International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 1
DOI: https://doi.org/10.17762/ijritcc.v11i1.5983
***Article Received**: 05 November 2022 **Revised**: 18 December 2022 **Accepted**: 25 December 2022*

Transfer Learning based Automated Essay Summarization

Rohith H P[1], Srinivas D B[2], Deepika K M[3], Kavitha Sooda[4], Karunakara Rai B[5]
[1]Dept. of Information Science and Engineering
Nitte Meenakshi Institute of Technology
Bangalore, India
rohith.hp@nmit.ac.in
[2]Dept. of Information Science and Engineering,
Nitte Meenakshi Institute of Technology
Bangalore, India
srinivas.db@nmit.ac.in
[3]Dept. of Information Science and Engineering,
Nitte Meenakshi Institute of Technology
Bangalore, India
deepika.km@nmit.ac.in
[4]Dept. of Computer Science and Engineering,
B.M.S. College of Engineering
Bangalore, India
kavithas.cse@bmsce.ac.in
[5]Dept. of Electronics & Communication,
Nitte Meenakshi Institute of Technology
Bangalore, India
karunakara.rai@nmit.ac.in

Abstract— The human evaluation of essays has become a very time-consuming process as the number of schools and universities has grown. The available software entities are unable to assess the sentiment associated with essays. Thus, we propose a model using Natural Language Processing to assess the essay based on both grammar and sentiment associated with the essay by using linear regression and ULMFiT (Universal Language Model Fine-tuning for Text Classification) models. Evaluation of essay is done in two parts. Part one is on essay grading with respect to grammar with maximum 12 and minimum 0 grade points and in part two score of 0/1 for sentiment analysis with 0 being negative and 1 being positive. The model can be used to score the essay and discard any essay with a score less than a specified value or specified sentiment score.

Keywords- *Linear regression, ULMFiT, essay scoring, sentiment analysis, Transfer Learning, Fastai, RNN, SLR*

I. INTRODUCTION

Artificial Intelligence (AI) and Natural Language Processing (NLP) are commonly used method in text analytics. NLP allows machines to decompose and interpret human language. NLP interpretation process helps in evaluation of essays which is extremely time and labour consuming. Existing automated evaluation open-source software tools fall short on many aspects such as time consuming, not very accurate. Students have to undergo several examinations to prove his/her intellectual ability in today's generation, and one of them is essay writing. Most of the existing essay evaluation tools don't consider sentiment analysis while evaluating the essay and they also don't take all content related features of essays, like nouns, adjectives, adverbs, punctuations, spelling errors, etc. Objective of our proposed model is to integrate both grammar and sentiment analysis together using linear regression and ULMFiT respectively. Features extracted for linear regression will be number of characters, words, sentences, average word length, lemmas, misspelled words and POS tagging (nouns, verbs, adjectives, and adverbs). Numerical Features extracted to indicate the language fluency and dexterity. Essays were tokenized and split using python utilities. The individual tokens were then used to compute word count, sentence count, character count and average word length.

The sentiment score will be derived using a classification model trained using ULMFiT, a popular and accurate transfer learning approach for sentiment analysis. We have taken an essay dataset from the Hewlett foundation which has a total of 12979 essays, each of an average length of 150 to 550 words per response. The responses in this dataset are written by students ranging from grade 7 to grade 10. The dataset contains:

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 1
DOI: https://doi.org/10.17762/ijritcc.v11i1.5983
***Article Received**: 05 November 2022 **Revised**: 18 December 2022 **Accepted**: 25 December 2022*

- **essay_id**- A unique identifier for each individual student essay.
- **essay_se**t- An identifier for each set of essays.
- **essay**- The ASCII text of a student's response.
- **rater1_domain**- Rater 1's domain score. Here, rater1 is a human who rates the essay. Every essay has this score.
- **rater2_domain**- Rater 2's domain score. Here, rater2 is a human who rates the essay. Every essay has this score.
- **domain_score**- Resolved score between the rater1 and rater2. Every essay has this score.

The rest of this paper is organized as follows: Section 2 describes related work. In Section 3 proposed architecture its implementation is discussed; results are discussed in section 4. Finally, we conclude our paper in Section 5.

