Expert Systems With Applications 208 (2022) 118103

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa

# A CBR for integrating sentiment and stress analysis for guiding users on social network sites

G. Aguado *, V. Julian, A. Garcia-Fornes, A. Espinosa

*Valencian Research Institute for Artificial Intelligence (VRAIn), Universitat Politècnica de València, Camino de Vera s/n, Valencia, Spain*

ARTICLE INFO

*Keywords:*
Multi-agent system
Social networks
Sentiment analysis
Stress analysis
Case-based reasoning

ABSTRACT

This work presents a Case-Based Reasoning (CBR) module that integrates sentiment and stress analysis on text and keystroke dynamics data with context information of users interacting on Social Network Sites (SNSs). The context information used in this work is the history of positive or negative messages of the user, and the topics being discussed on the SNSs. The CBR module uses this data to generate useful feedback for users, providing them with warnings if it detects potential future negative repercussions caused by the interaction of the users in the system. We aim to help create a safer and more satisfactory experience for users on SNSs or in other social environments. In a set of experiments, we compare the effectiveness of the CBR module to the effectiveness of different affective state detection methods. We compare the capacity to detect cases of messages that would generate future problems or negative repercussions on the SNS. For this purpose, we use messages generated in a private SNS, called Pesedia. In the experiments in the laboratory, the CBR module managed to outperform the other proposed analyzers in almost every case. The CBR module was fine-tuned to explore its performance when populating the case base with different configurations.

## 1. Introduction

In the daily environment in which people are immersed, there are a great number of on-line applications that are used to interact with others, obtain information, and perform different tasks. The tasks can be related to jobs, leisure, or other kinds of activities. Consequently, people are influenced by the information that exists and is being shared on on-line platforms and in applications. Among the most important and preeminent on-line applications are Social Network Sites (SNSs), which are used for interacting and communicating with other people. These sites offer a forum in which people (users) can post their information to be read by others. However, this kind of interaction is not free from risk. Vanderhoven, Schellens, Vanderlinde, and Valcke (2016) reviewed the risks and negative outcomes of users navigating and interacting in this context. De Moor et al. (2008), Livingstone, Haddon, Görzig, and Ólafsson (2011) reviewed different types of risk, which include content, contact, and commercial risks. Those involve the reception of harmful content, communication with dangerous individuals, and spam and aggressive marketing campaigns, respectively. Additionally, Vandenhoven, Schellens, and Valacke (2014) reported that teenagers face several risks while navigating SNSs, and they have special characteristics that make them more vulnerable to such risks.

Unless users navigating on-line sites are guided by the system or other entities, they are on their own when interacting with other users. They have to make decisions regarding how to interact and with whom to interact. Therefore, the decision-making process drives their interaction, and if they are not well informed, they might suffer the effects of risks and negative outcomes. Decision-making is affected by the emotional state of the person who makes the decision, as has been reported in George and Dane (2016). The authors reviewed the effect of incidental moods, discrete emotions, integral affect, and regret on decision-making, showing that they all affect it. Incidental moods and discrete emotions refer to affective states that are not directly linked with the task at hand and that can originate from other sources, (e.g., thinking of someone that is not directly linked with the task being performed). In contrast, integral affect originates from the task being worked on and not from external sources. Differently, regret is a negative and conscious emotional reaction to self decision-making. The authors showed that incidental moods affect decision making by altering the perception of people and that regret affects decision making acting as anticipating regret, (e.g., thinking of the negative outcome before it happens).

* Corresponding author.
*E-mail addresses:* guiagsar@dsic.upv.es (G. Aguado), vinglada@dsic.upv.es (V. Julian), agarcia@dsic.upv.es (A. Garcia-Fornes), aespinos@upvnet.upv.es (A. Espinosa).

https://doi.org/10.1016/j.eswa.2022.118103
Received 28 June 2021; Received in revised form 1 July 2022; Accepted 5 July 2022
Available online 9 July 2022

A system that guides the decision-making process of users is, therefore, useful to keep them from incurring risk and suffering its consequences. Since the decision-making process is influenced by emotion, the system could monitor the emotion of users in order to predict potentially negative repercussions coming from the interaction. In Thelwall (2017), stress was associated with an emotional state (high arousal and negative valence) and was used to construct an algorithm to detect the magnitude of stress and relaxation in texts. Therefore, it might be useful to incorporate a module that is specialized in detecting stress levels for a system that guides the decision process of users on on-line sites.

To achieve a system that can analyze user affective states and guide users, in Aguado, Julian, Garcia-Fornes, and Espinosa (2020a), we presented a Multi-Agent System (MAS) as a system that collects messages from users that are interacting on a SNS or in other social environments. It computes sentiment, stress, and performs a combined analysis to generate feedback to the user in the form of a warning that aims to prevent future negative repercussions in the social environment. Keystroke dynamics data is timing information and frequency of pulsation of keys that can be collected when a user is typing on a keyboard. It can be used as an additional source of information for a data analysis application. In Aguado, Julián, García-Fornes, and Espinosa (2020b), agents performing sentiment and stress analysis on keystroke dynamics data were created and used together with agents that analyze text data for performing sentiment and stress analysis. Different analyzers were proposed, including single modality analysis agents that performed sentiment and stress analysis on text or keystroke data and several late fusion analysis agents. The experiments were conducted with data from a private SNS called Pesedia (Bordera, 2016) to determine which analyzers were more effective at detecting the user states that propagated the most on the SNS. Finally, an Advisor agent that generates feedback for users on a SNS was designed according to the results of the experiments, using a combination of agents that perform analyses on text and keystroke data and a set of rules.

The detection of the emotional state and stress levels of a user when he or she is writing a message is not the only source of information available from the SNS that could be used to generate feedback. Other sources, such as the history of polarities and stress levels of the user, the history of the audience where the message is about to be posted, or the topics that could be extracted from the message are all examples of other sources of information that may prove to be useful in generating feedback to the user. These may obtain better results than a simple analysis on the messages to avoid potentially negative outcomes in the social environment. In Case-Based Reasoning (CBR) systems, a reasoner remembers previous situations that are similar to the current one and uses them to solve the new problem (Kolodner, 1993). Since the cases can contain several different features to represent a specific situation in the system, a CBR system could be used to combine different aspects of a user state as well as external factors to help decide what action the system should take in order to guide this user and potentially prevent a negative outcome.

We implemented and integrated a CBR module into a MAS to help it detect a case on a SNS where a user could generate a negative repercussion and to enable the system to prevent it by applying the corresponding action to each case. The system is able to take into consideration different sources of information and also previous interactions, and it can exploit this information to guide users. This MAS is a variation of the one presented in Aguado et al. (2020a) and the one presented in Aguado et al. (2020b). The MAS works by extracting information about a text message being posted on a SNS, which includes the text of the message, the keystroke dynamics data associated to it, the audience that may see it, the user that posted the message, and the time of the day. The MAS agents then perform sentiment and stress analysis on the text and keystroke dynamics data and use stored information about past analyses and topic detection in texts to generate more information for the CBR module. Finally, the CBR module in the MAS generates a case with this information and calculates which case from its case base is the most similar to the current case in order to further recommend an action to be performed (e.g., warning the user to avoid potential future risky situations). This process is explained extensively in Section 3.

In Aguado et al. (2020a), we conducted a set of experiments with data from the SNS Twitter.com to determine which of the analyses used in the MAS was able to detect the state of the users that propagated the most to the replies of the messages. As a metric of propagation, we used the most frequently detected state in the replies of the messages. In the present work, we have conducted a set of experiments to discover not only which of the analyses is able to detect a state that propagates the most in the SNS, but also to compare single analyses to the prediction of the CBR module. We performed a set of experiments with people at the laboratory using Pesedia for a period of one month, and we used this data to compare the analyses and the CBR module. We also performed experiments to analyze the difference in the error of the CBR module after populating the case base with different parameters. The experiments and results are discussed in Section 4.

Thus, the contribution of this paper is to demonstrate that by using information about different aspects of the users' affective state along with context information (modeled as a case and using a CBR module), a system is able to predict negative repercussions in an on-line social environment such as a SNS better than affective state detection methods alone. For this purpose, experiments were conducted with data from Pesedia to assess what the differences are between the analyses and the CBR module in predicting a state of the user that propagates the most on the SNS so that it would be more informative to generate a warning or feedback to prevent negative repercussions. This objective required the design and implementation of a CBR module that is able to generate cases of previous interactions of users in a SNS using the information of the history of analyses, information retrieved from the SNS, and analyses done at the moment to recommend an action to prevent potentially negative repercussions in a SNS. This CBR-based approach is a way to use information that is related to the user state and context of the interactions in order to predict potentially negative outcomes in the SNS. To the best of our knowledge, this has not been done before. Finally, we fine-tuned the CBR module by performing experiments varying its parameters and populating the case base to discover which set of parameters achieved the lowest error rate in the predictions of the CBR module.

The following sections are organized as follows. Section 2 presents a review of state-of-art works that are relevant to this work. Section 3 describes the MAS, the integrated CBR module and explains how the system works in general. Section 4 describes the experiments conducted with data from Pesedia and presents the conclusions drawn from them. Finally, Section 5 presents general conclusions and proposes future lines of work.

## 2. Related work

In this section, a brief review of state-of-art works on sentiment analysis and stress analysis on text and keystroke dynamics data is provided, as well as a review of current works on CBR-based systems and works where the user state is modeled in order to perform a task. Works in risk prevention and privacy aiding in SNSs are also reviewed.

According to Liu (2012), sentiment analysis is a research line that focuses on recognizing opinions, sentiments, evaluations, appraisals, attitudes, and emotions in different media (e.g., texts, images, audio). This analysis is based on the level of fine-grained analysis that is performed (e.g., document level, sentence level, and aspect level). Document-level and sentence-level sentiment analyses are performed on a whole document or individual sentences, respectively, while aspect-based sentiment analysis refers to the detection of sentiment in specific aspects of the text (e.g., sequences of words, single words) (Feldman, 2013). For document-level sentiment analysis, Basiri, Nemati, Abdar, Cambria, and Acharya (2021) proposed an architecture that

combines a text embedding layer with two independent bidirectional Long Short-Term Memory (LSTM) and bidirectional Gated Recurrent Unit (GRU) networks that extract both past and future contexts as semantic representations of the input text. An attention layer is applied to the outputs of both the LSTM and GRU models to place more or less emphasis on different words from the text input. Afterwards, the semantic representations are passed to a Convolutional Neural Network (CNN) layer and then to a dense layer that outputs a sentiment polarity. Experiments were performed comparing the proposed architecture with six recently published deep neural models for sentiment analysis on five review datasets and three Twitter datasets. The results showed that the proposed model outperformed the other six models in almost every case in terms of precision and F1, with the difference being greater in the review datasets. Additionally, Akhtar, Ekbal, and Cambria (2020) presented a Multi-Layer Perceptron (MLP)-based stacked ensemble architecture for sentiment and emotion intensity. It consists of four individual models (LSTM, CNN, GRU, and one feature-driven classical supervised model based on Support Vector Regression or SVR) that are stacked with a three-layer network that combines the outputs of the individual models to predict sentiment and emotion intensity. The stacked ensemble model outperformed the individual models and state-of-art models except in the task of prediction of two specific emotions (joy and sadness). Cambria, Li, Xing, Poria, and Kwok (2020) proposed SenticNet 6 as a long sentiment lexicon (composed of 200,000 words and multiword expressions). It was built using two types of models, sub-symbolic models and symbolic models. The sub-symbolic models (biLSTM and Bidirectional Encoder Representations from Transformers or BERT) generalize words and multi-word expressions into primitives that were later defined manually in terms of super-primitives, and symbolic models (logic and semantic networks) extract meaning and assign sentiment polarity to high-level concepts. The authors compared SenticNet with 15 popular sentiment lexica, testing against six commonly used benchmarks for sentence-level sentiment analysis. The results showed that SenticNet 6 was the best-performing lexicon. Finally, in aspect-based sentiment analysis, there are two main tasks to perform: aspect detection and sentiment classification (Schouten & Frasincar, 2016). Aspect detection is the process of generating a set of aspects from the source data used in the training so that these aspects can later be used in the sentiment analysis model to detect sentiment in texts. Sentiment classification is the process of labeling aspects with a sentiment polarity. In our system, we chose to use aspect-based sentiment analysis in order to be able to perform a more fine-grained analysis of the source text messages.

