JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue

JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)

An International Scholarly Open Access, Peer-reviewed, Refereed Journal

A sentiment-based chat-bot for cloud enabled Edu-care

[1]Ravish G K [2] K Thippeswamy

[1] Research Scholar, [2] Professor

[1]VTU -RRC, Belagavi, India

Abstract: Natural Language Processing (NLP) is a rapidly growing concept in Machine Learning. It is the technology used to aid computers to understand the human's natural language. It is not an easy task teaching machines to understand how we communicate. NLP allows machines to breakdown and interpret human language. It's at the core of tools we use every day from translation software, chat-bots, spam filters, and search engines, to grammar correction software, voice assistants, and social media monitoring tools. Recent years have featured a trend towards pre-trained language representations in NLP systems, applied in increasingly flexible and task agnostic ways for downstream transfer.

Keywords: Natural Language Processing, Cloud computing, Education, Chat-bot.

I. INTRODUCTION

Natural Language Processing (NLP) is a field of Artificial intelligence (AI) that can translate/ transform human languages to machine intelligible form using various techniques such as word stubbing, dictionary development, look-up tables, sentiment analysis, incentivization of tasks and many more. NLP combines the enormous set of linguistics strength and computer science to learn and understand the rules and structure of natural languages and create intelligent systems (run on machine learning and NLP algorithms) capable of understanding, analyzing, and extracting meaning from text and speech.

II. LITERATURE REVIEW

BERT (Bidirectional Encoder Representations from Transformers) [1], is rather recent language translation model. The significant distinction between customary interpretation frameworks and BERT is in the extraordinary element of BERT. It prepares the model in the two ways to be specific, the source to objective and objective to source utilizing content as well as setting. Thus, BERT model when pre-prepared for a given arrangement of dialects can be calibrated with only one extra yield layer to make best in class models for a wide scope of errands, for example, question noting and language deduction, without significant undertaking explicit design changes.

BERT is thoughtfully basic and observationally incredible. It has probably the most noteworthy exactness for language preparing undertakings including interpretation, change, setting mining and opinion examination and so forth the components extricated in pre-prepared BERT can wonderfully work on its exhibition over any traditional single layer or multi-faceted interpretation framework [2].

A new type of deep contextualized word representation that models both (1) complex characteristics of word use (e.g., syntax and semantics), and (2) how these uses vary across linguistic contexts (i.e., to model polysemy) [3] has been introduced. This word vectors are learned functions of the internal states of a deep bidirectional language model (biLM) [4], which is pre-trained on a large text corpus.

It is shown that these portrayals can be effectively added to existing models and fundamentally work on the best in class across six testing NLP issues, including question replying, printed entailment and opinion examination. An examination showing that uncovering the profound internals of the pre-prepared organization is essential, permitting downstream models to blend various sorts of quasi-managed signals is introduced also.

A few variations of the Long Short-Term Memory (LSTM) engineering for intermittent neural organizations have been proposed since its beginning in 1995. Lately, these organizations have turned into the best-in-class models for an assortment of AI issues. This has prompted a reestablished interest in understanding the job and utility of different computational parts of ordinary LSTM variations. In this paper, the principal enormous scope examination of eight LSTM variations on three delegate assignments: discourse

acknowledgment, penmanship acknowledgment, and polyphonic music displaying are presented. The hyper-boundaries of all LSTM variations for each errand were improved independently utilizing irregular hunt, and their significance was surveyed utilizing the incredible ANOVA structure. Altogether, summing up the aftereffects of 5400 exploratory runs (≈ 15 years of CPU time), which makes this review the biggest of its sort on LSTM organizations. The outcomes show that none of the variations can develop the standard LSTM engineering fundamentally and exhibit a negligible opportunity and the yield initiation capacity to be its most basic parts. The concentrated on hyper-boundaries are essentially free and infer rules for their productive change.

