Choosing the right loss function for multi-label Emotion Classification

Lluís-F. Hurtado*, José-Ángel González and Ferran Pla
Departament de Sistemes Informàtics i Computació Universitat Politècnica de València Camí de Vera sn, València, Spain

Abstract. Natural Language Processing problems has recently been benefited for the advances in Deep Learning. Many of these problems can be addressed as a multi-label classification problem. Usually, the metrics used to evaluate classification models are different from the loss functions used in the learning process. In this paper, we present a strategy to incorporate evaluation metrics in the learning process in order to increase the performance of the classifier according to the measure we are interested to favor. Concretely, we propose soft versions of the Accuracy, micro-$F_1$, and macro-$F_1$ measures that can be used as loss functions in the back-propagation algorithm. In order to experimentally validate our approach, we tested our system in an Emotion Classification task proposed at the International Workshop on Semantic Evaluation, SemEval-2018. Using a Convolutional Neural Network trained with the proposed loss functions we obtained significant improvements both for the English and the Spanish corpora.

Keywords: Deep Learning, loss function, multi-label classification, Natural Language Processing, Emotion Classification

1. Introduction

In recent years, Deep Learning has become one of the most sprawling approaches in many areas of artificial intelligence. Many Natural Language Processing (NLP) tasks has been benefited from the advances in this area. A wide range of these tasks can be addressed following a classification framework. Sentiment Analysis [12, 28], Language Identification [27], Emotion Classification [17, 20], and Stance Detection [16, 19, 31], among others, are examples of NLP tasks addressed as classification problems.

A classification problem can be defined as the problem of learning a function capable of assigning to each sample one or more classes of the set of classes defined for the task. Different metrics to evaluate the performance of a classification model are proposed in the literature such as, Accuracy, Precision, Recall, or $F_1$ measure, with some average methods (macro-averaging and micro-averaging) [30]. Depending on the characteristics of the system that we want to build, we will be interested in favoring one or other measure. In this sense, it is interesting to consider some mechanisms that allow us to incorporate this criterion in the learning phase of the model.

In Deep Learning approaches, the loss function is used by the back-propagation algorithm to guide the parameter estimation process. Although any differentiable function can be used as loss function, a few numbers of loss functions are usually used, without considering the measure used to evaluate the specific task. Two of the most used loss functions in the literature are the Cross-Entropy (CE), in its binary (BCE) or categorical versions, and the Mean Squared Error (MSE) [7]. These functions are individually computed for each sample, which means that class-level
performance during the learning process is not taken into account.

Some recent works proposed different loss functions for further improving machine learning systems. A study of how particular choices of loss functions affect deep models and their learning dynamics can be consulted in [10]. The authors show the results will lead to a wider adoption of several losses in Deep Learning area, in particular, non-classical loss functions such as Tanimoto loss and Cauchy-Schwarz Divergence.

Other works are oriented to optimize $F_1$ measure for multinomial regression models [4]. This work introduced a novel plug-in rule algorithm that estimates all parameters required for a Bayes-optimal prediction. The use of $F_1$ measure as training criterion for Deep Learning systems to an image cleaning and enhancement binary task was previously studied [25]. We extended their work by defining commonly used evaluation metrics as loss functions to train neural networks for multi-class and multi-label classification problems.

In order to test different loss functions on a multi-label corpus, we selected the E-c: Detecting Emotions Multilabel classification corpus used in subtask 5 of Task 1 (Affect in Tweets) at the International Workshop on Semantic Evaluation, SemEval-2018 [20]. This subtask has high interest in the Natural Language Processing and Social Media Analytics areas. It consists on given a tweet, assign it one, or more, of eleven given emotions that best represent the mental state expressed in the text.

The rest of the paper is organized as follows. Section 2 presents our proposal to use evaluation metrics as loss function. Section 3 describes the main characteristics of the addressed task. Section 4 presents the designed system, concretely the preprocessing, the tweet representation, and the Convolutional Neural Network (CNN) architecture we used. Section 5 presents the experimental work conducted. Finally, section 6 presents some conclusions and the future work.

2. Evaluation metrics as loss functions

We propose some loss functions for training neural networks in order to address classification problems. These functions are approximations to the classical metrics used to evaluate classification models. The proposed functions have the advantage of being differentiable and consequently, they can be used as loss functions in the back-propagation algorithm.

