

LSTM-Attention Architecture for Online Bilingual Sexism Detection

Notebook for the EXIST Lab at CLEF 2023

Srinivasa Ravi^{1,†}, Siddharth Kelkar^{1,†} and Anand Kumar Madasamy¹

¹National Institute of Technology Karnataka, Surathkal, Mangaluru, India

Abstract

The paper describes the results submitted by ‘Team-SMS’ at EXIST 2023. A dataset of 6920 tweets for training, 1038 for validation, and 2076 tweets for testing was provided by the task organizers to train and test our models. Our models include LSTM models coupled with attention layers and without attention. For calculation of soft scores according to the task we tried to mimic human performance by taking an average of different machine learning model predictions using Multinomial Naive Bayes, Linear Support Vector Classifier, Multi Layer Perceptron, XGBoost, LSTM using GloVe embeddings, and LSTM using fastText embeddings. We discuss our approach to remove the ambiguity in the labeling process and detailed description of our work.

Keywords

LSTM, Attention, GloVe, fastText, Sexism Identification, Classification

1. Introduction

Sexism, a pervasive form of gender-based discrimination, continues to be a critical social issue across the globe. It manifests in various contexts, such as workplaces, educational institutions, and online platforms, reinforcing harmful stereotypes and limiting opportunities for individuals based on their gender. As the world becomes increasingly interconnected through digital platforms and social media, detecting and addressing sexism in online content has become a pressing concern.

Deep learning, a subset of machine learning, has emerged as a powerful tool for tackling complex tasks by automatically learning intricate patterns and representations from vast amounts of data. Leveraging the capabilities of deep learning, researchers and technologists have been exploring innovative approaches to develop automated systems capable of detecting and combating sexism in online communications.

The shared tasks on sEXism Identification in Social neTworks (EXIST) at CLEF 2023 [1, 2] represent a systematic benchmark that attempts to tackle this challenge via machine learning and Natural Language Understanding (NLU). Also, the assumption that natural language expressions

CLEF 2023: Conference and Labs of the Evaluation Forum, September 18–21, 2023, Thessaloniki, Greece

[†]Equal contribution.


✉ srinivasar.211it070@nitk.edu.in (S. Ravi); siddharthkelkar.211it067@nitk.edu.in (S. Kelkar);

m_anandkumar@nitk.edu.in (A. K. Madasamy)

ORCID [0009-0002-6133-5253](https://orcid.org/0009-0002-6133-5253) (S. Ravi); [0009-0002-4010-0535](https://orcid.org/0009-0002-4010-0535) (S. Kelkar); [0000-0003-0310-4510](https://orcid.org/0000-0003-0310-4510) (A. K. Madasamy)



© 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)

have a single and clearly identifiable interpretation in a given context is a convenient idealization, but far from reality, especially in highly subjective tasks such as sexism identification. The learning with disagreements paradigm in this shared task aims to deal with this by letting systems learn from datasets where no gold annotations are provided but information about the annotations from all annotators, in an attempt to gather the diversity of view.

This paper presents our contribution to the tasks, algorithms to extract useful information from multiple annotations, describes our overall approach and the methods and models applied and summarizes the obtained results. We summarize our results for two tasks: *Sexism Identification* (Task 1) and *Source Intention* (Task 2).

2. Dataset Preprocessing

2.1. Ambiguity removal

A major challenge of the EXIST tasks [1, 2] is the small size of dataset (around 7000 tweets) which is uniformly divided into English and Spanish tweets. This small size provides difficulty in robust training of any model. Apart from that around 10% of the annotated tweets have ambiguous labels i.e. the 6 annotations provided for the label have no clear majority which if not processed further reduces the dataset size. To counter this, we formulated the following algorithms to calculate a hard label from the annotations.

Trust score

The ‘trust score’ for each of the annotators in the training data can be calculated by finding the ratio of correct annotations (from the tweets having majority labels) to total annotations by each annotator. The result is stored in a dictionary. This provides us a metric to remove ambiguities from the annotations and give hard labels to each ambiguous tweet by giving priority to annotations of annotators with higher trust score. The distribution for each annotator is uniformly distributed in the dataset thus removing any chances of bias. The generalized algorithm (Algorithm 1) to calculate the Trust Score in any similar dataset is presented here.

To further generalize this, according to the info provided in the training data about the annotators, they can be grouped into 6 classes with respect to gender and age (18-22F, 23-45F, 46+F, 18-22M, 23-45M, and 46+M). The individual trust scores are grouped into these classes and their average is taken to calculate the class-wise trust score. This gives us another metric to remove ambiguity from the tweets. The generalized algorithm (Algorithm 2) to do this is presented here.