II. Related Work

A universal Language Model Fine-tuning (ULMFiT), an effective transfer learning method that can be applied to any task in NLP and introduce techniques that are key for fine-tuning a language model [1]. Automatic essay classification system is created using a data set with 13000 essays [2]. Features like num of words, num of characters, num of sentences, average word length, lemma count, parts of speech count, and spell error count are used to build a model.
Manually annotated dataset for automated essay grading is done in [3]. The annotation was done for different attributes like content, organization, style, etc of the essays. Opinion mining on a particular product is done in [4]. Authors have done three level of analysis namely Document Level analysis, Sentence Level analysis, and Entity and Aspect Level analysis. Fundamental concepts and approaches to automatic text summarization is presented in [5]. Authors have proposed various methods of abstractive text summarization like a recurrent neural network, long short-term memory network, encoder-decoder model, and pointer generator mechanism. Trial and error were finished utilizing Google Collaboratory [6]. Various AI models are used to choose for best sentences for the synopsis. Comparison of LSI and LDA for full-text articles and their corresponding abstracts are presented in [7]. Authors also presented the comparison with and without lemmatization in three different areas like computer and information's science and the application. A framework is proposed [8] for removing formal semantic information from unstructured text utilizing an installed CNL. Sentimental Analysis is done on the pre-processed text reviews [9], following a sequence-to-sequence encoderdecoder with an attention layer is used for summarization method, preserving the semantics of the reviews. Various algorithms and methods are used to build text summarization tools [10] and these methods, in individual and together give different types of summaries. Their accuracy score can be compared to find the better and more concise summaries. AI approach is presented for text mining [11]. Estimating the comparability between, sentences, words, records and sections is a significant part indifferent assignments like text outline, data recovery, programmed paper scoring, record grouping, and machine interpretation and word-sense disambiguation.

Proposed architecture is represented in the form of a flowchart as shown in Fig. 1. The architecture uses a dataset consisting of the essays from kaggle (Hewlett foundation dataset). This data set contains eight essay sets each essay range from a median length of 150 to 550 words. Next feature extraction is done by anonymization all personal information such as name, address; organizations etc and essays are tokenized. During this process spelling errors is calculated using the pythons Enchant library. Regular Expression (RegEx) is used to specify the rules/set of rules for the number of possible strings that we can match and get a range of specific strings. Count of characters, words, sentences, avg word length, and lemmas are obtained using regular expressions.

III. Proposed architecture and implementation

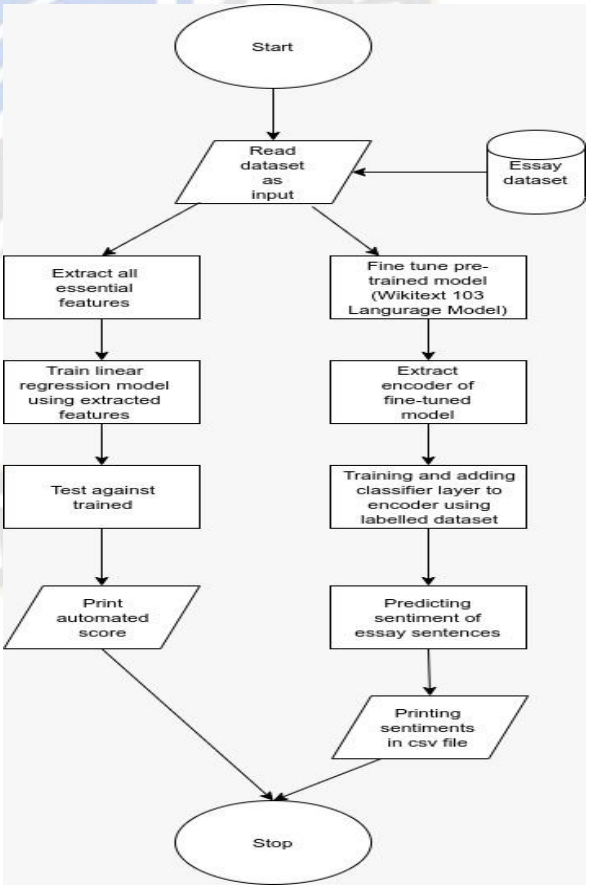

Fig. 1: Architecture

The linear regression model is trained by dividing the dataset into two segments of 70% and 30% respectively and using the