For the task of aspect detection, there are frequency-based methods that select terms with the highest frequency in the training data as aspects for the model (Hu & Liu, 2004). Conditional Random Fields (CRFs) is an example of generative models that are used for this task. They make use of a varied set of features (Jakob & Gurevych, 2010). Machine learning techniques can also be used, in Blei, Ng, and Jordan (2003) non-supervised machine learning techniques such as Latent Dirichlet Allocation (LDA) are employed for aspect detection.

For the task of sentiment classification, dictionary-based methods assign polarities to aspects in a dictionary called aspect set using machine learning techniques or other techniques. A sentiment polarity is then associated to a text based on the polarity of the aspects from the aspect set that are found in the text. For example, the most frequent polarity in the aspects found in the text, or machine learning techniques using Support Vector Regression with other techniques to obtain the features for training the model, and non-supervised methods that use relaxation labeling (Schouten & Frasincar, 2016).

Aspect-level sentiment analysis can be performed using hybrid techniques; this consist of obtaining aspects and assigning polarities simultaneously. In Nasukawa and Yi (2003), syntax-based methods are used to obtain words that are associated with a sentiment polarity. Then, other aspects are extracted by means of exploiting grammatical relations. In Li et al. (2010), CRFs are used to relate sentiment polarities to aspects by extracting the information from relations between words. Aspect-based sentiment analysis is able to extract fine-grained sentiment information from text. Nevertheless, in order to assess the state of the user and provide guidance and recommendations, it might be more useful to extract information about the context of the conversations (e.g., the topic being discussed, the listeners, etc.) and combine it with the use of sentiment classifiers.

Multimodal sentiment analysis is a line of research that aims to perform a fusion of different data types in sentiment analysis models in order to achieve better results than only using one data type. This line of research has recently been gaining an increased amount of attention from the research community. In multimodal sentiment analysis, three main approaches are used: early fusion, intermediate fusion, and late fusion (Huang, Zhang, Zhao, Xu, & Li, 2019). Early fusion or feature-level fusion combines different data type sources in a single data structure such as a feature vector that is later fed to a model. In contrast to early fusion, intermediate fusion refers to performing a fusion of data in intermediate layers of the model used. Late fusion refers to the combination of the output of models that use different data types to perform the sentiment classification.

As an example of early fusion, in Poria, Chaturvedi, Cambria, and Hussain (2016) the authors used audio, video, and text feature fusion using a multiple kernel learning classifier. An example of intermediate fusion is performed in Huang et al. (2019), where three models are presented. Two of them are unimodal models featuring sentiment classification using deep CNN on image data and a LSTM on text data. The third model features a combination of the output visual features from the CNN model and the output text features from the LSTM to feed a fully connected layer to obtain a sentiment label. The authors also presented a late fusion framework, combining the output of the three presented models to perform sentiment classification. Moreover, Camacho, Panizo-LLedot, Bello-Orgaz, Gonzalez-Pardo, and Cambria (2020) presented four dimensions of Social Network Analysis (SNA), along with four metrics to quantitatively measure them in existing or future technologies. These are: (1) pattern and knowledge discovery, which determines the amount of valuable knowledge that a tool can extract from data; (2) scalability, which measures how scalable a tool is; (3) information fusion and integration, which measures the capacity of SNA technologies to integrate and fuse information from different data types and different sources; and (4) visualization, which measures the capacity of the technology to allow the visualization of information that is extracted from data. The authors computed the proposed metrics on a set of 20 popular SNA-software tools. They concluded that even though current technologies are scalable, can already handle significant amounts of data, and there are a large number of tools that provide flexible methods to visualize the information, the content of SNSs is not being fully exploited. Most of the analyzed tools are only capable of processing two or three different data types, and only from one SNS. Therefore, there is considerable room for future research on data fusion and integration. Different strategies for sentiment analysis have been employed implementing the analysis of different data types in the literature. Nevertheless, many SNA technologies are not capable of processing more than two or three different data types and only extract information from one SNS. To the best of our knowledge, there is not an approach in the literature (other than our proposed system) that employs sentiment and stress analysis using text and keystroke dynamics data to guide and prevent negative repercussions of users navigating in on-line social environments.

Stress strength detection using sentiment analysis techniques has been addressed in the literature in the past. A derivation of the sentiment strength detection software on text data, SentiStrength (Thelwall, Buckley, Paltoglou, Cai, & Kappas, 2010), is made in Thelwall (2017) using a set of terms labeled with stress levels and a set labeled with relaxation levels to detect stress and relaxation levels in sentences. This algorithm also modifies the detected stress or relaxation level in a sentence based on several factors that are independent of the

analysis on the aspects found. For the training of the aspect sets, an unsupervised learning method that is later refined with a hill-climbing method is used. Sentiment and stress analysis has also been performed on keystroke dynamics data in the literature. Sentiment analysis on keystroke dynamics data was addressed in Lee, Tsui, and Hsiao (2015) In this work, International Affective Digitized Sounds (IADS) (Bradley & Lang, 2007) was used to induce different sentiments to users and their keystroke dynamics were recorded when they heard them. The effect of arousal on keystroke duration and keystroke latency was observed to be significant but the effect on accuracy rate of keyboard typing was not significant. In Vizer, Zhou, and Sears (2009), keystroke dynamics and linguistic features were used to successfully detect cognitive and physical stress from free text data. According to the authors, the accuracy of detection of cognitive stress was consistent with the accuracy obtained using affective computing methods. The accuracy of detection of physical stress was lower than the accuracy obtained when detecting cognitive stress, but it was enough to encourage further research. Even though these methods can extract information about stress levels and sentiment polarities, they do not use multiple data types, which might help a system to better model the user state.

As explained by Kolodner (1993), CBR systems are used to generate a case from different characteristics of the environment or user interaction and to compare this case to a database of previous cases in order to extract a potential solution or action to be taken in the current situation of a system. In Heras et al. (2009), the authors integrated a CBR module into a help desk software as a solution recommender in order to be able to help operators in customer support environments. In Marie et al. (2019), the authors performed image segmentation of deformed kidneys using a CBR system and a CNN. A comparison of the results showed that the CBR system succeeded in performing a better image segmentation than the CNN. Bridge and Healy (2012) proposed a CBR system to act as a recommendation system between users and a review site. The CBR system recommends users certain sentences from previous reviews found to be similar to the sentence they wrote. The case base is populated with reviews from Amazon.com. The reviews are decomposed by the CBR into words, sentences to recommend to users, and a helpfulness rate of the review, which is created by Amazon.com users. Ohana, Delany, and Tierney (2012) performed sentiment analysis using a CBR-based approach. In this approach, labeled customer reviews and five different sentiment lexicons are used to populate the case base. Cases are created when a document is successfully classified by at least one lexicon. The cases contain document statistics, the writing style of the review, and the associated solution. The solution is the lexicon(s) that were used to correctly predict the sentiment of the review that generated it. For prediction, the k most similar cases (1, 2, or 3 in the experiments) are retrieved, and the lexicons of the solutions are reused for the new case. In Ceci, Goncalves, and Weber (2016), domain ontology and natural language processing techniques are used to perform sentiment analysis, and case-based reasoning is used to learn from past sentiment polarizations. According to the authors, the accuracy obtained by the proposed model overcomes standard statistical approaches. Tian et al. (2014) presented a CBR system with a manually-constructed case base of emotion regulation scenarios for e-learning. The cases represent events in which e-learners have certain emotions and the solutions to the cases are the advice and sentences recommended to regulate their emotions. The similarity between the speaker's sentences and the sentences in the cases is measured in order to select one case for emotion regulation. The authors claim that the experimental results show that the proposed method plays a positive role in emotion regulation in interactive text-based applications.

Modeling the user state in a system can be useful to perform several tasks, including sentiment analysis tasks. In Seroussi, Zukerman, and Bohnert (2010), a nearest-neighbor collaborative approach was used to train user-specific classifiers, and those classifiers were later combined with user similarity measurements to solve a sentiment analysis task. Moreover, modeling the emotion from a group of entities has also been addressed. In Rincon, de la Prieta, Zanardini, Julian, and Carrascosa (2017), the authors modeled the emotions of a group of entities using the Pleasure, Arousal, and Dominance (PAD) emotional space. They used an Artificial Neural Network (ANN) to learn the emotion of the group when an event happens. There are also works that apply a CBR approach to sentiment analysis to later perform a task using the detected user sentiment. In Muhammad, Lawlor, Rafter, and Smyth (2015), the authors implemented a CBR system that used opinions mined from user-generated reviews to help users make decisions in recommendation systems; In Zhou, Jianxin Jiao, and Linsey (2015), a two-layer approach was used to detect implicit customer needs in on-line reviews. The first layer uses sentiment analysis to extract explicit customer needs from reviews, using Support Vector Machines (SVM). The second layer uses a CBR module to identify implicit characteristics of customer needs. Nevertheless, to the best of our knowledge, even though there are works that use CBR modules to improve the performance of sentiment analysis, none of the existing solutions use a CBR module in combination with different analyses of the user state (sentiment analysis, stress analysis, and different data types) or a combination of analyses and context information. Moreover, none of them use this information to generate recommendations and guide users in a system to help avoid potential future problems in on-line interaction.

Risk prevention, user guiding, and privacy aiding in SNSs are currently important topics. Privacy helping or aiding has been addressed in Xie and Kang (2015) by using a user interface that is designed for that purpose. The main privacy features in the system are visible to the users by introducing privacy reminders and customized privacy settings. In their work, privacy aiding is addressed in an indirect way. It is done by analyzing different aspects of the mental state of the users in order to determine if they are in a state that could lead to incurring risks from interaction with other users. An example of user protection in SNSs using sentiment analysis is provided in Upadhyay, Chaudhari, Ghale, Pawar, et al. (2017). The authors implemented an SNS that uses adult image detection, a message classification algorithm, and sentiment analysis in text messages. The system uses this information to ban users that incurred in on-line grooming and cyber-bullying. Although the use of sentiment analysis to prevent negative outcomes in SNSs has been addressed, it might be useful to use detection of different aspects of the users' mental state (e.g., sentiment analysis, stress analysis, combined analysis) together with context information to obtain better results.