Ambiguity resolution

Once the individual and class-wise trust scores are calculated, they can be used to resolve ambiguity by labelling ambiguous tweets. Either the individual trust scores or the class-wise trust scores can be used for this purpose. For each annotator giving a ‘YES’ or ‘NO’ annotation, the respective trust score (individual or class-wise) is added to the classes. The class having higher score will act as the label. This provides a way to remove ambiguity from the tweets

Algorithm 1: Individual Trust Score

Data: Dataset \mathcal{D} with N samples,

Each sample classified (into two class types, A and B) by a set of entities (entity $e \in \mathcal{E}$, $e : \mathcal{D} \rightarrow \{A, B\}$), $E_d(\subset \mathcal{E}) \forall d \in \mathcal{D}$ from the set of all entities \mathcal{E}

Result: $T : \mathcal{E} \rightarrow \mathbb{R}$, where $T(e) = \text{Trust Score of } e \forall e \in \mathcal{E}$,

Set of successfully classified samples \mathcal{D}_A and \mathcal{D}_B ,

Set of ambiguous samples $\mathcal{D}_{\text{ambi}}$

Algorithm:

$\mathcal{D}_A \leftarrow \phi$

$\mathcal{D}_B \leftarrow \phi$

foreach $e \in \mathcal{E}$ **do**

$\mathcal{N}(e) \leftarrow 0$

$T(e) \leftarrow 0$

foreach $d \in \mathcal{D}$ **do**

$n_A \leftarrow 0$

$n_B \leftarrow 0$

foreach $e \in E_d$ **do**

$\mathcal{N}(e) \leftarrow \mathcal{N}(e) + 1$

if $e(d) = A$ **then**

$n_A \leftarrow n_A + 1$

else

$n_B \leftarrow n_B + 1$

if $n_A > n_B$ **then**

$\mathcal{D}_A \leftarrow \mathcal{D}_A \cup \{d\}$

else if $n_B > n_A$ **then**

$\mathcal{D}_B \leftarrow \mathcal{D}_B \cup \{d\}$

foreach $d \in \mathcal{D}$ **do**

if $d \in \mathcal{D}_A$ **then**

$C \leftarrow A$

else

$C \leftarrow B$

foreach $e \in E_d$ **do**

if $e(d) = C$ **then**

$T(e) \leftarrow T(e) + \frac{1}{\mathcal{N}(e)}$

and helps in retaining the training data. The generalized algorithm (Algorithm 3) to do this is presented here.

After multiple runs, using the class-wise trust score for this purpose appears to perform better than using the individual scores, probably because the former provides more generalization and removes any individual bias.

Algorithm 2: Class-Wise Trust Score

Data: Set of all entities, \mathcal{E} ,

Set of all entity classes, \mathcal{C} ,

$S : \mathcal{E} \rightarrow \mathcal{C}$, where $S(e) =$ The entity class e belongs to $\forall e \in \mathcal{E}$,

$T : \mathcal{E} \rightarrow \mathbb{R}$, where $T(e) =$ Trust Score of $e \forall e \in \mathcal{E}$

Result: $T_g : \mathcal{C} \rightarrow \mathbb{R}$, where $T_g(c) =$ Class-Wise Trust Score of $c \forall c \in \mathcal{C}$

Algorithm:

foreach $c \in \mathcal{C}$ **do**

$T_g(c) \leftarrow 0$

$N(c) \leftarrow 0$

foreach $e \in \mathcal{E}$ **do**

$N(S(e)) \leftarrow N(S(e)) + 1$

$T_g(S(e)) \leftarrow T_g(S(e)) + T(e)$

foreach $c \in \mathcal{C}$ **do**

$T_g(c) \leftarrow \frac{T_g(c)}{N(c)}$

Algorithm 3: Ambiguity Resolution

Data: Dataset \mathcal{D} , \mathcal{E} , $E_d \forall d \in \mathcal{D}$ (same definitions as in Algorithm 1),

Set of successfully classified samples \mathcal{D}_A and \mathcal{D}_B ,

Set of ambiguous samples $\mathcal{D}_{\text{ambi}}$,

$T : \mathcal{E} \rightarrow \mathbb{R}$, where $T(e) =$ Trust Score of $e \forall e \in \mathcal{E}$,