Firstly, we formally define the classification problem and the metrics used for its evaluation. Next, we present the proposal of the new differentiable loss functions.

2.1. The classification problem

The classification problem can be defined as the problem of learning a function $f : \mathcal{X} \rightarrow \mathcal{Y}$ from a set of labeled samples $D$, where

- $\mathcal{C}$ is a finite set of classes $\mathcal{C} = \{c_1, \ldots, c|\mathcal{C}|\}$, and $|\mathcal{C}| > 1$
- $\mathcal{X}$ is an input space
- $\mathcal{P}(\mathcal{C})$ is the label power set of $\mathcal{C}$
- $\mathcal{Y}$ is the set of considered labels, $\mathcal{Y} \subseteq \mathcal{P}(\mathcal{C})$, and $D \equiv \{(x_1, y_1), \ldots, (x_m, y_m)\}$ is a data set of samples, where $x_i \in \mathcal{X}$ and $y_i \in \mathcal{Y}$

When $|\mathcal{C}| > 2$, the problem is called multi-class classification and when a sample can be assigned to more than one class (i.e. $\mathcal{C} \subset \mathcal{Y}$) the problem is called multi-label classification.

In order to automatically evaluate the performance of the classifier we assume that a labeled test set is available. Let $(\mathcal{X}, \mathcal{Y})$ be a test set consisting on $m$ samples $(x_i, y_i)$, $1 \leq i \leq m$, $(x_i \in \mathcal{X}, y_i \in \mathcal{Y})$, with a set of classes $\mathcal{C}$, $|\mathcal{C}| = n$. Let $O = \{\theta_i : 1 \leq i \leq m\}$ be the set of predictions of the classifier on the test set, where $\theta_i \in \mathcal{Y}$ is the set of classes assigned to the sample $x_i$ by the classifier.

One of the most used metrics to evaluate classifiers is Accuracy. In mono-label classification tasks, i.e. only one class per sample, Accuracy is defined as the percentage of correctly classified samples. For multi-label classification tasks, it is necessary to introduce an extension of the Accuracy metric. Equation (1) shows the formulation of multi-label Accuracy ($Acc$).

$$Acc = \frac{1}{m} \sum_{i=1}^{m} \frac{|y_i \cap \theta_i|}{|y_i \cup \theta_i|}$$  \hspace{1cm} (1)

where the numerator represents the number of correctly predicted classes for sample $i$, normalized by the size of the union of the predicted and correct class sets for sample $i$. Note that this metric is equivalent...
to Jaccard index per sample, averaged across all the samples.

The other metrics we study in this work are Precision, Recall, and \( F_1 \) measure. Moreover, there are two points of view to calculate a value for these metrics considering the complete test set, micro-averaging and macro-averaging.

Following the micro-averaging approach, we can define the micro-Precision, or just Precision \( (P) \), as the fraction of all classes generated by the classifier that have been correctly predicted; and the micro-Recall, or just Recall \( (R) \), as the fraction of all correct classes that have been correctly predicted by the classifier. Both metrics are formally defined in Equations (2) and (3).

\[
P = \frac{\sum_{i=1}^{m} |y_i \cap \theta_i|}{\sum_{i=1}^{m} |\theta_i|} \quad (2)
\]

\[
R = \frac{\sum_{i=1}^{m} |y_i \cap \theta_i|}{\sum_{i=1}^{m} |y_i|} \quad (3)
\]

**Micro-\( F_1 \)** \((m-F_1)\) is a particular case of micro-\( F_\beta \) measure where \( \beta = 1 \), that is, the harmonic mean of Precision and Recall. Equation (4) shows the formulation of micro-\( F_1 \).

\[
m-F_1 = \frac{2 \cdot P \cdot R}{P + R} = \frac{2 \cdot \sum_{i=1}^{m} |y_i \cap \theta_i| \cdot \sum_{i=1}^{m} |y_i|}{\sum_{i=1}^{m} |\theta_i| \cdot \sum_{i=1}^{m} |y_i|} = \frac{\sum_{i=1}^{m} |y_i \cap \theta_i| \cdot \sum_{i=1}^{m} |y_i|}{\sum_{i=1}^{m} |\theta_i| \cdot \sum_{i=1}^{m} |y_i|}
\]

\[
m-F_1 = 2 \cdot \frac{\sum_{i=1}^{m} |y_i \cap \theta_i|}{\sum_{i=1}^{m} |\theta_i| + \sum_{i=1}^{m} |y_i|} \quad (4)
\]

