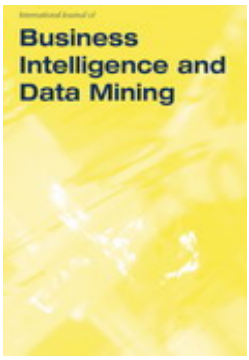


International Journal of Business Intelligence and Data Mining

ISSN online: 1743-8195 - ISSN print: 1743-8187
https://www.inderscience.com/ijbidm

DAMIAN – data accrual machine intelligence with augmented networks for contextually coherent creative story generation

B.J. Sowmya, D. Pradeep Kumar, R. Hanumantharaju, K.G. Srinivasa

DOI: 10.1504/IJBIDM.2022.10045744

Article History:
Received: 31 August 2021
Accepted: 27 December 2021
Published online: 30 November 2022

DAMIAN – data accrual machine intelligence with augmented networks for contextually coherent creative story generation

B.J. Sowmya*, D. Pradeep Kumar and R. Hanumantharaju

Department of Computer Science and Engineering,
M.S. Ramaiah Institute of Technology,
Affiliated to Visvesvaraya Technological University,
Machhe, Belagavi, Karnataka 590018, India
Email: sowmyabj@msrit.edu
Email: pradeepkumard@msrit.edu
Email: hmrcs@msrit.edu
*Corresponding author

K.G. Srinivasa

Department of Information Management and Coordination,
National Institute of Technical Teachers Training and Research,
Chandigarh-160019, India
Email: kgsrinivasa@gmail.com

Abstract: Cognitive computing refers to the usage of computer models to simulate human intelligence and thought process in a complex situation. Artificial intelligence (AI) is an augmentation to the limits of human capacity for a particular domain and works as an absolute reflection of reality. AI is where a computer program is able to efficiently make decisions without previous explicit knowledge and instruction. The concept of cognitive intelligence was introduced. The most interesting use case for this would be an AI bot that doubles as a digital assistant. This is aimed at solving core problems in AI like open domain question answering, context understanding, aspect-based sentiment analysis, text generation, etc. The work presents a model to develop a multi-resolution RNN to identify local and global context, develop contextual embedding via transformers to pass into a seq2seq architecture and add heavy regularisation and augment data with reinforcement learning, and optimise via recursive neural networks.

Keywords: cognitive computing; artificial intelligence; AI; data augmentation; human intelligence; recurrent neural network; transformer model.

Reference to this paper should be made as follows: Sowmya, B.J., Kumar, D.P., Hanumantharaju, R. and Srinivasa, K.G. (2023) 'DAMIAN – data accrual machine intelligence with augmented networks for contextually coherent creative story generation', *Int. J. Business Intelligence and Data Mining*, Vol. 22, Nos. 1/2, pp.115–130.

Biographical notes: B.J. Sowmya is an Assistant Professor in the Department of Computer Science and Engineering at M.S. Ramaiah Institute of Technology, Bangalore. Her area of interest is data analytics, machine learning, and internet of things. She is the co-author of several papers published in international journals, conferences, book chapters and a book.

D. Pradeep Kumar is an Assistant Professor in the Department of Computer Science and Engineering at M.S. Ramaiah Institute of Technology, Bangalore. His area of interest is data analytics, machine learning, and big data. He is the co-author of several papers published in international journals, conferences and book chapters.

R. Hanumantharaju is an Assistant Professor in the Department of Computer Science and Engineering at M.S. Ramaiah Institute of Technology, Bangalore. He is also a research scholar in the Department of Computer Science and Engineering at Siddaganga Institute of Technology, Affiliated to Visvesvaraya Technological University. His area of interest is internet of things, edge computing, embedded systems and distributed systems. He is the co-author of several papers published in international journals, conferences, book chapters and a book on internet of things.

K.G. Srinivasa is a Professor in the Department of Information Management and Emerging Engineering at National Institute of Technical Teacher Training and Research, Chandigarh (MHRD, Government of India). He is the recipient of All India Council for Technical Education – Career Award for Young Teachers; Indian Society of Technical Education – ISGITS National Award for Best Research Work Done by Young Teachers; Institution of Engineers (India) – IEI Young Engineer Award in Computer Engineering; Rajarambapu Patil National Award for Promising Engineering Teacher Award from ISTE – 2012; and IMS Singapore – Visiting Scientist Fellowship Award. He has published more than a hundred research papers in international conferences and journals.

1 Introduction

The data is generated daily, and most of it is either unintelligible or unprocessed, and no one has significant human competence to manage such levels of information (Yogatama et al., 2019). Augmenting this through machines, which process data faster is an obvious advantage. However, tasks that are ambiguously defined have no strict boundaries (Radford et al., 2018). The main aim is to build a system that can work efficiently to produce context relevant output while managing memory efficiently combining cognitive computing and deep learning (Chen et al., 2018). Cognitive Computing refers to the usage of a computer model to simulate human behaviour. These are complex and slow systems. Traditionally, deep learning models are fast and memory efficient systems, but cannot process abstract concepts like creativity (Noor, 2014). Intelligent devices capable of achieving greater accuracy can be programmed without depending on the above considerations. They can draw ideas, organise it, store it in a hierarchical data system and even draw a meaning from live talks. In terms of flexibility and efficiency, this kind of approach provides an essential advantage over conventional chat bots (Kulkarni et al., 2018).

