International Research Journal of Computer Science (IRJCS)
Issue 09, Volume 07 (September 2020)
ISSN: 2393-9842
https://www.irjcs.com/archives

IMPACT OF ARTIFICIAL INTELLIGENCE AND NATURAL LANGUAGE PROCESSING ON PROGRAMMING AND SOFTWARE ENGINEERING

Radha Guha, Ph.D.
Professor of Computer Science and Engineering
radhaguha9@gmail.com,radhaguha@yahoo.com

Publication History
Manuscript Reference No: IRJCS/RS/Vol.07/Issue09/SPCS10083
Received: 29, August 2020
Accepted: 03, September 2020
Published: 14, September 2020
DOI: https://doi.org/10.26562/irjcs.2020.v0709.003
Citation: Radha Guha(2020). Impact of Artificial Intelligence and Natural Language Processing on Programming and Software Engineering. IRJCS:: International Research Journal of Computer Science, Volume VII, 238-249. https://doi.org/10.26562/irjcs.2020.v0709.003
Peer-review: Double-blind Peer-reviewed
Editor: Dr.A.Arul Lawrence Selvakumar, Chief Editor, IRJCS, AM Publications, India

Abstract: The recent beta release of artificial intelligence natural language processor GPT-3 by OpenAI in June 2020, is triggering a lot of excitement in machine learning community as well as in software developers. The full form of GPT is generative pre-trained transformer. GPT-3 is a general purpose transformer based text generator, largest deep neural network model ever built by any artificial intelligence (AI) company. The mind blowing capability of the transformer based natural language processor (NLP) model is fascinating as well as awe inspiring to the programming community. It is a language model that takes input text in natural language and predicts output text in another form. It can write fiction, generate programming code and design websites, prompted by natural language description. This paper investigates how advancement in AI and NLP are impacting programming, software development process and software products and services. This paper suggests how software developers have to reposition and re-skill in this disruptive technology era when an AI language model like GPT-3 has answer to every question posed in natural language. In this paper first the triumph of AI in the last two decades culminating to the latest GPT-3 model is described and its capabilities are looked into. Then evolution of programming languages and software development methodologies in the changing scenario of parallel and distributed computing of big data, cloud computing, web services and AI integration are followed. What is observed is that now a day computer applications and mobile apps are just integration of conglomerate of components accessed over the internet through application programming interfaces (APIs). Thus the incumbent programmers are informed about this digital transformation, and machine learning and AI NLP advancement at a very rapid space; to prepare themselves for future software engineering needs and demands.
Keywords: Artificial Intelligence (AI), Natural Language Processing (NLP), Machine Learning, Programming Evolution, Software Engineering Evolution, GPT-3

I. INTRODUCTION:

History of Artificial Intelligence

Before talking about artificial intelligence let's see what is natural intelligence. Natural intelligence of human incorporates perceiving and analysing visual scene, understanding and speaking natural language like English, computing arithmetic and logic operations, having memory, continuously learning from new experiences and inferring for generalization and decision making. In artificial intelligence, we strive to teach a computer to mimic human intelligence. Human brain consists of approximately 100 billion neurons and those neurons are connected with each other with 100 to 500 trillion synapses. The neurons process the external stimuli from the nervous system and transmit the information through the synapses to other neurons to carry out all the complex tasks what we call natural intelligence.

The history of artificial intelligence [1], [2], [3], [4], [5], [6] is like this. In 1943, McCulloch and Pitts theorized a computing element is similar to a biological neuron of human brain. An illustration of a biological neuron and an artificial neuron is shown in Figure 1.

In the artificial neuron, output is a function (either Sigmoid or tanh or ReLU) of sum of *n* inputs (x_i) multiplied by connection strengths (w_i) given by: output $=f(\sum_{i=1}^{n} x_i w_i)$. McCulloch and Pitts, postulated that a network of these computing elements [Figure 2] with adjustable weights of interconnects resembling synapses in human brain, will be able to solve any computable function.

