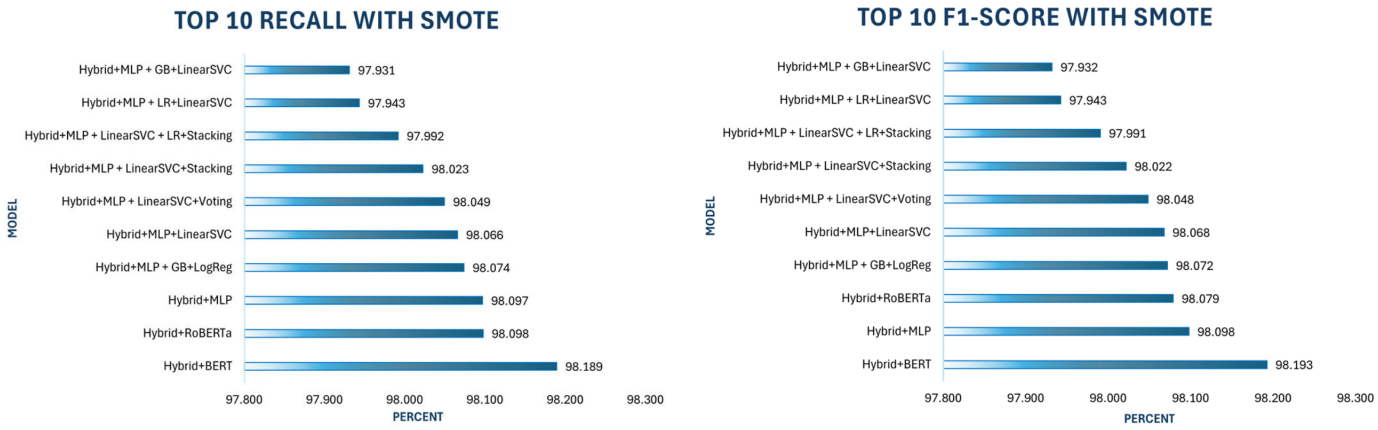


(**a**) (**b**)

Figure 13. Model performance: (**a**) top 10 model accuracy with SMOTE; (**b**) top 10 model precision with SMOTE.

Like model accuracy, the best precision (98.196%) was achieved by employing hybrid negation handling technique, utilizing BERT as its final prediction method, shown in Figure 13. The next highest precision (98.101%) was achieved by applying same negation (hybrid method) utilizing MLP and gradient boosting as its base model and the logistic regression as its final prediction. The third-highest precision (98.073%) was achieved by employing hybrid approach as its negation handling applying MLP and gradient boosting as the base model and logistic regression for its final prediction.

Figure 14 represents the ten highest recall (a) and F1-score (b). The best recall (98.189%) was achieved by employing hybrid negation, utilizing a BERT as its final prediction method. The second-highest recall (98.098%) was achieved by applying the same negation technique (hybrid approach) and using RoBERTa as the final prediction. The third-highest recall (98.097%) was achieved using hybrid negation handling approach employing MLP as its final model classifier.



(**a**) (**b**)

Figure 14. Model performance: (**a**) The top 10 model recall with SMOTE; (**b**) The top 10 model F1-score with SMOTE.

Like the recall score, the best F1-score (98.193%) was achieved by applying hybrid negation, using BERT as its final model prediction, shown in Figure 14. The second-highest F1-score (98.098%) was achieved using a similar negation handling method with MLP as its final classification model. The third-highest recall (98.079%) was achieved using hybrid negation handling approach, combined with RoBERTa as the final prediction model.

The lowest accuracy (69.121%) was achieved when we applied antonym–synonym-only using the gradient-boosting model classifier, as shown in Table 4. The second-lowest model accuracy (70.059%) was achieved when we applied the antonym–synonym negation with second rule-based method using the gradient-boosting model for its final model classifier. The third-lowest model accuracy (75.219%) was achieved when we applied TextBlob negation handling with gradient boosting as the classifier.

Table 4. The lowest model accuracies and precisions with SMOTE.

Lowest Accuracy		Lowest Precision	
Model	**Percent**	**Model**	**Percent**
Synonym-only + GB	69.121	Synonym + GB	72.626
Synonym + GB	70.059	Zero-Shot + BERT	74.060
TextBlob + GB	75.219	Synonym-only + GB	75.018

Next, our experiment achieved the lowest model precision (72.626%) when we used an antonym synonym with second rule-based method, applying gradient boosting as the model classification, as displayed in Table 4. The second-lowest precision (74.060%) occurred when we utilized the zero-shot negation handling technique with the BERT model classifier. Furthermore, our experiment showed that the next lowest precision (75.018%) occurred when we applied the antonym–synonym-only applying the gradient-boosting model classifier.

Table 5 represents the lowest model recalls and F1-scores with SMOTE imbalance technique. The lowest model recall of 69.193% occurred when we applied antonym–synonym-only negation handling with gradient boosting. The following lowest recall (69.882%) occurred when applying the zero-shot negation handling technique with the BERT classification model. The third-lowest recall (70.176%) was achieved by applying an antonym synonym with second rule-based negation technique and gradient boosting for its model classifier.

Table 5. The lowest model recalls and F1-scores with SMOTE.

Lowest Recall		Lowest F1-Score	
Model	**Percent**	**Model**	**Percent**
Synonym-only + GB	69.193	Synonym-only + GB	68.777
Zero-Shot + BERT	69.882	Synonym + GB	69.854
Synonym + GB	70.176	Zero-Shot + BERT	71.911

Furthermore, the lowest model F1-score (68.777%) was achieved when we applied the antonym–synonym-only feature with the gradient-boosting model classifier, as displayed in Table 5. The next lowest F1-score (69.854%) utilized the antonym synonym with second rule-based negation handling, employing gradient boosting as its model classifier. Lastly, applying the zero-shot negation technique with the BERT model classifier, we achieved the third-lowest F1-score of 71.911%.

Figure 15 summarizes the negation handling performance in this research study. Hybrid negation surpassed the other negation handling methods with the highest accuracy of 98.582%. Secondly, comparing average accuracies according to negation methods, we found that the hybrid method also achieved the highest average accuracy (95.789%). Next, matching the median accuracies based on the negation handling techniques, hybrid negation (96.836%) outperformed the others' medians. Furthermore, evaluating the accuracy of the modes by the negation handling methods, the hybrid approach (96.570%) surpassed the other methods. Lastly, assessing the minimum accuracy based on the negation han-

dling techniques, we found that hybrid negation still showed an outstanding performance (77.019%) compared to the other techniques. The summary statistics of the various negation handling techniques showed that text sentiment analysis using the hybrid negation technique outperformed the other negation handling methods.

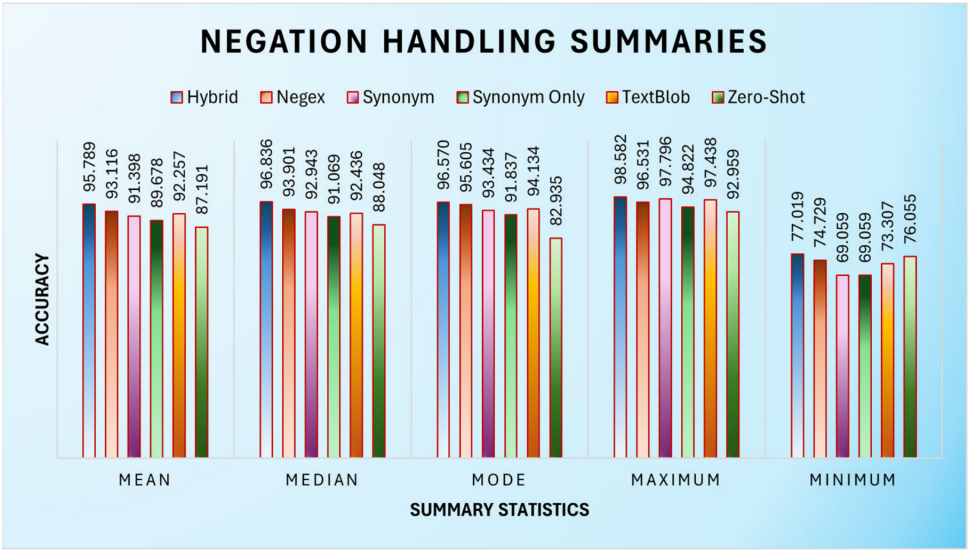


Figure 15. Negation handling summaries.