Classification bias $\beta \in \{A, B\}$

Result: Successfully classified \mathcal{D} into \mathcal{D}_A and \mathcal{D}_B , such that $\mathcal{D}_A \cup \mathcal{D}_B = \mathcal{D}$

Algorithm:

foreach $d \in \mathcal{D}_{\text{ambi}}$ **do**

$s_A \leftarrow 0$

$s_B \leftarrow 0$

foreach $e \in E_d$ **do**

if $e(d) = A$ **then**

$s_A \leftarrow s_A + T(e)$

else

$s_B \leftarrow s_B + T(e)$

if $s_A > s_B$ **then**

$\mathcal{D}_A \leftarrow \mathcal{D}_A \cup \{d\}$

$\mathcal{D}_{\text{ambi}} \leftarrow \mathcal{D}_{\text{ambi}} - \{d\}$

else if $s_B > s_A$ **then**

$\mathcal{D}_B \leftarrow \mathcal{D}_B \cup \{d\}$

$\mathcal{D}_{\text{ambi}} \leftarrow \mathcal{D}_{\text{ambi}} - \{d\}$

foreach $d \in \mathcal{D}_{\text{ambi}}$ **do**

$\mathcal{D}_\beta \leftarrow \mathcal{D}_\beta \cup \{d\}$

$\mathcal{D}_{\text{ambi}} \leftarrow \mathcal{D}_{\text{ambi}} - \{d\}$

2.2. Text preprocessing

The following preprocessing steps were performed on each tweet:

1. **Stopword Removal:** The function `stopword_removal` is applied to each row of the 'tweet' column using the `apply` method. It is assumed that the `stopword_removal` function removes commonly used words (stopwords) from the text, which are typically irrelevant for natural language processing tasks. This was done with the help of the NLTK [3] library.
2. **Lowercasing:** The `lower()` method is applied to the 'tweet' column, converting all the text to lowercase. This step is performed to ensure consistency in the text and to avoid treating words with different cases as different entities.
3. **URL Removal:** The regex, `r'https?:\/\/\/\S+'` is used to match and remove URLs from each tweet. It looks for patterns starting with 'http://' or 'https://' followed by any non-whitespace characters. This step aims to eliminate URLs, which are often irrelevant for text analysis or modeling.
4. **Website Removal:** The regex, `r'www\.[a-z]?\.?(com)+|[a-z]+\.(com)'` is used to match and remove website addresses from the tweets. It searches for patterns starting with 'www.' followed by an optional single letter and optional 'com', or it matches any combination of lowercase letters followed by '.com'. This step targets website links that were not captured by the previous URL removal step.
5. **HTML Entity Removal:** The regex, `r'&[a-z]+;'` matches and removes HTML entities from the tweets. HTML entities are special characters represented using codes like '&' for '&', '<' for '<', etc. This step is useful when dealing with text scraped from websites or social media platforms.
6. **Special Character and Symbol Removal:** `r'^a-z\s\(\-:\)\|\;=\`#\'` is the regex used to match and remove any character that is not a lowercase letter, whitespace, parentheses, hyphen, colon, backslash, forward slash, semicolon, equal sign, single quote, or hash symbol. This step removes special characters and symbols that might not carry significant meaning for text analysis or modeling.
7. **Mention Removal:** The regex, `r'@mention'` matches and removes the string '@mention' from the tweets. This step targets mentions of users or handles often found in social media text.

In summary, the code applies a series of text preprocessing steps to clean the 'tweet' column. These steps include stopword removal, lowercasing, URL and website removal, HTML entity removal, special character and symbol removal, and mention removal. The resulting 'tweet' column should contain cleaned and standardized text that is ready for further analysis or modeling tasks.

3. Evaluation metrics

For all tasks and all types of evaluation (hard-hard, hard-soft, and soft-soft), the official metric: ICM [4] is used. ICM is a similarity function that generalizes Pointwise Mutual Information (PMI) and can be used to evaluate system outputs in classification problems by computing their

similarity to the ground truth categories. As there is not any current metric that fits hierarchical multi-label classification problems in a learning with disagreement scenario, the organizers have defined an extension of ICM (ICM-soft) that accepts both soft system outputs and soft ground truth assignments.