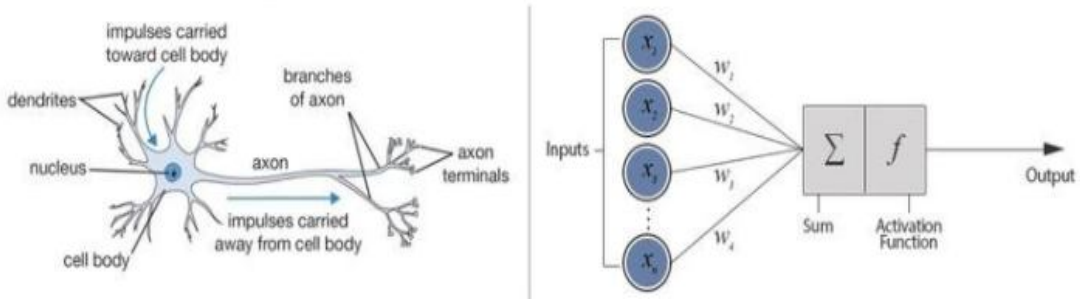

Figure 1: Biological Neuron vs. Artificial Neuron

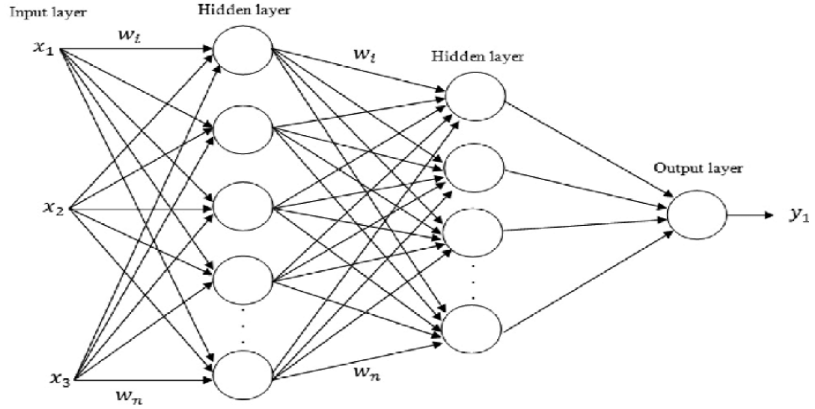

Figure 2: Multi-Layer Perceptron (MLP), a Neural Network Architecture

In the mean time, after the invention of silicon transistors in 1947, fist time digital computers were built. Since then advancement in computer hardware and software in tandem is creating computing revolution. In 1950, American computer scientist, Alan Turing defined that if a computer passes 'Turing Test' it can be declared intelligent. If a human user can be fooled after interacting with a computer, as if he has interacted with another human then the computer passes the Turing Test. In 1956, John McCarthy, professor of Stanford University coined the term 'Artificial Intelligence' (AI) for the first time and founded the AI discipline. He said intelligent features like learning can also be imparted to a computing machine. In 1962 Frank Rosenblatt explored that network of perceptrons, a mathematical model of biological neurons as in Figure 2, are capable of learning and pattern recognition in input data. They then employed this NN architecture to solve simple puzzles and play computer games.

Even though AI research started with bells and chimes during those days in 1950s and 1960s, it could not make much progress for the dearth of computing power. In the last two decades, AI research in deep neural network learning has picked up the steam with the advent of more powerful computers and parallel processing hardware [7], [8], [9]. Sequential processor i.e. Intel CPU's performance (MFLOPS) could be increased steadily only till the end of the previous century. Then parallel processing in graphics processing unit (Nvidia GPU, 1999) came to existence, whose throughput is much higher than CPUs. Then Google and Yahoo came up with big data processing hardware platform of Hadoop (2006), MapReduce and Spark etc. for distributed parallel processing of big data. Now Google has created application specific integrated circuit (ASIC) hardware frame work called tensor processing unit (TPU, 2018) as AI accelerator, specifically for neural network based machine learning (ML) algorithms. Machine learning and deep learning algorithms, needing matrix multiplication and addition are parallelizable and benefit from the parallel architecture of GPU and TPU. To ease the implementation of complex ANN based ML models, Google has come up with TensorFlow software platform in Python programming language, which is now open source from 2015 onwards. For large scale production deployment of software, Python API PyTorch exploits the power of GPU and TPU.

Also in the beginning of 1990s, Internet opened the door for accessing information from anywhere at any time. This started a smart revolution in the world with digital transformation, use of Internet of Things (IOTs) [10], [11], smart sensors, machine learning and deep learning algorithms to build smart products and services. Since then the amount of unstructured text data in the internet catapulted and it is growing with exponential velocity with every passing day. From late 1990s we are seeing smart AI products like robots for autonomous control in the industry.

Also software agents are getting used in the search engine for collecting relevant information by crawling in the internet. Businesses now can employ ML algorithms to analyse this big historical data to take wise decision for future.

Now a day, every major technology companies are investing heavily on artificial intelligence products and services to build a smarter cognitive environment that blurs the gap between man and machine. In 1996, IBM's DeepBlue computer defeated then reining world chess champion Gary Casparov. In 2005 Honda's ASIMO was the best humanoid robot. In 2011 IBM's Watson, a question answer Chabot defeated the champion of the general knowledge game show Jeopardy on American national TV.