5. Discussion

According to our experiment results, the highest model accuracy of 98.582%, was obtained when hybrid negation handling was combined with a BERT classifier for the final prediction. Using the same negation strategy with a BERT-based classification model also produced the highest precision, reaching 98.196%. Likewise, applying hybrid negation with BERT yielded the highest recall of 98.189%. The best F1-score, 98.193%, was achieved under the same configuration. Overall, hybrid negation consistently enabled classification models (particularly those using BERT) to outperform all other negation handling techniques.

Like the performance using the SMOTE imbalance technique, the highest accuracy in the control group (98.262%) was achieved by applying hybrid negation with BERT as its model classifier. In addition, the highest precision in the control group (97.750%) was achieved by utilizing a similar negation technique (hybrid) with BERT as its final prediction model. Furthermore, applying hybrid negation with BERT as its model classifier in the control group achieved the highest recall of 97.816%. Lastly, the highest F1-score (97.783%) in the control group was accomplished by applying the hybrid negation technique with BERT as the classifier.

The hybrid negation performed much better than the other negation handling techniques in the control experiment group. There were 15 out of 20 models with the best accuracy in our experiment (75%) that were accomplished when we utilized the hybrid negation technique. Two out of twenty best model accuracies (10%) in this research project were achieved when we applied the TextBlob negation handling method. Next, there were 1 out of 20 best model accuracies in this study (5%) that were achieved when we employed the Negex method. Additionally, 2 out of 20 best model accuracies (10%) in this project were achieved when we applied antonym–synonym with a second rule-based negation handling method.

Like the control group, hybrid negation consistently outperformed the other negation handling techniques when we applied the SMOTE imbalance method. There were 16 out of 20 best model accuracies (80%) that succeeded when we employed a hybrid approach. Next, 2 out of 20 (10%) achieved best accuracies when we applied antonym–synonym with

a second based-rule. At the same time, the TextBlob negation handling method succeeded in achieving 2 out of 20 best model accuracies (10%) in our experiment.

Table 6 represents an ablation study of the hybrid negation handling technique. It shows that reducing a component in hybrid negation decreased all model performances. Without adding a "*scope*" component to the hybrid, the model accuracy decreased the most (by 0.402%). The highest reduction in precision (by 0.441%) happened when we excluded a "*double negation*" component in the system. Next, the greatest drop in recall (by 0.594%) occurred when we eliminated "*scope*" components in the hybrid negation. Furthermore, the highest decrease in F1-score (by 0.515%) happened when we excluded "*scope*" components in the system. Consequently, the scope element in the hybrid negation seems to have greatly affected the system because misinterpreting the scope of sentences can lead to misinterpretation of the information which leads to inaccurate sentiment analysis.

Table 6. Ablation study model performance.

Treatments	Accuracy	Precision	Recall	F1-Score
No Scope	98.179	97.759	97.595	97.677
No Double	98.342	97.755	97.972	97.863
No Implicit	98.397	97.936	97.994	97.965
Uniform Weight	98.403	97.901	97.823	97.862
Full Hybrid	98.582	98.196	98.189	98.193

There is still a possibility that our hybrid model struggles to recognize linguistic ambiguity, such as in ambiguous sentences or sarcasm. Our model relies on predefined rules or features that may not be considered for nuanced human language use. The presence of sarcasm might lead to incorrect sentiment analysis. For example, "*Oh, fantastic! It was wonderful that the customer service hung up on me three times*", might be misclassified as a positive sentiment because of the words "*fantastic*" and "*wonderful*". The expression shows anger or dissatisfaction, which is considered a negative sentiment. Another possible misclassification hybrid model is when dealing with complex metaphors. For instance, "*The team's performance was a slow-motion train wreck*", might be translated into neutral, as it interpreted "*slow-motion train wreck*" as a noun, which is classified as neutral. In fact, the metaphorical meaning of the expression "*slow-motion train wreck*" describes a poor performance that is gradual and drawn out, which likely leads to negative sentiment.

For evaluation purposes, we conducted a comparative analysis of our proposed hybrid method against LLMs (large language models). We applied the "*cardiffnlp/twitter-roberta-base-sentiment*" model with the *transformers* library's *pipeline* function to perform sentiment analysis using the LLMs. Table 7 compares the performance of our suggested methods with that of the LLMs. For all metrics, the Hybrid methods surpass the LLM technique. The best model performance for the LLMs (accuracy of 93.855%, precision of 93.808%, recall of 93.833%, and F1-score of 93.807%) was achieved when we applied the MLP-LinearSVC classifier. On the other hand, our proposed methods using the same model (MLP-LinearSVC) achieved 98.067% accuracy, 98.072% precision, 98.066% recall, and 98.066% F1-score.

The descriptive statistics in Table 8 show that applying SMOTE improved overall model performance. The average accuracy increased from 90.379% to 92.764%; its median rose from 92.254% to 93.197%; the minimum accuracy grew from 69.059% to 69.121%; and its maximum accuracy advanced from 98.262% to 98.582%. In addition, utilizing an imbalance technique (SMOTE) also improved the standard error from 0.569 to 0.500, the standard deviation from 6.074 to 5.340, and the sample variance from 36.896 to 28.514. Consequently, applying SMOTE advanced model performance in this research project.

Table 7. Comparison of the Hybrid with LLMs.

Model	Hybrid				LLMs			
	Accuracy	**Precision**	**Recall**	**F1-Score**	**Accuracy**	**Precision**	**Recall**	**F1-Score**
BERT	98.582	98.196	98.189	98.193	91.021	88.747	89.888	89.314
RoBERTa	98.486	98.060	98.098	98.079	92.944	89.875	93.831	91.811
MLP	98.098	98.101	98.097	98.098	93.344	93.360	93.318	93.279
MLP + GB + LogReg	98.073	98.073	98.074	98.072	93.764	93.734	93.741	93.714
MLP + LinearSVC	98.067	98.072	98.066	98.068	93.855	93.808	93.833	93.807

Table 8. Descriptive statistics of the control and SMOTE accuracy.

Descriptive Statistics	Control	SMOTE
Mean	90.379	92.764
Standard Error	0.569	0.500
Median	92.254	93.197
Mode	93.193	94.134
Standard Deviation	6.074	5.340
Sample Variance	36.896	28.514
Kurtosis	1.990	7.124
Skewness	−1.390	−2.417
Range	29.203	29.461
Minimum	69.059	69.121
Maximum	98.262	98.582
Sum	10,303.155	10,575.119
Count	114	114

We utilized classification models without SMOTE as a control experiment while applying SMOTE as a treatment group. It seems that applying the SMOTE imbalance technique has improved the classification model accuracy (as shown in Table 8). To validate this assumption, we used a one-tailed *Wilcoxon Signed Rank Test* to examine our hypothesis.

$$H_0: P_0 = P_1$$

$$H_1: P_0 < P_1$$

H_0 is the null hypothesis, while H_1 represents the alternative hypothesis. P_1 describes the performance of the SMOTE model, and P_0 is designated as the performance of the control group (without the SMOTE technique).

Our null hypothesis asserts that the metric performance of the control group (without the SMOTE technique) is the same as that of the treatment group (with the SMOTE). On the other hand, the alternative hypothesis posits that the treatment class outperformed the control group. We used 99% confidence intervals with a 1% significance level (a z-score of 2.33 as the cutoff).

A z-score value (Table 9) was created for accuracy, precision, recall, and F1-score from the SMOTE model performance. The table shows that the z-score for accuracy (8.186723131) is extremely high, and the *p*-value close to zero. In addition, the z-score for precision (7.892674018), recall (8.729583031), and F1-score (8.636278986) are also exceptionally high, meaning that the *p*-value is extremely low, close to zero. Using a 99% confidence interval (significant level of 1%), the z-scores of SMOTE are higher than the 2.33. Therefore, we rejected the null hypothesis. The *W-test* results suggest that the SMOTE imbalance technique improves model performances in this negation handling in text sentiment analysis. The result corresponds to research studies [57–59] that show that the oversampling method en-

hances model performance for its ability not simply to generate synthetic minority samples but also to remove misclassified samples [57] and reduce noisy samples [59], which improves model performances, predictive accuracy, and mitigates the risk of overfitting [57].

