ISSN: 2583-9888
DOI: doi.org/10.51219/JAIMLD/siva-prasad-patnayakuni/21

URF PUBLISHERS
connect with research world

Journal of Artificial Intelligence, Machine Learning and Data Science

https://urfpublishers.com/journal/artificial-intelligence

Vol: 1 & Iss: 4

Research Article

A BERT based Deep Learning Approach for Suggestion Mining

Siva Prasad Patnayakuni[1]*and Naga Shiva Harish Yedidi[2]

[1]Senior Data Engineer, H-E-B, USA

[2]Gayatri Vidya Parishad College of Engineering (Autonomous), Visakhapatnam, Andhra Pradesh, India

Citation: Patnayakuni SP, Yedidi NSH. A BERT based Deep Learning Approach for Suggestion Mining. *J Artif Intell Mach Learn & Data Sci 2024*, 1(4), 19-27. DOI: doi.org/10.51219/JAIMLD/siva-prasad-patnayakuni/21

Received: 17 February, 2024; **Accepted:** 22 February, 2024; **Published:** 02 March, 2024

***Corresponding author:** Siva Prasad Patnayakun, Senior Data Engineer, H-E-B, USA, E-mail: sivaprasad.patnayakuni@gmail.com

A B S T R A C T

The textual information is uploaded tremendously in the internet with different forms of social network platforms like Twitter, Facebook, Blogs, Forums and Review Sites. Most of the customers depend on opinions of consumers to purchase any item or visiting any place. Sentiment analysis is one such research area to extract the opinions like positive, negative or neutral of consumers on a particular product or service by analysing their reviews. The reviewers also specify suggestions to companies or organizations to improve their quality of product or service. Suggestion mining is one type of sentiment analysis which is used to extract suggestions like tips, wishes and recommendations from the text by analysing their text. The researchers proposed different types of methods like rule based, feature based, machine learning based and deep learning based approaches for suggestion mining. In machine learning based approaches, it is difficult to represent the text by considering syntactic and semantic information of text. In this work, a Deep Learning based approach is proposed for suggestion mining to identify whether a sentence is a suggestion or non-suggestion. The SemEval 2019 Competition subtask-A suggestion mining dataset is used in this experiment. The word2vec and GloVe methods are used for generating word embeddings. Different deep learning techniques such as Recurrent Neural Networks, Long Short Term Memory, Gated Recurrent Unit and BERT models are used in this experimentation. The BERT model attained best accuracy for suggestion mining when compared with other deep learning techniques.

Keywords: BERT, RNN, LSTM, GRU, Suggestion Mining

1 Introduction

The rapid expansion of the internet has established a main way for humans to exchange information and ideas. Even though millions of hours of video and audio clips get uploaded and streamed online each day, written text is still the dominant medium of this information exchange. The texts generated by Review-based portals and online forums contain different types of opinions like positive, negative or neutral on products or services which is the subject of sentiment analysis. The information extracted from these sources offer valuable information to the customers about products and services. On the other hand, the users also specify different types of suggestions to the companies to improve their quality of product or service. Unlike opinions, suggestions can appear in different parts of text and also appear more sparsely. Suggestion mining or identification of suggestions within the text is a relatively new area which is gaining popularity among many private and public sector organizations, service providers and consumers/customers at large due to its number of uses.

Suggestion mining is defined as the extraction of suggestions from unstructured text, where the term 'suggestions' refers to the expressions of tips, advice, recommendations etc. We often see comments on products in product forums which are recommended or not recommended, and some users will consider

whether to purchase the product based on these comments. Suggestion mining is also defined as automatic extraction of recommendations from a given text. These texts that express user suggestions can usually be found in social media platforms, blogs, or product online forums. Suggestion mining is a trending research domain in recent times to enhance the product or service quality. To better recognize suggestions, instead of only matching feature words, one must have the ability to understand long and complex sentences.

Sentiment analysis is a process of computationally identifying and categorizing the opinions from unstructured data. This can be used to identify a user's perspective of a product like positive, negative or neutral. Opinion mining is used to identify whether the product is a success in the market or not. Suggestion mining finds out ways to enhance the product to satisfy the customers. Review texts are mainly used to identify the sentiments of the user. Besides sentiments, review texts also contain valuable information such as advice, recommendations, tips and suggestions on a variety of points of interest. These suggestions will help other customers to make their choices and the sellers improve their products. Suggestion mining is relatively a young field of research compared to sentiment analysis. While mining for suggestions, the propositional aspects like mood, modality, sarcasm, and compound statements have to be considered. It is observed that, in some cases, grammatical properties of the sentence alone can be used to identify the label, while in other cases semantics play a significant role in label classification[1].

Several applications such as Customer to customer suggestions, Enhancement of sentiment detection, product improvement, Recommender systems and Suggestion summarization used the techniques of suggestion mining[2]. The customers are giving suggestions like warnings, tips and advice about different commercial entities in blogs, reviews and other social media platforms. Suggestions are mined mainly for business people to improve the product or for fellow customers to detect advice. These suggestions are helpful for the other customers before selecting any entity. The sentiment analysis extracts opinions like positive, negative or neutral opinion on entities from the text. The sentiment analysis is poor in detection of irrelevant and neutral sentiment in the sentences. In this aspect, the suggestions are helpful for the experts of sentiment analysis to improve the detection of sentiment in the text. The suggestions of consumers about a product are helpful for the product developers, owners and implementers to enhance their product quality. The recommender systems recommend the products based on the opinions of the consumers on the product. The suggestion mining system improves the behaviour of recommender systems by adding additional information of suggestions to recommend the better products. The opinionated text contains different opinions on different entities. The suggestion mining techniques are used to summarize the different opinions on a product.