In analysing language expressions, phrases play a significant role, especially in machine learning and linguistic interpretation. Understanding phrases allows analytic devices to research phrases or sentences (Valipour and Wang, 2018). A cognitive network is a network that incorporates logical approaches – motivated through the course of learning by actual human beings to adapt and make strategic choices with a purpose to the fullest. a network is a cognitive network (Baggio et al., 2019). The division of AI was initially based on devices' cognitive actions. However, technical developments allowed AI to illustrate cognitive science principles and concentrate on ways of storing knowledge for people, animals or machines. This has contributed to the growth of machine intelligence that allows language or emotional understanding, training and preparation, problem solving and thought (Naveed Uddin, 2019). The world significantly split into groups, patterns, classes, subjects, and various highly complex subdivisions is the theoretical world everyone is living in. Very much, a search for correct knowledge limits the option of those facets of knowledge which fit well into a context and disregards or dismisses others who do not. This process of approval and denial may lift our unique awareness of this part, but fundamental truths with the whole would probably be ignored (Fiorini, 2018).

Almost 21 years ago, the influence of artificial intelligence (AI) was introduced but the great impact, which allows difficult efforts to do or to acquire information, particularly resembles human intelligence, so that it is possible to know how people think and to decide things over the last generations. One can be aware of how human mind neurons behave, i.e., AI, one can use the strategic word AI in order to understand about AI depending on knowledge and to learn from the past how a person can interpret information on the machine on the basis of the human interface. On the modern globe the newest creative technologies, especially in-Order, for the processing, exchange, creation and conversion of vital information into the precise structure of the desired relationship, are AI (Shankar et al., 2019). Hence, a system able to work under both domains and form the basis for the next generation of smart devices is proposed. The objective is to create a system capable of modelling human aspects like creativity and make use of the massive compute architecture afforded by artificial intelligence.

2 Literature survey

Dragoni and Rospocher (2018) mention and discuss the scope of cognitive computing via a special track, taking into consideration three of the most exemplary formats. It diverges into explaining the self-supervised learning paradigm that cognitive computing follows for hypothesis and behavioural sciences. It discusses the first of the three examples, where a cognitive computing architecture was used to exploit and map contextual graph embedding from twitter feeds to DBpedia. This is an exemplary use case since it enables a two-way learning for both Twitter data and DBpedia data. The second example uses cognitive computing for decision making using sentiment analysis on financial data. It uses stock and company data, reinforced with stock values and news concerning the markets. This puts in a case of reinforced and self-supervised learning that is on par with advanced neural network architectures like transformers. The last example in the paper deals with aspect-based sentiment analysis using ontology features using human annotated data.

Radford et al. (2018) authored a paper that is on using generative pre-training, which is essentially used for knowledge transfer and transfer teaching. By using pre-trained models, one is able to cut down costs and simulate learnt intelligence in networks. This can be explained as retaining information about larger complex tasks when taken out of that particular environment and making use of it in a similar but different task. The paper describes generative pretraining on diverse corpora and making use of discriminative fine tuning on specific tasks. Essentially, the working behind the generative class of algorithms called GANs. The paper's method beats specifically trained discriminative architectures on benchmark tests in 9 out of 12 classes. It is notable that this branch of learning also deals with semi- supervised learning and self-supervised learning as in the paper above. The architecture used here is a Transformer with masked self-attention (similar to BERT Uncased).

Ballester (2019) showcases the usage of reinforcement for self and semi supervised learning tasks. Although the major contribution discussed here pertains to robotics, it forms a clear hierarchical idea about cognitive behaviours and responses in terms of neural network architectures. Associative learning and relation building is explained and explored in this paper, which is an aspect of cognitive computing that one can aim to leverage in our model. It defines a cognitive architecture of associations, self-learning, symbol and sampling layers. Local coverage areas are nodes in the architecture for sampling layers, which mimic neural pathways for identification, similar to symbol layers which help in forming visual relationships for objects. They use an ensemble of five learning algorithms, which define incremental learning and optimisation for the other layers as well. Symbol layers use them to form internal representations which are understood by a cognitive system, but not semantically coherent to be understood by humans. A 6th learning algorithm is used for the Association layer, and top-down responses are used for the learning process as per a 7th algorithm.

Human centred cognitive intelligence via IoT, AI and cloud computing (Chen et al., 2018). Our area of interest is understanding cloud applications and their representation stateful machines to be used for persistent storage as well. The paper discusses issues in learning structured language data for a cognitive system and explores usage of Reinforcement Learning as the new paradigm for learning, since it closely mimics how humans learn from their environments. They use examples from AlphaGo and Chess, which explores forming connections and ideas using a class of generative deep learning algorithms. The paper also focuses on the importance of interaction between a cognitive system and humans to learn and mimic behaviours effectively. They reference using IBM Cognos and Google Assistant as prominent cognitive computing platforms that perform interactions leveraging cloud computing ideas.

Ahmed et al. (2014) diverge into the engineering challenges and applications, hardware and networking technologies for cognitive computing. The paper also focuses mainly on automatic deployment, improving smart systems by cognitive reinforcement, and characteristics of a cognitive system. The defining criteria of a cognitive system is that it should be able handle ambiguity and dynamic adaptation to environments via responses. The engineering behind a synapse-based core is also divulged into as neuromorphic chips are described in the paper. The paper showcases examples about cognitive cameras, cognitive robots, cognitive cars, cognitive UAVs, etc. It also gives an engineering viewpoint of augmenting human thinking and processes using cognitive computing, instead of replacing human effort or mimicking it. It goes to the length of

discussing agents that can converse with other cognitive services/products and humans alike as the goal for cognitive computing.