Table 9. Z-scores between the control and SMOTE method.

	Accuracy	Precision	Recall	F1-Score
Z-score	8.186723131	7.892674018	8.729583031	8.636278986

Next, to examine whether the hybrid negation handling performance surpassed that of the other negation techniques, we applied a one-tailed *Wilcoxon Signed Rank Test* with a 99% confidence interval (significance level 1%). The null hypothesis stated that the metric performance of hybrid negation was the same as that of the other negation techniques. In contrast, the alternative hypothesis stated that hybrid negation performed better than the other negation handling methods.

Table 10 presents z-scores for the hybrid negation technique compared with the other negation handling methods (Negex, synonym with second rule-based, synonym-only, TextBlob, and zero-shot). The z-scores were higher than the z-score of the confidence interval (z-score of 2.33). Consequently, we rejected the null hypothesis. Utilizing a 1% significance level (confidence interval of 99%), the *W-test* suggests that the hybrid negation technique significantly outperforms the other negation techniques.

Table 10. Z-scores between hybrid negation and the other methods.

	Negex	Synonym w/2nd Rule-Based	Synonym-Only	TextBlob	Zero-Shot	
Hybrid	5.344	5.373	5.373	5.373	5.344	Accuracy
	5.373	5.373	5.373	5.373	4.851	Precision
	3.024	5.243	5.330	5.373	5.359	Recall
	5.344	5.373	5.373	5.373	5.359	F1-score

Our research project surpassed the previous work studies. We successfully achieved the highest model performance (both with and without SMOTE) when we applied hybrid negation and the BERT model classification as the final prediction model. Our experiment, with an accuracy of 98.582%, outperformed the research study [25] (accuracy of 95.67%). The previous study employed Negex negation handling and SentiWordNet (SWN) sentiment computation in conjunction with the ANN classification model.

Additionally, without applying the SMOTE imbalance technique, our project with three classes achieved an accuracy of 98.262%, exceeding the previous research studies. A work study [25], having two classes (positive and negative), achieved the highest accuracy of 95.67%. Project [26] had the same three classes and achieved the best accuracy of 69.5%. Project [34] achieved an accuracy of 77.57%, experiment [37] achieved an accuracy of 77.3%, and study [34] achieved an accuracy of 91.79%. A research study [3] with three classes (positive, negative, and neutral) achieved the best accuracy of 67%.

The previous project [34] utilized antonym–synonym and SWN sentiment computation with a Logistic Regression model, achieving the highest model accuracy of 91.79%. Nevertheless, we outperformed the earlier project when we used the same negation handling method (antonym–synonym), a different sentiment computation (Vader SentimentIntensityAnalyzer), and a BERT model classifier (with SMOTE, achieving 94.822%, and without SMOTE, achieving 94.666%). Furthermore, when we modified the antonym–synonym handling method by applying a second rule-based, we achieved a much higher model performance (with SMOTE 97.796% and without SMOTE 97.723%).

According to the model performance, our model achieved the lowest accuracy of 69.059%, yet it outperformed the previous negation handling project. A research project [24], for example, achieved an accuracy of 58.67% by applying the hybrid RBF (radial basis function)-SVM (support vector machine) model with three classes (positive, negative, and neutral).

Most previous research projects [3,24,25,34] utilized SentiWordNet (SWN) to compute sentiment scores. Project [3] (an accuracy of 72%) applied rule-based negation by inverting the SWN sentiment score and word sense disambiguation model. In addition, the work study [24] achieved the highest accuracy of 58.67% by inverting the SWN polarity for negated expressions and utilizing a hybrid RBF-SVM model classifier. On the other hand, when we applied a different sentiment computation (TextBlob) and model classifier (BERT), we achieved better model performance, with SMOTE at 97.438% and without SMOTE at 97.368%. Consequently, applying a suitable sentiment computation improved classification model performance in text sentiment analysis.

6. Contributions

We would like to highlight the following contributions of this research project to negation handling in text sentiment analysis:

- This project designed and implemented a hybrid negation method using an MLP-learned weight mechanism that outperformed previous negation handling methods. Employing hybrid negation in this project, it achieved an accuracy of 98.582% using three classes (positive, negative, and neutral). This study outperformed the best previous work study [25] with a 95.67% accuracy rate, using two classes (positive and negative). The summary statistics of the hybrid negation method surpassed the other negation techniques.
- This research study experimented with six negation handling techniques (Hybrid, antonym–synonym, antonym–synonym with second rule-based approach, TextBlob, Negex with SpacyTextBlob, and zero-shot) to determine the most effective method for handling negation in text sentiment analysis. The previous research projects used a single negation handling technique [38]. For example, BERT memorization was applied, achieving an accuracy of 89% [38], while [29] utilizing neural network detection to identify negated expressions, achieving an F1-score of 93.09%. References [3] (accuracy of 67%) and [24] (accuracy of 58.67%) switched classes/polarity scores on identified negated tweets when handling negation in text sentiment analysis. References [25] (accuracy of 95.67%) and [26] (accuracy of 69.5%) applied the _NEG' suffix and scope of negation in their projects. The work study in [34] applied synonyms and antonyms to handle negation in text sentiment analysis, reaching the highest accuracy of 91.79%. Reference [39] applied synthetic, diminisher, and morphological methods to identify negated expressions with an accuracy of 83.3%.
- This project designed, progressed, and examined a total of 228 classification models. The previous studies on negation handling for sentiment analysis applied a maximum of six classification models [29,37], five model classifiers [25], four classifiers [39], and three classification models [26,34]. In contrast, others performed one model classifier only [24,36,38]. Our project achieved an accurate score of 98.582%, a precision of 98.196%, a recall of 98.189%, and an F1-score of 98.193%, surpassing the previous projects.
- Implementing a statistical hypothesis test, analysis in this research experiment enhanced the work studies' suggestion that applying the SMOTE method could improve model performance [57–59]. In addition, applying hybrid negation method in sentiment analysis significantly improved model performance.

7. Conclusions

The Internet has become the primary source of various information, and many individuals rely on mass media as a source of news. The web generates various opinions, comments, and expressions where online users share their views on social media platforms. Text mining has been a powerful tool for harvesting valuable information and a feasible instrument for analyzing public opinions on various issues on social media.

Sentiment analysis through machine learning and deep learning approaches have been implemented in various research projects. Text preprocessing plays a crucial role in sentiment analysis, significantly enhancing model performance. However, removing stop words as part of the text preprocessing could impact the result of sentiment analysis since negation words, such as "not", are included in the list of stop words. Many studies have been struggling to handle negation in complex and complicated structures.

This study's results suggest that applying hybrid negation technique has overcome the challenges of negation in complicated expressions. Employing hybrid approach surpassed other types of negation handling techniques. Our experiment achieved a model accuracy of 98.582%, precision of 98.196%, recall of 98.189%, and an F1-score of 98.193%. Additionally, the oversampling method, SMOTE, improved model performance by minimizing the impact of noisy samples.

8. Future Work

We want to apply this method to datasets from various resources with multilingual corpora. The goal is to test generalization, particularly with languages with different syntaxes. In addition, we aim to investigate the effect of negation handling sentiment analysis to various social networks datasets. Additionally, we intend to investigate whether there is a relationship between the proportion of sentiment classes and various datasets from diverse social media resources.

Author Contributions: Conceptualization, M.Q. and P.C.; methodology, M.Q. and P.C.; software, M.Q.; validation, M.Q. and P.C.; formal analysis, M.Q.; investigation, M.Q. and P.C.; resources, M.Q.; data curation, M.Q.; writing-original draft preparation, M.Q.; writing-review and editing, M.Q.; visualization, M.Q.; supervision, M.Q. and P.C.; project administration, M.Q.; funding acquisition, P.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding, and the APC was funded by UDC.

