MUCS@TechDOfication using FineTuned Vectors and N-grams

F Balouchzahi
Dept. of Computer Science
Mangalore University
Mangalore - 574199
India
frs_b@gmail.com

M D Anusha
Dept. of Computer Science
Mangalore University
Mangalore - 574199
India
anugowda251@gmail.com

H L Shashirekha
Dept. of Computer Science
Mangalore University
Mangalore - 574199
India
hlsrekha@gmail.com

Abstract

The increase in domain specific text processing applications are demanding tools and techniques for domain specific Text Classification (TC) which may be helpful in many downstream applications like Machine Translation, Summarization, Question Answering etc. Further, many TC algorithms are applied on globally recognized languages like English giving less importance for local languages particularly Indian languages. To boost the research for technical domains and text processing activities in Indian languages, a shared task named "TechDOfication2020" is organized by ICON'20. The objective of this shared task is to automatically identify the technical domain of a given text which provides information about coarse grained technical domains and fine grained subdomains in eight languages. To tackle this challenge we, team MUCS have proposed three models, namely, DL-FineTuned model applied for all subtasks, and VC-FineTuned and VC-ngrams models applied only for some subtasks. n-grams and word embedding with a step of fine-tuning are used as features and machine learning and deep learning algorithms are used as classifiers in the proposed models. The proposed models outperformed in most of subtasks and also obtained first rank in subTask1b (Bangla) and subTask1e (Malayalam) with f1 score of 0.8353 and 0.3851 respectively using DL-FineTuned model for both the subtasks.

1 Introduction

TC is one of the important areas of research in Natural Language Processing (NLP). Most of the TC experiments being conducted are on resource rich languages such as English, Spanish etc. giving less importance to resource poor languages particularly Indian languages. Further, with the increase in domain specific text processing applications, domain specific TC is gaining importance demanding specialized tools and techniques to handle domain specific text datasets (Sun et al., 2019) for many downstream applications like Machine Translation, Summarization, Question Answering etc. In this direction, a shared task named "TechDOfication2020: Technical Domain Identification" is organized in association with 17thInternational Conference on Natural Language Processing[1] (ICON) 2020 to automatically identify the technical domain of a given text in eight languages, namely, English, Bangla, Gujarati, Hindi, Malayalam, Marathi, Tamil, and Telugu. These text provides information about specific coarse grained technical domains and fine grained subdomains. Details of the shared tasks are provided in the task website[2]. Technical domain identification can be modeled as a domain specific TC task. In this paper, we describe our models, namely, VC-ngrams, VC-FineTuned and DL-FineTuned proposed by our team MUCS for technical domain identification in eight languages using n-grams and fine-tuned vectors as features.

n-grams features which are simple and scalable are utilized in many NLP tasks. With a bigger value of 'n', a model can store more contexts with a well-understood space-time tradeoff enabling many TC experiments to scale up efficiently. Word embedding or vector representation of words captures grammatical and semantic information of a word which can be an enlightening feature for many NLP applications. In this study, we investigate the two popular word vector models namely, GloVe[3] for English (subTask1a and subTask2a) and Bangla (subTask1b) and fastText[4] (we didn't get Glove for other languages) for rest of subtasks for learning word vectors. GloVe model produces a vector space with meaningful substructure, as evidenced

[1]http://www.iitp.ac.in/ ai-nlp-ml/icon2020/
[2]https://ssmt.iiit.ac.in/techdofication.html
[3]https://nlp.stanford.edu/projects/glove
[4]https://fasttext.cc

Proceedings of the 17th International Conference on Natural Language Processing: TechDOfication 2020 Shared Task, pages 1–5
Patna, India, December 18 - 21, 2020.

by its performance on a recent word analogy task[5] and it forces the model to encode the frequency distribution of words that occur near them in a more global context. FastText, an extension of the Word2Vec model is a word embedding method that helps to achieve the meaning of shorter words. It allows the embedding's to understand suffixes and prefixes and works well with rare words. So even if a word is not seen during training, it can be broken down into n-grams to obtain its embedding's. Rarely pre-trained word embeddings are available for domain specific text and even so infrequent for resource poor languages such as Persian and Indian languages. Hence, fine-tuning a word embedding using specific (technical) domain texts can improve the performance of TC as vectors for words missing in the pre-trained model will be updated by the domain specific texts (Liao et al., 2010) given for training.

