INGEOTEC at SemEval-2024 Task 10: Bag of Words Classifiers

Mario Graff[‡] and **Eric S. Tellez**[‡] and **Mireya Paredes**[‡]
and **Daniela Moctezuma**[†] and **José Ortiz-Bejar**[⋆]
[†] CentroGEO, Aguascalientes, México
[‡] CONACyT - INFOTEC, Aguascalientes, México
[⋆] Universidad Michoacana de San Nicolás de Hidalgo, Morelia, México
{mario.graff,eric.tellez,mireya.paredes}@infotec.mx
dmoctezuma@centrogeo.edu.mx
jose.ortiz@umich.mx

Abstract

The Emotion Recognition in Conversation subtask aims to predict the emotions of the utterance of a conversation. In its most basic form, one can treat each utterance separately without considering that it is part of a conversation. Using this simplification, one can use any text classification algorithm to tackle this problem. This contribution follows this approach by solving the problem with different text classifiers based on Bag of Words. Nonetheless, the best approach takes advantage of the dynamics of the conversation; however, this algorithm is not statistically different than a Bag of Words with a Linear Support Vector Machine.

1 Introduction

Sentiment analysis has been very useful for emotion detection in digital text. For example, in the analysis of a customer review, a sentiment analysis system can find whether the review is positive or negative. Today, this way of finding out what the sentiment expressed in the digital text has been popular due to its potential to have a feeling over what people are writing about. Recent studies have been conducted towards sentiment analysis, not only in a one-party text (text written by one person/source) but also within a multi-party conversational text (Hazarika et al., 2018)(Majumder et al., 2018)(Poria et al., 2019), known as Emotion Recognition in Conversation (ERC). ERC refers to the emotion detection of each of the phrases/utterances within a dialogue. For instance, in a conversation between two people (speaker 1, SP1, and speaker 2, SP2) saying the following, SP1: "I had an awful day", SP2 replies "Oh no, what happened?". SP1 may have a "sad" emotion and SP2 may be also "sad". However, following the conversation SP1 answers "Somebody ate my sandwich!" and SP2 replies "I can make you a new one right now!)". This answer provokes a change in the emotion of SP1 to "joy". The aim of ERC is precisely the detection of speakers' emotional changes involved within a dialogue. ERC research has become popular due to the vast amount of conversation sources in social media such as opinion mining in chat history, social media threads, debates, and understanding consumer feedback in live conversations, among others (Majumder et al., 2018).

To date, several studies have investigated ERC using different approaches (Hazarika et al., 2018)(Majumder et al., 2018) as it is summarized by (Poria et al., 2019). (Hazarika et al., 2018) presented a framework for emotion detection in conversations using a recurrent neural network (RNN) based memory network with multi-hop attention modeling. (Majumder et al., 2018) is a method based on an RNN that maintains information of each party separately and this information is used for emotion classification. Most recent studies have been focused on more elaborated proposals about emotion detection based on the context and the common sense knowledge within the conversation (Tu et al., 2022). Recently, (Jiang et al., 2024) presented a self-supervised model to better understand the semantics within the text associated with the order of the utterances.

In this paper, we present a model given the *Task 10: Emotion Discovery and Reasoning its Flip in Conversation(subtask 1)* of the SemEval-2024 workshop (Kumar et al., 2024) which consists of applying *Emotion Recognition in Conversation (ERC)* to Hindi-English code-mixed conversations. We propose to solve the challenge as a classification problem, using a Bag of Words (BoW) for the representation. Despite the usage of BoW is not so common anymore due to the current usage of more sophisticated techniques as deep learning, our work is built on the previous work presented in (Graff

Proceedings of the 18th International Workshop on Semantic Evaluation (SemEval-2024), pages 1115–1120
June 20-21, 2024 ©2024 Association for Computational Linguistics

et al., 2023a,b), leading to a unique and customized BoW for solving this specific problem.

