

# UMSNH at RestMex 2023: An XGBoost Stacking with Pre-Trained Word-Embeddings over Data Batches

Jaime Cerda-Flores<sup>1</sup>, Rodrigo Hernández-Mazariegos<sup>2</sup>, Jesús Ortiz-Bejar<sup>2</sup>,  
Félix Calderón-Solorio<sup>1</sup> and José Ortiz-Bejar<sup>1,\*</sup>

<sup>1</sup>División de Estudios de Posgrado, Facultad de Ingeniería Eléctrica, Universidad Michoana de San Nicolás de Hidalgo, Morelia, Michoacán, México

<sup>2</sup>Facultad de Ciencias Físico Matemáticas, Universidad Michoana de San Nicolás de Hidalgo, Morelia, Michoacán, México

## Abstract

The present work resumes the participation of the UMSNH Team at RestMex 2023. Our approach aims to categorize tourist destination reviews in Cuba, Colombia, and Mexico in three different aspects: the country to which the attraction belongs, the type of attraction, and the level of tourist satisfaction. For this purpose, a variety of different Word-Embeddings (FastText,  $\mu$ -TC, BERT) will be used to generate a baseline, after which combinations of these will be assembled using XGBoost to try to improve on their individual results.

## Keywords

Word-embeddings, BERT, FastText, stacking, XGBoost,  $\mu$ -TC, sentiment analysis, text categorization

## 1. Introduction

Text categorization is a natural language processing task that determines whether a language expression belongs to a class of interest. For instance, Sentiment Analysis consists in determining if an expression is positive, negative, or neutral. This task is commonly performed on written documents or voice data. The data can describe the public perception of a politician/product, a review for a movie, or a post on a social network (e.g., Facebook, Reddit, Twitter). It is possible to categorize texts/audio in different scales like humorousness, offensiveness, polarity, etc. This task is usually solved by a supervised learning method known as classification. A Classifier requires as an input a labeled corpus used to fit a function; the learned function will be capable of categorizing unseen samples.

Differently to others editions [1, 2], in the **RestMex 2023**, the organizers ask to analyze textual data; the task is stated by the organizer as "*Given an opinion about a Mexican tourist place, the goal is to determine the polarity, between 1 and 5, of the text, the type of opinion (hotel, restaurant or attraction) and, the country of the place of which the opinion is being given (Mexico, Cuba, Colombia)*". In broad terms, a text classification pipeline starts from a text corpus,

---


IberLEF 2023, September 2023, Jaén, Spain

\*Corresponding author.

✉ 1130834h@umich.mx (J. Cerda-Flores); 1301441a@umich.mx (R. Hernández-Mazariegos); jesus.ortiz@umich.mx (J. Ortiz-Bejar); felix.calderon@umich.mx (F. Calderón-Solorio); jose.ortiz@umich.mx (J. Ortiz-Bejar)



© 2023 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

which was previously labeled according to a predefined set of categories (training set). Then a preprocessing step is performed; the latter implies normalizing (case normalization, removing stop words, stemming, strategies to deal with links, usernames, hashtags, etc). From normalized documents, a vocabulary is produced through tokenization. A numerical representation for each model can be built and used from the vocabulary to fit a machine-learning algorithm [3, 4]. Transforming documents into a set of numerical vectors (often called Vector Space Model or VSM) is a challenging task. Fortunately, there is a vast collection of methodologies and research that deals exclusively with the topic.

Undoubtedly, the representation used directly impacts the performance of the classification algorithm; in other words, the quality of the model will depend on the VSM. For our approach, an ensemble of multiple of *light* and *fast* pre-trained models is used to produce VSMs, which are used to train a Multi-Layer Perceptron (MLP) to build and stacked VSM, which is finally ensembled through XGBoost classifier.

## 2. Task Description

The challenge we took for the RestMex 2023 was the Sentiment Analysis Task. For this assignment, the objective was to develop a system capable of classifying the polarity of an opinion given by tourists on different types of establishments in Mexico, Cuba, and Colombia. The system also had to determine the type of establishment the opinion was given on between three classes (Attractive, Hotel, and Restaurant) and the country in which it is located. Detailed task and dataset descriptions may be found in the overview document [5].

### 2.1. Data set

The training data set and the test data set were given by the organizers of RestMex 2023. The training data set consisted of an xlsx file with 251,702 sample reviews. Each sample has its corresponding Title column, Review column, Polarity label column, Country label column, and Type label column. The Polarity column represents the sentiment expressed in the review on a numeric scale ranging from 1 to 5, where 1 indicates a highly negative sentiment and a value of 5 represents a highly positive sentiment. There were no missing values for any columns in the training data set.

