UMSNH-INFOTEC@Dravidian-CodeMix-FIRE2020: An ensemble approach based on a multiple text representations

José Ortiz-Bejar[a], Jesus Ortiz-Bejar[a], Jaime Cerda-Jacabo[a], Mario Graff[b,c] and Eric S. Tellez[b,c]

[a]Universidad Michoacana de San Nicolás de Hidalgo, México, Michoacán, México
[b]CONACyT Consejo Nacional de Ciencia y Tecnología, Dirección de Cátedras, México
[c]INFOTEC Centro de Investigación e Innovación en Tecnologías de la Información y Comunicación, México

Abstract

This manuscript describes UMSNH-INFOTEC's participation in the first *Sentiment Analysis in Dravidian Code-Mixed text* task on FIRE 2020. Our solution combines several models that solve the task separately; we then construct a final decision through differential evolution and the linear combination of models' independently computed decision-values. The generic text categorization system μTC achieves the best performance when a single model is used, and our combined output improves individual performances.

Keywords

Code-Mixed, Word-embeddings, Sentiment Analysis, μTC

1. Introduction

One of the most common tasks involving natural language processing is the so-called Sentiment Analysis (SA). The main objective is to identify the feelings/intentions from a given text. The primary task is identifying the polarity of a text, i.e., if it is positive, negative, or neutral. Despite this simple definition, the task becomes challenging due to small context, errors, negations sentences, polysemy, and figurative language, among other language characteristics. From a machine learning perspective, the first step is to use a text model to transform messages into a vector space, and then, the points in this vector space are used to train a classification model for some specific task. In addition to the labels, the text model contains most of the language and domain knowledge that yields a successful classification.

Text models range from representing each message from sparse word frequency-based to semantic embeddings obtained by deep neural network-based approaches. While frequency-based are fitted only using the inputs text for the task, the word embeddings can be pre-trained over a large text collection. Embedding models can be fit for some specific dataset; this process

FIRE 2020: Forum for Information Retrieval Evaluation, December 16-20, 2020, Hyderabad, India
EMAIL: jortiz@umich.mx (J. Ortiz-Bejar); jesus.ortiz@umich.mx (J. Ortiz-Bejar); jcerda@umich.mx (J. Cerda-Jacabo); mario.graff@infotec.mx (M. Graff); eric.tellez@infotec.mx (E. S. Tellez)

	Mixed_feelings	**Negative**	**Positive**	**not-malayalam**	**unknown_state**
Training	289	549	2022	647	1344
Development	44	51	224	60	161
Test	70	138	565	177	398

Table 1
Malayalam-English data description

	Mixed_feelings	**Negative**	**Positive**	**not-tamil**	**unknown_state**
Training	1283	1448	7627	368	609
Development	141	165	857	29	68
Test	377	424	2075	100	173

Table 2
Tamil-English data description

requires a high amount of data and time, and therefore many times, embedding models are used as text transformers.

Even though there is a wide variety of pre-trained embeddings for many languages, the ones with few speakers account with limited resources. In this context, the frequency-based model exhibits a competitive performance against deep learning-based approaches. For our solution, we include multiple embeddings from the flair library [1].

The rest of the paper is organized as follow: first, the Dravidian Code-Mixed tasks are described briefly, at Sections 2. In Sections 3 and 4 basic ideas and models used for our approach are introduced. In Section 5, the core of our method is described. Section 6 discuss our results at the development phase; the final ranks are also presented. Finally, at Section 8 some conclusions are exposed.

2. Task description

The task is a sentiment analysis problem. This task has the particularity that code-mixed texts are part of the corpus data. In this context, the code-mixing term is used to defined texts which are non fully written in its native scripts. The task consists of two sentiment polarity problems for texts written in Malayalam-English and Tamil-English. The process of generating both corpora is introduced in [2, 3, 4, 5, 6].

Both Malayalam and Tamil datasets are split into training, development, and test collections. Messages are labeled with five polarity levels: *Mixed_feelings*, *Negative*, *Positive*, *not-malayalam* and *unknown_state*. Table 1 shows the label's distribution for Malayalam dataset; Table 2 describes the same information for Tamil dataset.