The rest of the paper is organized as follows: Section 2 presents the background; Section 3 introduces a description of the model; finally Section 4 shows a brief analysis of the results and Section 5 concludes the paper.

kumar2024semeval

2 Background

Emotion recognition has been a popular research field using Artificial Intelligence. According to (Saxena et al., 2020), several methods are applied for Emotion recognition including facial expression recognition, physiological signals recognition, speech signals variation, and text semantics, among others. Specifically, in this work, we focus on emotion recognition written in a digital text. Previously, emotion recognition in the text has been focused on the selection of emotional keywords (Seol et al., 2008) and the classification of their emotional state within a conversation. However, this keyword-based method presented some limitations as ambiguity and the lack of semantic and syntactic information. Emotion recognition in a conversation has shown to be a challenge due to the way emotions change over time. Other machine learning method have been applied as ICON (Hazarika et al., 2018) and DialogueRNN (Majumder et al., 2018), both using RNN. ICON (Hazarika et al., 2018) generates memories from the conversation to generate a good context for predicting emotions within a conversational video. DialogueRNN (Majumder et al., 2018) presented a model involving three aspects in a conversation: the speaker, the context of the previous phrases, and the emotion, according to them, taking into account these key aspects leads to a much better context representation. Commonsense knowledge is something difficult to pass over a machine, for that reason new approaches added external knowledge to help the machine to have a context (Speer et al., 2018)(Cambria et al., 2022) for solving new challenges as having an empathetic dialogue system (Ma et al., 2020).

3 System overview

The subtask Emotion Recognition in Conversation can be posed as a supervised learning problem. Without considering that emotions are part of a conversation, the problem can be seen as finding the mapping between an utterance and its associated emotion, i.e., it is a classification problem. In order to use the majority of classifiers, one needs to transform the utterance into a format amenable to the classifier selected. Generally, the representation acceptable for the majority of traditional classifiers is vectors.

Perhaps one of the most studied representations that transform a text into a vector is the Bag of Words (BoW); it is not so common anymore because it has been overcome by the use of deep learning techniques such as the attention mechanism (Vaswani et al., 2017). However, our participation is based solely on the use of BoW, following a similar approach used in previous competitions see (Graff et al., 2023a,b), and complementing it with an approach tailored for this specific subtask.

The realm of the BoW representation is that each token t of a text is associated with a vector $\mathbf{v_t} \in \mathbb{R}^d$. In this contribution, the i-th component of $\mathbf{v_t}$ corresponds to the token's Inverse-Document-Frequency (IDF) estimated in a collection of Hindi tweets (9.5 million), and the rest of the components of $\mathbf{v_t}$ are zero, i.e., $\forall_{j \neq i} \mathbf{v_t}_j = 0$. The set of all tokens is fixed, and these correspond to the vocabulary. The vocabulary is fixed to containing only $2^{17} = d$ elements, and this corresponds to the most frequent tokens found in the collection of tweets. Furthermore, given that only one component of each vector is different from zero, the set of all the vectors constituted a basis, and each text is represented in this vector space. Additionally, any token that is not found in the vocabulary is discarded from the representation.

Using this notation, a text x is represented by the sequence of its tokens, i.e., $(t_1, t_2, \ldots)$; the sequence can have repeated tokens, e.g., $t_j = t_k$. Then each token is associated with its respective vector $\mathbf{v}$ (keeping the repetitions), i.e., $(\mathbf{v_{t_1}}, \mathbf{v_{t_2}}, \ldots)$. Finally, the text x is represented as:

$$\mathbf{x} = \frac{\sum_t \mathbf{v_t}}{\|\sum_t \mathbf{v_t}\|}, \tag{1}$$

where the sum goes for all the elements of the sequence, $\mathbf{x} \in \mathbb{R}^d$, and $\|\mathbf{w}\|$ is the Euclidean norm of vector $\mathbf{w}$. The term frequency is implicitly computed in the sum because the process allows token repetitions.

The second representation is inspired by a self-supervised technique, particularly the procedure of masking tokens in a text and then developing an algorithm to predict the masked tokens.

The idea is pursued by creating M binary classification problems where the task is to predict the presence of a particular token; in this case, the tokens are words (defined as a string surrounded by spaces or punctuation symbols) or emojis. The words are selected based on their frequency; the most frequent words are not considered, and the words considered started when the plot of rank vs. frequency settles, i.e., it is when the flat part starts. On the other hand, all the emojis are considered. However, only the words and emojis where there are more than 1024 positive examples in the collection are kept. In total, there are 176 tweets and 2048 words.

There are 2,224 binary classification problems; each is solved using a BoW representation where the classifier is a Linear Support Vector Machine. Consequently, there are M binary text classifiers, i.e., $(c_1, c_2, \ldots, c_M)$. The utterance is represented using the decision values of the M binary classifiers; that is, the text lives in $\mathbb{R}^M$. As can be seen, each component is associated with either a word or emoji, and its value indicates the likelihood of its presence in the text. We refer to this representation as Dense. Finally, the classifier used with the dense representation is again a Linear Support Vector Machine.

