A Comparative Analysis of TF-IDF and Skip-Gram Word Embeddings for QA Text Classification using Deep Learning Architectures

Amir Sakib Saad
Department of Computer Science and Engineering
BRAC University, Dhaka
Dhaka, Bangladesh
amir.sakib.saad@g.bracu.ac.bd

Asef Ahmed Shimanto
Department of Computer Science and Engineering
BRAC University, Dhaka
Dhaka, Bangladesh
asef.ahmed.shimanto@g.bracu.ac.bd

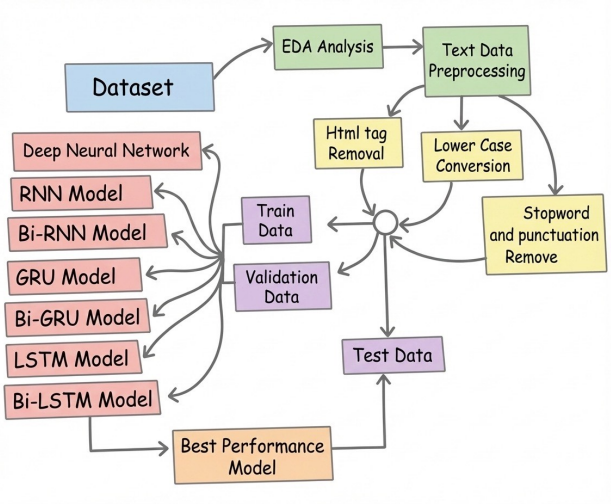
Faiyaz Zaman Saadman
Department of Computer Science and Engineering
BRAC University, Dhaka
Dhaka, Bangladesh
shah.faiyaz.zaman@g.bracu.ac.bd

***Abstract*—In this venture, diverse human language handling (NLP) approaches for automated script sorting within a question-and-answer (QA) structure are comprehensively weighed. The efficiency of two distinct content description strategies is examined in this endeavor: prediction-based Word2Vec (Skip-Gram) embeddings and frequency-based TF-IDF (Term Frequency-Inverse Document Frequency). Numerous machine learning and deep learning designs, including multinomial naive Bayesian, deep neural networks (DNN), and various recurrent neural networks (RNN) like simpleRNN, long short-term memory (LSTM), and gated recurrent units (GRU), were combined with these vectorization schemes. Sentiment charting and n-gram illustration are employed in exploratory data scrutiny (EDA) after a meticulous prose preparation phase that incorporates stop word straining, HTML tag extraction, and lowercase conversion. To establish how chronological setting influenced classification precision, performance was appraised on both unidirectional and bidirectional variations of the recurrent models. Confusion matrices and exhaustive classification summaries are used to display experimental figures and demonstrate how computational intricacy and forecasted accuracy are linked. In order to boost the consistency of automated quality assurance categorization setups, this inquiry offers perspectives on the ideal word embedding pairings and model layouts.**

***Index Terms*—RNN, LSTM, GRU, NLP, DNN, EDA, TF-IDF**

I. INTRODUCTION

The relationship between artificial intelligence (AI) and healthcare has emerged as a key area for enhancing information accessibility and accuracy in the current digital era. Medical QA (Question and Answer) systems, which offer prompt responses to complicated queries, are vital resources for both the general public and healthcare professionals. The creation and comparison of several natural language processing (NLP) models intended to efficiently categorize medical literature is the main goal of this research. This study aims to determine the best reliable techniques for comprehending and categorizing medical conversation using a range of structured approaches, from conventional statistical techniques to sophisticated deep learning frameworks. The meticulous data preparation procedure is the project's cornerstone. Words like HTML tags, punctuation, and common stop words that don't add to the semantic meaning of clinical questions are frequently seen in raw medical language. The project addresses this by implementing a custom cleanup function that eliminates non-alphabetic letters, filters out non-informative linguistic parts, and normalizes the text to lowercase. This stage is critical