In light of their adequacy in expansive pragmatic applications, LSTM networks have gotten an abundance of inclusion in logical diaries, specialized web journals, and execution guides. Nonetheless, in many articles, the surmising equations for the LSTM organization and its parent, RNN, are expressed aphoristically, while the preparation recipes are excluded inside and out. Furthermore, the strategy of "unrolling" a RNN is regularly introduced without legitimization all through the writing. The objective of this paper is to clarify the fundamental RNN and LSTM basics in a solitary archive. Drawing from ideas in signal handling, officially the accepted RNN plan is gotten from differential conditions. Each proposed proclamation is then demonstrated, which yields the RNN unrolling strategy. The survey of the hardships with preparing the standard RNN and address them by changing the RNN into the "Vanilla LSTM" network through a progression of intelligent contentions is the last assignment of LSTM machine plan and improvement. Every one of the situations relating to the LSTM framework along with definite depictions of its constituent elements are given to the expert. Though whimsical, the decision of documentation and the technique for introducing the LSTM framework underscores simplicity of comprehension. As a feature of the investigation, some new chances to enhance the LSTM framework are distinguished and joined into the Vanilla LSTM organization, creating the broadest LSTM variation to date. The objective per user has as of now been presented to RNNs and LSTM networks through various accessible assets and is available to an option instructive methodology.

The study presented so far can reveal two major concerns in the present systems used for cloud based edu-care platforms. The issues in security and delay performance on one hand and the level prediction of the learner based on the question and responses from the previous communications using machine learning on the other.

III. PROPOSED SYSTEM

BERT (Bidirectional Encoder Representations from Transformers) was proposed by Google in 2018. It is a natural language processing system. When it was proposed it achieve state-of-the-art accuracy on many NLP and NLU tasks such as:

- General Language Understanding Evaluation
- Stanford Q/A dataset SQuAD v1.1 and v2.0
- Situation With Adversarial Generations

Google research team had proposed as early as 2018, that the goal of NLP must be focused on improving the understanding of the sense and context of queries to their search engine. The research team collated the Google search queries and deduced that every day, only about 15% of unique and new queries are posted to their world-famous search engine. Therefore, the team stated that, the Google search engine needs to have a much better understanding of the language in order to comprehend the search query.

To improve the language understanding of the model. BERT is trained and tested for different tasks on a different architecture. Some of these tasks with the architecture discussed below.

System Design
BERT Architecture

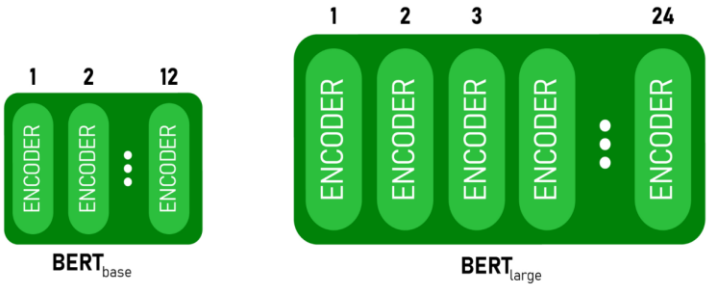

Figure 2. BERT Architectures

BERT's unique bi-directional multi-layer architecture was first defined and implemented by Wasvani etal (2017) and is used as the standard model architecture. The same has been released in the tensor2tensor library. Because the use of Transformers has become common and, in this work, the number of layers (i.e., Transformer blocks) are denoted as L, the hidden size as H, and the number of self-attention heads as A.

Primary report results are presented on two model sizes:

- BERTBASE (L=12, H=768, A=12, Total Parameters=110M)
- BERTLARGE (L=24, H=1024,A=16, Total Parameters=340M).

BERTBASE is set to have the same size as the OpenAI GPT for comparison purposes. However, the distinction among the two is that, BERT uses self-attention in both input-output and vice-versa directions, while the GPT Transformer uses constrained self-attention where every token can only attend to context to its left.