The test data set was also an xlsx file that contained 107,863 samples. Unlike the training data set, each sample only had an ID column (from 0 to 107862), a Title column, and a Review column.

The reviews were collected from TripAdvisor submissions by tourists between 2002 and 2022.

Table 1 shows the distribution of classes for each feature in the training data set. The distribution of classes for the test data set is unknown.

### 2.2. Evaluation

The organizers proposed the following equations to evaluate the sentiment score of a system  $k$ .

**Table 1**  
Distribution of Classes in Training Data Set

Polarity		Country		Type	
Class	Count	Class	Count	Class	Count
5	157095	Mexico	118776	Attractive	111188
4	60227	Colombia	66703	Restaurant	76042
3	21656	Cuba	66223	Hotel	64472
2	6952				
1	5772				

$$Res_p(k) = \frac{\sum_{i=1}^{|C|} ((1 - \frac{T_{C_i}}{T_C}) * F_i(k))}{\sum_{i=1}^{|C|} (1 - \frac{T_{C_i}}{T_C})} \quad (1)$$

$$Res_A(k) = \frac{F_A(k) + F_H(k) + F_R(k)}{3} \quad (2)$$

$$Res_C(k) = \frac{F_{Mex}(k) + F_{Cub}(k) + F_{Col}(k)}{3} \quad (3)$$

$$Sentiment(k) = \frac{2 * Res_p(k) + Res_A(k) + Res_C(k)}{4} \quad (4)$$

The evaluation of the Polarity classification is given by equation 1, and it was designed to give more weight to minority classes, being these the ones with more negative polarities. For Type and Country classification, the proposed equations 2 and 3, respectively, obtain the average of the Macro-F measures of each class. Equation 4 is the average of the previous equations, giving more weight to polarity.

### 3. Vector Space Models

The first step to transforming text into vectors involves building a vocabulary. The goal is to convert text sequences into minimal meaningful writing units (tokens); for example, split a sentence into words. As we deal with written language, obtaining the tokens requires the manipulation of character strings. It is required to identify punctuation, diacritics (dieresis, accents). In the case of English, you may want to divide the contractions (You're → you are). From the vocabulary, it is possible to build a vector representation.

Note that tokenization could be at the suffix/prefix level, syllables, or even letters. It is also possible to build units made up of 2,3 or  $n$  words; these tokens are known as  $n$ -grams and allow us to include concepts of more than one unit, for example, in English terms like *ice cream* or *New York*.

**Table 2**

Binary one-hot encoding for a simple text document

$d_1$ /vocabulary	rain	may	fall	and	storms	come
rain	<b>1</b>	0	0	0	0	0
may	0	<b>1</b>	0	0	0	0
fall	0	0	<b>1</b>	0	0	0
and	0	0	0	<b>1</b>	0	0
storms	0	0	0	0	<b>1</b>	0
may	0	<b>1</b>	0	0	0	0
come	0	0	0	0	0	<b>1</b>

### 3.1. Tokenizing and vectorizing

The tokenization process is a document segmentation process. The segmentation is dividing the text (unstructured information) into smaller units that can be accounted for discretely. The result of counting the occurrences of each term, known as *bag of words*, can be used directly as a vector representation of the document. Information retrieval and search are the most common application of this type of vector (bag of words) for document retrieval or search.

As an example, consider the following sentence  $d_1$  "*Rain may fall, and storms may come*" (the sentence was taken from the lyrics of the song '*Bob Lennon*' in the English version of the manga '*20th Century Boys*' by Urasawa, N. (2000-2006), published by VIZ Media). By tokenizing  $d_1$  by words, lower-casing, and removing punctuation, a simple way of obtaining a numerical representation of a text sequence is employing a binary representation of each token that exists in the vocabulary; this representation is known as *one-hot vectors*. Each word may be represented as a one-hot vector, where each one-hot binary vector will be the vocabulary size and will have only a 1 in the position that corresponds to the word it represents. Therefore, each document vector(matrix) is comprised of the list of the tokens in the document (see Table 2).

In Table 2, each row is the one-hot vector for one word in  $d_1$ , for instance the token *fall* is represented by the vector  $\{0, 0, 1, 0, 0, 0\}$ .

