A CBR for integrating sentiment and stress analysis for user guiding in social network sites

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

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

Abstract

In this work, a Case-Based Reasoning (CBR) module is proposed, which integrates sentiment and stress analysis on text data and keystroke dynamics data, and context information from people navigating and interacting in Social Network Sites (SNSs). The context used is for example the history of positive or negative messages of the user, or the topics being talked about. The CBR module uses this data to generate useful feedback to the user navigating, giving him or her a warning if it detects a potential future negative repercussion in the SNS caused by the interaction of the user in the system. In this way, we aim to help create a more safe and satisfactory experience for users inside of SNSs or 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 a future problem or negative repercussion in the SNS. For this purpose, we use messages generated in a private SNS called Pesedia at the laboratory. In the experiments, the CBR module managed to outperform the other proposed analyzers in almost every case. The CBR module was fine-tuned, exploring its performance when populating the case base with different configurations.

Keywords: multi-agent system, social networks, sentiment analysis, stress

[☆]Fully documented templates are available in the elsarticle package on CTAN.

[*]Corresponding author

Email 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)

analysis, case-based reasoning

1. Introduction

In the daily environment in which people are immersed lately, there is a strong influence of on-line applications that are used to interact with others, obtain information, and perform different tasks. Tasks can be related to jobs, leisure, or other kinds of activities. Therefore, people are influenced by the information existing and being shared in on-line platforms and applications. Between the most important and preeminent on-line applications there are Social Network Sites (SNSs), which are used to interact with other people and communicate, offering a forum in which people, or users in this context, can post their information to be read by others. However, this kind of interaction is not free from risks. In (Vanderhoven et al., 2016) risks and negative outcomes from users navigating and interacting have been reviewed. Moreover, different risk types have been reviewed in (De Moor et al., 2008) and (Livingstone et al., 2011), 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, in (Vandenhoven et al., 2014) authors reported that teenagers face several risks while navigating SNSs, and have characteristics making them more vulnerable to such risks.

Users navigating on-line sites, unless guided by the system or other entity are on their own when interacting with other users. They have to make decisions, which for example can be about how to interact or whom to interact with. Therefore, the decision-making process drives their interaction, and if the decision-making of users is not well performed and 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 & Dane, 2016), where authors review the effect of incidental moods, discrete emotions, integral affect, and regret on decision-making. Incidental moods and discrete emotions refer to affective states not directly linked with the task at hand and that can be originated from other sources, like for example thinking of someone that is not directly linked with the task being performed. Contrarily, 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. In this review, authors reported that incidental moods, discrete emotions, integral affect, and regret affect decision making. Authors also showed that incidental moods do so by altering the perception of people and that regret affects decision making acting as anticipating regret, as in thinking of the negative outcome before it happens.

A system that guides the decision-making process of users interacting is, therefore, useful to avoid them from incurring risks and suffering their consequences, and since the decision-making process is influenced by emotion, the system could monitor the emotion of users for predicting potentially negative repercussions derived from the interaction. Additionally, stress has been associated with an emotional state (high arousal and negative valence) and has been used to construct an algorithm for detecting stress and relaxation magnitude in texts in (Thelwall, 2017). Therefore, it might be useful to incorporate a module specialized in detecting stress levels for a system that guides the decision process of users on on-line sites.

For achieving building a system that can analyze user affective states and guide users navigating, in (Aguado et al., 2020a) we presented a Multi-Agent System (MAS) as a system that collects messages from users that are interacting in a SNS or other social environment and computes sentiment, stress, and a combined analysis for potentially generating feedback to the user, as a warning that aimed to avoid future negative repercussions on the social environment. Keystroke dynamics are timing information and frequency of pulsation of keys that can be collected when a user is typing on a keyboard and can be used as an additional source of information for a data analysis application. In (Aguado et al., 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. Experiments were conducted with data from a private SNS called Pesedia (Bordera, 2016) for discovering which analyzers were more effective at detecting user states that propagated more in the SNS. Finally, an Advisor agent that generates feedback for users in 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 are not the only sources of information available from the SNS that could be used to generate feedback. Other sources, like the historic of polarities and stress levels of the user when he or she interacted in the past, the one of the audience where the message is about to be posted, or the topics that could be extracted from the message are examples of other sources of information that may prove to be useful at generating feedback to the user that may perform better than simple analysis on the messages at avoiding potentially negative outcomes in the social environment. In Case-Based Reasoning (CBR) systems, a reasoner remembers previous situations similar to the current one and uses them to solve the new problem (Kolodner, 1993). The cases can contain several different features to represent a concrete situation in the system, so a CBR system could be used to combine different aspects of a user state, and also external factors to help decide what action should the system take to guide this user and potentially prevent a negative outcome.

We implemented and integrated a CBR module into a MAS, for helping it detect a case in a SNS where a user interacting could generate a negative repercussion, and for making the system able to prevent it by applying the corresponding action to each case. In this way, the system is able to take into consideration different sources of information and also previous interactions, and exploit this information for guiding 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 in a SNS, which are the text of the message, the keystroke dynamics data associated, 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 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 integrated into the MAS generates a case with this information and calculates which case from its case base is the most similar to the current case to further recommend an action to be performed, such as warning the user to avoid potential future risky situations. This process is explained extensively in section 3.

Regarding the metrics and results, 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 a state of the users that propagated the most to the replies of the messages. As a metric of propagation, the most detected state in the replies of the messages was used. In the present work, we conducted a set of experiments for discovering not only which of the analyses is able to detect a state that propagates more 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 used this data to compare the analyses and the CBR module. We also performed experiments for analyzing 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.

Therefore, the contribution of this paper is to demonstrate that by using information about different aspects of the users' affective state and context information together, 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 for assessing what the differences are between the analyses and the CBR module in predicting a state of the user that propagates more in the SNS, so 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 by using the information of the history of analyses, information retrieved from the SNS, and analyses done at the moment, and that can recommend an action to prevent potentially negative repercussions in a SNS. This CBR-based approach is a way to use information related to the user state and context of the interactions to predict potentially negative outcomes in the SNS that, to the best of our knowledge, has not been performed before. Finally, we fine-tuned the CBR module by performing experiments varying its parameters and populating the case base for discovering which set of parameters achieved the lowest error rate of the predictions of the CBR module.

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

2. Related work

In the present work, we integrated a CBR module into a MAS that previously used sentiment analysis, stress analysis, and a combined version of sentiment and stress with text data and keystroke data, to help the system know what is the state of users navigating a SNS or other on-line social environments. In this way, the system is able to generate feedback for users to avoid a potential future negative situation in the social environment. For this reason, a brief review of state-of-art works in sentiment analysis and stress analysis on text and keystroke dynamics data is given following, as well as a revision 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.