1. **Hard-Hard evaluation:** For systems that provide a hard, conventional output, the organizers provided a hard-hard evaluation. To derive the hard labels in the ground truth from the different annotators' labels, a probabilistic threshold was computed for each task. As a result, for Task 1, the class annotated by more than 3 annotators is selected; and for Task 2, the class annotated by more than 2 annotators is selected. Items for which there is no majority class (i.e. no class receives more probability than the threshold) are removed in this evaluation scheme. The official metric is the original ICM [4]. The systems are also compared with F1 (the harmonic average of precision and recall). In Task 1, F1 is used for the positive class, and in Task 2, the average of F1 for all classes is used.
2. **Hard-Soft evaluation:** For systems that provide a hard output, the organizers provided a hard-soft evaluation, comparing the categories assigned by the system with the probabilities assigned to each category in the ground truth. ICM-soft is the official evaluation metric in this variant. The probabilities of the classes for each instance are calculated according to the distribution of labels and the number of annotators for that instance.
3. **Soft-Soft evaluation:** For systems that provide probabilities for each category, the organizers provided a soft-soft evaluation that compares the probabilities assigned by the system with the probabilities assigned by the set of human annotators. As in the previous case, ICM-soft is used as the official evaluation metric in this variant.

4. Task 1: Sexism Identification

The 'Sexism Identification' task involved classifying tweets as being sexist or not. To do this, we employed the usage of various simple machine learning models and neural networks and analyzed the results.

4.1. Simple models

We tested five models, namely Multinomial Naive Bayes [5, 6], Linear SVC [7, 6], Bernoulli Naive Bayes [5, 6], Multilayer Perceptron [8, 6] and XGBoost [9]. The evaluation was done by removing ambiguous tweets from the validation set and the metric of the evaluation is the ICM and F1 score using the hard-hard evaluation mode of the python script provided. All implementations considered a 5-fold cross-validation process during the training stage. A hyperparameters analysis was conducted for all models, considering: different learning rates, alpha, beta, hidden layer sizes for the models for which they are applicable and optimal parameters were found. The results are detailed in Table 1. XGBoost [9], Multilayer Perceptron [8, 6] and Bernoulli Naive Bayes [5, 6] appear to perform significantly better than the others with ICM [4] scores of around 0.27 as compared to scores of 0.2026 and 0.2370 for Multinomial Naive Bayes [5, 6] and Linear SVC [7, 6] respectively.

Table 1

Results without applying ambiguity resolution

Model	ICM Score [4]	F1:YES	F1:NO	MARCO:F1
Multinomial Naive Bayes	0.2026	0.7021	0.7623	0.7322
Linear SVC	0.2370	0.7072	0.7778	0.7425
Bernoulli Naive Bayes	0.2714	0.7378	0.7753	0.7566
XGBoost	0.2740	0.7344	0.7796	0.7570
Multilayer Perceptron	0.2641	0.7286	0.7782	0.7534

Table 2

Results after applying ambiguity resolution

Model	ICM Score [4]	F1:YES	F1:NO	MARCO:F1
Multinomial Naive Bayes	0.1899	0.6987	0.7575	0.7281
Linear SVC	0.2249	0.7343	0.7492	0.7418
Bernoulli Naive Bayes	0.2361	0.7254	0.7642	0.7488
XGBoost	0.2515	0.7264	0.7725	0.7495
Multilayer Perceptron	0.2380	0.7163	0.7722	0.7443

The same models were also run after applying the ambiguity resolution algorithm with the class-wise trust scores. The results for this are detailed in Table 2.

As seen in the results after applying ambiguity resolution, the accuracy decreases marginally with an average reduction in the ICM [4] score of 8.49% across the models. So, it is not helping us to extract more useful features by increasing the training data. In the future, the algorithms could be improved by using a more complex combination of metrics to group and compute scores to increase the accuracy. These models are not directly part of the main submission but are used for the calculation of soft scores for Task 1. We tried to replicate human annotator performance using these models and taking a probability of the predictions spread across these models.

4.2. Soft-score calculation

The assumption that natural language expressions have a single and clearly identifiable interpretation in a given context is a convenient idealization, but far from reality, especially in highly subjective task as sexism identification. The learning with disagreements paradigm of EXIST 2023 [1, 2] aims to deal with this by letting systems learn from datasets from information about the annotations from all annotators, in an attempt to gather the diversity of views. Taking this into consideration for soft score calculation we sought to replicate human annotator expertise through the utilization of various models. The soft score calculation involved aggregating prediction probabilities (For example, if 4 models predicted YES and 2 predicted NO then the soft score would be YES : 0.67, NO : 0.33) from six distinct models, namely Multinomial Naive Bayes [5, 6], Linear Support Vector Classifier [7, 6], Multilayer Perceptron [8, 6], XGBoost [9], LSTM [10, 6] using GloVe [11] word embeddings , and LSTM using word fastText [12] word

