ICON 2020

17th International Conference on Natural Language Processing

Proceedings of the TechDOfication 2020 Shared Task

December 18 - 21, 2020
Indian Institute of Technology Patna, India

Introduction

These shared task proceedings concluded the shared task on Technical DOmain Identification, named as TechDOfication 2020, launched on 7th October 2020. The shared task was collocated with the 17th International Conference on Natural Language Processing (ICON 2020), held at IIT-Patna, India. The goal of the shared task was to automatically identify the technical domain of a given text in a specified language. The languages included in this task were English, Bangla, Gujarati, Hindi, Malayalam, Marathi, Tamil, and Telugu.

Two subtasks were part of this shared task. The first subtask was to identify a Coarse-grained technical domain like Computer Science, Communication Technology, Management, Math, Physics, Life Science, Law etc. The second subtask involved the task of identification of fine-grained subdomains for the Computer Science domain like Operating System, Computer Network, Database etc. The first subtask was conducted in all the mentioned languages while the second subtask only involved English.

We received nine system submissions and system description papers. Each system description paper was reviewed by two members of the reviewing committee – all papers were accepted. Macro F1 scores were used to evaluate the systems.

Both Machine Learning and Neural Network methods were used by different teams in this shared task. Support Vector Machine and Voting Classifiers were the predominantly used Machine Learning models. Different Neural architectures like Multi-Layer Perceptron, BiLSTM, BERT based models, Graph Convolutional Neural Networks, XLM-RoBERTa, Multichannel LSTM-CNN were also used. We would like to thank the ICON-2020 organizers, the shared task participants, the authors, and the reviewers for making this shared task successful.

Shared task page: `http://ssmt.iiit.ac.in/techdofication`
Main conference page: `https://www.iitp.ac.in/~ai-nlp-ml/icon2020/index.html`

Organizing Committee:

Dipti Misra Sharma (IIIT-Hyderabad)
Asif Ekbal (IIT-Patna)
Karunesh Arora (C-DAC, Noida)
Sudip Kumar Naskar (Jadavpur University)
Dipankar Ganguly (C-DAC, Noida)
Sobha L (AUKBC-Chennai)
Radhika Mamidi (IIIT-Hyderabad)
Sunita Arora (C-DAC, Noida)
Pruthwik Mishra (IIIT-Hyderabad)
Vandan Mujadia (IIIT-Hyderabad)

Table of Contents

Shared Task Program

Monday, December 21, 2020

+ 14:00 - 14:30 **Talk by Sobha L, AUKBC-Chennai**

+ 14:30 - 14:45 Shared Task Overview

Presentations

16:45 - 16:55	*Automatic Technical Domain Identification* Hema Ala and Dipti Sharma
16:58 - 17:08	*Technical Domain Identification using word2vec and BiLSTM* Koyel Ghosh, Dr. Apurbalal Senapati and Dr. Ranjan Maity
17:11 - 17:21	*A Graph Convolution Network-based System for Technical Domain Identification* Alapan Kuila, Ayan Das and Sudeshna Sarkar
17:24 - 17:34	*Fine-grained domain classification using Transformers* Akshat Gahoi, Akshat Chhajer and Dipti Mishra Sharma
17:37 - 17:47	*MUCS@TechDOfication using FineTuned Vectors and n-grams* Fazlourrahman Balouchzahi, M D Anusha and H L Shashirekha
17:50 - 18:00	*TechTexC: Classification of Technical Texts using Convolution and Bidirectional Long Short Term Memory Network* Omar Sharif, Eftekhar Hossain and Mohammed Moshiul Hoque
18:03 - 18:13	*An Attention Ensemble Approach for Efficient Text Classification of Indian Languages* Atharva Kulkarni, Amey Hengle and Rutuja Udyawar
18:16 - 18:26	*Multilingual Pre-Trained Transformers and Convolutional NN Classification Models for Technical Domain Identification* Suman Dowlagar and Radhika Mamidi
18:29 - 18:39	*Multichannel LSTM-CNN for Telugu Text Classification* Sunil Gundapu and Radhika Mamidi

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 17th International 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

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.

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

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, 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.

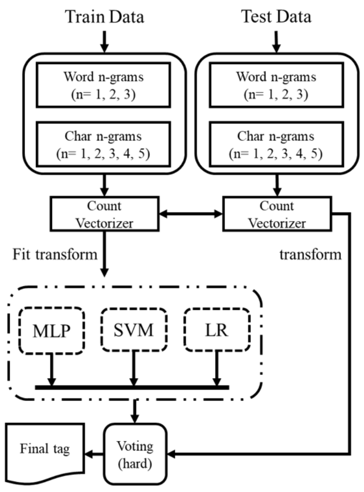

Train Data
Test Data
Word n-grams (n= 1, 2, 3)
Char n-grams (n= 1, 2, 3, 4, 5)
Word n-grams (n= 1, 2, 3)
Char n-grams (n= 1, 2, 3, 4, 5)
Count Vectorizer
Count Vectorizer
Fit transform
transform
MLP
SVM
LR
Final tag
Voting (hard)

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$_model$

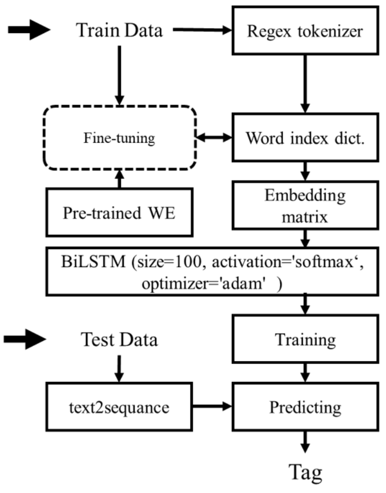


Figure 2: Structure of DL-FineTuned model

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.

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

[8]https://ssmt.iiit.ac.in/techdofication.html
[9]https://github.com/fazlfrs/TechDofication2020

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*.

A Graph Convolution Network-based System for Technical Domain Identification

Alapan Kuila, Ayan Das and Sudeshna Sarkar
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur,
Kharagpur, WB, India-721302
{alapan.kuila, ayan.das, sudeshna}@cse.iitkgp.ac.in

Abstract

This paper presents the IITKGP contribution at the Technical DOmain Identification (TechDOfication) shared task at ICON 2020. In the preprocessing stage, we applied part-of-speech (PoS) taggers and dependency parsers to tag the data. We trained a graph convolution neural network (GCNN) based system that uses the tokens along with their PoS and dependency relations as features to identify the domain of a given document. We participated in the subtasks for coarse-grained domain classification in the English (Subtask 1a), Bengali (Subtask 1b) and Hindi language (Subtask 1d), and, the subtask for fine-grained domain classification task within Computer Science domain in English language (Subtask 2a).

1 Introduction

Text classification is the task of assigning a category to a given piece of text based on its content from a predefined set of categories (Aggarwal and Zhai, 2012; Kowsari et al., 2019). It is a fundamental natural language processing (NLP) task with several downstream applications such as sentiment analysis, topic labeling and machine translation.

The ICON 2020 shared task on technical domain identification is essentially a text classification problem which involves identification of the technical domain in which a document belongs to a fixed set of domains. In this task, the participants are expected to develop a system that automatically identifies the technical domain of a given text (a small passage) in specified language.

We participated in the following subtasks:

1. **Subtask 1a:** Coarse grained domain classification in English.
2. **Subtask 1b:** Coarse grained domain classification in Bengali.
3. **Subtask 1d:** Coarse grained domain classification in Hindi.
4. **Subtask 2a:** Fine grained domain classification within Computer Science domain in English.

We applied PoS tagging and dependency parsing on the training and test data using PoS taggers and dependency parsers in the corresponding languages. We have used the PoS tags and the dependency relations of the tokens with their parent nodes in the dependency trees as features of the tokens in our system.

In the domain identification stage, the technical domain of a document is predicted from the representation of the document obtained using a GCNN, that takes the contextual representations of the constituent tokens of the document and a graph representation of the document as input. The contextual representations of the tokens are obtained by using a multi-layer bi-directional long-short term memory (LSTM) (Hochreiter and Schmidhuber, 1997; Schuster and Paliwal, 1997) that takes the representations of the tokens as input.

Corresponding to each subtask we submitted single runs of the systems.

2 Our Proposed Model

In this section, we will discus our proposed technical domain identification system in detail. The steps for predicting the domain of a given document are as follows.

S1. We partitioned the document into constituent sentences and tokenized each sentence into its constituent words.

S2. We PoS tagged and dependency parsed each sentence.

S3. We obtained the contextual representation of each token in the document by applying a multi-layer bi-directional LSTM on the representations of the tokens where we considered the entire document as a single sequence.

S4. We used the contextual representations of the tokens obtained in the previous step and the graph representation of the document as input to a GCNN to obtain the final feature representation of each word.

S5. We combined the final feature representations of the tokens to derive the document representation.

S6. We passed the document representation as input to a multi-layer perceptron (MLP) followed by a softmax layer to predict domain of the document.

2.1 Partitioning of Document into Sentences and Tokenization

We derived the sentences from the documents by partitioning the document on the following characters: ".", "?" and "!". The sentences were tokenized based on spaces. The tokens are essentially the space separated word in the sentences.

2.2 PoS Tagging and Dependency Parsing of Sentences

Before training or testing we PoS tagged and dependency parsed the sentences. We PoS tagged and parsed the English sentences in the subtasks *1a* and *2a* using the SpaCy library (Honnibal and Johnson, 2015; Honnibal and Montani, 2017). The Bangla and Hindi sentences for the subtasks *1b* and *1d* respectively were PoS tagged and dependency parsed using a LSTM based sequence tagger (Qi et al., 2018) and a bi-affine graph based dependency parser (Dozat et al., 2017). The Bangla tagger and parser were trained using a Bangla treebank developed in our institute. The Hindi tagger and parser were trained using the Universal Dependencies v2.0 Hindi treebank (Nivre et al., 2016).

2.3 Generation of Contextual Representation of Tokens

We obtained the contextual representation of the tokens in a document by passing the distributed representations of the tokens in the document as input to a 3-layer bi-directional LSTM. Here we treated the entire document as a sequence. Each token is represented as the concatenation of the embedding of the token, the embedding of its PoS tag, the embedding of its dependency relation with its parent in the dependency tree of the sentence, and, the character-level representation of the word obtained by applying a convolutional neural network (Goodfellow et al., 2016) on the embeddings of the constituent characters of the token.

All the embeddings were randomly initialized and updated during training phase.

2.4 Graph Convolution Neural Network to Generate the Document Representation

In this section, we discuss the implementation of GCNN (Kipf and Welling, 2016) used in our system.

2.4.1 Input Layer

The input to the the GCNN comprises of the contextual representations of the tokens in the document and the directed graph representation of the document in the form of an adjacency matrix.

2.4.2 Graph Construction

The number of nodes $|V|$ in the document graph is the total number of tokens contained in the document and the graph edges are represented by various intra- and inter-sentence dependency edges which are discussed below.

- **Syntactic dependency edges:** The edge set comprises of the edges between all token pairs that are linked by head-dependent relations in the dependency tree of the corresponding sentence. To consider that the information flows in both forward and backward directions of syntactic dependency arcs, we included both forward and backward edges in the graph edge set. More precisely, if there is an arc from a token t_i to a token t_j in the dependency tree, then $A(i,j) = 1$ and $A(j,i) = 1$.

- **Adjacent sentence edges:** In order to keep sequential information flows through the consecutive sentences we also connected the root nodes of the neighbouring sentences. Therefore, $A_{adj}(i,j) = 1$ and $A_{adj}(j,i) = 1$, if the tokens t_i and t_j are the root nodes of the consecutive sentences.

- **Self-node edges:** We also include self node edges in the graph. Therefore, for all the tokens t_i in the document, $A_(i,i) = 1$.

2.4.3 GCNN Layer

GCNN is an advanced version of CNN operating on graphs that induce the node features based on the properties of their neighbouring nodes. GCNN with one layer of convolution can capture information of only immediate neighbours. When multiple GCNN layers are stacked, information from larger neighbourhoods are accumulated. Let A be the adjacency matrix of the text-graph. A contains the edges as stated in Section 2.4.2 and the adjacency matrix is of the dimension $|V| \times |V|$, where $|V|$ is the number of tokens in the document. After stacking k GCNN layers we get the adjacency matrix for the k-th-order text graph A^k, where, $A^k = (A)^k$. A^k holds all the k-hop paths in the text-graph. The ith word representation of the $(k+1)$-layer text-graph (A^k) is calculated as:

$$h_i^{k+1} = g(h_i^k, A^k)$$

where, $g(.)$ refers to the graph convolution function and $+$ indicates element wise summation operation. The function $g(.)$ is defined as:

$$g\left(h_i^k, A^k\right) = \sigma(\sum_{j=1}^{|V|}(A^k(i,j)(W_A^k * h^k + b_A^k)))$$

Here, σ indicates the ReLU activation function. W_A^k and b_A^k are the weight matrix and bias item for A^k.

The initial value of the representation of the i^{th} token h_i^0 is the contextual representation of the token t_i from the output of the LSTM (Section 2.3).

In our system we have trained a stacked GCNN of 3 blocks. The model architecture is shown in Figure 1.

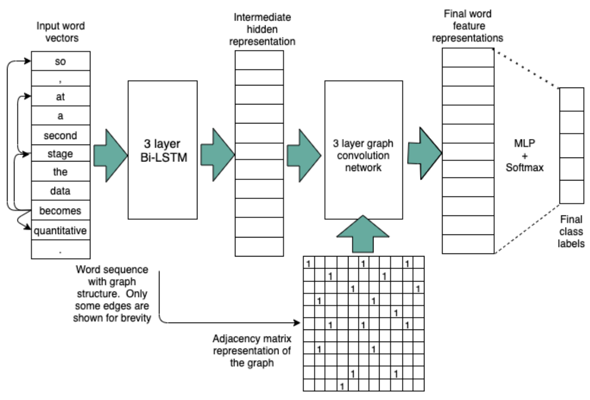

Input word vectors
so
,
at
a
second
stage
the
data
becomes
quantitative
.
3 layer Bi-LSTM
Intermediate hidden representation
3 layer graph convolution network
Final word feature representations
MLP + Softmax
Final class labels
Word sequence with graph structure. Only some edges are shown for brevity
Adjacency matrix representation of the graph

Figure 1: Block diagram of our proposed model

Sub-task	Classes	# of docs.		
		Train	Dev	Test
1a	cse, che, physics, law, math	23962	4850	2500
1b	bioche, mgmt, com_tech, phy, cse	58500	5842	1921
1d	other, mgmt, phy, com_tech, cse, math, bioche	148445	14338	4211
2a	ca, se, algo, cn, pro, ai, dbms	13580	1360	1929

Table 1: Statistics of datasets for the different subtasks

Training settings	
Loss function	Cross-entopy over the domain classes
Optimizer	Adam (lr=1e-4, eps=0.1)
System settings	
Token embeddings	140
PoS embeddings	20
Relation embeddings	20
Character embeddings	20
# of char CNN kernel	3
# of char CNN filter	30
LSTM layers	3
LSTM hidden size	100
GCNN stack	3
GCNN size	500, 200, 200
Dropout rate	0.25

Table 2: System settings and hyper-parameters

2.5 Document Representation and Domain Prediction

A representation of the document is obtained as the point-wise mean of the token representations obtained as the output of the GCNN (Section 2.4). Finally, the document representation is passed as input to a MLP layer followed by a softmax layer. The output of the softmax layer is a probability distribution over the possible domain classes. The class label with the highest probability value is predicted as the class of the document.

3 Data Description

The statistics of the data corresponding to the different subtasks in which we have participated are summarized in Table 1.

Subtask	Accuracy		Precision		Recall		F1-score	
	GCN	Best	GCN	Best	GCN	Best	GCN	Best
Subtask-1a	0.6564	**0.8156**	0.6850	**0.8155**	0.6564	**0.8156**	0.6560	**0.8144**
Subtask-1b	0.6878	**0.8335**	0.7432	**0.8420**	0.7023	**0.8515**	0.6897	**0.8353**
Subtask-1d	0.4656	**0.6117**	0.4706	**0.6478**	0.4523	**0.5989**	0.4523	**0.6044**
Subtask-2a	0.7029	**0.8252**	0.7021	**0.8265**	0.7034	**0.8252**	0.7008	**0.8244**

Table 3: Performance of our system for the different subtasks and comparison with overall best performing systems. The column heading *GCN* indicates *our system* and *Best* indicates overall best performing system.

4 Training Setup

In this section we present the settings and hyper-parameters used in our domain prediction system. In Table 2 we summarize the system settings.

5 Results

In this section, we present the performance of our systems corresponding to the different subtasks. In Table 3 we present the performance of our systems in terms of their accuracy, precision, recall and F1-score on the test data and compare them with the best results achieved for that subtask.

