Revista Facultad de Ingeniería, 32 (66), 2023, e16395

Revista Facultad de Ingeniería
Journal Homepage: https://revistas.uptc.edu.co/index.php/ingenieria

SentiFuzzy: A Twitter Sentiment Classifier Based on Fuzzy Logic

Jimena-Adriana Timaná-Peña[1]
Carlos-Alberto Cobos-Lozada[2]
Jason-Paul Anturi- Martínez[3]
José-Luis Paz-Realpe[4]

Received: July 30, 2023 **Accepted:** November 11, 2023 **Published:** December 1, 2023

Citation: J.-A. Timaná-Peña, C.-A. Cobos-Lozada, J.-P. Anturi- Martínez, J.-L. Paz-Realpe, "SentiFuzzy: A Twitter Sentiment Classifier Based on Fuzzy Logic," *Revista Facultad de Ingeniería*, vol. 32, no. 66, e16395, 2023. https://doi.org/10.19053/01211129.v32.n66.2023.16395

Abstract

In the sentiment classification process, the quality of the polarity varies depending on the characteristics or attributes possessed by the classifier and those of the tweet being analyzed; therefore, a sentiment classifier achieves its highest quality in scenarios in which its characteristics are similar to the characteristics of the tweet. This article presents SentiFuzzy, an algorithm that, based on the characterization of

[1] M. Sc. Universidad del Cauca (Popayán-Cauca, Colombia). gabrielvargas@unicauca.edu.co. ORCID: 0000-0001-9517-5732
[2] Ph.D. Universidad del Cauca (Popayán-Cauca, Colombia). ccobos@unicauca.edu.co. ORCID: 0000-0002-9977-2881
[3] Universidad del Cauca (Popayán-Cauca, Colombia). jpanturi@unicauca.edu.co
[4] Universidad del Cauca (Popayán-Cauca, Colombia). jlpaz@unicauca.edu.co

attributes of five sentiment classifiers recognized in the literature, implemented a series of inference rules and fuzzy sets, which allowed to define mathematical weights for each classifier; thus, to know which classifier should be selected according to the nature of the analyzed tweet. Additionally, these weights were optimized by the Hill-Climbing optimization algorithm, which yielded, in some scenarios, a higher polarity accuracy than that reported in the state of the art and, in other cases, a competitive polarity accuracy compared to the polarity reported by the compared classifiers.

Keywords: fuzzy logic; polarity; polarity classifiers; sentiment analysis; sentiment classifiers; twitter.

SentiFuzzy: Clasificador de sentimientos en Twitter basado en lógica difusa

Resumen

En el proceso de clasificación de sentimientos, la calidad de la polaridad varía en relación con las características o atributos que posee el clasificador y las del tuit que se analiza, por lo tanto, un clasificador de sentimiento logra su mayor calidad cuando se encuentra en escenarios en que sus características son similares a las características del tuit. En este artículo se presenta SentiFuzzy, un algoritmo que, a partir de la caracterización de atributos de cinco clasificadores de sentimientos reconocidos en la literatura, implementó una serie de reglas de inferencia y conjuntos difusos que permitió definir pesos matemáticos para cada clasificador y de esta manera saber qué clasificador debe ser seleccionado según la naturaleza del tuit analizado. Adicionalmente, dichos pesos se optimizaron a través del algoritmo Hill Climbing, lo que permitió obtener para algunos escenarios una exactitud de polaridad más alta que la reportada en el estado del arte y, en otros casos, una exactitud de polaridad competitiva frente a la polaridad reportada por los clasificadores comparados.

Palabras clave: análisis de sentimientos; clasificadores de polaridad; clasificadores de sentimientos; lógica difusa; polaridad; Twitter.