Sentiment analysis is a research line that focuses on recognizing opinion, sentiments, evaluation, appraisal, attitude, and emotion in different media (e.g. texts, images, audio) (Liu, 2012). Depending on the level of fine-grained analysis that is performed, document level, sentence level, and aspect level sentiment analysis are identified. Document and sentence-level sentiment analysis are performed on a whole document or 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). Regarding document-level sentiment analysis, in (Basiri et al., 2021) an architecture is proposed 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 LSTM and GRU models to pay more or less emphasis on different words from the text input. Following, the semantic representations are passed to a Convolutional Neural Network (CNN) layer, and finally to a dense layer that outputs a sentiment polarity. Experiments performed comparing the proposed architecture against six recently published deep neural models for sentiment analysis on five review and three Twitter datasets showed that the proposed model outperforms the other six models in almost every case in terms of precision and F1, being the difference greater in the reviews datasets. Additionally, in (Akhtar et al., 2020) a Multi-Layer Perceptron (MLP) based stacked ensemble architecture for sentiment and emotion intensity is presented. It consists of four individual models (LSTM, CNN, GRU, and one feature-driven classical supervised model based on Support Vector Regression or SVR) 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 concrete emotions (joy and sadness). In (Cambria et al., 2020) SenticNet 6 is proposed as a long sentiment lexicon (composed of 200,000 words and multiword expressions). It was built using both sub-symbolic models, as in deep learning models (biLSTM and Bidirectional Encoder Representations from Transformers or BERT) to generalize words and multi-word expressions into primitives, which are later defined manually in terms of super-primitives, and symbolic models (logic and semantic networks) to extract meaning and assign sentiment polarity to high-level concepts. Authors compared SenticNet with 15 popular sentiment lexica, testing against six commonly used benchmarks for sentence-level sentiment analysis, resulting in that SenticNet 6 was the best-performing lexicon. In our system, we chose to use aspect-based sentiment analysis to be able to perform a more fine-grained analysis of the source text messages. 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 these aspects can be later used in the sentiment analysis model to detect sentiment in texts. Sentiment classification is the process of labeling aspects with a sentiment polarity.

For the task of aspect detection, we can find 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 used for this task, making use of a varied set of features (Jakob & Gurevych, 2010); non-supervised machine learning techniques such as Latent Dirichlet Allocation (LDA) (Blei et al., 2003). 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, and then associate a sentiment polarity 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 (Schouten & Frasincar, 2016); machine learning techniques using Support Vector Regression with other techniques to obtain the features for training the model and non-supervised methods that use for example relaxation labeling (Schouten & Frasincar, 2016). Hybrid techniques obtain aspects and assign polarities simultaneously. In (Nasukawa & Yi, 2003) syntax-based methods are used to obtain words associated with a sentiment polarity and then extract other aspects 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, it might prove more useful for assessing the state of the user and perform guiding and recommendation to extract information about the context of the conversations (e.g. the topic being talked about, the listeners), along with the use of sentiment classifiers.

Multimodal sentiment analysis is a line of research aiming to perform a fusion of different data types in sentiment analysis models to achieve better results than using only one data type. This line of research has gained an increasing amount of attention from the research community recently. In multimodal sentiment analysis, three main approaches are used, which are early fusion, intermediate, and late fusion (Huang et al., 2019). Firstly, early fusion or feature-level fusion combines different data type sources in a single data structure like a feature vector that is later fed to a model. In (Poria et al., 2016) authors used audio, video, and text feature fusion using a multiple kernel learning classifier. Differently, intermediate fusion refers to performing a fusion of data in intermediate layers of the model used. Lastly, late fusion refers to the combination of the output of models that use different data types for performing the sentiment classification. An example of intermediate fusion is performed in (Huang et al., 2019), where three models are presented; two of them being unimodal models featuring sentiment classification using deep CNN on image data and a LSTM on text data, while the third model featured a combination of the output visual features from the CNN model and the output text features from the LSTM for feeding a fully connected layer with the combination for obtaining a sentiment label. Authors also presented a late fusion framework, combining the output of the three presented models for performing sentiment classification. Moreover, in (Camacho et al., 2020) four dimensions of Social Network Analysis (SNA) are presented, along with four metrics to quantitatively measure them in existing or future technologies. Those are pattern and knowledge discovery, which determines the amount of valuable knowledge that a tool can extract from data; scalability, which measures how scalable a tool is; information fusion and integration, which measures the capacity of SNA technologies to integrate and fusion information from different data types and different sources; visualization, that measures the capacity of the technology to allow for visualizing information extracted from data. Authors computed the proposed metrics on a set of 20 popular SNA-software tools, and concluded that even when current technologies are scalable, can already handle significant amounts of data, and there is a large number of tools that provide flexible methods to visualize the information, content of SNSs is not being fully exploited by the current tools, 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. Additionally, 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 at on-line social environments. In this work, we combine the affective state detection with context information from the interactions, to better predict negative repercussions in a SNS and employing a CBR engine.

Stress strength detection using sentiment analysis techniques has been addressed in the literature in the past. A derivation of the sentiment strength detection software SentiStrength (Thelwall et al., 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 independent of the analysis on the aspects found. For the training of the aspect sets an unsupervised learning method later refined with a hill-climbing method is used. Sentiment and stress analysis has been performed on keystroke dynamics data in the literature. Sentiment analysis on keystroke dynamics data was addressed in (Lee et al., 2015), since International Affective Digitized Sounds (IADS) (Bradley & Lang, 2007) was used for inducing 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 not the one of accuracy rate of keyboard typing. In (Vizer et al., 2009) keystroke dynamics and linguistic features were used for successfully detecting cognitive and physical stress from free text data. According to the authors, the accuracy of detection of cognitive stress was consistent with the obtained using affective computing methods, and the accuracy for detection of physical stress encourages further research, despite being lower than the one on cognitive stress. Even being able to extract information about stress levels and sentiment polarities, these methods do not use multiple data types, which might lead to a system being able to better model the user state.