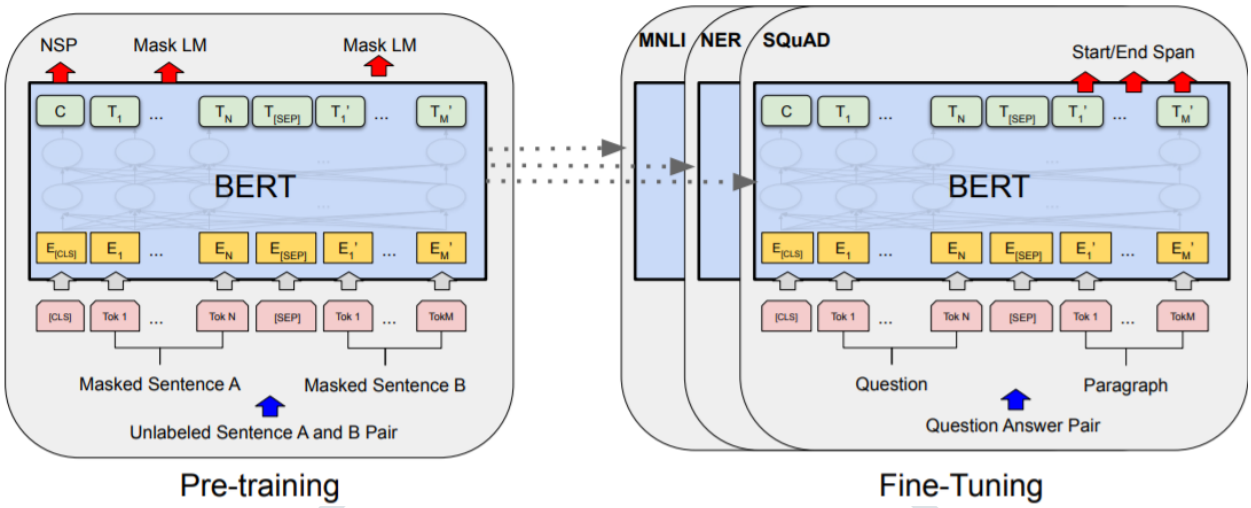

Figure 3. Overall pre-training and fine-tuning procedures for BERT.

Apart from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers).

3.1. Input/ Output Representations

To make BERT handle an assortment of down-stream errands, our feedback portrayal can unambiguously address both a solitary sentence and a couple of sentences (e.g., h Question, Answer i) in one symbolic arrangement. All through this work, a "sentence" can be a subjective range of adjoining message, instead of a real etymological sentence. A "succession" alludes to the information token grouping to BERT, which might be a solitary sentence, or two sentences stuffed together. WordPiece embeddings with a 30,000 symbolic jargon are used. The main badge of each succession is consistently an exceptional characterization token ([CLS]). The last secret state relating to this token is utilized as the total arrangement portrayal for grouping assignments. Sentence sets are stuffed together into a solitary grouping. There are two possible ways of separating the sentences. In the first place, they are separated with an uncommon token ([SEP]). Second, a mastered implanting is added to each token demonstrating whether it has a place with sentence An or sentence B. As displayed in Figure 8.4, mean information inserting as E, the last secret vector of the unique [CLS] token as C ∈ R H, and the last secret vector for the ith input token asTi∈ R H. For a given token, its feedback portrayal is developed by adding the relating token, section, and position embeddings. A perception of this development can be found in the figure below.

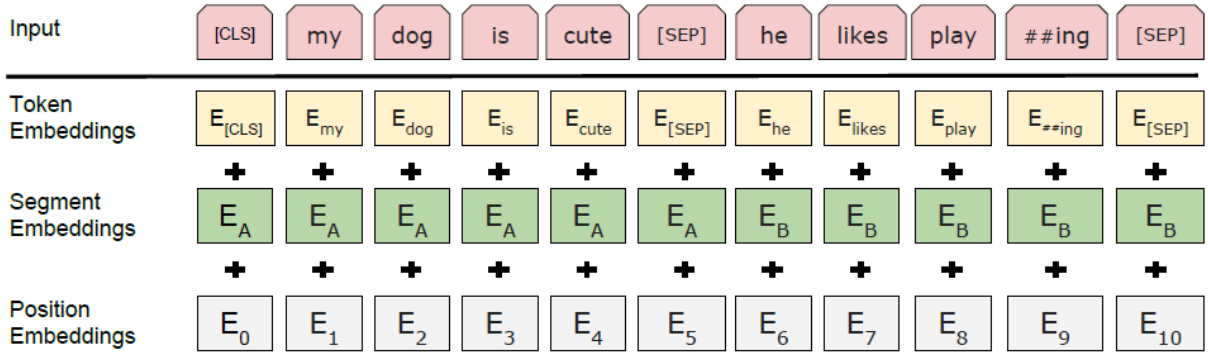

Figure 4. BERT input representation.

