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$_tasks.html$

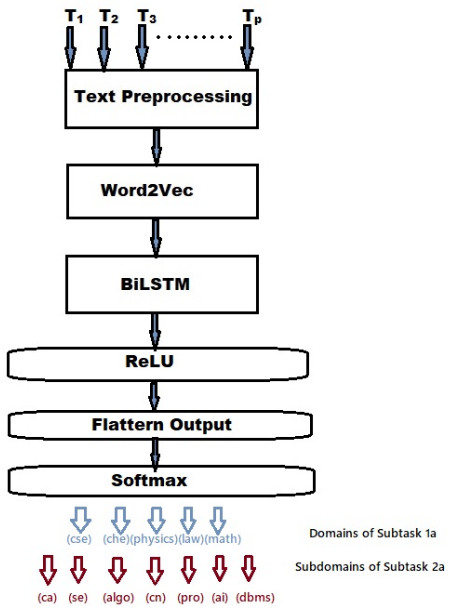

Figure 1: Architechture of BiLSTM + Word2Vec with ReLU and Softmax on the top

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.

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

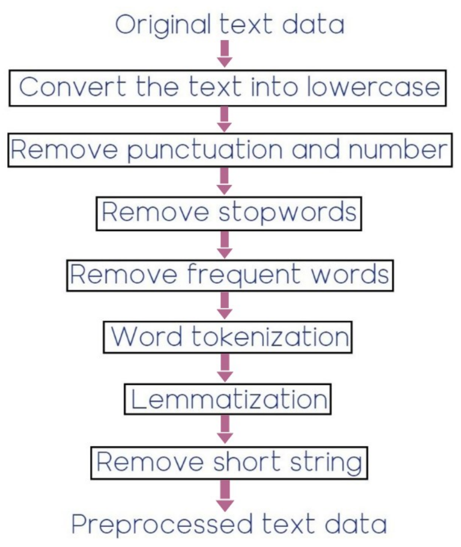

Figure 2: Text preprocessing steps used in this study

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.

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

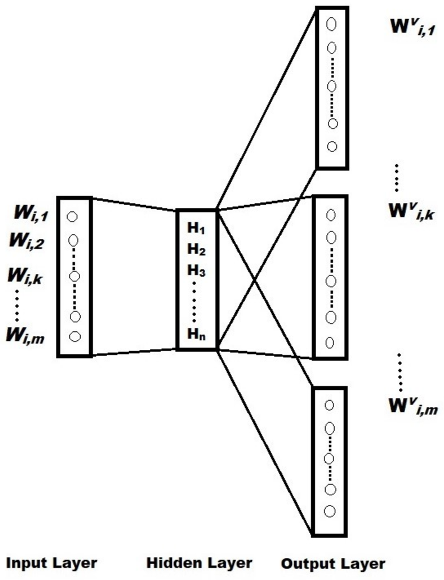

Figure 3: Word2Vec architechture

as input to BiLSTM for feature extraction, namely F^{BiLSTM}_i in equation

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

In Bidirectional LSTM, sequence data is pro-

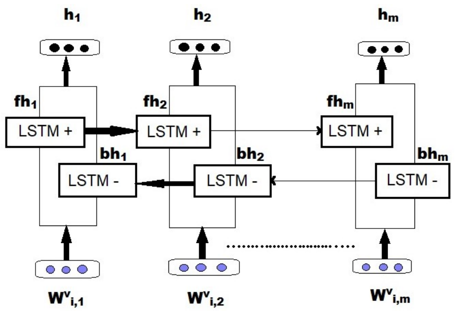

Figure 4: The architecture of basic BiLSTM

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

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

$$f_t = \sigma(W_f.X + b_f) \tag{3}$$

$$i_t = \sigma(W_i.X + b_i) \tag{4}$$

$$o_t = \sigma(W_o.X + b_o) \tag{5}$$

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

$$h_t = o_t * tanh(c_t) \tag{7}$$

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.