6 Error Analysis

We have inspected the outputs of our system on the development set. Thorough error analysis has revealed several type of errors. Aside from the errors propagated from the PoS tagger and dependency parser, some of the errors generated in the output are discussed bellow:

1. As some of the classes are very related and lots of common terms exist in some of these classes, the system has confused to detect the correct class.

   - **S1:** and so we have shown e i square of x is e i of x for all x .
   - **S2:** so , grad f at r t naught is a vector , whose dot product with any tangent vector is 0.

   These instances are from Subtask 1a. S1 is detected as *physics* (actual class: *math*) where as S2 is detected as *math* (actual class: *physics*). Same errors happen in case of Subtask 1b for classes: *cse* (Computer Science) and *com_tech*(Communication Technology)

2. Most of the candidate texts are single sentences. Sometimes system has failed to identify the correct class as it could not understand the contextual information from those single sentences. For example:

   - **S3:** Why that is what is the reason why?

System has failed to classify the domain of these sentences properly. From these errors, it is evident that the problem of domain selection (among highly related classes) is really a challenging task. The accuracy and F-1 score of our system are also lower than the best system submitted. In future we will apply some other data processing techniques and smarter machine learning models to improve our performance.

References

Charu C. Aggarwal and ChengXiang Zhai. 2012. A survey of text classification algorithms. pages 163–222.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proceedings of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 20–30, Vancouver, Canada. Association for Computational Linguistics.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. 2016. *Deep Learning*. MIT Press. http://www.deeplearningbook.org.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8):1735–1780.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.

Kamran Kowsari, Kiana Jafari Meimandi, Mojtaba Heidarysafa, Sanjana Mendu, Laura E. Barnes, and Donald E. Brown. 2019. Text classification algorithms: A survey.

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajič, Christopher Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proceedings of the 10th International Conference on Language Resources and Evaluation (LREC 2016)*, pages 1659–1666, Portorož, Slovenia. European Language Resources Association.

Peng Qi, Timothy Dozat, Yuhao Zhang, and Christopher D. Manning. 2018. Universal dependency parsing from scratch. In *Proceedings of the CoNLL 2018 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*, pages 160–170, Brussels, Belgium. Association for Computational Linguistics.

M. Schuster and K. K. Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Multichannel LSTM-CNN for Telugu Technical Domain Identification

Sunil Gundapu
Language Technologies Research Centre
KCIS, IIIT Hyderabad
Telangana, India
sunil.g@research.iiit.ac.in

Radhika Mamidi
Language Technologies Research Centre
KCIS, IIIT Hyderabad
Telangana, India
radhika.mamidi@iiit.ac.in

Abstract

With the instantaneous growth of text information, retrieving domain-oriented information from the text data has a broad range of applications in Information Retrieval and Natural language Processing. Thematic keywords give a compressed representation of the text. Usually, Domain Identification plays a significant role in Machine Translation, Text Summarization, Question Answering, Information Extraction, and Sentiment Analysis. In this paper, we proposed the Multichannel LSTM-CNN methodology for Technical Domain Identification for Telugu. This architecture was used and evaluated in the context of the ICON shared task "TechDOfication 2020" (task h), and our system got 69.9% of the F1 score on the test dataset and 90.01% on the validation set.

1 Introduction

Technical Domain Identification is the task of automatically identifying and categorizing a set of unlabeled text passages/documents to their corresponding domain categories from a predefined domain category set. The domain category set consists of 6 category labels: Bio-Chemistry, Communication Technology, Computer Science, Management, Physics, and Other. These domains can be viewed as a set of text passages, and test text data can be treated as a query to the system. Domain identification has many applications like Machine Translation, Summarization, Question Answering, etc. This task would be the first step for most downstream applications (i.e., Machine Translation). It decides the domain for text data, and afterward, Machine Translation can choose its resources as per the identified domain.

Majority of the research work in the area of text classification and domain identification has been done in English. There has been well below contribution for regional languages, especially Indian Languages. Telugu is one of India's old traditional languages, and it is categorized as one of the Dravidian language family. According to the Ethnologue[1] list, there are about 93 million native Telugu speakers, and it ranks sixteenth most-spoken languages worldwide.

We tried to identify the domain of Telugu text data using various Supervised Machine Learning and Deep Learning techniques in our work. Our Multichannel LSTM-CNN method outperforms the other methods on the provided dataset. This approach incorporates the advantages of CNN and Self-Attention based BiLSTM into one model.

The rest of the paper is structured as follows: Section 2 explains some related works of domain identification, Section 3 describes the dataset provided in the shared task, Section 4 addresses the methodology applied in the task, Section 5 presents the results and error analysis, and finally, Section 6 concludes the paper as well as possible future works.

2 Related Work

Several methods for domain identification and text categorization have been done on Indian languages, and few of the works have been reported on the Telugu language. In this section, we survey some of the methodologies and approaches used to address domain identification and text categorization.

Murthy (2005) explains the automatic text categorization with special emphasis on Telugu. In his research work, supervised classification using the Naive Bayes classifier has been applied to 800 Telugu news articles for text categorization. Swamy et al. (2014) work on representing and categorizing Indian language text documents using text mining techniques K Nearest Neighbour, Naive Bayes, and decision tree classifier.

[1]https://www.ethnologue.com/guides/ethnologue200

Categorization of Telugu text documents using language-dependent and independent models proposed by Narala et al. (2017). Durga and Govardhan (2011) introduced a model for document classification and text categorization. In their paper described a term frequency ontology-based text categorization for Telugu documents. Combining LSTM and CNN's robustness, Liu et al. (2020) proposed Attention-based Multichannel Convolutional Neural Network for text classification. In their network, BiLSTM encodes the history and future information of words, and CNN capture relations between words.

3 Dataset Description

We used the dataset provided by the organizers of Task-h of TechDOfication 2020 for training the models. The data for the task consists of 68865 text documents for training, 5920 for validation, 2611 for testing. For hyperparameter tuning, we used the validation set provided by the organizers. The statistics of the dataset are shown in Table 1. And the amount of texts for the dataset can be seen in Figure 1.

Labels(↓)	Train Data	Validation Data
cse	24937	2175
phy	16839	1650
com_tech	11626	970
bio_tech	7468	580
mgnt	2347	155
other	5648	390
Total	68865	5920

Table 1: Dataset Statistics

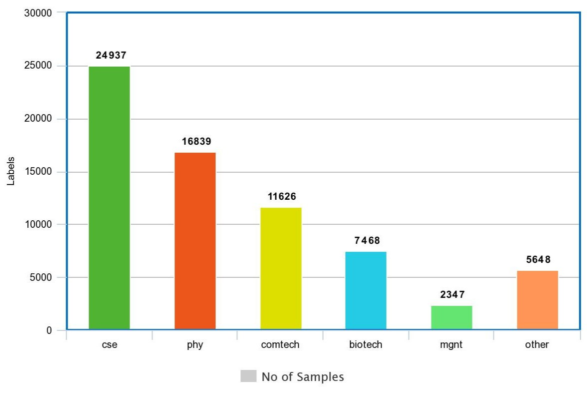


Figure 1: Number of samples per class

4 Proposed Approach

4.1 Data Preprocessing

The text passages have been originally provided in the Telugu script with the corresponding domain tags. The text documents have some noise, so before passing the text to the training stage, they are preprocessed using the following procedure:

- **Acronym Mapping Dictionary:** We created an acronym mapping dictionary. Expanded the English acronyms using the acronym mapping dictionary.

- **Find Language Words:** Sometimes, English words are co-located with Telugu words in the passage. We find the index of those words to translate into Telugu.

- **Translate English Words:** Translate the English words into the Telugu language, which are identified in the first stage of preprocessing. Google's Translation API[2] was used for this purpose.

- **Hindi Sentence Translation:** We can observe a few Hindi sentences in the dataset. We translated those sentences into Telugu using Google translation tool.

- **Noise Removal:** Removed the unnecessary tokens, punctuation marks, non-UTF format tokens, and single length English tokens from the text data.

4.2 Supervise Machine Learning Algorithms

To build the finest system for domain identification, we started with supervised machine learning techniques then moved to deep learning models. SVM, Multilayer Perceptron, Linear Classifier, Gradient Boosting methods performed very well on the given training dataset. These supervised models trained on the word level, n-gram level, and character level TF-IDF vector representations.

4.3 Multichannel LSTM-CNN Architecture

We started experiments with individual LSTM, GRU, CNN models with different word embeddings like word2vec, glove and fasstext. However, ensembling of CNN with self-attention LSTM model gave better results than individual models.

[2]https://py-googletrans.readthedocs.io/en/latest/

We develop a multichannel model for domain identification consisting of two main components. The first component is a long short time memory (Hochreiter and Schmidhuber, 1997) (henceforth, LSTM). The advantage of LSTM can handle the long term dependencies but does not store the global semantics of foregoing information in a variable-sized vector. The second component is a convolutional neural network (LeCun, 1989) (henceforth, CNN). The advantage of CNN can capture the n-gram features of text by using convolution filters, but it restricts the performance due to convolutional filters size. By considering the strengths of these two components, we ensemble the LSTM and CNN model for domain identification.

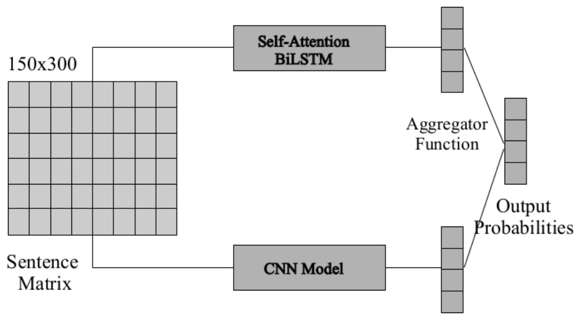


Figure 2: Multichannel LSTM-CNN Model

4.3.1 Self-Attention BiLSTM Classifier

The first module in architecture is Self-Attention based BiLSTM classifier. We employed this self-attention (Kelvin Xu and Bengion, 2015) based BiLSTM model to extract the semantic and sentiment information from the input text data. Self-attention is an intra-attention mechanism in which a softmax function gives each subword's weights in the sentence. The outcome of this module is a weighted sum of hidden representations at each subword.

The self-attention mechanism is built on BiLSTMs architecture (See figure 3), and it takes input as pre-trained embeddings of the subwords. We passed the Telugu fasttext (Grave et al., 2018) subword embeddings to a BiLSTM layer to get hidden representation at each timestep, which is the input to the self-attention component.

Suppose the input sentence S is given by the subwords $(w_1, w_2, ..., w_n)$. Let $\overrightarrow{h}$ represents the forward hidden state and $\overleftarrow{h}$ represents the backward hidden state at i^{th} position in BiLSTM. The merged representation k_i is obtained by combining

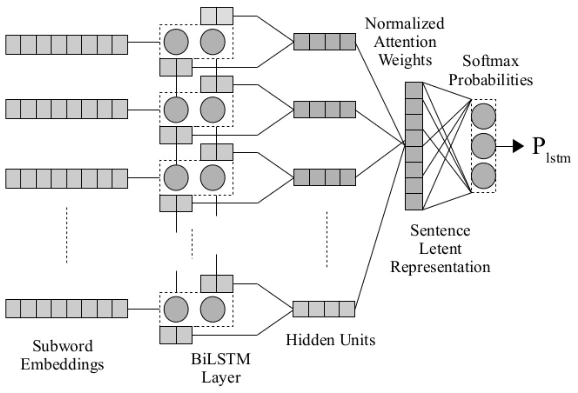


Figure 3: Self-Attention BiLSTM

the forward and backward hidden states. We concatenate the forward and backward hidden units to get the merged representations $(k_1, k_2, ...k_n)$.

$$k_i = [\overrightarrow{h_i}; \overleftarrow{h_i}] \tag{1}$$

The self-attention model gives a score e_i to each subword i in the sentence S, as given by below equation

$$e_i = k_i^T k_n \tag{2}$$

Then we calculate the attention weight a_i by normalizing the attention score e_i

$$a_i = \frac{exp(e_i)}{\sum_{j=1}^{n} exp(e_j)} \tag{3}$$

Finally, we compute the sentence S latent representation vector h using below equation

$$h = \sum_{i=1}^{a_i \times k_i} \tag{4}$$

The latent representation vector h is fed to a fully connected layer followed by a softmax layer to obtain probabilities P_{lstm}.

4.3.2 Convolutional Neural Network

The second component is CNN, which consider the ordering of the words and the context in which each word appears in the sentence. We present Telugu fasstext subword embeddings (Bojanowski et al. (2016)) of a sentence to 1D-CNN (See figure 4) to generate the required embeddings.

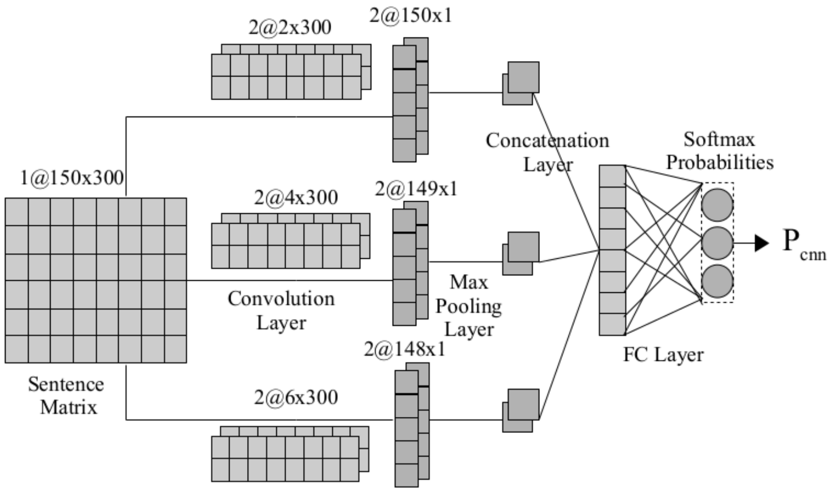


Figure 4: CNN Classifier

Initially, we present a $d \times S$ sentence embedding matrix to the convolution layer. Each row is a d-dimension fasstext subword embedding vector of each word, and S is sentence length. We perform convolution operations in the convolution layer with three different kernel sizes (2, 4, and 6). The purpose behind using various kernel sizes was to capture contexts of varying lengths and to extract local features around each word window. The output of convolution layers was passed to corresponding max-pooling layers. The max-pooling layer is used to preserve the word order and bring out the important features from the feature map. We change the original max-pooling layer in the convolution neural network with the word order-preserving k-max-pooling layer to preserve the inputted sentences word order. The order persevering max-pooling layer reduces the number of features while preserving the order of these words.

The max-pooling layer output is concatenated together fed to a fully connected layer followed by a softmax layer to obtain softmax probabilities P_{cnn}.

The CNN and BiLSTM models softmax probabilities are aggregated (See figure 2) using an element-wise product to obtain the final probabilities $P_{final} = P_{cnn} \circ P_{lstm}$. We tried various aggregation techniques like average, maximum, minimum, element-wise addition, and element-wise multiplication to combine LSTM and CNN models' probabilities. But an element-wise product gave better results than other techniques.

5 Results and Error Analysis

We first started our experiments with machine learning algorithms with various kinds of TF-IDF feature vector representations. SVM, MLP gave pretty good results on the validation dataset but failed on bio-chemistry and management data points. And CNN model with fasstext word embeddings seemed to be confused between the computer technology and cse data points (See table 2). Maybe the reason behind this confusion is that both datapoints quite similar at the syntactic level.

The self-attention based BiLSTM model outperforms the CNN model on physics, cse, and computer technology data points though it performs worse on the bio-chemistry and management data points. If we observe the training set, 75% of the data samples belong to physics, cse, and computer technology domains remaining 25% of data owned by the remaining domain labels. So we were assuming that an imbalanced training set was also one of the problems for misclassification of bio-chemistry and management domain samples. We tried to handle the this data-skewing problem with SMOTE (Chawla et al., 2002) technique, but it is doesn't work very well.

After observing the CNN and BiLSTM model results, we ensemble these two models for better results. The ensemble Multichannel LSTM CNN model outperforms all our previous models, achieving a recall of 0.90 with the weighted F1-score of 0.90 on the development dataset and a recall of 0.698 with an F1-score of 0.699 on the test dataset (See table 3).

	Validation Data				Test Data			
Model	Accuracy	Precision	Recall	F1-Score	Accracy	Precision	Recall	F1-Score
SVM	0.8612	0.8598	0.8601	0.866	-	-	-	-
CNN	0.8890	0.8874	0.8897	0.8902	0.6465	0.6558	0.6948	0.6609
BiLSTM	0.8706	0.8714	0.8702	0.8786	0.6465	0.6519	0.6927	0.6571
Multichannel	0.9001	0.9008	0.9006	0.9003	**0.6928**	**0.6984**	0.7228	0.6992
Organizers System	-	-	-	-	0.6855	0.6974	0.7289	0.7028

Table 2: Comparison between various model results

Measures(↓)	**Validation Data**	**Test Data**
precision	0.90	0.7228
recall	0.90	0.6984
f1score	0.90	0.6992
accuracy	0.9003	0.6928

Table 3: Performance of multichannel system

6 Conclusion

In this paper, we proposed a multichannel approach that integrates the advantages of CNN and LSTM. This model captures local, global dependencies, and sentiment in a sentence. Our approach gives better results than individual CNN, LSTM, and supervised machine learning algorithms on the Telugu TechDOfication dataset.