Additionally, we can compute the Precision, Recall and \( F_1 \) per class. Eqs. (5), (6) and (7) show the definition of these metrics for a specific class \( c \). Note that, we use the Iverson bracket notation \([ c \in y_i \cap \theta_i ]\) which has the value 1 if \( c \) belongs to \( y_i \cap \theta_i \) and 0 otherwise.

\[
P_c = \frac{\sum_{i=1}^{m} [ c \in y_i \cap \theta_i ]}{\sum_{i=1}^{m} [ c \in \theta_i ]} \quad (5)
\]

\[
R_c = \frac{\sum_{i=1}^{m} [ c \in y_i \cap \theta_i ]}{\sum_{i=1}^{m} [ c \in y_i ]} \quad (6)
\]

\[
F_{1c} = 2 \cdot \frac{\sum_{i=1}^{m} [ c \in y_i \cap \theta_i ]}{\sum_{i=1}^{m} [ c \in \theta_i ] + \sum_{i=1}^{m} [ c \in y_i ]} \quad (7)
\]

Following the macro-averaging approach, we can compute the macro-\( F_1 \) \((M-F_1)\) as the arithmetic mean of \( F_1 \) per class. Equation 8 shows the definition of macro-\( F_1 \).

\[
M-F_1 = \frac{2}{|C|} \cdot \sum_{c \in C} \frac{\sum_{i=1}^{m} [ c \in y_i \cap \theta_i ]}{\sum_{i=1}^{m} [ c \in \theta_i ] + \sum_{i=1}^{m} [ c \in y_i ]} \quad (8)
\]

In this case, all classes equally contribute to the global measure regardless of the number of samples.

### 2.2. Soft metrics

In this section, we present some differentiable functions that are approximate versions of the averaged metrics introduced above. These functions can be used as loss functions to train neural networks for multi-class and multi-label classification problems.

When neural networks are used to address classification problems, the last layer of the network needs as many neurons as classes. Let \( o_i \) be the output layer of the network when processing sample \( i \), and \( o_i^j \) the value of the \( j \)th neuron of \( o_i \), that is, the value assigned to class \( c_j \). Let \( y_i \) be the set of correct classes of sample \( i \) represented as a vector, thus \( y_i^j = 1 \) if sample \( i \) belongs to class \( c_j \), otherwise \( y_i^j = 0 \).

In order to determine, from \( o_i \), the classes assigned to the sample \( i \) by the model, it would be necessary to set a threshold and to select those classes for which the value of \( o_i^j \) is greater than that threshold. This would make the resultant function non-differentiable.
To solve this problem, in this work, we propose the use of the following approximations:

a) $|\theta_i| \approx \sum_{j=1}^{n} o^j_i$. The number of classes in the prediction for sample $i$, $|\theta_i|$, is approximated as the sum of the values of all the neurons in the output layer of the network, $\sum_{j=1}^{n} o^j_i$.

b) $|\gamma_i| = \sum_{j=1}^{n} y^j_i$. The number of classes in the reference for sample $i$, $|\gamma_i|$, is computed as the sum of the values in $y_i$, $\sum_{j=1}^{n} y^j_i$.

c) $|\gamma_i \cap \theta_i| \approx \sum_{j=1}^{n} y^j_i \cdot o^j_i$. The number of correctly predicted classes for sample $i$, $|\gamma_i \cap \theta_i|$, is approximated as the weighted sum of the output layer and the vector of correct classes for sample $i$, $\sum_{j=1}^{n} y^j_i \cdot o^j_i$.

d) $|\gamma_i \cup \theta_i| \approx \sum_{j=1}^{n} \left(y^j_i + o^j_i - y^j_i \cdot o^j_i\right)$. Applying the set theory for calculating the cardinality of a union set, the normalization factor of the Accuracy, $|\gamma_i \cup \theta_i|$, is computed using the three previous approximations.

