Unleashing the potential: harnessing generative artificial intelligence for empowering model training

Alexandra-Mihaela DUMITRU
Bucharest University of Economic Studies, Bucharest, Romania
Dumitrualexandra18@stud.ase.ro

Sorin ANAGNOSTE
Bucharest University of Economic Studies, Bucharest, Romania
Sorin.Anagnoste@fabiz.ase.ro

Marco SAVASTANO
Sapienza University of Rome, Rome, Italy
Marco.Savastano@uniroma1.it

Abstract. *Recent strides in generative artificial intelligence, particularly large language models, have been propelled by foundation models – learning algorithms trained on extensive and diverse datasets encompassing various subjects. This technology, inspired by the complexity of the human brain, unveils a new frontier in generative Artificial Intelligence (AI), showing its potential in creativity by generating innovative content based on absorbed data and user prompts. It is forecasted that the conversational AI and virtual assistant segment is experiencing the highest growth rate within the contact center industry, projected to fuel a 24% increase in the market during 2024. In spite of all remarkable performances, the incipient stage of generative AI calls for a careful consideration, as technological and ethical challenges demand attention and awareness. This research delves into the base principle which empowers users to build personalized chatbots trained on your data. This stand-alone footprint can further exemplify the transformative potential of generative artificial intelligence, extending its reach beyond professionals to individuals and tremendously remodeling the landscape of chatbots.*
Text generation lies at the intersection of computational linguistics and artificial intelligence, forming a specialized area within natural language processing. It implies a thorough procedure where a model is trained to be able to recognize and interpret the context of specific input data, subsequently generating text that pertains to the input's subject matter.
We have identified gap areas that require in-depth research. For instance, a broader number of papers relies solely on architecture optimization, performance comparison or application-specific studies. Therefore, this paper gives a bird's eye view of the effective algorithm flow of a traditional generative model, using Long Short-Term Memory networks – part of the recurrent neural networks part family. The purpose of the current study focuses to enrich the existing body of knowledge on how a response generation-based model operates, therefore paving the way for chatbots development and deployment.

Keywords: generative artificial intelligence, large language models, personalized chatbots, data, ethical challenges, text generation, natural language processing, long short-term memory.

Introduction

Generative AI (Gen AI) is set to profoundly influence all areas of the business world. A tremendous proof derives from the research findings of Chui, M et al (2023), which indicates that generative AI could potentially add an annual value of $2.6 trillion to $4.4 trillion across the 63 applications they have examined. Numbers speak for themselves, so the models that have been continuously deployed all over the markets since November 2022 outbreak. Over the years, the capabilities of

DOI: 10.2478/picbe-2024-0294

today's generative AI have been evolving since 2009. (Anagnoste, 2024) highlighted the gradual evolution of such technologies in one of his research projects: GPU computational possible, 2009 – Convolutional neural networks, 2012 – Generative adversarial networks, 2014 – Diffusion models, 2015 – Google chips development, 2016 – Google transformer model, 2017 – OpenAI's text-to-image diffusion model release-DALL-E, 2021 – OpenAI's ChatGPT release.

The launch of ChatGPT and GPT-4 (Generative Pre-training Transformer) represented a significant milestone in the advancement of large language models (LLMs) (Naveed, et al., 2024) (Li, Zhu, Ma, Liu, & Shah, 2023).

Recent years have seen remarkable progress in intelligent automation (Anagnoste, 2018), but especially in natural language processing, challenging previous beliefs by demonstrating that AI can now excel in tasks requiring cognitive and creative skills, traditionally thought to be human domains (Gruetzemacher, 2022).

The major objective resumes to exploring the effectiveness of such AI architecture, especially NLP techniques. By prospecting the insights from two carefully selected Word documents, this project is set to build and fine-tune an intelligent model that is capable of crafting responses that not only fit the context but also span a variety of specific subjects. This endeavor aims to weight the effects of integrating cutting-edge algorithms, such as deep learning and Long Short-Term Memory (LSTM) networks, on the improvement of the responses. Using data from just these two documents as a foundational dataset provides a focused yet robust basis for initial training and testing. This step represents a preliminary exploration, serving as a proof of concept for demonstrating the capabilities of intelligent response-generation systems.

The research question encapsulates the core essence of all outlined objectives, namely – How does an AI model, trained on a limited dataset and using deep learning and LSTM networks, reveal the underlying mechanism for generating contextually appropriate responses?

Literature review

In this section, a comprehensive review of the literature was carried out to examine current studies on incorporating artificial intelligence with natural language processing methods for generating responses, based on chosen articles. A well-detailed literature review has the target to give an exhaustive summary of the existing research field. In addition, the next synthesis will address the main research question.

A succinct explanation of concepts

In the evolving landscape of computer science, the field of artificial intelligence and its various subdisciplines play a pivotal role in the development of technologies that emulate human intelligence and creativity (IBM, 2023).

AI can take various forms – from facial detection and recognition, algorithms for search and recommendations, to fake detection algorithms, chatbots or digital assistants. (Săniuță & Filip, 2021). The field of AI has introduced a variety of terminologies that are frequently conflated or whose distinctions remain ambiguous to numerous end-users (Bezko, 2023).

Artificial intelligence encompasses the broad discipline of creating machines capable of performing tasks that typically require human intelligence, laying the groundwork for various specialized fields within computer science (Figure 1). Machine learning, a subset of AI, involves programming systems to learn from data, enabling them to make predictions on new data. Within ML, neural networks (NNs), inspired by the human brain's structure, facilitate complex pattern recognition through interconnected nodes (Caprasi, 2023). On the other hand, natural language

processing is a subset of AI that deals with the interaction between computers and human language. (Bezko, 2023).