As discussed in the previous section, we will handle the data imbalance problem efficiently in future work. And we will improve the performance of the unambiguous cases in cse and computer technology domains.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Nitesh Chawla, Kevin Bowyer, Lawrence Hall, and W. Kegelmeyer. 2002. Smote: Synthetic minority over-sampling technique. *J. Artif. Intell. Res. (JAIR)*, 16:321–357.

A. Durga and A. Govardhan. 2011. Ontology based text categorization - telugu documents.

Edouard Grave, Piotr Bojanowski, Prakhar Gupta, Armand Joulin, and Tomas Mikolov. 2018. Learning word vectors for 157 languages. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9:1735–80.

Ryan Kiros Kyunghyun Cho Aaron C. Courville Ruslan Salakhutdinov Richard S. Zemel Kelvin Xu, Jimmy Ba and Yoshua Bengion. 2015. Show, attend and tell: Neural image caption generation with visual attention. *CoRR*, pages 1735–80.

Y. LeCun. 1989. Generalization and network design strategies.

Zhenyu Liu, Haiwei Huang, Chaohong Lu, and Shengfei Lyu. 2020. Multichannel cnn with attention for text classification. *ArXiv*, abs/2006.16174.

Kavi Murthy. 2005. Automatic categorization of telugu news articles.

Swapna Narala, B. P. Rani, and K. Ramakrishna. 2017. Telugu text categorization using language models. *Global journal of computer science and technology*.

M. N. Swamy, M. Hanumanthappa, and N. Jyothi. 2014. Indian language text representation and categorization using supervised learning algorithm. *2014 International Conference on Intelligent Computing Applications*, pages 406–410.

Multilingual Pre-Trained Transformers and Convolutional NN Classification Models for Technical Domain Identification

Suman Dowlagar
LTRC
IIIT-Hyderabad
suman.dowlagar@research.iiit.ac.in

Radhika Mamidi
LTRC
IIIT-Hyderabad
radhika.mamidi@iiit.ac.in

Abstract

In this paper, we present a transfer learning system to perform technical domain identification on multilingual text data. We have submitted two runs, one uses the transformer model BERT, and the other uses XLM-ROBERTa with the CNN model for text classification. These models allowed us to identify the domain of the given sentences for the ICON 2020 shared Task, TechDOfication: Technical Domain Identification. Our system ranked the best for the subtasks 1d, 1g for the given TechDOfication dataset.

1 Introduction

Automated technical domain identification is a categorization/classification task where the given text is categorized into a set of predefined domains. It is employed in tasks like Machine Translation, Information Retrieval, Question Answering, Summarization, and so on.

In Machine Translation, Summarization, Question Answering, and Information Retrieval, the domain classification model will help leverage the contents of technical documents, select the appropriate domain-dependent resources, and provide personalized processing of the given text.

Technical domain identification comes under text classification or categorization. Text classification is one of the fundamental tasks in the field of NLP. Text classification is the process of identifying the category where the given text belongs. Automated text classification helps to organize unstructured data, which can help us gather insightful information to make future decisions on downstream tasks.

Traditional text classification approaches mainly focus on feature engineering techniques such as bag-of-words and classification algorithms (Yang, 1999). Nowadays, the sate-of-the-art results on text classification are achieved by various NNs such as CNN (Kim, 2014), LSTM (Hochreiter and Schmidhuber, 1997), BERT (Adhikari et al., 2019), and Text GCN (Adhikari et al., 2019). Attention mechanisms (Vaswani et al., 2017) have been introduced in these models, which increased the representativeness of the text for better classification. Transformer models such as BERT (Devlin et al., 2018) uses the attention mechanism that learns contextual relations between words or sub-words in a text. Text GCN (Yao et al., 2019) uses a graph-convolutional network to learn a heterogeneous word document graph on the whole corpus, which helped classify the text. However, of all the deep learning approaches, transformer models provided SOTA results in text classification.

In this paper, We present two approaches for technical domain identification. One approach uses the pre-trained Multilingual BERT model, and the other uses XLM-ROBERTa with CNN model.

The rest of the paper is structured as follows. Section 2 describes our approach in detail. In Section 3, we provide the analysis and evaluation of results for our system, and Section 4 concludes our work.

2 Our Approach

Here we present two approaches for the TechDOfication task.

2.1 BERT for TechDOfication

In the first approach, we use the pre-trained multilingual BERT model for domain identification of the given text. Bidirectional Encoder Representations from Transformers (BERT) is a transformer encoder stack trained on the large corpora. Like the vanilla transformer model (Vaswani et al., 2017), BERT takes a sequence of words as input. Each layer applies self-attention, passes its results through a feed-forward network, and then hands

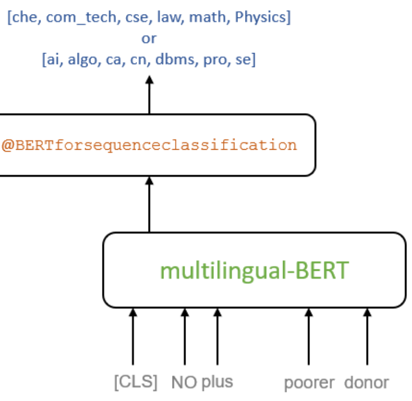

[che, com_tech, cse, law, math, Physics]
or
[ai, algo, ca, cn, dbms, pro, se]
@BERTforsequenceclassification
multilingual-BERT
[CLS] NO plus poorer donor

Figure 1: The architecture of the BERT model for sentence classification.

it off to the next encoder. The BERT configuration model takes a sequence of words/tokens at a maximum length of 512 and produces an encoded representation of dimensionality 768.

The pre-trained multilingual BERT models have a better word representation as they are trained on a large multilingual Wikipedia and book corpus. As the pre-trained BERT model is trained on generic corpora, we need to finetune the model for the given domain identification tasks. During finetuning, the pre-trained BERT model parameters are updated.

In this architecture, only the [CLS] (classification) token output provided by BERT is used. The [CLS] output is the output of the 12th transformer encoder with a dimensionality of 768. It is given as input to a fully connected neural network, and the softmax activation function is applied to the neural network to classify the given sentence.

2.2 XLM-ROBERTa with CNN for TechDOfication

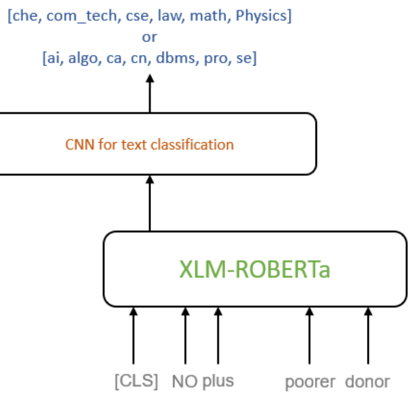

[che, com_tech, cse, law, math, Physics]
or
[ai, algo, ca, cn, dbms, pro, se]
CNN for text classification
XLM-ROBERTa
[CLS] NO plus poorer donor

Figure 2: The architecture of the XLM-ROBERTa with CNN for sentence classification.

XLM-ROBERTa (Conneau et al., 2019) is a transformer-based multilingual masked language model pre-trained on the text in 100 languages, which obtains state-of-the-art performance on cross-lingual classification, sequence labeling, and question answering. XLM-ROBERTa improves upon BERT by adding a few changes to the BERT model such as training on a larger dataset, dynamically masking out tokens compared to the original static masking, and uses a known preprocessing technique (Byte-Pair-Encoding) and a dual-language training mechanism with BERT in order to learn better relations between words in different languages. The given model is trained for the language modeling task, and the output is of dimensionality 768. It is given as input to a CNN (Kim, 2014) because convolution layers can extract better data representations than Feed Forward layers, which indirectly helps in better domain identification.

3 Experiment

This section presents the datasets used, the task description, and two models' performance on technical domain identification. We also include our implementation details and error analysis in the subsequent sections.

3.1 Dataset

We used the dataset provided by the organizers of TechDOfication ICON-2020. There are two subtasks, one is coarse-grained, and the other is fine-grained. The coarse-grained TechDOfication dataset contains sentences about Chemistry, Communication Technology, Computer Science, Law, Math, and Physics domains in different languages such as English, Bengali, Gujarati, Hindi, Malayalam, Marathi, Tamil, and Telugu. Whereas the fine-grained English dataset focuses on the Computer-Science domain with sub-domain labels as Artificial Intelligence, Algorithm, Computer Architecture, Computer Networks, Database Management system, Programming, and Software Engineering.

3.2 Implementation

For the implementation, we used the transformers library provided by HuggingFace[1]. The HuggingFace contains the pre-trained multilingual BERT, XLM-ROBERTa, and other models suitable

[1]https://huggingface.co/

for downstream tasks. The pre-trained multilingual BERT model used is *"bert-base-multilingual-cased"* and pre-trained XLM-R model used is *"xlm-roberta-base"*. We programmed the CNN architecture as given in the paper (Kim, 2014). We used the PyTorch library that supports GPU processing for implementing deep neural nets. The BERT models were run on the Google Colab and Kaggle GPU notebooks. We trained our classifier with a batch size of 128 for 10 to 30 epochs based on our experiments. The dropout is set to 0.1, and the Adam optimizer is used with a learning rate of 2e-5. We used the hugging face transformers pre-trained BERT tokenizer for tokenization. We used the BertForSequenceClassification module provided by the HuggingFace library during finetuning and sequence classification for the multilingual-BERT based approach.

3.3 Baseline models

Here, we compared the BERT model with other machine learning algorithms.

SVM with TF_IDF text representation We chose Support Vector Machines (SVM) with TF_IDF text representation for technical domain identification. SVM classifier and TF_IDF vector representation is obtained from the scikit-learn library (Pedregosa et al., 2011).

CNN: Convolutional Neural Network (Kim, 2014). We explored CNN-non-static, which uses pre-trained word embeddings.

3.4 Results

The results are tabulated in Table 1. We evaluated the performance of the method using macro F1. The multilingual-BERT model performed well when compared to the other SVM with TF-IDF and CNN models. Given all the languages, we have observed an increase of 7 to 25% in classification metrics for BERT compared to the baseline SVM classifier, it showed a 2 to 5% increase in classification metrics compared to the CNN classifier on the validation data. On the test data, multilingual BERT showed better performance in subtasks 1a, 1b, 1c, 1h and 2a whereas XLM-ROBERTa with CNN showed better performance in the subtasks 1d, 1e, 1f, 1g. This increase in classification metrics is due to the transformer model's and convolutional NN's capability, which learned better text representations from the generic data than other models.

4 Error Analysis

The multilingual-BERT model's confusion matrix is compared with the poorly performed model for languages, Hindi, and Tamil languages are shown in Figure 3. We chose Hindi and Tamil languages because, here, the difference in performance is more significant. For the Hindi subtask, the SVM classifier confused between "cse", "com_tech", and "mgmt" labels, whereas the BERT model performed better. For the Tamil subtask, the SVM classifier confused between "com_tech" and "mgmt" labels, whereas the BERT model performed better than the other models. This is because both the approaches (pre-trained multilingual-BERT and pre-trained XLM-ROBERTa with CNN) learned better representation of the above data than the other models that helped in technical document identification.

5 Conclusion and Future work

We used pre-trained bi-directional encoder representations using multilingual-BERT and XLM-ROBERTa with CNN technical domain identification for English, Bengali, Gujarati, Hindi, Malayalam, Marathi, Tamil, and Telugu languages. We compared the approaches with the baseline methods. Our analysis showed that pre-trained multilingual BERT and XLM-ROBERTa with CNN models and finetuning it for text classification tasks showed an increase in macro F1 score and accuracy metrics compared to baseline approaches.

Some datasets are large, like for the Hindi, Tamil, and Telugu, we can train the BERT and XLM-ROBERTa models from scratch and consider its hidden layer representation, and concatenate this with the representation of the pre-trained model. It might help to classify the datasets even better.

References

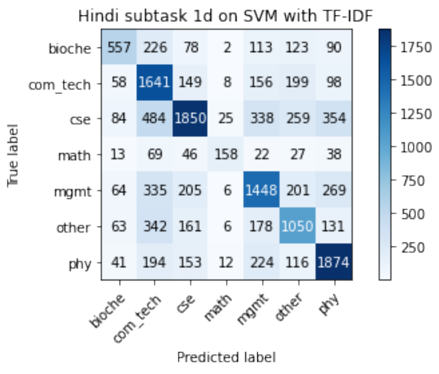
Ashutosh Adhikari, Achyudh Ram, Raphael Tang, and Jimmy Lin. 2019. Docbert: Bert for document classification. *arXiv preprint arXiv:1904.08398*.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

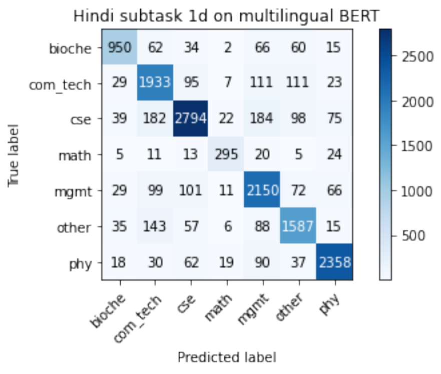
Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

	Classifier Models					
Dataset	**Validation**				**Test**	
	SVM	**CNN**	**M-Bert**	**XLM-R+CNN**	**M-Bert**	**XLM-R+CNN**
English subtask-1a	81.48	83.05	**88.87**	87.09	**79.84**	73.57
Bengali subtask-1b	66.35	85.78	**86.81**	85.71	**80.35**	78.17
Gujarati subtask-1c	69.63	86.27	**87.21**	86.89	**68.67**	66.73
Hindi subtask-1d	58.21	81.03	**83.40**	82.13	59.89	**60.44**
Malayalam subtask-1e	80.60	92.51	**94.72**	93.40	34.47	**34.86**
Marathi subtask-1f	73.32	86.89	**87.42**	86.37	59.52	**59.89**
Tamil subtask-1g	65.95	85.75	**87.50**	86.54	49.24	**51.34**
Telugu subtask-1h	71.98	88.07	**90.28**	89.43	**67.17**	62.26
English subtask-2a	70.24	72.53	**77.36**	76.77	**78.98**	78.07

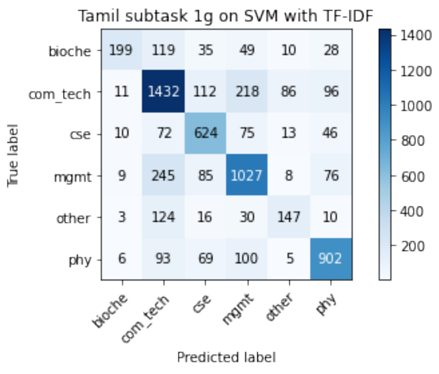
Table 1: macro F1 on validation and test data for all the subtasks



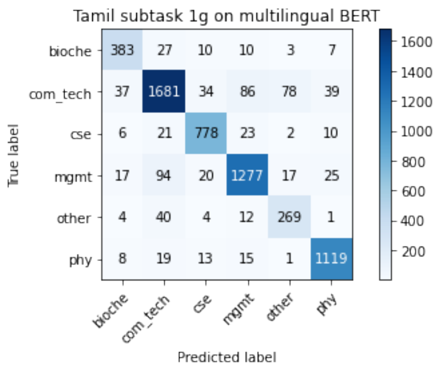
(a)



(b)



(c)



(d)

Figure 3: Confusion matrix on the given validation data for the Hindi and Tamil languages

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Yiming Yang. 1999. An evaluation of statistical approaches to text categorization. *Information retrieval*, 1(1-2):69–90.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Technical Domain Identification using word2vec and BiLSTM

Koyel Ghosh
ghosh.koyel8@gmail.com
Dr. Apurbalal Senapati
a.senapati@cit.ac.in
Dr. ranjan Maity
r.maity@cit.ac.in
CSE Department
Central Institute of Technology, Kokrajhar(Assam)

Abstract

Coarse-grained and Fine-grained classification tasks are mostly based on sentiment or basic emotion analysis. Now, switching from emotion and sentiment analysis to another domain, in this paper, we are going to work on technical domain identification. The task is to identify the technical domain of a given English text. In the case of Coarse-grained domain classification, such a piece of text provides information about specific Coarse-grained technical domains like Computer Science, Physics, Math, etc, and in Fine-grained domain classification, Fine-grained subdomains for Computer science domain, it can be like Artificial Intelligence, Algorithm, Computer Architecture, Computer Networks, Database Management system, etc. To do the task, Word2Vec skip-gram model is used for word embedding, later, applied the Bidirectional Long Short Term memory (BiLSTM) model to classify Coarse-grained domains and Fine-grained sub-domains. To evaluate the performance of the approached model accuracy, precision, recall, and F1-score have been applied.