CBR systems are used to generate a case out of different characteristics of the environment or user interaction, and compare this case to a database of previous cases for extracting a potential solution or action to be taken in the current situation of a system (Kolodner, 1993). In (Heras et al., 2009), authors integrated a CBR module into a helpdesk software as a solution recommender, to be able to help operators in customer support environments. In (Marie et al., 2019), authors performed image segmentation of deformed kidneys using a CBR system and a CNN and compared them, resulting in that the CBR system succeeded in performing the best image segmentation. In (Bridge & Healy, 2012), a CBR system is proposed to act as a recommendation system between users and a review site. Users are recommended by the CBR system certain phrases from previous reviews found similar to the one that they are writing. The case base is populated using reviews from Amazon.com, decomposing them into words appearing on it, phrases to recommend to users, and the helpfulness rate of the review, created by Amazon.com users. Sentiment analysis using a CBR-based approach was performed in (Ohana et al., 2012). 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. Cases contain document statistics and the writing style of the review that generated it, and the solution associated which is the lexicons used to correctly predict sentiment on the review that generated the case. 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 et al., 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 authors, the accuracy obtained by the proposed model overcomes standard statistical approaches. A CBR system with a manually-constructed case base of emotion regulation scenarios for e-learning is presented in (Tian et al., 2014). The cases represent events in which e-learners suffer from certain emotions, and the solution to the cases is the advice and phrases to regulate their emotions. Similarity between the speaker sentences and the sentences in the cases is performed to select one case to apply for emotion regulation. Authors claim that the experimental results show that the proposed method has 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 et al., 2010) a nearest-neighbor collaborative approach was used to train user-specific classifiers, and those classifiers were later combined with user similarity measurements for solving a sentiment analysis task. Moreover, modeling the emotion from a group of entities has also been addressed. In (Rincon et al., 2017) authors modeled the emotions of a group of entities using the Pleasure, Arousal, and Dominance (PAD) emotional space. Authors used an Artificial Neural Network (ANN) to learn the emotion of the group when an event happens. Furthermore, there are works that apply a CBR approach to sentiment analysis to later perform a task with the detected user sentiment. In (Muhammad et al., 2015) authors implemented a CBR system that used opinions mined from user-generated reviews to help users decide in recommendation systems; In (Zhou et al., 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), and the second layer uses a CBR module to identify implicit characteristics of customer needs. Nevertheless, to the best of our knowledge, even when 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 using 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, which may enhance the performance of the system in predicting possible negative repercussions and help avoid them.

Risk prevention, user guiding, and privacy aiding in SNSs is an important topic nowadays. Privacy helping or aiding has been addressed in (Xie & Kang, 2015), by means of designing a user interface aiming to that purpose, with the main features of privacy in the system being visible to users by introducing privacy reminders and also customized privacy settings. In this work, privacy aiding is addressed in an indirect way, by analyzing different aspects of the mental state of the user to discover if they could be in a state that could lead to incurring risks from the interaction with other users. An example of user protection in SNSs by means of using sentiment analysis is performed in (Upadhyay et al., 2017), since authors implemented an SNS that used adult image detection, a message classification algorithm, and sentiment analysis in text messages. The system used 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 prove useful to use detection of different aspects of the user mental state (e.g. sentiment analysis, stress analysis, combined analysis), together with a CBR module using context information to generate feedback that helps prevent negative outcomes, which to the best of our knowledge remains unexplored.

3. System description

In a previous effort to tackle user state detection for guiding users navigating in SNSs (Aguado et al., 2020b), a MAS was presented, which computed sentiment, stress, and combined analysis of sentiment and stress on text data and keystroke dynamics data of user messages when they interacted in 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 based on the FIPA-ACL language. Moreover, we can find three different agent types on the MAS.

Firstly, there are the Presentation agents, in charge of communicating with the SNS, receiving data, and sending feedback to the users. Secondly, the Logic agents perform analyses on the messages and generate feedback if it is needed. Finally, the Persistence agent is the one who performs the data storage and retrieval tasks. Experiments with data from Twitter.com were conducted comparing which analyzer detected a state that propagated more in the network, therefore being more useful for preventing potentially negative repercussions.

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 detected values of affective states from users interacting and context information from interactions. This CBR module consists of two logic type agents. One agent performs the selection of 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, and sends the prediction of the case selected to the Advisor agent for potentially generating 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 cases. In this way, the agent makes the cases more likely to be selected if the predictions made with them were correct (the messages that were predicted using those cases to generate a negative or positive repercussion in the SNS did so), 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 can be seen in Figure 1.

The Presentation agent has assigned the tasks of receiving data from the SNS and sending feedback to the users navigating. Regarding the case of the Logic agents, we find a pipeline of agents, that perform the process needed to

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

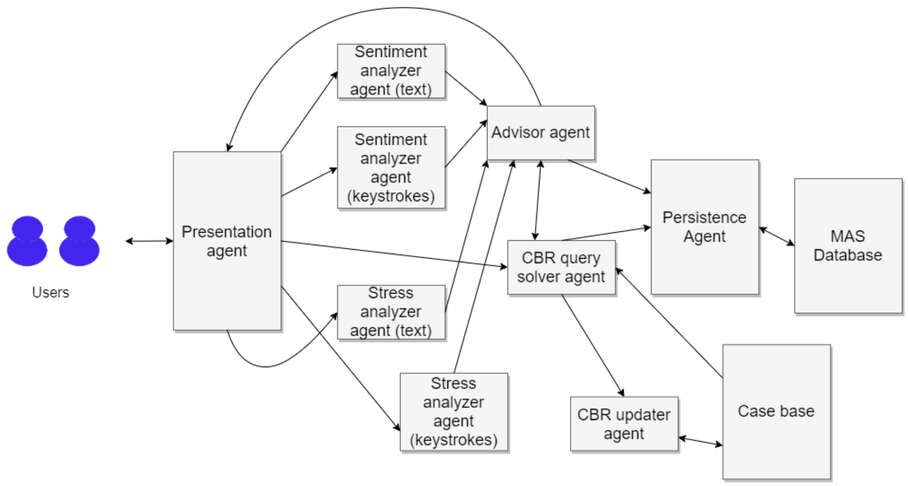


Figure 1: Architecture of the MAS

generate feedback to the user. This pipeline begins when the Presentation agent sends the data of messages to the Sentiment and Stress analyzer agents on text and the ones on keystroke data. When the sentiment and stress analyses have been computed, they send the outputs to the Advisor agent, who sends the data gathered about the message and output of analyses to the CBR query solver agent, who is in charge of finding the best matching case on 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 on the case base, and later gives the information of the solution of the selected case to the Advisor agent (whether this message could generate a negative repercussion or not), for creating the feedback to the user if the message is deemed negative. The feedback is stored in the database and sent to the Presentation agent, who delivers it to the SNS and to the user.

The Sentiment and Stress analyzer agents that work with text data perform an analysis with feed-forward ANNs, which use text embeddings to transform the text into embedding arrays. The embeddings used were pre-trained with the unsupervised algorithm GloVe (Pennington et al., 2014), using the Spanish Billion Words Corpus (Cardellino, 2019). These analyzers give as output the class negative or positive sentiment in the case of the Sentiment analyzer, and low or high stress level in the case of the Stress analyzer. The Sentiment and Stress analyzer agents working with keystroke data also use ANNs, which are fed with the arrays of keystroke features including timing and frequency of pulsation features, and give as output the same as the case of text data (positive or negative for sentiment analysis and low or high stress for stress analysis). 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 language Python, in its version 3.5.2. The architectures of the ANNs for the four analyzer agents are displayed in Figures 2 and 3 for the ANNs that operate with text embeddings and compute sentiment analysis and stress analysis, respectively, and in Figure 4 for the ANN that operates with keystroke dynamics data and computes sentiment analysis. The keystroke dynamics stress analysis ANN has the same architecture as the keystroke dynamics sentiment analysis ANN, 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 concrete 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 allow to find nonlinear patterns in non-parametrized data (Grossi & Buscema, 2007), such as text messages or arrays of timing and frequencies of pulsation, which are the data used to learn sentiment and stress level states in this work.