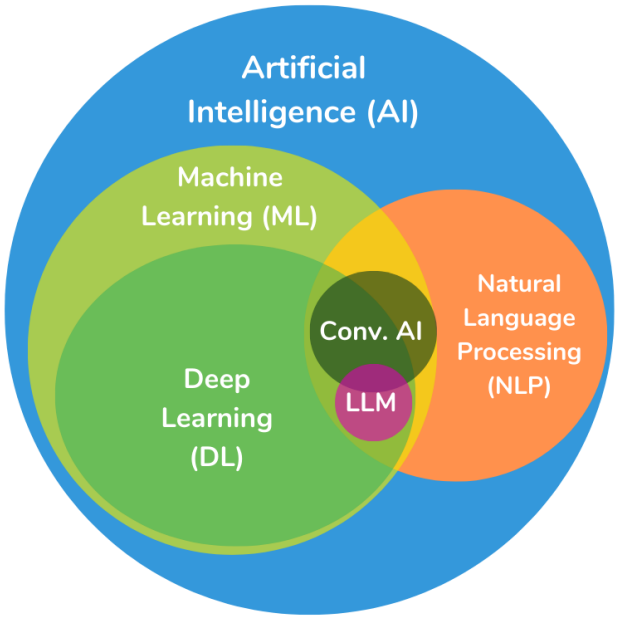

Figure 1. The relationship between concepts

Source: (Bezko, 2023)

Deep learning (DL), a further specialization of NNs, utilizes networks with multiple layers to analyze data with a depth of complexity, leading to more sophisticated learning and decision-making capabilities. According to Gartner (2023), "enables machines to learn from a representation of artifacts from data and models, and use it to generate brand-new, completely original artifacts that preserve a likeness to original data or models". Generative AI, which falls under the umbrella of DL, focuses on creating new content, including text, images, and sounds, by learning from existing materials (Caprasi, 2023). Conversational AI, a specialized area within natural language processing, is dedicated to the creation of chatbots and virtual assistants capable of engaging in dialogue with human users (Bezko, 2023).

At the climax of these advancements, large language models represent a type of generative AI that excels in producing human-like text, drawing on extensive textual data to mimic the nuances of human language and thought (Caprasi, 2023).

Evolution of Generative AI and LLMs through NLP

The conversational AI and virtual assistant segment is experiencing the highest growth rate within the contact center industry, projected to fuel a 24% increase in the market during 2024, according to Gartner's forecast. (Gartner, 2023)

According to Yao, Y et al (2023), LLMs do not only revolutionize, but mark a significant advancement beyond traditional language models. Originally, language models relied on statistical approaches, forming the foundation of computational linguistics. However, the introduction of transformer technology has dramatically expanded their scope and capabilities (Zhao, et al., 2023).

(Wolfram, 2023) exhaustively explains what is going on inside such generative models, giving a rough outline of why it is that and how it can precisely produce what we perceive as

meaningful text. Consequently, the core function of ChatGPT – for instance – is to provide a logical continuation of any given text, aiming for what would be expected based on extensive data from billions of webpages and other texts. Impressively, when ChatGPT composes something like an essay, it continually predicts the next appropriate word or token based on the existing text, thus enabling it to add one word at a time. It's worth noting that by "token" it might refer to a full word or just a fragment of one, which explains how it occasionally generates novel words.

A comprehensive language model consists of an extensive set of parameters that are initially trained through various tasks, such as filling in blanks within sentences and predicting subsequent words.

These models are meticulously trained on extensive datasets, enabling them to understand and generate text that closely resembles human speech. Equipped with potentially hundreds of billions of parameters, these LLMs are refined by analyzing large volumes of text. Their development has led to major advancements in NLP (Feng, et al., 2021) and their application spans a wide array of sectors, including risk assessment (Novelli, Casolari, Rotolo, Taddeo, & Floridi, 2023), programming (Cai, et al., 2023), detecting vulnerabilities (Jain, Gervasoni, Ndhlovu, & Rawat, 2023), analyzing medical texts (Thirunavukarasu, et al., 2023), and optimizing search engines (Arcila, 2023) (Yao, et al., 2023).

The development of NLP has transitioned from statistical language models to neural language models, and further evolved from pre-trained language models (PLMs) to LLMs. Traditional language modeling focuses on training models for specific tasks within a supervised framework, whereas PLMs undergo training in a self-supervised environment on vast text corpus (Devlin, Chang, Lee, & Toutanova, 2019) (Peters, et al., 2018) (Lewis, et al., 2019). The goal is to acquire a universal representation that can be applied across a multitude of NLP tasks (Naveed, et al., 2024).

LLMs have developed capabilities such as reasoning and decision-making, attributes not directly trained for but emerging from their vast scale. The trend of releasing more LLMs over time is on the rise, with several notable models introduced through the years being highlighted in figure underneath. The chart (Figure 2) showcases a growing shift towards instruction-tuned and open-source models, underscoring the dynamic changes and emerging trends within the field of natural language processing research.

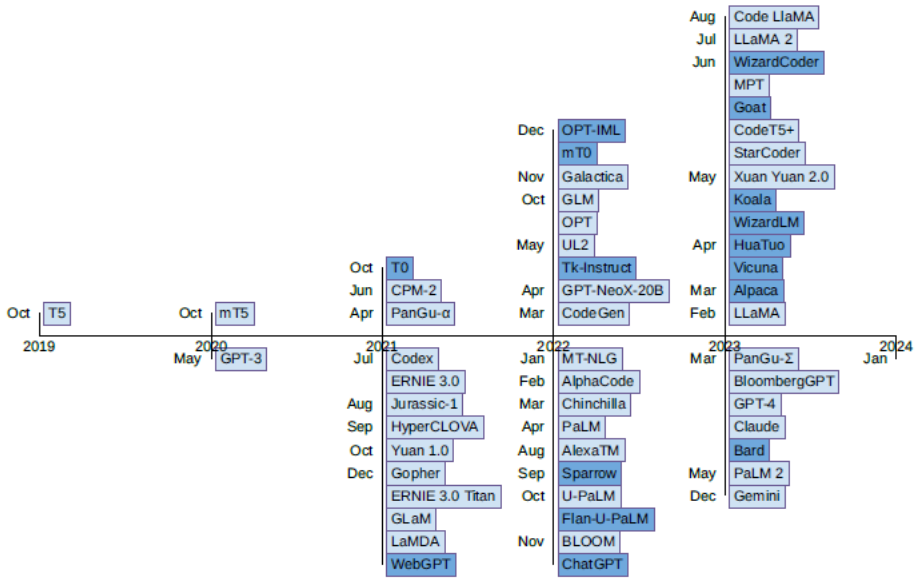

Figure 2. The timeline of LLM releases: light blue for 'pre-trained', dark for 'instruction-tuned', upper half open-source and lower half closed-source models