Xu et al. (2018) describe in this paper that generation of sentences is hard because of maintaining semantic coherence. Discrete tokens need to be locally and globally corrected in the sequence produced, which is one of the major issues in dialogue generation. This paper deals with using multiresolution RNNs to combat this problem. The technique used in this paper is an auto encoder matching (AEM) model. Two auto encoders learn semantic matches and a mapping module learns utterance-level representations. This tandem leads to generation of sequences that have high semantic coherence and fluency. The AEM makes use of autoencoders for both the encoding and the decoding architecture, based on LSTMs. A simple feedforward network is used as the mapping module. The sentences generated are limited to 15 words to take into account long term dependency challenges in LSTMs. The optimisation follows adaptive moments (Adam). Data leakage is a concern since the final model is created by combining both the training and the validation data sets.

El Ouanjli et al. (2019) have written a paper that is primarily a survey of current developments and implementations for machine translation systems as well as natural language generation systems, dealing with coherence, semantic correctness and context representations in a language. The paper starts off with explaining the breakthrough RNN architectures like LSTMs and GRUs, and then diverges into encoder-decoder formats for the same. The paper also divulges into usage of generative networks like GANs (SeqGAN) for discrete token generation using a massive vocabulary and context vectors acting as an auxiliary language model. Task oriented models are attributed to the usage of reinforcement learning and its application in dialogue generation - which is our area of concern. Text summarisation, text retrieval, image translation, question answering, etc. are all explained with generative modelling and retrieval-based methods. Personalisation and context understanding is also a field of interest in the paper, with dialogue systems and review generation being the topics of discussion. Liang et al. (2018) From a cluster-level perspective, an efficient ensemble clustering approach for multi-view mixed data is suggested. To begin, the k-prototypes clustering technique is used to generate a set of clustering solutions for each view many times. Then, taking into account all of the clustering solutions, a cluster-cluster similarity matrix is created. Following that, the METIS algorithm does meta-clustering using the similarity matrix. After that, using majority voting to assign the objects to their appropriate clusters based on the meta-clustering, the final clustering results are achieved. The suggested algorithm's related temporal complexity is also examined. The superiority of our suggested approach was proved by experimental findings on different multi-view datasets.

From a cluster-level perspective, an efficient ensemble clustering approach for multi-view mixed data is suggested (Liang et al., 2018). To begin, the K-prototypes clustering technique is used to generate a set of clustering solutions for each view many times. Then, taking into account all of the clustering solutions, a cluster-cluster similarity matrix is created. Following that, the METIS algorithm does meta-clustering using the similarity matrix. After that, using majority voting to assign the objects to their appropriate clusters based on the meta-clustering, the final clustering results are achieved. The suggested algorithm's related temporal complexity is also examined. The superiority of our suggested approach was proved by experimental findings on different multi-view datasets.

To handle multimodal data, the author presents new deep neural network models Bhandari et al., 2020). The suggested models allow for smooth merging of multimodal data while also reducing the dimensionality of the input feature space. A multimodal stacked autoencoder is used in combination with a multi-layer perceptron-based regression model in this design. The architecture is suggested in two different forms. To demonstrate the relevance of multimodality for emotion identification, experiments were conducted on the multimodal benchmark dataset (RECOLA). The suggested architectures are taught utilising efficient training algorithms that are tailored to decrease the number of tuneable parameters in multimodal applications. The findings are promising, and the suggested method is computationally less expensive than previous methods. The performance is superior to or comparable to that of other strategies.

To demonstrate the relevance of multimodality for emotion identification, experiments were conducted on the multimodal benchmark dataset (RECOLA) (Kar and Mukherjee, 2021). The suggested architectures are taught utilising efficient training algorithms that are tailored to decrease the number of tuneable parameters in multimodal applications. The findings are promising, and the suggested method is computationally less expensive than previous methods. The performance is superior to or comparable to that of other strategies.

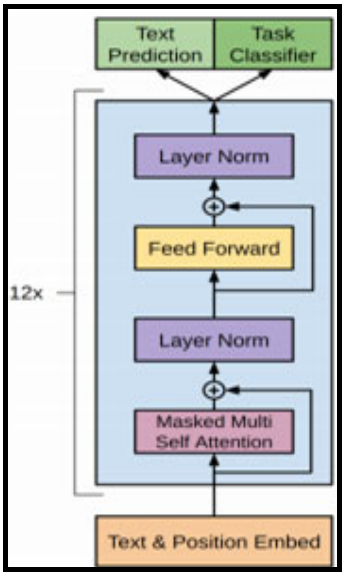
From the work of the above cited references, we have taken valuable insights and considerations while building a comparison metric for the results. The result is not strictly quantified in natural language generation, which is a primary domain of the project. focusing on general linguistic intelligence, activity-entity relationships and ground truth has led to a better comparison of results. Since we lack the hardware infrastructure to support the computing power required, we have scaled the problem and execution domains considerably to fit with our requirements. This also proved to be invaluable for the next steps into developmental cycles of our project, as one of the references enlightened us about the engineering challenges of a potential system, which closely resembles ours. This has given us insight into modularising and structuring our development and code and adding in incremental changes to accommodate new ideas about our domain.

3 Design and implementation

Our overarching objective of this work is to develop an architecture that has the ability to stimulate creativity. To achieve this, we employ three main strategies when designing our architecture. A Figure 1 of the architecture is shown below: Three of the strategies were employed in defining the context of the proposed architecture listed below.

Strategy 1	Develop a multi-resolution RNN to identify local and global context.
Strategy 2	Develop contextual embedding via transformers to pass into a seq2seq architecture.
Strategy 3	Add heavy regularisation and augment data with reinforcement learning and optimise via recursive neural networks.