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 1
DOI: https://doi.org/10.17762/ijritcc.v11i1.5983
***Article Received**: 05 November 2022 **Revised**: 18 December 2022 **Accepted**: 25 December 2022*

first segment of the dataset for training the model. The "train_test_split" is used to split the essay set into two partitions which each consist of 70% and 30% of all the essays in the dataset respectively. The extracted features along with the domain score from each essay will be passed as parameters to the linear regression model in order to conduct supervised learning in order to train the model to predict the score of a new essay which is not part of the dataset of essays.Fig.2 and Fig.3 shows tokenization and numerical feature extraction steps.

```
def sentence_to_wordlist(raw_sentence):

    clean_sentence = re.sub("[^a-zA-Z0-9]"," ", raw_sentence)
    tokens = nltk.word_tokenize(clean_sentence)

    return tokens
```

```
def tokenize(essay):
    stripped_essay = essay.strip()

    tokenizer = nltk.data.load('tokenizers/punkt/english.pickle')
    raw_sentences = tokenizer.tokenize(stripped_essay)

    tokenized_sentences = []
    for raw_sentence in raw_sentences:
        if len(raw_sentence) > 0:
            tokenized_sentences.append(sentence_to_wordlist(raw_sentence))

    return tokenized_sentences
```

Fig.2: Tokenization

```
def word_count(essay):

    clean_essay = re.sub(r'\W', ' ', essay)
    words = nltk.word_tokenize(clean_essay)

    return len(words)
```

```
def char_count(essay):

    clean_essay = re.sub(r'\s', '', str(essay).lower())

    return len(clean_essay)
```

```
def sent_count(essay):

    sentences = nltk.sent_tokenize(essay)

    return len(sentences)
```

Fig.3: Numerical Features

A regular expression is used to find or match a character or sequence of characters, using a specialized syntax as shown in Fig 2.Various parts-of-speech such as nouns, adjectives adverbs and verbs are good proxies to test vocabulary. Essays were tokenized into sentences before the tagging process. Before tagging, the essays are divided into a collection of words using tokenization. pos_tag() function which comes under NLTK library is used to compute and tag a tokenized sentence, i.e a special tag is added to the front of the tokenized word. Then we can obtain the count of Noun, Adjective, Verb and Adverb. This tagging process is done by adding N to noun, J to adjective, V to verb, R to adverb etc. In this way parts of speech tagging and counting the parts of speech is done. Fig.4 shows snap shot the process.

```
def count_pos(essay):

    tokenized_sentences = tokenize(essay)

    noun_count = 0
    adj_count = 0
    verb_count = 0
    adv_count = 0

    for sentence in tokenized_sentences:
        tagged_tokens = nltk.pos_tag(sentence)

        for token_tuple in tagged_tokens:
            pos_tag = token_tuple[1]

            if pos_tag.startswith('N'):
                noun_count += 1
            elif pos_tag.startswith('J'):
                adj_count += 1
            elif pos_tag.startswith('V'):
                verb_count += 1
            elif pos_tag.startswith('R'):
                adv_count += 1

    return noun_count, adj_count, verb_count, adv_count
```

Fig. 4: Parts of Speech

```
def count_spell_error(essay):

    clean_essay = re.sub(r'\W', ' ', str(essay).lower())
    clean_essay = re.sub(r'[0-9]', '', clean_essay)
    import enchant
    d=enchant.Dict("en_US")


    mispell_count = 0

    words = clean_essay.split()

    for word in words:
        if(d.check(word)==False):
            #print("HEY")
            mispell_count +=1
    return mispell_count
```

Fig. 5: Spell Check

Snap shot for spell check is as shown in Fig.5. We have used Enchant from python library for spell check. Punctuation is a good indicator of a well structured and organized essay. Lemma count is a good indicator of lexicography and semanticity. Lemma words are nothing but the base form of a given word. A clean essay is first obtained from the given input essay with the help of regex. Clean essays are further tokenized to collection of words. Each word is then tagged with the help of pos_tag. Tagged words are then stripped down to their base word form. Then lemma_append() and lemmatize() functions are used to make a count of lemma words.Fig. 6: shows the code snippet for Lemma Count.
ULMFiT is used to develop a model for the sentiment score. For the purpose of training our model we choose the Wikitext 103 [12]. Wikitext 103 is a dataset contains more than hundred