## 3. System description

In a previous effort to tackle user state detection to guide users navigating on SNSs, Aguado et al. (2020b) presented a MAS that computed sentiment, stress, and combined analysis of sentiment and stress on text data and keystroke dynamics data of user messages when they interacted on a SNS. The MAS uses the SPADE[1] multi-agent platform to implement the agents of the system. There are several agents in the MAS that perform different tasks and communicate with each other using a messaging interface that is based on the FIPA-ACL language. Moreover, there are three different agent types on the MAS: (1) the Presentation agents, which are in charge of communicating with the SNS, receiving data, and sending feedback to the users; (2) the Logic agents, which perform analyses on the messages and generate feedback if it is needed; and (3) the Persistence agent, which performs the data storage and retrieval tasks. Experiments with data from Twitter.com were conducted to determine which analyzer detected a state that propagated the most in the network, thus demonstrating it to be more useful in preventing potentially negative repercussions.

[1] https://spade-mas.readthedocs.io/en/latest/.

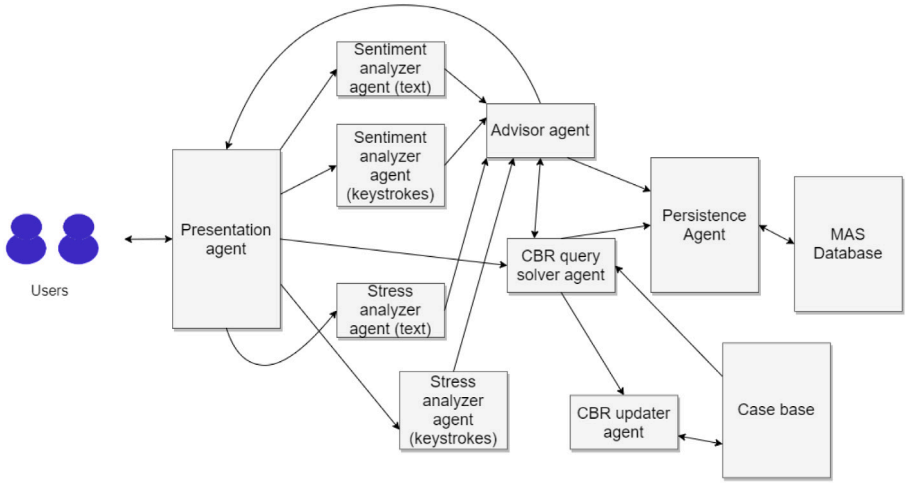


**Fig. 1.** Architecture of the MAS.

In this work, we present a new version of the MAS, which incorporates a CBR module in the pipeline of agents. The task of the CBR module is to predict a positive or negative repercussion in an on-line social environment based on the detected values of affective states from users interacting in the environment and context information from interactions. This CBR module consists of two logic type agents. One agent selects a case from the case base when a new message appears in the MAS based on the similarity of the new case associated with this message (that is generated by this agent) to the ones in the case base. The prediction of the case selected is sent to the Advisor agent to potentially generate feedback as warnings to the user when necessary. The other agent is in charge of updating the case base, adding new cases based on new messages received and also updating the priority of the cases. Thus, the agent makes the cases either more likely to be selected (if the predictions made with them were correct) or more unlikely, even erasing the case when the priority is under a set threshold (if the predictions made were not correct). The architecture of the system is shown in Fig. 1.

The Presentation agent is assigned the tasks of receiving data from the SNS and sending feedback to the users navigating on the SNS. For the Logic agents, we find a pipeline of agents that perform the process needed to generate feedback to the user. This pipeline begins when the Presentation agent sends the data of the messages to the Sentiment and Stress analyzer agents that work on text data and to the Sentiment and Stress analyzer agents that work on keystroke data. When the sentiment and stress analyses have been computed, the outputs are sent to the Advisor agent, which sends the data gathered about the message and the output of the analyses to the CBR query solver agent, which is in charge of finding the best matching case in the case base. The Advisor agent also sends the output of the analyses and the messages to the Persistence agent to save this information in the database. The CBR query solver agent generates a case associated with the message being analyzed, performs matching with cases in the case base, and later gives the information of the solution of the selected case to the Advisor agent to create feedback if the message is deemed negative. The feedback is stored in the database and sent to the Presentation agent, which delivers it to the SNS and to the user.

The Sentiment and Stress analyzer agents that work with text data perform an analysis using feed-forward ANNs. The ANNs use text embeddings to transform the text into embedding arrays. The embeddings used were pre-trained with the unsupervised algorithm GloVe (Pennington, Socher, & Manning, 2014), using the Spanish Billion Words Corpus (Cardellino, 2019). The Sentiment analyzer provides the positive or negative sentiment class as output. The stress analyzer provides the low or high stress level class as output. The Sentiment and Stress analyzer agents that work with keystroke data also use ANNs, which are fed with the arrays of keystroke features including timing and frequency of pulsation features. They provide the same output classes as the analyzers working with text. The different analyzers using ANNs were trained with Tensorflow (https://www.tensorflow.org) version 1.8.0 and Keras (https://keras.io) version 2.2.0 in the Python language version 3.5.2. Figs. 2 and 3 show the architectures of the ANNs for the ANNs that operate with text embeddings and compute sentiment analysis and stress analysis, respectively. Fig. 4 shows the architecture for the ANN that operates with keystroke dynamics data and computes sentiment analysis. Both the keystroke dynamics stress analysis ANN and the keystroke dynamics sentiment analysis ANN have the same architecture, with the exception of the output which is high or low stress instead of positive or negative class. The parameters of the ANNs were set in the training process aiming for the best accuracy of the models and a balanced confusion matrix. These parameters were the specific architecture, the dropout rates, the activation function for the dense layers, the dimensionality in the internal dense layers output, the loss function, and the optimizer. ANNs were chosen for the implementation of the analyzer agents because they can find nonlinear patterns in non-parametrized data (Grossi & Buscema, 2007) such as text messages or arrays of timing and frequencies of pulsation. This is the data that is used to learn sentiment and stress level states in this work.

In the architecture of the ANN for text sentiment analysis, the flatten layer converts the embeddings obtained from the text input in the embedding layer (which uses tokens generated from a text message) into a one-dimensional vector that feeds a dense layer. After the dense layer, a dropout layer with a dropout rate of 0.25 acts as a regularization mechanism. Other dropout rates such as 0.5 and 0.1 were used, but 0.25 achieved the best results. Afterwards, there are other dense and dropout layers (with 0.25 as dropout rate), and, finally, the output layer, which is also a dense layer. The activation function in the dense layers selected was the sigmoid function. The dimensionality in the internal dense layers output selected was 64. The loss function selected was binary cross-entropy and the selected optimizer was Adam (Kingma & Ba, 2014). The architecture of the ANN for the stress analysis on text has the same parameters and architecture, except that there are no internal dense or dropout layers. Finally, the ANNs for the keystroke dynamics analysis have the same parameters and architecture as the ANN for sentiment analysis of text, except that the loss function used was categorical cross-entropy. This worked best for the ANNs working with keystroke dynamics array data. These ANNs also have a different input. It is the array of floating-point numbers corresponding to typing speed features and frequencies of pulsation of common keys, which is fed to the first dense layer (there are no embedding or flatten layers). These features are summarized in Table 1.

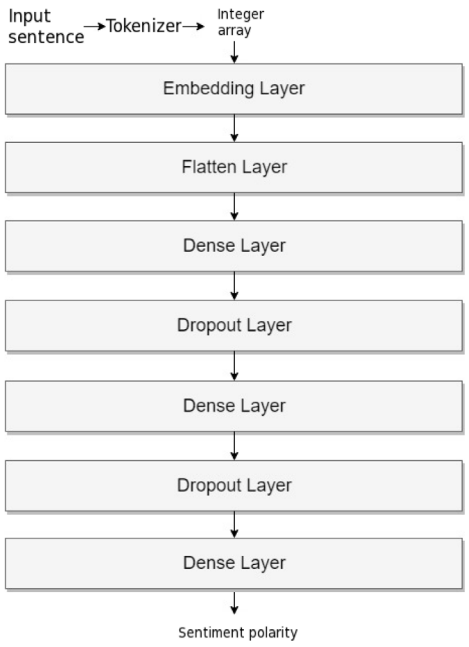


**Fig. 2.** Architecture for the sentiment analysis ANN that works with text embeddings.

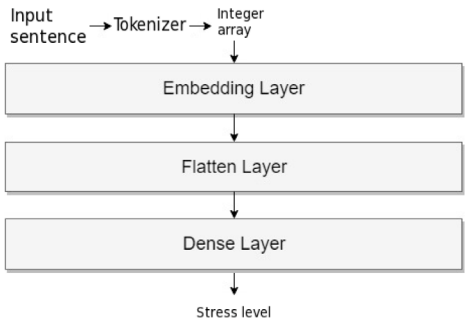


**Fig. 3.** Architecture for the stress analysis ANN that works with text embeddings.

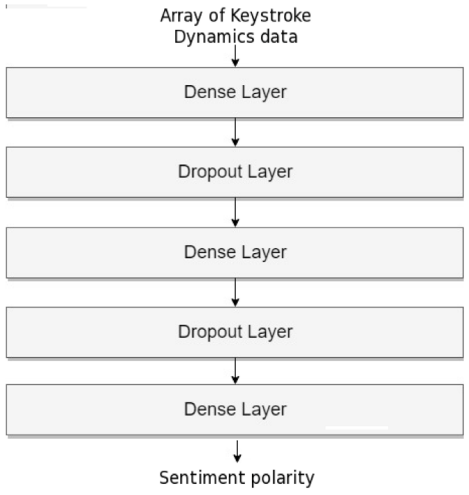


**Fig. 4.** Architecture for the sentiment analysis ANN that works with keystroke dynamics data.

The dataset used to train the ANN models was created by both males and females ranging in age from 12 to 15 years old in the SNS Pesedia. The data was labeled with positive or negative emotions and high or low stress levels using self-report (the users could choose to label their messages and only the labeled ones were added to the dataset). The training dataset contains 6475 labeled text messages with associated keystroke dynamics data.

**Table 1**
Text typing speed and key frequency features used as input for the keystroke dynamics ANN models.

| Text typing speed features | Key frequency features |
|---|---|
| Key press | Enter |
| Key release and press interval | Space bar |
| Key press and second press interval | Back space |
| Key release and second release interval | Delete |
| Key press related to digraphs | Key up |
| Key release and press interval related to digraphs | Key down |
| Key press related to trigraphs | Key left |
| Key release and press interval related to trigraphs | Key right |
| Digraph typing | Shift |
| Trigraph typing | Home |
| General typing speed | End |
| | Page up |
| | Page down |
| | Caps lock |

The process that is carried out by the CBR query solver agent and the process performed by the CBR updater agent (which updates the case base periodically) is detailed in Section 3.1. Finally, the Persistence agent performs the actions needed to store and retrieve information about sentiment and stress labels or past predictions and messages.