Source: (Naveed, et al., 2024)

Document analysis and Recurrent Neural Networks

Document representation focuses on converting unstructured documents into formats that machines can understand, preserving the semantic essence of the content (Zhanga, Lia, & Wang, 2019). Within Zhanga, W et al research paper, multiples ideas are brought out around document analysis and textual modeling, proposing LSTM network model as a reliable method to learn about distributed vector representations of documents.

Text generation in deep learning can be classified into three main models: the Vector-Sequence Model, used for tasks like image captioning where the input is fixed and output varies; the Sequence-Vector Model, suitable for classification with variable input and fixed output; and the Sequence-to-Sequence Model, for situations where both input and output are variable, commonly applied in language translation. The Sequence-to-Sequence Model is the most prevalent among text generation applications due to its flexibility in handling various sizes of input and output (Fatima, Imran, Zenun Kastrati, & Soomro, 2022).

Deeply diving into this topic, (Sak, Senior, & Beaufays, 2014) emphasize that due to the complex and evolving nature of speech, which exhibits elaborate correlations across multiple timescales, recurrent neural networks (RNNs) are better suited for its analysis compared to feedforward neural networks. The cyclical connections inherent in RNNs enable them to effectively capture and model the sequential data of speech, making them a superior choice for this type of task.

Lately, recurrent neural networks based on long-short-term memory (Hochreiter & Schmidhuber, 1997) have proven successful in capturing the contextual relationships between words in documents, commonly used to generate distributed vector representations of text (Liu & Lapata, 2018) (Boom, Canneyt, Demeester, & Dhoedt, 2016), according to what Zhanga, W et al

had scrutinized. Additionally, the RNN reinforcement is well-documented across numerous studies and publications – besides the article of Olah (2015), Sak, H et al (2014) and Kang, H et al (2020) admit recurrent neural networks have achieved notable success in tasks that involve sequence analysis and prediction, including handwriting recognition and language modeling.

In his research paper, Christopher Olah (2015) emphasizes that humans use prior knowledge to understand new information, unlike traditional neural networks which lack this ability to build on previous data.

Recurrent neural networks utilize loops within their architecture, allowing them to pass information from one step to the next, which is crucial for processing sequential data. Unlike traditional neural networks, RNNs can be seen as multiple copies of the same network, each transferring data to its successor (Figure 3). This capability makes them particularly adept at tasks that involve understanding sequences, such as speech recognition or language translation, by maintaining a form of memory over the inputs they process, the same research of Christopher Olah concluded (2015).

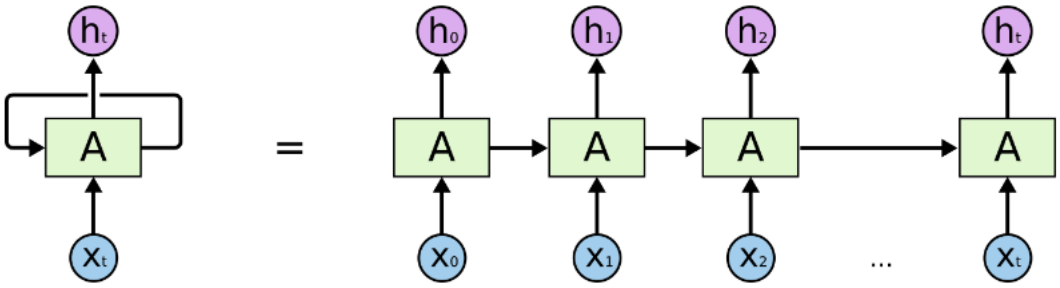

Figure 3. A RNN loop

Source: (Olah, 2015)

In the described diagram, a segment of a neural network processes an input (x_t) to produce an output (h_t), with a loop allowing the transfer of information from one step to the next. This looping mechanism, characteristic of recurrent neural networks, gives the impression of complexity. However, upon closer examination, RNNs can simply be seen as repetitions of the same network, where each iteration passes information forward. Unrolling this loop visually illustrates how RNNs sequentially manage data, akin to multiple network copies communicating in sequence (Figure 4).

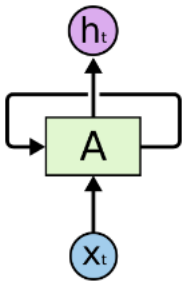

Figure 4. An unfolded RNN

Source: (Olah, 2015)

The structure of recurrent neural networks is inherently suited to process sequences and lists due to their chain-like configuration, making them the ideal choice for handling such types of data (Medium, 2019). The architecture is expressed as a multiple time dependencies alongside long-term dependencies, according to Lindemann, B et al study (2021).

LSTM network architecture, long-term dependencies paradigm and how these models operate

Long Short-Term Memory networks, often referred to simply as "LSTMs," are a specific type of recurrent neural network designed to learn long-term dependencies. Introduced by (Hochreiter & Schmidhuber, 1997), their design has been enhanced and popularized by numerous researchers since. LSTMs have shown remarkable success across a broad range of applications and have become a staple in the field (Olah, 2015).