3. A language independent approach

At its basis, a SA task may be posed as a classification problem. For this case, a set of texts T and a set of labels Θ are given as input to fit model f capable of predicting the class for new input texts. The classification problem can be tackled in *agnostic* way by learning the relationship between the input T and the output Θ i.e. the model $f(\mathscr{L}(t_i)) = \theta_i$ where each label $\theta_i \in \Theta$ is the class label (output) for the $t_i \in T$ is only learned from the input set. Such a model f works over a transformed t_i, where the transformation is performed by $\mathscr{L}$. The possible text transformations are vast; therefore, they should be adapted for a specific task, input data, and classification strategy. The selection of the best transformation can be performed by hyper-parameter optimization. A state-of-the-art language-independent approach is μTC [7]. μTC optimizes $\mathscr{L}$ by exploring a configuration space comprised of different text transformations, tokenizers, and weighting schemes. The exploration is lead by a meta-heuristic and aims to produce effective configuration for Support Vector Machine Classifiers (SVM). As our knowledge about Dravidian dialects is limited, and the input data are in multiple languages, a language-independent approach is best suited for the proposed solution. Our solution aims to integrate a language-independent model enriched by pre-trained embedding models.

For convenience we assume that the output $f(\mathscr{L}(t_i)) = [p_1, p_2, ..., p_m]$ is a decision function in a vector form of size m, where m is the number of different labels in Θ. Then, each p_k is the score for t_i belonging to class k and the class for t_i is the k where p_k is maximum.

4. Pre-trained Embeddings

Since its introduction by Tomas Mikolov [8], pre-trained word embeddings are often used to train models when there are not large amounts of data for a given task. Even though classic embeddings have limited capacity on learning rare words, Character embeddings [9], and Pair-Byte embeddings [10] come to alleviate this situation. While embeddings like Word2Vec[8] and Glove[11] learn from prefix/suffix at the sub-word level, character-based embeddings learn representations specific to the task and domain. On the other hand, Byte-Pair embeddings use a variable-length encoding to iteratively merge the most frequent symbols pairs into a new symbol. This strategy makes both of them a suitable alternative to handle words that are not part of the input vocabulary and small training corpus.

5. Our solution approach

Our model is comprised of an optimized language-independent model μTC with pre-trained embeddings specific for Tamil, Malayalam, and multi-language embeddings. All models are ensembled by a linear combination of SVM classifier decision functions for each text transformations. The weights for each model are adjusted by using differential evolution. The model is represented at Eq. 1.

$$\Theta_p = \alpha_1 f(\mathscr{L}_1(T)) + \alpha_1 f(\mathscr{L}_2(T)) \ldots \alpha_m f(\mathscr{L}_m(T)), \qquad (1)$$

Model	Precision	Recall	F_{score}
μTC	0.613	0.676	0.631
Char/BP-Multi	0.598	0.686	0.593
Char/BP-Tamil/BP-Multi	0.607	0.694	0.612
BP-Tamil/BP-Multi	0.601	0.692	0.609
BP-Multi	0.573	0.681	0.586
Linear combination	**0.629**	**0.696**	**0.657**

Table 3
Performance for the different evaluated text models for Tamil-English

where Θ_p are the predicted values for the linear combination of the multiple SVM decisions function for each text transformations, and α is the contribution of each model to the final decision function.

The previous formulation allows us to state our solution as a constrained optimization problem, where α coefficients must be optimized to maximize a given fitness function. The weighted F_{score} score is maximized for this problem; Eq. 2 defines the optimization problem.

$$\begin{aligned} \underset{\alpha_i}{\text{maximize}} \quad & F_{score}(\Theta, \Theta_p) \\ \text{subject to} \quad & 0 \leq \alpha_i \leq 1. \end{aligned} \tag{2}$$

6. Experiments and results

We carry out experiments on five models and combinations: the optimized μTC, Byte-Pair embeddings for Multi-language (BP-Multi), language-specific Byte-Pair embeddings, and Character Embeddings. The model selected for submission was the one exhibiting the best scores for the evaluated metrics (weighted versions of F_{score}, Precision, and Recall). Table 3 shows the performances on the Tamil-English development dataset. As can be seen, the best single model performance is achieved by the μTC model, while the linear combination increases close to 2%, in our metrics.

On the other hand, Table 4 shows the performance for each single text model and the proposed linear combination on the Malayalam-English development dataset. Again the best model performance is achieved by μTC, but for the Malayalam, the increase achieved by the linear combination is lower than for the Tamil-English task.

