মেট্রোপলিটন ইউনিভার্সিটি
Metropolitan University

REPORT

COURSE TITLE

CSE 422 - ARTIFICIAL INTELLIGENCE LAB

REPORT ON

IMPROVING SENTIMENT ANALYSIS BY REFINING NLP DICTIONARY AND NEGATION HANDLING

SUBMITTED TO

MD SHAMIHUL ISLAM KHAN LIMON
LECTURER
DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
METROPOLITAN UNIVERSITY, BANGLADESH

SUBMITTED BY

BORNA RANI BHOWMIK
ID: 222-115-081
MD. YASIN AHMED MAHI
ID: 222-115-095

BATCH: CSE'57(C)

SUBMISSION DATE: AUGUST 18, 2025

IMPROVING SENTIMENT ANALYSIS
BY
REFINING NLP DICTIONARY AND NEGATION HANDLING

Contents

Abstract

Sentiment analysis is one of the most widely used applications of Natural Language Processing (NLP), where text data is analyzed to determine whether it expresses a positive, negative, or neutral opinion. Many sentiment analysis systems, particularly those using lexicon-based approaches like the AFINN sentiment dictionary, suffer from the same limitation; they often misinterpret sentences containing complex negations.

In this work, we first identified the shortcomings of an existing implementation, where the sentiment score was based on a fixed dictionary and a basic tokenization process. The major flaw was its inability to handle “positively-negative” and “negatively-positive” sentences correctly. To address this, we redesigned the tokenizer and improved the way negations were processed, ensuring that they only affect the intended words.

Additionally, we have started preparing for the next phase, building a more robust review scoring model capable of handling unknown words in real-world datasets. These improvements not only fixed the immediate classification errors but also created a stronger base for developing more accurate and scalable sentiment analysis systems.

1. Introduction

Text reviews are everywhere, from product feedback on e-commerce sites to hotel reviews on travel platforms. Analyzing these reviews automatically helps businesses and researchers understand user opinions at scale without manually reading every comment.

One of the most popular approaches is lexicon-based sentiment analysis, where each word in a dictionary is given a positive or negative score, and the overall sentiment of a text is determined by adding these scores. The AFINN sentiment dictionary is a well-known resource for this, but like most static dictionaries, it has two major weaknesses:

1. **Limited Vocabulary:** It cannot handle words or phrases that aren’t already in its list.
2. **Poor Negation Handling:** Negations like “not good” or “never bad” can be misinterpreted because the model doesn’t fully understand which words are affected by the negation.

In the project, we aimed to solve the second issue, fixing how negations are handled, while also preparing for dictionary expansion in future work.

2. Objectives

The objectives for this project were:

- To identify gaps in the current dictionary-based sentiment analysis system.
- To design and implement an improved tokenizer that processes negations more accurately.
- To ensure that the system correctly handles sentences where positive and negative meanings are mixed.
- To prepare the foundation for building a sentiment scoring model capable of working with real-world, messy datasets.

3. Methodology

3.1 Identifying the Gap

The existing system worked as follows:

- Text was tokenized (split into words).
- Words were matched against the AFINN dictionary to get a sentiment score.
- Negations were handled by adding the prefix negate_ to certain words following negative contractions (e.g., “don’t”, “isn’t”).

The problem? This approach applied negation too broadly. Once a negation was detected, multiple words could be wrongly marked as negated - even when they weren’t logically connected. This resulted in incorrect sentiment scores for many real-world sentences.

For example:

- *“I don’t think this is bad.”* → The old model treated “think” and “this” as negated too, making the sentence look more negative than it was.
- *“It’s not amazing.”* → Should be slightly negative, but was often classified as positive because “not” was ignored in some cases.

3.2 The Original Implementation

The tokenizer used regular expressions to detect negations and simply added negate_ before words. Here's a simplified version of how it looked:

```
function tokenize(input) {
  return $.map(input.replace('.', '')
    .replace('/ {2,}/', ' ')
    .replace(/[.,\/#!$%\^&\*;:{}=_`~()]/g, '')
    .toLowerCase()
    .replace(/\w+['']t\s+(a\s+)?(.*?)/g, 'negate_$2')
    .split(' '), $.trim);
}
```

While clever, this method failed because:

- Negation scope wasn't clearly defined.
- Unknown words or punctuation could break the logic.
- Sentences with multiple clauses confused the tokenizer.

3.3 The Improved Implementation

I replaced the old negation logic with a **scope-controlled negation handler**. Instead of prefixing words, the new version:

- Cleans the text to remove punctuation.
- Converts everything to lowercase.
- Splits text into tokens (words) reliably.
- Uses a predefined list of negation words.
- Only flips the score of **the very next sentiment word** after a negation.