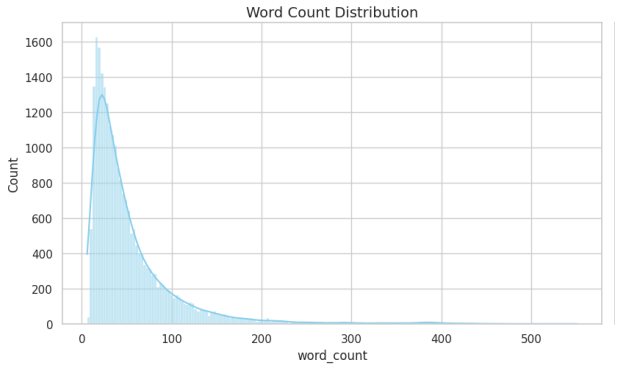


because the quality and cleanliness of the input sequence have a significant impact on the performance of complicated neural networks like supervised recurrent units (GRU) and long-term memory (LSTM). Feature engineering and text representation have been extensively studied. Word2Vec (Skip-gram) and TF-

IDF (Inverse Term Frequency-Document Frequency) are the two primary methods that have been compared. The statistical significance of words in a corpus is extracted using TF-IDF, which serves as the foundation for conventional machine learning models such dense neural networks and basic Bayesian polynomials. On the other hand, dense vector embeddings are produced using Word2Vec's Skip-gram implementation. In medical language, where the closeness and link between terms like "symptom," "diagnosis," and "treatment" are crucial, these embeddings enable models to capture semantic and contextual relationships. Lastly, the project uses descriptive performance measures to highlight empirical validation. The study used descriptive confusion matrices, precision-recall curves, and F1 scores in addition to basic accuracy to assess the models' performance. This study offers a thorough understanding of how vectorization strategies impact the outcomes by contrasting a TF-IDF-based dense network with a Skip-Gram-based deep neural network (DNN). In the end, our effort advances the overarching objective of creating more intelligent, pertinent systems that can comprehend the subtleties of medical terminology to deliver trustworthy information.

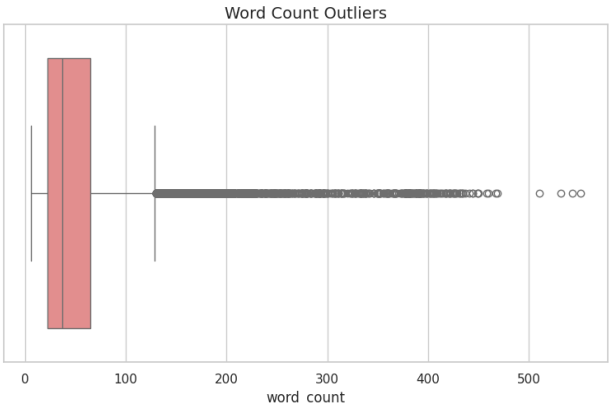
II. METHODOLOGY

Exploratory data analysis, normally known as arsenic EDA, is an important initial measure in data analysis, where the basic structure, form, and features of the data are fully understood. This measure measures the accuracy, completeness, and suitability of the data for analysis rather than building an angstrom model from the data. EDA helps identify whether the dataset has missing hour angle values, outliers, or abnormal values. It helps determine the distribution of the data. In addition, the correlation between different features is analyzed, which is necessary for the approaching feature choice and model development. Descriptive statistics and assorted visual image techniques, such as arsenic scatter plots, box plots, and histograms, are normally used to complete EDA. When EDA is performed properly, there are fewer misconceptions about the data, and as a consequence, machine learning or deep learning models can supply more accurate consequences.