Unlike opinions, suggestions can be more likely extracted also by pattern matching. Some researchers extracted suggestions by using different heuristic features and keywords such as suggest, recommend, advise[3]. Some works deal with domain terminology, thesaurus, linguistic parser and extraction rules[4]. Linguistic rules were also used for identification and extraction of suggestions in sentiment expressions[5]. Suggestion mining can be realized as standard text classification and classified into two classes such as suggestion and non-suggestion.

The traditional approach to any text classification problem is by extracting features from the dataset or by encoding numeric vectors to represent the text. When using a text classifier, one of the simplest ways to represent text is to use bag-of-words (BOW), where each word (feature) in the text is stored together with their relative frequency and ignoring word position. It also requires huge corpus of labelled data to have good representation of the data. A more advanced way to represent features is by using word embeddings, where each feature is mapped to a vector of numbers also called as embedding. One popular way to create these word embeddings is to use word 2vec[6]. Word 2vec can capture the semantic meaning of words, providing more competent feature representations than the previously mentioned BOW approach. The embeddings from word2vec work relatively well, but are still missing an essential aspect of natural language context. In this work, we present a deep learning based approach for suggestion mining that will eliminate the steps required to manually extract features by leveraging pre-trained embeddings generated using BERT to form contextual representation of the textual statements that preserves the semantics and syntax of the language. The experiment conducted with RNN, LSTM, GRU and BERT techniques on the dataset provided in Sem Eval 2019 competition task A.

This work is structured in 8 sections. The existing works of suggestion mining is analysed in Section 2. The description about dataset is presented in Section 3. The proposed deep learning based approach for suggestion mining is presented in Section 4. The word embeddings are explained in Section 5. The deep learning techniques are described in Section 6. The experimental results are expressed in Section 7. The conclusions and future directions of our work mentioned in Section 8.

2. Related Work

Suggestion mining is defined as the extraction of suggestions from unstructured text[7]. Suggestion mining is still a relatively young research area as compared to other natural language processing issues like sentiment analysis[3]. While suggestion mining is of great commercial value for organizations to improve the quality of their entities by considering the positive and negative opinions collected from platforms. The target of this task is to automatically classify the sentences collected from online reviews and forums into two classes which are suggestion and non-suggestion[8].

Authors[9] presented a system for Suggestion Mining task. The proposed system employed ensemble of Domain-Adversarial Neural Networks (DANN) by using Structured Self-Attentive Sentence Embedding as a feature extractor. The part-of-speech tagging is used to achieve better adaptation towards target domain and the DANN extended with a Target Preserving component in a form of words decoder for target domain sentences. The proposed system reached F1 score up to 0.778 on test dataset. They achieved 7th rank in this subtask B.

Authors[10] described a system based on directed un-weighted graphs for Suggestion Mining Sub Task A of Sem Eval 2019. During the evaluation, the proposed system got 31st rank in the competition subtask A dataset. They discussed the results of a system in the development, evaluation and post evaluation. In this system, each class in the dataset is represented as directed unweighted graphs. Then, the comparison is carried out with each class graph which results in a vector. This vector is used as features by a machine learning algorithm. The model is evaluated

on hold on strategy. The binary class system achieved evaluation value of 0.35. Authors[11] proposed an approach by using expert defined rules and compared with machine learning models. They observed that the expert rules achieved best performance when compared with machine learning model. For the machine learning approach, they tried with wide spread of feature types such as character n-grams, token unigrams, token n-grams, syntactic structure of the main clause, syntactic rewrites of all constituents and syntactic n-grams.

Authors[12] discusses the rule-based method to find out the suggestions from the reviews. They have identified two kinds of 'wishes' namely the desire to improve the product and the desire to purchase the product. They have formulated the rules using modal verbs and certain sentence patterns. Authors[13] develop an ontology-based knowledge representation for suggestion mining.

Authors[14] proposed a suggestion mining system for the task A and B in SemEval-2019. This system used linguistic rule-based method for feature extraction, SMOTE sampling technique is used for data augmentation to solve the imbalance problem in the dataset and deep learning technique (Convolutional Neural Network (CNN)) for classification. The Bag of Words (BOW) features are used in MLP, RF and CNN classifier for suggestion mining. The results of CNN are compared with Random Forest classifier (RF) and Multi-Layer Perceptron (MLP) model. They found that the performance of CNN model is good when compared to MLP and RF classifiers for both the subtasks.

Authors[15] presented a study about different classification algorithms like Random Forest (RF), Logistic Regression (LR), Sublinear Support Vector Classification (SSVC), Multinomial Naive Bayes (MNB), Linear Support Vector Classification (LSVC), CNN and Variable Length Chromosome Genetic Algorithm-Naive Bayes (VLCGA-NB). The developed system attained F1-scores of 0.37 and 0.47 on evaluation data of task B and task A respectively. They find that the results of this system outperformed the base line results of suggestion mining. The MNB achieved best F1-score for non-suggestion class and SSVC attained best F1-score for Suggestion class.

Authors[16] built a domain-independent suggestion mining model to classify suggestion sentences by developing a hybrid approach which uses rule-based features along with RNN. The proposed model obtained a F1-score of 74.49% for suggestion mining which is higher than baseline accuracy of 57.77%. Ilia Markov et al., experimented [P46] with different types of handcrafted features, digits, sentiment features, verbs and function words for subtask A and handcrafted features for subtask-B. They used handcrafted keywords of 57 for subtask-A and keywords of 77 for subtask-B. The Support Vector Machines (SVM) classifier is used for training purpose. Term frequency measure is used to determine the feature value in the vector representation. The proposed system attained F1-scores of 51.18 and 73.30 for task A and B respectively.