After the competition ended, we decided to include in the comparison a procedure to combine (using Stacking (Graff et al., 2020)) the BoW and Dense representation, namely StackBoW. The idea is to make a convex combination of the class probabilities predicted by these two classifiers. The approach is to use the training set with k-fold cross-validation to estimate the decision function of these two classifiers on the training set. These decision functions are then transformed with softmax to obtain probabilities, and then an optimizer is used to estimate the convex combination. There are 8 emotions, so the optimizer needs to find 8 coefficients, each corresponding to a class. For example, let $\mathbf{p}_b \in \mathbb{R}^8$ be the probability given by the BoW classifier, $\mathbf{p}_d \in \mathbb{R}^8$ corresponds to the dense classifier, and $\beta \in \mathbb{R}^8$ are the estimated coefficients, then the prediction of the StackBoW is $\beta \odot \mathbf{p}_b + (1-\beta) \odot \mathbf{b}_d$ where $\odot$ is the pointwise product.

The last system, INGEOTEC, corresponds to an approach that takes advantage of the conversation dynamics. It considers the current utterance, the previous, and the next. In the extremes, either the next or the previous are empty utterances. Let $\mathbf{x}$, $\mathbf{x}_p$, and $\mathbf{x}_n$ be the dense representation, then it is computed the similarity between the current utterance ($\mathbf{x}$) and the previous and next utterance, as follows: $s_p = \rho \odot \mathbf{x} \cdot \mathbf{x}_p$, and $s_n = \rho \odot \mathbf{x} \cdot \mathbf{x}_n$. At first, ρ is a vector of ones, so s_p and s_n are the cosine similarity because the dense representations have unit length.

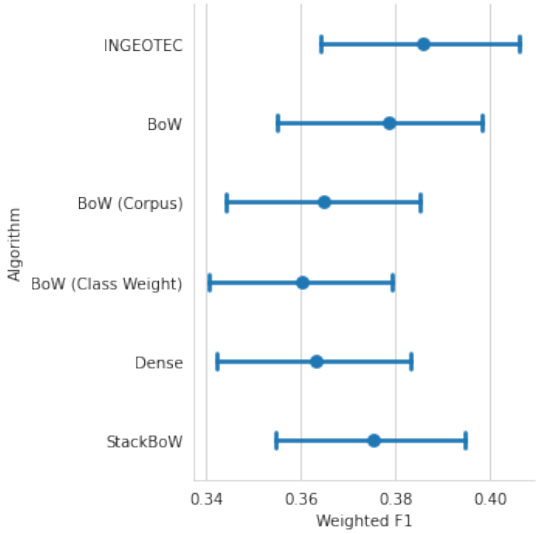
Using s_p and s_n, the contribution of each representation is computed by converting the similarity to a probability; this is done with the softmax as $\mathbf{s} = \mathsf{softmax}(1, s_p, s_n)$. Using $\mathbf{s}$ another representation is created which is the convex combination between $\mathbf{x}$, $\mathbf{x}_p$, and $\mathbf{x}_n$, i.e., $\mathbf{x}_s = \mathbf{s}_1\mathbf{x} + \mathbf{s}_2\mathbf{x}_p + \mathbf{s}_3\mathbf{x}_n$. Using $\mathbf{x}$ and $\mathbf{x}_s$, the dense representation used is the concatenation of them, i.e., $\mathbf{w} = [\mathbf{x}, \mathbf{x}_s]$. The final dense representation, $\mathbf{w}$, is used in a linear equation combined with softmax to predict the probabilities of each class. The probabilities obtained in the previous step are combined, using a convex combination, with the decision function of a BoW classifier (transformed with softmax). It is important to mention that the parameters, ρ, the coefficients to create the final convex combination, and the parameters of the linear equation of the dense representation $\mathbf{w}$ are optimized with gradient descent.

4 Results

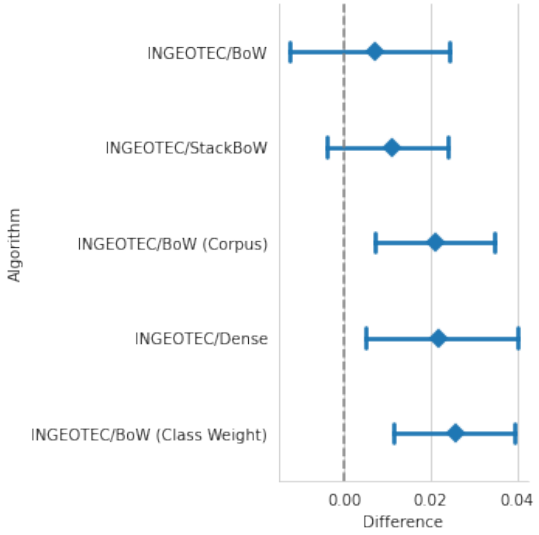
The systems' performance analysis starts with the information presented in Table 1. The table presents the performance, in terms of F1 scores per class, of the BoW (Class Weight) classifier, the Dense classifier, and their convex combination, namely StackBoW. The BoW (Class Weight) classifier is identified using the term *Class Weight* to indicate that the Linear Support Vector Machine was optimized by giving a weight inversely proportional to the class frequencies to each sample. This configuration is also used in the Dense classifier.