### 3.1. CBR module

The CBR module, which is integrated into the proposed MAS, is composed of the CBR query solver agent and the CBR updater agent, as well as the case base and several data structures that are needed for the functioning of the module. These data structures correspond to dictionaries for storing the priorities of cases, the predictions made by the CBR module, the parent structure of messages from the SNS, the degree of matching of potential new cases, and the historical values of states detected by the CBR module in messages written by users. There is also a file that contains the last time when the module updated the case base. The traditional four steps in the CBR cycle are performed by these two agents. Those steps are retrieve, reuse, revise, and retain (Aamodt & Plaza, 1994). The process carried out by the agents for each step is described in the following subsections. After receiving the output of the analyses, the author of the message, the audience of the message, and the time of the day when it was created, the CBR module creates a case with the following:

1. The time of the day.
2. The output of the text Sentiment analyzer as an integer (0 or 1 for negative or positive class, respectively).
3. The output of the text Stress analyzer as an integer (0 or 1 for negative or positive class, respectively).
4. The output of the keystroke Sentiment analyzer as an integer (0 or 1 for negative or positive class, respectively).
5. The output of the keystroke Stress analyzer as an integer (0 or 1 for negative or positive class, respectively).
6. The computed history of messages detected as negative or positive composed by the user writing the message sent to the CBR module as the average from the last set of interactions from this user (up to ten). The negative messages are labeled with 0 and the positive messages are labeled with 1 in the code (e.g., if 3 of the last 4 user messages were labeled as positive and 1 as negative, then the average would be computed as 3 + 0/4 = 0.75).
7. The computed history of messages detected as negative or positive in messages written by the audience as the average among all viewers (from the averaged values per user).

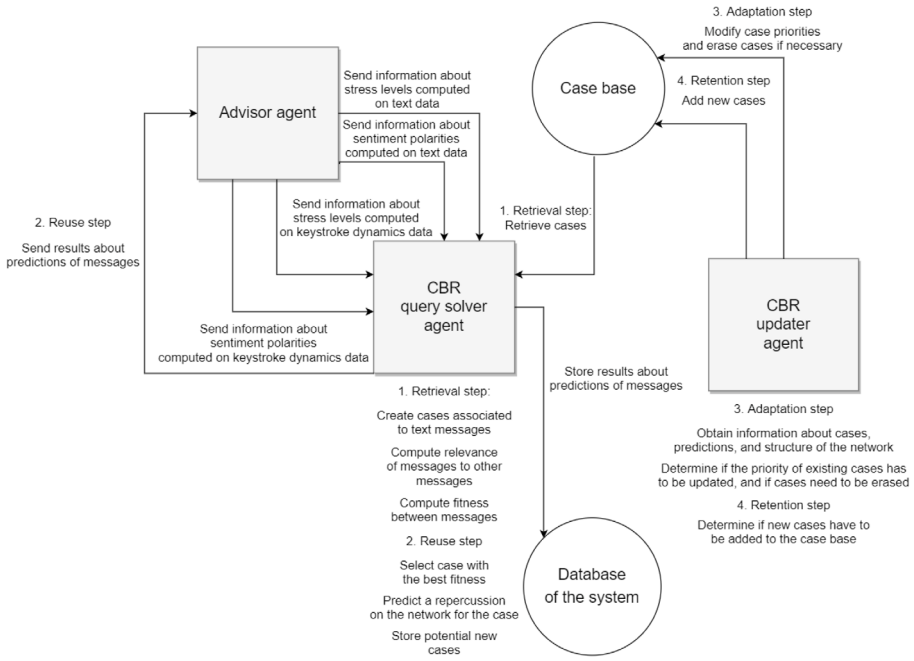


**Fig. 5.** Diagram of actors and actions that form the CBR module and the elements that interact with it in the system.

8. The computed topic found in the text message, using a trained Latent Semantic Indexing (LSI) (Deerwester, Dumais, Furnas, Landauer, & Harshman, 1990) model and the gensim[2] library.

The history for users is computed using a window of the last ten messages for each user and the outputs of a late fusion method of sentiment and stress analysis on text and keystroke dynamics data on the messages. Thus, the history is computed as the average of the output from the decision-fusion approach detected in the last ten messages written for each user. For the history of the audience of a message, an average of the history values for every user pertaining to the audience is used. The solution or final prediction derived from a case is whether the case will create a positive (represented as 0) or negative (represented as 1) repercussion on the SNS after the message is posted. This repercussion is computed as positive if there are more replies to the message that originated the case predicted to be positive by the CBR; otherwise it is negative. The solution is created either by using the solution of a case in the case base that matches the new case in the retrieve step or by using the combined value of sentiment and stress values detected from the four analyzer agents of the MAS. If there is no matching, a negative solution or value of 1 is associated with the case if the sentiment from the text sentiment analyzer is negative and the stress level from the text stress analyzer is high stress or if the sentiment from the keystroke dynamics sentiment analyzer is negative and the stress level from the keystroke dynamics stress analyzer agent is high stress. Otherwise, the value is 0. Fig. 5 shows a diagram of the functioning of the CBR module, with different software agents, actors, and possible actions.

### 3.2. Retrieve step

The retrieve step is performed by the CBR query solver agent. In this step, cases are retrieved from the case base, and the similarity between the cases retrieved and the new case created representing the message sent to the system is computed. The CBR query solver agent is in charge of the process of generating a case from the information related to a text message. The agent also extracts cases from the case base and finds the one that best matches the case generated from the message data and stores information about potential new cases to be added by the CBR updater agent.

After the case that represents the message being written in the SNS is created, the CBR query solver agent initiates a loop, checking cases in the case base (from the highest priority to the lowest) but with a limitation on the number of cases that can be checked (which is fixed). Nevertheless, the loop could halt if a certain number of cases matching the case generated from the message are found, which is also fixed. Those amounts were set to 100 for the maximum number of cases to be checked and 10 for the maximum number of cases to be selected. To assess whether or not a case matches the case generated from the message, and to what degree, two functions were coded. One function checks for the relevance of the case to be compared (which means that the case has at least bare minimum similarities with the case from the message). The second function computes the degree of matching between the two cases, checking the similarities between each defining feature in the cases (e.g., the topics from the texts). This function uses a weighted sum of the computed similarities of the features to compute the final degree of matching. First, the relevance is checked. Then, if the case to be compared is relevant (it has bare minimum similarities to the case generated from the message), the degree of matching between the two cases is computed. At the end of the process, the case that obtained the highest degree of matching is selected.

The relevance function process is illustrated in Algorithm 1 and works as follows: A loop is run for all of the features that could exist in a case. Every feature is checked to see if it is in the original message. If so, the corresponding feature is added to a counter of features to be matched (feature counter). Afterwards, a loop is run for every feature found in the original message in the previous step. In every iteration, if the feature is found in the new message to be compared to the original message, it is checked to compare the features of both cases, using comparisons based on the data type. The similarity between two string features is evaluated using the ratio of similarity from a SequenceMatcher object of the difflib python library, which is a float in the range [0, 1]. This value is computed as follows:

$$ratio = \frac{2 * M}{T}$$

where $M$ is the number of matches found between the strings, and $T$ is the total number of characters from both sequences. If the comparison

[2] https://radimrehurek.com/gensim/.

results in a basic match (the features are similar by at least 50%), then 1 is added to a counter of similarities found (similarity counter). Finally, if the variable accounting for those similarities is equal to or greater than a third of the number of features in the feature counter that is computed in the first loop, then the result is that the message is relevant. Otherwise, it is not relevant to the original message to which it is compared. The degree of matching function is illustrated in Algorithm 2 and works in the same way as the relevance function, except for two differences. First, it checks any existing differences in the features between cases and adds a percentage of the weight of the feature being compared (corresponding to the degree of matching of the features) to the similarity counter. Second, it later provides the quotient between the similarity counter and the value of a counter of the sum of the weights of the features to match.

```
Data: Case A and case B to compare with A
Result: Relevance of the level of similarity of B to A
sum_relevance = 0;
features_found = [];
for all the possible features of cases do
    if feature exists in A then
        Append feature to features_found;
    end
end
for each feature in features_found do
    if feature exists in B then
        Compare the value of the feature in A and B;
        if feature is integer and feature from A is equal than feature
         from B then
            sum_relevance += 1;
        else
            if feature is floating point number and absolute difference
             is 0.5 or less then
                sum_relevance += 1;
            else
                if feature is string and feature from A is 50% or more
                 similar to feature from B then
                    sum_relevance += 1;
                end
            end
        end
    end
end
if sum_relevance >= round(length(features_found) / 3) then
    Return B similarity to A is relevant;
else
    Return B similarity to A is not relevant;
end
```

**Algorithm 1:** Relevance function

### 3.3. Reuse step

In the reuse step, the solution associated with the case selected in the previous step is used for the system in order to provide an answer about whether or not the message sent by a user in a SNS could generate a negative repercussion in the network or social environment. When the CBR query solver agent has selected the case, then the information about the solution assigned to it (which is the prediction of the CBR module for whether or not the case will generate a negative repercussion) is taken as the prediction for the new case associated to the message being written. This information is sent to the Persistence and Advisor agents. If the prediction is negative, then a warning is generated in the Advisor agent and sent to the Presentation agent for the feedback to be delivered to the user. The Persistence agent simply stores the information about the prediction in the database.

```
Data: Case A and case B to compare with A
Result: Matching degree of B with A
sum_matching_degree = 0;
features_found = [];
sum_weights = 0;
for all the possible features of cases do
    if feature exists in A then
        Append feature to features_found;
        sum_weights += weight of feature;
    end
end
for each feature in features_found do
    if feature exists in B then
        Compare the value of the feature in A and B;
        Add to sum_matching_degree the fraction of the feature
         weight corresponding to the percentage of similarity
         found between this feature from case A and from case B;
    end
end
Return sum_matching_degree / sum_weights;
```

**Algorithm 2:** Matching degree function

The CBR query solver agent also stores new potential cases that could be added later to the case base by the CBR updater agent. The cases generated from new messages are used for this task. If the degree of matching of new cases is below a threshold, meaning that the new case is different from the cases in the case base by at least the percentage of the threshold (a value between 0 and 1), then they are added as potential new cases.

### 3.4. Revise step

In the revise step, the update of the state of the CBR module is performed to fit the ever-changing state of a real-life scenario in which the system is supposed to be used. The process performed in this step is the adaptation of existing cases in the case base, based on what the system observes and compares with its own previous predictions. To adapt the cases, the priorities assigned to them are modified. These priorities measure how well the cases have performed when used for prediction and thus if they are likely to be useful for new predictions. The CBR updater agent has a set time interval between updates, so the cases in the case base are not updated with every interaction on the SNS, but rather with the interactions that happened in that interval. This potentially leads to more useful information from the repercussion of messages. In other words, one message with only one reply gives the repercussion to one message; however if there are several answers, the agent can compute the repercussion to several messages, which may be more informative for updating the priority of cases. The update of priorities is based on the computed real repercussion of messages that have been predicted to have a positive or negative repercussion by the CBR query solver agent. Initially, the priorities of cases are set to zero.