3. Description of Dataset

"Task 9 - Suggestion mining from online reviews and forums" is introduced in Sem Eval 2019 competition which has two subtasks[8]. Subtask A is to classify a sentence into a suggestion or a non-suggestion. Subtask B is a cross-domain testing in which the model learned from a domain-specific dataset and this model is used to classify dataset from a new domain. For subtask-A, suggestion forum dataset is used for training and testing. For subtask-B, suggestion forum dataset is used for training and hotel review dataset is used for testing purposes. The suggestion forum training dataset has 2085 instances of suggestion class and 6415 instances of non-suggestion class. The trial test set for suggestion forum dataset has equal number of instances (296) for both classes. In this work, subtask A dataset is considered for experiment. The dataset for suggestion mining task consists of feedback posts on Universal Windows Platform available on uservoice.com. The training and development sentences are combined in the final dataset. The final dataset divided into 70% data for training and 30% of data for testing. The dataset description is expressed in **Table 1**.

Table 1: Description about dataset for subtask A.

	Training	Development	Test
Suggestions Sentences	2085	296	87
Non Suggestions Sentences	6415	296	746
All	8500	592	833

The researchers presented their results with different performance metrics like recall, precision, accuracy, F1-Score. In this work, accuracy metric is used for testing the suggestion mining performance. Accuracy is number of test sentences predicted their suggestion correctly from total sentences. In this work, different pre-processing techniques are applied on the dataset to remove irrelevant data. The pre-processing of input samples is one of the most important phases in natural language processing. For user generated content, pre-processing is even more important due to noisy and ungrammatical text. For this suggestion mining task, we used different pre-processing techniques such as removal of non-alphabetic characters, expanding the contraction words, lowercase conversion and removal of punctuation characters.

4. Proposed Deep Learning based Approach for Suggestion Mining

In this work, deep learning based approach is proposed for suggestion mining. The proposed approach is displayed in Fig. 1.

Figure 1: The deep learning based approach for suggestion mining.

In this approach, different pre-processing techniques are applied on the dataset to prepare the dataset for extracting suitable words. After cleaning the dataset, extract the words.

The Word2Vec and GloVe methods are used to generate the word embeddings. The word embeddings are given as input to different deep learning techniques such as RNN, LSTM, GRU and BERT. The deep learning techniques predict the accuracy of suggestion mining. The next sections explain the word embedding techniques and deep learning techniques.

5. Word Embeddings

One of the most basic approaches for representing the text is a Bag-Of-Words (BOW) model, where each word in the text gets transformed into a vector and mapped with another vector consisting of the corresponding word counts. This is simple and quite effective procedure in many cases, but it comes with some issues. One of the main ones is that a BOW representation disregards the position of individual words in a text. Natural language is heavily context dependant and removing a word from its context changes its semantic meaning completely. Another problem is the sparse encoding of the vectors that makes them sub-optimal for machine learning algorithms[17].

An alternative approach to BOW is to use a vector semantics method, where each word is represented in a multidimensional semantic space. This method assumes that words which are frequently co-occurring also share some semantic meaning. These co-occurring words are modelled as vectors also referred to as embeddings. A semantic similarity measure is obtained by calculating the dot product of two embeddings which indicates how close they are to each other in the semantic space and how similar the words represented by the embeddings are. The embeddings are based on a term-term matrix[17]. This means that each word has its own row in the matrix with columns corresponding to all the words in the vocabulary. Each cell indicates how often the word in the row co-occurs with the word in the column. These rows are constitutes the embeddings. However, due to how natural language is structured, each word in the vocabulary will only co-occur with a small subset of the total number of words. As a consequence, most of the cells in the matrix are going to be empty which means that a large portion of each embedding will be filled with zeros. This is often called sparse vector representations.

The sparse vectors are often replaced by dense vectors. The dense vectors offer some advantages over the sparse vectors, where the most prominent one is that their fewer dimensions is a better fit for feature representation in machine learning systems. A system that uses shorter, more dense vectors, have to learn substantially fewer weights than a system using sparse vectors. This has a significant impact on the training times of the system. Also, due to the reduced number of weights, a system with dense vectors is less susceptible to over-fitting problems[17]. All machine learning techniques require representation of text in numerical format so that it can be consumed by the neural networks. The representation of the words is learned by training models with enormous text data. The model tries to form representation of the word by taking into account the words adjacent to it. Words which are similar to each other will have their positions nearby to each other in vector space and vice versa for dissimilar words.

A Numerical representation which can capture syntactic and semantics of the language is the most effective way of a good embedding method. Traditionally researchers have spent a lot of time in formulating embeddings for a particular dataset before even moving to the actual task. Modern word embedding techniques reduces the number of dimensions from thousands to few hundreds. One of the important parameter in word embedding is deciding the number of dimension for representation which is best figured out empirically. Higher number of dimensions might bring better accuracy sometimes but it also brings in extra computational cost. A right balance needs to be figured out and is generally dependent on the dataset and the problem which needs to be solved. To generate dense vector representation of words, Word2vec needs a local context window parameter to define the number of words to consider while training embeddings.

Several researchers used word2vec and fastext for representing text data into vector format. Word2Vec used neural network to form representation of words by learning their associations with other words from large corpus of text. Fastext is an extension to word 2vec which learns embeddings of n-gram or characters in text instead of the whole word. The embeddings were trained from scratch using only the dataset. Word 2Vec and Fastext models used gensim implementation to train their word embeddings by using both skip gram and Continuous bag of words (CBOW) approaches. CBOW is a type of architecture used in word2vec model which tries to predict a word given other words in context. Skip-gram on the other hand tries to predict the context given the target word. Even though methods like word 2vec was used as feature representation to great success on many different tasks, the technique has its limitations, mainly with capturing the context of each word embedding.