e) $\sum_{i=1}^{m} [c_j \in \theta_i] \approx \sum_{i=1}^{m} o^j_i$. The number of samples for which the class $c_j$ is predicted, $[c_j \in \theta_i]$, is approximated as the sum of the $j$th component of the output layer of the network for all samples, $\sum_{i=1}^{m} o^j_i$.

f) $\sum_{i=1}^{m} [c_j \in \gamma_i] = \sum_{i=1}^{m} y^j_i$. The number of samples with class $c_j$, $[c_j \in \gamma_i]$, is computed as the sum of the $j$th component of the vector of correct classes for all samples, $\sum_{i=1}^{m} y^j_i$.

g) $\sum_{i=1}^{m} [c_j \in \gamma_i \cap \theta_i] \approx \sum_{i=1}^{m} y^j_i \cdot o^j_i$. The number of samples with class $c_j$ correctly predicted, $[c_j \in \gamma_i \cap \theta_i]$, is approximated as the weighted sum of the $j$th component of the output layer and the vector of correct classes for all the samples, $\sum_{i=1}^{m} y^j_i \cdot o^j_i$.

Using these approximations, soft versions of the evaluation metrics can be defined. Equations 9, 10 and 11 present soft versions of Accuracy (SAcc), micro-$F_1$ (Sm-$F_1$) and macro-$F_1$ (SM-$F_1$). Note that the sign has been inverted because they are loss functions that should be minimized.

$$SAcc = -\frac{1}{m} \sum_{i=1}^{m} \frac{\sum_{j=1}^{n} (o^j_i \cdot y^j_i)}{\sum_{j=1}^{n} (o^j_i + y^j_i - o^j_i \cdot y^j_i)}$$

(9)

$$Sm-F_1 = -2 \cdot \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} (o^j_i \cdot y^j_i)}{\sum_{i=1}^{m} \sum_{j=1}^{n} (o^j_i + y^j_i)}$$

(10)

$$SM-F_1 = -2 \cdot \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} (o^j_i \cdot y^j_i)}{\sum_{i=1}^{m} \frac{\sum_{j=1}^{n} (o^j_i + y^j_i)}{n}}$$

(11)

These functions are approximations of Acc, $m-F_1$ and $M-F_1$ with the advantage of being able to work with continuous values of $o^j_i$. Nevertheless, when $o^j_i \in \{0, 1\}$, the evaluation metrics and their soft versions are equivalent.

In Section 4, devoted to the experimental work, a study of how the soft functions approximate the evaluation metrics is presented.

Soft functions have been defined to satisfy that they were differentiable functions. Equations (12), (13) and (14) show the derivatives of the three soft metrics (SAcc, Sm-$F_1$ and SM-$F_1$) with respect to $o^j_k$, $1 \leq k \leq m$, $1 \leq l \leq n$.

$$\frac{\partial \text{Acc}}{\partial o^j_k} = -\frac{1}{m} \cdot \frac{y^j_k \cdot \left(\sum_{j=1}^{n} (o^j_k + y^j_k) \cdot (\sum_{j=1}^{n} (o^j_k - y^j_k))^2\right)}{\left(\sum_{j=1}^{n} (o^j_k + y^j_k - o^j_k \cdot y^j_k)\right)^2}$$

(12)

$$\frac{\partial \text{Sm-F}_1}{\partial o^j_k} = -2 \cdot \frac{y^j_k \cdot \sum_{i=1}^{m} \sum_{j=1}^{n} (o^j_i + y^j_i) - \sum_{i=1}^{m} \sum_{j=1}^{n} (o^j_i \cdot y^j_i)}{\left(\sum_{i=1}^{m} \sum_{j=1}^{n} (o^j_i + y^j_i)\right)^2}$$

(13)
a multi-label classification task, each tweet can have
Jaccard index) as defined in Equation 1. Since this is
the teams in E-c task was multi-label accuracy (or
Spanish.
labels per tweet is about 2.3 for English and 1.7 for
figures, it can be inferred that the average number of
is also showed the total number of tweets. From these
teams must use for their machine-learning systems. It
of training, development, and test that participating
number of samples per emotion in the partitions
emotions tags.
Table 1 shows the details of this corpus both for
English and Spanish languages. It can be seen the
number of samples per emotion in the partitions
of training, development, and test that participating
teams must use for their machine-learning systems. It
is also showed the total number of tweets. From these
figures, it can be inferred that the average number of
labels per tweet is about 2.3 for English and 1.7 for
Spanish.