For our example, a  $7 \times 6$  matrix since the vocabulary is only made up of a single sentence. In a matrix document, one indicates that the token is part of the document, and zero indicates that that term is not. This type of representation is often used in neural networks and language modeling. The importance of any vector representation is that it allows transforming sentences written in natural language into a space where it is possible to perform mathematical operations to apply classification models.

One of the disadvantages of having a matrix representation of one-hot vectors is that they are highly dispersed; their storage can be inefficient if done in a matrix way, while if we do it through lists or some other dispersed structure, there is an increase in the complexity of the operations. For instance, in the Spanish language, it is considered that there are around 100000 words [6]. As a result, a one-hot vector for a word will be a 100000-dimensional vector with only a few elements different from 0. Even with the previous disadvantage, this vector would preserve the concepts that appear in the document; it would be similar to an index of

terms in a book. Moreover, a binary vector is enough to determine whether or not the word appears in the document due to it is frequently used in search and information retrieval. A simple "improvement" for binary one-hot encoding is counting how many times each token appears in the sentence. The latter can be seen as assigning a weight based on per document word frequency.

### 3.2. Frequency Vector Spaces

Even though identifying and counting terms in the vocabulary is helpful for simple problems, such as computing usage statistics or performing keyword searches, for more complex cases we might also want to perform tasks in which the importance within the document or the collection of the words is relevant. For example, it is not desirable for an email to be classified as spam if it only contains a common term in that type of message.

There are plenty of weighted models; they aim to provide a representation that captures the importance of each token. One of the most popular representations is frequency-based, like TF and TF-IDF (Term Frequency Inverse Document Frequency). TF refers to the frequency of a term in a document, while IDF implies that TF will be weighed based on the frequency of that same term in the entire collection. Schemes like TF-IDF help to relate a token to a specific document in a corpus, assigning a numerical value to the importance of that word in the given document depending on its frequency within the entire corpus.

## 4. $\mu$ -TC

The  $\mu$ -TC system works with frequency vector spaces.  $\mu$ -TC is a versatile and compact tool created to handle various tasks related to text classification, regardless of the specific domain or language involved. This is achieved by treating the creation of efficient text classifiers as a problem of optimizing combinations of function parameters applied over a corpus of text. The functions involved encompass text modifications that normalize a given text, tokenization options for the text, and weighting techniques, followed by vectorizing the resulting vocabulary, which is used to train a Support Vector Machine (SVM). For the approach evaluated in this work, the output probabilities of the SVM are stacked as the input for an XGB Classifier to perform new predictions. Even though  $\mu$ -TC is not a word embedding or deep learning approach, it is a competitive scheme used successfully in multiple text classification contests (for instance [7, 8]). For a detailed description of how  $\mu$ -TC works, the reader may consult [9].

## 5. Word-Embeddings

In the same way as the frequency-based techniques, word embeddings are used to transform text into numeric vectors. The main difference is that word embeddings are dense vectors usually learned by a deep neural network. Each word is assigned a unique vector so that similar words must have vectors close to each other. Word embeddings are based on the idea that the words around them influence a word's meaning. In the following subsection, a brief review of some of the embeddings we used is given.

## 5.1. FastText

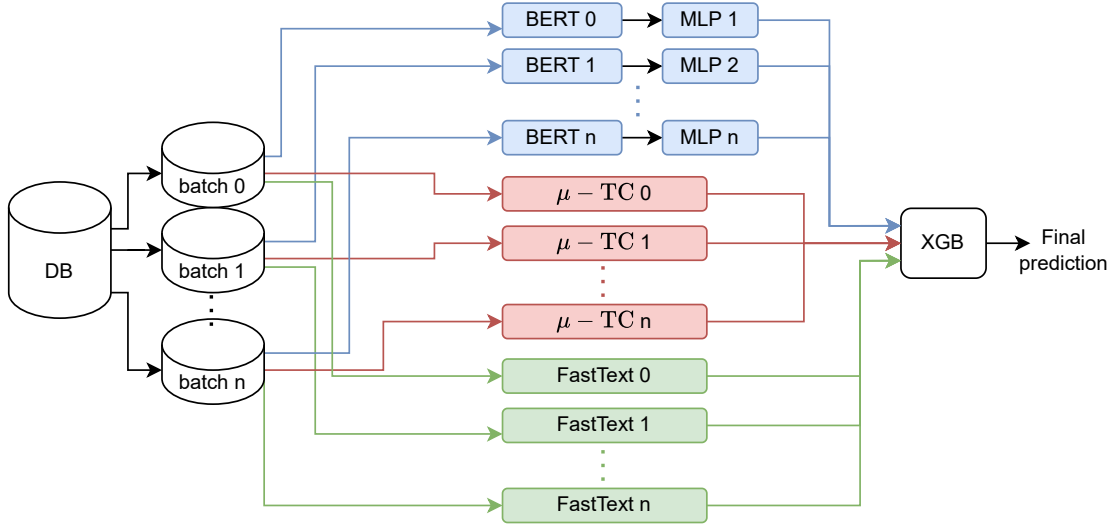
FastText is a library developed by the Facebook AI Research lab for text classification and representation. FastText is based in *skipgram* and *cbow* architectures introduced by Mikolov in [10]. This tool transforms text into continuous vectors, enabling its application to supervised and unsupervised training tasks. Like with  $\mu$ -TC, it is also essential to find the best combination of function parameters available for building efficient models, although we used base parameters for our work. The classification task is done using multinomial logistic regression, taking the sentence or document vector obtained during the model-building phase as the features. An interested reader in FastText will find a full description in [11].