As Sak, H et al (2014) explored, the long short-term memory architecture addresses certain limitations of traditional recurrent neural networks, making it particularly well-suited for acoustic modeling tasks. Its design allows it to effectively capture long-term dependencies, enhancing its performance in complex modeling scenarios. But what precisely constitutes the problem of long-term dependencies?

At times, it's sufficient for humans to consider only the most recent information to complete a current task, necessitating a focus on the immediate context to inform decision-making or predictions effectively (Medium, 2019). RNNs have the potential to link past information to current tasks, such as understanding a present video frame based on previous ones. However, their effectiveness diminishes with long-term dependencies due to difficulty in maintaining the connection over large gaps between relevant information and its application. While theoretically capable, RNNs in practice struggle to learn these long-term dependencies, a problem not faced by LSTMs, which are designed to overcome this limitation and effectively handle both short and long-term data dependencies (Olah, 2015) (Medium, 2019).

This is technically called Vanishing gradient problem. In a far-reaching article from Medium (2019), the vanishing gradient problem is broadly spelt out. It occurs when gradients shrink as they are propagated back to earlier layers in a recurrent neural network, leading to minimal or no changes in the weights of these layers. This issue prevents the network from learning from long sequences, as the small gradients don't allow for significant adjustments in the network's parameters, hindering its ability to converge to a good solution. In essence, without adequate gradient values, the network fails to learn effectively, especially for data points that have long-term dependencies (Wang, 2019).

For LSTM models, “remembering information for longer periods of time is their default behavior”. At a first glance, the LSTM network comprises a memory cell that stores values and three types of gates – an input gate, an output gate and a forget gate (Medium, 2019). Fundamentally, as (Sak, Senior, & Beaufays, 2014) explains, the LSTM architecture features unique components known as memory blocks within its recurrent hidden layer. All these blocks have memory cells that possess self-connections, being able to maintain the network's temporal state. Furthermore, the gates regulate the information flow within the network, and are referred to as multiplicative units.

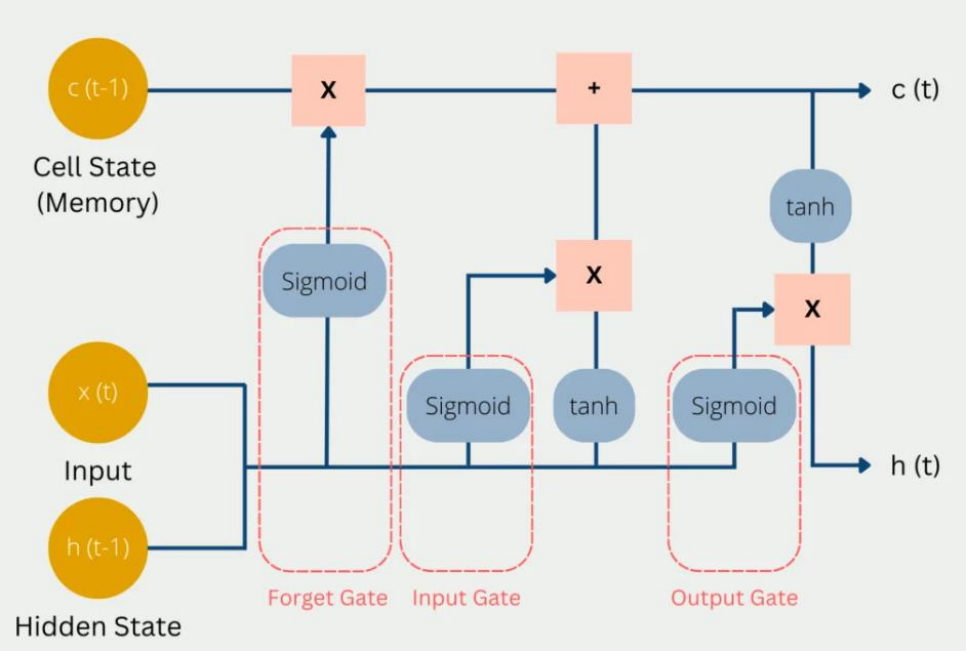

Figure 5. LSTM Cell Architecture

Source: (Data Base Camp, 2022)

An LSTM unit is composed of four distinct feedforward neural networks. Each network features an input and an output layer, with every neuron in the input layer being connected to every neuron in the output layer. Consequently, this structure gives rise to four fully connected layers within the LSTM unit.

Within an LSTM unit, three out of its four feedforward neural networks function as gates for managing information: the forget gate, the input gate, and the output gate. These gates control the processes of deleting, adding, and using memory content, respectively.

Forget gate determines which information to retain or discard from the cell state by considering the current input and the previous hidden state. A sigmoid function, producing outputs between 0 and 1, helps decide the fate of the information; a 0 signifies that the information can be forgotten, while a 1 indicates it should be kept. This decision directly affects the cell state by multiplying these values, effectively removing unneeded information.

Input gate evaluates the importance of the new input for the task at hand by integrating it with the previous hidden state and a weight matrix. The significant information identified is then added to the cell state, updating it to a new state that represents the current long-term memory for the next iteration.

Finally, the output of the LSTM is possible only through output gate. The cell state is then modified by the tanh function to help in this selection process. The outcome can vary depending on the application, such as producing a word that complements a sentence's meaning (Data Base Camp, 2022) (Calzone, 2022).

The LSTM architecture features two main components for storing information: the Cell State, which serves as the unit's long-term memory, and the Hidden State, akin to short-term memory familiar from standard neural networks. The Cell State retains crucial information over long sequences, while the Hidden State captures immediate data from previous computation steps. This dual memory system enables LSTMs to effectively manage and utilize both recent and relevant historical information for each computational step, which includes processing the current

input $x(t)$, the previous Cell State $c(t-1)$, and the previous Hidden State $h(t-1)$ (Data Base Camp, 2022).