The official competition metric used for ranking
the teams in E-c task was multi-label accuracy (or
Jaccard index) as defined in Equation 1. Since this is
a multi-label classification task, each tweet can have
one or more gold emotion labels, and one or more
predicted emotion labels. Apart from the official com-
petition metric (multi-label accuracy), we also report
micro-averaged F-score and macro-averaged F-score
in order to validate our proposal.

4. System description

In this section, we present the main characteristics
of the system developed in this work. Concretely,
we discuss the tweet preprocessing, the external
resources used to add polarity/emotion information
to the model, the tweet representation, and the Deep
Learning architecture.

We applied a tweet preprocessing that consisted in
a tokenization process by means of the TweetMotif
package. Then, we applied a normalization step
consisted in lowering the words, and in addition,
for the Spanish language we removed some language-
specific characters, for example, accents, dieresis,
specific language characters (i.e. “ñ” character in
Spanish language), etc.

Moreover, we carried out a normalization process
over unicode emoticons. This process can be useful
due to the great variability of the emoticons. In addition,
the embedding models used in this work did not
included these symbols. To solve this, we replaced
the unicode emoticons by their short name extracted
from the Unicode Common Locale Data Repository,
which contains a textual description of the emoticon’s
shape, e.g. ☺ → “Slightly Smiling Face”. It’s impor-
tant to notice that similar emoticons have similar
textual descriptions, consequently, it makes possible
to establish relationships among them by using their
descriptions.

Regarding to the external resources, we used
distributed representations of words, concretely,
embeddings obtained using Word2Vec [13, 14] mod-
els, in order to consider the semantic/contextual
information of each token. Moreover, we used several
lexicons to consider polarity/emotion information.
This polarity/emotion information was combined
with the embeddings of the words.

On the one hand, for the English task, we used
the following lexicons: AFFIN [24], Bing Liu’s
Opinion [8], MPQA [34], Sentiment140 [5], Sen-
tiWordnet [1], NRC Emotion Lexicon [18], NRC
As word embeddings, we used the pretrained model
by [6] with more than 400 million English tweets.
On the other hand, for the Spanish task, we used
the following lexicons: ELHPolar [29], ISOL [21], MLSenticon [3] and the Spanish version of NRC Emotion Lexicon. As word embeddings, we trained a skip-gram model, with 300 dimensions for each word, from 87 million Spanish tweets collected for the experimental work presented in this paper.

We represented each tweet $x$ as a matrix $S \in \mathbb{R}^{n \times (d+i)}$, where $n$ is the maximum number of words per tweet, $d$ is the dimensionality of word embeddings and $i$ is the dimensionality of the polarity/emotion features. In order to obtain this representation, we use an embedding model $h(w) \in \mathbb{R}^d$ and a set of lexicons $h'(w) = [h_1(w), h_2(w), \ldots, h_i(w)] \in \mathbb{R}^i$ (note that all the features extracted from lexicons, for the word $w$, are concatenated).

Therefore, given a tweet $x$ with $n$ tokens, $x = w_1, w_2, \ldots, w_n$, we represented it as a matrix $S$ in which, each row $i$ is the concatenation of the embedding of $w_i$ ($h(w_i)$) and a vector with the polarity values of $w_i$ in each lexicon ($h(w_i)$), $S = [h(w_1), h(w_2), \ldots, h(w_n)]$. If a word $w_i$ is out of vocabulary for the embedding models, we replace its embedding by the embedding of the word “unknown”, $h(w_i) = h(unknown)$. Similarly, if $w_i$ is not included in any lexicon, $h'(w_i) = [0, 0, \ldots, 0] \in \mathbb{R}^i$. Given a tweet, it can belong to several classes, from the set of classes $C$, in an independent way i.e. $y = \{y_1, y_2, \ldots, y_{|C|} : y_i \in \{0, 1\}\}$ with $|C| = 11$.

Moreover, due to the variable length of the tweets, we used zero padding at the start of a tweet if it does not reach the maximum specified length. Otherwise, if the length of a tweet is greater than the maximum, we truncated the tweet at the end. In the English task the average length of words per tweet is $n_{avg} = 19$, and the maximum length is $n_{max} = 85$. We decided to set the length $n$ as the mean of $n_{avg}$ and $n_{max}$. In the Spanish task, $n_{avg} = 16$, $n_{max} = 82$, therefore $n = \frac{n_{avg} + n_{max}}{2} = 49$.