In the architecture of the text sentiment analysis ANN 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. Following the dense layer, a dropout layer with a

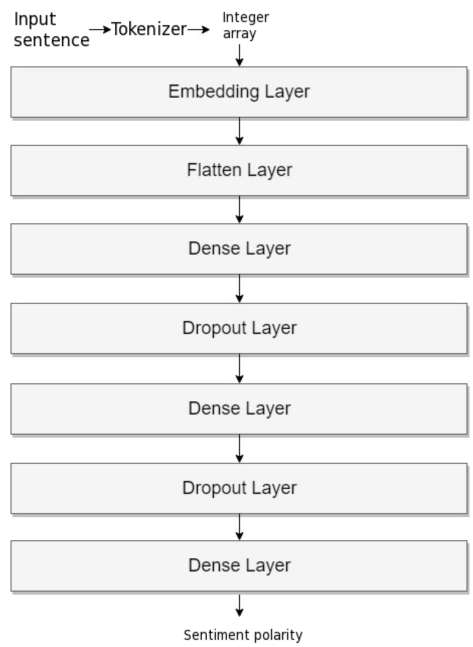


Figure 2: Architecture of the text embeddings sentiment analysis ANN

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. Following, 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 binary cross-entropy, and the optimizer Adam (Kingma & Ba, 2014). The architecture of the text stress analysis ANN has the same parameters and architecture, except that there are not internal dense and dropout layers. Finally, the keystroke dynamics ANNs have the same parameters and architecture as the text sentiment analysis ANN, except that the loss function used was categorical cross-entropy, that worked best for the ANNs working with keystroke dynamics array data. These ANN also have a different input which is the array of floating-point numbers corresponding to typing speed features and frequencies of pulsation of common keys, which is

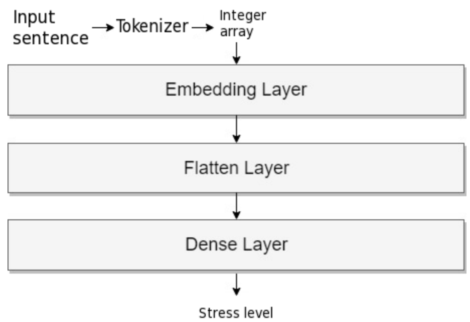


Figure 3: Architecture of the text embeddings stress analysis ANN

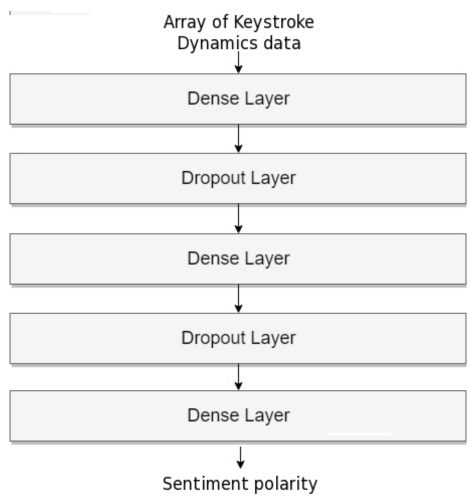


Figure 4: Architecture of the keystroke dynamics sentiment analysis ANN

fed to the first dense layer (there are no embedding and flatten layers). These features are summarized in Table 1.

The dataset used to train the ANN models was created by people of young age (between 12 and 15 years old), both male and female, in the SNS Pesedia, and labeled with positive or negative emotion and high or low stress level 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 6,475 labeled text messages, with associated keystroke dynamics data.

Table 1: Text typing speed and key frequency features used as input of 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 by the CBR query solver agent and the one performed by the CBR updater agent (who updates the case base periodically) will be detailed following in this section. 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 MAS proposed, is formed by the CBR query solver agent and the CBR updater agent, in addition to the case base and several data structures needed for the functioning of the module. These data structures correspond to dictionaries for storing priorities of cases, predictions made by the CBR module, parent structure of messages from the

SNS, the matching degree of potential new cases, and historical values of states detected by the CBR module on 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 by the agents for every step is elaborated in the following subsections. First of all, 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:

1. The time of the day.
2. Output of the text Sentiment analyzer as an integer (0 or 1 for negative or positive class, respectively).
3. Output of the text Stress analyzer as an integer (0 or 1 for negative or positive class, respectively).
4. Output of the keystroke Sentiment analyzer as an integer (0 or 1 for negative or positive class, respectively).
5. Output of the keystroke Stress analyzer as an integer (0 or 1 for negative or positive class, respectively).
6. 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 ones with 1 in the code (e.g. if within the last 4 user messages 3 were labeled as positive and 1 as negative, then the average would be computed as $3 + 0 / 4 = 0.75$).
7. Computed history of messages detected as negative or positive in messages written by the audience, as the average between all viewers (from the averaged values per user).
8. Computed topic found in the text message, using a trained Latent Semantic Indexing (LSI) (Deerwester et al., 1990) model and the gensim[2] library.

The history for users is computed using a window of the ten past 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. Therefore, 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 case of 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 in 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 that are predicted to be positive by the CBR, and negative otherwise. The solution is created either using the solution of a case in the case base that matches the new case in the retrieve step, or if none does using the combined value of sentiment and stress values detected from the four analyzer agents of the MAS. In the second option, a negative solution or value 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, and positive or 0 otherwise. A diagram of the functioning of the CBR module, with different software agents, actors, and possible actions is shown in Figure 5.

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

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

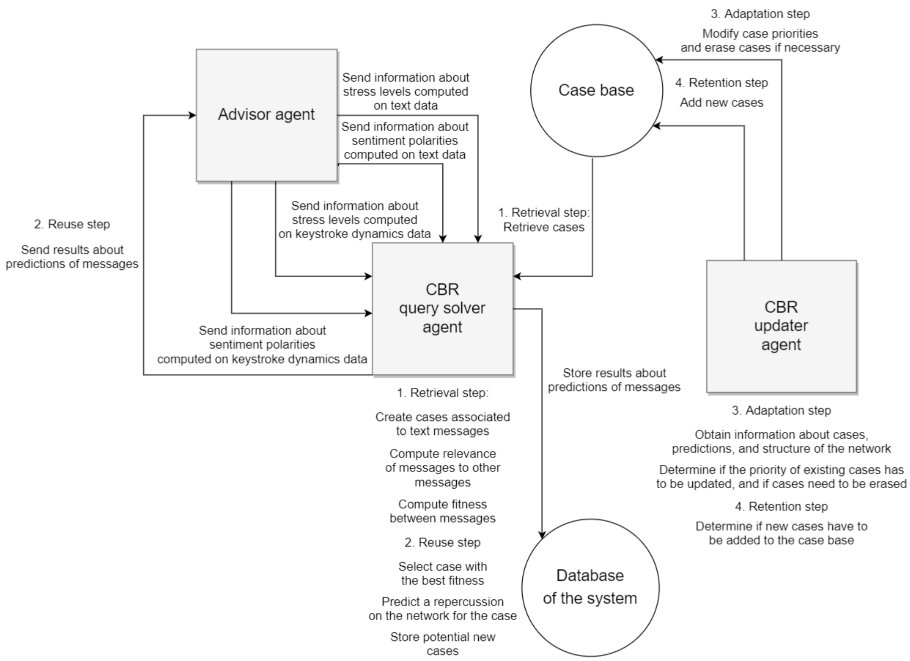


Figure 5: Diagram of actors and actions that form the CBR module and the elements that interact with it in the system