Table 5 shows the α values for each one of the vector representation when the ensemble model is optimized, see Eq. 2. For both cases, the model in which the highest contribution is μTC; however, its contribution is greater for the Malayalam dialect. The second with the higher weight is corresponding to the staked model using Character and Byte-Pair for multi-language embeddings. On the other hand, for Tamil dialect, the weights the Char/BP-Multi is the model with the lowest contribution while stacking Byte-Pair Tamil and multi-language embeddings contribute with the second-highest α value.

Model	Precision	Recall	F_{score}
μTC	0.670	0.677	0.667
Char/BP-Multi	0.647	0.657	0.649
Char/BP-Tamil/BP-Multi	0.631	0.637	0.631
BP-Tamil/BP-Multi	0.629	0.637	0.630
BP-Multi	0.638	0.648	0.640
Linear combination	**0.674**	**0.683**	**0.668**

Table 4
Performance for the different evaluated text models for Malayalam-English

Model	Malayalam	Tamil
μTC	**0.9472**	**0.7017**
Char/BP-Multi	0.7071	0.0110
Char/BP-Lang/BP-Multi	0.1289	0.1373
BP-Lang/BP-Multi	0.3115	0.4892
BP-Multi	0.1075	0.3142

Table 5
Model's weighting in our linear-combination ensembling schema, i.e., α_i values in Eq. 2. Each column lists the weighting vector for a single language.

7. A brief review of μTC best model's parameters

For the sake of completeness, Table 6 shows the parameters for the best model obtained by the μTC system. We briefly summarize the involved parameters described in [12]. Parameters in Table 6 may be roughly divided into preprocessing and weighting schemes.

Parameters with *handler* suffix (i.e. emoji, hashtag, number, url, and user) have three possible options: *delete*, *group* and identity. These *delete* removes occurrences of entities of the specified kind while *group* option will change occurrences by a common identifier of the kind. For instance, setting emoji-handler as *delete* will remove all emojis in the text. In contrast, *group* option indicates that emoji's occurrences must be replaced with the special token _*emo*; this operation is designed for tasks that take advantage of the syntactic information of the token, regardless of the precise value. The identity operator leaves the instances untouched. On the other hand, binary parameters like diacritic-removal, duplication-removal, and punctuation-removal instruct if symbols must be removed or not. The lower-case parameter establishes whether the text is lower case or maintained as it is.

Furthermore, μTC allows the use of three different tokenizers:

- Words n-grams. This scheme tokenizes text into words and then produces all possible sub-sequences of n words (i.e., $m-n+1$ tokens for a text with m words). For this parameter, Malayalam's best performance model includes tokens of length 2, 3, 5, and 9. On the other hand, the Tamil model uses tokens of 1 and 3 words.
- Sentences q-grams. This approach produces n-grams at the character level, i.e., each token is a sub-string of size n. Here, the best model for Tamil dialect has tokens of length 2 and

Parameter name	Dravidian dialect	
	Malayalam	Tamil
lower-case	True	False
emojis-handler	identity	group
hashtag-handlers	identity	delete
url-handler	group	group
user-handler	group	group
number-handler	delete	identity
diacritic-removal	True	True
duplication-removal	False	True
punctuation-removal	True	True
q-grams	$3, 2, 1$	$3, 2$
n-grams	$2, 3, 5, 9$	$1, 3$
skip-grams	none	$(2, 1)$
weighting scheme	tfidf	tf
token-max-filter	1	1
token-min-filter	-1	-1

Table 6
*mu*TC best configuration parameters for Tamil and Malayalam texts

3. In contrast, the Malayalam includes sequences of 1, 2, and 3 characters.

- Skip-grams are word *n*-grams that skip middle parts in words sub-sequences. For this tokenizer, it must specify the sub-sequence length and the number of middle-words to skip. This class of tokens is not used for the Malayalam best configuration, but for the Tamil best model from continuous word sequences of length three, the middle word is removed, obtaining (2,1) skip-grams.