Jimena-Adriana Timaná-Peña; Carlos-Alberto Cobos-Lozada; Jason-Paul Anturi- Martínez; José-Luis Paz-Realpe

SentiFuzzy: classificador de sentimentos do Twitter baseado em lógica difusa

Resumo

No processo de classificação de sentimento, a qualidade da polaridade varia em relação às características ou atributos que o classificador possui e os do tweet que está sendo analisado, portanto, um classificador de sentimento atinge sua maior qualidade quando é encontrado em cenários em que suas características são semelhantes às características do tweet. Este artigo apresenta o SentiFuzzy, um algoritmo que, baseado na caracterização de atributos de cinco classificadores de sentimento reconhecidos na literatura, implementou uma série de regras de inferência e conjuntos fuzzy que permitiram definir pesos matemáticos para cada classificador e desta forma saber qual classificador deveria ser selecionado de acordo com a natureza do tweet analisado. Adicionalmente, estes pesos foram otimizados através do algoritmo Hill Climbing, o que permitiu obter para alguns cenários uma precisão de polaridade superior à reportada no estado da técnica e, noutros casos, uma precisão de polaridade competitiva face à polaridade reportada pelo comparador. classificadores.

Palavras-chave: análise de sentimento; classificadores de polaridade; classificadores de sentimento; lógica difusa; polaridade; Twitter.

I. INTRODUCTION

More than 500 million messages are posted daily on the microblogging site Twitter [1]. These are known as tweets and are limited by a fixed number of characters in each publication. By analyzing the posted messages, it is possible to identify a sentiment associated with a polarity that provides information about the sentimental state of the user when posting a message. Polarity allows classifying the message as positive, negative, or neutral [2]. A message is considered positive when it is possible to identify a positive or benevolent tendency or feeling in its words, emoticons, hashtags, etc.; neutral when it does not express any emotion or is in the balance between negative and positive feelings; and negative when it contains words that generally attack a subject, be it a person, brand, animal, or thing.

Sentiment analysis on this message type is a demanding and sometimes ineffective task mainly due to the employed syntax (use of elongations, contractions, emoticons) and the context and semantics expressed in the text [3]. The percentage of the F-measure (F1) of polarity classification in sentiment analysis, used as a measure of quality and comparison, only reaches a maximum of 81.02% for Twitter messages, according to the articles analyzed in the systematic review presented in 2019 [4]. Considering that this measure can be better, this research focused on designing an algorithm called SentiFuzzy that seeks to increase the quality of polarity assignment in sentiment analysis from the results delivered by five sentiment classifiers, three of them identified in [4], and two additional classifiers widely used by the scientific community [5-[6], thus defining a unified polarity. Because some sentiment classifiers obtain higher quality in polarity assignment than others depending on the characteristics of the tweets they analyze, in this work, a recognition module for characteristics of the analyzed tweet was designed, which allows identifying its current attributes (characteristics) and defining the most appropriate classifier for its classification. Thus, the proposed algorithm seeks to leverage each sentiment analyzer's advantages and moderate the disadvantages through a priority system. It seeks to give more weight to the classifier that best suits the characteristics of the tweet and less to the classifier with characteristics more different from the tweet, because it will probably yield inaccurate results. This system

seeks to obtain a consistent and accurate match concerning the characteristics of the tweet and those of the analyzers/classifiers. The design of this priority system is based on fuzzy logic techniques that use inference rules initially created manually according to the characteristics of each sentiment classifier, then generated, and optimized through the Hill Climbing optimization algorithm [7].

The priority system infers the weights of each classifier, which will be used together with the quality of the polarity assignment calculated by a proprietary statistical system designed to compute the final resulting polarity. The five classifiers and the proposed algorithm were evaluated with four datasets of tweets, where the SentiFuzzy algorithm managed to outperform the reported quality in several scenarios and obtained competitive results against several state-of-the-art classifiers.

II. METHODOLOGY

A. Overview of the SentiFuzzy Algorithm

The SentiFuzzy algorithm comprises four modules: Sentiment Classifier, Characteristics Identifier, Fuzzy Logic, and Statistic, as illustrated in Figure 1. Each module is explained below. To calculate the final polarity of a tweet, the polarity and quality assigned by each of the five selected classifiers (Sentiment Classifier Module) are initially defined. Then, from the Characteristics Identifier Module, the characteristics of the tweet are obtained and passed to the Fuzzy Logic Module, which delivers a list of weights that indicates the priority each classifier should have in the tweet. Finally, the Statistic Module defines the algorithm's polarity for the tweet.

1) Sentiment Classifier Module. This module receives a tweet (string) as input from a text file and a set of classifiers evaluates it, each delivering the polarity for it. The classifiers used in this module are presented in Table 1. Table 2 describes the attributes used to characterize them.