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 1
DOI: https://doi.org/10.17762/ijritcc.v11i1.5983
***Article Received**: 05 November 2022 **Revised**: 18 December 2022 **Accepted**: 25 December 2022*

million tokens that are featured on Wikipedia in the Good and Featured articles. The dataset is divided into three a). Training set: contains 28,475 articles, 103 million tokens and has vocabulary strength of 267,735 words b) Validation set: contains 60 articles and 217,646 tokens c) Test set: comprises of almost the same attributes as the validation set. To fine-tune our language model, we have changed weights accordingly to get fine-tuned to our target dataset.

```
def count_lemmas(essay):

    tokenized_sentences = tokenize(essay)

    lemmas = []
    wordnet_lemmatizer = WordNetLemmatizer()

    for sentence in tokenized_sentences:
        tagged_tokens = nltk.pos_tag(sentence)

        for token_tuple in tagged_tokens:

            pos_tag = token_tuple[1]

            if pos_tag.startswith('N'):
                pos = wordnet.NOUN
                lemmas.append(wordnet_lemmatizer.lemmatize(token_tuple[0], pos))
            elif pos_tag.startswith('J'):
                pos = wordnet.ADJ
                lemmas.append(wordnet_lemmatizer.lemmatize(token_tuple[0], pos))
            elif pos_tag.startswith('V'):
                pos = wordnet.VERB
                lemmas.append(wordnet_lemmatizer.lemmatize(token_tuple[0], pos))
            elif pos_tag.startswith('R'):
                pos = wordnet.ADV
                lemmas.append(wordnet_lemmatizer.lemmatize(token_tuple[0], pos))
            else:
                pos = wordnet.NOUN
                lemmas.append(wordnet_lemmatizer.lemmatize(token_tuple[0], pos))

    lemma_count = len(set(lemmas))

    return lemma_count
```

Fig. 6: Lemma Count

ULMFiT is used to develop a model for the sentiment score. To train our model we choose Wikitext 103 dataset. Training set contains 28,475 articles, 103 million tokens and 267,735 words. Validation set contains 60 articles and 217,646 tokens. We used a language model called fastai to train Wikitext 103 dataset. In our work, we employed a standard neural network model [13] with a dropout rate of 0.3. Fastai has a function called learn_lm.rec order. plot that plots a graph between loss and learning rate (Fig. 7). The next step is to develop a classifier or the sentiment analyzer using fine-tuned language model. The purpose of this is to predict the next word of a sentence. However, we do not need this. Thus, we replace the last layer of the language model with the sentiment classifier. To do so we have used RNN encoder and decoder. The encoder reads an input sequence and outputs a single vector, and the decoder takes the vector and produces an output sequence.

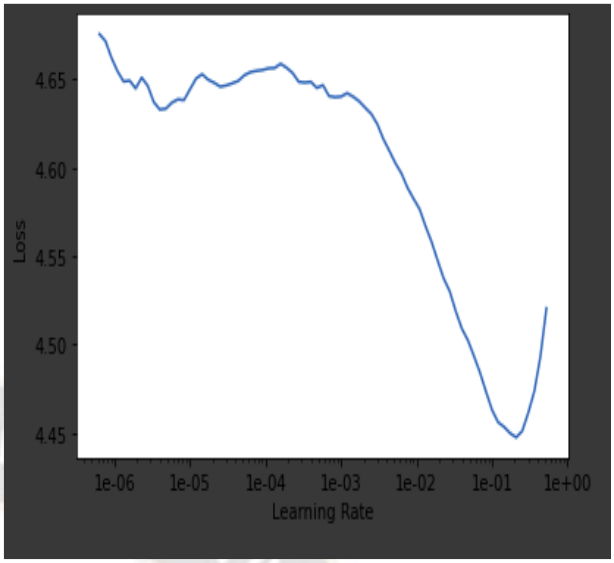

Fig. 7: Loss and Learning Rate

Using the slanted triangular learning rates (STLR), we trained the RNN over 4 epochs. The details of the epochs are as follows- 1.

Epoch 1: Here, we froze the entire model except last layer. In our case, the last layer group is the sentiment analyzer. This is done to make sure the well-trained weights are not modified, which are the crucial part of transfer learning. The learning rate used here is e^{-2} and accuracy achieved at this stage was 74.58%.