From 2000 onwards, several social media networking sites like Facebook, Twitter and Linkedin etc. have popped up for online engagement of users. Born in 2004, Facebook is the most popular free social networking site to stay connected with friends and family. Facebook manages two and a half billion users with hundreds of Peta bytes (PB) of texts, clicks, images and video data in their gigantic data storage centre using AI based technology. Machine learning and deep learning algorithms targets user specific advertisement for Facebook's business and also can detect bad content in the post and comments of users to take necessary action.

From the beginning of 2010, virtual assistants like Amazon's Alexa, Apple's Siri and Google Assistant are part of our daily lives. Alexa, and Google Assistant are voice activated virtual assistant which can control smart devices of a smart home, read news papers, play music from the internet, book Uber ride, call phone numbers, shop online and recommend things. When the above two products are for home environment, Apple's Siri is locked in the phone and can do most of the above tasks as well as give step by step driving direction to a driver. Microsoft's Cortana is also similar to Alexa, Siri and Google Assistant. IBM's Watson, the chatbot, can also assist in business processes like customer support. DJI, a Chinese company is making Drones with AI technology that use object recognition algorithm in images to avoid collision. Drones can be used for remote surveillance or for pizza delivery. Google Cars, Tesla and many other companies are building autonomous vehicle that will sense the environment with radar, lidar, GPS, odometer, computer vision and AI navigation algorithm and drive itself.

Google's DeepMind is a general purpose AI model capable of doing variety of tasks. In 2016, DeepMind's AlphaGO took a giant leap in AI success by beating the world's champion in Chinese board game GO. GO is much harder than Chess, because it has many more possible moves than Chess. AlphaGo uses reinforcement learning and mimics the learning process of a human brain. DeepMind is also impacting the healthcare industry hugely with predictive analysis using AI and ML. Amazon's Web service (AWS) and Microsoft's Azure cloud service manages business processes using machine learning and artificial intelligence. All the above mentioned products and services need smart embedded software with search and analytical skills for decision making.

The digital transformation, use of Internet of Things and use of social networking sites are generating a lot of unstructured or semi structured text data. Thus this era is called big data era with Zetta Bytes (ZB) of data in 2020. Eighty percent of the data in the internet is text. Thus AI products and services that retrieve information from the internet and analyze them need natural language processing (NLP) and understanding. NLP is a new field of study of AI in the big data era and is more important than structured numerical computation. Now the latest natural language processor GPT-3 released in June 2020, has been trained by the entire internet text data. Right now to prevent the misuse and malicious activity of the powerful model only the API is released through which the model can be accessed. In the next section we will follow the journey of NLP to reach GPT-3 success.

The organization of the paper is as follows. In Section 2, NLP history and current state of the art NLP model GPT-3 is explored. Section 3, elaborates the philosophy behind evolution of programming languages from first generation to fifth generation and of changing trends of programming. Section 4, investigates software engineering methodologies and models from 1960s to till today. It also addresses the challenges of digital transformation and penetration of machine learning and artificial intelligence field in software engineering to the incumbent programmers. Section 5, concludes the paper.

II. JOURNEY TO STATE OF THE ART AI NATURAL LANGUAGE PROCESSOR (NLP): GPT-3

Structured data mining field is well established now with many machine learning techniques viz. Classification, clustering, association rule mining and regression analysis etc. But today eighty percent of the raw data hosted in the internet is unstructured text data in natural languages. Natural language is a set of vocabulary that abides by a set of rules or grammar to construct sentences. So a new field of AI, named natural language processing (NLP) or text mining has just emerged to teach a machine to understand human communication. There are many applications of text mining like information retrieval, document classification, named entity recognition, Twitter sentiment analysis, customer relationship management (CRM), cyber crime prevention, contextual advertising, text summarization, language translation, question answering and text generation etc.