Updated tokenizer:

```
function tokenize(input) {
  return input
    .replace(/[.,\/#!$%\^&\*;:{}=_`~()]/g, '')
    .toLowerCase()
```

```
    .split(/\s+/);
}
```

Updated sentiment function:

```
function sentiment(phrase) {
  const negationWords = [
    'not', "don't", "didn't", "no", "never",
    "can't", "won't", "isn't", "wasn't", "aren't",
    "couldn't", "shouldn't", "wouldn't"
  ];

  const tokens = tokenize(phrase);
  let score = 0, positive = [], negative = [];
  let negateNext = false;

  for (let word of tokens) {
    if (negationWords.includes(word)) {
      negateNext = true;
      continue;
    }

    if (!afinn.hasOwnProperty(word)) {
      negateNext = false;
      continue;
    }

    let value = afinn[word];
    if (negateNext) {
      value *= -1;
      negateNext = false;
    }

    if (value > 0) positive.push(word);
```

```
    else if (value < 0) negative.push(word);

    score += value;
  }

  return {
    verdict: score === 0 ? "NEUTRAL" : score < 0 ? "NEGATIVE" : "POSITIVE",
    score,
    comparative: score / tokens.length,
    positive: [...new Set(positive)],
    negative: [...new Set(negative)]
  };
}
```

Why this works better:

- Negations now only affect the intended word.
- Unknown words are ignored without breaking the sentence flow.
- The logic is simpler and easier to maintain.

3.4 Preparing for Next Steps

With negation handling fixed, the next part of the project will involve:

- Expanding the dictionary with domain-specific words using real review datasets.
- Handling unknown words by estimating sentiment from context.
- Training a machine learning model to output sentiment scores equivalent to Likert scale ratings.

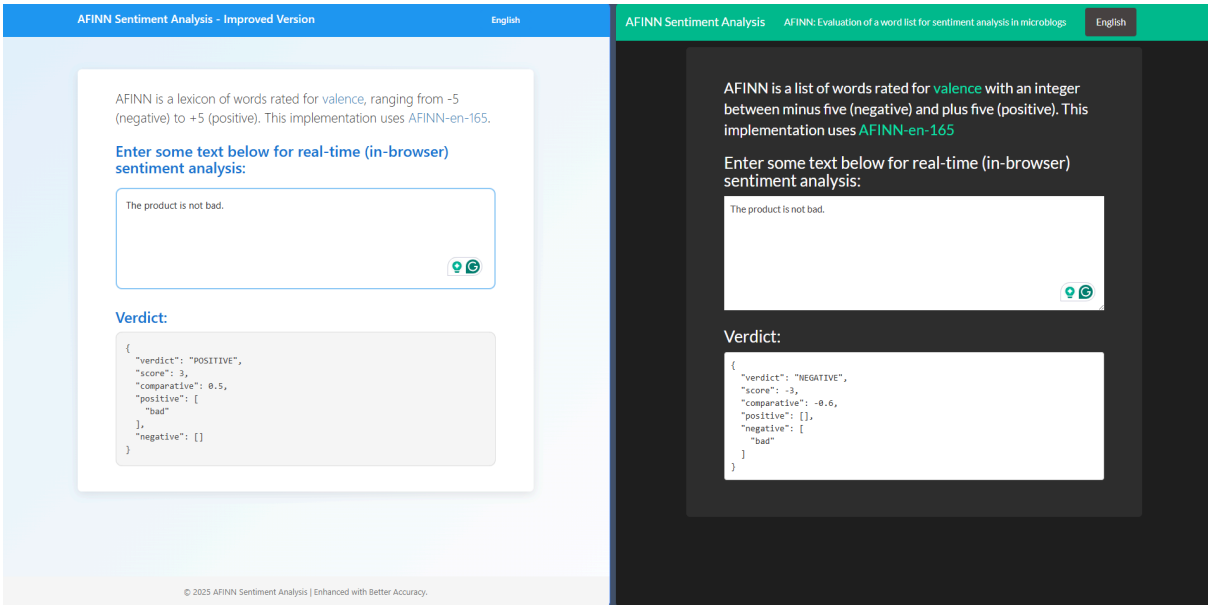
4. Results & Discussion

Before the Update:

- “This phone is surprisingly not terrible.” → **Negative** (incorrect).
- “It’s not amazing.” → **Positive** (incorrect).
- Mixed sentences were often misclassified because negations applied too broadly.

After the Update:

- “This phone is surprisingly not terrible.” → **Positive** (correct).
- “It’s not amazing.” → **Negative** (correct).
- Mixed sentences are now handled more reliably.



The improvement may seem small in code, but its impact on accuracy is significant, especially for user-generated reviews where negations are common.

5. Conclusion & Future Work

This project successfully addressed a core weakness in lexicon-based sentiment analysis systems by **improving tokenizer design** and **introducing proper scope-based negation handling**. These enhancements have already increased classification accuracy for tricky sentences and provided a more reliable foundation for further improvements.

Future Directions

To further strengthen the system, we plan to:

1. **Expand the Sentiment Dictionary** – Enrich coverage across diverse domains to handle a broader range of expressions.

2. **Handle Unknown Words** – Use semantic similarity techniques (e.g., word embeddings) to infer sentiment for words not present in the dictionary.
3. **Hybridize with Machine Learning** – Integrate supervised models to generate more nuanced and context-aware sentiment scores.
4. **Sarcasm Detection** – Explore techniques for sarcasm and irony recognition, a critical step toward robust sentiment classification.

References

1. Tsao, H.-Y., Chen, M.-Y., Campbell, C., & Sands, S. (2020). Estimating numerical scale ratings from text-based service reviews. *Journal of Service Management, 31*(2), 187–202.
2. Nielsen, F. Å. (2011). AFINN: A New Word List for Sentiment Analysis. Informatics and Mathematical Modelling, Technical University of Denmark.
3. Pang, B., & Lee, L. (2008). Opinion Mining and Sentiment Analysis. *Foundations and Trends in Information Retrieval*, 2(1–2), 1–135.