The input embeddings are the sum of the token embeddings, the segmentation embeddings, and the position embeddings.

3.2. Masked Language Model:

Instinctively, it is sensible to accept that a profound bidirectional model is completely more remarkable than either a left-to-right model or the shallow connection of a left-to-right and an option to-left model. Lamentably, standard restrictive language models must be prepared left-to-right or right-to-left, since bidirectional molding would permit each word to by implication "see itself", and the model could inconsequentially foresee the objective word in a multifaceted setting. The previous is frequently alluded to as a "Transformer encoder" while the left-setting just form is alluded to as a "Transformer decoder" since it tends to be utilized for text age. To prepare a profound bidirectional portrayal, some of the information tokens are covered aimlessly, and afterward anticipate those

concealed tokens. The affinity towards this strategy as a "covered LM" (MLM), even though it is regularly alluded to as a Closed task in writing (Taylor, 1953). For this situation, the last secret vectors comparing to the masked tokens are taken care of into a yield softmax over the vocabulary, as in a standard LM. In the entirety of the examinations, If one masks 15% of all WordPiece tokens in each grouping indiscriminately. As opposed to de-noising auto-encoders (Vincent et al., 2008), Only veiled words are anticipated instead of recreating the whole info. Albeit this permits the user to get a bidirectional pre-prepared model, a drawback is that a bungle between pre-preparing and calibrating, since the [MASK] token doesn't show up during tweaking. To moderate this, "covered" words are not generally supplanted with the genuine [MASK] token. The preparation information generator picks 15% of the symbolic situations at arbitrary for expectation. On the off chance that the ith token is picked, the ith token is supplanted with (1) the [MASK] token 80% of the time (2) an irregular token 10% of the time (3) the unaltered ith token 10% of the time. Then, at that point, Ti will be utilized to anticipate the first token with cross entropy loss.

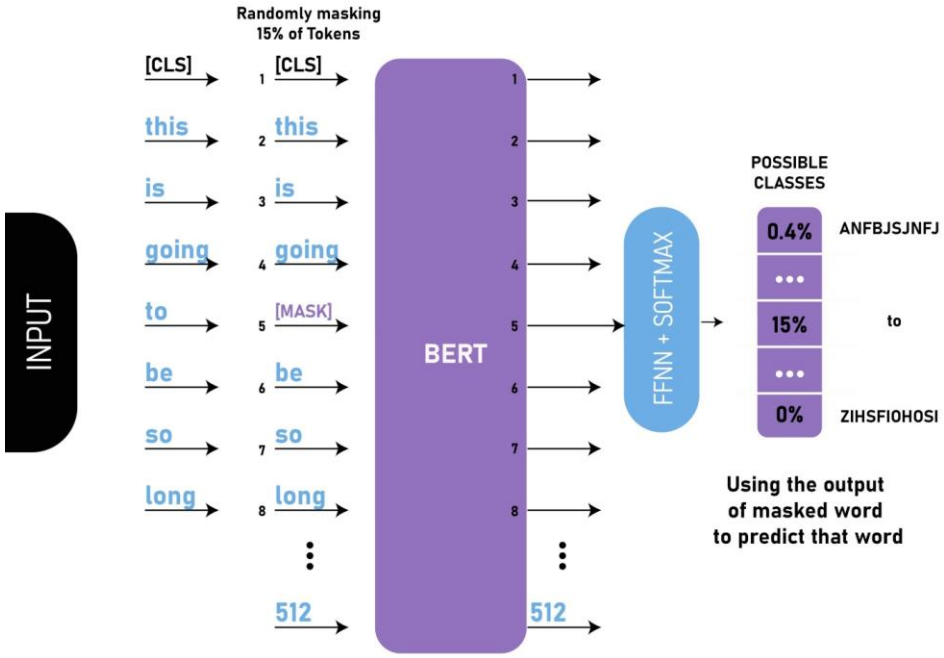

Figure 5. Masked Language Model

3.3. Next Sentence Prediction (NSP):

Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between two sentences, which is not directly captured by language modeling. To train a model that understands sentence relationships, binarized next sentence prediction task is pre-trained so that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences, A and B for each pre-training example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as NotNext). As shown in Figure 1, C is used for next sentence prediction (NSP).