The word 2vec and GloVe use the co-occurrences between a context word and a target word. Context-dependent embeddings uses whole sequences to capture the context of a target word to create embeddings. This method of selecting sequences instead of individual words to infer the context provides information about the sequential relationships of the words in a sequence, instead of just observing if they occur in the proximity of a target word. One example of these richer embeddings is context 2vec[18]. Context 2vec builds upon the word 2vec architecture, and introduces bi-directional Long Short Term Memory (LSTM) networks[19] to replace the context modelling of averaged word embeddings in a fixed window.

6. Deep Learning Approaches

In machine learning techniques, documents are classified based on the past observations and the features are designed using handcrafted features or extracted using other ML techniques. These features are then trained in a supervised manner with a suitable ML algorithm. Support Vector Machine (SVM), Naive Bayes, Decision Tree are few examples of classifiers.

Traditionally, in any text classification problem huge amounts of resources are spent in understanding data, handcrafting features and most importantly generating embeddings. Embeddings are vector representation for text data. In cases where data is sparse, representation of text data will never achieve a satisfactory level. Recently many pre-trained language models were open sourced with the purpose of NLP research without expensive training pre-requisites. This was achieved by training models with a large corpus of data in an unsupervised manner. Different training techniques were employed for training the models but some popular techniques included predicting a word in the given sequence that followed that word. This helped model to understand the structure of the language and form a statistical basis for predicting the probability of a word to appear next. These Language Models are then fine-tuned by training them on specific tasks which has a definite goal.

Every language model has their own final layer which can perform variety of NLP tasks such as text classification, Named Entity Recognition, Question Answering etc. In this work, we experimented with different deep learning techniques like RNN, LSTM, GRU and BERT.

6.1 Recurrent neural network (RNN)

Recurrent Neural Network (RNN)[20] variants became quite common in most NLP researches due to its ability to capture long term dependency among sequences using its internal state memory. Recurrent Neural Networks (RNN) is an extension of Multi-Layer Perceptron (MLP). In NLP, the handling of sequences of data causes particular problems for the MLPs, as it has no way of storing the previous state in a sequence. This is equivalent of trying to understand a written sentence but forgetting each word when reading. Additionally, text is rarely normalised in length and cannot be split up easily while keeping the original information intact. Authors[21] and[22] solved this by feeding the previous hidden state in the network back to the current state. This allows the network to keep a form of internal memory of what has been seen previously in the sequence. This solves two problems such as model dependencies, handling different lengths of inputs. RNNs solved these and quickly became the popular standard for most forms of NLP using Neural Networks.

However, vanishing or "exploding" gradients[23] makes RNNs difficult to train well. Additionally, these models have a tendency to become large, especially in NLP, and therefore take considerable computational resources to train. RNN"s are widely used in applications such as text classification, image captioning and video summarization. The basic RNN takes input from the previous time step and current input to predict the next outcome. The weights are updated for each time step by back-propagation using the error calculated for that time step. RNNs do a great job at extracting features from temporal data but they sometimes face the problem of vanishing gradients. h_0, h_1, h_2, h_3,… h_t are the inputs to the time steps t = 0, 1, 2, 3, …., t which are used along with x_1, x_2, x_3, …., x_t to predict the output y_1, y_2, y_3, …., y_t.

Figure 2: Details out the high-level architecture of RNN.

The RNN output calculation is based on iteratively calculating the output by using the following equations (1) and (2). The equation (1) is used to compute the current hidden state output by using previous hidden state output and current input.

$$h_t = H\left(W_{hx}\, x_t + W_{hh}\; h_{t-1} + b_h\right) \quad (1)$$

$$y_t = W_{hy}\; h_t + b_y \quad (2)$$

In Equations (1) and (2), x_t is the input sequence at the current time slice t, y_t is the output sequence at time slice t, and h represents the hidden vector sequence from time slice 1 to T. W and b represents weight matrices and biases respectively. Lastly, an activation function used for the hidden layer is H. Tanh activation function is used in this work.

6.2. Long short-term memory (LSTM)

One of the complexities of NLP is that text may have future dependencies as well as past ones. Thus takes the form of the meaning of words being determined by context not yet expressed in a sentence. These problems are quite significant in NLP, as humans may read further into a sentence before comprehending the context of a word. Authors[19] tried to solve many of the RNN's problems with the Long Short-Term Memory (LSTM). They used gates in order to update the hidden states in a more efficient manner. LSTMs uses input, forget, cell and output gates. The purpose of the gates is to let the network learn when to update the hidden state, in effect adding a layer of de-noising in the hidden state, making it less likely to update the hidden state with non-important information. The model has been expanded by Authors[25] with additional gates from the original ones. While these gates improve the vanishing and exploding gradient problems associated with regular RNNs, they do not completely solve them.

The basic RNN model however suffered from vanishing gradient[23] problem arising out of continuous multiplication between matrices of weights and cost correction while back propagating. The longer these errors have to propagate in layers they show more tendencies to shrink leaving the model very small values to learn anything from. LSTM (Long Short Term Memory) solve this problem to great extent when a 'forget gate' was added to focus only on the important part of the sequences by creating a connection between activation function of forget gate and the computation of the gradients in the network. LSTM's take longer to train and sometimes suffer from over-fitting issues. They also have high computational cost and are sensitive to the type of parameter initialization techniques used.

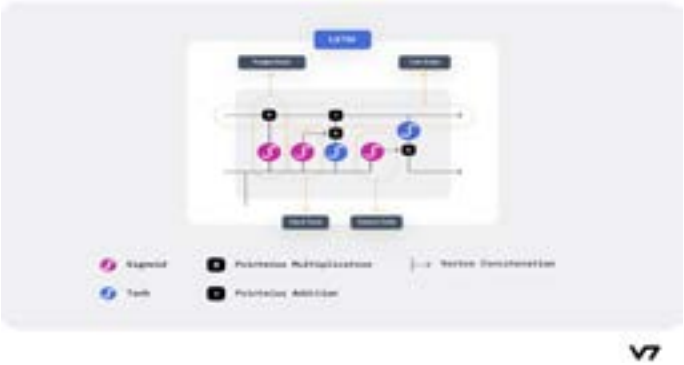

Figure 3: Details out the LSTM Cell and its operations.