Methodology

This section outlines the research methodology employed for the present study analysis. Touching the main objective of decrypting how a generative model works behind, the whole step-by-step process was meticulously defined to provide a transparent and comprehensive explanation of how each phase was executed.

The proposed work has used the Google Collaboration notebook, which is an open-source Web platform, alongside Python code to establish an experimental environment. This platform facilitated the code development and execution (Appendix), and uploading of documents containing corpus of text. A generative prototype was built in the attempt to explore the complexities involved in developing models, highlighting the integration of RNN techniques. With a special emphasis on the LSTM algorithm, the paper aims to enlighten the particular neural network architecture favored for processing sequential data.

As a dataset to train the model on, two scientific papers were selected – paper 1: "Robotic Automation Process - The next major revolution in terms of back-office operations improvement" and paper 2: "Robotic Automation Process – The operating system for the digital enterprise". These articles have been merged into a single Word document which was further used as a training foundation.

The following steps has been followed: Input – Data preparation – Model initialization – Loss calculation – Backpropagation – Hyperparameter tuning – Hyperparameter tuning – Output (Razin, Karim, Mridha, Rifat, & Alam, 2021).

The dataset preparation phase involves utilizing selected papers to form the foundation of the training dataset. During this phase, each sentence from these papers is subjected to preprocessing steps such as tokenization, stemming, and removing stopwords. For example, take a sentence from Paper 1: "For Romania, a key destination for SSCs and BPOs, this technology will make them more competitive, but also will lead to a creation of a series of high-paid jobs while eliminating the low-input jobs."

Tokenization is the process of dividing text into smaller pieces. This implies dividing the paragraphs down into sentences and further dividing those sentences into words. (Noaman, Sarhan, & Rashwan, 2018) Thus, it undergoes into a series of tokens as it follows: "for", "Romania", ",", "a", "key", "destination", "for", "SSCs", "and", "BPOs", ",", "this", "technology", "will", "make", "more", "competitive", ",", "but", "also", "will", "lead", "to", "a", "creation", "of", "a", "series", "of", "high-paid", "jobs", "while", "eliminating", "the", "low-input", "jobs".

Afterwards stemming process steps in, which means reducing words to their basic or root form (Kumar, 2022). For instance, taking the words "automate," "automation," and "automated" from Paper 2: ["automate", "automation", "automated"]. Stemming processes these words to identify and reduce them to their shared root or stem: ["automat", "automat", "automat"].

Then, stopwords are common words with minimal semantic value that frequently appear in diverse text environments. During our data preprocessing phase, eliminating stopwords helps the generative AI model concentrate on significant content instead of routine linguistic components. Words like "the," "is," "in," and "of" are usually found in the list of stopwords (Mungalpara, 2022). For instance, examine the following sentence that has been tokenized and stemmed from Paper 1: ["unveil", "the", "power", "of", "AI", "in", "respons", "generat"]. After removing stopwords, this sentence is transformed into: ["unveil", "power", "AI", "respons", "generat"].

Initialization – the model initialization phase encompasses setting up the architecture of an LSTM-based neural network, detailing various parameters that determine its structure and operational characteristics. Each layer functions as a mechanism for identifying sequential patterns in the input data. The initial layer is equipped with 256 LSTM units, and the subsequent layer contains 250 LSTM units.

Loss Calculation – the model assesses its performance by computing a loss function, which quantifies the difference between the predicted outputs and the actual ground truth.

Backpropagation – using backpropagation, the model fine-tunes its internal parameters, e.g weights and biases. During it, gradients of the loss function with respect to each parameter are calculated and used to update the model's weights in the direction that reduces the loss. This iterative process continues across multiple training epochs, gradually refining the model's ability to generate coherent and contextually relevant text.

Training Process – the model undergoes several iterations, known as epochs, during its training phase. For instance, by setting the training to 20 epochs, the model is exposed to and learns from the dataset a total of 20 times. Modifying the batch size, for example, increasing it from 20 to 128, helps to achieve a balance between computational efficiency and the stability of the model, taking into account the computing resources at hand.

Hyperparameters refer to the settings that are predetermined by the user rather than being learned through the training phase. Within the framework of LSTM networks, such parameters encompass the quantity of hidden layers, the count of LSTM units per layer, the learning rate, batch size, dropout rate, among others, as described by (Luaran & Alfred, 2022). Hyperparameters may appear more than once in an algorithm flow, as it has direct relation with improving the model's performance.

Word Embedding Size – this value represents the dimensionality of the word embeddings utilized in the model. In the present case, the embedding dimension is established at 100, defining the size of the dense vector representations that the embedding layer learns.

LSTM Units in the First Layer – this detail specifies the number of LSTM units present in the model's initial layer. This layer has 256 LSTM units.

LSTM Units in the Second Layer – this one indicates the quantity of LSTM units too in the model's subsequent layer. This time, the second layer comprised 250 LSTM units.

Dropout Rate – it is employed with a rate of 0.2. During training, this technique randomly drops out a proportion of neurons from the LSTM layers, helping to prevent overfitting by encouraging the network to learn more robust features and reducing co-adaptation among neurons.

Learning Rate – this parameter, set to 0.001 in this experiment, determines the magnitude of adjustments made to the model's weights during each training iteration. A lower learning rate ensures smoother convergence during optimization, while a higher rate may lead to faster but less stable training.

Number of Training Epochs – it defines the total rounds (iterations) of training the model engages in, passing through the full training dataset each time. 89 epochs were needed so the model was trained and capable to refine its internal parameters and improve its generative content.

Batch Size – during training, the dataset is divided into batches, with each batch containing a specified number of samples. The batch size, set to 128 in this study, influences the trade-off between computational efficiency and model stability. Larger batch sizes may accelerate training but could also lead to memory constraints.