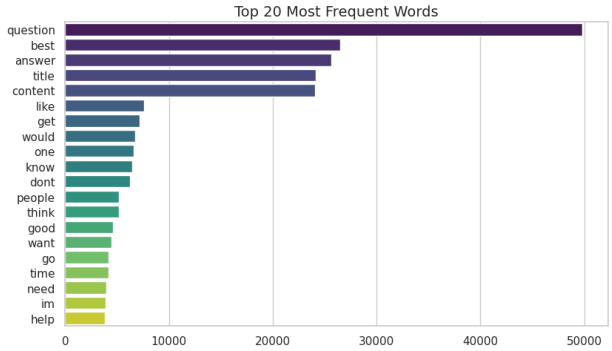
This distribution is strongly right-skewed. Positively skews distribution. This means that most of the data values in the dataset are concentrated on the left side, i.e., towards lower berth values. We can see that the number of words in most texts or documents is expressed as 20 to 60. The high extreme, 'Peak,' of the graph bespeaks that the number of texts with an angstrom length of 20-25 words is high (about 1600 More). This is named the manner in which statistics.

The box plot entitled word count outlier exemplifies an angstrom distribution that is heavily skewed to the right, bespeaking that the majority of the data points are concentrated at lower berth values, while a significant number of extreme observations pull the tail toward the higher end. The central box, which represents the interquartile scope where the center 50% of the data reside, is positioned approximately between 20 and 70 words, with the angstrom median line appearance approaching the 40-word mark. The box plot titled "Word



Count Outliers" illustrates a distribution that is heavily skewed to the right, indicating that the majority of the data points are concentrated at lower values while a significant number of extreme observations pull the tail toward the higher end. The central box, which represents the interquartile range where the middle 50% of the data resides, is positioned roughly between 20 and 70 words, with a median line appearing near the 40-word mark. The relatively short left whisker suggests a firm floor for the minimum word count, whereas the right whisker extends to approximately 130 words before transitioning into a dense and persistent trail of outliers. These outliers, depicted as individual circles, extend far beyond the typical range to reach a maximum of over 550 words. This visual pattern reveals that while the dataset primarily consists of short-form content, it also contains a substantial volume of "long-tail" entries that are statistically distinct from the core population and may require specialized handling, such as truncation or normalization, during data preprocessing.

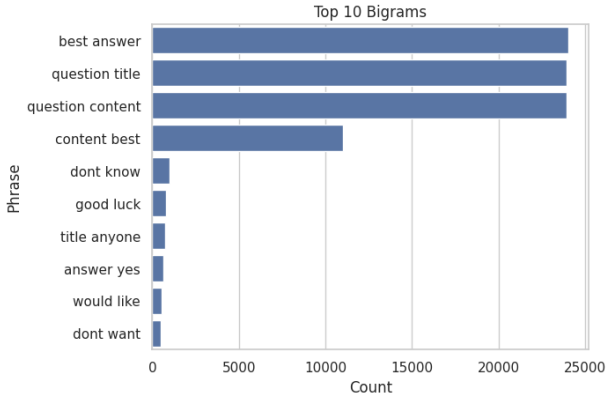
The horizontal bar chart titled "Top 20 Most Frequent Words" illustrates the dominance of specific terms within a text dataset, revealing a clear hierarchical structure where a small group of words occurs significantly more often than the rest. The word "question" stands as the most prominent outlier, appearing approximately 50,000 times, which is nearly double the frequency of the next most common words, "best" and "answer," both of which hover around the 25,000 to 27,000 range alongside "title" and "content." This initial cluster of



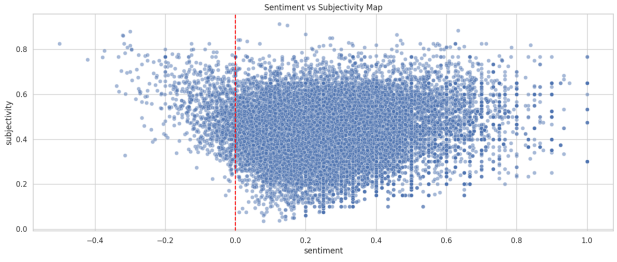
five words suggests the dataset is likely sourced from a Q&A platform or an inquiry-based community where structural elements of a post are frequently mentioned. Following this top tier, there is a sharp drop in frequency to a secondary group of words like ”like,” ”get,” and ”would,” which appear fewer than 10,000 times each. As the list descends toward ”help” at the bottom of the top 20, the frequencies become much more uniform and stable, consisting largely of common verbs and pronouns that characterize natural conversational language. The color gradient—shifting from deep purple for high-frequency terms to bright yellow for lower ones—further emphasizes this steep decline in word usage, highlighting how a few specialized terms define the core thematic focus of the entire corpus.