Table 1 is organized in three row blocks. The first one identified with the parameter β presents the coefficients used to make the convex combination of BoW and Dense. The second-row block contains the performance (F1 scores per class) estimated in k-fold cross-validation in the training set, and the third block corresponds to the performance in the test set. It can be observed from the table that the performance of StackBoW in the k-fold cross-validation is better than that of its components. This improvement is not reflected in all the cases in the test set; nonetheless, the convex combination is better than its components in the weighted

F1 score.



(a) Weighted F1 score and its estimated confidence intervals (90%) for the different systems.



(b) Difference in performance and its confidence interval (90%) between the best system (namely, INGEOTEC) and the rest.

Figure 1: Analysis of the weighted F1 score in the test set obtained by different algorithms. The dashed line corresponds to zero. An interval crossing the dashed line indicates the difference is not statistically significant with confidence of 90%.

Figure 1 complements the information presented in Table 1. Figure 1a presents the performance using the weighted F1 score on the test set for all the systems tested and its associated 90% confidence interval (the confidence intervals were estimated using the procedure described in (Nava-Muñoz et al., 2023)). The figure also includes the difference in performance (Figure 1b) between the best-performing systems, namely INGEOTEC, and the rest of the systems. The difference in performance shows the 90% confidence interval. The performance of INGEOTEC is 0.3861. The comparison figure includes a dash line that it is set in zero, consequently any confidence interval that intersect with the dash line indicates that the difference in performance is not statistical significant. Using this information, it can be observed that INGEOTEC is similar to the BoW classifier –it is worth mentioning that BoW weights all samples with 1, which makes it different and BoW (Class Weight)– and StackBoW.

5 Conclusion

We have described the algorithms tested on the Emotion Recognition in Conversation task. Most of the approaches treat this problem by looking at each utterance separately. The only system taking advantage of the dynamic of the conversation is the INGEOTEC system; this system is the one having the best performance. Nonetheless, as Figure 1b shows, it is not statistically different than a BoW classifier. The BoW classifier is the simplest model one can start experimenting with.

References

Erik Cambria, Qian Liu, Sergio Decherchi, Frank Xing, and Kenneth Kwok. 2022. SenticNet 7: A commonsense-based neurosymbolic AI framework for explainable sentiment analysis. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 3829–3839, Marseille, France. European Language Resources Association.

Mario Graff, Sabino Miranda-Jiménez, Eric S. Tellez, and Daniela Moctezuma. 2020. EvoMSA: A Multilingual Evolutionary Approach for Sentiment Analysis. *Computational Intelligence Magazine*, 15(1):76 – 88.

Mario Graff, Daniela Moctezuma, Eric Tellez, and Sabino Miranda. 2023a. Ingeotec at DA-VINCIS: Bag-of-Words Classifiers. *CEUR Workshop Proceeding*, 3496:1–10.

Mario Graff, Daniela Moctezuma, Eric Tellez, and Sabino Miranda. 2023b. Ingeotec at restmex: Bag-of-words classifiers. *CEUR Workshop Proceedings*, 3496:1–11.

Devamanyu Hazarika, Soujanya Poria, Rada Mihalcea, Erik Cambria, and Roger Zimmermann. 2018. ICON: Interactive conversational memory network for multimodal emotion detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2594–2604, Brussels, Belgium. Association for Computational Linguistics.