Figure 1 Architecture design (see online version for colours)



All the three strategies can be combined with the transformer model as the crux. Taking a Text input, we include the position of each text token (for attention purposes). We embed them using transformer embeddings. This will help us recognise local and global context (using multi-resolution RNN). Next, we normalise the output to avoid gradient overflow. We do this using batch normalisation layers. We forward this network and normalise it again and add a transformer embedding lookup. This will convert our embeddings back to textual form, which is human readable.

3.1 Algorithms

3.1.1 RNN

An RNN is a recurrent neural network (RNN) which is the de-facto algorithm for temporal and sequential data. It builds temporal sequences that are able to identify and map patterns directly on embedded data easily. An improvement over this is the memory efficient recursive neural network architecture, which was also featured into the project. The paradigm extends to more than one linear architecture, often with stacked layers geared towards non-linear pattern identification and mapping. Examples of such architectures are Char-RNNs, which have had a resounding success for next character prediction. Hence the evolution of the same suite of algorithms to generalise better over a sequence – using multi resolution heads with Attention mechanisms at different layers of the network, allowing us to maintain context not only locally but also globally.

3.1.2 Transformer

A transformer model is a deep machine learning model, used notably and primarily in the field of natural language processing. Like RNNs, Transformers work on sequential data, and have exceptional uses in machine translation and machine comprehension tasks.

While RNNs require data to be processed in order, transformers have no such requirement. This allows the Transformer to learn representations parallelly, which had been a major problematic point with RNNs. They have quickly become state-of-the-art architectures in NLP, replacing previously renowned architects like the LSTM and the GRU. The transformer, in essence, is a seq2seq architecture – an architecture that transforms one sequence into another. The two main components of a transformer are the Encoders chained together and the Decoders chained together. The function of each encoder is to process its input vectors to generate what are known as encodings, which contain information about the parts of the inputs which are relevant to each other. It passes its set of generated encodings to the next encoder as inputs. Each decoder does the opposite, taking all the encodings and processing them, using their incorporated contextual information to generate an output sequence. To achieve this, each encoder and decoder makes use of an attention mechanism, which for each input, weighs the relevance of every input and draws information from them accordingly when producing the output. Each decoder also has an additional attention mechanism which draws information from the outputs of previous decoders before the decoder draws information from the encodings. Both the encoders and decoders have a final feed-forward neural network for additional processing of the outputs and contain residual connections and layer normalisation steps.

3.2 Algorithm working process

Step 1 Collect Data from Wikipedia, Onto Notes, Yahoo Chat dataset, etc.

Step 2 Clean and process the data. Processing involves removing html and formatting tags, links to and unnecessary information that acts as noise for our network.

Step 3 Use a multi-resolution RNN to form global and local context for all sentences and words. This will be used in the encoding scheme for Attention as well.

Step 4 Use a decoding-based transformer to generate embeddings on the dataset. With the use of masked attention layers, this will provide context vectors that have been pruned.

Step 5 Add batch normalisation to prevent gradient explosion during training.

Step 6 Use a seq2seq architecture and convert embedding's back to readable text format. This is similar to the word2vec's find function that localises on the basis of context. With embeddings from a transformer (which can read context bi-directionally), the system has a very good contextual understanding.

Step 7 Add regularisation between the layers to aid in stopping exploding gradients.

Step 8 Develop a language model using these words and embeddings as lookup IDs.

Step 9 To prevent deterministic outputs, describe a range of contextually correct words to use as each word is generated. This allows the model to have the aspect of 'creativity', if trained on a big enough corpus.

Step 10 Augment output data using reinforcement learning and language modelling concepts.

We dropped the use of recursive neural networks because it would not generalise as easily on our architecture, and it was immensely computing intensive since it follows the eager execution paradigm in tensor flow.

Using transformers as the core of our work, we present our engine data accrual machine intelligence with augmented networks (D.A.M.I.A.N).

The modules are all written in python using the tensor flow framework to build the network architecture. The module is focused on generative retraining on data to generate contextual embedding, which is a standalone block augmented by multi-resolution attention heads and temporal sequences from the RNNs.

Data augmentation in natural language can be done by either enriching the source, or by finding patterns and mapped non-linearity's in the data. We propose a system built on both - mapping non-linearity over long sequences via attention, and enriching data using reinforcement learning. reinforcement learning and knowledge based AIemulate human behaviour and provide inherent information that would otherwise not be apparent and explicit with other methods consistent with machine learning and deep learning.

The implemented modules have been summarised below:

3.3 Language model module

The LM module focuses on collecting, collating, processing and then predicting the next word in the sequence. This uses a Multi-resolution RNN to get global and local context, on the basis of which a word is predicted.

3.4 Contextual feature extraction module

This module is focused around extracting features from text and forming vector representations of the embeddings. The core foundation is built on a Transformer, which produces the embedding's required.

3.5 Data augmentation module

This is the post-processing part of the algorithm where we add cognitive computing and reinforcement learning designs to augment the implicit knowledge in our data. This helps us generalise better for any given sequence of words. We form tasteful automata for persistent storage, and the displayed data can be either stored on disk or discarded after a single run of the algorithm.

With the current trend of AI systems that can only replicate and efficiently implement repeated actions, we need a system capable of simulating creativity and taking decisions in an ambiguous environment, such as natural language processing. NLP boasts areas where probabilistic learning hits a dead end, and there is no effective way of validation for generative data. As a result, we rely on human intelligence and creativity for tasks that are well documented and understood at large, but do not have explicit instruction to train an intelligent system on. We bring an ensemble of established methods into play, creating an engine that can now generate data creatively.

4 Results

Figure 2 shows sample code written for our encoder-decoder architecture. The code here is the first working sample of our project that would take an input and then use the LM and transformer module to generate sentences using the entered string as prefix. Originally, we generated n samples to ensure that our model was non-deterministic in its working. The code displayed here was written in base tensor flow, v1.15.