a case from the information related to a text message. The agent also extracts cases from the case base and finds the one that does the best matching with the case generated from the message data, and store 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 one with the highest priority to the lowest, but with a limitation in 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 select. For assessing whether or not one case does match with the case generated from the message or not, and in which grade, two functions were coded. One function checks for the relevance of the case to compare (which means that the case at least has bare minimum similarities with the case from the message), and the second function computes the matching degree between the two cases, checking the similarities between each defining feature in the cases (e.g. the topics from the texts). The matching degree function uses a weighted sum of the computed similarities of the features to compute the final matching degree. Firstly, the relevance is checked, then if the case to compare is relevant in the sense that it has bare minimum similarities to the case generated from the message, the matching degree between the two cases is computed. At the end of the process, the case that obtained the highest matching degree is selected.

The relevance function process is illustrated in Algorithm 1, and works as follows: run a loop for all the features that could exist in a case, and for every feature check if it exists in the original message to be compared to another. If this is the case, the corresponding feature is added to a variable accounting for the features to be matched. Following, a loop runs 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 compare to the original message, it is checked to compare the features of both cases, using comparisons according to 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 the total amount of characters from both sequences. If the comparison results in a basic match (the features are similar in at least a 50%), then 1 is added to a variable accounting for the found similarities. Finally, if the variable accounting for found similarities is equal or greater than a third of the number of features in the variable accounting for the features to be matched 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 matching degree function is illustrated in Algorithm 2, and works in the same way as the relevance function, except for two differences. Firstly, it checks any existing differences in the features between cases, and adds to the variable accounting for found similarities the proportion of the weight of the feature being compared equal to the degree of matching of the features. Secondly, it later gives as a result of matching degree the quotient between the variable accounting for found similarities and the value of a variable accounting for a 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