Data Availability Statement: The data is publicly available on ResearchGate at https://www.researchgate.net/publication/364110620_Covid19_Public_Tweets?channel=doi&linkId=633a5ad876e39959d6903819&showFulltext=true (accessed on 27 September 2025).

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

Appendix A.1. Control Model Performance

Table A1 shows model performance of the control group (without SMOTE) in this research study.

Table A1. Control model performance (without SMOTE).

No.	Negation Handling	Base Model	Final Prediction	Control			
				Accuracy	Precision	Recall	F1-Score
1	Hybrid		BERT	98.262	97.750	97.816	97.783
2	Hybrid		RoBERTa	98.030	97.349	97.503	97.426

Table A1. *Cont.*

No.	Negation Handling	Base Model	Final Prediction	Control			
				Accuracy	Precision	Recall	F1-Score
3	Synonym		BERT	97.723	97.605	97.689	97.647
4	TextBlob		BERT	97.368	96.955	96.584	96.769
5	TextBlob		RoBERTa	97.218	96.717	96.425	96.571
6	Synonym		RoBERTa	97.125	96.952	97.055	97.004
7	Hybrid	MLP + LinearSVC	Voting	96.838	96.157	95.597	95.870
8	Hybrid		MLP	96.835	96.173	95.499	95.823
9	Hybrid	MLP + GB	LinearSVC	96.822	96.006	95.656	95.829
10	Hybrid	MLP	LinearSVC	96.761	95.971	95.580	95.771
11	Hybrid	LinearSVC + LR	MLP	96.758	96.362	95.119	95.708
12	Hybrid	MLP + LR	LinearSVC	96.745	96.020	95.384	95.692
13	Hybrid		LinearSVC	96.732	96.307	95.161	95.706
14	Hybrid	MLP + GB	LogReg	96.701	95.788	95.501	95.642
15	Hybrid	MLP + LinearSVC + LR	Stacking	96.685	95.817	95.442	95.626
16	Hybrid	LinearSVC	MLP	96.579	96.085	94.940	95.485
17	Hybrid	MLP + LinearSVC	LogReg	96.570	95.640	95.321	95.478
18	Hybrid	MLP + LinearSVC + LR	Voting	96.570	95.917	95.037	95.459
19	Hybrid	MLP + LinearSVC	Stacking	96.544	95.653	95.288	95.467
20	Negex		BERT	96.474	95.542	95.087	95.314
21	Negex		RoBERTa	96.097	94.979	94.495	94.736
22	Hybrid	LinearSVC + GB	MLP	96.030	95.633	94.156	94.847
23	Synonym-only		BERT	94.666	94.561	94.562	94.561
24	Synonym-only		RoBERTa	94.439	94.294	94.312	94.303
25	Negex	LinearSVC + GB	MLP	93.928	93.246	91.053	92.041
26	Negex	LinearSVC	MLP	93.874	93.320	91.050	92.064
27	Negex	LinearSVC + LR	MLP	93.736	93.020	90.929	91.867
28	Negex		LinearSVC	93.720	92.894	90.688	91.678
29	Synonym	LinearSVC	MLP	93.631	93.569	93.251	93.382
30	Negex	MLP + LinearSVC + LR	Voting	93.615	92.235	91.318	91.753
31	Hybrid		LR	93.609	93.682	90.354	91.777
32	Synonym	MLP + GB	LogReg	93.401	93.238	93.156	93.195
33	Negex	MLP + LinearSVC	Voting	93.379	92.023	90.640	91.286
34	Synonym	MLP + LinearSVC	Stacking	93.337	93.136	93.008	93.067
35	Synonym	LinearSVC + LR	MLP	93.292	93.210	92.899	93.030
36	Synonym	LinearSVC + GB	MLP	93.193	93.054	92.799	92.909
37	Synonym		LinearSVC	93.193	93.056	92.805	92.912
38	Synonym-only		LinearSVC	93.193	93.056	92.805	92.912
39	Synonym	MLP + LR	LinearSVC	93.177	92.968	92.817	92.887
40	Synonym	MLP + LinearSVC	Voting	93.165	92.977	92.917	92.940
41	Negex	MLP + GB	LinearSVC	93.152	91.419	91.085	91.249
42	Negex	MLP + LinearSVC	Stacking	93.082	91.218	91.000	91.093
43	Synonym		MLP	93.018	92.852	92.650	92.741
44	Synonym-only		MLP	93.018	92.852	92.650	92.741
45	Negex	MLP + GB	LogReg	92.992	91.128	90.793	90.945
46	Synonym	MLP + GB	LinearSVC	92.986	92.819	92.696	92.744
47	Synonym	MLP	LinearSVC	92.944	92.734	92.669	92.697
48	Synonym	MLP + LinearSVC + LR	Stacking	92.941	92.809	92.552	92.664
49	Negex	MLP + LR	LinearSVC	92.896	91.395	90.076	90.691
50	Negex	MLP	LinearSVC	92.858	90.924	90.565	90.738
51	Negex		MLP	92.813	90.588	90.623	90.599
52	Negex	MLP + LinearSVC	LogReg	92.695	90.823	90.378	90.595
53	Negex	MLP + LinearSVC + LR	Stacking	92.695	90.698	90.497	90.596
54	Synonym	MLP + LinearSVC + LR	Voting	92.548	92.335	92.213	92.266
55	TextBlob	MLP + LinearSVC + LR	Voting	92.462	90.794	90.100	90.431

Table A1. *Cont.*

No.	Negation Handling	Base Model	Final Prediction	Control			
				Accuracy	Precision	Recall	F1-Score
56	TextBlob	MLP + GB	LogReg	92.411	90.772	89.935	90.333
57	Synonym	MLP + LinearSVC	LogReg	92.283	91.983	91.978	91.969
58	TextBlob	MLP + LinearSVC	Stacking	92.226	90.141	90.053	90.094
59	TextBlob	MLP + LinearSVC	Voting	92.021	90.400	89.368	89.853
60	TextBlob	MLP + LR	LinearSVC	91.973	89.853	89.809	89.823
61	TextBlob		MLP	91.865	89.859	89.520	89.683
62	TextBlob	MLP	LinearSVC	91.855	89.681	89.585	89.623
63	TextBlob	MLP + GB	LinearSVC	91.846	89.769	89.287	89.519
64	TextBlob	MLP + LinearSVC + LR	stacking	91.737	89.590	89.521	89.552
65	TextBlob	MLP + LinearSVC	LogReg	91.663	89.346	89.313	89.321
66	Synonym-only	LinearSVC	MLP	91.398	91.300	91.148	91.148
67	Synonym-only	LinearSVC + LR	MLP	91.386	91.257	91.121	91.132
68	TextBlob	LinearSVC + GB	MLP	91.363	89.454	87.568	88.405
69	TextBlob		LinearSVC	91.334	89.648	87.494	88.432
70	TextBlob	LinearSVC	MLP	91.242	89.524	87.403	88.332
71	TextBlob	LinearSVC + LR	MLP	91.210	89.568	87.547	88.426
72	Synonym-only	LinearSVC + GB	MLP	91.136	91.022	90.955	90.916
73	Synonym-only	MLP + LinearSVC	Voting	90.108	89.849	89.880	89.849
74	Synonym-only	MLP + GB	LogReg	90.095	89.863	89.871	89.846
75	Synonym-only	MLP + LinearSVC + LR	Stacking	90.079	89.795	89.916	89.847
76	Negex		LR	90.044	90.192	84.804	86.844
77	Synonym-only	MLP + GB	LinearSVC	89.971	89.827	89.681	89.735
78	Synonym-only	MLP	LinearSVC	89.907	89.992	89.398	89.631
79	Synonym-only	MLP + LinearSVC + LR	Voting	89.891	89.553	89.778	89.644
80	Synonym		LR	89.805	89.915	89.163	89.410
81	Synonym-only		LR	89.805	89.915	89.163	89.410
82	Synonym-only	MLP + LinearSVC	Stacking	89.785	89.696	89.341	89.475
83	Zero-Shot		RoBERTa	89.625	77.560	73.758	75.611
84	Synonym-only	MLP + LR	LinearSVC	89.616	89.464	89.235	89.318
85	Synonym-only	MLP + LinearSVC	LogReg	89.501	89.421	89.246	89.314
86	TextBlob		LR	88.814	87.410	83.737	85.164
87	Zero-Shot		BERT	87.990	76.766	70.637	73.574
88	Negex		RF	87.042	89.238	79.446	82.321
89	Hybrid		RF	85.713	89.484	77.548	80.589
90	TextBlob		RF	85.026	87.209	77.379	79.986
91	Zero-Shot		LinearSVC	84.420	79.002	62.796	66.733
92	Zero-Shot		LR	84.380	85.781	59.564	63.453
93	Zero-Shot	LinearSVC	MLP	83.935	81.717	62.545	66.883
94	Zero-Shot	LinearSVC + LR	MLP	83.740	83.251	62.532	66.833
95	Zero-Shot	LinearSVC + GB	MLP	83.575	78.110	62.111	65.947
96	Zero-Shot		MLP	83.490	70.045	64.737	66.810
97	Zero-Shot	MLP + GB	LinearSVC	83.340	74.096	64.973	64.531
98	Zero-Shot	MLP + LinearSVC	Stacking	83.100	79.273	64.675	66.550
99	Zero-Shot	MLP	LinearSVC	83.060	74.077	64.929	64.499
100	Zero-Shot	MLP + LR	LinearSVC	83.050	73.925	65.006	64.458
101	Zero-Shot	MLP + GB	LogReg	83.015	79.117	64.188	66.098
102	Zero-Shot	MLP + LinearSVC	LogReg	82.935	78.072	64.663	66.117
103	Zero-Shot	MLP + LinearSVC + LR	Stacking	82.935	76.825	64.715	65.685
104	Zero-Shot	MLP + LinearSVC	Voting	82.655	73.727	64.630	64.174
105	Zero-Shot	MLP + LinearSVC + LR	Voting	82.335	76.107	64.446	65.226
106	Zero-Shot		RF	82.150	84.547	58.029	63.278
107	Synonym		RF	80.465	81.069	79.356	79.730
108	Synonym-only		RF	80.465	81.069	79.356	79.730