The Equation (3) computes the new information that is to be stored in input gate i.

$$i_t = \sigma\left(W_i \,.\, \left[h_{t-1}, x_t\right] + b_i\right) \quad (3)$$

The Equation (4) related to forget gate f which determines the information that is not important for the model to store.

$$f_t = \sigma\left(W_f \,.\, \left[h_{t-1}, x_t\right] + b_f\right) \quad (4)$$

The Equation (5) related to output gate o which provide activation to the final output of the LSTM block.

$$o_t = \sigma\left(W_o \,.\, \left[h_{t-1}, x_t\right] + b_o\right) \quad (5)$$

The Equation (6) is related to internal memory unit which is used to store the previous information.

$$\tilde{C}_t = \tanh\left(W_c . \left[h_{t-1}, x_t\right] + b_c\right) \quad (6)$$

The Equation (7) computes current hidden state output Ct by using the information from previous hidden state and current input.

$$Ct = \sigma\left(f_t * C_{t-1} + i_t * \tilde{C}_t \right) \quad (7)$$

The Equation (8) computes final hidden state output by combining Ct and output gate. This hidden state is a vector representation of the data which is then further used for a variety of applications such as classification and captioning.

$$h_t = o_t * \tanh\left(C_t\right) \quad (8)$$

Where, the activation functions used are sigmoid function (σ) and hyperbolic tangent function (tan h), i_t, f_t, o_t, C_t, $C\tilde{t}$ represents the input gate, forget gate, output gate, memory cell content and new memory cell content, respectively. Three gates are made up of the sigmoid function, and the output of the particular cell is scaled up by using the hyperbolic tangent function. The sigmoid function outputs values between 0 and 1. The sigmoid function determines the percentage of information to be passed through the gate.

6.3 Gated recurrent unit (GRU)

GRU is another type of RNNs with memory cells. They are similar to LSTM but with simpler cell architecture. GRU also has gating mechanism to control the flow of information through cell state but has fewer parameters and does not contain an output gate. The GRU cell structure consists of two gates, r is a reset gate, and z an update gate. The reset gate regulates the flow of new input to the previous memory, and the update gate determines how much of the previous memory to keep. If we compare GRU with LSTM, the update gate is the combination of the input and forget gate and the previous hidden state is connected to the reset gate directly. Another difference is in the exposure of memory content. As GRUs do not have an output gate, it exposes all of its memory content, whereas in LSTM the memory content to be used or seen by other units/cells in the network is managed by the output gate[26].

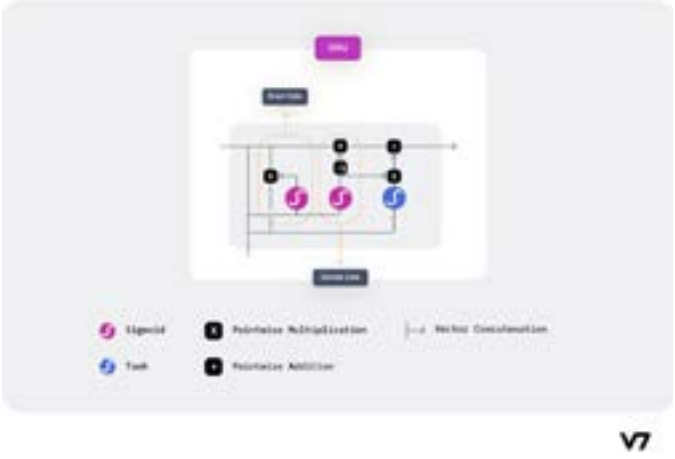

Figure 4: Details out the GRU Cell and its operations.

The gating mechanism in both LSTM and GRU cells makes them the perfect choice for long-term dependencies. Both LSTM and GRU networks performed well and unable to conclude which one was better than other. GRU is less complex and computationally more efficient as compared to LSTM. The following Equations explain the mathematical working of GRU. Equation (9) specifies the Update gate z_t which is computed by using the current input and the previous hidden state output.

$$z_t = \sigma\left(W_z . \left[h_{t-1}, x_t\right]\right) \quad (9)$$

Equation (10) is related to reset gate which determines how much information about past information to forget. Both, update and reset gate use sigmoid functions to constrain the output between 0 and 1.

$$r_t = \sigma\left(W_r . \left[h_{t-1}, x_t\right]\right) \quad (10)$$

In equation (11), $\hat{h}t$ computes the current memory that is to be passed to final output.

$$\tilde{h}_t = \tanh\left(W_h . \left[r_t * h_{t-1}, x_t\right]\right) \quad (11)$$

The equation (12) computes the final memory ht at the time step t.

$$h_t = \left(1 - z_t\right) * h_{t-1} + z_t * h_t \quad (12)$$

6.4 BERT

The landscape of NLP changed with the advent of deep learning and achieved state-of-the-art results in many benchmark tasks. It allowed models to learn hidden patterns among the data and devised a technique to form representation of the text data. Sequential models like RNN suffered from high computational and time costs, CNN who are less sequential in nature suffered in performance when it came to long sentence length. Every next step depends on the output of the previous time step in sequential model which stalls training simultaneously. Transformer reduced the costs by parallelizing computation for every token in a sequence by employing attention mechanism. This allowed training of big models on high volume of corpus. Open GPT[27] and its successor a transformer based model which could work on a wide variety of tasks such as text classification, textual entailment, question answering and semantic similarity assessment. While Open GPT was trained in a unidirectional model, BERT which came later was a bi-directional model and outperformed in most NLP benchmark tasks and achieved high rankings in General Language Understanding Evaluation (GLEU)[28] score.