The provided word cloud serves as a visual representation of text data, where the physical size of each word directly correlates with its frequency or importance within the underlying dataset. Dominant terms such as question, answer, content, title, and best are prominently displayed in the largest fonts, suggesting that the source material likely originates from an interactive information-sharing environment like a QA forum, a customer support database, or a knowledge management system. The high density of words like ”people,” ”friend,” ”life,” and ”know” points toward a focus on human interaction and social exchange, while secondary terms like ”work,” ”time,” ”year,” and ”need” add a layer of practical or task-oriented context. By consolidating these keywords into a single visualization, the cloud highlights a discourse centered on seeking solutions, evaluating the quality of information, and addressing everyday inquiries or personal experiences. The lack of highly specialized technical jargon suggests the content is general-purpose and accessible, emphasizing utility and collaborative problem-solving through the repetition of verbs like ”make,” ”think,” and ”find.”

This horizontal bar chart, titled ”Top 10 Bigrams,” illustrates the frequency of two-word combinations within a dataset, highlighting a strong focus on information exchange and content structure. The three most frequent bigrams—best answer, question title, and question content—all share a nearly identical, dominant count exceeding 20,000 occurrences, which strongly suggests that the data is sourced from a structured QA platform or a technical support forum where users categorize their posts into titles and bodies while seeking validated solutions. These are followed by ”content best,” which appears approximately half as often, likely indicating a recurring theme of evaluating the quality of information.



The remaining bigrams, such as ”dont know,” ”good luck,” ”answer yes,” and ”would like,” appear with significantly lower frequency, representing common conversational fillers or expressions of intent and politeness within the user interactions. Overall, the significant gap between the top four phrases and the rest of the list underscores a highly standardized language pattern revolving around the formal components of a question-and-answer ecosystem.



This scatter plot, titled Sentiment vs Subjectivity Map, visualizes the emotional tone and personal nature of a dataset by plotting individual data points along two axes. The horizontal axis represents sentiment, ranging from -0.5 (negative) to 1.0 (positive), while the vertical axis measures subjectivity from 0.0 (objective/factual) to nearly 1.0 (subjective/opinion-based). A clear observation is the high density of points concentrated

to the right of the red dashed line at 0.0, indicating that the overall sentiment of the discourse is predominantly positive. Furthermore, the bulk of the data is clustered between 0.3 and 0.7 on the subjectivity scale, suggesting that the content is largely composed of personal opinions, experiences, or qualitative assessments rather than purely clinical or objective facts. The wide spread of points, particularly the ”fanning out” effect as sentiment becomes more positive, implies that users express a diverse range of subjective views when they are happy or satisfied, whereas negative sentiments appear slightly more sparse and less varied in their subjectivity. This distribution reinforces the idea of a social or community-driven platform where helpfulness and positive engagement are the norm, aligning with the ”best answer” and ”good luck” themes seen in previous charts.