Table A1. *Cont.*

No.	Negation Handling	Base Model	Final Prediction	Control			
				Accuracy	Precision	Recall	F1-Score
109	Hybrid		GB	77.019	84.437	67.196	70.115
110	Zero-Shot		GB	76.055	71.147	51.552	55.531
111	Negex		GB	74.729	82.395	63.378	65.630
112	TextBlob		GB	73.307	80.188	62.206	63.812
113	Synonym		GB	69.059	75.168	66.867	67.230
114	Synonym-only		GB	69.059	75.168	66.867	67.230

Appendix A.2. SMOTE Model Performance

Table A2 represents model performance applying SMOTE imbalance technique in this experiment.

Table A2. Model performance with SMOTE.

No.	Negation Handling	Base Model	Final Prediction	Control			
				Accuracy	Precision	Recall	F1-Score
1	Hybrid		BERT	98.582	98.196	98.189	98.193
2	Hybrid		RoBERTa	98.486	98.060	98.098	98.079
3	Hybrid		MLP	98.098	98.101	98.097	98.098
4	Hybrid	MLP + GB	LogReg	98.073	98.073	98.074	98.072
5	Hybrid	MLP	LinearSVC	98.067	98.072	98.066	98.068
6	Hybrid	MLP + LinearSVC	Voting	98.049	98.053	98.049	98.048
7	Hybrid	MLP + LinearSVC	Stacking	98.022	98.030	98.023	98.022
8	Hybrid	MLP + LinearSVC + LR	Stacking	97.992	97.995	97.992	97.991
9	Hybrid	MLP + LR	LinearSVC	97.943	97.950	97.943	97.943
10	Hybrid	MLP + GB	LinearSVC	97.932	97.943	97.931	97.932
11	Hybrid	LinearSVC + GB	MLP	97.905	97.909	97.907	97.903
12	Hybrid	LinearSVC	MLP	97.905	97.909	97.907	97.903
13	Hybrid		LinearSVC	97.903	97.907	97.905	97.901
14	Hybrid	LinearSVC + LR	MLP	97.903	97.907	97.903	97.900
15	Hybrid	MLP + LinearSVC + LR	Voting	97.901	97.910	97.901	97.901
16	Hybrid	MLP + LinearSVC	LogReg	97.866	97.874	97.867	97.864
17	Synonym		BERT	97.796	97.737	97.713	97.725
18	TextBlob		BERT	97.438	97.092	96.652	96.871
19	Synonym		RoBERTa	97.355	97.209	97.312	97.261
20	TextBlob		RoBERTa	97.320	96.913	96.442	96.677
21	Negex		BERT	96.531	95.799	95.026	95.411
22	Negex		RoBERTa	96.320	95.291	94.836	95.063
23	Negex	MLP + LinearSVC	LogReg	95.720	95.724	95.725	95.716
24	Negex	MLP + LR	LinearSVC	95.688	95.689	95.692	95.684
25	Negex	MLP + LinearSVC + LR	Voting	95.671	95.677	95.676	95.665
26	Negex		LinearSVC	95.605	95.646	95.615	95.593
27	Negex	LinearSVC + GB	MLP	95.605	95.646	95.615	95.593
28	Negex	LinearSVC	MLP	95.605	95.646	95.615	95.593
29	Negex	LinearSVC + LR	MLP	95.605	95.646	95.615	95.593
30	Negex	MLP + LinearSVC + LR	Stacking	95.588	95.594	95.592	95.583
31	TextBlob	MLP + GB	LogReg	95.477	95.508	95.478	95.467
32	TextBlob	MLP + LinearSVC + LR	Voting	95.460	95.477	95.462	95.450
33	Negex	MLP + LinearSVC	Voting	95.454	95.465	95.456	95.449
34	Negex	MLP + GB	LogReg	95.452	95.464	95.455	95.446
35	Negex	MLP + GB	LinearSVC	95.450	95.461	95.450	95.442