Many models provide end to end framework which include extracting features from data and downstream NLP tasks. With the development of such frameworks, transfer learning[29] became popular where the learning of these pre-trained frameworks can be transferred onto other applications. This reduced the need for intensive training with high volume of data. With transfer learning pre-trained models could adapt to new dataset with fine-tuning on relatively small data, which not only made possible for individual researchers to get good results but it also reduced the overall cost to generate embedding for these models.

To perform sequence to sequence tasks, the dominant method has some form of recurrent model like recurrent neural networks (RNNs). These methods have been relatively well-performing but suffer from some important drawbacks, where the most glaring is the lack of parallelization of the computational steps[30]. This flaw is due to the sequential nature of a recurring network, where the hidden state is not only dependent on the current input, but also all of the hidden states in the previous time steps.

To be able to calculate a new hidden state, all of the previous hidden states have to be calculated in a sequential manner. While this shortcoming exists for every type of input to the network, it is the most prevalent during long sequence lengths, when memory limitations often reduce the batching possibilities. Authors[30] proposed the use of Transformer which is a solution to solve all the limitations. The Transformer builds upon the concept of attention which provides mechanisms to abandon recurrence altogether in combination with a network structure of encoders and decoders. This combination allows for better task performance and training times when compared to recurrent neural networks.

Deep neural networks (DNNs) have proven to be successful in many NLP tasks, for instance, word embedding extraction with word 2vec and language modelling[31]. One major limitation of a DNN is, however, that they require both the input and target vectors to be of fixed dimensionality. The limitations lead to problems on tasks where the length of the vectors is unknown, which often is the case due to the variability of natural language. One solution to this problem is to use encoders and decoders[32]. The encoder-decoder architecture introduces recurrent neural network (RNN) structures in the encoding layer to encode the input sentence to a vector of fixed length, pass it through the DNN, and subsequently decode it in the decoder layer. This allows the DNN always to be passed a vector of fixed length, regardless of the actual length of the input sequence.

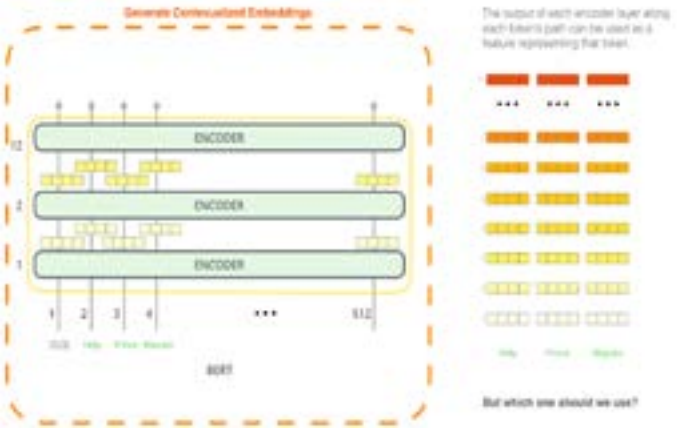

Figure 5: Architecture of BERT and output selection.

The first building block of the encoder-decoder framework is the encoder layer. Multiple recurrent network structures, such as RNNs, gated recurrent units (GRU), or convolutional networks, may be implemented for this purpose. However, Authors[33] showed that multi-layered LSTMs are especially suited as encoders. This is a result of their ability to handle long sequences, unlike for example RNNs that have problems with exploding or vanishing gradients[34]. During run time, the encoder gets its input of a text represented as a sequence of vectors, $x = (x_1, \ldots, x_{tx})$. From this sequence, a context vector, c is generated. This vector serves as contextualized representation of the whole input sequence. The context vector is described in the equation (13).

$$c = q\left(\{h_1, \ldots\ldots, h_{tx}\}\right) \quad (13)$$

The hidden state h, at time t is represented in equation (14).

$$h_t = f\left(x_t, h_{t-1}\right) \quad (14)$$

Where, q and f are recurrent functions, h_t is dependant both on the current input x_t and the former hidden state h_{t-1}, while the context vector c contains all of the previous hidden states. After being processed by the encoder, the context vector will be passed to the decoder, where it will be used to decode an output sequence.

Like the encoder, the decoder can be implemented with many network types, but LSTMs is the most commonly used. The purpose of the decoder is to take the contextualized representation of an input sequence, and generate some output sequence from it. This generation is done sequentially for each word in the output sequence, until an EOS (end-of-sequence) token is produced. The decoding of a sequence is based on the sequence context vector, c, passed from the encoder, and all of the earlier decoder states for the current sequence. The hidden state h of the decoder at time t is expressed in equation (15).

$$h_t = f\left(h_{t-1}, y_{t-1}, c\right) \quad (15)$$

where y_{t-1} is the previous output token and c the context vector. The token y to output is inferred from a conditional distribution and it is represented in equation (16).

$$P\left(y_t \mid y_{t-1}, y_{t-2}, \ldots.y_1, c\right) = g\left(h_t, y_{t-1}, c\right) \quad (16)$$

Where, g is a softmax function providing probable output tokens given the current time step. A softmax function g will then deliver all the possible tokens as a probability distribution. One intuitive way to choose which token to output from this probability distribution is to use an argmax function and pick the token with the highest probability.

Pre-trained language representations, for example, the popular word2vec, has shown to be very useful for a wide array of NLP-tasks. Context2vec manages to capture the context of words, improving on the word2vec model. Authors[35] did refine this approach even further; to capture the context of the words in a sequence, they used an architecture based on language models, named ELMo. ELMo showed much improved results compared to previous methods and is an important predecessor to BERT. Another critical step towards BERT is the Open AI GPT architecture[36]. Open AI GPT (Generative Pre-training Transformer) shares many similarities with ELMo (Embeddings from Language Models), and also incorporates the Transformer instead of recurrent structures.