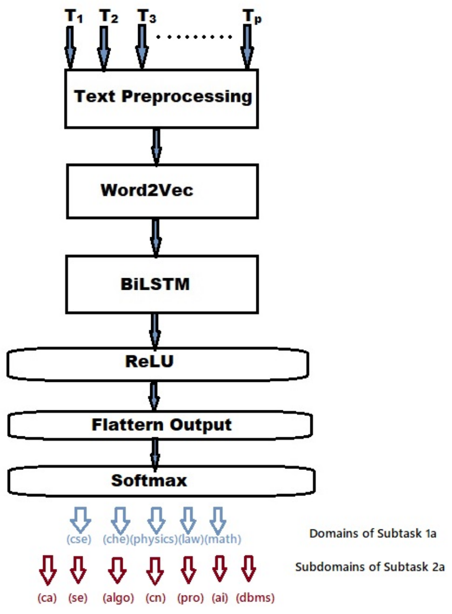
1 Introduction

ICON2020[1] has organized a shared task, details here: https://ssmt.iiit.ac.in/techdofication.html where they share some DATASETs for the Shared Task on Identification of a Technical Domain from Text. Among them, here, we are working with Subtask-1a Coarse-grained Domain Classification - English and Subtask-2a Fine-grained Domain Classification - Computer Science datasets. In this paper, system description of our approached model on identification of a technical domain from text and the result of this approach has been discussed. There are lots of work already have done successfully (Akhtar et al., 2020) in the coarse-grained and fine-grained classification with sentiment (Cortis et al., 2017) and emotion analysis (Mohammad and Bravo-Marquez, 2017) dataset. Often, in the classification task, Word2Vec or fast-text or GloVe or all-combined approach (Salur and Aydin, 2020) is used to utilize the effectiveness of different word embedding algorithms. To get a better result on a domain specific corpus Occupational Safety and Health Administration(OSHA), a hybrid deep neural network with Word2Vec was used (Zhang, 2019). ESIM with SuBiLSTM (Ensemble) and ESIM with SuBiLSTM-Tied (Ensemble) approaches (Brahma, 2018) performed well on the Stanford Sentiment Treebank dataset (Socher et al., 2013), both in its binary (SST-2) and fine-grained (SST-5) forms. A similar approach is applied to the question classification i.e TREC dataset (Voorhees, 2006), both in its 6 class(TREC-6) and 50 class (TREC-50) forms. Bidirectional dilated LSTM with attention (Schoene et al., 2020) used for another fine-grained dataset (Klinger et al., 2018). (Melamud et al., 2016) proposed a BiLSTM neural network architecture based on Word2vec's CBOW architecture. Some very old approach on domain classification (Bernier-Colborne et al., 2017). (Zhang, 2019) is based on accident causes classification with the approach deep learning and Word2Vec, they compare their model with others where bi-gram, n-gram was used for text representation. In (Xie et al., 2019), author added attention layer with BiLSTM for short text fine-grained sentiment classification to get a better accuracy.

2 Methodology

In this section, dataset, data preprocessing, word embedding and the structure of the BiLSTM with Word2Vec model will be discussed. Approached architechture of BiLSTM with Word2Vec is shown in Figure 1

[1]https://www.iitp.ac.in/ai-nlp-ml/icon2020/shared$_{t}asks.html$



T_1 T_2 T_3 .... T_p
Text Preprocessing
Word2Vec
BiLSTM
ReLU
Flattern Output
Softmax
(cse) (che)(physics)(law)(math)
Domains of Subtask 1a
Subdomains of Subtask 2a
(ca) (se) (algo) (cn) (pro) (ai) (dbms)

Figure 1: Architechture of BiLSTM + Word2Vec with ReLU and Softmax on the top

2.1 Dataset

Here, Technical Domain Identification dataset[2] in its Coarse-grained Domain Classification - English (Subtask-1a) form and Fine-grained Domain Classification - Computer Science (Subtask-2a) form has been used. Table 1 shows the details of the dataset for the shared task. In case of Coarse-grained Domain Classification, a piece of text which provides information about specific Coarse-grained domain like computer science domain. No of domains or classes are: Computer Science (cse), Chemistry (che), Physics (phy), Law (law), Math (math). In case of Fine-grained Domain Classification, no of subdomains or classes from Computer Science are: Computer Architecture (ca), Software Engineering (se), Algorithm (algo), Computer Networks (cn), Programming (pro), Artificial Intelligence (ai), Database Management system (dbms). Table 2 shows the details of the domains and subdomains distribution in traning and dev dataset. For prediction purpose test set has been provided without labelling of domains or sub domains. So, later in this paper, dev set is used to evaluate the accuracy.

2.2 Preprocessing of the Data

Based on the analysis from previous studies, deep learning needs text data in numeric form. To encode text data into a numeric vector, lots of encoding techniques like Bag of words, Bi-gram, n-gram, TF-IDF, Word2Vec etc are used. So, before encoding, text data need to be cleaned, noise-free to increase the classification performance. Figure 2 shows all the intermediate steps of preprocessing.

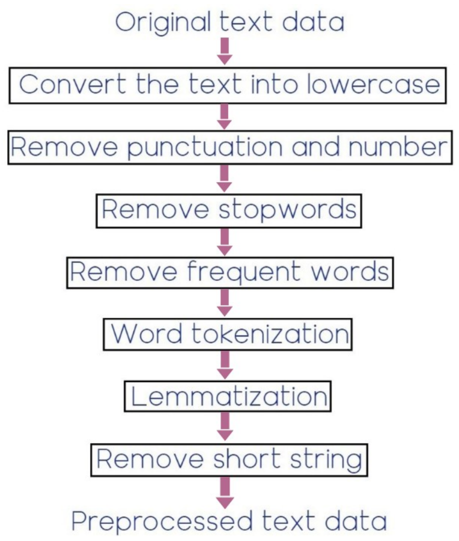


Figure 2: Text preprocessing steps used in this study

Cleaning or preprocessing of the data is as important as model building. Text preprocessing procedure can be different depending on the task and dataset we use. In our case, we used following steps:

Convert the text into lowercase: All words should be either in lower or uppercase to avoid redundancy. Suppose there are two words "artificial" and "Artificial", machine will treat them as separate word if we avoid this step.

Removing punctuation and number: Punctuation and numbers often doesn't add extra meaning to the text. This text has several punctuation ( ) , ; . etc and numbers (0-9). String library which has 32 punctuation, is used to remove all these punctuation from text to get better result.

Removing stopwords: The most common words in a language like "the", "a", "have", "is", "to" etc are called stopwords. As these words do not add any important meaning, these can be removed.

Remove frequent word: Some words which are frequently used in text but not listed in stopwords has been removed.

Word tokenization: To break the long sentence into words, we applied tokenization.

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

Dataset	Training	Dev	Test	label	Length(Max)	Vocabulary size
Subtask-1a	23,959	4,850	2,500	5	74	10,722
Subtask-2a	13,580	1,360	1,930	7	266	7,397

Table 1: Data set

Dataset	Domain	Train	Dev
Subtask-1a	cse	4,770	970
	che	4,733	970
	physics	4,787	970
	law	4,829	970
	Math	4,840	970
Subtask-2a	ca	1,947	180
	se	1,940	200
	algo	1,951	200
	cn	1,940	180
	pro	1,922	200
	ai	1,940	200
	dbms	1,940	200

Table 2: Dataset statistics

Lemmatization: Stemming and lemmatization both processes have almost the same goal i.e. to reduce inflectional forms of each word and convert those to a common root form but both are different in the sense of result we get. Stemming simply chop off the inflections of each word but sometimes the resultant word may not carry any valid meaning but lemmatization does it properly with the use of language's full vocabulary to apply a morphological analysis to the words and return the base or dictionary form of a word, which is known as the lemma so the words can be analyzed as a single item.

Here, the effectiveness of lemmatization process has been applied on the text to get the desired result.

Remove short string: After performing all the required processes in text processing, still, some words are in the text which is very short in length. So, it required to remove the words having a length less than or equal to 2.

Label encoding: As labelled domains and the subdomains on the texts, are words so, we need to encode them into an unique number. Like, cse - 0, che - 1, physics - 2, law - 3, math - 4 for subtask 1a and ca - 0, se - 1, algo - 2, cn - 3, pro - 4, ai - 5, dbms - 6 for subtask 2a.

Here, we use Natural Language Toolkit (NLTK)(Wagner, 2010) for tokenization, lemmatization and removing stopwords. After these steps we get the maximum length of a sentence i.e maximum number of words present in a sentence as mentioned in Table 1.

2.3 Deep neural network with Word2Vec

In this study, deep neural network with Word2Vec approach is applied. The entire methodology of this approach has two phases: training of Word2Vec skip-gram model on the datasets to get the vocabulary and the text representation, then deep neural network is used utilizing the learned word embedding in the previous step.

2.3.1 Word Embedding

Any neural network model needs a vector representation of a word. So, we need an embedding layer before building a deep learning model.

Word2Vec models proposed by Thomas Mikolov at google (Mikolov et al., 2013), are used for learning word embedding. The advantage of Word2Vec is that similarity and relationship between words can be derived from the learned vector (Khatua et al., 2019). It can be obtained using two methods (both involve neural network): Skip-gram and Common Bag of Words (CBOW).

As mentioned in (Mikolov et al., 2013), skip-gram works great with a small amount of data and does well to represent rare words. On the other hand, CBOW is faster and has better representation for more frequent words.

TechinalDOfication dataset has some rare words like "streptococcus", "polymerization"," hessian", "kyoto" as these words are related to specific technical domain. So, to represent these words well we are using Word2Vec here.

The training dataset consisting of p numbers of texts is denoted as

$$D = \{T_1, T_2, T_3, .., T_i, ...T_p\}$$

where T_i is the ith number of text and p is equal to the total numbers of texts present in a training dataset *e.g.* 23,959 in Subtask-1a and 13,580 in Subtask-2a. Given a text T_i, the text having m words i.e length of the text is denoted as

$$T_i = \{w_{i,1}, w_{i,2}, w_{i,3}, ..., w_{i,k}, .., w_{i,m}\},$$

where $w_{i,k}$ denotes the kth word in the ith text. Now, Word2Vec skip-gram model is trained using the training dataset used for this study. To train word embedding, we fit the parameters as embedding dimension = 300, window = 10 and saved the trained Word2Vec model for the next step.

We embed each word $w_{i,k}$ to our pre-trained word vector after loading the model into memory i.e each word in the text is converted into a d-dimension embedding vector, where $w^v_{i,k} \in \mathrm{R}^d$ is d-dimension embedding vector of kth word. The word level embedding as

$$T^v_i = \{w^v_{i,1}, w^v_{i,2}, w^v_{i,3}, ..., w^v_{i,k}, ..., w^v_{i,m}\}.$$

Figure 3 shows the Word2Vec architecture, where $H = H_1, H_2, H_3, ..H_n$ is a hidden layer.

2.3.2 Classification model

LSTM is an extension of Recurrent Neural Network (RNN) (Hochreiter and Schmidhuber, 1997), capable of learning long dependencies. They were introduced by (Sulehria and Zhang, 2007).

In this section, deep neural network BiLSTM is used for the classification. Now, we give T^v_i as input to BiLSTM for feature extraction, namely F^{BiLSTM}_i in equation

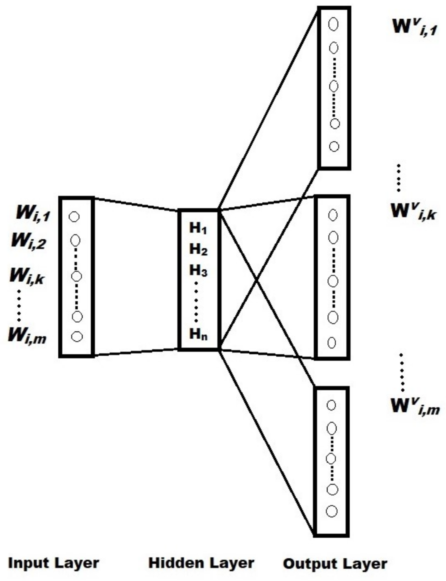


Figure 3: Word2Vec architechture

$$F^{BiLSTM}_i = BiLSTM(T^v_i) \quad (1)$$

In Bidirectional LSTM, sequence data is pro-

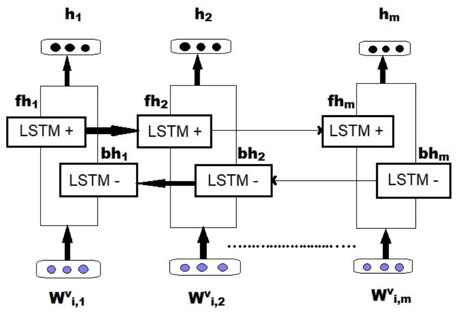


Figure 4: The architecture of basic BiLSTM

cessed in both directions with forward LSTM and backward LSTM layer and these two hidden layer connected to the same output layer. The LSTM neural networks contain three gates and a cell memory state. For a single LSTM cell, it can be computed as

$$X = \frac{h_t - 1}{w^v_{i,k}} \quad (2)$$

$$f_t = \sigma(W_f.X + b_f) \quad (3)$$

$$i_t = \sigma(W_i.X + b_i) \quad (4)$$

$$o_t = \sigma(W_o.X + b_o) \quad (5)$$

$$c_t = f_t * c_{t-1} + i_t * tanh(W_c.X + b_c) \quad (6)$$

$$h_t = o_t * tanh(c_t) \quad (7)$$

where W_f, W_i, W_0 are the weight matrices and b_f, b_i, b_0 are the bias of LSTM cell during training. σ denotes the sigmoid function. $w^v_{i,k}$ is the word embedding vector as input unit to LSTM, h_t is the hidden vector, So, h_m can denote a text. Simple BiLSTM architecture is shown in Figure 4. In the architecture $\{w^v_{i,1}, w^v_{i,2}, w^v_{i,3}, ..., w^v_{i,k}, ..., w^v_{i,m}\}$ denotes the word vector, m is the length of a text. $\{fh_1, fh_2, ..., fh_m\}$ and $\{bh_1, bh_2, ..., bh_m\}$ represent the forward hidden vector and the backward hidden vector respectively. $\{h_1, h_2, h_3, .., h_t, .., h_m\}$ represents final hidden layer. the final hidden vector h_t of the BiLSTM is shown as following equation:

$$h_t = [fh_t, bh_t] \tag{8}$$

In the BiLSTM layer, 20% dropout is used. After feeding input to BiLSTM layer, time Distributed wrapper is used along with dense layer where the activation function is rectified linear unit $(ReLU)$ and on the top of the layers dense is applied with $softmax$ activation function after $Flatten$ the output generated from the previous layer.

3 Result and conclusion

Domain	Precision	Recall	f1-score
cse	0.26	0.33	0.29
che	0.17	0.21	0.19
physics	0.23	0.19	0.21
law	0.22	0.16	0.19
math	0.16	0.15	0.16

Table 3: Result of CITK (our team) on Subtask-1a dataset (dev set)

Domain	Precision	Recall	f1-score
ca	0.29	0.23	0.26
se	0.16	0.17	0.16
algo	0.11	0.14	0.13
cn	0.14	0.14	0.14
pro	0.18	0.17	0.18
ai	0.18	0.20	0.19
dbms	0.23	0.20	0.22

Table 4: Result of CITK (our team) on Subtask-2a dataset (dev set)

As, it was a prediction task on test dataset and presently, we don't have labeled test dataset, dev dataset is used here to evaluate the model performance. From the Table 3 and Table 4, we can see that BiLSTM with Word2Vec didn't produce any good Precision, Recall, f1-score and accurecy 22% on both cases which are also very low. To evaluate the model performance, F1 score proposed by Buckland and Gey (Buckland and Gey, 1994) has been widely used in literature.

Team Name	Accuracy	Precision	Recall	f1-score
ICON2020	0.8156	0.8155	0.8156	0.8143
CITK	0.2204	0.2264	0.2204	0.2204

Table 5: comparison between highest score and our score(CITK) on Subtask-1a dataset (test set)

Team Name	Accuracy	Precision	Recall	f1-score
fineapples	0.8252	0.8265	0.8252	0.8244
CITK	0.2306	0.2344	0.2302	0.2307

Table 6: comparison between highest score and our score(CITK) on Subtask-2a dataset (test set)

Table 5 and Table 6 shows the result on test dataset published by ICON2020. Here, we only include highest score along with our score to show the comparison of the performances. In case of Subtask-1a dataset, "ICON2020" team produced good accuracy, precision, recall and f1-score compared to other teams including our "CITK" team. "Fineapples" team produced best result for Subtask-2a dataset.