Table 1: F1 scores per class, in the training set, using k-fold cross-validation, and in the test set. β corresponds to the value of the convex combination. BoW and Dense are the base classifiers of StackBoW and the predictions of these are combined using β.

	anger	contempt	disgust	fear	joy	neutral	sadness	surprise
β	0.21	0.25	0.96	0.73	0.54	0.35	0.54	0.71
k-fold cross-validation								
BoW (Class Weight)	0.23	0.21	0.14	0.20	0.41	0.52	0.26	0.15
Dense	0.22	0.20	0.12	0.20	0.38	0.46	0.28	0.21
StackBoW	0.24	0.21	0.16	0.23	0.43	0.5	0.29	0.19
Test set								
BoW (Class Weight)	0.23	0.16	0.10	0.18	0.39	0.47	0.24	0.27
Dense	0.25	0.23	0.05	0.13	0.41	0.45	0.32	0.27
StackBoW	0.26	0.22	0.13	0.18	0.44	0.46	0.29	0.27

Dazhi Jiang, Hao Liu, Geng Tu, Runguo Wei, and Erik Cambria. 2024. Self-supervised utterance order prediction for emotion recognition in conversations. *Neurocomputing*, 577:127370.

Shivani Kumar, Md Shad Akhtar, Erik Cambria, and Tanmoy Chakraborty. 2024. Semeval 2024 – task 10: Emotion discovery and reasoning its flip in conversation (ediref). In *Proceedings of the 2024 Annual Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Yukun Ma, Khanh Linh Nguyen, Frank Z. Xing, and Erik Cambria. 2020. A survey on empathetic dialogue systems. *Information Fusion*, 64:50–70.

Navonil Majumder, Soujanya Poria, Devamanyu Hazarika, Rada Mihalcea, Alexander F. Gelbukh, and Erik Cambria. 2018. Dialoguernn: An attentive RNN for emotion detection in conversations. *CoRR*, abs/1811.00405.

Sergio Nava-Muñoz, Mario Graff Guerrero, and Hugo Jair Escalante. 2023. Comparison of Classifiers in Challenge Scheme. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, 13902 LNCS:89–98.

Soujanya Poria, Navonil Majumder, Rada Mihalcea, and Eduard H. Hovy. 2019. Emotion recognition in conversation: Research challenges, datasets, and recent advances. *CoRR*, abs/1905.02947.

Anvita Saxena, Ashish Khanna, and Deepak Gupta. 2020. Emotion recognition and detection methods: A comprehensive survey. *Journal of Artificial Intelligence and Systems*, 2:53–79.

Yong-Soo Seol, Dong-Joo Kim, and Han-Woo Kim. 2008. Emotion recognition from text using knowledge-based.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2018. Conceptnet 5.5: An open multilingual graph of general knowledge. *Preprint*, arXiv:1612.03975.

Geng Tu, Jintao Wen, Cheng Liu, Dazhi Jiang, and Erik Cambria. 2022. Context- and sentiment-aware networks for emotion recognition in conversation. *IEEE Transactions on Artificial Intelligence*, 3(5):699–708.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems 30 (NIPS 2017)*, pages 5998–6008. Curran Associates, Inc.

Appendix A: Library Usage

This appendix aims to illustrate the use of BoW, Dense, and StackBOW that are implemented using EvoMSA (`evomsa.readthedocs.io`) (Graff et al., 2020). The first step is to install the library, which can be done using the Anaconda package manager with as follows:

```
conda install -c conda-forge EvoMSA
conda install -c conda-forge IngeoML
```

Once EvoMSA is installed, one must load a few libraries.

```
from EvoMSA import BoW, DenseBoW
from EvoMSA.back_prop import StackBoW
from IngeoML.utils import soft_comp_weighted_f1,
    support
```

The BoW classifier is trained with the following instruction; it is assumed that the list D contains as elements dictionaries with two keys: text and klass; the latter is used as the emotion.

```
bow = BoW(lang='hi').fit(D)
```

Let us assume that the test set is in a list of dictionaries, G, where the utterance is in the key text. Then, the following instruction is used to predict the emotion of each utterance.

```
emotions = bow.predict(G)
```

BoW (Class Weight) is initialized with the following instructions.

```
kwargs = dict(dual='auto', class_weight='balanced')
bow = BoW(lang='hi',
          voc_size_exponent=17,
          estimator_kwargs=kwargs).fit(D)
```

On the other hand, the Dense classifier is trained using the following command.

```
kwargs = dict(dual='auto', class_weight='balanced')
dense = DenseBoW(lang='hi',
                 voc_size_exponent=17,
                 estimator_kwargs=kwargs).fit(D)
```

The StackBoW classifier is trained with the next step. Finally, all the classifiers have the method *predict* to forecast the emotions of any given utterance.

```
kwargs = dict(class_weight=support)
stack_bp = StackBoW(lang='hi',
                    deviation=soft_comp_weighted_f1,
                    voc_size_exponent=17,
                    optimizer_kwargs=kwargs).fit(D)
```