The system that popularized context-dependent embeddings on a wider scale is ELMo (Embeddings from Language Models), developed by[35]. ELMo builds upon the TagLM architecture. TagLM implements a pre-trained recurrent language model (LM) in conjunction with a word embedding model to compute context embeddings of each word in an input sentence. The purpose of a language model is to predict the next word in a sequence, something that requires information about both the syntactic and semantic roles of words in context. TagLM takes this inherent information about the context of words in the language model to create a LM embedding for each word in the sequence, which is concatenated with the respective word embedding to create the context-dependent embeddings. This, in turn, is provided as input to the sequence tagging model in the TagLM architecture.

GPT (Generative Pre-training Transformer), developed at Open AI by[36], refines the usage of language models for language representations, that showed to be effective in both TagLM and ELMo. However, instead of the recurrent methods to capture the context of words in these architectures, GPT builds on the attention mechanisms found in the Transformer. Additionally, GPT implements what[37] refers to as a" fine-tuning approach" to

pre-trained language representation. The fine-tuning approach builds upon the Universal Language Model Fine-tuning (ULMFiT) method, proposed by[38]. ULMFiT successfully applied transfer learning on a pre-trained language model, allowing a general language model to be fine-tuned with domain-specific data, for a specific task. In other words, the pre-trained language model parameters are updated and changed to fit a specific downstream task.

The most known implementation of said transformer networks, namely BERT (Bidirectional Encoder Representations from Transformers) further improved upon the results of the earlier mentioned architectures. It was subsequently implemented as a part of the Google search engine, with an estimated improvement of the search queries by around 10%[39]. BERT pre-trains a language model with a technique using bi-directional transformers, which means that the BERT model can capture the context of words in all directions, in all layers, resulting in what[37] call deep bidirectional representations. The BERT model can also be fine-tuned for several downstream tasks, in the same manner as GPT.

7. Experimental Results

In this work, the experiment conducted with various deep learning techniques like RNN, LSTM, GRU and BERT for suggestion mining. The machine learning techniques used the feature vectors to generate the classification model. We observed that it was difficult to identify the suitable features that differentiate the sentences into suggestion or non-suggestion sentences. The deep learning techniques automatically extract the features which are suitable for classifying the sentences into suggestion sentence or not. The word embeddings play an important role in the process of deep learning classification process. In this work, word 2vec word embedding technique is used to generate the word vectors. The hyper parameters used in the deep learning models show a great difference in the accuracies of classification.

In this work, the deep learning models used different types of hyper parameters such as activation function = sigma and tanh function, epochs count = 10, dropout rate = 30%, neurons count in hidden layer = 1000, back-propagation technique = ADAM optimizer, Embedding Vector Size = 300, hidden layers count = 3, learning rate = 0.001, maximum sequence length = 1000, transformer layers = 24, self-attention heads = 20, hidden vector size = 768. The accuracies of suggestion mining are displayed in **Table 2**.

Table 2: The accuracies of deep learning techniques for suggestion mining.

Deep learning Technique	Accuracy
RNN	82.71
LSTM	84.37
GRU	84.65
BERT	87.29

The **Table 2** shows the accuracies of suggestion mining when experimented with different deep learning techniques such as RNN, LSTM, GRU and BERT. The BERT technique attained good accuracy of 87.29 for suggestion mining. It was observed that the BERT performance is good when compared with other deep learning techniques like RNN, LSTM and GRU. The LSTM and GRU show equal performance for suggestion mining.

8. Conclusion and Future Scope

In recent times, people are using internet for communication and sharing their opinions on products, services, places, people etc. The people opinions and experiences are more helpful to others to know about different types of entities. Sentiment analysis is one research are which extracts the positive, negative or neutral sentiment on the entities from textual reviews. Suggestion mining is one type of sentiment analysis which extracts the suggestions, wishes, tips or recommendation sentences in the text which are helpful for the companies to improve the quality of their entities. The researchers proposed various solutions to suggestion mining based on machine learning and deep learning techniques. In this work, the experiment conducted with different deep learning techniques like RNN, LSTM, GRU and BERT. The BERT technique shows good accuracy of 87.29% when compared with other techniques.

In future work, we are planning to increase the size of pre-trained vectors as well as increase the size of the training dataset. We are also planning to implement imbalance techniques to solve the problems in imbalance in training dataset.

9. References

1. Sapna Negi, Paul Buitelaar. Inducing distant supervision in suggestion mining through part-of-speech embeddings. arXiv preprint, 2017.
2. Sapna Negi, Kartik Asooja, Shubham Mehrotra, et al. A study of suggestions in opinionated texts and their automatic detection. In: Proceedings of the Fifth Joint Conference on Lexical and Computational Semantics, ACL Anthology, 2016;170-178.
3. Sapna Negi, Paul Buitelaar. Towards the extraction of customer-to-customer suggestions from reviews. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, ACL Anthology, 2015;2159-2167.
4. Caroline Brun, Caroline Hagege. Suggestion mining: Detecting suggestions for improvement in users' comments. Research in Computing Science, 2013;70: 5379-5362.
5. Amar Viswanathan, Prasanna Venkatesh, Bintu G Vasudevan, et al. Suggestion mining from customer reviews. In AMCIS, 2011.
6. Mikolov T, Sutskever I, Chen K, et al. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems, 2013;3111-3119.
7. Sapna Negi, Maarten de Rijke, Paul Buitelaar. Open domain suggestion mining: Problem definition and datasets. arXiv preprint, 2018.
8. Sapna Negi, Tobias Daudert, Paul Buitelaar. Semeval-2019 task 9: Suggestion mining from online reviews and forums. In: Proceedings of the 13th international workshop on semantic evaluation, 2019;877-887.
9. Mateusz Klimaszewski, Piotr Andruszkiewicz. "WUT at SemEval-2019 Task 9: Domain-adversarial neural networks for domain adaptation in suggestion mining". In: Proceedings of the 13th international workshop on semantic evaluation (SemEval-2019), Minneapolis, Minnesota, USA, 2019;1262-1266.
10. Usman Ahmed, Humera Liaquat, Luqman Ahmed, et al. "Suggestion Miner at SemEval-2019 Task 9: Suggestion Detection in Online Forum usingWord Graph". In: Proceedings of the 13th international workshop on semantic evaluation (SemEval-2019), Minneapolis, Minnesota, USA, 2019;1242-1246.
11. Nelleke Oostdijk , Hans van Halteren. "Team Taurus at SemEval-2019 Task 9: Expert-informed pattern recognition