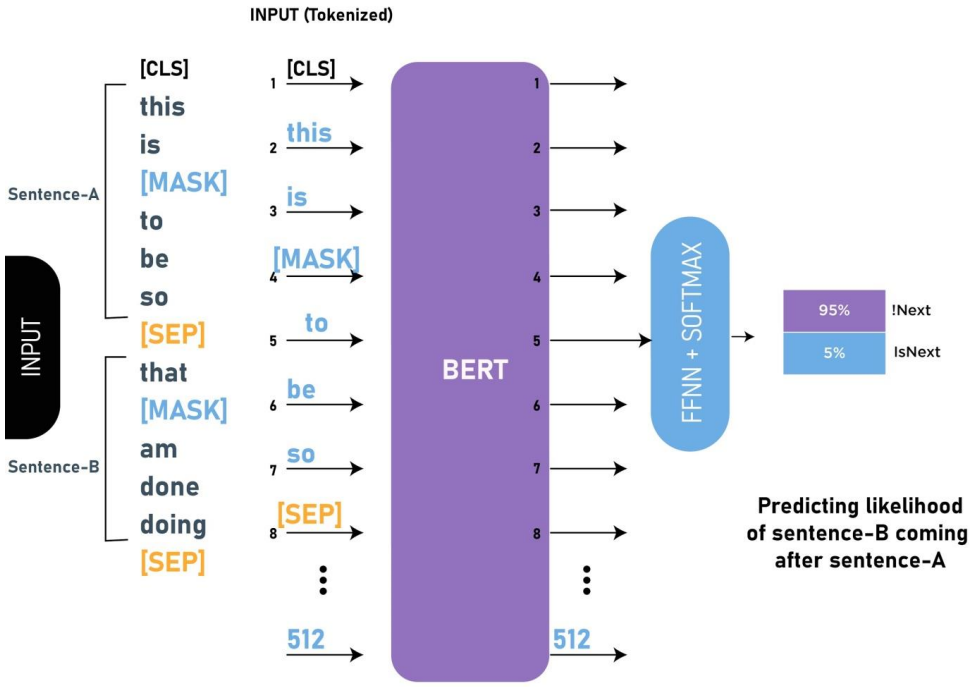

Figure 6. Next Sentence Prediction.

3.4. ELMo Word Embeddings:

ELMo(Embeddings for Language Models) is different from these embeddings because it gives embedding to a word based on its context i.e, contextualized word-embeddings. To generate embedding of a word, ELMo looks at the entire sentence instead of a fixed embedding for a word. ELMo uses a bidirectional LSTM trained for the specific task to be able to create those embeddings. This model is trained on a massive dataset in the language of our dataset, and then we can use it as a component in other architectures that are required to perform specific language tasks. ELMo gained its language understanding from being trained to predict the next word in a sequence of words – a task called Language Modeling. This is convenient because we have vast amounts of text data that such a model can learn from without labels.

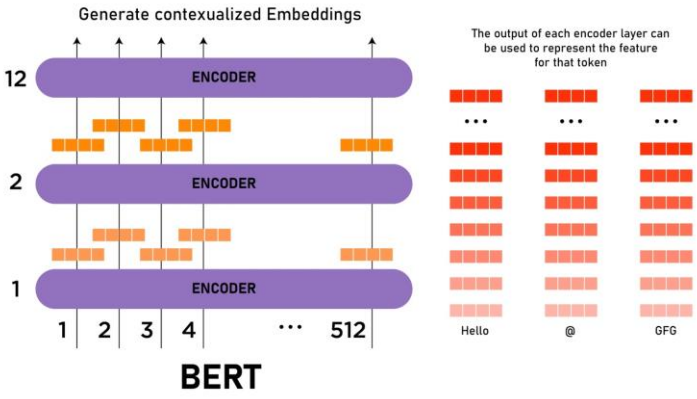

Figure 7. Elmo Contextualize Embeddings Architecture.