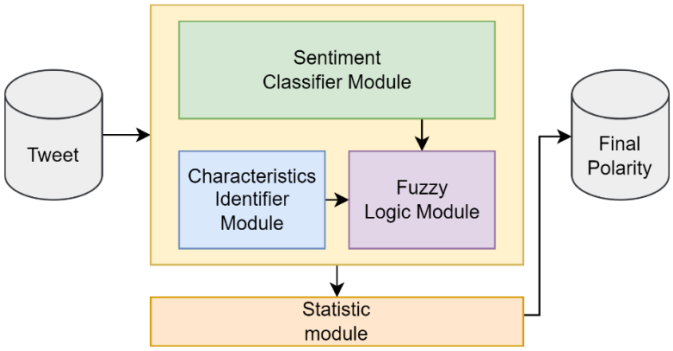

Fig. 1. SentiFuzzy Algorithm Modules.

Table 1. Classifiers used in the Sentiment Classification Module.

ID.	Classifier name
C1	Text Analytics API [8]
C2	Natural Language Understanding IBM API [9]
C3	Tweepy [10]
C4	SentiStrength [5]
C5	Vader Sentiment [6]

Table 2. Attribute Characterization.

ID	Attribute Name	Attribute Description	Value
A1	Message size	Defines the message size; here, the classifier has greater certainty in defining the polarity of a message.	• Short text (1 to 15 characters). • Medium text (16 to 145 characters). • Long text (146 to 280 characters).
A2	Quantity of emoticons	Defines the number of ASCII emoticons of a message with which the classifier has more certainty in defining the polarity of a message.	• None (0 emoticons). • Few (1 to 3 emoticons). • Medium (4 to 45 emoticons). • Many (46 to 116 emoticons).
A3	It has hashtag	Defines the classifier's ability to interpret messages with words that are preceded by the "#" character.	• Yes • No
A4	It has elongations	Defines the classifier's ability to interpret messages with words deformed by excess characters.	• Yes • No
A5	It has contractions	Defines the classifier's ability to interpret messages that have words with contractions.	• Yes • No

Each sentiment classifier has specific values for these attributes or characteristics, as shown in Table 3. These characteristics can be viewed as strengths or weaknesses of the classifier.

Table 3. Attributes identified for each sentiment classifier.

		Attributes				
		A1	**A2**	**A3**	**A4**	**A5**
Classifiers	C1	Medium text and long text	Few	Yes	Yes	Yes
	C2	Medium text and long text	Few	Yes	No	Yes
	C3	Short text, Medium text, and Long text	Many	Yes	No	No
	C4	Short text, Medium text, and Long text	Many	Yes	Yes	No
	C5	Short text, Medium text, and Long text	Medium	No	No	Yes

Each classifier outputs a polarity measure according to the characteristics of the analyzed tweet. The more similar the characteristics of the tweet are to those of the classifier, the higher the quality of the resulting polarity. Finally, the Sentiment Classifier Module delivers a list with the polarity measures obtained from each classifier, which will be used later as an input parameter in the Statistic Module to calculate the final polarity.

2) Characteristics Identifier Module. The tweet enters this module to identify its characteristics. The module comprises five internal processes: size calculation, emoticon identification, hashtag identification, contraction identification, and word elongation identification. The size calculation counts the number of characters in the string representing the message. The message is divided by spaces (tokens), obtaining a list of words for emoticon identification. Each word is searched in the emoticon dictionary. Finally, the number of emoticons present in the message is defined. For hashtag identification, the message is divided into a list of words, and it is determined if each word contains the "#" character. For contraction identification, the message is divided into a list of words, and it is determined if a word matches any element of the contraction list. For elongation identification, the message is divided into a list of words; then, it checks whether there is a sequence of three consecutive identical characters in the analyzed word.

Finally, the Characteristics Identifier Module returns a five-position array containing the size of the message, the number of emoticons found, and the value of true or false in the remaining three positions corresponding to whether the message had hashtag, contractions, and elongations.

3) Fuzzy Logic Module. This module works with the results delivered by the Sentiment Classifier and Characteristics Identifier modules. This module was implemented using the XFuzzy development environment and allows obtaining a numerical value from the defuzzification process that works from a set of inference rules defined for each classifier. Fuzzy sets and inference rules are generated according to the characteristics of each one. The fuzzy sets contain mathematical functions evaluated from numerical limits, initially established manually. The inference rules are established according to the characteristics of each classifier. Therefore, each classifier will have a numerical value based on the similarity of its characteristics and the tweet characteristics. The obtained numerical value is known as weight and represents the priority with which the classifier prevails over the others.