Regarding to the Deep Learning system, we used a Convolutional Neural Network (CNN) architecture inspired in [35], which has shown competitive results in several text classification tasks such as Customer Review Dataset [8], Opinion Polarity [33] and Irony Detection [32]. We used this Deep Learning system for all the experiments conducted.

Following the CNN architecture, we applied one-dimensional convolutions with variable height filters in order to extract the temporal structure of the tweet over several region sizes. Note that the width of the filters is constant and equal to $d$ and we only modify the height of the filters.

Fig. 1 summarizes the model configuration and the hyper-parameters that we used in all the experimental work. As can be seen, we used 6 different region sizes (the filter height ranges from 1 to 6) and 128 filters for each region size. A total of 768 different filters were used in this architecture. After applying the filters, we obtained 128 output feature maps for each region size. Note than the output feature maps have length $n$ due to we used a “same padding” scheme.

In order to extract the most salient features for each region size, we applied 1D Global Max Pooling to the feature maps of each region size. Therefore, we obtained 6 vectors with 128 components, that were concatenated and used as input to a fully-connected layer which performs the multi-label classification decision. We used sigmoid activation functions to model the probability of each class independently of the probability of the other classes.

### Table 1

<table>
<thead>
<tr>
<th>Emotion</th>
<th>English</th>
<th></th>
<th></th>
<th></th>
<th>Spanish</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Train</td>
<td>Dev</td>
<td>Test</td>
<td>$\sum$</td>
<td>Train</td>
<td>Dev</td>
<td>Test</td>
<td>$\sum$</td>
</tr>
<tr>
<td>Anger</td>
<td>2544</td>
<td>315</td>
<td>1101</td>
<td>3960</td>
<td>1155</td>
<td>209</td>
<td>919</td>
<td>2283</td>
</tr>
<tr>
<td>Anticipation</td>
<td>978</td>
<td>124</td>
<td>425</td>
<td>1527</td>
<td>415</td>
<td>94</td>
<td>321</td>
<td>830</td>
</tr>
<tr>
<td>Disgust</td>
<td>2602</td>
<td>319</td>
<td>1099</td>
<td>4020</td>
<td>521</td>
<td>98</td>
<td>423</td>
<td>1042</td>
</tr>
<tr>
<td>Fear</td>
<td>1242</td>
<td>121</td>
<td>485</td>
<td>1848</td>
<td>373</td>
<td>74</td>
<td>298</td>
<td>745</td>
</tr>
<tr>
<td>Joy</td>
<td>2477</td>
<td>400</td>
<td>1442</td>
<td>4319</td>
<td>1087</td>
<td>201</td>
<td>873</td>
<td>2161</td>
</tr>
<tr>
<td>Love</td>
<td>700</td>
<td>132</td>
<td>516</td>
<td>1348</td>
<td>261</td>
<td>55</td>
<td>245</td>
<td>561</td>
</tr>
<tr>
<td>Optimism</td>
<td>1984</td>
<td>307</td>
<td>1143</td>
<td>3434</td>
<td>378</td>
<td>66</td>
<td>278</td>
<td>722</td>
</tr>
<tr>
<td>Pessimism</td>
<td>795</td>
<td>100</td>
<td>375</td>
<td>1270</td>
<td>578</td>
<td>115</td>
<td>495</td>
<td>1188</td>
</tr>
<tr>
<td>Sadness</td>
<td>2008</td>
<td>265</td>
<td>960</td>
<td>3233</td>
<td>845</td>
<td>143</td>
<td>644</td>
<td>1632</td>
</tr>
<tr>
<td>Surprise</td>
<td>361</td>
<td>35</td>
<td>170</td>
<td>566</td>
<td>169</td>
<td>37</td>
<td>122</td>
<td>328</td>
</tr>
<tr>
<td>Trust</td>
<td>357</td>
<td>43</td>
<td>153</td>
<td>553</td>
<td>175</td>
<td>31</td>
<td>122</td>
<td>328</td>
</tr>
<tr>
<td>$\sum$</td>
<td>16048</td>
<td>2161</td>
<td>7869</td>
<td>26078</td>
<td>5951</td>
<td>613</td>
<td>4740</td>
<td>11820</td>
</tr>
<tr>
<td>#samples</td>
<td>6838</td>
<td>886</td>
<td>3259</td>
<td>10983</td>
<td>3561</td>
<td>679</td>
<td>2854</td>
<td>7094</td>
</tr>
</tbody>
</table>
Moreover, with the aim of improving the generalization and speed up the model convergence, we used BatchNormalization [9] after each convolution and after the input layer. To achieve non-linearity after each convolution, we used ReLU [23] activation functions. As optimization algorithm, we used RMSprop to learn the parameters of the network with respect to the proposed loss functions.