The CBR updater agent performs the case-update loop as shown in Algorithm 3. This agent runs a loop for all of the cases that were selected by the CBR query solver agent. For every case, another loop is run for every message that matched it and was given a prediction based on the solution of the selected case, checking the repercussion of the message to see if it is the same as the predicted value. If it coincides, the priority of the case used for prediction is raised; if it does not coincide, then the priority is decreased. At a fixed low priority limit, the cases are also erased. To check the repercussion of the messages, the MAS has data about the messages and the parent structure of the messages. Therefore, the CBR module agents have the information about the parents and the children of messages. Finally, the repercussion is computed as the most frequently predicted value by the

```
Data: Information about predictions from the CBR module and
      parent structure of messages
Result: Update of cases in the case base
for each case C in the case base used to predict in the previous interval
 between updates do
    for each prediction P done with C for a case D, performed in the
     previous interval between updates do
        Compute the most predominant predicted value for the
         children (replies) of the message associated with D;
        Compare the computed predominant value with P;
        if the predominant value coincides with P then
            Increase the priority of C by 1;
        else
            Decrease the priority of C by 1;
            if priority is under a set threshold then
                Delete C from the case base;
                break;
            end
        end
    end
end
Save the updated cases in the case base;
```

**Algorithm 3:** Update of cases in the case base

CBR module in the replies of a given message. The information about the cases selected, the predictions performed, and the parent structure of messages is first stored by the CBR query solver agent when cases are created and selected; it is then used by the CBR updater agent when it is time to update the case base.

### 3.5. Retain step

As mentioned above, the CBR query solver stores information about potential new cases, which are generated when messages are sent to the MAS, and cases are created. For this task, the degree of matching of the new case with the cases in the case base is used. If the degree of matching of new cases is below a threshold (between 0 and 1), they are added as potential new cases.

The CBR updater agent updates the case base with the messages that were assigned as potential new cases by the CBR query solver agent. If the limit of messages in the case base (which is a fixed number) is reached, then no additional cases get added, and they remain as pending cases until the existing cases start to get erased by reaching a low priority limit in the previous step.

### 3.6. Example of the functionality of the system

To better understand how the system works, consider the following practical scenario. Users interact with each other on a SNS or in other on-line social environments that are connected to the proposed MAS and publish a post on the walls of the network or in a group. Before actually publishing the message, the information about the audience of the message, the user writing the message, the text, and the keystroke dynamics data of the message are sent to the MAS, where the Presentation agent receives this data. The Presentation agent sends messages to the Sentiment analyzers, the Stress analyzers, and the CBR query solver agent so that the analyzers can analyze the text and keystroke data and give the Advisor agent the outputs of their respective analyses. The Advisor agent gives the results of the analyses to the CBR query solver agent, which uses this information and the information from the Presentation agent to build a case representing the message being written on the SNS. The CBR query solver agent then computes the relevance. If it is relevant, it computes the degree of matching of the new case with cases in the case base and selects the best fitting case in order to give a prediction of positive or negative repercussions in the network of the message being written. Finally, the CBR query solver agent hands the prediction back to the Advisor agent. If it is a prediction with a negative value, a warning is generated and sent to the user who is interacting on the SNS in order to prevent potentially negative outcomes. Nevertheless, the user can choose to ignore the message and continue posting. A new case may get added (from the case created with the data associated with the message written on the SNS), and the priorities are updated in the case base if the repercussions on the SNS show that the predictions made were correct or incorrect, raising or decreasing the priority of the cases, respectively. Information about predictions, sentiment polarities, stress levels, and messages are also stored in the MAS database.

## 4. Experiments with data from Pesedia and the CBR module

There are two main objectives of these experiments. First, to fine-tune the CBR module by determining what the most important features are in the cases and the best configuration for the CBR module to populate the case base and achieve a low error rate in the prediction of the module. Second, to demonstrate the following hypothesis: By using information about different aspects of the users' affective state and context information together in a CBR engine, a system is able to predict negative repercussions in an on-line social environment such as a SNS better than affective state detection methods. Therefore, the two experiments performed with data from our private SNS Pesedia are discussed in this section.

The dataset used is the Pesedia dataset. It contains text messages, associated keystroke dynamics data, and other details of the text messages that are necessary for constructing the cases in the CBR module. The dataset was constructed by gathering data from the Pesedia SNS in July of 2018 and July of 2019, for a period of one month each time. Nevertheless, the number of samples gathered in 2019 is larger than the number of samples gathered in 2018. Additionally, only those gathered in 2018 were labeled with sentiment polarity and stress level (positive/negative and low/high stress level, respectively). There are 1609 samples from 2019 and 302 from 2018 in total.

Common affective state analysis datasets like the Standford Sentiment Treebank (Socher, Bauer, Manning, & Ng, 2013; Socher, Perelygin et al., 2013) consist of one data type and labels for a specific affective state (in our case, text and sentiment). Another example is the IMDB movie review dataset (Maas et al., 2011), which is a binary sentiment analysis dataset that consists of reviews from the Internet Movie Database (IMDb) labeled as positive or negative. In contrast, the Pesedia dataset contains both text and keystroke dynamics data of user messages, other information related to user interaction on the SNS, and also sentiment and stress level labels. This allows us to create cases that contain different information about affective states of the user and other context information that is derived from single user interactions in the social environment. This enables the creation of cases with different information related to interactions on SNSs and the comparison of predictions derived from the CBR and from other analysis methods. To the best of our knowledge, a dataset that combines text, keystroke dynamics data, context information from interactions among users in social networks, and also sentiment and stress levels labels does not exist. Examples of these datasets can be found in Table 2.

### 4.1. Experiments for fine-tuning the CBR module

In this section, the experiments performed altering parameters of the CBR module before populating the case base and checking the error rate of the module are examined. Specifically, the different configurations used and the process followed for checking the error of prediction of the module are described and the results presented. We altered the following parameters of the CBR module:

**Table 2**
Comparison between common datasets and the Pesedia dataset. The samples from the IMDB dataset are fragments of the movie review found in the dataset.

| Dataset | Label or labels type | Sample | Label or labels |
|---|---|---|---|
| Pesedia | sentiment (0 or 1 for positive or negative), stress level (0 or 1 for low or high stress level), time of the day, user ID, audience IDs | Pobre chaval, ¿por qué llora? | 1, 0, 11, 1050, (740,91,1050,361) |
| | | tanto trabajar no se puede eh! | 1, 1, 10, 2503, (872,91,2503,345) |
| Standford Sentiment Treebank | sentiment (0 most negative, 1 most positive) | opera that leaves no heartstring untugged and no liberal cause | 0.5 |
| | | ' I know how to suffer ' and if you see this film you 'll know too . | 0.22222 |
| IMDB movie reviews | sentiment, positive or negative | I have to say that this is one of the best movies I have ever seen! | Positive |
| | | In my opinion, these are two great actors giving horrible performances, | Negative |

- Weights of the different features in the cases. The weights associated with the case features (e.g., sentiment polarity, topic of the text) have been changed to assess whether the system learns to predict better when giving different levels of importance to the distinct features in the cases.
- Update interval for the CBR updater agent. The time interval between updates determines how much time we allow the interactions to occur in the system before the CBR module uses the information about repercussions and potential new cases to update its case base. We measure the differences in error rate when populating the case base at different update intervals.

The experimental procedure is detailed below. The Pesedia dataset was used for this process. A loop runs, processing an unlabeled user reply and the unlabeled original message that was replied to by that message, and then a labeled reply message in every iteration. The unlabeled messages are used to feed the CBR module, so that it can process them and later update the case base according to this information. Afterwards, the labeled message is also sent to the CBR module for processing, and the answer of the CBR module is kept for computing the error of the system. Finally, a comparison is made between the label of the message and the response of the CBR module. If it is the same, then a counter of correctly analyzed messages is increased. The ratio of correctly predicted messages by the CBR module is shown and stored every iteration in order to be able to draw conclusions later. Messages are sent as a set of features to the CBR module (e.g., text of the message, ID of the author), and the module creates an associated case by using the output of the Sentiment and Stress analyzers, a topic model, and other information related to the messages. Therefore, there is no need to provide labels to the CBR module, and both unlabeled and labeled messages can be provided to the CBR module for this experiment.

To examine the error rate of the module in the process of populating the case base with different configurations while it is being used for predictions, we define a metric. This metric is the error of the system in a window of messages (Windowed Error or WE). The error of the system is computed for several windows of time where the system analyzes a set number of messages. The $WE_i$ or windowed error in the window of messages $i$ is computed as follows:

$$WE_i = \sum_{j=c*i}^{c*i+c} e_j$$

where $e_j$ is 1 if the module detected a different state than the label on the labeled message at iteration $j$; otherwise, it is 0 (being the message used in iteration $j$ the message $j$), $i = 0, 1, 2, \ldots, n$, where n is the number of windows of messages for which we compute an error minus one, and c is a fixed constant for the size of the window. In our case, we used 25 messages for each window. The windowed error is meant to be a measure of errors found in a fixed set of interactions between a SNS and the CBR module. These interactions correspond to a fixed set of text messages that is published and sent to the CBR to be able to analyze differences found at different stages of the process (e.g., when the case base is at the initial state, intermediate states, and the last state).

Several experiments were conducted using different configurations of weights in the case features. Since the different possibilities of weight configurations are infinite, we created a selection of weight settings, aiming to test the configurations that led to the most informative results about the performance of the CBR module. For each experiment, different update intervals for the CBR module update were tested (between 10 and 100 s). The configurations for each experiment were the following:

- Experiment 1 (10_sen_str_day_and_history): equal weight for sentiment and stress features in the cases (10); small value in the weights of time of the day and history of the audience features (3); history of the author of the message (5) and topic of the message (10) (the weights of which remain unaltered).
- Experiment 2 (10_sen_str_no_day_and_history): the same as the first experiment, except no value is added to the weights of time of the day and history of the audience features (0).
- Experiment 3 (20_sen_day_and_history): doubled value of the weight representing sentiment than the weight representing stress on the message (20 and 10, respectively), and the rest of the weights unaltered with respect to the first experiment.
- Experiment 4 (20_sen_no_day_and_history): the same as the third experiment, except no value is added to the weights of time of the day and history of the audience features (0).
- Experiment 5 (20_str_day_and_history): doubled value of the weight representing stress than the weight representing sentiment on the message (20 and 10, respectively), and the rest of the weights unaltered with respect to the first experiment.
- Experiment 6 (20_str_no_day_and_history): the same as the fifth experiment, except no value is added to the weights of time of the day and history of the audience features (0).

Fig. 6 shows the results for the experiments keeping the same weight in the sentiment and stress features while changing others. Figs. 6(a), 6(c), and 6(e) show the results for the 10_sen_str_day_and_history experiment with 10, 50, and 100 s of update interval in the CBR module. Figs. 6(b), 6(d), and 6(f) show the results for the 10_sen_str_no_day_and_history experiment. For visual comparison, the figures with a certain update interval in which changes are made to the time and day and history of the audience features are followed by the figures with the same update interval but no changes in those features. Fig. 7 shows the results for the experiments doubling the weight of the sentiment feature and changing other features. Fig. 8 shows the results of the experiments doubling the stress feature weight while again changing other features.