Validation and Testing – the model is subjected to validation on a separate dataset, perhaps a sample from Paper 1 and Paper 2, to ensure its generalization. Rigorous testing ensures the model

responds coherently to diverse contextual cues, aligning with the thematic diversity of the chosen articles.

Both training and tuning hyperparameters represents a continuous improvement process. The LSTM model evolves by absorbing the comprehensive knowledge embedded in the chosen articles, fine-tuning its cognitive structure to produce responses that reflect the contextual insights encapsulated in Paper 1 and Paper 2.

Results and discussions

The operating model performed a thorough training process, made up of 89 iterations, to leverage its content generation capabilities. The whole iterative process was a trial and error that in the end revealed 89 iterations were optimal or enough to enable the model to produce text.

The generated text is “Automation capabilities on more than 2 typically asked questions and different industries – establishing the next operating 60 of the digital workforce applications since the world is done to make methodology so here is a huge opportunity for organizations for employees and organizations and BPOs the new norm all in the same boat using an inappropriate delivery method applying rationally software implementation with based on 1 decisions also can be eliminated with cognitive how what can be used in order to confirm that the past year a pattern arose in terms of what is not a great experience for your intelligence". The output generated post-training illustrates the model's nuanced understanding, nevertheless with some limitations in achieving a fully human-like text quality.

The evolution of the model's performance over the training period is quantitatively evidenced by the metrics of loss and accuracy. Initially, at the first epoch, the model exhibited a loss of 6.8689 and an accuracy of merely 0.0423, indicating a substantial gap from desired performance levels. However, by the 89th iteration, a marked improvement was observed, with the loss significantly reduced to 1.0204 and accuracy increased to 0.8142. Showing evolution, the model displays a high level of efficiency and also potential to generate coherent, but not yet entirely human-like responses.

Epoch 1/89 61/61 [==============================] - 77s 1s/step - loss: 6.8689 - accuracy: 0.0423

Epoch 89/89 61/61 [==============================] - 77s 1s/step - loss: 1.0204 - accuracy: 0.8142

Additional training rounds might be necessary to refine the model's accuracy further. Nevertheless, considering the constraints posed by the parameters and the dataset volume provided, the model demonstrated an admirable level of efficiency, producing intelligible content that, while not perfectly mimicking human speaking, represents a substantive step towards the comprehension of these sophisticated AI-driven text generation technologies.

Research limitations

The major limitations of this experimental prototype are mainly related to the volume of data, quality, model complexity and number of performed training rounds. Therefore, concerning the input given to the model, it is not large or diversified enough to enable the generation of highly accurate and huma-like text. Having not enough data, this constraint can affect model’s ability to extrapolate across multiple topics or contexts or even capture the essence of natural language. Additionally, the generated content, despite being coherent and relevant to the initial dataset, still falls short of completely mimicking human writing styles.

In the context of this AI experiment, the acquisition of additional data requires the optimization of the algorithm relative to the data volume. The larger the dataset, the more room for improvement – time to fine-hone the algorithm for peak performance. When you introduce a wide variety of inputs to the model, it becomes better equipped to make sense of data it's never seen before - a crucial skill for avoiding failures. Take a model that can adapt, and you'll find it excels in countless situations, thriving on the very diversity of data that might once have been a weakness.

Regarding the complexity and training side, even though 89 iterations were identified as optimal for what it was intended to be achieved, this number may not suffice for reaching the highest possible accuracy or for fully exploring the model's capacity. There's an implication that more extensive training or a different model architecture could yield better results.

Lastly, the focus on automation capabilities-type of dataset used for training the model and the specific parameters used for the model may limit its applicability or performance in other domains or for other types of textual content.

Conclusions

This research study uses long-short term memory techniques and recurrent neural networks to highlight and describe the intricate mechanism underlying a generative artificial intelligence model. The topic is creative due to the pioneering endeavors that add to the body of literature by illuminating aspects of the discipline that have not yet been addressed.

The study, empowered by specialty literature, has shown the unique capabilities of long short-term memory networks, which surpass traditional recurrent neural networks by effectively managing information over longer periods. This higher performance is attributed to their superior architecture, comprising a cell state and three regulatory gates, designed to selectively remember and forget information. To further reinforce their advantage, LSTMs also solve the crucial problem of vanishing gradients that impede RNNs. This endeavor adds to a better comprehension of LSTMs' importance in developing the science of neural networks by thoroughly examining their architecture and functionality.

With its comprehensive review of generative systems, the study aims to provide the research community with consistent starting points. Moreover, it offers insights for the comprehension and development of models, which may enable accessibility to create an AI model from the ground up. This article's primary goal is to inspire and direct future research, enabling the creation of more advanced LLMs, by providing fundamental knowledge. This work offers new insights into the mechanisms of AI-driven creativity through a thorough examination of the creative processes within this AI prototype. In an effort to develop a flexible and durable model with consistent outputs, it represents a substantial advancement in the training and operation of generative AI systems.

By applying the model to a bigger and more varied dataset, future research aims to validate this experiment and improve clarity, deep comprehension, and a more accurate and flexible text output that resembles that of a human.

References

Anagnoste, S. (2018). Robotic Automation Process - The operating system for the digital enterprise. *Proceedings of the International Conference on Business Excellence*, *12*(1), 54-69. doi:https://doi.org/10.2478/picbe-2018-0007

Anagnoste, S. (2024, March). Today’s GenAI capabilities have been developing since 2009. *OnStrategy*. Bucharest: OnStrategy.

Arcila, B. B. (2023). Is it a Platform? Is it a Search Engine? It's Chat GPT! The European Liability Regime for Large Language Models. *J. Free Speech L.*, *3*, 455.

Bezko, G. (2023, December 7). *Understanding AI, ML & Co. in Contact Centers: Definitions and Explanations*. Retrieved from Miarec: https://blog.miarec.com/contact-centers-ai-definition

Boom, C. D., Canneyt, S. V., Demeester, T., & Dhoedt, B. (2016). Representation learning for very short texts using weighted word embedding aggregation. *Pattern Recognit. Lett.*, 150–156.