After applying preprocessing on the texts of Subtask 2a dataset, we get maximum sentence length is 266 but other texts are not that long except one. Here, we trained Word2Vec model only with the given training dataset which is very small dataset to perform good embedding. Word embedding training gives us 10,722 unique words in subtask 1a and 7,397 in subtask 2a. Quite obvious, dealing with Fine grained dataset compare to Coarse grained dataset is somehow challenging as vocabulary size is very small.

Word embedding seems very important here, In most of the cases pre-trained word embedding such as Google News dataset[3] (about 100 billion words) is used, which contains 300-dimensional vectors for 3 million words and phrases. The archive is available here: `GoogleNews-vectors-negative300.bin.gz`. but in our case, some words like "streptococcus", "poly-merization"," hessian", "kyoto" are missing from the large Google News dataset. Those words are very important to

[3]https://code.google.com/archive/p/word2vec/

identify the specific domains, so, those words can't be ignored. Combining Google News dataset and ICON2020 shared task dataset with can be a solution that needs some experiments such as concatenating them removing possible correlations by performing Principal component analysis (PCA)(Basirat, 2018).

References

M. S. Akhtar, A. Ekbal, and E. Cambria. 2020. How intense are you? predicting intensities of emotions and sentiments using stacked ensemble [application notes]. *IEEE Computational Intelligence Magazine*, 15(1):64–75.

Ali Basirat. 2018. A generalized principal component analysis for word embedding.

Gabriel Bernier-Colborne, Caroline Barrière, and Pierre André Ménard. 2017. Fine-grained domain classification of text using termium plus.

Siddhartha Brahma. 2018. Improved sentence modeling using suffix bidirectional lstm. *arXiv: Learning*.

Michael Buckland and Fredric Gey. 1994. The relationship between recall and precision. *J. Am. Soc. Inf. Sci.*, 45(1):12–19.

Keith Cortis, Andre Freitas, Tobias Daudert, Manuela Hurlimann, Manel Zarrouk, Siegfried Handschuh, and Brian Davis. 2017. Semeval-2017 task 5: Fine-grained sentiment analysis on financial microblogs and news.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Aparup Khatua, Apalak Khatua, and Erik Cambria. 2019. A tale of two epidemics: Contextual word2vec for classifying twitter streams during outbreaks. *Information Processing Management*, 56:247–257.

Roman Klinger, Orphee de clercq, Saif Mohammad, and Alexandra Balahur. 2018. Iest: Wassa-2018 implicit emotions shared task.

Oren Melamud, Jacob Goldberger, and Ido Dagan. 2016. context2vec: Learning generic context embedding with bidirectional lstm. pages 51–61.

Tomas Mikolov, Ilya Sutskever, Kai Chen, G.s Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *Advances in Neural Information Processing Systems*, 26.

Saif Mohammad and Felipe Bravo-Marquez. 2017. Wassa-2017 shared task on emotion intensity. pages 34–49.

Mehmet Salur and Ilhan Aydin. 2020. A novel hybrid deep learning model for sentiment classification. *IEEE Access*, 8:58080 – 58093.

Annika Marie Schoene, Alexander P. Turner, and Nina Dethlefs. 2020. Bidirectional dilated lstm with attention for fine-grained emotion classification in tweets. In *AffCon@AAAI*.

Richard Socher, A. Perelygin, J.Y. Wu, J. Chuang, C.D. Manning, A.Y. Ng, and C. Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. *EMNLP*, 1631:1631–1642.

Humayun Karim Sulehria and Ye Zhang. 2007. Hopfield neural networks: A survey. In *Proceedings of the 6th Conference on 6th WSEAS Int. Conf. on Artificial Intelligence, Knowledge Engineering and Data Bases - Volume 6*, AIKED'07, page 125–130, Stevens Point, Wisconsin, USA. World Scientific and Engineering Academy and Society (WSEAS).

Ellen Voorhees. 2006. The trec question answering track. *Nat. Lang. Eng*, 7:361–378.

Wiebke Wagner. 2010. Natural language processing with python, analyzing text with the natural language toolkit by steven bird; ewan klein; edward loper. *Language Resources and Evaluation*, 44:421–424.

Jun Xie, Bo Chen, Xinglong Gu, Fengmei Liang, and Xinying Xu. 2019. Self-attention-based bilstm model for short text fine-grained sentiment classification. *IEEE Access*, 7:1–1.

Fan Zhang. 2019. A hybrid structured deep neural network with word2vec for construction accident causes classification. *International Journal of Construction Management*, pages 1–21.

Automatic Technical Domain Identification

Hema Ala
LTRC, IIIT-Hyderabad, India
hema.ala@research.iiit.ac.in

Dipti Misra Sharma
LTRC, IIIT-Hyderabad, India
dipti@iiit.ac.in

Abstract

In this paper we present two Machine Learning algorithms namely Stochastic Gradient Descent and Multi Layer Perceptron to Identify the technical domain of given text as such text provides information about the specific domain. We performed our experiments on Coarse-grained technical domains like Computer Science, Physics, Law, etc for English, Bengali, Gujarati, Hindi, Malayalam, Marathi, Tamil, and Telugu languages, and on fine-grained sub domains for Computer Science like Operating System, Computer Network, Database etc for only English language. Using TFIDF as a feature extraction method we show how both the machine learning models perform on the mentioned languages.

1 Introduction

We can frame Automatic Domain Identification of given text as a text classification problem where one needs to assign predefined categories to given texts. Text classification is a classic topic for Natural Language Processing (NLP), the range of text classification research goes from designing the best features to choosing the best possible machine learning classifiers. Therefore we use Term Frequency & Inverse Document Frequency (TFIDF) to represent our text in terms of vectors, so that the machine learning algorithms will find the relationships between them and classifies the given text. Many machine learning algorithms showed the best performances on text classification , but very limited number studies have explored technical domains like computer science, chemistry, management, etc that too on Indian languages (Kaur and Saini, 2015). There are numerous applications of text classification in Natural Language Processing Tasks like Machine Translation,etc. For these tasks technical domain identification would be the first process. It determines the domain for a given input text, subsequently Machine Translation can choose its resources as per the identified domain. The task can also be viewed at the coarse-grained or fine-grained level based on the requirement. We did our experiments on data provided by ICON TechDOfication-2020 shared task, for English, Bengali, Gujarati, Hindi , Malayalam , Marathi, Tamil and Telugu languages for coarse grained domain classification . For fin-grained classification we have Computer Science domain in English.

2 Related Work

We treat Automatic Technical Domain Identification as a text classification task where we assign predefined categories like chem for chemistry, cs for computer science for the given text. Text classification is a fundamental task in NLP applications and it is a crucial technology in many applications, such as web search, ads matching, and sentiment analysis. Many researchers found variety of algorithms to solve the text classification problem.

The algorithms will vary based on the langauge of text and domain of the text as well. McCallum et al. (1998) compared the theory and practice of two different first-order probabilistic classifiers, both of which make the naive Bayes assumption. The multinomial model is found to be almost uniformly better than the multi-variate Bernoulli model. Joachims (1999) introduced Transductive Support Vector Machine for text classification. While general Support Vector Machines (SVMs) try to produce a general decision function for a learning task, Transductive Support Vector Machines take a particular test set into account and try to minimize misclassification of just those particular samples. Nigam et al. (1999) used maximum entropy for text classification by computing the conditional distribution of the class given the text, and compared accuracy to naive Bayes and showed that maximum entropy is sometimes significantly better, but also sometimes worse. Lodhi et al. (2002) proposed a novel approach for categorizing text documents based on the use of a special kernel called string subsequence kernel. Machine learning for text classification is the foundation of document categorization, news filtering, document routing, and personalization.

In text domains, effective feature selection is crucial to make the learning task efficient and more accurate, based on this point Forman (2003) presented an extensive comparative study of twelve feature selection metrics like Document Frequency, etc for the high-dimensional domain of text classification, focusing on support vector machines and 2-class problems, typically with high class skew. In social media such as Twitter, Facebook the users may become overwhelmed by the raw data. One solution to this problem is the classification of short text, In Sriram et al. (2010) they did the same, they proposed an approach to use a small set of domain-specific features extracted from the author's profile and text to classify the text to a predefined set of generic classes such as News, Events, Opinions, Deals, and Private Messages. Apart from machine learning algorithms there are some deep learning techniques as well for text classification. In contrast to traditional methods,Lai et al. (2015) introduced a recurrent convolutional neural network for text classification without human designed features. In their model, they apply a recurrent structure to capture contextual information as far as possible when learning word representations, which may introduce considerably less noise compared to traditional window-based neural networks.

Conneau et al. (2016) presented a new architecture (VD-CNN) for text processing which operates directly at the character level and uses only small convolutions and pooling operations. Joulin et al. (2016) used fasttext for word features and then averaged to get a sentence representation for text classification. Yao et al. (2019)proposed a novel approach for text clasiification termed as Graph Convolutional Networks termed as Text-GCN, it can capture global co-occurence information and uses limited labelled texts/documents well. Though there exists a lot of work on text classification, very few works are done for technical domains and on Indian languages like ours. Therefore we present our approach on the provided Indian Languages along with technical domains.

3 Approach

We evaluate our two models namely, Stochastic Gradient Decent and Multi Layer Perceptron on technical domains(Chemistry,Communication Technology, Computer Science, Law , Math and Physics,Bio-Chemistry, Management) for coarse grained technical domain classification for all above mentioned languages(though the number of domains may differ from language to language). For fine grained technical domain classification we have only Computer science in which sub-domains include AI, Algorithm , Computer Architecture, Computer Networks , Database Management system , Programming and Software Engineering for English. We used TFIDF for all experiments.

3.1 Term Frequency & Inverse Document Frequency (TF-IDF)

We use TF-IDF as our feature extraction method in our experiments, The most basic form of weighted word feature extraction is Term frequency (Salton and Buckley, 1988) TF, where each word is mapped to a number corresponding to the number of occurrences of that word in the whole corpora. Methods that extend the results of TF generally use word frequency as a boolean or logarithmically scaled weighting.

$$W(d,t) = TF(d,t) * log(\frac{N}{df(t)}) \qquad (1)$$

(Jones, 1972) proposed Inverse Document Frequency (IDF) as a method to be used along with term frequency in order to lessen the effect of implicitly common words in the corpus. IDF assigns a higher weight to words with either high or low frequency term in the document. This combination of TF and IDF is well known as Term Frequency-Inverse document frequency (TF-IDF). The mathematical representation of the weight of a term in a document by TF-IDF is given in Equation 1. Here N is the number of documents and $df(t)$ is the number of documents containing the term t in the corpus. The first term in equation 1 improves the recall while the second term improves the precision.

3.2 Stochastic Gradient Decent (SGD)

We used SGD classifier from scikit-learn (Pedregosa et al., 2011). SGD has been successfully applied to large-scale and sparse machine learning problems often encountered in text classification and natural language processing. Though SGD is an optimizer it's alone can be used as a classifier for text classification using different loss functions. The class SGDClassifier implements a plain stochastic gradient descent learning routine which supports different loss functions and penalties for classification. We performed our experiments with hinge loss which is equivalent to linear Support Vector Machine (SVM).

3.3 Multi Layer Perceptron

For Multi Layer Perceptron classifier which in the name itself connects to a Neural Network. Unlike other classification algorithms such as Support Vectors or Naive Bayes Classifier, MLPClassifier relies on an underlying Neural Network to perform the task of text classification. MLPClassifier trains iteratively, at each time step the partial derivatives of the loss function with respect to the model parameters are computed to update the parameters. It can also have a regularization term added to the loss function that shrinks model parameters to prevent overfitting. In our experiments we used MLPClassifier from (Pedregosa et al., 2011).

4 Experiments & Results

We evaluate two machine learning algorithms on Data provided by ICON TechDOfication-2020 shared task. The data statistics in terms of number of sentences for all languages is mentioned in table 1. We are provided with various technical domains like physics chemistry etc by ICON TechDOfication 2020 shared task for mentioned languages, however the domains in each language are different. We have Physics(phy), Maths(math), Chemistry(che), Law(law) and Computer Science(cse) n English. Similarly Bengali and Gujarathi have BioChemistry(bioche), cse, Communication Technology(com-tech), Management(mgmt) and phy. Hindi and Telugu have bioche,cse , phy, mgmt, com-tech and other, where Hindi has extra math domain. Malayalam has cse, bioche, com-tech domains, and for Marathi we have bioche, com-tech, phy and cse. In fine-grained domain identification like identifying sub-domain of Computer Science, we have AI (ai),Algorithm (algo),Computer Architecture (ca), Computer Networks (cn), Database Management system (dbms),Programming (pro) and Software Engineering (se) subdomains.

As mentioned in section 3.1 we used TFIDF for all experiments in this paper. For SGD classifier we used hinge loss, and we took alpha as 0.00001, it is a constant that multiplies the regularization term. The higher the value, the stronger the regularization. Maximum number of iterations taken for this algorithm is 15. In MLPClassifier we used relu activation function, solver as sgd which used to find the gradients and optimize the loss function. We adopted the same alpha as SGD Classifier. As MLP is neural network based classifier, there is a need to give hidden layer sizes, we used [100,90] for two hidden layers apart from input and output layer.

Lang.	Train	Dev	Test
English	23962	4850	2500
Bengali	58500	5843	1923
Gujarati	36009	5724	2683
Hindi	148445	14338	4212
Malayalam	40669	3390	1515
Marathi	41997	3780	1789
Tamil	72483	6190	2071
Telugu	68865	5920	2612
English(CS)	13580	1360	1930

Table 1: Data Statistics (no. of sentences)
English(CS) is fine-grained classification task for Computer Science Domain in English

Lang.	Acc.	P	R	F1
English	0.76	0.76	0.76	0.76
Bengali	0.66	0.71	0.68	0.66
Gujarati	0.58	0.57	0.58	0.57
Hindi	0.43	0.44	0.41	0.40
Malayalam	0.44	0.47	0.4	0.37
Marathi	0.48	0.5	0.47	0.43
Tamil	0.44	0.43	0.45	0.36
Telugu	0.55	0.6	0.56	0.57
English(CS)	0.70	0.70	0.70	0.70

Table 2: Classification Report for SGD Classfier
English(CS) is fine-grained classification task for Computer Science Domain in English
Acc:Accuracy P:Precision R:Recall F1:F1-score

We present Accuracy, Precision, Recall and F1-score for all the tasks(for all mentioned languages) as shown in table 2 and table 3 for SGD classifier and for MLP classifier respectively. If we observe the results both the models performed well on English compared to other languages. Motivated from this our future work will be to improve the accuracy on Indian languages. MLP classifier outperformed SGD in almost all tasks. If we talk

Lang.	Acc.	P	R	F1
English	0.77	0.77	0.77	0.77
Bengali	0.66	0.70	0.68	0.66
Gujarati	0.60	59	0.6	0.58
Hindi	0.43	0.5	0.42	0.43
Malayalam	0.44	0.47	0.38	0.36
Marathi	0.5	0.5	0.48	0.44
Tamil	0.45	0.44	0.5	0.39
Telugu	0.54	0.6	0.51	0.52
English(CS)	0.62	0.64	0.62	0.63

Table 3: Classification Report for MLP Classfier English(CS) is fine-grained classification task for Computer Science Domain in English
Acc:Accuracy P:Precision R:Recall F1:F1-score

about fine grained technical domain identification, SGD outperformed MLP classifier. Comparatively Malayalam and Tamil got less scores in both the algorithms. From all the experiments we can conclude that we can use MLP classifier for Technical Domain Identification but still there is a huge need of improving or coming up with new algorithms for morphologically rich Indian languages.

5 Conclusion & Future Work

we are in the process of exploring many different algorithms for Technical Domain Identification. In the future we want to work on other possible languages for possible technical domains. In this paper we showed two machine learning algorithms(SGD and MLP). TFIDF doesn't depend on any language or domain specific resources hence, we preferred TFIDF as feature extraction method for both the ML algorithms presented in the experiments. From the results we can conclude that Multi Layer Perceptron is performing better on these technical domains for the provided languages.

References

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781*.

George Forman. 2003. An extensive empirical study of feature selection metrics for text classification. *Journal of machine learning research*, 3(Mar):1289–1305.

Thorsten Joachims. 1999. Transductive inference for text classification using support vector machines. In *Icml*, volume 99, pages 200–209.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*.

Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2016. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*.

Jasleen Kaur and Jatinderkumar R Saini. 2015. A study of text classification natural language processing algorithms for indian languages. *The VNSGU Journal of Science Technology*, 4(1):162–167.

Siwei Lai, Liheng Xu, Kang Liu, and Jun Zhao. 2015. Recurrent convolutional neural networks for text classification. In *Twenty-ninth AAAI conference on artificial intelligence*.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text classification using string kernels. *Journal of Machine Learning Research*, 2(Feb):419–444.

Andrew McCallum, Kamal Nigam, et al. 1998. A comparison of event models for naive bayes text classification. In *AAAI-98 workshop on learning for text categorization*, volume 752, pages 41–48. Citeseer.