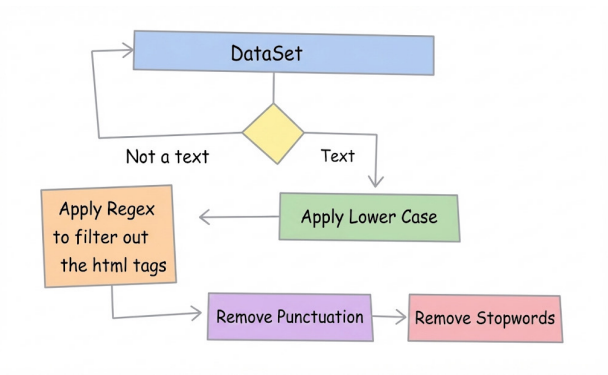
	QA Text	Class	char_count	word_count	sentiment	subjectivity
0	question title el chocolate es lo mas rico per...	Society & Culture	190	32	0.297917	0.514583
1	question title inlaws called momdad first name...	Family & Relationships	339	52	0.360417	0.406250
2	question title personals yahoo page colored pi...	Business & Finance	127	20	0.287500	0.434722
3	question title many african leaders foreign ac...	Politics & Government	174	25	0.186111	0.413889
4	question title get burned letter scar show way...	Health	260	40	0.606667	0.581111
...	...	...	...	...	...	...
23906	question title download driver bt voger 105 ma...	Computers & Internet	220	24	0.700000	0.566667
23907	question title guys correct etiquette deal que...	Society & Culture	461	67	0.137500	0.368750
23908	question title communicate boyfriends mom ques...	Family & Relationships	554	87	0.211111	0.562054
23909	question title language would like learn quest...	Education & Reference	336	48	0.144048	0.197024
23910	question title find wwwquikpaydaycom web quest...	Business & Finance	235	40	0.625000	0.625000

Sentiment (Polarity) Sentiment analysis usually measures ”polarity”—whether a piece of text is positive, negative, or neutral.

The algorithm compares the words in your QA Text against a pre-defined dictionary where words have assigned scores. For example, ”excellent” might be +0.8, while ”terrible” is 0.8. The Formula: The final score is often the average of the scores of all sentiment-bearing words in the string. It ranges from -1.0 (very negative) to 1.0 (very positive). A score of 0 is neutral. To understand if users are frustrated (negative sentiment in ”Health” or ”Computers”) or happy. To see if certain topics (e.g., ”Politics Government”) trigger more hostile interactions compared to others. Subjectivity quantifies the amount of personal opinion, emotion, or judgment versus factual information in the text. the algorithm looks for ”subjective words” (adjectives like beautiful, annoying, great) versus ”objective words” (nouns and factual verbs). It ranges from 0.0 to 1.0. 0.0: Very objective (factual, like a news report). 1.0: Very subjective (pure opinion or emotion).In categories like ”Education Reference,” you might want low subjectivity (facts). In ”Family Relationships,” you expect high subjectivity (personal stories).It helps distinguish between a user asking for a technical fix (objective) and a user venting about a bad experience (subjective).

Textual data preprocessing is the process of cleaning and ”standardizing” raw text so that machine learning algorithms can understand it. Because computers are great at math but poor at understanding language nuances, we must transform messy, human-written text into a structured format. However there can be diverse way to preprocessed the textual data. But here we use a customized pipeline that handles the text of the focused dataset.

NLP Preprocessing Workflow Formalization



The function $f(text)$ transforms a raw input string into a cleaned format through a series of mathematical and logical operations.

1. Input Validation

The function first ensures the input x is a member of the set of strings S. If $x \notin S$, the function returns the identity:

$$f(x) = \{\ ProceedtoStep2if x \in S xif x \notin S$$

2. Normalization and Noise Removal

Let T_{raw} be the input string. We define the following operations:

- **Case Folding:** $T_1 = \{c.lower() \mid \forall c \in T_{raw}\}$
- **Regex Cleaning:** $T_2 = sub(pattern = \langle[^{>}]+\rangle, repl = \epsilon, T_1)$
- **Whitespace Handling:** $T_3 = replace(`n', '', T_2)$

3. Punctuation Stripping

Let P be the set of all punctuation characters defined in `string.punctuation`. The transformation is:

$$T_4 = \{c \mid c \in T_3, c \notin P\}$$

4. Tokenization and Stop Word Filtering

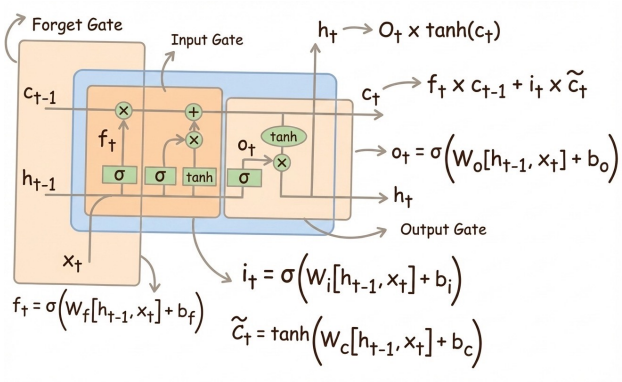
The string is split into a sequence of tokens $W = \{w_1, w_2, \ldots, w_n\}$. Given a set of stop words S, the filtered sequence W_{clean} is defined as:

$$W_{clean} = \{w_i \in W \mid w_i \notin S\}$$

5. Re-joining

The final output T_{final} is the concatenation $(\oplus)$ of the elements in W_{clean} separated by a space character σ:

$$T_{final} = \bigoplus_{i=1}^{|W_{clean}|} (w_i + \sigma)$$



MODEL A: LSTM ARCHITECTURE

1) **Forget Gate (f_t):** Decides what to remove from the cell state.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f)$$

2) **Input Gate (i_t):** Decides what new information to store.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i)$$

Candidate Update:

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] + b_C)$$

3) **Cell State Update:** The old state is multiplied by the forget factor, and the new candidate memory is added.

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

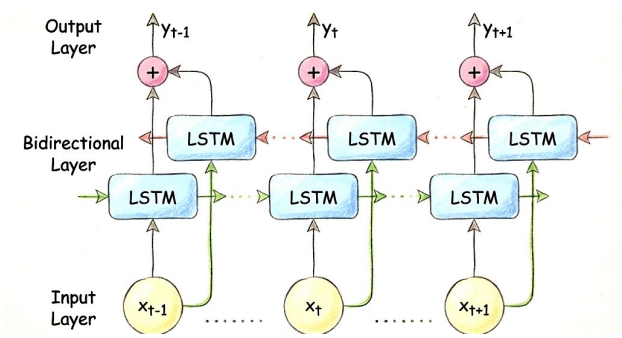
4) **Output Gate (o_t):** Decides what to output as the hidden state based on the filtered cell state.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

Final Hidden State:

$$h_t = o_t * \tanh(C_t)$$

MODEL B: BI-DIRECTIONAL LSTM ARCHITECTURE

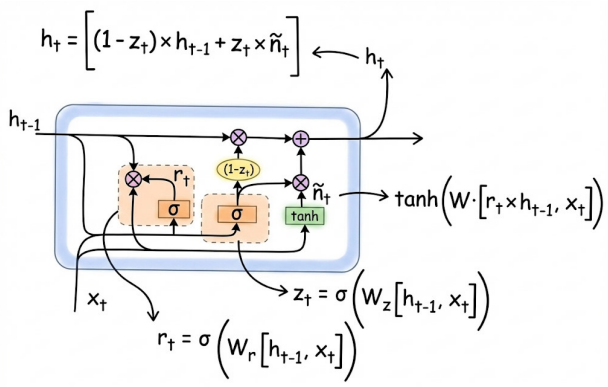


- **Structure:** It utilizes two parallel LSTM tracks: one processing the sequence forward and one backward.
- **Dataflow:** The input x_t is sent to both LSTMs. The final hidden state h_t is the concatenation of the forward LSTM output $\vec{h}_t$ and backward LSTM output $\overleftarrow{h}_t$:

$$h_t = [\vec{h}_t \parallel \overleftarrow{h}_t]$$

- **Key Advantage:** It captures context from both the past and the future, making it highly effective for complex NLP tasks prior to the rise of Transformers.