Given the previous convolutional network architecture, \( f: \mathbb{R}^{n \times (d + v)} \rightarrow \mathbb{R}^C \), the steps to assign a set of classes to a tweet \( x \) are the following: first we obtain its representation matrix \( S \); second, we make a forward pass in order to obtain the probability that \( x \) belongs to each class independently, \( f(x) = \{o_1, o_2, ..., o_n\} \); and finally, we consider that the tweet \( x \) belongs to class \( j \) if \( f(x)_j \geq 0.5 \).

5. Experimental results

In this section, we present the experimental work conducted in order to evaluate the performance of the loss functions proposed in Section 2. We study the impact of these loss functions on the overall results on the Emotion Classification task proposed at SemEval2018 competition.

A parameter that has a high impact on the calculation of the loss functions is the batch size. Therefore, we first studied the behavior of the functions by varying the batch size.

This study was carried out on the development sets for English and Spanish defined in Section 3 (see Table 1). We considered 30 training epochs and the SAcc, SM-F1 and Sm-F1 loss functions. Fig. 2 shows the achieved results for English corpora. It can be seen the values of the evaluation metric per epoch on the development set for the CNN trained with the SAcc, SM-F1, and Sm-F1 loss functions, respectively, by varying the batch size \( b \in \{16, 32, 64, 128\} \).

As it can be seen in Fig. 2, in all cases, lower values of \( b \) produce faster convergence, that is, the maximum value of the evaluation metric is obtained in initial epochs. We hypothesized that, the faster convergence is because more updates are made in each training epoch. Moreover, in all cases, the maximum value of the evaluation metric is obtained with the minimum batch size considered, \( b = 16 \). It can be also observed that generally, the batch sizes that obtained the best results were \( b = 16 \) and \( b = 32 \).

In addition, we can also observe that with largest batch sizes, \( b = 128 \), the model takes more time to converge, but when it becomes stable, the results were similar compared to those obtained with smaller batch sizes.

We performed a similar experimentation for the Spanish corpora and we have observed a similar
behavior than in the English corpora, that is, lower batch sizes produce faster convergence.

From the performed study of how influence batch size on loss function optimization, we can conclude that the best size is $b = 16$, both for the English and Spanish data sets.

Once the best value of the batch size was set, our objective was to compare the performance of the system trained using the loss functions proposed in this paper with the performance obtained using BCE and MSE as loss functions, on the test sets of the Emotion Classification task. To make this comparison more accurate, the confidence intervals of the three metrics used to evaluate the systems in the competition were computed.

In order to compute the confidence intervals, we used the Bootstrap Confidence Intervals [22] approach. First, from the set of hypotheses provided by the system that we want to evaluate, we generated up to 5000 resamples by sampling with replacement from this original set of hypotheses. Each resample had the same size of the original set. Next, the value
of the evaluation measure is calculated for each of the resamples. Finally, we compute the 95% confidence interval using the bootstrap distribution.

Tables 2 and 3 show the results obtained by our systems both for the English and the Spanish corpora. An asterisk (*) on values means that these values are statistically significant.

It can be seen that the models trained with the proposed loss functions obtained the best results. This occurs both for English and Spanish test sets with significant improvements in all cases except when evaluating with $m-F_1$ on the Spanish test set. In addition, it can be observed how, generally, the model trained with the loss function that approximates the evaluation metric used, obtained the best results for that evaluation metric. This is true for all the cases studied, except for the $m-F_1$ measure for Spanish in which it achieved the second-best result (note that this is the only case where the improvements are not significant). Compared with the official results of the competition\(^2\) our best result achieved the 1st position of 13 teams for Spanish and the 7th place of 34 teams for English.