Figure 2 Sample code for our architecture (see online version for colours)

```
            context=context,
            batch_size=batch_size,
            temperature=temperature, top_k=top_k
        )

        saver = tf.train.Saver()
        ckpt = tf.train.latest_checkpoint(os.path.join(models_dir, model_name))
        saver.restore(sess, ckpt)

        while True:
            raw_text = input("Model prompt >>> ")
            while not raw_text:
                print('Prompt should not be empty!')
                raw_text = input("Model prompt >>> ")
            context_tokens = enc.encode(raw_text)
            generated = 0
            for _ in range(nsamples // batch_size):
                out = sess.run(output, feed_dict={
                    context: [context_tokens for _ in range(batch_size)]
                })[:, len(context_tokens):]
                for i in range(batch_size):
                    generated += 1
                    text = enc.decode(out[i])
                    print("=" * 40 + " SAMPLE " + str(generated) + " " + "=" * 40)
                    print(text)
            print("=" * 80)
```

Figure 3 shows the fully functional and integrated backend for our project. We use sample training data from Wikipedia and Yahoo Chat dataset, which after processing sits > 1.2 GB in size. Due to this massive size of textual data, we shifted to a VM with high processing and computation power. The output sample here is one we generated during training. We found that the model generalises well on contexts it can identify easily, and we decided to use the last layer of the model as a feature extraction metric. Using this last layer, we were able to use smaller datasets to fine-tune our data generation w.r.t a particular niche.

Figure 4 shows the deployed model that uses tensor flow backend and a GPU for processing. A flask server is used for deployment and is shown here calling all the data loaders and tensor flow APIs we used for processing. Each run of the model uses the flask app and re-instantiates the tensor flow APIs.

Figure 3 Sample training and outputs from the model (see online version for colours)

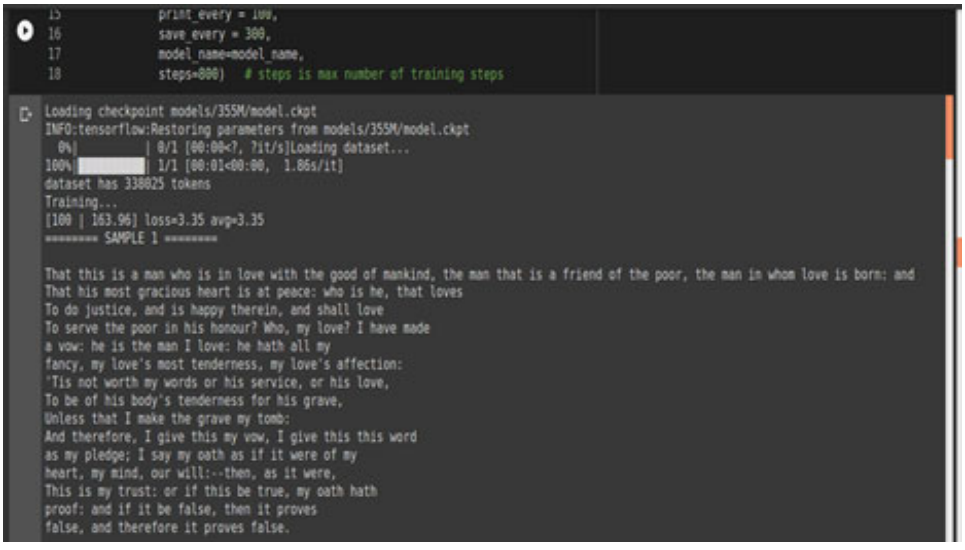


Figure 4 Running flask app from the terminal (see online version for colours)