Several Machine Learning (ML) and Deep Learning (DL) models are providing effective and accurate results for TC by reducing false positives (Bhargava et al., 2016). Many DL models are using Bidirectional Long Short Term Memory (BiLSTM) which contains two LSTMs: one taking the commitment to a forward course, and the other in a retrogressive way. BiLSTMs are at the core of a few neural models accomplishing cutting edge execution in a wide assortment of undertakings in NLP (Bhargava et al., 2016). BiLSTM model can use the pre-trained word embeddings provided by fastText and GloVe.

The rest of the paper is organized as follows. Section 2 highlights the related work followed by the proposed methodology in Section 3. Experiments and results are described in Section 4 and the paper finally concludes in Section 5.

2 Related Work

Researchers round the globe have developed several approaches for domain specific TC. Some of related ones are described below:

A strategy to develop sentence vectors (sent2vec) by averaging the word embeddings is proposed by (Liu, 2017) to explore the effect of word2vec on the performance of sentiments analysis of citations. The authors trained the ACL-Embeddings (300 and 100 measurements) from ACL collection and also examined polarity-specific word embeddings (PS-Embeddings) for characterizing positive and negative references. The generated embedding is fed to SVM classifier and using 10-fold cross validation they obtained a macro-f1 score of 0.85 and the weighted-f1 score of 0.86 and proved the efficiency of word2vec on classifying positive and negative citations.

A domain-specific intent classification for Sinhala language proposed by (Buddhika et al., 2018) utilized a feed-forward neural network with back propagation. They trained their model on a banking domain-related data set with Mel Frequency Cepstral Coefficients extracted from a Sinhala speech corpus of 10 hours and obtained 74% results on identification accuracy of speech queries. (Zhou et al., 2016) proposed a combination of two models BLSTM-2D Pooling and BLSTM-2D CNN and tested it on six TC tasks, including sentiment analysis, question classification, subjectivity classification, and newsgroups classification to compare their models with the state-of-the-art models. The authors evaluated the performance of proposed models on different lengths of sentences and then conducted a sensitivity analysis of 2D filter and max-pooling size. BLSTM-2DCNN model achieved excellent performance on 4 out of 6 tasks and obtained 52.4% and 89.5% test accuracies on Stanford Sentiment Treebank-1 and Treebank-2 datasets respectively.

(Rabbimov and Kobilov, 2020) explored a multi-class TC task to classify Uzbek language text. They collected articles on ten classes taken from the Uzbek "Daryo" online news and used six diverse ML algorithms namely, Multinomial Naıve Bayes (MNB), Decision Tree Classifier (DTC), SVM, Random Forest (RF) and LR. Hyper parameters for the classifiers were obtained by Grid search algorithms with 5-fold cross-validation. The results obtained illustrates that, in many cases the character n-grams gives better results than the word level n-grams. However, among all,SVM with the Radial Basis Function kernel (RBF SVM) using TF-IDF and character level four grams features obtained best results with an accuracy of 86.88

3 Methodology

Inspired by (Liu, 2017) to use word embeddings as features for classification, (Zhou et al., 2016) to use BiLSTMs for classification and (Rabbimov and Kobilov, 2020) for using ML algorithms for classification, we proposed three models, namely,

[5]https://dzone.com/articles/glove-and-fasttext-two-popular-word-vector-models

Voting Classifiers using n-grams (VC-ngrams), Voting Classifiers using FineTuned word vectors (VC-FineTuned) and Deep Learning using FineTuned word vectors (DL-FineTuned). Each model is built in two steps namely, i) Feature Extraction and ii) Model Construction, as explained below:

3.1 Feature Extraction

n-grams and word embeddings features are utilized in the proposed models. Linguistic models have proven their efficiency in many studies. Hence, n-grams features including Char n-grams (1, 2, 3, 4, 5) along with word n-grams (1, 2, 3) are extracted from input data and CountVectorizer[6] library is used to generate n-grams count vectors which are used in VC-ngrams model.

The pre-trained word embeddings-GloVe, with a vector size 300 is used for English and Bangla subtasks and fastText with a size of 300 for the subtasks of rest of the languages as features after fine tuning using the training data. Fine tuning of vectors in a pre-trained model by training data of a specific task helps in generating vectors for words missing in pre-trained models, and it can lead to higher performance specifically in fine-grained tasks such as domain specific TC. Fine-tuned vectors are utilized to build embedding matrix to train VC-FineTuned and DL-FineTuned models.