Cai, Y., Mao, S., Wu, W., Wang, Z., Liang, Y., Ge, T., . . . Duan, N. (2023). Low-code LLM: Visual Programming over LLMs. (C. University, Ed.) *"arXivpreprintarXiv:2304.08103*. doi:https://doi.org/10.48550/arXiv.2304.08103

Calzone, O. (2022, February 21). *An Intuitive Explanation of LSTM*. Retrieved from Medium: https://medium.com/@ottaviocalzone/an-intuitive-explanation-of-lstm-a035eb6ab42c

Caprasi, C. (2023, July 21). *Artificial Intelligence, Machine Learning , Deep Learning, GenAI and more*. Retrieved from Medium - Women in Technology: https://medium.com/womenintechnology/ai-c3412c5aa0ac

Chui, M., Hazan, E., Roberts, R., Singla, A., Smaje, K., Sukharevsky, A., . . . Zemmel, R. (2023). *The economic potential of generative AI: The next productivity frontier.* McKinsey & Company, McKinsey Digital. Retrieved from https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/the-economic-potential-of-generative-ai-the-next-productivity-frontier#introduction

Data Base Camp. (2022, June 4). *Long Short-Term Memory Networks (LSTM)- simply explained!* Retrieved from Data Base Camp: https://databasecamp.de/en/ml/lstms

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *arXiv:1810.04805v2*. doi:https://doi.org/10.48550/arXiv.1810.04805

Fatima, N., Imran, A. S., Zenun Kastrati, S. M., & Soomro, A. (2022). A Systematic Literature Review on Text Generation Using Deep Neural Network Models. *IEEE Access*, *10*.

Feng, S. Y., Gangal, V., Wei, J., Chandar, S., Vosoughi, S., Mitamura, T., & Hovy, E. (2021). A Survey of Data Augmentation Approaches for NLP. *Findings of the Association for Computational Linguistics: ACL-IJCNLP*, 968–988.

Gartner. (2023, July 31). *Gartner Says Conversational AI Capabilities Will Help Drive Worldwide Contact Center Market to 16% Growth in 2023*. Retrieved from Gartner : https://www.gartner.com/en/newsroom/press-releases/2023-07-31-gartner-says-conversational-ai-capabilities-will-help-drive-worldwide-contact-center-market-to-16-percent-growth-in-2023

Gartner. (2023). *Generative AI*. Retrieved from Gartner : https://www.gartner.com/en/information-technology/glossary/generative-ai

Gruetzemacher, R. (2022). The Power of Natural Language Processing. *AI And Machine Learning*. Retrieved from https://hbr.org/2022/04/the-power-of-natural-language-processing

Hochreiter, S., & Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, *9*(8), 1735-1780. doi:https://doi.org/10.1162/neco.1997.9.8.1735

IBM. (2023, July 6). *AI vs. Machine Learning vs. Deep Learning vs. Neural Networks: What's the difference?* Retrieved from IBM: https://www.ibm.com/blog/ai-vs-machine-learning-vs-deep-learning-vs-neural-networks/

Jain, R., Gervasoni, N., Ndhlovu, M., & Rawat, S. (2023). A Code Centric Evaluation of C/C++ Vulnerability Datasets for Deep Learning Based Vulnerability Detection Techniques. *Proceedings of the16th Innovations in Software Engineering Conference*, 1–10.

Kang, H., Wu, H., & Zhang, X. (2020). Generative Text Steganography Based on LSTM Network and Attention Mechanism with Keywords. *Electronic Imaging*.

Kumar, T. S. (2022, August 26). *Natural Language Processing – Sentiment Analysis using LSTM*. Retrieved from Analytics Vidhya: https://www.analyticsvidhya.com/blog/2021/06/natural-language-processing-sentiment-analysis-using-lstm/

Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., . . . Zettlemoyer, L. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension. *arXiv preprint arXiv:1910.13461*.

Li, X., Zhu, X., Ma, Z., Liu, X., & Shah, S. (2023). Are ChatGPT and GPT-4 General-Purpose Solvers for Financial Text Analytics? An Examination on Several Typical Tasks. *Cornell University Computer Science - Computation and Language*. doi:https://doi.org/10.48550/arXiv.2305.05862

Lindemann, B., Müller, T., Vietz, H., Jazdi, N., & Weyrich, M. (2021). A survey on long short-term memory networks for time series prediction. *Procedia CIRP 99*, 650-655.

Liu, Y., & Lapata, M. (2018). Learning Structured Text Representations. *Trans. Assoc. Comput. Linguist., 6*, 63-75. doi:https://doi.org/10.48550/arXiv.1705.09207

Luaran, N., & Alfred, R. (2022). Assessment of the Optimization of Hyperparameters in Deep LSTM for Time Series Sea Water Tidal Shift. *Research Square* .

Medium. (2019). *Recurrent Neural Network and Long Term Dependencies*. Retrieved from Medium: https://infolksgroup.medium.com/recurrent-neural-network-and-long-term-dependencies-e21773defd92

Mungalpara, J. (2022, July 26). *Stemming Lemmatization Stopwords and N-Grams in NLP*. Retrieved from Medium: https://jaimin-ml2001.medium.com/stemming-lemmatization-stopwords-and-n-grams-in-nlp-96f8e8b6aa6f

Naveed, H., Khan, A. U., Qiu, S., Saqib, M., Anwar, S., Usman, M., . . . Mian, A. (2024). A Comprehensive Overview of Large Language Models. *Preprint*. Retrieved from https://doi.org/10.48550/arXiv.2307.06435

Noaman, H. M., Sarhan, S. S., & Rashwan, M. A. (2018). Enhancing recurrent neural network-based language models by word tokenization. *Human-centric Computing and Information Sciences, 8*(12). doi:https://doi.org/10.1186/s13673-018-0133-x

Novelli, C., Casolari, F., Rotolo, A., Taddeo, M., & Floridi, L. (2023). Taking AI Risks Seriously: a Proposal for the AI Act. *AI & SOCIETY*, 1-5.