IV. IMPLEMENTATION

4.1. Django Blog Application:

The blog application is developed using Python Django. Django is a high-level Python web framework that enables rapid development of secure and maintainable websites. There are two kinds of users. An admin and a normal user. The admin can login and create a blog post. A normal user can only view the post.

Each blog will have a comments section where the user can post a review of a blog. A speech to text feature is provided so the review can either be typed or spoken. After submitting the review, the text data is sent to our TensorFlow BERT Model and the sentiment of the text will be extracted and will be scored.

After the sentiment is predicted along with the score, the rating out of 10 will be displayed as a score of the blog above the comment.

The use of the sentiment is not restricted to this. When viewing the home page, recent blogs that are created will be displayed. When viewing all the blogs, it can be sorted by the top ratings which will be calculated by analyzing all the comments of all the blogs. Along with the top-comment sort, blogs can also be sorted by date added. There is an analytics page for the admin where he/she can view the statistics of all the blogs such as most liked blog, least liked blog etc.

4.2. Application Development:

The frontend used for this application was built using HTML, CSS and JavaScript. The backend was built using Django. Python was the backend programming language. The speech to text module was built using JavaScript. The text data from the comments would be appended to a list and then input to the Machine Learning model.

The Machine Learning model was built using TensorFlow Keras. The dataset used to train the model was the IMDb dataset. With the help of the now trained model, all the comments' sentiment will be determined and stored in the database.

V. RESULTS AND DISCUSSION

Table 1. Responses for various question key words

				Iteration 1		Iteration 2		Iteration 3		Iteration 4	
Sl no	Learning Levels	Queries	Number of queries	Correct Responses	Incorrect Responses	Correct Responses	Incorrect Responses	Correct Responses	Incorrect Responses	Correct Responses	Incorrect Responses
1	Memory	What	17000	16753	247	16787	213	16838	162	16923	77
		Define	12985	12255	730	12280	705	12317	668	12379	606
		Recall	6895	5968	927	5980	915	5998	897	6028	867
		Draw	4657	4086	571	4095	562	4108	549	4129	528
		Iterate	4616	3981	635	3989	627	4001	615	4022	594
		List	6124	6112	12	6124	0	6124	0	6124	0
2	Knowledge	Understand	10356	10255	101	10276	80	10307	49	10356	0
		Compare	8994	8968	26	8986	8	8994	0	8994	0
		Differentiate	8263	7997	266	8013	250	8038	225	8079	184
		Contrast	7324	6819	505	6833	491	6854	470	6889	435
		Infer	5603	4948	655	4958	645	4973	630	4998	605
3	Problem solving	Solve	4755	3874	882	3882	874	3894	862	3914	842
		Apply	3638	2650	988	2656	982	2664	974	2678	960
		Derive	2520	2426	94	2431	89	2439	81	2452	68
		Obtain	1402	1202	200	1205	197	1209	193	1216	186
		Arrive	285	192	93	193	92	194	91	195	90
4	Creativity	Design	1833	1816	17	1820	13	1826	7	1833	0
		Develop	1956	1346	610	1349	607	1354	602	1361	595
		Build	3068	2963	105	2969	99	2978	90	2993	75
		Discuss	3968	3917	51	3925	43	3937	31	3957	11
		Debate	5002	4143	859	4152	850	4165	837	4186	816
5	Values	Learn	6412	6365	47	6378	34	6398	14	6412	0
		Grow	7358	7198	160	7213	145	7235	123	7272	86
		Strengthen	8656	8541	115	8559	97	8585	71	8628	28
		Help	9774	9658	116	9678	96	9708	66	9757	17