3.2 Model Construction

Construction of the three models is explained below:

- **VC-ngrams model:** An ensemble Voting Classifier with three ML classifiers namely, Multilayer Perceptron (MLP), Logistic Regression (LR), and Support Vector Machine (SVM) have been trained on n-grams count vectors obtained in Feature Extraction step. Default parameters are used for SVM and LR classifiers and for MLP, hidden layer sizes are set to (150, 100, 50) and maximum iteration, activation, solver, and random state have been set to 300, Relu, Adam and 1 respectively. Structure of VC-Ngrams model is shown in figure-1.

- **DL-FineTuned model:** This model is created based on DL architecture using pre-trained word embeddings. A pre-trained word embedding of size 300 is fine-tuned using the specific language training set and the obtained vectors are used to build embedding matrix. This embedding matrix is used as input for Sequential model from Keras[7] library to build a BiLSTM network of size 100 with activation and optimizer parameters set to "softmax" and "adam" respectively. The output dimensions are configured based on the corresponding subTask's labels (e.g. seven labels for subTask2a) as given by the organizers. The constructed model has been trained for dynamic epochs till loss value got stabilized (not more than 20 epochs). Structure of DL-FineTuned model is shown in Figure 2.

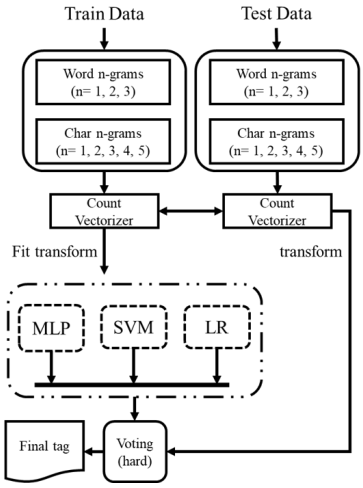

Figure 1: Structure of VC-ngrams model

- **VC-FineTuned model:** The architecture of VC-FineTuned model is driven from merging the features extraction part of DL-FineTuned model and model construction part of VC-Ngrams model. In this model, a pre-trained word embedding of size 300 is fine-tuned using the specific language training set and the resulting vectors are used to build an ensemble Voting Classifier with similar estimators as the VC-ngrams model, namely, MLP, SVM, and LR.

The DL-FineTuned model is applied for all the subtasks whereas due to lack of pre-trained models and the long time taken for training the models, VC-ngrams model is applied for English and Gujarathi subTask1 and English subTask2 and VC-FineTuned model is applied for English, Gujarathi and Malayam subtasks1.

[6] https://scikit-learn.org/stable/modules/generated/sklearn.feature$_extraction.text.CountVectorizer.html$

[7] https://keras.io/guides/sequential$_model$

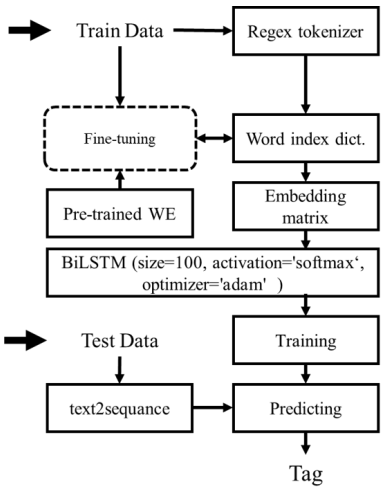

Figure 2: Structure of DL-FineTuned model

subtask-language	Train Set	Dev. Set	Test Set	No. Classes
subTask1a-English	23959	4850	2500	5
subTask1b-Bangla	53574	5842	1922	5
subTask1c-Gujarati	32316	4698	2508	5
subTask1d-Hindi	128031	13468	2670	7
subTask1e-Malayalam	33891	3390	1051	3
subTask1f-Marathi	38650	3780	1788	4
subTask1g-Tamil	69478	4710	2070	6
subTask1h-Telugu	59645	5920	2098	6
subTask2a-English	13580	1360	1929	7

Table 1: Size of Datasets provided by TechDOfication 2020

4 Experimental Results

4.1 Datasets

Datasets provided by TechDOfication 2020 consists of train, development, and test sets for nine subtasks of eight languages namely, English, Bangla, Gujarati, Hindi, Malayalam, Marathi, Tamil, and Telugu and the details of the datasets are given in Table 1. Only the training datasets for English subtasks are balanced and the subtasks of all other languages are not balanced. Further, subTask1e (Malayalam) has only three classes and subTask1d (Hindi) and subTask2a (English) have seven classes each. More details about the datasets are available in task website[8] and also in Github repository[9].