An analysis of the experimental results is presented below. As can be observed, the error obtained by the CBR module in the 10_sen_str_day_and_history experiment is low in general; there is only a small increase in the initial parts of the experiment with the largest update interval. Nevertheless, when the effect of the time of the day and history of the audience features is removed (weights set to zero) in the 10_sen_str_no_day_and_history experiment while sharing the same weights on the other features as in the previous case, there is still a small error in general. However, the error that in the previous experiment appeared with the largest update interval appears earlier at the 50 s update interval and stays in the 100 s update interval.

In contrast, in the experiments where the weights for sentiment and stress features of the cases are altered, a general trend of more errors is found in general, being higher in the case when the sentiment weight is considered to be twice as important as the stress weight. In

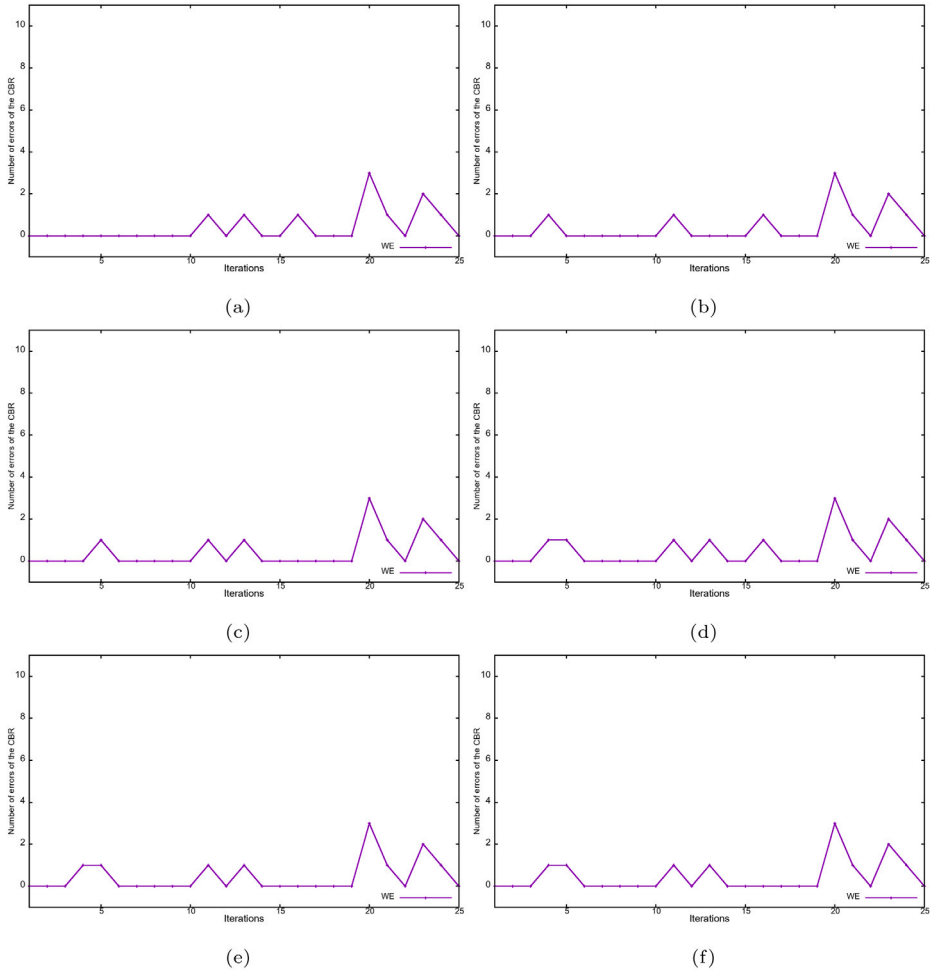


**Fig. 6.** Error in the '10_sen_str_day_and_history' experiment and in the '10_sen_str_no_day_and_history' experiment (left and right columns, respectively). From top to bottom in increasing update intervals of 10, 50, and 100 s.

the case of the 20_sen_day_and_history and 20_sen_no_day_and_history experiments, the error gets progressively and visibly smaller as the frequency of updates of the CBR module increases. Moreover, in the case of these two experiments, setting the weight of the time of the day and the history of the audience features to zero appears to reduce the error to some extent.

In the case of the 20_str_day_and_history and 20_str_no_day_and_history experiments, the effect of the time of the day and history of the audience features is the same as in the experiments where the weights for the sentiment and stress features are the same. In other words, there are more errors in the case where these two features are set to zero. Moreover, in these two experiments, altering the update interval does not change the observed error.

To conclude, the error observed was lower when not altering the weights for the sentiment and stress features (both were the same). Additionally, in this experimentation, the error diminished when setting the update intervals to be more frequent (not causing it to increase in any of the experiments), demonstrating that the system is able to learn and achieve fewer errors with the settings used in the case of this dataset. The effect of the time of the day and history of the audience of the message features in general slightly reduced the errors, except in the experiments where the sentiment weight was considered to be twice as important as the stress weight. However, in these experiments, a high error was found, which might indicate that modifying the sentiment feature weight in this way is not ideal for reducing the error of the CBR module, as is the case when altering the weight for the stress feature in the same way. Finally, it can be observed that the messages from the message windows 19 to 24 created some errors in most of the cases, which might indicate that messages pertaining to the end of the dataset could be somewhat noisy.

### *4.2. Experiments for comparing the CBR module against different sentiment and stress analysis methods*

We propose the following hypothesis: Using information about different aspects of the users' affective state and context information together in a CBR engine for predicting positive and negative repercussions in an on-line social environment such as a SNS is more effective than using affective state detection methods. For this purpose, we performed experiments populating the case base with Pesedia data and then used a static version of the CBR module (without updates) to predict negative or positive repercussions caused by messages in the Pesedia SNS that were not the messages used for populating the case base in the first step. We also used different analyzers (individual sentiment and stress analysis and combined versions) to compare the results obtained between the CBR module and the different analyzers. The dataset used was the Pesedia dataset.

To compare the capacity to detect positive and negative repercussions on the SNS, we used the propagation of the detected state in the network as a metric. This metric measures the percentage of messages that were detected by an analysis method to have the same state as the ones directly influenced by it. In our study, we use the replies to a message as the messages that are directly influenced by this message (the message that was replied to). In this way, we are able to compute a

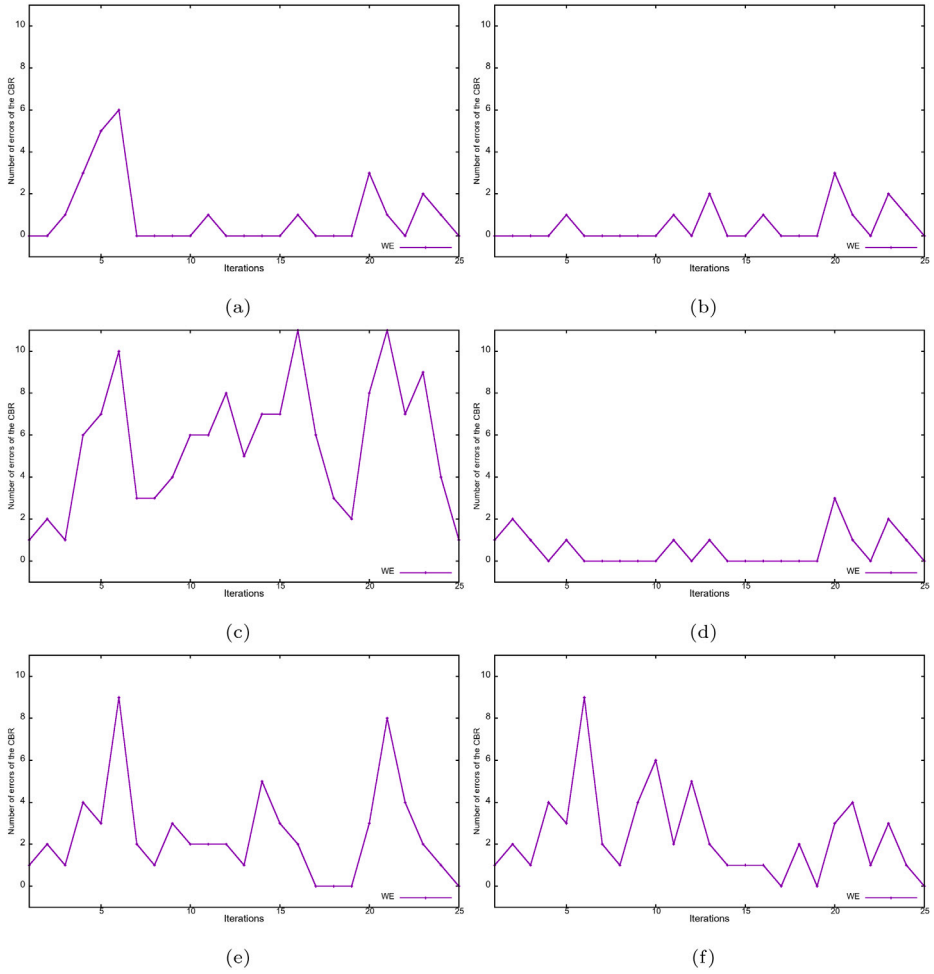


**Fig. 7.** Error in the '20_sen_day_and_history' experiment and in the '20_sen_no_day_and_history' experiment (left and right columns, respectively). From top to bottom in increasing update intervals of 10, 50, and 100 s.

metric that shows which analyzer is able to detect a state that is found more frequently in the messages influenced by the message analyzed. This helps the system prevent potentially negative outcomes (such as a negative state spreading through the network). The Propagation of Detected Value (PDV) metric is computed as follows:

$$PDV = \frac{messages\_with\_propagated\_state}{messages\_with\_replies}$$

where messages_with_replies is the total number of messages that generated replies being analyzed, and messages_with_propagated_state is the aggregated value of messages with the propagated state, which are messages with the same detected state as is present in most of their replies (the state most frequent of the states detected in their replies).

We used Sentiment and Stress analyzers on both text data and keystroke data. We used combined versions of Sentiment and Stress analyzers in text data using two versions (*or* and *and* combined analysis as a late fusion method of sentiment and stress analysis). We did the same for keystroke data. The *or* and *and* versions of combined analysis of sentiment and stress refer to the use of the union or intersection of the outputs of Sentiment and Stress analyzers, respectively. They were applied at the decision level (after the sentiment and stress analyses were computed). Thus, by using the *or* version of combined analysis, a negative class is assigned if either the sentiment polarity is negative or the stress level is high. Otherwise, the resulting class output is positive. For the case of the *and* version, a negative class is given as output when both negative sentiment and high stress levels are detected, and a positive class is given otherwise. We used our CBR module for prediction using optimized parameters. In these experiments, labels are not necessary. Therefore, the use of the labeled or unlabeled sets of samples for populating the case base and for comparing predictions is done to assess differences between using smaller or larger amounts of data when populating the case base and comparing predictions (since the sets of samples are different in size). For the experimental setup, we performed experiments with the following data:

- smaller_data: use of the labeled samples to populate the case base and the unlabeled samples to compare predictions.
- larger_data: use of the unlabeled samples to populate the case base and the labeled samples to compare predictions.
- larger_third: use of the first third of the unlabeled samples to populate the case base and the rest of the unlabeled samples to compare predictions.
- larger_half: use of the first half of the unlabeled samples to populate the case base and the rest of the unlabeled samples to compare predictions.
- larger_third_and_smaller: use of the first third of the unlabeled samples and all of the labeled samples to populate the case base and the rest of the unlabeled samples to compare predictions.
- larger_half_and_smaller: use of the first half of the unlabeled samples and all of the labeled samples to populate the case base and the rest of the unlabeled samples to compare predictions.