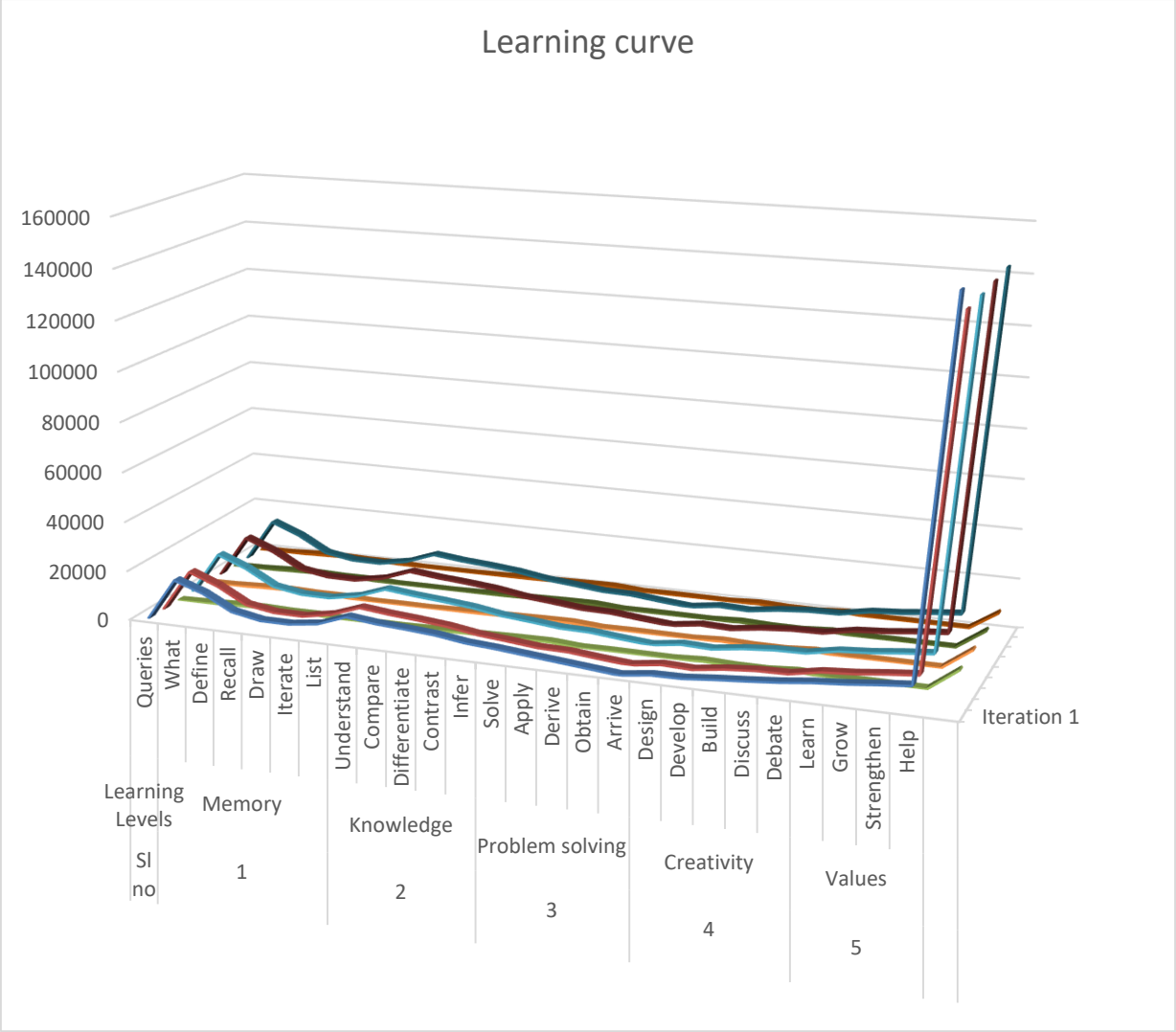

Figure 8. The learning curve of the chat-bot

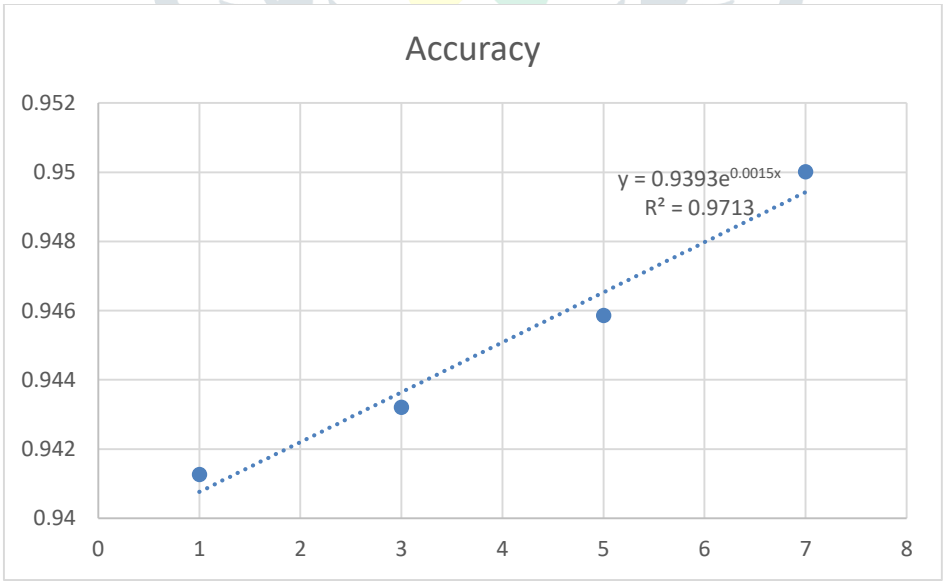

Figure 9. Curve fitting for the accuracy

VI. CONCLUSIONS AND FUTURE WORK

A real-world application using Sentiment Analysis and speech recognition is successfully built and demonstrated. The admin can create a blog and the user can comment on the post. After a blog is posted, the user can comment. The sentiment is later extracted and stored in the database. Upon multiple comments, the average is calculated and stored in the database. The same score will be displayed as a rating to the blog post. There is also an analytics page where the scores of all the blogs can be viewed.

This research work is the basic extension of how sentiment analysis can be used in a real world application. Here, a blog site where an application which uses the BERT model which can be completely used to filter reviews provided by the users based on sentiments is presented. To make the process of inculcating this system much easier for developers there can be a plan on providing a plugin similar to a browser web extension where one can add it to a website just by installing it as an extension for the browser.

One of the other goals of this project is to build an AI chat-bot that analyzes each text by the user and provide a human-like response. The developed chat-bot would be better than the existing chat bots because of sentiment filtering. Along with that, if the user exhibits an alarming behavior, the chat-bot should contact the nearest help center to reach the user immediately.