Olah, C. (2015). *Understanding LSTM Networks*. Retrieved from colah's blog: https://colah.github.io/posts/2015-08-Understanding-LSTMs/

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., & Zettlemoyer, L. (2018, June). Deep Contextualized Word Representations. (M. Walker, H. Ji, & A. Stent, Eds.) *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2227–2237. doi:10.18653/v1/N18-1202

Razin, M. J., Karim, M. A., Mridha, M. F., Rifat, S. M., & Alam, T. (2021). A Long Short-Term Memory (LSTM) Model for Business Sentiment Analysis Based on Recurrent Neural Network. *Sustainable Communication Networks and Application*. doi:https://doi.org/10.1007/978-981-15-8677-4_1

Sak, H., Senior, A., & Beaufays, F. (2014). Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. *INTERSPEECH*, 338-342.

Săniuță, A., & Filip, S.-O. (2021). Artificial Intelligence: An Overview of European and Romanian Startups Landscape and the Factors that Determine their Success. *Strategica. Shaping the Future of Business and Economy*, 872-884.

Thirunavukarasu, A. J., Ting, D. S., Elangovan, K., Gutierrez, L., Tan, T. F., & Ting, D. S. (2023). Large language models in medicine. *Nature medicine*, *29*(8), 1930–1940.

Wang, C.-F. (2019, January 8). *The Vanishing Gradient Problem*. (T. D. Science, Editor) Retrieved March 1, 2024, from Medium: https://towardsdatascience.com/the-vanishing-gradient-problem-69bf08b15484

Wolfram, S. (2023, February 14). What Is ChatGPT Doing … and Why Does It Work? Retrieved from https://writings.stephenwolfram.com/2023/02/what-is-chatgpt-doing-and-why-does-it-work/

Yao, Y., Duan, J., Xu, K., Cai, Y., Sun, Z., & Zhang, Y. (2023). A Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly. *Preprint submitted to Elsevier*.

Zhanga, W., Lia, Y., & Wang, S. (2019). Learning document representation via topic-enhanced LSTM model. *Knowledge-Based Systems*, *174*, 194–204. doi:https://doi.org/10.1016/j.knosys.2019.03.007

Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., . . . Wen, J.-R. (2023). A Survey of Large Language Models. (C. University, Ed.) doi:https://doi.org/10.48550/arXiv.2303.18223

Appendix

```
doc_path_1  = "Robotic Automation Process – The next major revolution in terms of back office operations improvement_Training article 1.docx"
doc_path_2 = 'Robotic Automation Process – The operating system for the digital enterprise_Training article 2.docx'
```

```
import numpy as np
!pip install np_utils
!pip install python-docx
from docx import Document
import sys
from keras.models import Sequential
from keras.layers import Embedding, LSTM, Dense
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
from gensim.models import Word2Vec
from keras.layers import Dense, Dropout, LSTM
from tensorflow.keras import utils
from keras.callbacks import EarlyStopping
from keras.optimizers import Adam
```

```
[ ]  def read_text_from_word(doc_path):
         # Load the Word document
         doc = Document(doc_path)

         # Read the text from paragraphs
         text_from_word = ""
         for paragraph in doc.paragraphs:
             text_from_word += paragraph.text + "\n"

         return text_from_word
```

```
# Read text from the first Word document
text_from_doc1 = read_text_from_word(doc_path_1)

# Read text from the second Word document
text_from_doc2 = read_text_from_word(doc_path_2)
```

```
tokenizer = Tokenizer()
tokenizer.fit_on_texts([text_from_doc1,text_from_doc2])
total_words = len(tokenizer.word_index) + 1

# Creating input sequences
input_sequences = []
for article in [text_from_doc1,text_from_doc2]:
    token_list = tokenizer.texts_to_sequences([article])[0]
    for i in range(1, len(token_list)):
        n_gram_sequence = token_list[:i+1]
        input_sequences.append(n_gram_sequence)

max_sequence_length = max(len(seq) for seq in input_sequences)
input_sequences = pad_sequences(input_sequences, maxlen=max_sequence_length, padding='pre')
X, y = input_sequences[:, :-1], input_sequences[:, -1]
y = np.eye(total_words)[y]

# LSTM model
model = Sequential()
model.add(Embedding(input_dim=total_words, output_dim=100, input_length=max_sequence_length-1))
model.add(LSTM(units=256, return_sequences=True))
model.add(Dropout(0.2))
model.add(LSTM(units=250))
model.add(Dense(units=total_words, activation='softmax'))

# Compile the model
model.compile(optimizer=Adam(learning_rate=0.001), loss='categorical_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X, y, epochs=89, batch_size=128)z

# Save the weights
model.save_weights("trained_weights.h5")
```

```
[ ]  seed_text = "Automation"  # Initial seed text
     generated_text = seed_text  # Store the generated text
     for _ in range(100):
         tokenized_input = tokenizer.texts_to_sequences([seed_text])[0]
         padded_input = pad_sequences([tokenized_input], maxlen=max_sequence_length-1, padding='pre')

         predicted_index = np.argmax(model.predict(padded_input), axis=-1)[0]
         predicted_word = tokenizer.index_word[predicted_index]

         seed_text += " " + predicted_word  # Update seed text with newly predicted word
         generated_text += " " + predicted_word  # Append predicted word to generated text

     print("Generated Content:")
     print(generated_text)
```

DOI: 10.2478/picbe-2024-0294, pp. 3616-3633, ISSN 2558-9652 |
Proceedings of the 18th International Conference on Business Excellence 2024