In Table 3, we show the results of PDV for different analyzers and the CBR module. For each experiment, we present two rows of results in the table. The results for analyzers working with text input are

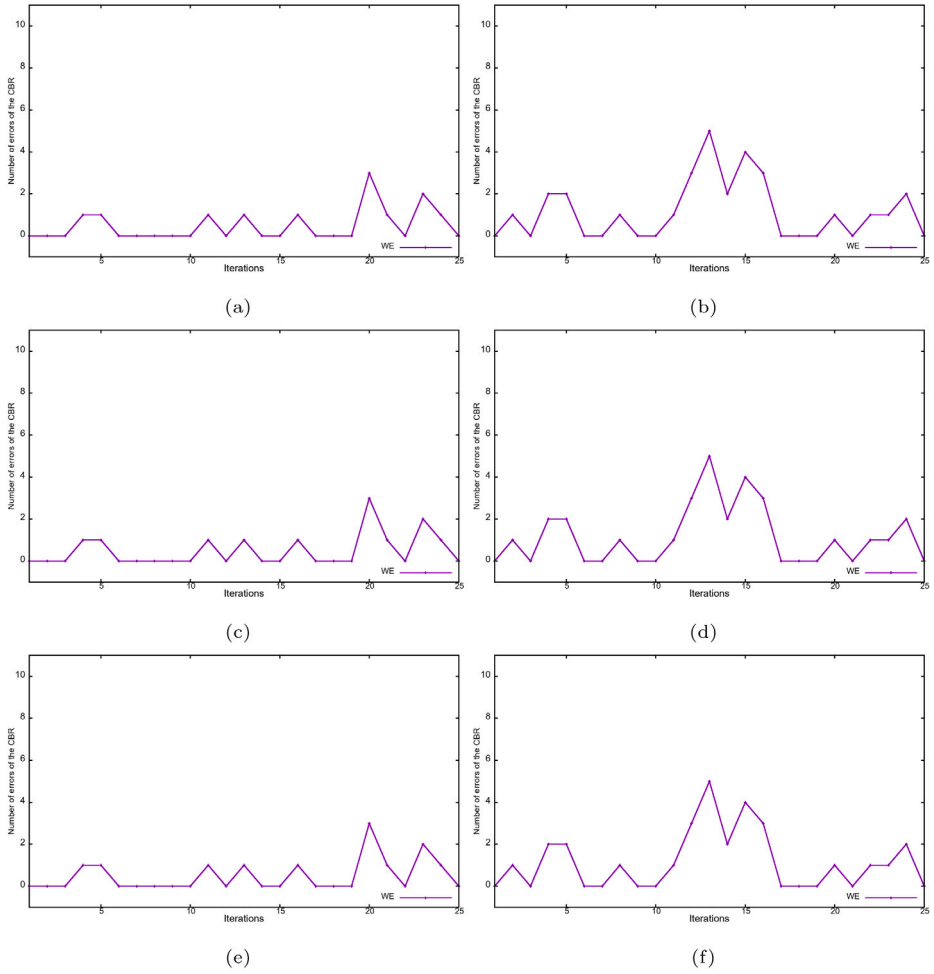


**Fig. 8.** Error in the '20_str_day_and_history' experiment and in the '20_str_no_day_and_history' experiment (left and right columns, respectively). From top to bottom in increasing update intervals of 10, 50, and 100 s.

shown in the upper row, and the results for analyzers working with keystroke dynamics data are shown in the lower row. Finally, since the CBR module utilizes information from both analyses on text data and keystroke data, only one cell per experiment is presented. Each result shown in Table 3 is the average of four different experiments performed on the four different partitions of data in the test set of samples used. To do this, we partitioned the corresponding test set and used one partition for each experiment. We show the average of the four results in the table.

Finally, we present an analysis of the experimental results for the comparison of the CBR module and other analysis methods. As Table 3 shows, there are differences between the results obtained by the different analyzers and the CBR module. The different experiments were conducted with the aim of exploring whether the CBR module was able to obtain better results in terms of PDV than the non-CBR-based Sentiment, Stress, and Combined analyzers (on text data and on keystroke dynamics data). We performed experiments populating the case base with different data samples and different amounts of samples to ascertain whether or not the factors of using different data or different data sizes influences the results. As Table 3 shows, the CBR module was able to outperform the different non-CBR-based analyzers in almost every experiment, except the case of the smaller_data experiment, where the case base was populated using only the labeled data samples. In this case, when using a small number of samples, the CBR module performance is similar to the Sentiment analyzer using text and the *or* Combined analyzer of sentiment and stress on text data, but its performance is lower than the rest of the analyzers. As commented above, the results are not a consequence of using labeled samples since labels were not used in this experiment. Nevertheless, the CBR module is able to outperform every analyzer when populating the case base with amounts of data such as the case of the larger_data experiment, and the performance was not affected by using or mixing different partitions of data samples when populating the case base. Moreover, using only a third of the unlabeled data samples to populate the case base and the remaining two thirds to test performance proved to be enough to not decrease the performance of the CBR module.

## 5. Conclusions and future lines of work

In this work, different Sentiment and Stress analyzers using text and keystroke dynamics data have been combined using a CBR module. They have been integrated into a MAS for guiding and making recommendations to users who navigate on on-line social platforms or in environments based on their emotional state and stress levels. The sentiment and stress analyses have been implemented in individual agents that perform sentiment and stress analysis on text data and on keystroke data. These agents communicate with other agents in a MAS in order to be able to receive data from messages of users on on-line social platforms in real-time, analyze the data, and hand it over to a CBR module. This module uses the information of the output of the analyses and context of the conversations to predict potentially negative outcomes derived from the user interaction. Since the different

**Table 3**
Comparison between analyzers and the CBR module.

| Experiment | sentiment analysis | stress analysis | or combined analysis | and combined analysis | CBR module |
|---|---|---|---|---|---|
| smaller_data | 0.7055<br>0.9183 | 0.9584<br>0.9995 | 0.6947<br>0.9179 | 0.9895<br>1 | 0.7395 |
| larger_data | 0.6603<br>0.7638 | 0.8135<br>0.9342 | 0.6703<br>0.756 | 0.9413<br>0.9559 | 1 |
| larger_third | 0.6842<br>0.9173 | 0.9499<br>0.9993 | 0.6671<br>0.9167 | 0.9899<br>1 | 1 |
| larger_half | 0.686<br>0.9156 | 0.9475<br>0.9992 | 0.6666<br>0.9149 | 0.9897<br>1 | 1 |
| larger_third_and_smaller | 0.6842<br>0.9173 | 0.9499<br>0.9993 | 0.6671<br>0.9167 | 0.9899<br>1 | 1 |
| larger_half_and_smaller | 0.686<br>0.9156 | 0.9475<br>0.9992 | 0.6666<br>0.9149 | 0.9897<br>1 | 1 |

functions are implemented in several agents in the MAS, the tasks of the system can be parallelized to work in real-time scenarios.

The CBR approach allows using information about user interaction and user state in order to perform predictions on on-line platforms and to recommend actions to users. Context information such as the topic being discussed and the history of predictions of the system of the user writing a message and the users in the audience are used together with sentiment polarity and stress level detected in text and keystroke dynamics data of messages. The CBR module successfully predicts potentially negative repercussions (as in negative sentiment polarity or high stress levels spreading through users) on an on-line social platform when users write messages, using the information described above.

To assess whether the CBR module that uses information from aspects of the users' affective state and context information was more effective in predicting negative repercussions than affective state detection methods, experiments were conducted with Pesedia data. Several experiments were conducted, changing the amount of data that was fed to the CBR module for populating the case base and for testing the performance. The CBR module managed to outperform the different analyzers, except in the case of an experiment where the case base was populated using a small partition of the data from the dataset. In this case, the performance of the CBR module was similar to that of the Sentiment analyzer using text and the *or* Combined analyzer of sentiment and stress on text data; however, it was lower than the other analyzers. Experiments for fine-tuning the CBR module before testing the differences between the module and affective state detection methods were also conducted with Pesedia data. The aim of these experiments was to assess the error of the system when making predictions after populating the case base with different configurations of weights in the case features and using different update intervals. For these experiments, a labeled corpus of data from Pesedia was used and compared to the prediction of the CBR module by conducting several experiments varying weights and update intervals. The experiments showed that, with the configurations used and Pesedia data, shorter update intervals lead to fewer errors from the CBR module and that certain configurations in the weights of the case features lead to fewer errors than other configurations.

For future lines of work, there are many unexplored possibilities. First, new features could be introduced in the cases in order to account for additional information to the system when making predictions. One example of a feature that might be useful is the level of tiredness of users, which would give more information to the system about the real-time state of users when interacting and which might lead to better performance. Machine learning techniques could be used to measure the level of tiredness in users, using text data, keystroke data, or both. In addition, a second potentially interesting line of work would be to use the system for not only predicting negative sentiment and high level of stress spread throughout an on-line social environment, but also to use it to predict other phenomena that could be of interest to the system for guiding and making recommendations to users, such as predicting cyber-bullying or on-line grooming. Moreover, the system could be used to give different feedback to users. It could be used to warn a user that his or her message might be inappropriate if the users in the audience of the message have a recent history of negative states detected by the system based on the output of the CBR module. Finally, although the feed-forward ANN architecture is used in this version of the system, in the future, other network architectures could be integrated and tested.

## CRediT authorship contribution statement

**G. Aguado:** Investigation, Conceptualization, Methodology, Software, Writing – original draft, Data curation, Writing – reviewing & editing. **V. Julian:** Supervision, Validation, Writing – reviewing & editing. **A. Garcia-Fornes:** Supervision, Validation, Writing – reviewing & editing. **A. Espinosa:** Supervision, Validation, Writing – reviewing & editing.

## Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

## Data availability

The authors do not have permission to share data.

## Acknowledgments

This work was financed by the project TIN2017-89156-R of the Ministry of Economy, Industry and Competitiveness, Government of Spain.

## References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, *7*(1), 39–59. http://dx.doi.org/10.3233/AIC-1994-7104.

Aguado, G., Julian, V., Garcia-Fornes, A., & Espinosa, A. (2020a). A multi-agent system for guiding users in on-line social environments. *Engineering Applications of Artificial Intelligence*, *94*, Article 103740. http://dx.doi.org/10.1016/j.engappai.2020.103740.

Aguado, G., Julián, V., García-Fornes, A., & Espinosa, A. (2020b). Using keystroke dynamics in a multi-agent system for user guiding in online social networks. *Applied Sciences*, *10*(11), 3754. http://dx.doi.org/10.3390/app10113754.

Akhtar, M. S., Ekbal, A., & Cambria, E. (2020). How intense are you? Predicting intensities of emotions and sentiments using stacked ensemble [application notes]. *IEEE Computational Intelligence Magazine*, *15*(1), 64–75. http://dx.doi.org/10.1109/MCI.2019.2954667.