Next, we present an analysis of the performance of our system at class level. Tables 4 and 5 show the results of $P$, $R$, and $F_1$ for all classes with the CNN trained using SM-$F_1$ and Sm-$F_1$ as loss function respectively.

From the results for the English corpus, it can be observed that the system trained with SM-$F_1$ achieved higher values of $F_1$ for the minority classes, e.g. Trust ($F_1 = 17.74$ on development and $F_1 = 24.79$ on test) and Surprise ($F_1 = 19.34$ on development and $F_1 = 26.02$ on test), compared to those obtained by the system trained with Sm-$F_1$ (for Trust: $F_1 = 2.47$ on development and $F_1 = 15.19$ on test).

\(^2\)https://competitions.codalab.org/competitions/17751#results
and for Surprise: $F_1 = 10.58$ on development and $F_1 = 16.56$ on test). This occurs because of SM-$F_1$ consider the performance of all the classes in the same way, independently of the number of samples per class. In contrast, Sm-$F_1$ favors majority classes. A similar behavior can also be observed in the results for the Spanish corpus. This is an important aspect to keep in mind in the design of a classification system.

Finally, an analysis of how the loss functions approximate the evaluation metrics is presented. For the English corpus, Fig. 3 shows loss function and the evaluation metric on training, development and test sets for the CNN trained using SAcc, SM-$F_1$, and Sm-$F_1$, respectively.

It can be observed how, throughout all the training epochs, the proposed loss functions follow the same trend as the evaluation metrics. This seems to indicate that they are good approximations to the evaluation metrics and consequently, we can make decisions about the evaluation metrics by looking only at the values obtained by the loss functions.

Another interesting aspect to note is that the loss functions can be considered as a lower bound of the evaluation metrics. This can be observed in all the previous cases, since, for the different epochs, the loss functions never overestimate the values of the evaluation metric.

In addition, it draws attention how SM-$F_1$, although it correctly estimates the trend of the metric $M_F$ throughout the different epochs, obtains values much smaller than these ones. On the other hand, the SAcc and Sm-$F_1$ loss functions, follow the trend of the evaluation metrics to which they approximate and they obtain values more similar to those obtained by those metrics. A similar behavior was observed in all the partitions for Spanish language.
6. Conclusions and future work

In this paper we have proposed soft versions of the Accuracy, micro-$F_1$, and macro-$F_1$ measures that can be used as loss functions in the learning process of Deep Learning models.

The proposed method imposes minor changes in the training phase, only the change of the loss function is required in the training process.

We conducted an extensive experimental work on a multi-label text classification task in order to validate our approach. In all the experiments conducted on the "E-c: Detecting Emotions Multilabel classification" task proposed at Semeval-2018, when loss functions adapted to the metrics were used, the results were improved compared to those obtained using classical loss functions and this improvement was statistically significant at 95% confidence level.

These results for the Spanish language outperformed the best results published at Semeval-2018 competition. Competitive results for the English language were also obtained.

We verified that the convergence of the models trained with the proposed loss functions is faster, i.e. the best results are obtained with fewer epochs, in comparison with classical loss functions. Moreover, we observed that the proposed loss functions adequately approximates the evaluation metrics. This is interesting because more robust models can be obtained, using these loss functions, with respect to the evaluation metrics.

In this work we only addressed some evaluation metrics that are commonly used in multi-label text classification tasks. However, following the proposal of this work, it would be possible to extend this technique to other evaluation metrics as loss functions.

As future work, we plan to do a deeper study about the properties of the loss functions introduced in this work.

Firstly, because we softened the evaluation metrics with the aim of making them differentiable, there are differences between the softened version (loss function, before the discretization process) and the evaluation metric (after the discretization process). We plan to explore how to minimize these differences by using some kind of regularization on the predictions.

Finally, it is interesting to explore the capacity of our loss functions to address other tasks with different characteristics and difficulties such as extremely imbalanced classification problems.

Acknowledgments

This work has been partially supported by the Spanish MINECO and FEDER funds under project AMIC (TIN2017-85854-C4-2-R) and the GiSPRO project (PROMETEU/2018/176). Work of José-Ángel González is also financed by Universitat Politècnica de València under grant PAID-01-17.

References