Table A2. *Cont.*

No.	Negation Handling	Base Model	Final Prediction	Control			
				Accuracy	Precision	Recall	F1-Score
36	TextBlob	MLP + LR	LinearSVC	95.423	95.425	95.423	95.419
37	Negex	MLP + LinearSVC	Stacking	95.371	95.396	95.377	95.363
38	Negex		MLP	95.354	95.381	95.357	95.349
39	Negex	MLP	LinearSVC	95.346	95.352	95.349	95.341
40	TextBlob	MLP + GB	LinearSVC	95.346	95.362	95.347	95.341
41	TextBlob	MLP	LinearSVC	95.329	95.376	95.330	95.318
42	TextBlob	MLP + LinearSVC + LR	stacking	95.325	95.328	95.325	95.319
43	TextBlob	MLP + LinearSVC	Stacking	95.279	95.289	95.280	95.274
44	Hybrid		LR	95.224	95.313	95.237	95.214
45	TextBlob		MLP	95.164	95.189	95.166	95.155
46	TextBlob	MLP + LinearSVC	Voting	95.149	95.176	95.147	95.144
47	TextBlob	MLP + LinearSVC	LogReg	95.126	95.126	95.125	95.119
48	Synonym-only		BERT	94.822	94.678	94.705	94.692
49	Synonym-only		RoBERTa	94.576	94.429	94.463	94.446
50	TextBlob	LinearSVC + GB	MLP	94.134	94.183	94.140	94.118
51	TextBlob	LinearSVC	MLP	94.134	94.183	94.140	94.118
52	TextBlob	LinearSVC + LR	MLP	94.134	94.179	94.134	94.113
53	TextBlob		LinearSVC	94.134	94.183	94.140	94.118
54	Synonym	LinearSVC + GB	MLP	93.434	93.450	93.457	93.430
55	Synonym	LinearSVC	MLP	93.434	93.450	93.457	93.430
56	Synonym	LinearSVC + LR	MLP	93.434	93.450	93.457	93.430
57	Synonym		LinearSVC	93.434	93.450	93.457	93.430
58	Zero-Shot	MLP + LR	LinearSVC	92.959	92.956	92.947	92.924
59	Synonym	MLP + GB	LogReg	92.764	92.769	92.779	92.768
60	Synonym	MLP + LinearSVC	Voting	92.761	92.775	92.779	92.762
61	Zero-Shot	MLP	LinearSVC	92.757	92.782	92.745	92.720
62	Synonym-only		MLP	92.757	92.778	92.774	92.756
63	Synonym	MLP + LinearSVC + LR	Voting	92.621	92.623	92.637	92.624
64	Zero-Shot	MLP + LinearSVC	LogReg	92.612	92.673	92.600	92.569
65	Synonym	MLP + LinearSVC + LR	Stacking	92.605	92.613	92.620	92.608
66	Zero-Shot	MLP + LinearSVC	Stacking	92.583	92.651	92.570	92.543
67	Synonym	MLP + GB	LinearSVC	92.573	92.591	92.584	92.582
68	Zero-Shot	MLP + LinearSVC	Voting	92.572	92.649	92.573	92.543
69	Synonym		MLP	92.571	92.574	92.571	92.565
70	Zero-Shot	MLP + LinearSVC + LR	Voting	92.557	92.581	92.545	92.515
71	Zero-Shot		MLP	92.509	92.644	92.497	92.460
72	Synonym	MLP + LR	LinearSVC	92.508	92.511	92.530	92.505
73	Zero-Shot	MLP + LinearSVC + LR	Stacking	92.503	92.507	92.490	92.468
74	Synonym	MLP	LinearSVC	92.498	92.516	92.519	92.495
75	Zero-Shot	MLP + GB	LogReg	92.478	92.579	92.465	92.431
76	Zero-Shot	MLP + GB	LinearSVC	92.473	92.506	92.461	92.431
77	Synonym	MLP + LinearSVC	Stacking	92.373	92.386	92.393	92.374
78	Synonym	MLP + LinearSVC	LogReg	92.346	92.353	92.362	92.351
79	Negex		LR	92.224	92.412	92.242	92.199
80	Synonym-only	LinearSVC + GB	MLP	91.837	91.964	91.855	91.808
81	Synonym-only	LinearSVC	MLP	91.837	91.964	91.855	91.808
82	Synonym-only	LinearSVC + LR	MLP	91.837	91.964	91.855	91.808
83	Synonym-only		LinearSVC	91.837	91.964	91.855	91.808
84	Synonym-only	MLP + LinearSVC	Voting	91.644	91.671	91.645	91.619
85	Synonym-only	MLP + LinearSVC	Stacking	91.642	91.655	91.654	91.625
86	Synonym-only	MLP + LinearSVC	LogReg	91.617	91.620	91.627	91.604
87	Zero-Shot	LinearSVC + LR	MLP	91.601	91.579	91.586	91.544
88	Zero-Shot	LinearSVC	MLP	91.593	91.566	91.578	91.533

Table A2. *Cont.*

No.	Negation Handling	Base Model	Final Prediction	Control			
				Accuracy	Precision	Recall	F1-Score
89	Zero-Shot	LinearSVC + GB	MLP	91.584	91.556	91.570	91.524
90	Zero-Shot		LinearSVC	91.584	91.556	91.570	91.524
91	Synonym-only	MLP + LinearSVC + LR	Voting	91.480	91.496	91.492	91.463
92	Zero-Shot		RF	91.450	91.819	91.430	91.403
93	Synonym-only	MLP	LinearSVC	91.323	91.328	91.334	91.308
94	Synonym-only	MLP + GB	LinearSVC	91.002	91.012	91.013	90.991
95	Synonym-only	MLP + LinearSVC + LR	Stacking	90.927	90.923	90.935	90.923
96	Synonym-only	MLP + LR	LinearSVC	90.918	90.950	90.931	90.909
97	Synonym-only	MLP + GB	LogReg	90.813	90.828	90.825	90.796
98	TextBlob		LR	90.729	90.886	90.742	90.691
99	Hybrid		RF	90.378	91.089	90.401	90.395
100	Zero-Shot		LR	89.939	89.802	89.919	89.843
101	Synonym		LR	89.814	89.960	89.849	89.799
102	Negex		RF	89.523	90.382	89.551	89.510
103	Zero-Shot		RoBERTa	89.210	79.138	74.186	76.582
104	TextBlob		RF	88.493	89.271	88.510	88.440
105	Zero-Shot		BERT	88.105	74.060	69.882	71.911
106	Synonym-only		LR	88.066	88.441	88.089	88.026
107	Synonym-only		RF	82.190	82.650	82.216	82.146
108	Synonym		RF	81.451	81.890	81.491	81.430
109	Hybrid		GB	79.936	83.601	80.015	79.865
110	Negex		GB	76.454	80.663	76.523	76.218
111	Zero-Shot		GB	76.422	77.024	76.408	76.360
112	TextBlob		GB	75.219	78.904	75.267	75.008
113	Synonym		GB	70.059	72.626	70.176	69.854
114	Synonym-only		GB	69.121	75.018	69.193	68.777

Appendix B. Software & Reproducibility Artifacts

Software versions. Python 3.12.12; PyTorch 2.9; Transformers 4.57.3; scikit-learn 1.6.1; imbalanced-learn 0.14.1; spaCy 3.8.11 (en_core_web_sm 3.8); TextBlob 0.19.0; NumPy 2.0.2; pandas 2.2.2.

Artifacts. Upon acceptance, we plan to release preprocessing scripts, training configs, fold splits, seed files, and a notebook that regenerates Table X and Figure X from saved per-fold outputs.

Appendix B.1. Ablation Study

What we ablate. Starting from the full Hybrid method (rules for explicit/implicit/double negation + dependency-scope modeling + TextBlob clause priors + MLP clause-weighting), we create progressively simplified variants:

- Hybrid (full)—all components enabled.
  - MLP weights—remove the MLP; use fixed rule weights for clauses;
  - Implicit & double—keep explicit negation and dependency scopes, but drop implicit (e.g., “hardly” and “barely”) and double-negation handling;
  - Dependency scopes—apply only sentence-local rules (no dependency-based scope), with TextBlob priors.
- TextBlob baseline—polarity from TextBlob only (no explicit negation rules).

Protocol. We used the same stratified 5-fold CV, preprocessing, and SMOTE placement (training folds only). Metrics are macro-averaged over folds. We tested Hybrid (full) versus

each ablation using paired Wilcoxon signed rank (one-sided, $\alpha = 0.01$) and adjust p-values with Holm–Bonferroni.

Ablation on the Hybrid negation model (BERT backbone, SMOTE in training folds only). Bars show mean macro-F1 across 5 folds; error bars are standard error. The full Hybrid method significantly outperforms every ablation (paired Wilcoxon, Holm–Bonferroni-adjusted $p < 0.01$).

Table A3. Ablation results (macro-averaged across 5 folds). Replace "…" with the means ± SE after reruns.

Variant	Acc. (Mean ± SE)	Prec. (Mean ± SE)	Rec. (Mean ± SE)	F1 (Mean ± SE)
Hybrid (full)	98.403 ± …	97.968 ± …	97.883 ± …	97.926 ± …
– MLP weights	…	…	…	…
– Implicit & double	…	…	…	…
– Dependency scopes	…	…	…	…
TextBlob baseline	97.438 ± …	97.092 ± …	96.652 ± …	96.871 ± …

Appendix B.2. Reproducibility & Implementation Details

Data & labels. We used a public Twitter corpus of 156,539 posts (26 September 2021–27 March 2022) with 3 classes (positive/neutral/negative). We removed retweets, expand contractions (e.g., aren't → are not) and stripped URLs, punctuation, numbers, and non-text artifacts. Class ratios are reported in Appendix A; original tweet IDs are retained for inspection.

Splits and leakage control. We performed stratified 5-fold cross-validation. All preprocessing, vectorization, model fitting, and SMOTE oversampling occurred inside each training fold only; validation/test folds were never oversampled. Transformer fine-tuning uses an internal 9:1 split from the training fold for early stopping.

Hybrid implementation. We parsed with spaCy to extract dependency scopes (main, relative, temporal, conditional, causal, and contrast). TextBlob provides clause-level polarity priors. Rules handle explicit, implicit, and double negation. A 2-layer MLP maps clause features (scope type, depth, length, and negation cues) to [0, 1] weights via a sigmoid; the sentence score is the weighted, sign-corrected aggregation of clause polarities. The MLP was trained per training fold with Adam, cross-entropy loss, and early stopping (patience = 3).