MODEL C: GRU ARCHITECTURE



1) **Reset Gate (r_t):** Determines how much of the past information to forget.

$$r_t = \sigma(W_r \cdot [h_{t-1}, x_t])$$

2) **Update Gate (z_t):** Determines how much of the past information to pass along to the future.

$$z_t = \sigma(W_z \cdot [h_{t-1}, x_t])$$

3) **Current Memory Content ($\tilde{h}_t$):**

$$\tilde{h}_t = \tanh(W \cdot [r_t * h_{t-1}, x_t])$$

4) **Final State Update:**

$$h_t = (1 - z_t) * h_{t-1} + z_t * \tilde{h}_t$$

MODEL D: BI-DIRECTIONAL GRU ARCHITECTURE

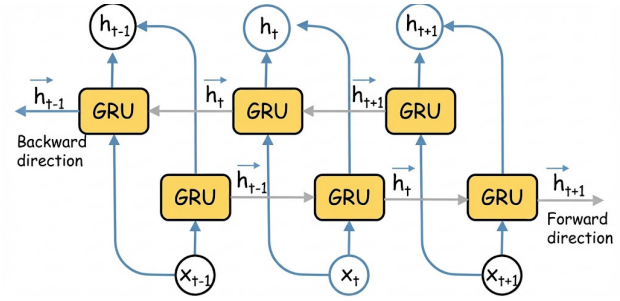
- **Concept:** Applies the bidirectional processing strategy to GRU cells to capture dependencies from both past and future time steps.
- **Architecture:**
  - **Structure:** Two parallel GRU layers operating in opposite directions.
  - **Dataflow:** The forward hidden state $\vec{h}_t$ and backward hidden state $\overleftarrow{h}_t$ are updated independently and then concatenated:

$$\vec{h}_t = GRU_{fwd}(x_t, \vec{h}_{t-1})$$

$$\overleftarrow{h}_t = GRU_{bwd}(x_t, \overleftarrow{h}_{t+1})$$

$$h_t = [\vec{h}_t \parallel \overleftarrow{h}_t]$$

- **Performance:** It offers a balance between the high performance of Bi-LSTMs and the computational efficiency of standard RNNs. It is often faster to train than Bi-LSTM with comparable accuracy for smaller datasets.



MODEL E: RNN ARCHITECTURE

Step-by-Step Data Processing

1) **Input:** At time t, the network receives input x_t and the hidden state from the previous step h_{t-1}.
2) **Combine:** These two vectors are concatenated or combined using matrix multiplication.
3) **Activation:** The combined data is passed through an activation function (usually $\tanh$) to squash values between -1 and 1.
4) **Output:** This produces the new hidden state h_t, which serves as the memory for the next step and (optionally) produces an output y_t.

Mathematical Formulation

- **Update Rule:**

$$h_t = \tanh(W_h \cdot h_{t-1} + W_x \cdot x_t + b)$$

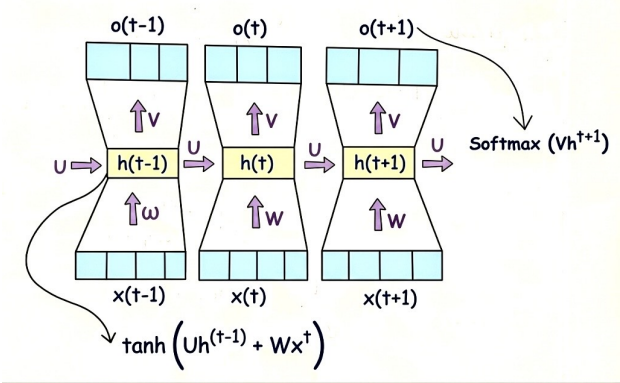
- **Output Rule:**

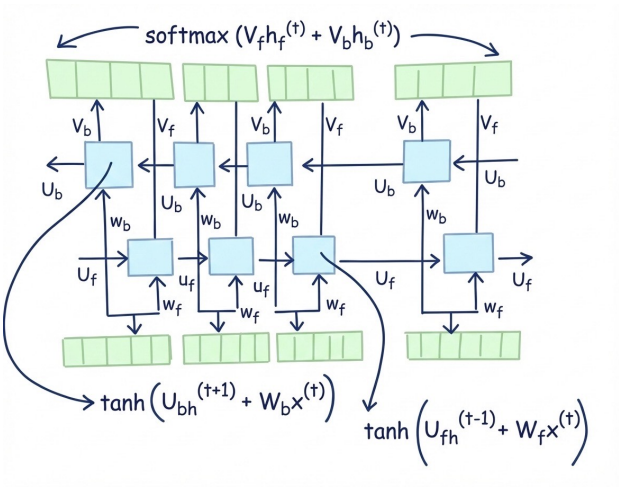
$$y_t = W_y \cdot h_t + b_y$$

(Where W are weight matrices and b are bias vectors)

Limitations

- **Vanishing Gradient:** As sequences get longer (e.g., > 10 steps), the gradients used for learning shrink exponentially, making it impossible for the network to learn dependencies between distant time steps.