Kamal Nigam, John Lafferty, and Andrew McCallum. 1999. Using maximum entropy for text classification. In *IJCAI-99 workshop on machine learning for information filtering*, volume 1, pages 61–67. Stockholom, Sweden.

F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523.

Bharath Sriram, Dave Fuhry, Engin Demir, Hakan Ferhatosmanoglu, and Murat Demirbas. 2010. Short text classification in twitter to improve information filtering. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval*, pages 841–842.

Liang Yao, Chengsheng Mao, and Yuan Luo. 2019. Graph convolutional networks for text classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7370–7377.

Fine-grained domain classification using Transformers

Akshat Gahoi **Akshat Chhajer** **Dipti Mishra Sharma**

Language Technologies Research Center
International Institute of Information Technology, Hyderabad, India

{akshat.gahoi,akshat.chhajer}@research.iiit.ac.in
dipti@iiit.ac.in

Abstract

The introduction of transformers in 2017 and successively BERT in 2018 brought about a revolution in the field of natural language processing. Such models are pretrained on vast amounts of data, and are easily extensible to be used for a wide variety of tasks through transfer learning. Continual work on transformer based architectures has led to a variety of new models with state of the art results. RoBERTa (Liu et al., 2019) is one such model, which brings about a series of changes to the BERT (Devlin et al., 2018) architecture and is capable of producing better quality embeddings at an expense of functionality. In this paper, we attempt to solve the well known text classification task of fine-grained domain classification using BERT and RoBERTa and perform a comparative analysis of the same. We also attempt to evaluate the impact of data preprocessing specially in the context of fine-grained domain classification.
The results obtained outperformed all the other models at the ICON TechDOfication 2020 (subtask-2a) Fine-grained domain classification task and ranked first. This proves the effectiveness of our approach.

1 Introduction

The transformer-based language models have been showing promising progress on a number of different natural language processing (NLP) benchmarks. The combination of transfer learning methods with large-scale transformer language models is becoming a standard in modern NLP and has resulted in many state-of-the-art models.
Compared to LSTMs(Greff et al., 2015), the main limitations of bidirectional LSTMs is their sequential nature, which makes training in parallel very difficult. The transformer architecture solves that by completely replacing LSTMs by the so-called attention mechanism (Vaswani et al., 2017). With attention, we are seeing an entire sequence as a whole, therefore it is much easier to train in parallel.
Text classification is a well-known task in Natural Language Processing, which aims at automatically providing additional document-level metadata (e.g. domain, genre, author).

One of text classification tasks: domain classification can be divided into two categories:-

- Course-grained domain classification
- Fine-grained domain classification

Course-grained domain classification aims to classify the input into varied and unrelated domains such as chemistry, law, computer science etc. On the other hand, fine-grained domain classification aims to classify the input into closely related subdomains under a higher level domain. Example includes classification of text on physics into different topics like relativity, mechaincs, or quantum mechanics. The latter is found to be a significantly more challenging task due to the similarity and lack of distinction between the inputs attributed to the fact that they are under an umbrella of a common domain.
Although there is substantive work done on domain classification in general (Young-Bum Kim, 2018), there has been less emphasis on fine-grained domain classification and the various augmentations to data that can be done to achieve higher performances in that task. This paper looks at the task of fine-grained domain classification in context of transformers. It paper will provide a comparison between the widely used **BERT** and **RoBERTa** embeddings for the task as well as attempt to observe the impact of data pre-processing in the context of fine-grained domain classification.
On blind test corpora of 1929 text samples, the proposed model in this paper led to F1 score of

0.824 at the ICON TechDOfication 2020 shared task (subtask-2a). This result helped us bag the leading position on the leaderboard.

2 Dataset

For the study, dataset from the ICON TechDOfication 2020 (subtask-2a) was used. The entire collection consisted of 14910 text samples from English spanning across 7 sub-domains of **Computer Science**. Table 1 shows the sub-domains and the distribution of data.

Sub-domain	Code	Samples
Artificial Intelligence	ai	2140
Algorithm	algo	2131
Computer Architecture	ca	2127
Computer Networks	cn	2140
Database Management System	dbms	2140
Programming	pro	2122
Software Engineering	se	2140

Table 1: Dataset distribution across domains

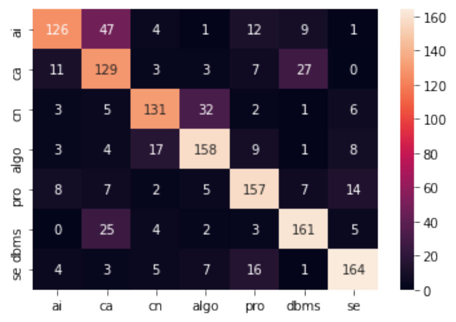
The average number of characters in the text samples was 177.6 and the average number of tokens observed in the same was 36.3.

A collection of 1929 text samples served as the blind test set for this task.

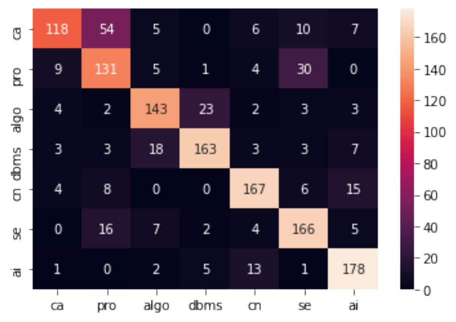
3 BERT vs RoBERTa

BERT is a bi-directional transformer for pre-training over huge amount of unlabeled textual data to learn a language representation. It can be then used to fine-tune for specific machine learning tasks like text classification. BERT outperformed the NLP state-of-the-art on several challenging tasks, attributed to the bidirectional transformer, novel pre-training tasks of Masked Language Model(Song et al., 2019) and Next Sentence Prediction(Shi and Demberg, 2019).

RoBERTa has a very similar architecture as compared to BERT with improved training methodology and more data. To improve the training, RoBERTa removes the Next Sentence Prediction task from BERT's pre-training and introduces dynamic masking so that the masked token changes during the training epochs. Originally BERT is trained for 1M steps with a batch size of 256 sequences. RoBERTa on the other hand is trained with 125 steps of 2K sequences and 31K steps with 8K sequences of batch size. Large batches are also easier to parallelize via distributed parallel training.



(a) BERT with no preprocessing



(b) RoBERTa with no preprocessing

Figure 1: Confusion matrices for dev set without preprocessing

4 System Overview

The section presents an overview of the system which was used to evaluate the scores described in the Results section of the paper.

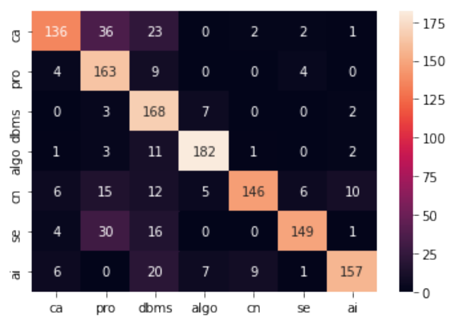
4.1 Pre-Processing

In the first approach, only one-hot-encoding for the labels was done and the raw text was fed as it is to both the transformers.

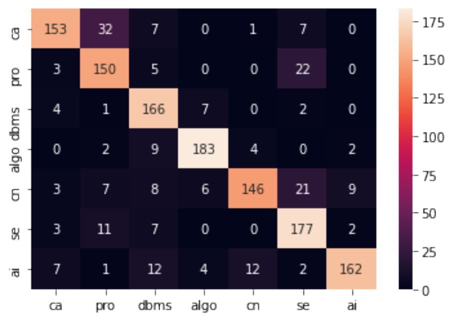
In the second approach the raw data was preprocessed keeping in mind the nature of fine-grained domain classification task. First, tokenization was done on the text using spaCy and the stop words were filtered out. Next, the tokens were passed through a counter and the top 20 tokens from the entire corpus were identified and then removed. As domain classification relies more on the keywords than the sentence structures, the data was cleaned. Lastly, the text was reconstructed from the remaining tokens. This was done to reduce the generalization amongst the sub-domains as the text had a lot of common terms from the higher level computer science domain itself.

4.2 Training

In total, 4 models were trained using BERT/RoBERTa and with/without pre-processing.



(a) BERT with preprocessing



(b) RoBERTa with preprocessing

Figure 2: Confusion matrices for dev set with preprocessing

The training was done using the concept of transfer learning. The pretrained bert-base-uncased and roberta-bert were taken and further fine tuning on it was done using the training dataset. The learning rate used was 4e-5 with 128 batch size. Each of the models were trained for 4 epochs.

5 Results and Evaluation

All the models were evaluated on the dev dataset and the results are presented in Table 2. It is clear that pre-processing indeed increases the f1 score and makes a substantive difference in fine-grained domain classification. This is because the common terms from the higher level common domains are removed and more distinction is created in the text samples for sub-domains.

In Figure 1 (results on dev set without preprocessing), we can see that RoBERTa miss classifies only slightly a less number text samples compared to BERT with both performing very similar. However, there is difference seen when preprocessing is done and frequent words are removed. In Figure 2 (results on dev set with preprocessing), we can see that BERT miss-classifies 91 text samples as '*Database Management System*' which reduces to 48 when using RoBERTa. Similarly, 87 miss-classifications done by BERT as '*Programming*' are corrected to 54 by RoBERTa. However it is seen that RoBERTa tends to miss-classify text samples as '*Software Engineering*' often.

In both the cases, RoBERTa performed better than BERT however, with a small margin. The best performing model (RoBERTa with preprocessing) was then evaluated using the ICON TechDOfication 2020 (subtask-2a) test dataset. The results obtained are shown in Table 3. It was observed that for different models also documents gets misidentified between a common pair of domains hence defining a close relation between the two domains. So this experiment can also be done to determine two closely related domains among a huge variety of domains.

Transformer	**Pre-processed**	**Precision**	**Recall**	**F1**
bert-base-uncased	no	0.761	0.752	0.756
roberta-base	no	0.788	0.783	0.781
bert-base-uncased	yes	0.832	0.812	0.810
roberta-base	yes	0.842	0.837	0.835

Table 2: Results for the dev set

Transformer	**Accuracy**	**Precision**	**Recall**	**F1**
RoBERTa	0.825	0.826	0.825	0.824

Table 3: Final model results on the test dataset

6 Conclusion

In this paper, we did a comparative analysis of BERT and RoBERTa in the context of fine-grained domain classification. Furthermore, the impact of pre-processing was also explored. It was found that pre-processing and removal of common terms from data helps the model perform better as more distinction is created between the sub-domains. The results indicate that RoBERTa performs slightly better than BERT in all the cases.

The model proposed in this paper ranked first in the ICON TechDOfication 2020 (subtask-2a) with an F1 score of 0.824.

7 Future Work

This paper shows how good transformers can perform for the task of multi class text classification. The main difference in the results comes from the embeddings being used. Thus, a very high performing multilingual model can be created if enough data and pre-trained language models are available for Indian languages. Hence, the goal can be to create BERT embeddings for other languages and extend the work done by AI4Bharat/IndicBERT (Kakwani et al., 2020). This can then not only be used for text classification tasks but also any other multi-lingual state-of-the-art natural processing task.

References

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, and Jürgen Schmidhuber. 2015. LSTM: A search space odyssey. *CoRR*, abs/1503.04069.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, Gokul N.C., Avik Bhattacharyya, Mitesh M. Khapra, and Pratyush Kumar. 2020. IndicNLPSuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for Indian languages. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 4948–4961, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Wei Shi and Vera Demberg. 2019. Next sentence prediction helps implicit discourse relation classification within and across domains. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5790–5796, Hong Kong, China. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2019. MASS: masked sequence to sequence pre-training for language generation. *CoRR*, abs/1905.02450.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *CoRR*, abs/1706.03762.

Anjishnu Kumar Ruhi Sarikaya Young-Bum Kim, Dongchan Kim. 2018. Efficient large-scale neural domain classification with personalized attention.

TechTexC: Classification of Technical Texts using Convolution and Bidirectional Long Short Term Memory Network

Omar Sharif[†], **Eftekhar Hossain***, **and Mohammed Moshiul Hoque**[†]
[†]Department of Computer Science and Engineering
*Department of Electronics and Telecommunication Engineering
Chittagong University of Engineering and Technology, Bangladesh
[†]{omar.sharif, moshiul_240}@cuet.ac.bd
[*]eftekhar.hossain@cuet.ac.bd

Abstract

This paper illustrates the details description of technical text classification system and its results that developed as a part of participation in the shared task TechDofication 2020. The shared task consists of two sub-tasks: (i) first task identify the coarse-grained technical domain of given text in a specified language and (ii) the second task classify a text of computer science domain into fine-grained sub-domains. A classification system (called 'TechTexC') is developed to perform the classification task using three techniques: convolution neural network (CNN), bidirectional long short term memory (BiLSTM) network, and combined CNN with BiLSTM. Results show that CNN with BiLSTM model outperforms the other techniques concerning task-1 of sub-tasks (a, b, c and g) and task-2a. This combined model obtained f_1 scores of 82.63 (sub-task a), 81.95 (sub-task b), 82.39 (sub-task c), 84.37 (sub-task g), and 67.44 (task-2a) on the development dataset. Moreover, in the case of test set, the combined CNN with BiLSTM approach achieved that higher accuracy for the subtasks 1a (70.76%), 1b (79.97%), 1c (65.45%), 1g (49.23%) and 2a (70.14%).

1 Introduction

Due to the substantial growth and effortless access to the Internet in recent years, an enormous amount of unstructured textual contents have generated. It is a crucial task to organize or structure such a voluminous unstructured text in manually. Thus, automatic classification can be useful to manipulate a huge amount of texts, and extract meaningful insights which save a lot of time and money. Text categorization is a classical NLP problem which aims to categorize texts into organized groups. It has a wide range of applications like machine translation, question answering, summarization, and sentiment analysis. There are several approaches available to classify texts according to their labels. However, deep learning method outperforms the rule-based and machine learning-based models because of their ability to capture sequential and semantic information from texts (Minaee et al., 2020). We propose a classifier using CNN (Jacovi et al., 2018), and BiLSTM (Zhou et al., 2016) to classify technical texts in the computer science domain. Furthermore, by sequentially adding these networks, remarkable accuracy in several shared classification tasks can be obtained. The rest of the paper is organized as follows: related work given in section 2. Section 3 describes the dataset. The framework described in section 4. The findings presented in section 5.

2 Related Work

CNN and LSTM have achieved great success in various NLP tasks such as sentence classification, document categorization, sentiment analysis, and summarization. Kim (2014) used convolution neural network to classify sentences. A method used contents and citations to classify scientific document (Cao and Gao). Zhou et al. (2016) used 2-D max pooling and bidirectional LSTM to classify texts. Zhou et al. (2015) combined CNN and LSTM to classify sentiment and question type. Their system achieved superior accuracy than CNN and LSTM individually. Hossain et al. (2020) used LSTM to classify sentiment of Bengali text documents. Their system got maximum accuracy with one layer of LSTM followed by three dense layers. Ranjan et al. (2017) proposed a document classification framework using LSTM and feature selection algorithms. Ameur et al. (2020) combined CNN and RNN methods to categorize Arabic texts. They used dynamic, fine-tuned words embedding to get effective result on open-source Arabic dataset.

3 Dataset

To develop the classifier model, we used the dataset provided by the organizers of the shared task[1]. This shared task consists of two subtasks: subtask-1 and subtask-2. Subtask-1 aims to the identification of coarse-grained domain for a piece of text. Organizers provided data including eight different languages (English, Bangla, Hindi, Gujarati, Malayalam, Marathi, Tamil, Telugu) each having a different number of classes for this task. In subtask-2, the goal is to find the fine-grained sub domain of a text from the computer science domains. Seven classes such as artificial intelligence, algorithm, computer architecture, computer networks, database management systems, programming, software engineering are available in this subtask-2. The number of training, validation and test texts for each of the task is different. Summary of the dataset presents in table 1.

Task	No. of classes	Train	Dev	Test
task-1a	5	23962	4850	2500
task-1b	5	58500	5842	1923
task-1c	5	36009	5724	2682
task-1d	7	148445	14338	4211
task-1e	3	40669	3390	1514
task-1f	4	41997	3780	1788
task-1g	6	72483	6190	2070
task-1h	6	68865	5920	2611
task-2a	7	13580	1360	1929

Table 1: Dataset description

4 System Overview

Figure 1 shows the schematic diagram of the proposed system. The system has four major parts: preprocessing, feature extraction, classifier model and prediction. After processing the raw texts, Word2Vec word embedding technique is applied on the processed texts to extract features. After exploiting inherent features of the texts, the model trained with CNN, BiLSTM and combination of CNN & BiLSTM.. Finally, the trained model will use to predict the class on the development set.