B. Definition of Inference Rules

A specific set of inference rules was defined for each classifier based on the characteristics or attributes previously identified in Table 3. Equation (1) defines the total number of inference rules per classifier (TIR), where vA_i represents the number of possible values that the attribute A_i can take.

$$TIR = \prod_{i=1}^{5} vA_i \qquad (1)$$

The total number of inference rules per classifier is 96, covering all combinations of the five defined attributes (characteristics) that a tweet could have. See Figure 2.

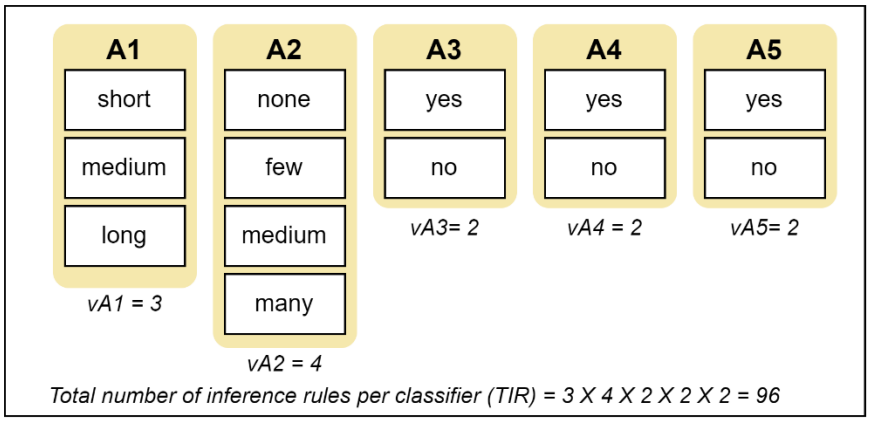

Fig. 2. Total number of inference rules per classifier.

Figure 3 shows part of the logical structure used to define the inference rules for each classifier.

```
if(size == short & emo == none & has == yes & elo == yes & con == yes) -> weight = E;
if(size == short & emo == none & has == yes & elo == yes & con == no) -> weight = C;
if(size == short & emo == none & has == yes & elo == no & con == yes) -> weight = C;
if(size == short & emo == none & has == yes & elo == no & con == no) -> weight = A;
if(size == short & emo == none & has == no & elo == yes & con == yes) -> weight = E;
if(size == short & emo == none & has == no & elo == yes & con == no) -> weight = C;
if(size == short & emo == none & has == no & elo == no & con == yes) -> weight = C;
if(size == short & emo == none & has == no & elo == no & con == no) -> weight = A;
```

Fig. 3. Excerpt of the Inference Rule Set.

The figure above shows that the structure of the inference rules is composed of *size*, referring to the size of the tweet; *emo*, indicating the number of emoticons; *has*, if the tweet has a hashtag; *elo*, if the tweet has elongations; and *con*, if the tweet has contractions. The result is known as weight and reflects the priority with which each classifier should be treated on the scale A, B, C, D, and E, being E the lowest priority and A is the highest. In addition to defining the inference rules, numerical ranges were also defined for each fuzzy variable (*size*, *emo*, *has*, *elo*, *con*), which may vary according to the characteristics of each classifier.

Figure 4 shows the numerical ranges assigned to each fuzzy variable. Each fuzzy value of each variable is associated with a mathematical function assigned according to the nature of the tweets' characteristics. Among the assigned mathematical

functions, there were bell, trapezoid, rectangle, and singleton. Finally, a total of 480 rules (5 classifiers per 96 inference rules) were created, the fuzzy system is based on these rules to infer the priority with which one classifier should be treated over another depending on the scenario in which it is found.

```
type Size [0.0,180.0;256]
 {
  short xfl.bell(0.0,15.0);
  medium xfsg.trapezoid(15.0,40.0,120.0,144.0);
  long xfl.bell(280.0,54.0);
}

type Binary [0.0,1.0;256]
{
  no xfl.singleton(0.0);
  yes xfl.singleton(1.0);
}
```

```
type Emoji [0.0,116.0;256]
 {
  none xfl.rectangle(0.0,0.99);
  few xfl.rectangle(1.0,3.0);
  medium xfl.trapezoid(2.0,8.0,40.0,45.0);
  many xfl.rectangle(45.0,116.0);
}
type Weight [0.0,1.0;256]
{
  E xfl.bell(0.0,0.15);
  D xfl.bell(0.25,0.15);
  C xfl.bell(0.5,0.15);
  B xfl.bell(0.75,0.15);
  A xfl.bell(1.0,0.15);
}
```