Backbones & baselines. We compared the following six negation strategies: Hybrid, antonym–synonym, antonym–synonym + rule, TextBlob, NegEx + spaCyTextBlob, and zero-shot. Classifiers include BERT, RoBERTa, LinearSVC, Logistic Regression, MLP, Random Forest, and gradient boosting. The best pipeline is Hybrid + BERT + SMOTE with 98.403 Acc/97.968 Prec/97.883 Rec/97.926 F1 (macro).

Hyperparameters. Tokenization/FT: bert-base-uncased/roberta-base, max_len = 128, batch = 32, lr = 2×10^{-5}, epochs = 3–5, early stopping on validation loss. MLP (clause weights): hidden = [64, 32], dropout = 0.2, lr = 1×10^{-3}, batch = 64, epochs = 10–20, early stopping. Classical models: scikit-learn defaults unless noted; C tuned over {0.5, 1, 2} for LinearSVC/LogReg via inner CV on the training fold. SMOTE: k_neighbors = 5 via imbalanced-learn; applied after vectorization within the training fold.

Statistics. We report macro accuracy, precision, recall, and F1. For inferential testing, we use paired Wilcoxon signed rank tests (one-sided, $\alpha = 0.01$) on per-fold scores to compare (i) SMOTE vs. No-SMOTE within a pipeline and (ii) Hybrid vs. each baseline. We adjusted family-wise error with Holm–Bonferroni and report W, z, and adjusted p-values.

Compute & seeds. Experiments were run on a single NVIDIA GPU (e.g., RTX 3090) or CPU. We fixed seeds for NumPy, PyTorch, and scikit-learn and set PYTHONHASHSEED = 0. For CUDA determinism, we disabled non-deterministic CuDNN kernels where applicable; remaining nondeterminism was negligible (<0.02 pp on macro-F1 across repeats).

References

1. McClain, C.; Vogels, E.A.; Perrin, A.; Sechopoulos, S.; Rainie, L. *The Internet and the Pandemic*; Pew Research Center: Washington, DC, USA, 2021.
2. Horrigan, J.B. *The Internet as a Resource for News and Information about Science*; Pew Research Center: Washington, DC, USA, 2006.
3. Diamantini, C.; Mircoli, A.; Potena, D. A Negation Handling Technique for Sentiment Analysis. In Proceedings of the International Conference on Collaboration Technologies and Systems (CTS), Orlando, FL, USA, 31 October–4 November 2016.
4. Wong, A.; Ho, S.; Lyness, D.; Olusanya, O.; Antonini, M.V. The use of social media and online communications in times of pandemic COVID-19. *J. Intensive Care Soc.* **2020**, *22*, 255–260. [CrossRef]
5. Dean, B. X (Twitter) Statistics: How Many People Use X? 30 January 2025. [Online]. Available online: https://backlinko.com/twitter-users (accessed on 2 March 2025).
6. Tao, D.; Yang, P.; Feng, H. Utilization of text mining as a big data analysis tool for food science and nutrition. *Compr. Rev. Food Sci. Food Saf.* **2020**, *19*, 875–894. [CrossRef]
7. Oyebode, O.; Ndulue, C.; Adib, A.; Mulchandani, D.; Suruliraj, B.; Orji, F.A.; Chambers, C.T.; Meier, S.; Orji, R. Health, Psychosocial, and Social Issues Emanating From the COVID-19 Pandemic Based on Social Media Comments: Text Mining and Thematic Analysis Approach. *J. Med. Internet Res.* **2021**, *9*, 22734. [CrossRef]
8. Hassan, A.; Mahmood, A. Deep Learning approach for sentiment analysis of short texts. In Proceedings of the 3rd International Conference on Control, Automation and Robotics (ICCAR), Nagoya, Japan, 22–24 April 2017.
9. Mahmud, Q.I.; Mohaimen, A.; Islam, M.S.; Jannat, M.-E. A support vector machine mixed with statistical reasoning approach to predict movie success by analyzing public sentiments. In Proceedings of the 20th International Conference of Computer and Information Technology (ICCIT), Dhaka, Bangladesh, 22–24 December 2017.
10. Reis, J.C.S.; Correia, A.; Murai, F.; Veloso, A.; Benevenuto, F. Supervised Learning for Fake News Detection. *IEEE Intell. Syst.* **2019**, *43*, 76–81. [CrossRef]
11. Qorib, M.; Kim, J. Fake Tweets Detection and Its Impacts on the 2020 U.S. Election Prediction. In Proceedings of the 2024 International Symposium on Networks, Computers and Communications (ISNCC), Washington, DC, USA, 22–25 October 2024.
12. Mahata, D.; Friedrichs, J.; Shah, R.R.; Jiang, J. Detecting personal intake of medicine from Twitter. *IEEE Intell. Syst.* **2018**, *33*, 87–95. [CrossRef]
13. Branz, L.; Brockmann, P. Sentiment Analysis of Twitter Data: Towards Filtering, Analyzing and Interpreting Social Network Data. In Proceedings of the DEBS '18: The 12th ACM International Conference on Distributed and Event-based Systems, Hamilton, New Zealand, 25–29 June 2018.
14. JAlbornoz, C.-D.; Vidal, J.R.; Plaza, L. Feature engineering for sentiment analysis in e-health forums. *PLoS ONE* **2018**, *13*, e0207996.
15. Cambria, E. Affective Computing and Sentiment Analysis. *IEEE Intell. Syst.* **2016**, *31*, 102–107. [CrossRef]
16. Qorib, M.; Oladunni, T.; Denis, M.; Ososanya, E.; Cotae, P. COVID-19 vaccine hesitancy: Text mining, sentiment analysis and machine learning on COVID-19 vaccination Twitter dataset. *Expert Syst. Appl.* **2023**, *212*, 118715. [CrossRef]
17. Singh, T.; Kumari, M. Role of Text Pre-Processing in Twitter Sentiment Analysis. *Procedia Comput. Sci.* **2016**, *89*, 549–554. [CrossRef]
18. Palomino, M.A.; Aider, F. Evaluating the Effectiveness of Text Pre-Processing in Sentiment Analysis. *Appl. Sci.* **2022**, *12*, 8765. [CrossRef]
19. Firza, N.; Bakiu, A.; Monaco, A. Machine Learning for Quality Diagnostics: Insights into Consumer Electronics Evaluation. *Electronics* **2025**, *14*, 939. [CrossRef]
20. Krouska, A.; Troussas, C.; Virvou, M. The effect of preprocessing techniques on Twitter Sentiment Analysis. In Proceedings of the 2016 7th International Conference on Information, Intelligence, Systems & Applications (IISA), Chalkidiki, Greece, 13–15 July 2016.
21. Alam, S.; Yao, N. The impact of preprocessing steps on the accuracy of machine learning algorithms in sentiment analysis. *Comput. Math. Organ. Theory* **2019**, *25*, 319–335. [CrossRef]
22. Prakriti. Removing Stop Words Using NLTK. 26 March 2024. [Online]. Available online: https://www.naukri.com/code360/library/removing-stop-words-using-nltk (accessed on 2 March 2025).
23. Okpala, I.; Rodriguez, G.R.; Tapia, A.; Halse, S.; Kropczynski, J. A Semantic Approach to Negation Detection and Word Disambiguation with Natural Language Processing. In Proceedings of the 2022 6th International Conference on Natural Language Processing and Information Retrieval, Bangkok, Thailand, 17–19 December 2023.
24. Gupta, I.; Joshi, N. Enhanced Twitter Sentiment Analysis Using Hybrid Approach and by Accounting Local Contextual Semantic. *J. Intell. Syst.* **2019**, *29*, 1611–1625. [CrossRef]