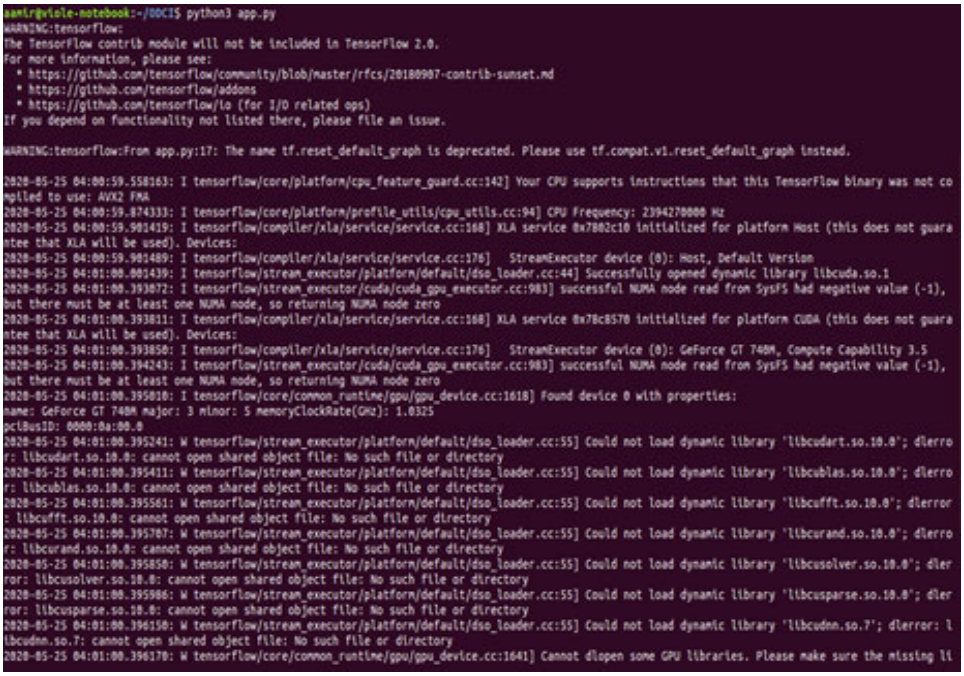


Figure 5 Live flask server on port 5,000

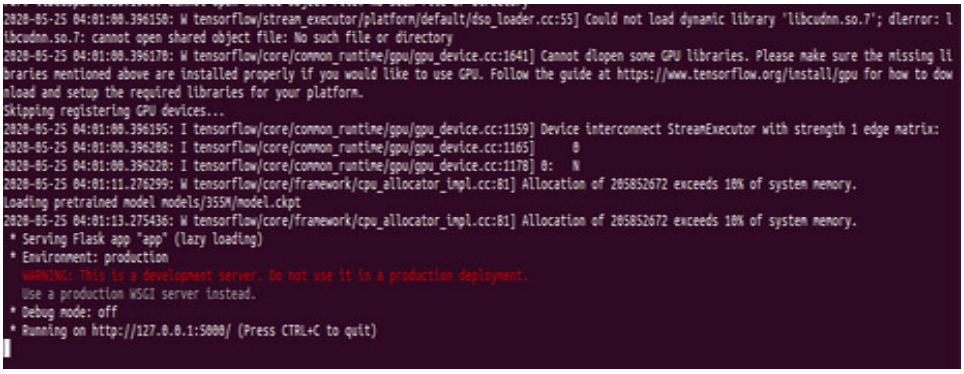
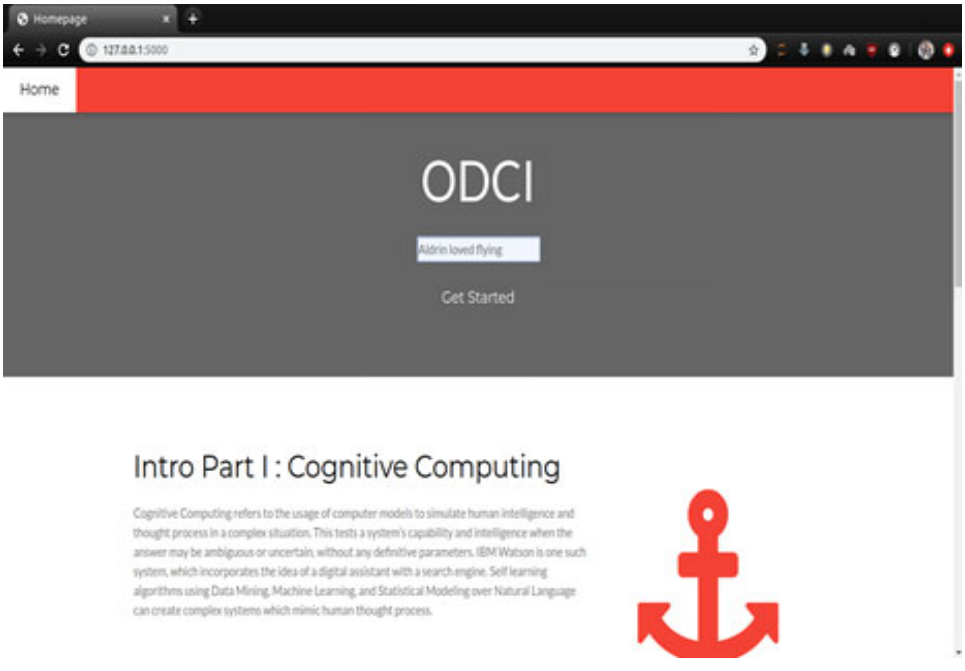


Figure 6 Flask app UI with sample input (see online version for colours)



As we can see in Figure 5, a lazy loading-based flask server is active on the port 500. This port address will act as the home page for the application developed. Using the link on which the server is running, we use the web application for input and output.

Figure 6 shows the flask app UI that we have developed. This serves as the homepage for our webapp. The web application has an input text box, where we can enter a string prefix for story generation.

Figure 7 shows the scrollable part of the homepage which serves as the precursor to the project. Basic information can be seen and inferred from this homepage. The web app has been designed using only HTML and CSS.

Figure 8 shows the model output rendered on the result page of the flask app. This shows the output generated as formatted HTML text, and shows the time taken for

generation + pruning. Due to the massive size of the model, generation can sometimes take up to 60 seconds.

Figure 7 Flask app homepage (see online version for colours)