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

3.3. Reuse step

In the reuse step, the solution associated with the case selected in the previous step is used for the system to give an answer about whether the message sent by a user in a SNS could generate a negative repercussion in the network or social environment or not. 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 if the case will generate a negative repercussion or not) 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 delivering the feedback to the user. The Persistence agent simply stores the information about the prediction in the database. The CBR query solver agent also stores new potential cases that could be later added to the case base by the CBR updater agent. For this task, the cases generated from new messages are used. If the matching degree of new cases is below a threshold, meaning that the new case is different from the cases in the case base at least by the percentage given by the threshold, being this 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, conditioned to what the system observes and compares with its own previous predictions. For adapting the cases, priorities assigned to cases are modified. These priorities measure how well cases have performed when used for predicting and therefore 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 in the SNS, but with the interactions that happened in that interval. The agent works in this way for potentially leading to more useful information from the repercussion of messages (one message with only one reply gives the repercussion to one message, but 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.

```
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

The CBR updater agent performs the case-update loop as is shown in Algorithm 3. This agent runs a loop for all the cases that were selected by the CBR query solver agent, and for every case, another loop is done for every message that matched with 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 than 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 messages and parent structure of the messages, so the CBR module agents have the information about the parents and children of messages. Finally, the repercussion is computed as the most present predicted value by the CBR module on the replies of a given message. The information about cases selected, predictions performed, and parent structure of messages is first stored by the CBR query solver agent when cases are created and selected, and then used by the CBR updater agent when it is time for updating the case base.

3.5. Retain step

As mentioned before, the CBR query solver stores information about potential new cases, which are generated when messages are sent to the MAS, and cases are created associated with them. For this task, the matching degree of the new case with the cases on the case base is used. If the matching degree of new cases is below a threshold (value 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 amount, 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

For allowing a better comprehension of how the system works, consider the following practical scenario. When users are interacting in a SNS or other online social environment which is 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 text, and 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 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 will use this information and the one handed out by the Presentation agent to build a case representing the message being written in the SNS. The CBR query solver agent then computes the relevance, and if relevance is true, it computes the matching degree of the new case with cases in the case base and selects the best fitting case, to give a prediction of positive or negative repercussions in the network, caused by the message being written. Finally, the CBR query solver agent hands back the prediction to the Advisor agent, and if it is a prediction of negative value, a warning is generated and sent to the user interacting in the SNS 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 in the SNS), and priorities updated in the case base if the repercussions on the SNS show that 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

In these experiments, the main aims are two. Firstly, to fine-tune the CBR module by investigating what the most important features are in the cases, and what is the best configuration of the CBR module for populating the case base and achieving a low error rate in the prediction of the module. Secondly, to demonstrate the following hypothesis: 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, in this section, the two experiments performed with data from our private SNS Pesedia will be discussed.

The dataset used is the Pesedia dataset. It contains text messages and associated keystroke dynamics data and other necessary details of the text messages 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, both times during a period of one month. Nevertheless, the number of samples gathered in 2019 is larger than the one of the samples gathered in 2018. Additionally, only those gathered in 2018 were labeled with sentiment polarity and stress level (positive or negative and low or high stress level, respectively). There are 1609 samples from 2019 and 302 from 2018 in total.

Common affective state analysis datasets like Standford Sentiment Treebank (Socher et al., 2013a,b) consist of one data type and labels for a concrete affective state (in this case text and sentiment). Other example is the IMDB movie reviews dataset (Maas et al., 2011), which is a binary sentiment analysis dataset consisting of reviews from the Internet Movie Database (IMDb) labeled as positive or negative. Contrarily, the Pesedia dataset contains both text and keystroke dynamics data of user messages, other information related to the user interaction in the SNS, and also sentiment and stress level labels. This allows to create cases that contains different information about affective states of the user and other context information, derived from single user interactions in the social environment, which suits the purpose of the current work, because it enables the creation of cases with different information related to interaction in SNSs, and the comparison of prediction derived from the CBR and the one from other analysis methods. Additionally, to the best of our knowledge, a dataset that combines text, keystroke dynamics data and context information from interactions between users in social networks, that also contains sentiment and stress levels labels does not exist. Examples of the mentioned datasets can be found

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 negative or positive), 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

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 will be examined. Concretely, the different configurations used and the process followed for checking the error of prediction of the module will be elaborated and results presented. We altered the following parameters of the CBR module:

- 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 for assessing whether the system learns to predict better when giving different 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 let the interactions happen 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 following. For this process, the Pesedia dataset was used. A loop runs, processing an unlabeled user reply and the unlabeled original message that was replied by that message, and then a labeled reply message in every iteration. The unlabeled messages are used to feed the CBR module, so it processes them and can later update the case base according to this information. Following, 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, 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 a case associated by using the output of 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 messages unlabeled and labeled can be provided to the CBR module for this experiment.

For examining the error rate of the module in the process of populating the case base with different configurations while it is used for predicting, we define a metric. This metric is error of the system in a window of messages (Windowed Error or WE), thus the error of the system is computed for several windows of time where the system analyzes a set amount 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 and otherwise 0 (being the message used in iteration j the message j), $i = 0, 1, 2, ..., n$, 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, more concretely a fixed set of text messages 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 states).