25. Mukherjee, P.; Badr, Y.; Doppalapudi, S.; Srinivasan, S.M.; Sangwan, R.S.; Sharma, R. Effect of Negation in Sentences on Sentiment Analysis and Polarity Detection. *Procedia Comput. Sci.* **2021**, *185*, 370–379. [CrossRef]
26. Gupta, I.; Joshi, N. Feature-Based Twitter Sentiment Analysis With Improved Negation Handling. *IEEE Trans. Comput. Soc. Syst.* **2021**, *8*, 917–927. [CrossRef]
27. Ferdinand, W.; Girsang, A.S. Negation Handling on XLNet Using Dependency Parser for Sentiment Analysis. In Proceedings of the 2024 International Conference on Intelligent Cybernetics Technology & Applications (ICICyTA), Bali, Indonesia, 17–19 December 2024.
28. Russo, I.; Caselli, T.; Strapparava, C. SemEval-2015 Task 9: CLIPEval Implicit Polarity of Events. In Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015), Denver, CO, USA, 4–5 June 2015.
29. Singh, P.K.; Paul, S. Deep Learning Approach for Negation Handling in Sentiment Analysis. *IEEE Access* **2021**, *9*, 102579–102592. [CrossRef]
30. Hyscaler. Sentiment Analysis: A Step-by-Step Guide. 20 August 2024. [Online]. Available online: https://hyscaler.com/insights/sentiment-analysis-guide/ (accessed on 3 December 2025).
31. Song, C.; Zhang, Y.; Gao, H.; Yao, B.; Zhang, P. Large Language Models for Subjective Language Understanding: A Survey. *arXiv* **2025**, arXiv:2508.07959. [CrossRef]
32. Hii, D. Using Meaning Specificity to Aid Negation Handling in Sentiment Analysis. Master's Thesis, University of California at Irvine, Irvine, CA, USA, 2019.
33. Naldi, M.; Petroni, S. A Testset-Based Method to Analyse the Negation-Detection Performance of Lexicon-Based Sentiment Analysis Tools. *Computers* **2023**, *12*, 18. [CrossRef]
34. Farooq, U.; Mansoor, H.; Nongaillard, A.; Ouzrout, Y.; Qadir, M.A. Negation Handling in Sentiment Analysis at Sentence Level. *J. Comput.* **2017**, *12*, 470–478. [CrossRef]
35. Díaz, N.P.C. Negation and Speculation Detection for Improving Information Retrieval Effectiveness. In Proceedings of the Fifth BCS-IRSG Symposium on Future Directions in Information Access (FDIA 2013) (FDIA), 3 September 2013.
36. Kamal, L.H.; McKee, G.T.; Othman, N.A. Naïve Bayes with Negation Handling for Sentiment Analysis of Twitter Data. In Proceedings of the 2022 9th Intl. Conference on Soft Computing & Machine Intelligence, Toronto, ON, Canada, 26–27 November 2022.
37. Ghag, K.V.; Shah, K. Negation Handling for Sentiment Classification. In Proceedings of the 2016 International Conference on Computing Communication Control and Automation (ICCUBEA), Pune, India, 12–13 August 2016.
38. GTejada, N.C.; Scholtes, J.C.; Spanakis, G. *A Study of BERT's Processing of Negations to Determine Sentiment*; Maastricht University: Maastricht, The Netherland, 2021.
39. Lal, U.; Kamath, P. Effective Negation Handling Approach for Sentiment Classification using synsets in the WordNet lexical database. In Proceedings of the 2022 First International Conference on Electrical, Electronics, Information and Communication Technologies (ICEEICT), Trichy, India, 16–18 February 2022.
40. Abadi, V.N.M.; Ghasemian, F. Enhancing Persian text summarization through a three-phase fine-tuning and reinforcement learning approach with the mT5 transformer model. *Sci. Rep.* **2025**, *15*, 80. [CrossRef]
41. Sun, Y.; Zhang, J.; Yu, Z.; Zhang, Y.; Liu, Z. Bidirectional Long Short-term Neural Network Based on the Attention Mechanism of the Residual Neural Network (ResNet–BiLSTM–Attention) Predicts Porosity through Well Logging Parameters. *ACS Omega* **2023**, *8*, 24083–24092. [CrossRef]
42. Qorib, M.; Oladunni, T.; Denis, M.; Osasanya, E.; Cotae, P. COVID-19 Vaccine Hesitancy: A Global Public Health and Risk Modelling Framework Using an Environmental Deep Neural Network, Sentiment Classification with Text Mining and Emotional Reactions from COVID-19 Vaccination Tweets. *Int. J. Environ. Res. Public Health* **2023**, *20*, 5803. [CrossRef]
43. Qorib, M. *COVID-19 Public Tweets*; Researchgate: Washington, DC, USA, 2022; Available online: https://www.researchgate.net/publication/364110620_Covid19_Public_Tweets?channel=doi&linkId=633a5ad876e39959d6903819&showFulltext=true (accessed on 2 March 2025).
44. HaCohen-Kerner, Y.; Miller, D.; Yigal, Y. The influence of preprocessing on text classification using a bag-of-words representation. *PLoS ONE* **2020**, *15*, e0232525. [CrossRef] [PubMed]
45. Uysal, A.K.; Gunal, S. The impact of preprocessing on text classification. *Inf. Process. Manag.* **2014**, *50*, 104–112. [CrossRef]
46. Vijayarani, S.; Ilamathi, J.; Nithya, N.S. Preprocessing Techniques for Text Mining—An Overview. *Int. J. Comput. Sci. Commun. Netw.* **2015**, *5*, 7–16.
47. Chapman, W.W.; Bridewell, W.; Hanbury, P.; Cooper, G.F.; Buchanan, B.G. A simple algorithm for identifying negated findings and diseases in discharge summaries. *J. Biomed. Inform.* **2001**, *34*, 301–310. [CrossRef] [PubMed]
48. Romera-Paredes, B.; Torr, P. An embarrassingly simple approach to zero-shot learning. In Proceedings of the 32nd International Conference on Machine Learning, Lille, France, 6–11 July 2015.
49. Lampert, C.H.; Nickisch, H.; Harmeling, S. Learning to detect unseen object classes by between-class attribute transfer. In Proceedings of the 2009 IEEE Conference on Computer Vision and Pattern Recognition, Miami, FL, USA, 20–25 June 2009.
50. Larochelle, H.; Erhan, D.; Bengio, Y. Zero-data learning of new tasks. *AAAI* **2008**, *1*, 3.

51. Nam, J.; Mencía, E.L.; Fürnkranz, J. All-in Text: Learning Document, Label, and Word Representations Jointly. In Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016.
52. Pushp, P.K.; Srivastava, M.M. Train once, test anywhere: Zero-shot learning for text classification. In Proceedings of the ICLR 2018, Vancouver, BC, Canada, 30 April–3 May 2017.
53. Ma, Y.; Cambria, E.; Gao, S. Label embedding for zero-shot fine-grained named entity typing. In Proceedings of the COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers, Osaka, Japan, 11–16 December 2016.
54. Levy, O.; Seo, M.; Zettlemoyer, L. Zero-Shot Relation Extraction via Reading Comprehension. *arXiv* **2017**, arXiv:1706.04115. [CrossRef]
55. Xiang, M.; Grove, J.; Giannakidou, A. Semantic and pragmatic processes in the comprehension of negation: An event related potential study of negative polarity sensitivity. *J. Neurolinguist.* **2016**, *38*, 71–88. [CrossRef]
56. Intellectus360. "Intelectus Consulting," Intelectus Consulting, 2025. [Online]. Available online: https://www.statisticssolutions.com/free-resources/directory-of-statistical-analyses/how-to-conduct-the-wilcox-sign-test/ (accessed on 23 October 2025).
57. Han, Y.; Joe, I. Enhancing Machine Learning Models Through PCA, SMOTE-ENN, and Stochastic Weighted Averaging. *Appl. Sci.* **2024**, *14*, 9772. [CrossRef]
58. Limanto, S.; Buliali, J.L.; Saikhu, A. GLoW SMOTE-D: Oversampling Technique to Improve Prediction Model Performance of Students Failure in Courses. *IEEE Access* **2024**, *12*, 8889–8901. [CrossRef]
59. Meng, D.; Li, Y. An imbalanced learning method by combining SMOTE with Center Offset Factor. *Appl. Soft Comput.* **2022**, *120*, 108618. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.