Basiri, M. E., Nemati, S., Abdar, M., Cambria, E., & Acharya, U. R. (2021). ABCDM: An attention-based bidirectional CNN-RNN deep model for sentiment analysis. *Future Generation Computer Systems*, *115*, 279–294. http://dx.doi.org/10.1016/j.future.2020.08.005.

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent dirichlet allocation. *Journal of Machine Learning Research*, *3*, 993–1022.

Bordera, J. (2016). *PESEDIA. Red social para concienciar en privacidad*. Valencia, Spain: Universitat Politècnica de València.

Bradley, M. M., & Lang, P. J. (2007). *The international affective digitized sounds (2nd edition; IADS-2): Affective ratings of sounds and instruction manual: Tech. Rep. B-3*, Gainesville, FL: University of Florida.

Bridge, D., & Healy, P. (2012). The GhostWriter-2.0 case-based reasoning system for making content suggestions to the authors of product reviews. *Knowledge-Based Systems*, *29*, 93–103. http://dx.doi.org/10.1016/j.knosys.2011.06.024.

Camacho, D., Panizo-LLedot, A., Bello-Orgaz, G., Gonzalez-Pardo, A., & Cambria, E. (2020). The four dimensions of social network analysis: An overview of research methods, applications, and software tools. *Information Fusion*, *63*, 88–120. http://dx.doi.org/10.1016/j.inffus.2020.05.009.

Cambria, E., Li, Y., Xing, F. Z., Poria, S., & Kwok, K. (2020). SenticNet 6: Ensemble application of symbolic and subsymbolic AI for sentiment analysis. In *Proceedings of the 29th ACM international conference on information & knowledge management* (pp. 105–114). Association for Computing Machinery, http://dx.doi.org/10.1145/3340531.3412003.

Cardellino, C. (2019). Spanish billion words corpus and embeddings. URL https://crscardellino.github.io/SBWCE/ (Accessed 6 March 2021).

Ceci, F., Goncalves, A. L., & Weber, R. (2016). A model for sentiment analysis based on ontology and cases. *IEEE Latin America Transactions*, *14*(11), 4560–4566. http://dx.doi.org/10.1109/TLA.2016.7795829.

De Moor, S., Dock, M., Gallez, S., Lenaerts, S., Scholler, C., & Vleugels, C. (2008). Teens and ICT: Risks and opportunities. Belgium: TIRO. http://www.belspo.be/belspo/fedra/proj.asp?l=en&COD='TA/00/08 (Accessed 6 March 2021).

Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., & Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, *41*(6), 391–407. http://dx.doi.org/10.1002/(SICI)1097-4571(199009)41:6<391::AID-ASI1>3.0.CO;2-9.

Feldman, R. (2013). Techniques and applications for sentiment analysis. *Communications of the ACM*, *56*(4), 82–89. http://dx.doi.org/10.1145/2436256.2436274.

George, J. M., & Dane, E. (2016). Affect, emotion, and decision making. *Organizational Behavior and Human Decision Processes*, *136*, 47–55. http://dx.doi.org/10.1016/j.obhdp.2016.06.004.

Grossi, E., & Buscema, M. (2007). Introduction to artificial neural networks. *European Journal of Gastroenterology & Hepatology*, *19*, 1046–1054. http://dx.doi.org/10.1097/MEG.0b013e3282f198a0.

Heras, S., García-Pardo, J. A., Ramos-Garijo, R., Palomares, A., Botti, V., Rebollo, M., et al. (2009). Multi-domain case-based module for customer support. *Expert Systems with Applications*, *36*(3), 6866–6873. http://dx.doi.org/10.1016/j.eswa.2008.08.003.

Hu, M., & Liu, B. (2004). Mining opinion features in customer reviews. In *Proceedings of the 19th national conference on artifical intelligence* (pp. 755–760). Springer.

Huang, F., Zhang, X., Zhao, Z., Xu, J., & Li, Z. (2019). Image–text sentiment analysis via deep multimodal attentive fusion. *Knowledge-Based Systems*, *167*, 26–37. http://dx.doi.org/10.1016/j.knosys.2019.01.019.

Jakob, N., & Gurevych, I. (2010). Extracting opinion targets in a single-and cross-domain setting with conditional random fields. In *Proceedings of the 2010 conference on empirical methods in natural language processing* (pp. 1035–1045). Association for Computational Linguistics.

Kingma, D. P., & Ba, J. (2014). Adam: A method for stochastic optimization. URL http://arxiv.org/abs/1412.6980 cite arxiv:1412.6980Comment: Published as a conference paper at the 3rd International Conference for Learning Representations, San Diego, 2015.

Kolodner, J. (1993). *Case-based reasoning*. Morgan Kaufmann.

Lee, P.-M., Tsui, W.-H., & Hsiao, T.-C. (2015). The influence of emotion on keyboard typing: an experimental study using auditory stimuli. *PLoS One*, *10*(6), Article e0129056. http://dx.doi.org/10.1371/journal.pone.0129056.

Li, F., Han, C., Huang, M., Zhu, X., Xia, Y.-J., Zhang, S., et al. (2010). Structure-aware review mining and summarization. In *Proceedings of the 23rd international conference on computational linguistics* (pp. 653–661). Association for Computational Linguistics.

Liu, B. (2012). *Sentiment analysis and opinion mining*. Morgan & Claypool Publishers.

Livingstone, S., Haddon, L., Görzig, A., & Ólafsson, K. (2011). *Risks and safety on the internet: the perspective of European children: full findings and policy implications from the EU Kids Online survey of 9-16 year olds and their parents in 25 countries*. EU Kids Online, EU Kids Online, Deliverable D4. EU Kids Online Network, London, UK. http://eprints.lse.ac.uk/33731/ (Accessed 6 March 2021).

Maas, A. L., Daly, R. E., Pham, P. T., Huang, D., Ng, A. Y., & Potts, C. (2011). Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: human language technologies* (pp. 142–150). Portland, Oregon, USA: Association for Computational Linguistics, URL http://www.aclweb.org/anthology/P11-1015.

Marie, F., Corbat, L., Chaussy, Y., Delavelle, T., Henriet, J., & Lapayre, J.-C. (2019). Segmentation of deformed kidneys and nephroblastoma using case-based reasoning and convolutional neural network. *Expert Systems with Applications*, *127*, 282–294. http://dx.doi.org/10.1016/j.eswa.2019.03.010.

Muhammad, K., Lawlor, A., Rafter, R., & Smyth, B. (2015). Great explanations: Opinionated explanations for recommendations. In *International conference on case-based reasoning* (pp. 244–258). Springer, http://dx.doi.org/10.1007/978-3-319-24586-7_17.

Nasukawa, T., & Yi, J. (2003). Sentiment analysis: Capturing favorability using natural language processing. In *Proceedings of the 2nd international conference on knowledge capture* (pp. 70–77). ACM, http://dx.doi.org/10.1145/945645.945658.

Ohana, B., Delany, S. J., & Tierney, B. (2012). A case-based approach to cross domain sentiment classification. In *International conference on case-based reasoning* (pp. 284–296). Springer.

Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing* (pp. 1532–1543). Association for Computational Linguistics, http://dx.doi.org/10.3115/v1/D14-1162.

Poria, S., Chaturvedi, I., Cambria, E., & Hussain, A. (2016). Convolutional MKL based multimodal emotion recognition and sentiment analysis. In *2016 IEEE 16th international conference on data mining* (pp. 439–448). IEEE, http://dx.doi.org/10.1109/ICDM.2016.0055.

Rincon, J., de la Prieta, F., Zanardini, D., Julian, V., & Carrascosa, C. (2017). Influencing over people with a social emotional model. *Neurocomputing*, *231*(C), 47–54. http://dx.doi.org/10.1016/j.neucom.2016.03.107.

Schouten, K., & Frasincar, F. (2016). Survey on aspect-level sentiment analysis. *IEEE Transactions on Knowledge and Data Engineering*, *28*(3), 813–830. http://dx.doi.org/10.1109/TKDE.2015.2485209.

Seroussi, Y., Zukerman, I., & Bohnert, F. (2010). Collaborative inference of sentiments from texts. In *Proceedings of the 18th international conference on user modeling, adaptation, and personalization* (pp. 195–206). Springer-Verlag, http://dx.doi.org/10.1007/978-3-642-13470-8_19.

Socher, R., Bauer, J., Manning, C. D., & Ng, A. Y. (2013). Parsing with compositional vector grammars. In *Proceedings of the 51st annual meeting of the association for computational linguistics (volume 1: long papers)* (pp. 455–465).

Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C. D., Ng, A. Y., et al. (2013). Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 conference on empirical methods in natural language processing* (pp. 1631–1642).

Thelwall, M. (2017). TensiStrength: Stress and relaxation magnitude detection for social media texts. *Information Processing & Management*, *53*(1), 106–121. http://dx.doi.org/10.1016/j.ipm.2016.06.009.

Thelwall, M., Buckley, K., Paltoglou, G., Cai, D., & Kappas, A. (2010). Sentiment strength detection in short informal text. *Journal of the American Society for Information Science and Technology*, *61*(12), 2544–2558. http://dx.doi.org/10.1002/asi.21416.

Tian, F., Gao, P., Li, L., Zhang, W., Liang, H., Qian, Y., et al. (2014). Recognizing and regulating e-learners' emotions based on interactive Chinese texts in e-learning systems. *Knowledge-Based Systems*, *55*, 148–164. http://dx.doi.org/10.1016/j.knosys.2013.10.019.

Upadhyay, A., Chaudhari, A., Ghale, S., Pawar, S., et al. (2017). Detection and prevention measures for cyberbullying and online grooming. In *2017 international conference on inventive systems and control* (pp. 1–4). IEEE, http://dx.doi.org/10.1109/ICISC.2017.8068605.

Vandenhoven, E., Schellens, T., & Valacke, M. (2014). Educating teens about the risks on social network sites. *Media Educational Research Journal*, *43*(22), 123–131. http://dx.doi.org/10.3916/C43-2014-12.

Vanderhoven, E., Schellens, T., Vanderlinde, R., & Valcke, M. (2016). Developing educational materials about risks on social network sites: a design based research approach. *Educational Technology Research and Development*, *64*(3), 459–480. http://dx.doi.org/10.1007/s11423-015-9415-4.

Vizer, L. M., Zhou, L., & Sears, A. (2009). Automated stress detection using keystroke and linguistic features: An exploratory study. *International Journal of Human-Computer Studies*, *67*(10), 870–886. http://dx.doi.org/10.1016/j.ijhcs.2009.07.005.

Xie, W., & Kang, C. (2015). See you, see me: Teenagers' self-disclosure and regret of posting on social network site. *Computers in Human Behavior*, *52*(C), 398–407. http://dx.doi.org/10.1016/j.chb.2015.05.059.

Zhou, F., Jianxin Jiao, R., & Linsey, J. S. (2015). Latent customer needs elicitation by use case analogical reasoning from sentiment analysis of online product reviews. *Journal of Mechanical Design*, *137*(7), http://dx.doi.org/10.1115/1.4030159.