Fig. 4. Ranges of fuzzy logic rules

4) Statistic Module. This module depends on the results of the three previous modules and calculates the final polarity. The Statistic Module sends the final polarity obtained to the persistence component, which stores it as output data in a text file. The Statistic Module starts from the list of weights and the quality values assigned to the resulting polarity of each classifier. The final polarity delivered results from a mathematical calculation involving the polarity quality and the weights given by the fuzzy logic module. The process of calculating the final polarity FP is obtained by Equation (2), where n is the number of classifiers, W_i is the weight value of each classifier, P_i is the polarity value of each classifier (value between 0 and 1, where 0 is negative polarity, 0.5 neutral polarity, and 1 positive polarity).

$$FP = \frac{1}{n}\sum_{i=1}^{n}\left(\frac{W_i}{\sum_{i=1}^{n} W_i} \times P_i\right) \quad (2)$$

C. SentiFuzzy Algorithm

Table 4 presents the proposed algorithm in pseudocode, followed by its description.

Table 4. SentiFuzzy Algorithm

SentiFuzzy Algorithm

INPUT

1. tweet
2. Classifiers C [ ] // array of selected classifiers

ASSIGN

3. Characteristics CH [ ] // array with the characteristics of the tweet
4. Polarities P [ ] // resulting polarity of each classifier
5. Weight W [ ] // Weight for each classifier

OUTPUT

6. FP // final polarity

START

7. for i ←0 to CH.length-1 do
8. CH[i]← getTweetCharacteristics(tweet) // identifies the characteristics of the tweet
9. end for
10. for i ←0 to C.length-1 do
11. P[i]← C[i].classify(tweet) // gets the polarity of the tweet for the current classifier
12. W[i] ← fuzzyLogicProcess(P[i], CH) // getting weight scores
13. end for
14. FP ← statisticProcess(P,W)
15. return FP

END

The input variables are declared in the INPUT block, in lines 1 and 2, in this case, the tweet to be analyzed, and C corresponds to an array of sentiment classifiers (five for the current implementation).

The ASSIGN block formed from lines 3 to 5 declares CH, an array containing the characteristics of the read tweet; P, an array containing the polarity of the evaluated tweet for each classifier; and W, an array containing the weights of each classifier for the evaluated tweet.

The OUTPUT block formed by line 6 contains the FP variable to store the final polarity.

From lines 7 to 9, the CH characteristics vector is filled in according to the read tweet.

Through a repetitive structure between lines 10 and 13, the following actions are performed: In line 11, the *classify* method present in each classifier is called. This method is called directly from the library or API of each selected sentiment classifier.

The method receives the tweet as input parameter and returns the polarity, which is finally assigned to the polarity array *P*. In line 12, the *fuzzyLogicProcess* method is called, which internally executes the Fuzzy Logic module and allows obtaining the weights of each classifier for the evaluated tweet; these values are assigned to the *W* array. Next, the statistic module is called in line 14 using the *statisticProcess* method, which has the weights array *W* and the polarity array *P* as input parameters. Finally, the algorithm returns the final polarity *FP* for the evaluated tweet.

D. Optimization of the SentiFuzzy Algorithm

To improve the accuracy of the polarity delivered by the SentiFuzzy algorithm, the Hill-Climbing optimization algorithm was implemented to optimize the numerical values of the fuzzy set boundaries. This optimization algorithm aims to obtain the most appropriate numerical ranges in each fuzzy set without setting them manually. In Table 5, the optimization of the SentiFuzzy algorithm is presented in pseudocode, followed by its description.

In the INPUT block formed by line 1, the input variable initValues is declared; it stores the limits established by default or manually for each mathematical function defined in each of the fuzzy sets, which are structured in an array.