Epoch 2: Here, we froze the entire model up to the second last layer group. The learning rate used here was a function slice ($1e^{-03}/(2.6^4)$, $1e^{-03}$). This basically implies that the starting learning rate is $^{1e-03}/(2.6^4)$, while the stop rate is 1e-03. Basically, we are passing a list of learning rates which will then get applied to all the different layer groups. Accuracy achieved at this stage was 80.38%.

Epoch 3: Here, we froze the entire model up to the third last layer group. The learning rate used here was a function slice ($5e^{-03}/ (2.6^4)$, $5e^{-03}$). Accuracy achieved at this stage was 81.99%.

Epoch 4: Here, we unfroze the entire model and trained it as a whole. The learning rate used here was a function slice ($5e^{-03}/(2.6^4)$, $1e^{-03}$). We achieved the maximum accuracy, 82.32% as shown in Fig. 8.

IV. RESULTS

Submissions are scored by passing the entries on to the linear regression model that we have trained, which will in turn give a specific score for the essay based on the features such as no. of nouns, no. of verbs, no. of adjectives, no. of adverbs, no. of spelling errors, no. of characters, no. of words, average word length and no. of lemmas extracted from it, i.e. 12 being the highest and 0 being the lowest for the current essay list from

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 1
DOI: https://doi.org/10.17762/ijritcc.v11i1.5983
***Article Received**: 05 November 2022 **Revised**: 18 December 2022 **Accepted**: 25 December 2022*

the dataset. Using our trained sentiment analyzer that has been trained using our labeled dataset, we provide the sentiment of the whole essay based on the average of all the essay sentences and their respective sentiments, i.e. either 1 for positive sentiment or 0 for negative sentiment**.** The accuracy of the sentiment analysis model was achieved after the 4th epoch, and was 82.32%. Our model can be trained on a dataset of college essays that have been accumulated over time to give the score to new essays which are part of university applications.

epoch	train_loss	valid_loss	accuracy
0	0.438858	0.456902	0.823151

Fig. 8: final accuracy

To get better results, we made use of the concept of granularity and divided the essays into their composite sentences. To predict the sentiment of the essays, the essay sentences were passed through the model, and the sentiment was obtained.Fig.9 shows the sentiments of essay sentences from the dataset.

	sentence	essay_set	sentiment
0	Dear local newspaper, @CAPS1 best friend, @LOC...	76	1
1	Computer not only teaches hand-eye coordinatio...	76	1
2	@CAPS1 friend wasn't the only person who learn...	76	1
3	About @PERCENT1 of people learn hand-eyes coor...	76	1
4	@CAPS3 might takes some month to learn, but on...	76	1
...	...	...	...
603	I remember a tiny when I was on the computer f...	100	0
604	I know the computer is fun playing games, talk...	100	0
605	In conclusion just go outside and have fun!	100	1
606	@CAPS1 you want to wast a nice sunny day by st...	100	0
607	I know I @CAPS1!	100	1

608 rows × 3 columns

Fig. 9: sentiments of essay sentences

As we can see above, the sentiments are represented as follows- 1. Positive labels are represented by 1 2. Negative labels are represented by 0 Upon obtaining the sentiments, the average of the sentiments for each of the essay sentences of every essay was calculated. The averages were a number between 0 and 1.Fig.9 shows the average sentiments of essay sentences.

```
0.7368421052631579
0.9130434782608695
0.59375
0.47058823529411764
0.2
0.8
1.0
0.75
0.8
0.36666666666666664
0.8518518518518519
0.5526315789473685
0.8333333333333334
0.9285714285714286
0.875
0.8571428571428571
0.92
0.8095238095238095
0.7083333333333334
0.7714285714285715
1.0
0.7
0.3939393939393939
0.5263157894736842
```

Fig. 10: average Sentiments of essays

The final essay sentiment was then calculated based on the concept of rounding- any average equal to or above the number 0.5 resulted in a sentiment of 1 or positive, while any average below 0.5 resulted in sentiment of 0 or negative. Fig. 11 shows the final sentiments of essays.