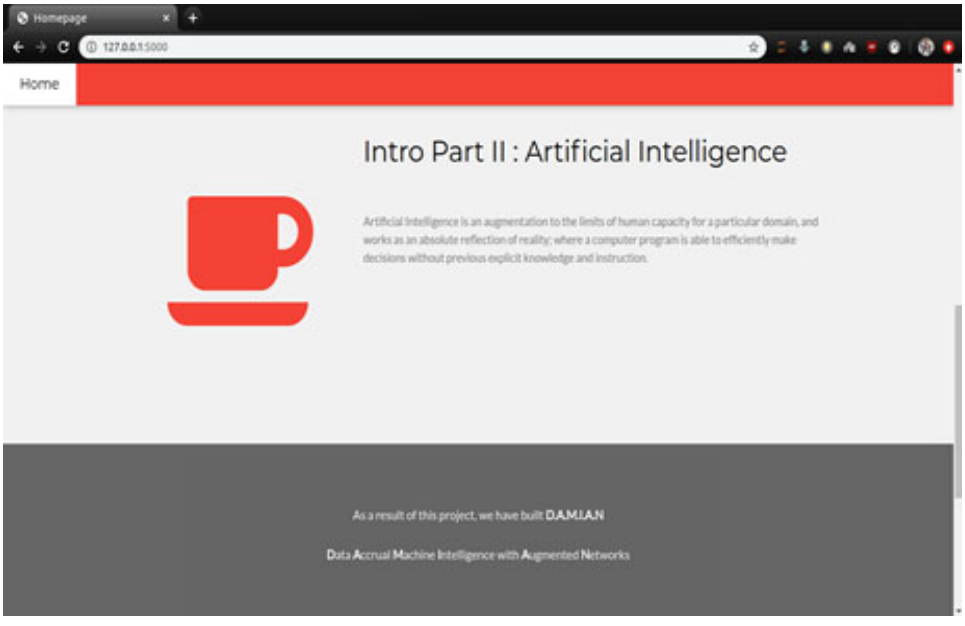
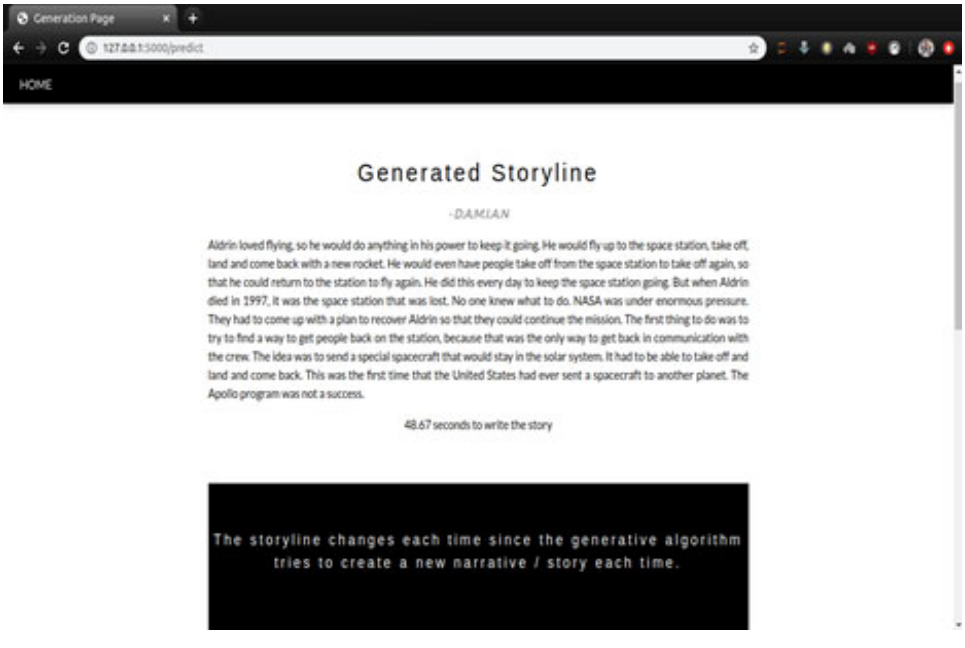
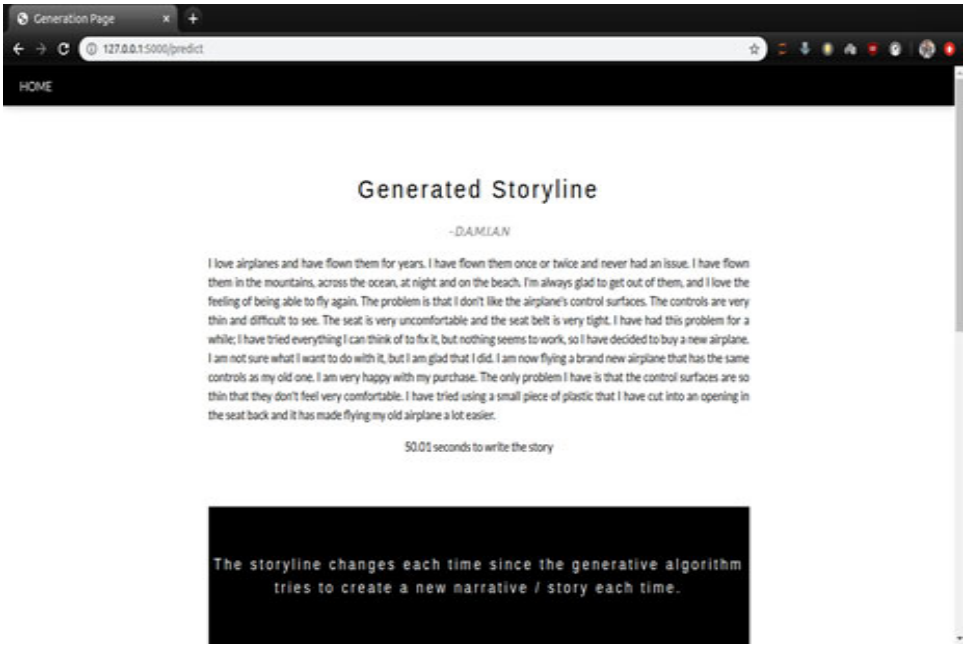


Figure 8 Output for the sample input text (see online version for colours)



This Figure 9 shows us another sample story generated by our program. The result page has been dynamically linked back to the homepage, so that one can input multiple sample sequences without having to run the app from the terminal for every execution.