for suggestion mining". Proceedings of the 13th international workshop on semantic evaluation (SemEval-2019), Minneapolis, Minnesota, USA, 2019;1247-1253.

12. Janardhanan Ramanand, Krishna Bhavsar, and Niranjan Pedanekar. 2010. Wishful thinking - finding suggestions and 'buy'wishes from product reviews. In: Proceedings of the NAACL HLT 2010 workshop on computational approaches to analysis and generation of emotion in text, Association for Computational Linguistics, 2010;54-61.
13. Amar Viswanathan, Prasanna Venkatesh, Bintu G Vasudevan, et al. Suggestion mining from customer reviews. In AMCIS, 2011.
14. Rajalakshmi S, Angel Deborah S, S Milton Rajendram, et al. "SSN-SPARKS at SemEval-2019 Task 9: Mining Suggestions from Online Reviews using Deep Learning Techniques on Augmented Data". In: Proceedings of the 13th international workshop on semantic evaluation (SemEval-2019), Minneapolis, Minnesota, USA, 2019;1237-1241.
15. Tirana Noor Fatyanosa, Al Hafiz Akbar Maulana Siagian, Masayoshi Aritsugi. "DBMS-KU at SemEval-2019 Task9: Exploring Machine Learning Approaches in Classifying Text as Suggestion or Non-Suggestion". In: Proceedings of the 13th international workshop on semantic evaluation (SemEval-2019), Minneapolis, Minnesota, USA, 2019;1185-1191.
16. Aysu Ezen-Can, Ethem F Can. "Hybrid RNN at SemEval-2019 Task 9: Blending information sources for domain-independent suggestion mining". IN: Proceedings of the 13th international workshop on semantic evaluation (SemEval-2019), Minneapolis, Minnesota, 2019;1199-1203.
17. Jurafsky D, Martin JH. Speech and language processing (3rd edition draft). Prentice-Hall, 2019.
18. Melamud O, Goldberger J, Dagan I. Context2vec: Learning generic context embedding with bidirectional LSTM. In: Proceedings of the 20th SIGNLL conference on computational natural language learning, 2016;51-61.
19. Hochreiter S, Schmidhuber J. Long short-term memory. Neural Comput, 1997;9: 1735-1780.
20. Mikolov T, Zweig G. Context dependent recurrent neural network language model. IEE, 2012; 234-239.
21. John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities". Proc Natl Acad Sci U S A, 1982;79: 2554-2558.
22. David E Rumelhart, Geoffrey E Hinton, Ronald J Williams. "Learning representations by back-propagating errors". In: Cognitive modelling, 1986;523: 533-536.
23. Hochreiter S. The vanishing gradient problem during learning recurrent neural nets and problem solutions. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 1998;6: 107-116.
24. Felix A Gers, Jürgen Schmidhuber, Fred Cummins, et al. "Learning to forget: Continual prediction with LSTM". In: Proc ICANN'99 Int. Conf. on Artifical Neural Networks,1999;850-855.
25. Klaus Greff. "LSTM: A Search Space Odyssey". In: IEEE Transactions on Neural Networks and Learning Systems, 2017;21622388.
26. J Chung, C Gulcehre, K Cho. Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv1412.3555, 2014.
27. Radford A, Wu J, Child R. Language models are unsupervised multitask learners.
28. Wang A, Singh A, Michael J, et al. GLUE: A multi-task benchmark and analysis platform for natural language understanding. 2019.
29. Pan SJ, Yang Q. A Survey on transfer learning. IEEE transactions on knowledge and data engineering, 2010;22: 1345-1359.
30. Vaswani A, Shazeer N, Parmar N, et al. Attention is all you need. Advances in Neural Information Processing Systems, 2017:30: 5998-6008.
31. Bengio Y, Ducharme R, Vincent P, et al. A neural probabilistic language model. J Mach Learn Res, 2003;3: 1137-1155.
32. Cho K, van Merrienboer B, Gulcehre C, et al. Learning phrase representations using rnn encoder-decoder for statistical machine translation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), 2014: 1724-1734.
33. Sutskever I, Vinyals O, Le QV. Sequence to sequence learning with neural networks. Proceedings of the 27th International Conference on Neural Information Processing Systems, 2: 3104-3112.
34. Bengio Y, Simard P, Frasconi P. Learning long-term dependencies with gradient descent is difficult. IEEE Transactions on Neural Networks, 1994;5: 157-166.
35. Peters M, Neumann M, Iyyer M, et al. Deep contextualized word representations. Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 1: 2227-2237.
36. Radford A, Narasimhan K, Salimans T, et al. Improving language understanding by generative pre-training. Technical report. OpenAI, 2018.
37. Devlin J, Chang MW, Lee K, et al. BERT: Pre-training of deep bidirectional ransformers for language understanding. Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, 2019;1: 4171-4186.
38. Howard J, Ruder S. Universal language model fine-tuning for text classification. Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 1: 328-339.
39. Rogers A, Kovaleva O, Rumshisky A. A primer in bertology: What we know about how bert works. arXiv,2002.