## 5.2. BERT

Bidirectional Encoder Representations from Transformers (BERT), first introduced in [12] is a deep learning approach tool capable of managing several NLP tasks, including text classification. A transformer is a type of neural network architecture that employs a self-attention mechanism that allows the model to weigh the importance of different words in a sentence to determine their relationship. BERT processes text using a bidirectional approach. This means BERT can analyze a word in the context of both preceding and following words, providing a better understanding of the word's meaning. It was pre-trained using text from the English Wikipedia and the Toronto BookCorpus utilizing Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). In our case, the resulting model is then fine-tuned to a specific task, which is text classification. Given that the BERT model used was pre-trained on an English database and the samples from the RestMex data sets were in Spanish, it was necessary to use a multi-lingual model that generates aligned vector spaces.[13]. You may be referred to [14] for a comprehensive survey about BERT.

## 6. An ensemble word-embedding batched approach

For the proposed task consisting of a *pretty massive* data set, a *modest* computer (Intel(R) Xeon(R) W-2125 CPU @ 4.00GH, with 32 GB ram) is not enough to process all data at the same time for most of the pre-trained word-embeddings; Our team used a batched approach where the 251702 documents were divided into  $n$  batches using  $n - k$  as training set and the remaining  $k$  for model validation. Figure 1 depicts our approach. First, data is split into  $n$  equal-size batches, one  $\mu$ TC, and a FastText model is fitted for each batch; as BERT is not a complete classification system (we only use the pre-trained model), documents in batches are transformed in BERT sentences embeddings by averaging all words in each document. A Multi-Layer Perceptron (MLP) is fitted. A new VSM is built by concatenating the probabilities classes output for each of the fitted models; the new VSM is used to fit an XGB classifier. The XGB classifier performs the final prediction. Please note that the input dimension of the data fed into XGBoost is given by  $C \times n \times n_m$ , where  $C$  represents the number of classes,  $n$  denotes the number of training batches, and  $n_m$  represents the number of used VSMs. In our experiments,  $n_m$  is limited to the set  $\{1, 2, 3\}$ .



**Figure 1:** Pipeline classification based on multi word-embedding stacking over data batches

It is worth mentioning that an instance of the pipeline is run for each of the three sub-tasks proposed for the RestMEX 2023 (i.e., Attraction, Country, and polarity).

## 6.1. XGBoost

XGBoost[15] is an optimized distributed gradient boosting open source library designed to be highly efficient, flexible, and portable. It works under the gradient boosting framework. Gradient boosting tries to, given two weak classifiers models  $F_1, F_2$ , build a stronger classifier  $F_3$  following a Taylor expansion where:

$$\gamma_2 = \operatorname{argmin}_{\gamma} L(y_{true}, F_1(x) + \gamma F_2(x)) \quad (5)$$

$$F_3 = F_1 + \gamma_2 F_2 \quad (6)$$

Where  $L$  is the predefined lost function. Once we have  $F_3$ , we can iterate over all our weak models to improve gradually; note how this is a Taylor expansion of first order; XGBoost uses a second-order Taylor expansion introducing the gradient and hessian of  $F_1$ .