In the ASSIGN block from lines 2 to 4, S and R variables of the data type Solution are declared, which internally store an array and implements the methods fitness(), newRandomSolution(), and tweaks(), which will be described later. The optimal variable takes the value of 1, indicating the optimal number to be reached, representing 100% quality in polarity classification of tweaks.

In line 5, the variable BEST in type Solution is created and initialized with the values established in initValues. In line 6, the variable S invokes the method newRandomSolution(), which fills the internal vector found in said variable with random values representing the limits of the mathematical functions defined in each fuzzy set.

Table 5. SentiFuzzy algorithm optimization pseudocode.

```
Optimized SentiFuzzy algorithm
INPUT
    1.  Array initValues
ASSIGN
    2.  Solution S, R
    3.  Function F // In this case the optimization function is the SentiFuzzy algorithm
    4.  optimal = 1
START
    5.  BEST = new Solution (initValues)
    6.  S.newRandomSolution()
    7.  for times = 1 to n do
    8.      R = new Solution(S)
    9.      R.tweak(F)
    10.     if R.fitness() > S.fitness() then
    11.         S = R
    12.     end if
    13.     if S.fitness() > BEST.fitness() then
    14.         BEST = S
    15.     end if
    16.     if BEST.fitness() >= optimal then
    17.         break for
    18.     end if
    19. end for
    20. return BEST
END
```

From lines 9 to 21, a repetitive structure that iterates n times is executed, where n is a number previously defined by the user. In line 8, a copy of variable S is made and stored in variable R. In line 9, the variable R calls the method tweak (F), which performs small numeric changes in each array value contained as an attribute in the variable R. The values with the new changes are checked on F, the instance of the SentiFuzzy algorithm. In line 10, it is compared whether the fitness (accuracy of the polarity) of the variable R —to which the changes in the boundary values of the fuzzy sets were made— is greater than the fitness of the variable S —whose boundary values were randomly generated at the beginning or correspond to the best ones

defined up to the current iteration—, if the condition is satisfied, then S becomes the value of R.

In line 13, we compare whether S's fitness is better than BEST's. If the condition is satisfied, changing the fuzzy set value boundaries generates a better polarity than the polarity delivered by BEST, which initially works with default set boundaries. Therefore, the BEST solution happens to have a value of S. Line 16 asks whether the fitness or polarity of BEST is greater than or equal to the set optimal value. If the condition is met, the algorithm exits the repetitive structure, otherwise, it will continue iterating. Finally, line 22 returns the fitness or polarity stored in the BEST variable, representing the best polarity value reached.

III. RESULTS

Each sentiment classifier identified in Table 1, along with the optimized SentiFuzzy algorithm, were evaluated using the following datasets: vader_twitter (4200 tweets), sentistrength_twitter (4242 tweets), tweet_semevaltest (6087 tweets), and stanford_tweets (359 tweets), identified as DS1, DS2, DS3, and DS4, respectively. Each dataset was downloaded from the repository indicated [11]. These datasets were selected because the chosen classifiers had previously used them.

The resulting polarities obtained by the classifiers, including the proposed algorithm, were compared with those stored in the evaluated dataset to obtain the confusion matrix, as shown in Table 6.

Table 6. Confusion Matrix

		Predicted Polarity		
		Positives	Neutral	Negatives
Actual Polarity	Positives	True Positives (TP) *(a)*	False Neutral (FN) (b)	False Negatives (FNe) *(c)*
	Neutral	False Positives (FP) *(d)*	True Neutral (TN) *(e)*	False Negatives (FNe) *(f)*
	Negatives	False Positives (FP) *(g)*	False Neutral (FN) *(h)*	True Negative (TNe) *(i)*

Equation (3) defines the measure of accuracy; the percentage of hits between the records defined by the classifier as belonging to a class and those classified in that class within the evaluated dataset.

$$accuracy = \frac{correctly\ classified\ tweets}{total\ tweet\ registers} = \frac{a+e+i}{a+b+c+d+e+f+g+h+i} \quad (3)$$

The F-Score measure defines the percentage of hits globally over the dataset. This measure represents the harmonic mean of the precision and the recall of the classifier over the dataset [12]. The calculation is given in Equation (4).

$$F-score = 2\,\frac{(precision \times recall)}{(precision\ +\ recall)} \quad (4)$$