7.1. Weighting

Two frequency-based weighting schemes are used over vector space for the bag of word representations obtained using those mentioned above preprocessing and tokenization strategies. Any of term frequency (TF) and term frequency-inverse document frequency (TFIDF) are selected along with at frequency thresholds using token-min-filter c_{min}, and token-max-filter c_{max}. Where all tokens do not reaching the frequency c_{min}*freq* or having frequency greater than c_{max}*max-freq* are delete. The term *max-freq* stands for the frequency of the most repeated token in the collection. Table 6 indicates that no token filtering is applied to the vocabularies.

7.2. Final Rank-list

We decided to submit the linear combination based on its development dataset's performance. Table 7 shows the rank for our approach for each task. In both cases, our method was ranked above-average performance.

Task name	Precision	Recall	F_{score}	Rank
Tamil-English	0.61	0.68	0.63	3
Malayalam-English	0.68	0.69	0.68	6

Table 7
UMNSH-INFOTEC's final rank (determined based on performance over test dataset)

8. Conclusions

This manuscript presents a model solution for the Dravidian code-mixed text task that integrates language-specific knowledge from pre-trained models with a language-independent model. Our approach is based on the linear combination of several independently created models using differential evolution. Our proposal can be implemented with open-source libraries, using just a relatively few code lines, and achieves competitive performances for the evaluated tasks. The scripts and data used to implement our approach are available at *Github*[1].

References

[1] A. Akbik, D. Blythe, R. Vollgraf, Contextual string embeddings for sequence labeling, in: COLING 2018, 27th International Conference on Computational Linguistics, 2018, pp. 1638–1649.

[2] B. R. Chakravarthi, V. Muralidaran, R. Priyadharshini, J. P. McCrae, Corpus creation for sentiment analysis in code-mixed Tamil-English text, in: Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL), European Language Resources association, Marseille, France, 2020, pp. 202–210. URL: https://www.aclweb.org/anthology/2020.sltu-1.28.

[3] B. R. Chakravarthi, N. Jose, S. Suryawanshi, E. Sherly, J. P. McCrae, A sentiment analysis dataset for code-mixed Malayalam-English, in: Proceedings of the 1st Joint Workshop on Spoken Language Technologies for Under-resourced languages (SLTU) and Collaboration and Computing for Under-Resourced Languages (CCURL), European Language Resources association, Marseille, France, 2020, pp. 177–184. URL: https://www.aclweb.org/anthology/2020.sltu-1.25.

[4] B. R. Chakravarthi, R. Priyadharshini, V. Muralidaran, S. Suryawanshi, N. Jose, E. Sherly, J. P. McCrae, Overview of the track on Sentiment Analysis for Dravidian Languages in Code-Mixed Text, in: Proceedings of the 12th Forum for Information Retrieval Evaluation, FIRE '20, 2020.

[5] B. R. Chakravarthi, R. Priyadharshini, V. Muralidaran, S. Suryawanshi, N. Jose, E. Sherly, J. P. McCrae, Overview of the track on Sentiment Analysis for Dravidian Languages in Code-Mixed Text, in: Working Notes of the Forum for Information Retrieval Evaluation (FIRE 2020). CEUR Workshop Proceedings. In: CEUR-WS. org, Hyderabad, India, 2020.

[1] https://github.com/kyriox/dravidian-codemixed

[6] B. R. Chakravarthi, Leveraging orthographic information to improve machine translation of under-resourced languages, Ph.D. thesis, NUI Galway, 2020.

[7] E. S. Tellez, D. Moctezuma, S. Miranda-Jiménez, M. Graff, An automated text categorization framework based on hyperparameter optimization, Knowledge-Based Systems 149 (2018) 110–123. URL: https://github.com/INGEOTEC/microtc.

[8] T. Mikolov, K. Chen, G. Corrado, J. Dean, L. Sutskever, G. Zweig, word2vec, URL https://code. google. com/p/word2vec 22 (2013).

[9] B. Heinzerling, M. Strube, Bpemb: Tokenization-free pre-trained subword embeddings in 275 languages, arXiv preprint arXiv:1710.02187 (2017).

[10] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, C. Dyer, Neural architectures for named entity recognition, arXiv preprint arXiv:1603.01360 (2016).

[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), 2014. doi:10.3115/v1/D14-1162.

[12] E. Tellez, D. Moctezuma, S. Miranda-Jiménez, M. Graff, An automated text categorization framework based on hyperparameter optimization, Knowledge-Based Systems (2018). doi:10.1016/j.knosys.2018.03.003.