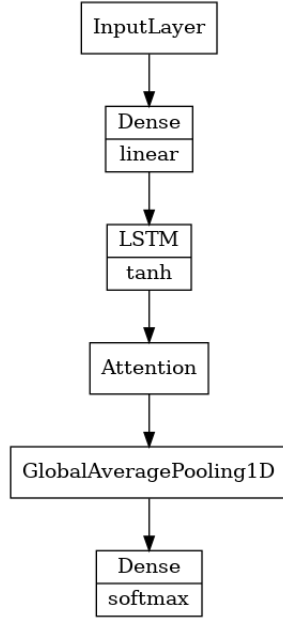


Figure 1: LSTM [10] + Attention [13] architecture for Task 1: Sexism Identification, via TensorFlow Keras [14, 6] `plot_model`.

embeddings. By leveraging the collective insights from these diverse models, we aimed to attain a robust and accurate soft score, mirroring the performance of human annotators. This approach allowed us to harness the strengths of each model and effectively gauge the quality of the text processing framework. The exact procedure is detailed in Algorithm 4.

Algorithm 4: Soft Score Calculation

Data: \mathcal{T} , pieces of text to be classified,

$m : \mathcal{T} \rightarrow \{\text{YES}, \text{NO}\} \in \mathcal{M}$, the set of models being used to classify

Result: Soft scores, \mathcal{S} , where $\mathcal{S} : \mathcal{T} \rightarrow \{\{Y, N\} \mid 0 \leq Y, N \in \mathbb{R} \leq 1, Y + N = 1\}$

Algorithm:

$N = n(\mathcal{M})$

foreach $t \in \mathcal{T}$ **do**

$Y = 0$

$N = 0$

foreach $m \in \mathcal{M}$ **do**

if $m(t) = \text{YES}$ **then**

$Y = Y + \frac{1}{N}$

else

$N = N + \frac{1}{N}$

$\mathcal{S}(t) = \{Y, N\}$

4.3. Deep learning models

We submitted a total for 3 runs for this task and used neural network models for all of them. The final models for all the runs were trained on both the training and validation data. The exact details are given below.

Model for Run 1

We employed 200-dimensional GloVe [11] Twitter word embeddings for English texts, while 300-dimensional fastText [12] word embeddings were utilized for Spanish texts. For English texts, the model architecture comprised of a dense [8] layer designed to convert the 200-dimensional embeddings into a more compact representation of 120 dimensions. Subsequently, a 120-unit LSTM [10] layer was incorporated, with a 20% dropout [15] rate to enhance generalization. To capture the contextual dependencies within the text, an attention [13] layer and a pooling [16] layer were included in the model. Finally, a dense layer with softmax [17] activation was employed to convert the representations into two dimensions (for classification), facilitating meaningful predictions. The model architecture is shown in Figure 1.

For the model for Spanish texts, a dense layer was used to convert the 300-dimensional embeddings to 150-dimensions. This was followed by a 150 unit LSTM layer with dropout followed by a Dense layer of 32 units with dropout and finally output (of 2 dimensions) using softmax activation. The soft score calculation involved the usage of Algorithm 4.

Model for Run 2

In this run, we employed 300-dimensional fastText [12] word embeddings for both English and Spanish texts. The model architecture was designed to maximize performance and accuracy, comprising a 150-unit LSTM [10] layer, followed by a Dropout [15] layer to mitigate overfitting. This was succeeded by a Dense [8] layer with 32 units, followed by another Dropout layer, ultimately culminating in the final output utilizing softmax [17] activation. This run employed same model architecture for both English and Spanish texts. The soft score prediction employed the same process of taking the probabilistic score of various model predictions.

Model for Run 3

The model and word embedding specifications for this run are the same as in Run 2. This run employed our ambiguity resolution algorithm using class-wise trust score to increase the data for training.

4.4. Results

Run 1 received a rank of 26/53 on soft-soft evaluation, 51/65 on hard-hard evaluation and 52/65 on hard-soft evaluation on ALL text instances. The significantly better soft-soft score indicates that such a method of employing various machine learning models to mimic human performance and creating soft probabilistic scores can be useful in future work in other tasks.

Run 2 received a rank of 27/53 on soft-soft evaluation, 52/65 on hard-hard evaluation and rank 53/65 on hard-soft evaluation on ALL text instances.