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

4.1 Preprocessing

In this step, all the punctuation's (,.;:"!) and flawed characters (#,$, %,*,@) removed from the input texts. Texts are having a length of fewer than two words also discarded. Deep learning algorithms could not possibly learn from the raw texts. Thus, a numeric mapping of the input texts is created. A vocabulary of K unique words is developed and each input text encoded into numeric sequences based on word index in vocabulary. By applying the pad sequence method, each sequence converted into fixed-length vector. We choose optimal sequence length 100 as most of the length of the text ranges between 30-70 words. In order to maintain a fixed length of inputs, zero paddings are used with the short text, and extra values discarded from the long sequences.

4.2 Feature Extraction

To extract features from texts and capture semantic property of a word Word2Vec (Mikolov et al., 2013) embedding technique is used. Embedding maps textual data into a dense vector by solving the sparsity problem. We use the default embedding layer of Keras to produce embedding matrix. Embedding layer has three parameters: vocabulary size, embedding dimension and length of texts. Embedding dimension determine the size of the dense word vector. The entire corpus is fitted into the embedding layer for a specific subtask and choose 100 as embedding dimension for all the subtasks. Features extracted from the embedding layer propagated the rest of the network.

4.3 Classifier Model

In this work, CNN and BiLSTM are used for initial model building. However, after combining these methods, we get superior results in several subtasks (Zhou et al., 2015). A description of the proposed architecture illustrates in the subsequent paragraphs.

CNN: In CNN, convolution filters capture the inherent syntactic and semantic features of the texts. The proposed classifier considers two layers, one dimensional CNN. In each layer, there are 128 filters with kernel size 5. To downsample the features on CNN max-pooling technique is utilized where pool size is 1×5. We have used a non-linear activation function 'relu' with CNN.

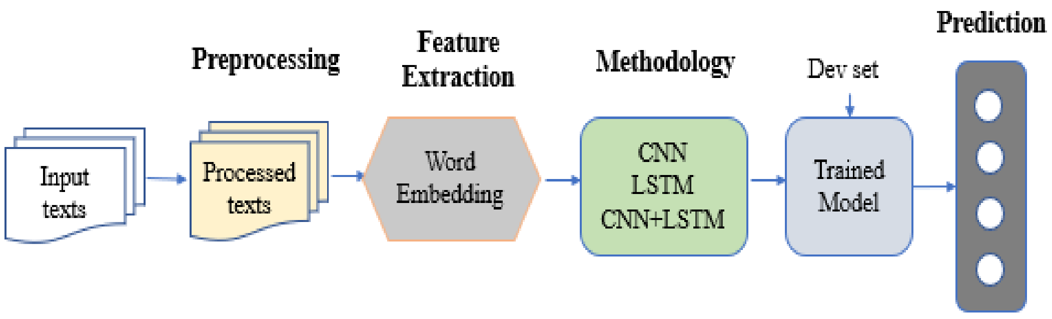


Figure 1: Schematic diagram of our system

BiLSTM: We use Bidirectional LSTM network to capture the sequential features from the input text and to avoid vanishing/exploding gradient problems of simple RNN. We use two layers of BiLSTM on top of each other, where each layer has 128 LSTM cells. In order to reduce the overfitting on training data, the dropout technique is used with a dropout rate of 0.2. After achieving the hidden representation form, the LSTM layer output passed to the softmax layer for classification.

CNN+BiLSTM: In this approach, we merge CNN and BiLSTM models with marginal modification in network architecture. Previously, we used two layers of CNN and BiLSTM, whereas in this technique, discard one layer from each network and combine them sequentially. Word embedding features is feed to the CNN, which has 128 filters. After max pooling with a window of size 5, features of CNN propagated to the LSTM layer. It has 128 bidirectional cells to capture the sequential information. In order to mitigate overfitting, a dropout layer is added with a dropout rate of 0.2. Finally, the softmax layer gets input from the LSTM and perform classification.

4.4 Prediction

The goal of the prediction module is to determine the technical domain of an input text that it has never seen before. For the prediction, sample instances are processed and converted into numerical sequences by the tokenizer. Trained model use this sequence to predict the associated class of the input text.

5 Experiments

Google co-laboratory platform is used to conduct experiments. Deep learning model developed with Keras=2.4.0 framework with tensorflow=2.3.0 in the backend. For data preparation and evaluation, we use python=3.6.9 and secikit-learn=0.22.2.

5.1 Hyperparameter Settings

Performance of deep learning models heavily depends on the hyperparameters used in training. To choose the optimal hyperparameters for the proposed model, we played with different combinations. We choose parameter values based on its effect on the output. Table 2 exhibits the values of different hyperparameters considered to train the proposed model. Adam optimizer is used with a learning rate of 0.001. The model trained with a batch size of 128 until a training accuracy of 98% reached. We use Keras callbacks to save the intermediate model during training with best validation accuracy. The trained model used to predict on the instances of development set.

Hyperparameters	**Optimum value**
Embedding dimension	100
Padding length	100
Filters	128
Kernel size	5
Pooling type	max
Window size	5
LSTM cell	128
Dropout rate	0.2
Optimizer	‘adam’
Learning rate	0.001
Batch size	128

Table 2: Hyperparameter Settings

5.2 Results

We determine the superiority of the models based on their weighted f_1 score on the development set of different tasks. Table 3 shows the evaluation results of CNN, BiLSTM and CNN+BiLSTM.

Task	CNN			Bi-LSTM			CNN+Bi-LSTM		
	P	R	F	P	R	F	P	R	F
task-1a (English)	81.48	81.36	81.4	82.81	82.52	82.52	82.9	82.54	**82.63**
task-1b (Bangla)	81.96	81.94	81.91	81.49	81.38	81.39	82.04	81.97	**81.95**
task-1c (Gujarati)	82.63	82.39	82.38	82.79	82.05	82.06	82.58	82.41	**82.39**
task-1d (Hindi)	79.46	79.0	79.07	79.86	79.54	**79.52**	79.65	79.44	79.48
task-1e (Malayalam)	91.51	91.53	91.52	91.87	91.89	**91.86**	91.38	91.36	91.32
task-1f (Marathi)	85.93	85.93	85.84	86.47	86.53	**86.48**	86.57	86.51	86.38
task-1g (Tamil)	84.26	84.07	84.02	84.36	84.34	84.3	84.56	84.63	**84.37**
task-1h (Telegu)	86.85	86.82	86.78	87.64	87.34	**87.41**	87.17	87.14	87.13
task-2a (English)	64.36	63.82	63.83	66.45	65.51	65.72	67.86	67.35	**67.44**

Table 3: Evaluation results of three models on different tasks where P, R, F denotes precision, recall and weighted f_1 score.

Task	Method	A	P	R	F
task-1a (English)	CNN+BiLSTM	70.76	71.50	70.76	70.63
task-1b (Bangla)	CNN+BiLSTM	79.97	81.50	82.41	80.25
task-1c (Gujarati)	CNN+BiLSTM	65.45	1.95	1.81	1.86
task-1d (Hindi)	BiLSTM	57.28	57.13	55.99	54.57
task-1e (Malayalam)	BiLSTM	31.37	0.32	0.18	0.19
task-1f (Marathi)	BiLSTM	63.09	65.98	61.38	59.81
task-1g (Tamil)	CNN+BiLSTM	49.23	48.38	61.34	43.70
task-1h (Telegu)	BiLSTM	52.82	0.76	0.64	0.68
task-2a (English)	CNN+BiLSTM	70.14	71.51	70.19	70.40

Table 4: Evaluation results on the test set. Here A, P, R, F denotes accuracy, precision, recall and weighted f_1 score respectively.

The results revealed that BiLSTM model achieved the higher f_1 score of $79.52\%, 91.86\%, 86.48\%$ and 87.41% for tasks 1d, 1e, 1f and 1h. It outperforms CNN model for all tasks. The reason behind the superior results of LSTM because of its capability to capture long-range dependencies. However, combined CNN and BiLSTM provide interesting insights. It outdoes previous BiLSTM model in tasks 1a, 1b, 1c, 1g and 2a by obtaining $82.63\%, 81.95\%, 82.39\%, 84.37\%$ and $67.4\%4$ f_1 scores. The model achieved 2% rise in f_1 score concerning task-2a where the fine-grained domain of a text is identified. In all the cases, there exists a small difference ($< 0.5\%$) between the result of BiLSTM and CNN+BiLSTM. By analyzing the results, it observed that for a task with less number of classes, all models achieved quite similar performance. However, when the number of classes increased, the BiLSTM and CNN+BiLSTM models performed better than CNN. It is because the CNN model could not capture sequential feature as well compare to LSTM.

Table 4 shows the output of the best run on the test set for each tasks. Based on the performance of the development set, methods are selected to predict on the test set. Therefore, we use CNN+BiLSTM model to predict on the tasks 1a, 1b, 1c, 1g and 2a. Model achieved $70.63\%, 80.25\%, 1.86\%, 43.70\%$ and 70.4% weighted f_1 scores on these tasks respectively. Unlike other tasks, precision, recall, and f_1 score are much lower for task 1c compare to the validation results. This lower score might happen due to some mistake during evaluation. Task 2a get better f_1 score on the test set to compare to the development set. For other cases, the performance of the methods degraded on the development set.

BiLSTM method used to get the outputs for tasks 1d, 1e, 1f and 1h. Model obtained 57.13%, 0.32%, 65.98% and 0.76% weighted f_1 scores on these tasks. It suspected that some errors might occur during evaluation for tasks 1e and 1h. The model achieved 91.86% and 87.41% f_1 scores on these tasks for the validation set but got an implausible result on the test set. This error might occur due to

Unicode issues of different languages. Our system also encountered an error when data read from the text file. The performance of BiLSTM method decreased in the test set than the validation set for all tasks.

Precision, recall and f_1 score have fallen for each task in the test set except task 2a. Weighted f_1 score has increased by 2.5% in the test set. For all the tasks, we observed a substantial variation between the development set and test set results. There might be two possible reasons behind this unpredictable nature of the models. First one, model is overfitted on the training set. Thus, it gets better results on training and validation set but poor results on the test set. The second one, test data are more diverse than training data. Suppose significant overlap does not exist between the train and test features. In that case, the model indeed performs poor on the test data since the models learn from the characteristics of training data.

6 Conclusion

This paper presents a detail description of the proposed system and its evaluation for the technical texts classification in different languages. As the baseline method, we used CNN and BiLSTM, and compare these methods with the proposed model (combined CNN and BiLSTM). Each model is trained, tuned and evaluated separately for subtasks 1 and 2. The proposed method showed better performance in terms of accuracy for subtasks (a, b, c, g) of task 1 and task 2a on development set. However, in the case of test set, the system performed better for the subtasks 1a, 1b, 1c, 1g and 2a. More dataset can be included for improved performance. In future, the attention mechanism may be explored to observe its effects on text classification tasks.

References

Mohamed Seghir Hadj Ameur, Riadh Belkebir, and Ahmed Guessoum. 2020. Robust arabic text categorization by combining convolutional and recurrent neural networks. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 19(5):1–16.

Minh Duc Cao and Xiaoying Gao. Combining contents and citations for scientific document classification. In *Int. Conf. on Advances in Artificial Intelligence*.

Eftekhar Hossain, Omar Sharif, Mohammed Moshiul Hoque, and Iqbal H Sarker. 2020. Sentilstm: A deep learning approach for sentiment analysis of restaurant reviews. *arXiv preprint arXiv:2011.09684*.

Alon Jacovi, Oren Sar Shalom, and Yoav Goldberg. 2018. Understanding convolutional neural networks for text classification. *arXiv preprint arXiv:1809.08037*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Shervin Minaee, Nal Kalchbrenner, Erik Cambria, Narjes Nikzad, Meysam Chenaghlu, and Jianfeng Gao. 2020. Deep learning based text classification: A comprehensive review. *arXiv preprint arXiv:2004.03705*.

Mr Nihar M Ranjan, YR Ghorpade, GR Kanthale, AR Ghorpade, and AS Dubey. 2017. Document classification using lstm neural network. *Journal of Data Mining and Management*, 2(2):1–9.

Chunting Zhou, Chonglin Sun, Zhiyuan Liu, and Francis Lau. 2015. A c-lstm neural network for text classification. *arXiv preprint arXiv:1511.08630*.

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*.

An Attention Ensemble Approach for Efficient Text Classification of Indian Languages

Atharva Kulkarni[1], **Amey Hengle**[1], and **Rutuja Udyawar**[2]

[1]Department of Computer Engineering, PVG's COET, Savitribai Phule Pune University, India.
[2]Optimum Data Analytics, India.
[1]{atharva.j.kulkarni1998, ameyhengle22}@gmail.com
[2]rutuja.udyawar@odaml.com

Abstract

The recent surge of complex attention-based deep learning architectures has led to extraordinary results in various downstream NLP tasks in the English language. However, such research for resource-constrained and morphologically rich Indian vernacular languages has been relatively limited. This paper proffers team SPPU_AKAH's solution for the TechDOfication 2020 subtask-1f: which focuses on the coarse-grained technical domain identification of short text documents in Marathi, a Devanagari script-based Indian language. Availing the large dataset at hand, a hybrid CNN-BiLSTM attention ensemble model is proposed that competently combines the intermediate sentence representations generated by the convolutional neural network and the bidirectional long short-term memory, leading to efficient text classification. Experimental results show that the proposed model outperforms various baseline machine learning and deep learning models in the given task, giving the best validation accuracy of 89.57% and f1-score of 0.8875. Furthermore, the solution resulted in the best system submission for this subtask, giving a test accuracy of 64.26% and f1-score of 0.6157, transcending the performances of other teams as well as the baseline system given by the organizers of the shared task.

1 Introduction

The advent of attention-based neural networks and the availability of large labelled datasets has resulted in great success and state-of-the-art performance for English text classification (Yang et al., 2016; Zhou et al., 2016; Wang et al., 2016; Gao et al., 2018). Such results, however, for Indian language text classification tasks are far and few as most of the research employ traditional machine learning and deep learning models (Joshi et al., 2019; Tummalapalli et al., 2018; Bolaj and Govilkar, 2016a,b; Dhar et al., 2018). Apart from being heavily consumed in the print format, the growth in the Indian languages internet user base is monumental, scaling from 234 million in 2016 to 536 million by 2021 [1]. Even so, just like most other Indian languages, the progress in NLP for Marathi has been relatively constrained, due to factors such as the unavailability of large-scale training resources, structural un-similarity with the English language, and a profusion of morphological variations, thus, making the generalization of deep learning architectures to languages like Marathi difficult.

This work posits a solution for the TechDOfication 2020 subtask-1f: coarse-grained domain classification for short Marathi texts. The task provides a large corpus of Marathi text documents spanning across four domains: Biochemistry, Communication Technology, Computer Science, and Physics. Efficient domain identification can potentially impact, and improve research in downstream NLP applications such as question answering, transliteration, machine translation, and text summarization, to name a few. Inspired by the works of (Er et al., 2016; Guo et al., 2018; Zheng and Zheng, 2019), a hybrid CNN-BiLSTM attention ensemble model is proposed in this work. In recent years, Convolutional Neural Networks (Kim, 2014; Conneau et al., 2016; Zhang et al., 2015; Liu et al., 2020; Le et al., 2017) and Recurrent Neural Networks (Liu et al., 2016; Sundermeyer et al., 2015) have been used quite frequently for text classification tasks. Quite different from one another, the CNNs and the RNNs show different capabilities to generate intermediate text representation. CNN models an input sentence by utilizing convolutional filters to identify the most influential n-grams of differ-

[1]https://home.kpmg/in/en/home/insights/2017/04/indian-language-internet-users.html

ent semantic aspects (Conneau et al., 2016). RNN can handle variable-length input sentences and is particularly well suited for modeling sequential data, learning important temporal features and long-term dependencies for robust text representation (Hochreiter and Schmidhuber, 1997). However, whilst CNN can only capture local patterns and fails to incorporate the long-term dependencies and the sequential features, RNN cannot distinguish between keywords that contribute more context to the classification task from the normal stopwords. Thus, the proposed model hypothesizes a potent way to subsume the advantages of both the CNN and the RNN using the attention mechanism. The model employs a parallel structure where both the CNN and the BiLSTM model the input sentences independently. The intermediate representations, thus generated, are combined using the attention mechanism. Therefore, the generated vector has useful temporal features from the sequences generated by the RNN according to the context generated by the CNN. Results attest that the proposed model outperforms various baseline machine learning and deep learning models in the given task, giving the best validation accuracy and f1-score.

2 Related Work

Since the past decade, the research in NLP has shifted from a traditional statistical standpoint to complex neural network architectures. The CNN and RNN based architectures have emerged greatly successful for the text classification task. Yoon Kim was the first one who applied a CNN model for English text classification. In this work, a series of experiments were conducted with single as well as multi-channel convolutional neural networks, built on top of randomly generated, pretrained, and fine-tuned word vectors (Kim, 2014). This success of CNN for text classification led to the emergence of more complex CNN models (Conneau et al., 2016) as well as CNN models with character level inputs (Zhang et al., 2015). RNNs are capable of generating effective text representation by learning temporal features and long-term dependencies between the words (Hochreiter and Schmidhuber, 1997; Graves and Schmidhuber, 2005). However, these methods treat each word in the sentences equally and thus cannot distinguish between the keywords that contribute more to the classification and the common words. Hybrid models proposed by (Xiao and Cho, 2016) and (Hassan and Mahmood, 2017) succeed in exploiting the advantages of both CNN and RNN, by using them in combination for text classification.