## 6.2. Experimental setup

For our experiments, the number of batches  $n$  was fixed to 20, where four ( $k = 4$ ) are separated as a validation set and 16 for training the model; a cross-validation with five folds is performed to measure the generalization capacity for our model. For the FastText and  $\mu$ TC classifier, the default system parameters are kept and applied to each of the 20 batches. For  $\mu$ TC, no preprocessing is used over the text because it is one of the parameters optimized by  $\mu$ TC. On the other hand, for BERT and FastText, the only preprocessing applied is normalizing the text

to lowercase. Please note that while FastText and  $\mu$ TC predictions are fed forwardly to XGB, BERT is used to produce document embeddings by averaging the vector representation for the words in each text; the embeddings are used to train an MLP, which outputs are the inputs used to the staking process.

### 6.3. Results

Table 3 summarizes the scores  $Sentiment_P$ ,  $Res_p$ ,  $Res_C$ , and  $Res_A$  for the five folds using 20 batches for all seven possible systems combinations on the training data set. BERT produces the best result when a single model is used. In contrast, the best ensemble uses a BERT and FastText combination. To ease reading best results are bold-faced.

**Table 3**  
Results for 5 folds over 20 data batches

Ensemble	$Sentiment_P$	$Res_P$	$Res_C$	$Res_A$
$\mu$ TC	0.6425	0.3833	0.8483	0.9553
BERT	0.6873	0.4669	0.8496	0.9658
FastText	0.6797	0.4435	0.8727	0.9593
$\mu$ TC + BERT	0.6914	0.4608	0.8712	0.9730
$\mu$ TC + FastText	0.6826	0.4420	0.8830	0.9634
BERT + FastText	<b>0.7004</b>	<b>0.4688</b>	0.8913	0.9726
$\mu$ TC + BERT + FastText	0.6960	0.4592	<b>0.8921</b>	<b>0.9735</b>

Table 4 depicts the rank in which each system placed for the competition on the test data set as well as its corresponding Sentiment Track Score, ours being UMSNH-Team\_sentiment\_results\_xgb\_ensemble (bold-faced) in 9th place. All teams were able to submit as many system results as they wished. Still, only their highest-scoring result is taken into account.

## 7. Conclusions and future work

From the results, it is clear that assembling improves on the results given by the individual embeddings. However, it is noticeable that when using a combination of the three models, there is a slight decrease in the score for Sentiment and Polarity, which indicates that doing so generates a little loss of information. Given the results from the RestMex competition, where our work placed us in the middle of the other teams, the presented methodology could be considered as a baseline for different events as it is relatively lightweight and computationally undemanding.

As for future work, we would like to experiment with different sizes of batches to see if this affects the results somehow, as well as improving the models generated by FastText through parameter optimization. We can now use heavier embedding methods (ELMo, RoBERTA) due to the recent acquisition of a more powerful server capable of processing the complete data set without splitting it into batches. Having more types of embeddings, we can include combinations



**Table 4**  
RestMex 2023 Sentiment Analysis Track Results

Run	Sentiment Track Score	Rank
LKE-IIMAS-Team_RUN_2	0.7790190145	1
javier_alonso-Team_sentiment_sub6	0.766199648	2
IIMAS-UNAM-Team_resultados	0.7500702313	3
INGEOTEC-Team	0.7375714733	4
UCT-UA-Team_run_01	0.7190152899	5
BUAA-Team_M1_M1_M2	0.7189219428	6
Dataverse-Team_Results_RestMex23	0.7173586609	7
SENA-Team_i-1	0.700868302	8
<b>UMSNH-Team_sentiment_results_xgb_ensemble</b>	<b>0.6991728136</b>	<b>9</b>
Camed_CU-ES-Team_camed_camila_y_eduardo_results_output	0.6977828325	10
JL-Team-Added	0.6888903674	11
ITT-Team-Topics_k=1	0.6814168808	12
Algiedi-Team-5000	0.6691846165	13
Arandanito-Team_POS-ADJ+NOUN+VERB	0.6392190289	14
ABCD-Team_submission	0.4727653249	15
Olga-Team-dfTestfinal_LyS-SALSA	0.3144048884	16
The_Last-Team	0.2295272137	17

of these in the optimization process instead of using a brute-force method. We aim to improve the performance of our system by further exploring the mentioned steps.