Table 3

Results for all runs submitted for Task 1 in the ALL INSTANCES category

Soft vs Soft				
Run	Rank	ICM-Soft	ICM-Soft Norm	Cross Entropy
SMS_1	26	0.3142	0.547	1.665
SMS_2	27	0.3142	0.547	1.665
Hard vs Hard				
Run	Rank	ICM-Hard	ICM-Hard Norm	F1
SMS_3	52	0.2469	0.5233	0.6717
SMS_1	54	0.2369	0.517	0.6787
SMS_2	55	0.2369	0.517	0.6787
Hard vs Soft				
Run	Rank	ICM-Soft	ICM-Soft Norm	
SMS_3	53	-0.5638	0.4052	
SMS_1	55	-0.6017	0.399	
SMS_2	56	-0.6017	0.399	

Table 4

Results for all runs submitted for Task 1 in the ES category

Soft vs Soft				
Run	Rank	ICM-Soft	ICM-Soft Norm	Cross Entropy
SMS_1	26	0.3061	0.506	1.6297
SMS_2	27	0.3061	0.506	1.6297
Hard vs Hard				
Run	Rank	ICM-Hard	ICM-Hard Norm	F1
SMS_3	52	0.1937	0.4663	0.6766
SMS_1	54	0.1809	0.4578	0.6888
SMS_2	55	0.1809	0.4578	0.6888
Hard vs Soft				
Run	Rank	ICM-Soft	ICM-Soft Norm	
SMS_3	53	-0.5495	0.3557	
SMS_1	56	-0.6433	0.3392	
SMS_2	57	-0.6433	0.3392	

Run 3 received a rank of 49/65 on hard-hard evaluation and rank 50/65 on hard-soft evaluation. This run with the algorithm performed marginally better which indicates that improving the algorithm may lead to better results. The complete results are detailed in Tables 3, 4 and 5.

The results for hard-hard and hard-soft evaluation are not very promising which indicates other training methods should be employed to get better results. Combined training rather than parallel training of Spanish and English texts could prove helpful (which we tried for runs of Task 2). On the other hand, the runs gave very promising results on the soft score metric which proves that such an attempt to employ various models to create soft score can be developed further to get better results.

Table 5
Results for all runs submitted for Task 1 in the EN category

Soft vs Soft				
Run	Rank	ICM-Soft	ICM-Soft Norm	Cross Entropy
SMS_1	27	0.253	0.5871	1.7046
SMS_2	28	0.253	0.5871	1.7046
Hard vs Hard				
Run	Rank	ICM-Hard	ICM-Hard Norm	F1
SMS_3	49	0.3015	0.5875	0.6658
SMS_1	51	0.2833	0.5764	0.665
SMS_2	52	0.2833	0.5764	0.665
Hard vs Soft				
Run	Rank	ICM-Soft	ICM-Soft Norm	
SMS_3	51	-0.641	0.4581	
SMS_1	52	-0.6411	0.4581	
SMS_2	53	-0.6411	0.4581	

5. Task 2: Source Intention

The ‘Source Intention’ task involved identifying whether a tweet was sexist or not and subsequently classifying the sexist tweets based on their source intention into Direct, Reported or Judgemental categories.

5.1. Deep learning models

We submitted a total of 3 runs for this task each of whose details are given below. The final models for runs 1 and 2 were trained on both training and validation data whereas the model for run 3 was trained only on the training data. The exact details of the models are given below.

Model for Run 1

For this run, we used 200-dimensional GloVe [11] embeddings for English and 300-dimensional fastText [12] embeddings for Spanish. For labelling, we selected the label which was in clear majority with respect to all the annotations. The model consists of two parallel paths. One of the paths starts with a dense [8] layer (size 200 for English and 300 for Spanish) which squeezes the input tensors into 50-dimensions. This is followed by a dropout [15] of 20%, followed by a 50-unit LSTM [10] layer with a dropout of 25%. This is finally passed to an attention layer whose scores are fed to the other path. The other parallel path is a replica of this path but without the final attention [13] layer. The output from the LSTM layer is concatenated with the attention scores from the other path. This is followed by pooling [16] and a dense layer of 4 units with softmax [17] activation for final classification. The model architecture is shown in Figure 2.