4.2 Results

The number of participated teams and submitted runs given in Table 2 illustrates that more teams have registered for English subtasks and number of runs submitted for English subtasks are also more compared to other subtasks. Performance of the models for the subtasks on the development set in terms of F1 score are shown in Table 3.

While DL-FineTuned model is applied for all the subtasks VC-ngrams and VC-FineTuned models are applied only for some subtasks. It can be observed that DL-FineTuned models perform better for some subtasks and VC-ngrams models perform better for some other subtasks. Also the proposed DL-FineTuned model obtained first position in subTask 1b (Bangla) and subTask1e (Malayalam) with f1 score of 0.8353 and 0.3851 respectively. The results illustrate that fine-tuning the vectors by specific domain text for the subtasks using DL-FineTuned model have performed better compared to other two models. However, VC-ngrams applied on the three subtasks have also performed well.

[8]https://ssmt.iiit.ac.in/techdofication.html

[9]https://github.com/fazlfrs/TechDofication2020

Conclusion and future works

We, team MUCS proposed three models namely DL-FineTuned model applied for all subtasks, and VC-FineTuned and VC-ngrams models applied for only few subtasks to identify technical domain of a given text in Indian languages. The results reported by TechDOfication 2020 task organizers illustrate that the proposed DL-FineTuned model outperformed in most of the subtasks and also obtained first rank in subTask 1b (Bangla) and subTask 1e (Malayalam) with f1 score of 0.8353 and 0.3851 respectively. We would like to improve our proposed models by applying other features and also learning models such as Transfer Learning.

References

Rupal Bhargava, Yashvardhan Sharma, and Shubham Sharma. 2016. Sentiment analysis for mixed script indic sentences. In *2016 International conference on advances in computing, communications and informatics (ICACCI)*, pages 524–529. IEEE.

Darshana Buddhika, Ranula Liyadipita, Sudeepa Nadeeshan, Hasini Witharana, Sanath Javasena, and

SubTask-language	No. Teams	No. Runs
subTask1a	13	24
subTask1b	9	14
subTask1c	8	15
subTask1d	9	13
subTask1e	8	14
subTask1f	9	15
subTask1g	8	13
subTask1h	9	16
subTask2a	12	22

Table 2: No. teams and submitted runs for each subtasks

subTask-language	Model(s)	F1 score	Rank
subTask1a-English	DL-Fine Tuned	0.7294	15
	VC-ngrams	0.7754	8
	VC-FineTuned	0.7352	14
subTask1b-Bangla	**DL-FineTuned**	**0.8353**	**1**
subTask1c-Gujarati	DL-FineTuned	0.0184	15
	VC-ngrams	0.0187	13
	VC-FineTuned	0.1530	12
subTask1d-Hindi	DL-FineTuned	0.79	6
subTask1e-Malayalam	**DL-FineTuned**	**0.3851**	**1**
	VC-FineTuned	0.3167	8
subTask1f-Marathi	DL-FineTuned	0.5664	8
subTask1g-Tamil	DL-FineTuned	0.4433	4
subTask1h-Telugu	DL-FineTuned	0.6461	7
subTask2a-English	DL-FineTuned	0.7113	12
	VC-ngrams	0.7691	8

Table 3: Performance of the models on the subTasks

Uthayasanker Thayasivam. 2018. Domain specific intent classification of sinhala speech data. In *2018 International Conference on Asian Language Processing (IALP)*, pages 197–202. IEEE.

Chunyuan Liao, Hao Tang, Qiong Liu, Patrick Chiu, and Francine Chen. 2010. Fact: fine-grained cross-media interaction with documents via a portable hybrid paper-laptop interface. In *Proceedings of the 18th ACM international conference on Multimedia*, pages 361–370.

Haixia Liu. 2017. Sentiment analysis of citations using word2vec. *arXiv preprint arXiv:1704.00177*.

IM Rabbimov and SS Kobilov. 2020. Multi-class text classification of uzbek news articles using machine learning. In *Journal of Physics: Conference Series*, volume 1546, page 012097.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *China National Conference on Chinese Computational Linguistics*, pages 194–206. Springer.

Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional lstm with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639*.