	essay_is	sentiment
0	76	1
1	77	1
2	78	1
3	79	0
4	80	0
5	81	1
6	82	1
7	83	1
8	84	1
9	85	0
10	86	1
11	87	1
12	88	1
13	89	1
14	90	1
15	91	1
16	92	1
17	93	1
18	94	1
19	95	1
20	96	1
21	97	1
22	98	0
23	99	1

Fig.11. final sentiments of essays

V. CONCLUSION

Essay evaluation is becoming increasingly time consuming and labour intensive. Thus, we proposed a Natural Language Processing to assess the essay based on both grammar and sentiment associated with it by using linear regression and

International Journal on Recent and Innovation Trends in Computing and Communication
ISSN: 2321-8169 Volume: 11 Issue: 1
DOI: https://doi.org/10.17762/ijritcc.v11i1.5983
***Article Received**: 05 November 2022 **Revised**: 18 December 2022 **Accepted**: 25 December 2022*

ULMFiT models. The data set used in our work contains eight essay sets each essay range from a median length of 150 to 550 words. Feature extraction is done by anonymization process. During this process spelling errors is calculated using the pythons Enchant library. Regular Expression (RegEx) is used to specify the rules/set of rules for the number of possible strings that we can match and get a range of specific strings. Count of characters, words, sentences, avg word length, and lemmas are obtained using regular expressions. Finally, the sentiment score will be derived using a classification model trained using ULMFiT.

REFERENCES

[1]. Jeremy Howard and Sebastian Ruder.. Universal Language Model Fine-tuning for Text Classification. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Melbourne, Australia. Association for Computational Linguistics, pages 328–339, 2018.

[2]. Song, Shihui, and Jason Zhao. "Automated Essay Scoring Using Machine Learning." –Mach. Learning Sessions, Stanford University, 2012 – Citeseer

[3]. Sandeep Mathias and PushpakBhattacharyya.. ASAP++: Enriching the ASAP Automated Essay Grading Dataset with Essay Attribute Scores. In Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018), Miyazaki, Japan. European Language Resources Association (ELRA), 2018.

[4]. T. K. Shivaprasad and J. Shetty, "Sentiment analysis of product reviews: A review," *International Conference on Inventive Communication and Computational Technologies (ICICCT)*, 2017, pp. 298-301, 2017, doi: 10.1109/ICICCT.2017.7975207.

[5]. Batra, P., Chaudhary, S., Bhatt, K., Varshney, S. and Verma, S., 2020, August. A Review: Abstractive Text Summarization Techniques using NLP. In 2020 International Conference on Advances in Computing, Communication & Materials (ICACCM), pp. 23-28.

[6]. Singh, P., Chhikara, P. and Singh, J., 2020, February. An ensemble approach for extractive text summarization. In 2020 International Conference on Emerging Trends in Information Technology and Engineering (ic-ETITE), pp. 1-7.

[7]. Nazemi, K., Klepsch, M.J., Burkhardt, D. and Kaupp, L., 2020, September. Comparison of full-text articles and abstracts for visual trend analytics through natural language processing. In 2020 24th International Conference Information Visualisation (IV) (pp. 360-367).

[8]. Safwat, H., Gruzitis, N., Davis, B. and Enache, R., 2016, February. Extracting semantic knowledge from unstructured text using embedded controlled language. In 2016 IEEE Tenth International Conference on Semantic Computing (ICSC), pp. 87-90.

[9]. Shah, J., Sagathiya, M., Redij, K. and Hole, V., 2020, July. Natural language processing based abstractive text summarization of reviews. In 2020 International Conference on Electronics and Sustainable Communication Systems (ICESC), pp. 461-466.

[10]. Adhikari, S., 2020, March. Nlp based machine learning approaches for text summarization. In 2020 Fourth International Conference on Computing Methodologies and Communication (ICCMC), pp. 535-538.

[11]. Kudi, P., Manekar, A., Daware, K. and Dhatrak, T., 2014, December. Online Examination with short text matching. In 2014 IEEE Global Conference on Wireless Computing & Networking (GCWCN), pp. 56-60.

[12]. https://www.salesforce.com/products/einstein/ai-research/the-wikitext-dependency-language-modeling-dataset/

[13]. Srivastava, Nitish, et al. "Dropout: a simple way to prevent neural networks from overfitting." *The journal of machine learning research* 15.1 (2014): 1929-1958.

[14]. Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).