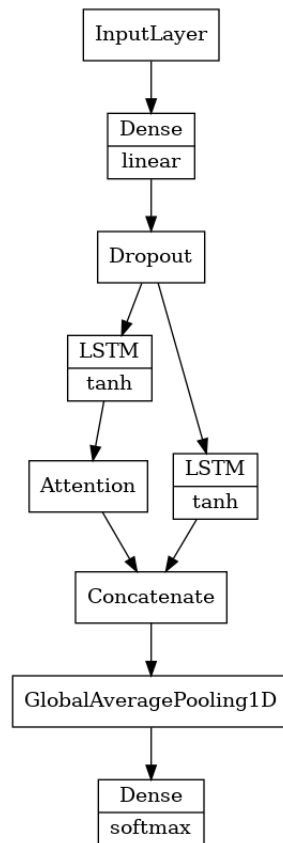


Figure 2: Parallel LSTM [10] + Attention [13] architecture for Task 2: Source Intention, via TensorFlow Keras [14, 6] p1ot_model.

Model for Run 2

For this run, 200-dimensional fastText [12] embeddings (reduced from 300-dimensions) were used for both Spanish and English texts and a single model was trained on both languages. The labelling procedure is the same as in Run 1. The model architecture is also identical to that of Run 1, with the slight difference that 80-unit LSTM [10] networks were used instead of 50-units.

Model for Run 3

This run is perfectly identical to Run 2 with only a slight difference in training methodology where the validation data set was not used to train the final model unlike the previous runs.

5.2. Results

Run 1 received a rank of 19/32 on hard-hard evaluation and 10/33 on hard-soft evaluation on ALL text instances. It also received a rank of 2/32 on hard-soft evaluation on ES text instances. The significantly better hard-soft evaluation performance indicates that such a model performs

Table 6

Results for all runs submitted for Task 2 in the ALL INSTANCES category

Hard vs Hard				
Run	Rank	ICM-Hard	ICM-Hard Norm	F1
SMS_1	19	-0.0892	0.6533	0.3654
SMS_3	20	-0.1226	0.6461	0.3504
SMS_2	23	-0.2571	0.6175	0.3246
Hard vs Soft				
Run	Rank	ICM-Soft	ICM-Soft Norm	
SMS_2	6	-5.627	0.6978	
SMS_3	8	-5.9579	0.6894	
SMS_1	10	-6.0774	0.6863	

Table 7

Results for all runs submitted for Task 2 in the ES category

Hard vs Hard				
Run	Rank	ICM-Hard	ICM-Hard Norm	F1
SMS_1	20	-0.1314	0.6185	0.328
SMS_3	21	-0.1314	0.6185	0.328
SMS_2	23	-0.2641	0.5892	0.3295
Hard vs Soft				
Run	Rank	ICM-Soft	ICM-Soft Norm	
SMS_1	2	-4.693	0.6871	
SMS_3	3	-4.693	0.6871	
SMS_2	8	-5.3008	0.6697	

Table 8

Results for all runs submitted for Task 2 in the EN category

Hard vs Hard				
Run	Rank	ICM-Hard	ICM-Hard Norm	F1
SMS_1	18	-0.0554	0.6949	0.3828
SMS_3	20	-0.1261	0.6805	0.3592
SMS_2	25	-0.2786	0.6495	0.3142
Hard vs Soft				
Run	Rank	ICM-Soft	ICM-Soft Norm	
SMS_2	8	-6.3311	0.7271	
SMS_3	16	-8.2171	0.6857	
SMS_1	19	-8.4236	0.6812	

most like human annotators and using such an architecture might be useful in future tasks.

Run 2 received a rank of 23/32 on hard-hard evaluation and 6/33 on hard-soft evaluation on ALL text instances. It also received ranks of 8/32 and 8/33 on hard-soft evaluation on ES and EN text instances respectively. The results of this run appear to be the most consistent across languages and indicates that training on both languages together might result in better performance on texts containing both of the languages.

Run 3 received similar ranks to Run 2 on ALL text instances but a rank of 3/32 on hard-soft evaluation on ES text instances (compared to 8/32) and 16/33 on hard-soft evaluation on EN text instances (compared to 8/33). Therefore, it seems to perform slightly better at Spanish and worse in English. The complete results are detailed in Tables 6, 7 and 8.

All the runs were performed on the same general architecture with variation in the training procedure. The results for hard-hard evaluation are not promising indicating other architectures might be more suitable for it. However, the results of hard-soft evaluation indicates that such an architecture might be more suitable for mimicking the human annotating process (as it appears to predict correctly for higher probabilities) and may be explored and developed further for future tasks.

6. Conclusions and Future Work

In this paper, we have detailed the work done by the SMS team in *Task 1: Sexism identification* and *Task 2: Source Intention* of the EXIST Lab at CLEF 2023 [1, 2]. The aim of our experiments was to test how LSTM + Attention models give results by incorporating different training methods of parallel or combined training, different word embeddings, different architectures etc.