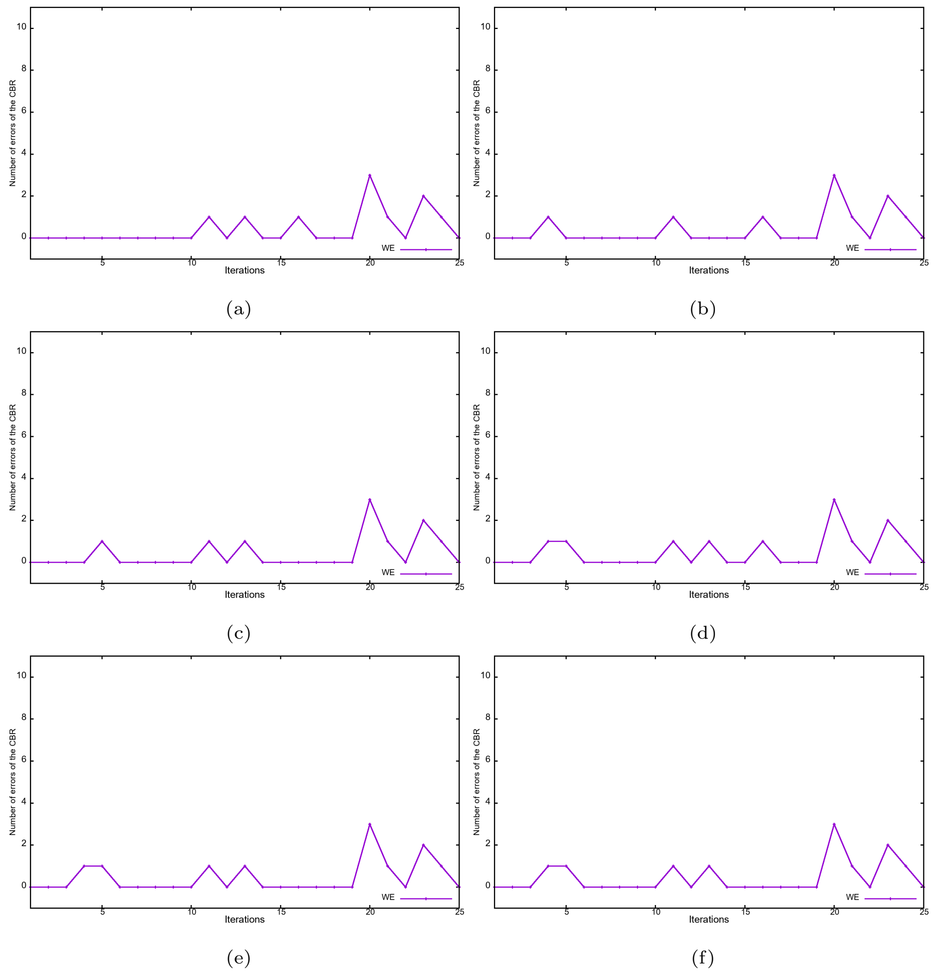
Regarding the experimental setup, several experiments were conducted, with different configurations of weights in the case features. Since the different possibilities of weight configurations are infinite, we created a selection of weight settings, aiming to test the configurations that led to more informative results about the performance of the CBR module. For every experiment, different update intervals for the CBR module update were tested, between 10 and 100 seconds. The configurations for each experiment are the following:

- 10_sen_str_day_and_history: equal weight for sentiment and stress features in the cases (10), and small value in the weights of time of the day and history of the audience features (3), history of the author of the message and topic of the message weights unaltered, being 5 and 10 respectively.
- 10_sen_str_no_day_and_history: the same case as the previous, except no value is added to the weights of time of the day and history of the audience features (0).
- 20_sen_day_and_history: doubled value of the weight representing sentiment than the one representing stress on the message (20 and 10 respectively), and the rest of the weights unaltered with respect to the first case.
- 20_sen_no_day_and_history: the same as the previous case, except no value is added to the weights of time of the day and history of the audience features (0).
- 20_str_day_and_history: doubled value of the weight representing stress than the one representing sentiment on the message (20 and 10 respectively), and the rest of the weights unaltered with respect to the first case.
- 20_str_no_day_and_history: the same as the previous case, except no value is added to the weights of time of the day and history of the audience features (0).

In figure 6 results for the experiments keeping the same weight in the sentiment and stress features while changing others are shown. Subfigures 6a, 6c, and 6e show the results for the 10_sen_str_day_and_history experiment with 10, 50, and 100 seconds of update interval in the CBR module, and subfigures 6b, 6d, and 6f 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 on those features. In the same way, results for the experiments doubling the weight of the sentiment feature and changing others are shown in Figure 7, and results of the experiments doubling the stress feature weight while again changing others are shown in Figure 8.

Following, an analysis of the experimental results will be performed. As is shown in the figures that present the error obtained by the CBR module in the 10_sen_str_day_and_history experiment, the error is low in general, observing 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 than in the previous case, it is shown 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 second update interval, and stays in the 100 second update interval.

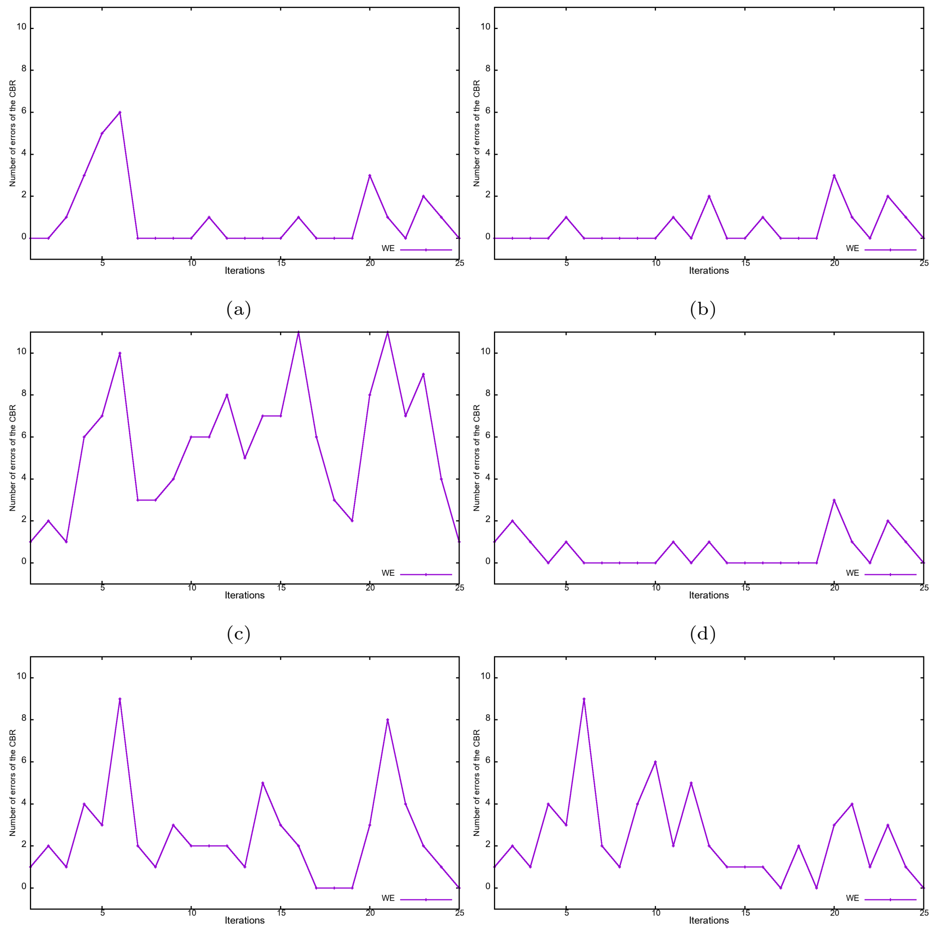
Contrarily as in the previous case, in the experiments where the weights for sentiment and stress features of the cases are altered, a general trend of more error is found in general, being higher in the case when the sentiment weight is considered two times as important than the stress one. In the case of the 20_sen_day_and_history and 20_sen_no_day_and_history experiments, the error is progressively and visibly smaller as the frequency of updates of the CBR module



(a) (b) (c) (d) (e) (f)

Figure 6: Error in the '10_sen_str_day_and_history' experiment and in the '10_sen_str_no_day_and_history' experiment with the 10, 50, and 100 seconds update intervals. From up to down figures for both experiments are shown in increasing order of update interval, and from left to right the figure corresponding to the '10_sen_str_day_and_history' experiment is shown, followed by the figure corresponding to the '10_sen_str_no_day_and_history' experiment

is increased. Additionally, in the case of these two experiments, setting to zero the weight of the time of the day and history of the audience features appears to reduce the error to some extent.



(a) (b) (c) (d) (e) (f)

Figure 7: Error in the '20_sen_day_and_history' experiment and in the '20_sen_no_day_and_history' experiment with the 10, 50, and 100 seconds update intervals. From up to down figures for both experiments are shown in increasing order of update interval, and from left to right the figure corresponding to the '20_sen_day_and_history' experiment is shown, followed by the figure corresponding to the '20_sen_no_day_and_history' experiment

In the case of the 20_str_day_and_history and 20_str_no_day_and_history experiments, the same effect of the time of the day and history of the audience features can be observed than on the experiments where the weights for the sentiment and stress features were the same, which is the appearance of more

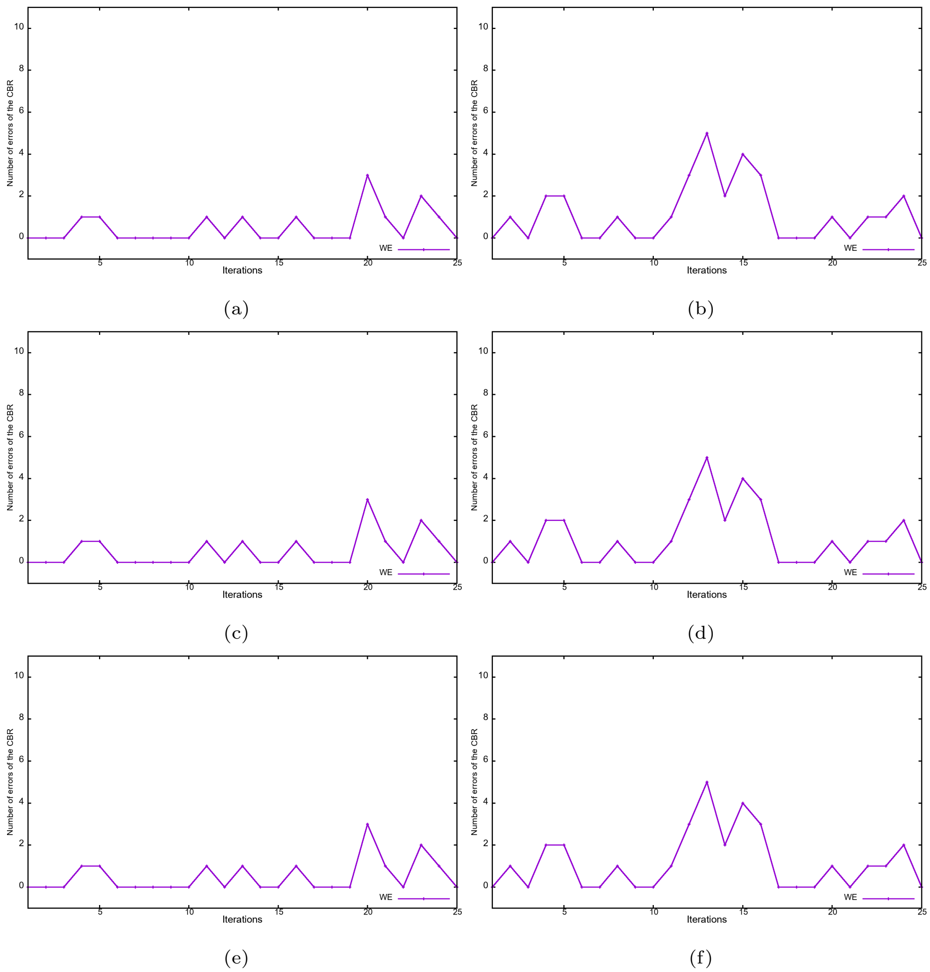


Figure 8: Error in the '20_str_day_and_history' experiment and in the '20_str_no_day_and_history' experiment with the 10, 50, and 100 seconds update intervals. From up to down figures for both experiments are shown in increasing order of update interval, and from left to right the figure corresponding to the '20_str_day_and_history' experiment is shown, followed by the figure corresponding to the '20_str_no_day_and_history' experiment

error 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 and being the same for both. Additionally, in this experimentation the error showed to diminish 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 lower error with the settings used and in the case of this dataset. The effect of the time of the day and history of the audience of the message features showed to reduce slightly the error as a general trend, except in the experiments where the sentiment weight was considered twice as important as the stress weight, although in these experiments a high error is 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 create some error in most of the cases, which might indicate that messages pertaining to the end of the dataset could be noisy to some extent.

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

We propose the following hypothesis to test: 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 using a static version of the CBR module (without updates) for predicting negative or positive repercussion 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), and compare the results obtained between the CBR module and the different analyzers. The dataset used was again the Pesedia dataset.

For comparing the capacity to detect positive and negative repercussions in the SNS, we use 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 the case of our study, we use the replies to a message as the messages directly influenced by this message (the message that was replied to). In this way, we are able to compute a metric that shows which analyzer is able to detect a state that is found more in the messages influenced by the message analyzed, therefore being able to better help the system prevent potentially negative outcomes (as in a negative state spreading through the network). The metric (PDV or Propagation of Detected Value) is computed as follows:

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

Where messages_with_replies is the total amount of messages that generated replies being analyzed, and messages_with_propagated_state is the aggregated value of messages with propagated state, which are messages with the same detected state as is present in most of their replies. By the state present in most replies, we refer to the state that has the most frequency from the states detected in the replies.

Related to the analyzers, we used Sentiment and Stress analyzers on text data, the same with keystroke data, 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), and the same for keystroke data again. 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, applied at decision level (after the sentiment and stress analyses have been computed). In this way, 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 level are detected, and positive otherwise. We used our CBR module for predicting 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 for assessing differences between using lower or larger amounts of data when populating the case base and comparing predictions (since the sets of samples are different in size). As 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 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.