A. Evaluation of Results

Table 7 shows that the proposed SentiFuzzy optimized algorithm obtains a better quality in the polarity assignment measured in Accuracy in three of the four datasets evaluated concerning the five selected sentiment classifiers. When evaluating dataset 3, the proposed algorithm, despite not being the best, obtains competitive results against the other five classifiers. Concerning the average value (last row), the highest value is obtained by SentiFuzzy, followed by Text Analytics, SentiStrength, Natural Understanding Language, Vader Sentiment, and Tweepy. Applying Friedman's non-parametric statistical test yields the following ranking: SentiFuzzy as the best (rank 1.5), followed by Text Analytics (2.5), then SentiStrength (3.25), Vader Sentiment (3.5), Natural Understanding Language (4.25), and Tweepy (6). This test reports a statistic of 13.571429 (based on a chi-square distribution with 5 degrees of freedom) and a P-value of 0.018573626218943128 86; it indicates that there is at least one classifier with results that differ from the others. Applying Holm's post hoc with 95% significance, it is obtained that only SentiFuzzy and Tex Analytics obtain better results (dominate) than those reported by Tweepy; the other pairwise comparisons do not show a dominant relationship.

Table 7. Polarity accuracy of each classifier (Accuracy)

	Text Analytics	Natural Understanding Language	Tweepy	SentiStrength	Vader Sentiment	SentiFuzzy algorithm (Proposed)
DS1	0.79547619	0.53428571	0.32666666	0.71809523	0.59547619	**0.87476190**
DS2	0.55233380	0.58345120	0.52640264	0.59806695	0.58462989	**0.60348892**
DS3	**0.68473796**	0.59421718	0.57286019	0.59848858	0.63955971	0.61228848
DS4	0.76880222	0.70752089	0.27298050	0.65738161	0.45125348	**0.81337047**
Avg	0,70033754	0,604868745	0,42472750	0,643008093	0,567729818	**0,725977443**

Table 8 shows how the proposed algorithm outperforms measure F1 in two of the four datasets, reaching a high percentage: 92% in DS1 and 84% in DS4. Regarding datasets DS2 and DS3, the results are competitive with the other classifiers. The results of the Friedman test with a statistic of 18.142857 and a P-value of 0.002772592049973266 also show that at least one classifier has values with different distributions. The ranking places SentiFuzzy and Text Analytics (rank 1.5) first, followed by SentiStrength (3.25) and Natural Understanding Language (4.25), then Vader Sentiment (4.5), finally Tweepy (6). Applying Holm's post hoc with 95% significance, we obtain that SentiFuzzy and Text Analytics dominate the results reported by Tweepy, and Text Analytics dominates the results reported by Vader Sentiment.

Table 8. Polarity accuracy of each classifier (F-Score).

	Text Analytics	Natural Understanding Language	Tweepy	SentiStrength	Vader Sentiment	SentiFuzzy algorithm (Proposed)
DS1	0.81498101	0.60081632	0.43844320	0.77694046	0.681298068	**0.916108565**
DS2	**0.75970149**	0.45513413	0.30508474	0.52039381	0.454826733	0.664812287
DS3	**0.66486729**	0.47780126	0.32326913	0.54042873	0.5384939	0.647413919
DS4	0.81318681	0.69565217	0.35235732	0.66846361	0.501265823	**0.838196286**
Avg	0,76318415	0,55735097	0,35478860	0,626556653	0,543971131	**0,766632764**

IV. Discussion

This research indicates an improvement in the quality of the polarity assignment for Twitter sentiment analysis, supported by the average values obtained in accuracy, F1, and Friedman's rankings. However, analyzing the experimental results with Holm's post hoc, only significative improvements are reached against the worst classifier, in this case, Tweepy. Although the average results obtained by SentiFuzzy are better in several datasets against the other classifiers, it is only statistically significant against Tweepy. Therefore, more evaluations over more datasets are required to ensure the reliability, robustness, and generalizability of the findings while controlling the error rate; this helps prevent false findings and strengthens the overall quality of the statistical analysis.

V. Conclusions

SentiFuzzy is a fuzzy logic rule-based algorithm related to the results of five sentiment classifiers. The sentiment classifiers with which the algorithm works were selected with different characteristics, and these are evaluated based on their virtues and weaknesses.