## References

- [1] M. Á. Álvarez-Carmona, R. Aranda, S. Arce-Cardenas, D. Fajardo-Delgado, R. Guerrero-Rodríguez, A. P. López-Monroy, J. Martínez-Miranda, H. Pérez-Espinosa, A. Y. Rodríguez-González, Overview of rest-mex at iberlef 2021: Recommendation system for text mexican tourism, *Procesamiento del Lenguaje Natural* (2021).
- [2] M. Á. Álvarez-Carmona, Á. Díaz-Pacheco, R. Aranda, A. Y. Rodríguez-González, D. Fajardo-Delgado, R. Guerrero-Rodríguez, L. Bustio-Martínez, Overview of rest-mex at iberlef 2022: Recommendation system, sentiment analysis and covid semaphore prediction for mexican tourist texts, *Procesamiento del Lenguaje Natural* 69 (2022) 289–299.
- [3] M. A. Álvarez Carmona, R. Aranda, A. Y. Rodríguez-Gonzalez, D. Fajardo-Delgado, M. G. Sánchez, H. Pérez-Espinosa, J. Martínez-Miranda, R. Guerrero-Rodríguez, L. Bustio-Martínez, Ángel Díaz-Pacheco, Natural language processing applied to tourism research: A systematic review and future research directions, *Journal of King Saud University - Computer and Information Sciences* 34 (2022) 10125–10144. URL: <https://www.sciencedirect.com/science/article/pii/S1319157822003615>. doi:<https://doi.org/10.1016/j.jksuci.2022.10.010>.
- [4] A. Diaz-Pacheco, M. A. Álvarez Carmona, R. Guerrero-Rodríguez, L. A. C. Chávez, A. Y. Rodríguez-González, J. P. Ramírez-Silva, R. Aranda, Artificial intelligence methods to support the research of destination image in tourism. a systematic re-

- view, *Journal of Experimental & Theoretical Artificial Intelligence* 0 (2022) 1–31. URL: <https://doi.org/10.1080/0952813X.2022.2153276>. doi:10.1080/0952813X.2022.2153276. arXiv:<https://doi.org/10.1080/0952813X.2022.2153276>.
- [5] M. Á. Álvarez-Carmona, Á. Díaz-Pacheco, R. Aranda, A. Y. Rodríguez-González, L. Bustio-Martínez, V. Muñis-Sánchez, A. P. Pastor-López, F. Sánchez-Vega, Overview of rest-mex at iberlef 2023: Research on sentiment analysis task for mexican tourist texts, *Procesamiento del Lenguaje Natural* 71 (2023).
- [6] *Diccionario de la Lengua Española*, 23rd ed., Real Academia Española, 2014. URL: <https://www.rae.es/>.
- [7] J. Ortiz-Bejar, V. Salgado, M. Graff, D. Moctezuma, S. Miranda-Jiménez, E. S. Tellez, Ingeotec at ibereval 2018 task haha:  $\mu$ tc and evomsa to detect and score humor in texts, in: *Proceedings of the Third Workshop on Evaluation of Human Language Technologies for Iberian Languages (IberEval 2018) co-located with 34th Conference of the Spanish Society for Natural Language Processing (SEPLN 2018)*, 2018.
- [8] D. Moctezuma, J. Ortiz-Bejar, E. S. Tellez, S. Miranda-Jiménez, M. Graff, Ingeotec solution for task 4 in tass'18 competition., in: *TASS@ SEPLN*, 2018, pp. 111–115.
- [9] E. S. Tellez, D. Moctezuma, S. Miranda-Jiménez, M. Graff, An automated text categorization framework based on hyperparameter optimization, *Knowledge-Based Systems* 149 (2018) 110–123. URL: <https://github.com/INGEOTEC/microtc>.
- [10] T. Mikolov, K. Chen, G. Corrado, J. Dean, L. Sutskever, G. Zweig, word2vec, URL <https://code.google.com/p/word2vec> 22 (2013).
- [11] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, J. Dean, Distributed representations of words and phrases and their compositionality, in: *Advances in neural information processing systems*, 2013, pp. 3111–3119.
- [12] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).
- [13] N. Reimers, I. Gurevych, Making monolingual sentence embeddings multilingual using knowledge distillation, in: *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, 2020. URL: <https://arxiv.org/abs/2004.09813>.
- [14] F. A. Acheampong, H. Nunoo-Mensah, W. Chen, Transformer models for text-based emotion detection: a review of bert-based approaches, *Artificial Intelligence Review* (2021) 1–41.
- [15] T. Chen, C. Guestrin, XGBoost: A scalable tree boosting system, in: *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '16*, ACM, New York, NY, USA, 2016, pp. 785–794. URL: <http://doi.acm.org/10.1145/2939672.2939785>. doi:10.1145/2939672.2939785.