Table 3: Comparison between analyzers and the CBR module

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

Every result presented 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. For this purpose, we partitioned the corresponding test set and used one partition for each experiment, showing the average of the four results in the table.

Finally, an analysis of the experimental results regarding the comparison of the CBR module and other analysis methods will be presented. As shown in Table 3, 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 Sentiment, Stress, and Combined analyzers non-CBR-based (on text data and on keystroke dynamics data). Performing experiments populating the case base with different data samples and different amounts of samples was done to ascertain whether the factors of using different data or different data sizes influences or not the results. As can be seen in Table 3, the CBR module was able to outperform the different analyzers non-CBR-based 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 before, 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 did not get affected by using or mixing different partitions of data samples when populating the case base. Additionally, using only a third of the unlabeled data samples to populate the case base, and the remaining two thirds to test performance showed 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, and have been integrated into a MAS for guiding and recommending users that navigate on-line social platforms or 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 for being able to receive data from messages of users in on-line social platforms in real-time, analyze the data, and hand it over to a CBR module that uses the information of the output of the analyses and context of the conversations for predicting potentially negative outcomes derived from the user interaction. Since the different 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 for using information about user interaction and user state together to perform predictions in on-line platforms, and recommend actions to users. In this work, context information such as the topic being talked about and the history of predictions of the system for the user writing and the users in the audience of a message are used together with sentiment polarity and stress level detected on text and keystroke dynamics data of messages. The CBR module implemented successfully predicts potentially negative repercussions (as in negative sentiment polarity or high stress levels spreading through users) in an on-line social platform when users write messages, using the information described above.

For assessing whether the CBR module using 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 CBR module performance resulted similar to the Sentiment analyzer using text and the 'or' Combined analyzer of sentiment and stress on text data, but 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 conducted with Pesedia data as well. In these experiments, the aim was to assess the error of the system when predicting 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, conducting several experiments varying weights and update intervals. The experiments showed that with the used configurations and with Pesedia data shorter update intervals lead to less error from the CBR module, and that certain configurations in the weights of the case features lead to less error than others.

For future lines of work, there are unexplored possibilities. Firstly, new features could be introduced in the cases to account for additional information to the system when making predictions. One example of a feature that might prove useful is the tiredness of users, which would give more information to the system about the real-time state of users when interacting, and 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 through an on-line social environment, but also use it to predict other phenomena that could be of interest to the system for guiding and recommending users, such as predicting cyber-bullying or on-line grooming, based on the detection of certain topics and states of the users interacting. 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 might be integrated and tested.

Acknowledgments. This work was financed by the project TIN2017-89156-R of the Spanish government.

References

References

Aamodt, A., & Plaza, E. (1994). Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI communications*, *7*, 39–59. doi: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*, 103740. doi: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*, 3754. doi: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*, 64–75. doi: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. doi: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*. Master's thesis Universitat Politècnica de València Valencia, Spain.

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

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. doi:10.1016/j.knosys.2011.06.024.

Camacho, D., Panizo-LLedot, Á., 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. doi: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. doi:10.1145/3340531.3412003.

Cardellino, C. (2019). Spanish Billion Words Corpus and Embeddings. URL: https://crscardellino.github.io/SBWCE/ accessed March 6, 2021.

Ceci, F., Goncalves, A. L., & Weber, R. (2016). A model for sentiment analysis based on ontology and cases. *IEEE Latin America Transactions*, *14*, 4560–4566. doi: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 March 6, 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*, 391–407. doi: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*, 82–89. doi:10.1145/2436256.2436274.

George, J. M., & Dane, E. (2016). Affect, emotion, and decision making. *Organizational Behavior and Human Decision Processes*, *136*, 47–55. doi: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. doi:10.1097/MEG.0b013e3282f198a0.

Heras, S., García-Pardo, J. Á., Ramos-Garijo, R., Palomares, A., Botti, V., Rebollo, M., & Julián, V. (2009). Multi-domain case-based module for customer support. *Expert Systems with Applications*, *36*, 6866–6873. doi: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. doi: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*, e0129056. doi:10.1371/journal.pone.0129056.

Li, F., Han, C., Huang, M., Zhu, X., Xia, Y.-J., Zhang, S., & Yu, H. (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, Deliverable D4. EU Kids Online Network, London, UK. `http://eprints.lse.ac.uk/33731/`. Accessed March 6, 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. doi:`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. doi:`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. doi:`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 (EMNLP)* (pp. 1532–1543). Association for Computational Linguistics. doi: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 (ICDM)* (pp. 439–448). IEEE. doi: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*, 47–54. doi: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*, 813–830. doi: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* (p. 195–206). Springer-Verlag. doi:10.1007/978-3-642-13470-8_19.

Socher, R., Bauer, J., Manning, C. D., & Ng, A. Y. (2013a). 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., & Potts, C. (2013b). 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*, 106–121. doi: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*, 2544–2558. doi:10.1002/asi.21416.

Tian, F., Gao, P., Li, L., Zhang, W., Liang, H., Qian, Y., & Zhao, R. (2014). Recognizing and regulating e-learners' emotions based on interactive chinese texts in e-learning systems. *Knowledge-Based Systems*, *55*, 148–164. doi: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 (ICISC)* (pp. 1–4). IEEE. doi: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*, 123–131. doi: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*, 459–480. doi: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*, 870–886. doi: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*, 398–407. doi: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*. doi:10.1115/1.4030159.