Since the introduction of the attention mechanism (Vaswani et al., 2017), it has become an effective strategy for dynamically learning the contribution of different features to specific tasks. Needless to say, the attention mechanism has expeditiously found its way into NLP literature, with many works effectively leveraging it to improve the text classification task. (Guo et al., 2018) proposed a CNN - RNN attention-based neural network (CRAN) for text classification. This work illustrates the effectiveness of using the CNN layer as a context of the attention model. Results show that using this mechanism enables the proposed model to pick the important words from the sequences generated by the RNN layer, thus helping it to outperform many baselines as well as hybrid attention-based models in the text classification task. (Er et al., 2016) proposed an attention pooling strategy, which focuses on making a model learn better sentence representations for improved text classification. Authors use the intermediate sentence representations produced by a BiLSTM layer in reference with the local representations produced by a CNN layer to obtain the attention weights. Experimental results demonstrate that the proposed model outperforms state-of-the-art approaches on a number of benchmark datasets for text classification. (Zheng and Zheng, 2019) combine the BiLSTM and CNN with the attention mechanism for fine-grained text classification tasks. The authors employ a method in which intermediate sentence representations generated by BiLSTM are passed to a CNN layer which is then max pooled to capture the local features of a sentence. The local feature representations are further combined by using an attention layer to calculate the attention weights. In this way, the attention layer can assign different weights to features according to their importance to the text classification task.

The literature in NLP focusing on the resource-constrained Indian languages has been fairly restrained. (Tummalapalli et al., 2018) evaluated the performance of vanilla CNN, LSTM, and multi-Input CNN for the text-classification of Hindi and Telugu texts. The results indicate that CNN based models perform surprisingly better as compared to LSTM and SVM using n-gram features. (Joshi et al., 2019) have compared different deep learn-

Label	Training Data	Validation Data
bioche	5,002	420
com_tech	17,995	1,505
cse	9,344	885
phy	9,656	970
Total	41,997	3,780

Table 1: Data distribution.

ing approaches for Hindi sentence classification. The authors have evaluated the effect of using pre-trained fasttext Hindi embeddings on the sentence classification task. The finest classification performance is achieved by the Vanilla CNN model when initialized with fasttext word embeddings fine-tuned on the specific dataset.

3 Dataset

The TechDOfication-2020 subtask-1f dataset consists of labelled Marathi text documents, each belonging to one of the four classes, namely: Biochemistry (bioche), Communication Technology (com_tech), Computer Science (cse), and Physics (phy). The training data has a mean length of 26.89 words with a standard deviation of 25.12.

Table 1 provides an overview of the distribution of the corpus across the four labels for training and validation data. From the table, it is evident that the dataset is imbalanced, with the class Communication Technology and Biochemistry having the most and the least documents, respectively. It is, therefore, reasonable to postulate that this data imbalance may lead to the overfitting of the model on some classes. This is further articulated in the Results section.

4 Proposed Model

This section describes the proposed multi-input attention-based parallel CNN-BiLSTM. Figure 1 depicts the model architecture. Each component is explained in detail as follows:

4.1 Word Embedding Layer

The proposed model uses fasttext word embeddings trained on the unsupervised skip-gram model to map the words from the corpus vocabulary to a corresponding dense vector. Fasttext embeddings are preferred over the word2vec (Mikolov et al., 2013) or glove variants (Pennington et al., 2014), as fasttext represents each word as a sequence

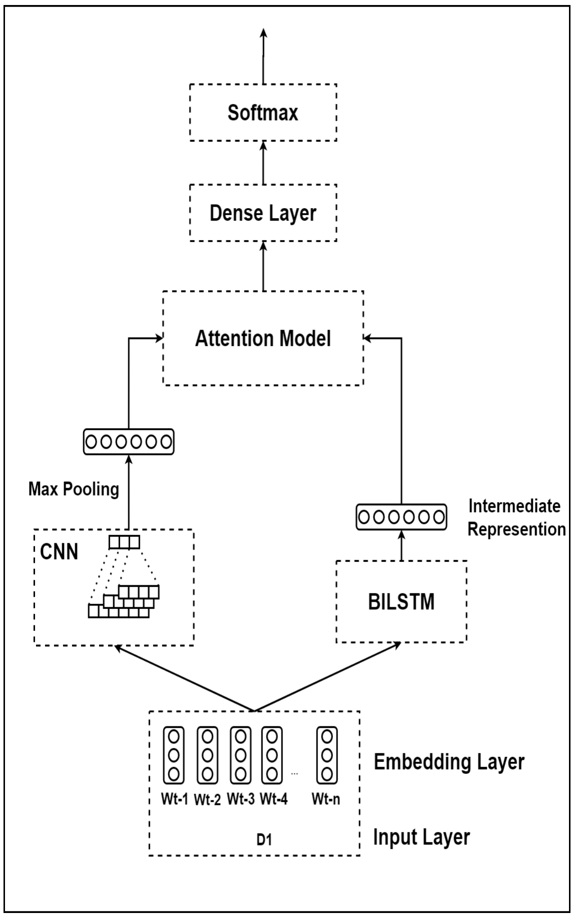


Figure 1: Model Architecture.

of character-n-grams, which in turn helps to capture the morphological richness of languages like Marathi. The embedding layer converts each word w_i in the document $T = \{w_1, w_2, ..., w_n\}$ of n words, into a real-valued dense vector e_i using the following matrix-vector product:

$$e_i = Wv_i \tag{1}$$

where $W \in \mathbb{R}^{dx|V|}$ is the embedding matrix, $|V|$ is a fixed-sized vocabulary of the corpus and d is the word embedding dimension. v_i is the one-hot encoded vector with the element e_i set to 1 while the other elements set to 0. Thus, the document can be represented as real-valued vector $e = \{e_1, e_2, ..., e_n\}$.

4.2 Bi-LSTM Layer

The word embeddings generated by the embeddings layer are fed to the BiLSTM unit step by step. A Bidirectional Long-short term memory (Bi-LSTM) (Graves and Schmidhuber, 2005) layer is just a combination of two LSTMs (Hochreiter and Schmidhuber, 1997) running in opposite directions.

This allows the networks to have both forward and backward information about the sequence at every time step, resulting in better understanding and preservation of the context. It is also able to counter the problem of vanishing gradients to a certain extent by utilizing the input, the output, and the forget gates. The intermediate sentence representation generated by Bi-LSTM is denoted as h.

4.3 CNN Layer

The discrete convolutions performed by the CNN layer on the input word embeddings, help to extract the most influential n-grams in the sentence. Three parallel convolutional layers with three different window sizes are used so that the model can learn multiple types of embeddings of local regions, and complement one another. Finally, the sentence representations of all the different convolutions are concatenated and max-pooled to get the most dominant features. The output is denoted as c.

4.4 Attention Layer

The linchpin of the model is the attention block that effectively combines the intermediate sentence feature representation generated by BiLSTM with the local feature representation generated by CNN. At each time step t, taking the output h_t of the BiLSTM, and c_t of the CNN, the attention weights α_t are calculated as:

$$u_t = tanh(W_1 h_t + W_2 c_t + b) \tag{2}$$

$$\alpha_t = Softmax(u_t) \tag{3}$$

Where W_1 and W_2 are the attention weights, and b is the attention bias learned via backpropagation. The final sentence representation s is calculated as the weighted arithmetic mean based on the weights $\alpha = \{\alpha_1, \alpha_2, ..., \alpha_n\}$, and output of the BiLSTM $h = \{h_1, h_2, ..., h_n\}$. It is given as:

$$s = \sum_{t=1}^{n} \alpha_t * h_t \tag{4}$$

Thus, the model is able to retain the merits of both the BiLSTM and the CNN, leading to a more robust sentence representation. This representation is then fed to a fully connected layer for dimensionality reduction.

4.5 Classification Layer

The output of the fully connected attention layer is passed to a dense layer with softmax activation to predict a discrete label out of the four labels in the given task.

5 Experimental Setup

Each text document is tokenized and padded to a maximum length of 100. Longer documents are truncated. The work of (Kakwani et al., 2020) is referred for selecting the optimal set of hyperparameters for training the fasttext skip-gram model. The 300-dimensional fasttext word embeddings are trained on the given corpus for 50 epochs, with a minimum token count of 1, and 10 negative examples, sampled for each instance. The rest of the hyperparameter values were chosen as default (Bojanowski et al., 2017). After training, an average loss of 0.6338. was obtained over 50 epochs. The validation dataset is used to tune the hyperparameters. The LSTM layer dimension is set to 128 neurons with a dropout rate of 0.3. Thus, the BiLSTM gives an intermediate representation of 256 dimensions. For the CNN block, we employ three parallel convolutional layers with filter sizes 3, 4, and 5, each having 256 feature maps. A dropout rate of 0.3 is applied to each layer. The local representations, thus, generated by the parallel CNNs are then concatenated and max-pooled. All other parameters in the model are initialized randomly. The model is trained end-to-end for 15 epochs, with the Adam optimizer (Kingma and Ba, 2014), sparse categorical cross-entropy loss, a learning rate of 0.001, and a minibatch size of 128. The best model is stored and the learning rate is reduced by a factor of 0.1 if validation loss does not decline after two successive epochs.

6 Baseline Models

The performance of the proposed model is compared with a host of machine learning and deep learning models and the results are reported in table 3. They are as follows:

Feature Based models: Multinomial Naive Bayes with bag-of-words input (MNB + BoW), Multinomial Naive Bayes with tf-idf input (MNB + TF-IDF), Linear SVC with bag-of-words input (LSVC + BoW), and Linear SVC with tf-idf input (LSVC + TF-IDF).

Basic Neural Networks: Feed forward Neural network with max-pooling (FFNN), CNN with max-pooling (CNN), and BiLSTM with maxpooling (BiLSTM)

Complex Neural Networks: BiLSTM +attention (Zhou et al., 2016) , serial BiLSTM-CNN

Metrics	bioche	com_tech	cse	phy
Precision	0.9128	0.8831	0.9145	0.8931
Recall	0.7976	0.9342	0.8949	0.8793
F1-Score	0.8513	0.9079	0.9046	0.8862

Table 2: Detailed performance of the proposed model on the validation data.

(Chen et al., 2017), and serial BiLSTM-CNN + attention.

7 Results and Discussion

The performance of all the models is listed in Table 3. The proposed model outperforms all other models in validation accuracy and weighted f1-score. It achieves better results than standalone CNN and BiLSTM, thus reasserting the importance of combining both the structures. The BiLSTM with attention model is similar to the proposed model, but the context is ignored. As the proposed model outperforms the BiLSTM with attention model, it proves the effectiveness of the CNN layer for providing context. Stacking a convolutional layer over a BiLSTM unit results in lower performance than the standalone BiLSTM. It can be thus inferred that combining CNN and BiLSTM in a parallel way is much more effective than just serially stacking. Thus, the attention mechanism proposed is able to successfully unify the CNN and the BiLSTM, providing meaningful context to the temporal representation generated by BiLSTM. Table 2 reports the detailed performance of the proposed model for the validation data. The precision and recall for communication technology (com_tech), computer science (cse), and physics(phy) labels are quite consistent. Biochemistry (bioche) label suffers from a high difference in precision and recall. This can be traced back to the fact that less amount of training data is available for the label, leading to the model overfitting on it.

8 Conclusion and Future work

While NLP research in English is achieving new heights, the progress in low resource languages is still in its nascent stage. The TechDOfication task paves the way for research in this field through the task of technical domain identification for texts in Indian languages. This paper proposes a CNN-BiLSTM based attention ensemble model for the subtask-1f of Marathi text classification. The parallel CNN-BiLSTM attention-based model unifies the intermediate representations generated by both the models successfully using the attention mechanism. It provides a way for further research in adapting attention-based models for low resource and morphologically rich languages. The performance of the model can be enhanced by giving additional inputs such as character n-grams and document-topic distribution. More efficient attention mechanisms can be applied to further consolidate the amalgamation of CNN and RNN.

Label	Validation Accuracy	Validation F1-Score
MNB + Bow	86.74	0.8352
MNB + TF-IDF	77.16	0.8010
Linear SVC + Bow	85.76	0.8432
Linear SVC + TF-IDF	88.17	0.8681
FFNN	76.11	0.7454
CNN	86.67	0.8532
BiLSTM	89.31	0.8842
BiLSTM + Attention	88.14	0.8697
Serial BiLSTM-CNN	88.99	0.8807
Serial BiLSTM-CNN + Attention	88.23	0.8727
Ensemble CNN-BiLSTM + Attention	**89.57**	**0.8875**

Table 3: Performance comparison of different models on the validation data.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Pooja Bolaj and Sharvari Govilkar. 2016a. A survey on text categorization techniques for indian regional languages. *International Journal of computer science and Information Technologies*, 7(2):480–483.

Pooja Bolaj and Sharvari Govilkar. 2016b. Text classification for marathi documents using supervised learning methods. *International Journal of Computer Applications*, 155(8):6–10.

Tao Chen, Ruifeng Xu, Yulan He, and Xuan Wang. 2017. Improving sentiment analysis via sentence type classification using bilstm-crf and cnn. *Expert Systems with Applications*, 72:221–230.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann Lecun. 2016. Very deep convolutional networks for text classification. *arXiv preprint arXiv:1606.01781.*

Ankita Dhar, Himadri Mukherjee, Niladri Sekhar Dash, and Kaushik Roy. 2018. Performance of classifiers in bangla text categorization. In *2018 International Conference on Innovations in Science, Engineering and Technology (ICISET)*, pages 168–173. IEEE.

Meng Joo Er, Yong Zhang, Ning Wang, and Mahardhika Pratama. 2016. Attention pooling-based convolutional neural network for sentence modelling. *Information Sciences*, 373:388–403.

Shang Gao, Arvind Ramanathan, and Georgia Tourassi. 2018. Hierarchical convolutional attention networks for text classification. In *Proceedings of The Third Workshop on Representation Learning for NLP*, pages 11–23.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural networks*, 18(5-6):602–610.

Long Guo, Dongxiang Zhang, Lei Wang, Han Wang, and Bin Cui. 2018. Cran: a hybrid cnn-rnn attention-based model for text classification. In *International Conference on Conceptual Modeling*, pages 571–585. Springer.

A. Hassan and A. Mahmood. 2017. Efficient deep learning model for text classification based on recurrent and convolutional layers. In *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pages 1108–1113.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Ramchandra Joshi, Purvi Goel, and Raviraj Joshi. 2019. Deep learning for hindi text classification: A comparison. In *International Conference on Intelligent Human Computer Interaction*, pages 94–101. Springer.

Divyanshu Kakwani, Anoop Kunchukuttan, Satish Golla, NC Gokul, Avik Bhattacharyya, Mitesh M Khapra, and Pratyush Kumar. 2020. inlpsuite: Monolingual corpora, evaluation benchmarks and pre-trained multilingual language models for indian languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4948–4961.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Hoa T Le, Christophe Cerisara, and Alexandre Denis. 2017. Do convolutional networks need to be deep for text classification? *arXiv preprint arXiv:1707.04108*.

Pengfei Liu, Xipeng Qiu, and Xuanjing Huang. 2016. Recurrent neural network for text classification with multi-task learning. *arXiv preprint arXiv:1605.05101*.

Zhenyu Liu, Haiwei Huang, Chaohong Lu, and Shengfei Lyu. 2020. Multichannel cnn with attention for text classification. *arXiv preprint arXiv:2006.16174*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Martin Sundermeyer, Hermann Ney, and Ralf Schlüter. 2015. From feedforward to recurrent lstm neural networks for language modeling. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 23(3):517–529.

Madhuri Tummalapalli, Manoj Chinnakotla, and Radhika Mamidi. 2018. Towards better sentence classification for morphologically rich languages. In *Proceedings of the International Conference on Computational Linguistics and Intelligent Text Processing*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need.

Yequan Wang, Minlie Huang, Xiaoyan Zhu, and Li Zhao. 2016. Attention-based lstm for aspect-level sentiment classification. In *Proceedings of the 2016 conference on empirical methods in natural language processing*, pages 606–615.

Yijun Xiao and Kyunghyun Cho. 2016. Efficient character-level document classification by combining convolution and recurrent layers.

Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*, pages 1480–1489.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. In *Advances in neural information processing systems*, pages 649–657.

Jin Zheng and Limin Zheng. 2019. A hybrid bidirectional recurrent convolutional neural network attention-based model for text classification. *IEEE Access*, 7:106673–106685.

Peng Zhou, Wei Shi, Jun Tian, Zhenyu Qi, Bingchen Li, Hongwei Hao, and Bo Xu. 2016. Attention-based bidirectional long short-term memory networks for relation classification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 207–212.

Author Index