We have come to the conclusion that combined processing of multilingual training data gives the best overall performance due to the results of run 2 of Task 2 which scored an overall ICM-Soft Norm of 0.6978, the highest among the three runs and performed more consistently across English and Spanish (with a rank of 8 in both) as compared to the other two runs.

In the future, we plan to continue to develop and assess our ambiguity resolution algorithms to improve the training phase of the system by removing the ambiguity and getting more features to train. The algorithms can be tested on larger datasets with a similar structure of being labelled by individual annotators. As we have utilized a combination of age and gender to group and compute scores for ambiguity resolution, a more complex combination of metrics could be used in datasets where more information regarding the annotators is available which might yield better results. We also plan to do a detailed analysis on the mimicking of human performance using various machine learning models as represented in this paper on other similar tasks and refine the process.

References

- [1] L. Plaza, J. C. de Albornoz, R. Morante, E. Amigó, J. Gonzalo, D. Spina, P. Rosso, Overview of EXIST 2023 – Learning with Disagreement for Sexism Identification and Characterization, in: A. Arampatzis, E. Kanoulas, T. Tsirika, S. Vrochidis, A. Giachanou, D. Li, M. Aliannejadi, M. Vlachos, G. Faggioli, N. Ferro (Eds.), *Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Fourteenth International Conference of the CLEF Association (CLEF 2023)*, Thessaloniki, Greece, September 2023.
- [2] L. Plaza, J. C. de Albornoz, R. Morante, E. Amigó, J. Gonzalo, D. Spina, P. Rosso, Overview of EXIST 2023 – Learning with Disagreement for Sexism Identification and Characterization

- (Extended Overview), in: M. Aliannejadi, G. Faggioli, N. Ferro, M. Vlachos (Eds.), Working Notes of CLEF 2023 - Conference and Labs of the Evaluation Forum, 2023.
- [3] S. Bird, E. Klein, E. Loper, Natural language processing with Python: analyzing text with the natural language toolkit, " O'Reilly Media, Inc.", 2009.
 - [4] E. Amigo, A. Delgado, Evaluating extreme hierarchical multi-label classification, in: Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Association for Computational Linguistics, Dublin, Ireland, 2022, pp. 5809–5819. URL: <https://aclanthology.org/2022.acl-long.399>. doi:10.18653/v1/2022.acl-long.399.
 - [5] D. J. Hand, K. Yu, Idiot's bayes: Not so stupid after all?, International Statistical Review / Revue Internationale de Statistique 69 (2001) 385–398. URL: <http://www.jstor.org/stable/1403452>.
 - [6] F. Chollet, et al., Keras, 2015. URL: <https://github.com/fchollet/keras>.
 - [7] N. Cristianini, E. Ricci, Support Vector Machines, Springer US, Boston, MA, 2008, pp. 928–932. URL: https://doi.org/10.1007/978-0-387-30162-4_415. doi:10.1007/978-0-387-30162-4_415.
 - [8] S. Haykin, Neural networks: a comprehensive foundation, Prentice Hall PTR, 1994.
 - [9] 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.
 - [10] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural computation 9 (1997) 1735–1780.
 - [11] J. Pennington, R. Socher, C. Manning, GloVe: Global vectors for word representation, in: Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics, Doha, Qatar, 2014, pp. 1532–1543. URL: <https://aclanthology.org/D14-1162>. doi:10.3115/v1/D14-1162.
 - [12] P. Bojanowski, E. Grave, A. Joulin, T. Mikolov, Enriching word vectors with subword information, Transactions of the Association for Computational Linguistics 5 (2017) 135–146.
 - [13] D. Bahdanau, K. Cho, Y. Bengio, Neural machine translation by jointly learning to align and translate, 2016. arXiv:1409.0473.
 - [14] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL: <https://www.tensorflow.org/>, software available from tensorflow.org.
 - [15] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, Journal of Machine Learning Research 15 (2014) 1929–1958. URL: <http://jmlr.org/papers/v15/srivastava14a.html>.
 - [16] H. Gholamalinezhad, H. Khosravi, Pooling methods in deep neural networks, a review, CoRR abs/2009.07485 (2020). URL: <https://arxiv.org/abs/2009.07485>. arXiv:2009.07485.

- [17] J. S. Bridle, Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition, in: NATO Neurocomputing, 1989.