Figure 9 Another sample story created by D.A.M.I.A.N. (see online version for colours)



5 Conclusions

The text generation works for smaller sequences, smaller input strings and longer text generation constraints heavily penalise the quality and coherence of the text generated. However, using fine tuning metrics for post-processing can help with repetition on longer texts. A better trained model may be able to generalise better as well, so larger models for the same dataset can be augmented by stacking more transformer layers. Transformers have been the standard architectures for state-of-the-art implementations and uses, our project is the first to embark on an implementation for story, script and other similar sequence generation tasks with fine-tuning. This opens avenues to connect better to the modern world as ideas for movie plots, digital writing assistants, etc. Another valid and interesting use case is the augmentation of research data, since researchers often face the problem of having less data to work with. Using this engine, one can generate more sample data that represents similar ideas and context as training data provided. As we have seen, the language model has a marked decrease in quality with smaller prefix texts and longer text generation constraints. The work presents a model to develop a multi-resolution RNN to identify local and global context, develop contextual embedding via transformers to pass into a seq2seq architecture and add heavy regularisation and augment data with reinforcement learning, and optimise via recursive neural networks. All the three strategies can be combined with the Transformer model as the crux. Taking

a text input, we include the position of each text token (for attention purposes). We embed them using transformer embeddings. This will help us recognise local and global context (using multi-resolution RNN). Next, we normalise the output to avoid gradient overflow. We do this using batch normalisation layers. We forward this network and normalise it again and add a transformer embedding lookup. This will convert our embeddings back to textual form, which is human readable. This can be attributed as a side effect to the non-deterministic nature of the engine developed. However, the project still shows promising results with larger models or better fine tuning on datasets. Extending post pruning as a mechanism for text generation for repudiation of text replication in generated sequences is an active interest for our work. Another exceptional application would be training the engine to generate stories/ideas given images - using image captioning as input to describe an image, then writing a story around it. This would essentially simulate not only creativity but also show cognitive intelligence, similar to an AI playing Pictionary.

References

Ahmad, N., Boota, M.W. and Masoom, A.H. (2014). Smart phone application evaluation with usability testing approach', *Journal of Software Engineering and Applications*, Vol. 7, No. 12, p.1045.

Baggio, G., Bassoli, R. and Granelli, F. (2019) 'Cognitive software-defined networking using fuzzy cognitive maps', *IEEE Transactions on Cognitive Communications and Networking*, Vol. 5, No. 3, pp.517–539.

Ballester, P.L. (2019) 'Semi-supervised learning methods for unsupervised domain adaptation in medical imaging segmentation'.

Bhandari, D., Paul, S. and Narayan, A. (2020) 'Deep neural networks for multimodal data fusion and affect recognition', *International Journal of Artificial Intelligence and Soft Computing*, Vol. 7, No. 2, pp.130–145.

Chen, M., Herrera, F. and Hwang, K. (2018) 'Cognitive computing: architecture, technologies and intelligent applications', *IEEE Access*, Vol. 6, pp.19774–19783.

Dragoni, M. and Rospocher, M. (2018) 'Applied cognitive computing: challenges, approaches, and real-world experiences', *Progress in Artificial Intelligence*, Vol. 7, No. 4, pp.249–250.

El Ouanjli, N., Derouich, A., El Ghzizal, A., Motahhir, S., Chebabhi, A., El Mourabit, Y. and Taoussi, M. (2019). Modern improvement techniques of direct torque control for induction motor drives-a review', *Protection and Control of Modern Power Systems*, Vol. 4, No. 1, pp.1–12.

Fiorini, R.A. (2018). The Logic of Cognitive Genetic Structures', in *IEEE 17th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, July, IEEE, pp.306–315.

Huang, K., Ma, X., Song, R., Rong, X., Tian, X. and Li, Y. (2019) 'An autonomous developmental cognitive architecture based on incremental associative neural network with dynamic audiovisual fusion', *IEEE Access*, Vol. 7, p.8789–8807.

Kar, R. and Mukherjee, S. (2021) 'Determination of approximate fuzzy membership function using linguistic input-an approach based on interpolation', *International Journal of Hybrid Intelligence*, Vol. 2, No. 1, pp.4–14.

Kulkarni, R., Kulkarni, H., Balar, K. and Krishna, P. (2018) 'Cognitive natural language search using calibrated quantum mesh', in *IEEE 17th International Conference on Cognitive Informatics & Cognitive Computing (ICCI* CC)*, IEEE, July, pp. 174-178.

Liang, J., Shi, Q. and Zhao, X. (2018). Multi-view data ensemble clustering: a cluster-level perspective', *International Journal of Machine Intelligence and Sensory Signal Processing*, Vol. 2, No. 2, pp.97-120.

Naveed Uddin, M. (2019) 'Cognitive science and artificial intelligence: simulating the human mind and its complexity', *Cognitive Computation and Systems*, Vol. 1, No. 4, pp.113–116.

Noor, A.K. (2014) 'Potential of cognitive computing and cognitive systems', *Open Engineering*, Vol. 5, No. 1.

Radford, A., Narasimhan, K., Salimans, T. and Sutskever, I. (2018) 'Improving language understanding by generative pre-training'.

Shankar, R.S., Deshai, N., Murthy, K. S. and Gupta, V.M.N.S.S.V.K.R. (2019) 'The source of growing knowledge by cognitive artificial intelligence', *IEEE International Conference on System, Computation, Automation and Networking (ICSCAN)*, IEEE, March, pp. 1-6.

Valipour, M. and Wang, Y. (2018) 'A cognitive machine learning system for phrases composition and semantic comprehension', in *IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, IEEE, October, pp.24–29.

Xu, J., Ren, X., Zhang, Y., Zeng, Q., Cai, X. and Sun, X. (2018) 'A skeleton-based model for promoting coherence among sentences in narrative story generation'.

Yogatama, D., d'Autume, C.D.M., Connor, J., Kocisky, T., Chrzanowski, M., Kong, L. and Blunsom, P. (2019) 'Learning and evaluating general linguistic intelligence'.