With the characterization of the classifiers, it was possible to create a series of inference rules and fuzzy sets, from which mathematical weights were defined for the classifiers according to their classification nature. Through this process, it was possible to know which classifier was the most appropriate to analyze the tweet.

The numerical values of the limits of the mathematical functions in the fuzzy sets were assigned using the Hill-Climbing optimization algorithm, which improved the polarity result delivered by SentiFuzzy.

This first version of the SentiFuzzy algorithm evaluates the tweet based on only five attributes: tweet size, number of emoticons, presence of hashtag, elongations, and contractions. However, context situations, idioms, and sarcasm were not considered.

Preprocessing the tweet to normalize the message due to misspellings, use of elongations, jargon, hashtags, context, sarcasm, abbreviations, acronyms, and contractions could have directly influenced the polarity delivered by a specific sentiment classifier. The proposed algorithm did not perform data preprocessing since it analyzed the tweet as raw data, obtaining satisfactory and competitive results against the other classifiers and where an improvement in polarity accuracy was observed, both in the Accuracy and F-Score measures.

The solution approach proposed by SentiFuzzy enables thinking about different improvement options: 1) include more base classifiers; 2) define more or better attributes to characterize the classifiers; 3) automatically generate rules based on decision trees or Ripper-like coverage algorithms using data from running the algorithms with different datasets of tweets; and 4) evaluate other metaheuristics to optimize classifier weights.

ACKNOWLEDGMENTS

Professors Jimena Adriana Timaná Peña and Carlos Alberto Cobos Lozada are grateful for the contributions made by Universidad del Cauca, where they work as Associate and Full Professor, respectively.

AUTHORS' CONTRIBUTION

Jimena-Adriana Timaná-Peña: Research, Supervision, Methodology, Validation, Writing-review, and Editing.

Carlos-Alberto Cobos-Lozada: Research, Conceptualization, Validation, Writing-review, and Editing.

Jason-Paul Anturi- Martínez: Research, Methodology, Software, Validation, Writing-original draft.

José-Luis Paz-Realpe: Research, Methodology, Software, Validation, Writing-original draft.

REFERENCES

[1] M. Ahlgren, *55+ Twitter Statistics, Facts & Trends for 2023*, 2023. https://www.websiterating.com/research/twitter-statistics/

[2] A. Ankit, N. Saleena, "An Ensemble Classification System for Twitter Sentiment Analysis," *Procedia Computer Science*, vol. 132, pp. 937-946, 2018. https://doi.org/10.1016/j.procs.2018.05.109

[3] S. Al-Azani, E.-S. M. El-Alfy, "Early and Late Fusion of Emojis and Text to Enhance Opinion Mining," *IEEE Access*, vol. 9, pp. 121031-121045, 2021. https://doi.org/10.1109/ACCESS.2021.3108502

[4] J. Anturi, et al., "Clasificadores para el Análisis de Sentimientos en Twitter: Una revisión," in *Computer Science, Electronics and Industrial Engineering*, 2019.

[5] D. H. Wahid, S. N. Azhari, *Senti Strength ID*, 2016. https://github.com/masdevid/sentistrength_id

[6] C. Hutto, E. Gilbert, "VADER: A Parsimonious Rule-based Model for Sentiment Analysis of Social Media Text," in *Eighth International Conference on Web and SocialMedia*, 2014. https://doi.org/10.1609/icwsm.v8i1.14550

[7] S.Tari, M. Basseur, A Goëffon, "Expansion-based Hill-climbing" *Information Sciences*, vol. 649, e119635, 2023. https://doi.org/10.1016/j.ins.2023.119635

[8] Microsoft, *Text Analytics*, 2023. https://azure.microsoft.com/en-us/services/cognitive-services/text-analytics/

[9] IBM, *Natural Language Understanding*, 2023 https://www.ibm.com/watson/services/natural-language-understanding/

[10] Geeksforgeeks, *Tweepy*, 2023. https://www.geeksforgeeks.org/twitter-sentiment-analysis-using-python/

[11] M. Araujo, *iFeel Benchmarking Datasets*, 2016. https://bitbucket.org/matheusaraujo/ifeel-benchmarking-datasets/src/master/

[12] Y. Sasaki, *The Truth of the F-Measure*, 2007. https://www.cs.odu.edu/~mukka/cs795sum09dm/Lecturenotes/Day3/F-measure-YS-26Oct07.pdf