MODEL E: BI-DIRECTIONAL RNN ARCHITECTURE

Step-by-Step Data Processing

1) **Forward Pass:** One RNN layer calculates hidden states from $t = 1$ to $t = N$.

$$\vec{h}_t = \sigma(W_{\vec{h}} x_t + V_{\vec{h}} \vec{h}_{t-1} + b_{\vec{h}})$$

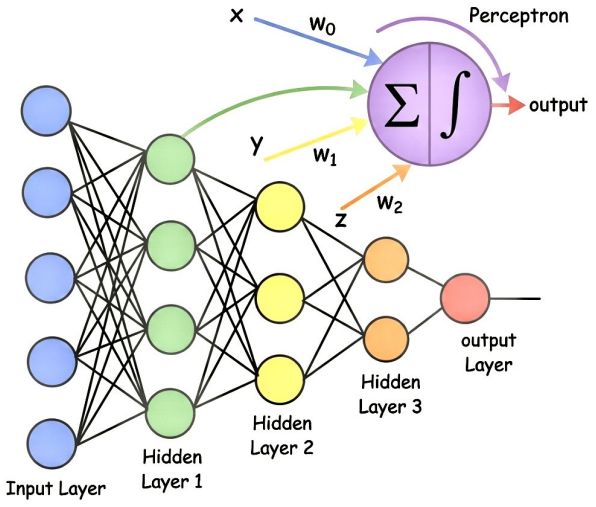
2) **Backward Pass:** A second RNN layer calculates hidden states from $t = N$ to $t = 1$.

$$\overleftarrow{h}_t = \sigma(W_{\overleftarrow{h}} x_t + V_{\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}})$$

3) **Integration:** At every time step t, the final output is generated by combining the hidden state from the forward layer ($\vec{h}_t$) and the backward layer ($\overleftarrow{h}_t$).

$$y_t = g(W_y[\vec{h}_t \parallel \overleftarrow{h}_t] + b_y)$$

MODEL E: DEEP NEURAL NETWORK ARCHITECTURE



Plaintext Structure

$$[Input] \rightarrow [Hidden1] \rightarrow [Hidden2] \rightarrow \ldots \rightarrow [Output]$$

$$(x) \rightarrow (h_1) \rightarrow (h_2) \rightarrow (y)$$

Detailed Dataflow

1) **Input:** Features are fed into the input layer as a flat vector x.
2) **Weighted Sum:** Each neuron in Layer 1 calculates a weighted sum of inputs plus a bias:

$$z = Wx + b$$

3) **Activation:** The sum is passed through a non-linear activation function (e.g., ReLU, Sigmoid):

$$a = \sigma(z)$$

4) **Propagation:** This output a becomes the input for the next layer. This process repeats until reaching the final output layer.

REFERENCES

[1] Kim, D., Yokota, T., Suzuki, T., Lee, S., Woo, T., Yukita, W., Koizumi, M., Tachibana, Y., Yawo, H., Onodera, H., Sekino, M., Someya, T. (2020). Ultraflexible organic light-emitting diodes for optogenetic nerve stimulation. Proceedings of the National Academy of Sciences of the United States of America, 117(35), 21138–21146. https://www.jstor.org/stable/26968782

[2] Wickert, A. D. (2014). The ALog: Inexpensive, Open-Source, Automated Data Collection in the Field. Bulletin of the Ecological Society of America, 95(2), 166–176. http://www.jstor.org/stable/bullecosociamer.95.2.166