REFERENCES

[1]. https://arxiv.org/abs/1810.04805 - BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding
[2]. https://arxiv.org/abs/1802.05365 - Deep contextualized word representations
[3]. https://arxiv.org/abs/1706.03762 - Attention Is All You Need
[4]. https://arxiv.org/abs/1503.04069 - LSTM: A Search Space Odyssey
[5]. https://arxiv.org/abs/1808.03314 - Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network
[6]. https://medium.com/analytics-vidhya/bow-vs-bert-classification-1f24ba87240f - BoW to BERT: Classify This!
[7]. https://tfhub.dev/google/collections/bert/1 - TensorFlow BERT
[8]. https://www.geeksforgeeks.org/getting-started-with-transformers/ - Getting Started with Transformers
[9]. https://www.geeksforgeeks.org/explanation-of-bert-model-nlp - Explanation of BERT Model – NLP
[10]. https://www.geeksforgeeks.org/understanding-bert-nlp - Understanding BERT – NLP
[11]. https://www.geeksforgeeks.org/sentiment-classification-using-bert - Sentiment Classification Using BERT
[12]. https://www.geeksforgeeks.org/self-attention-in-nlp - Self -attention in NLP
[13]. https://towardsdatascience.com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270 - BERT Explained: State of the art language model for NLP
[14]. https://tfhub.dev/google/collections/bert - TensorFlow BERT
[15]. https://www.geeksforgeeks.org/getting-started-with-transformers - Getting Started with Transformers
[16]. https://developer.mozilla.org/en-US/docs/Web/API/Web_Speech_API - Mozilla Web Speech API