Automatic text data mining is very challenging because natural language like English has the problem of synonymy and polysemy. For example “ocean” and “sea” are synonymous but how does a computer know that? And the word “bank” is polysemous, it may mean financial institution or river bank or a heap and a machine has to understand its right meaning depending on the context of the word. Unstructured text corpus is first converted into structured term document matrix (TDM) for downstream numerical computations. Each cell (*i*, *j*) in TDM contains a numeric score for term (*i*) in document (*j*) equal to term frequency (TF) in a document, multiplied by inverse document frequency (IDF) [14], [15]. This is a vector space representation of words where word ordering was not considered naively; and is called a bag of words concept. Latent Semantic Analysis (LSA) [12] [15] in 1993 employed matrix decomposition algorithm like singular value decomposition (SVD) to carry out document to document similarity or term to term similarity for information retrieval from the corpus. In 2003, Latent Dirichlet Allocation (LDA) [13] [15] also employed another matrix decomposition with some prior topic distribution parameters alpha (α) and beta (β), and it performed better than LSA. LSA and LDA algorithms works for information retrieval, document classification, sentiment analysis like tasks. But for other NLP tasks like question answering and language inference, deeper understanding of word context is required.

Since then a lot of advancement in natural language understanding has happened where distributed semantic and contextual occurrences of words are utilized for better results in language modeling. Word representation needs to capture not only semantic meaning but also higher level concepts like anaphora, long term dependencies, agreement and negation etc. An example of anaphora is in this sentence; “This plane has some limitation in landing but landed safely”. An example of long term dependency is in this sentence; “I grew up in France, right now I am visiting India, I speak fluent French”. The fascinating capability to understand natural language in machine is happening today due to the rapid advancement in neural network architecture viz. deep learning and availability of enormous parallel processing power of GPUs and TPUs. Various neural network algorithms like Word2Vec (2013) [16], Global Vector (2015) [17], Long Short Term Memory (LSTM) RNN deep neural network (2015) [18], [19], ELMO (2017) [20], Transformer [21], BERT (2018) [22], and GPT-3 (2020) [27] etc. have been released in succession within a few years as shown in Table 1.

Table 1: Neural Network Algorithms for NLP

2013	2014	2014	2017	2017	2018	2018	Feb. 2019	Jun. 2020
Word2Vec	Glove	Seq2Seq LSTM	ELMO	Transformer	BERT	GPT-1	GPT-2	GPT-3

In Word2Vec (2013) algorithm [16], proposed by Mikolov et al., a single layer shallow neural network with a fixed size context window (10 to 20 terms/grams) for a target word is used to transform each word into a lower dimension (< 300) dense vector of real numbers called word embeddings. In a ***n***-gram model a document is broken down to tokens of size ***n***, to count each words surrounding ***n*** words as its context. GloVe (Pennington et al., 2014) [17] was another word embedding algorithm where not only local context but also global context from global word to word co-occurrence matrix was considered and gave better accuracy than Word2Vec in downstream NLP tasks. In word embeddings similar words will have closer vector representation. A famous example is vector[king] – vector[queen] = vector[man] – vector[woman]. Both Word2Vec and GloVe are unsupervised algorithm as no human made labels are required with the raw text.

A language model predicts the next word given its context words, with some probability score. But English sentences have variable length, so accordingly the context window size should vary. For context aware word meaning disambiguation, feed forward multi layer perceptron (MLP) architecture like Figure 2 is not suitable, because MLP has fixed size input structure. Conceptually, context aware word meaning computation can be implemented in recurrent neural network (RNN) [18], [19] architecture because it has a loop for information to persist for any length as shown in Figure 3.

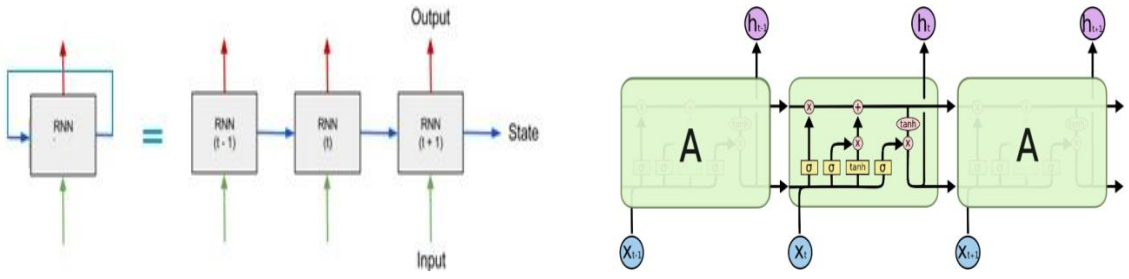

Figure 3: Simple RNN (left) and LSTM –RNN (right)

Thus deep learning sequence to sequence RNN architecture was constructed to generate an output text sequence from an input text sequence. RNN can encode a sentence meaning and can predict the next word. One drawback of RNN is vanishing or exploding gradient problem of gradient descent algorithm, for which long term dependency in the text could not be captured.

To overcome that gated recurrent unit (GRU) and long short term memory (LSTM) network architecture (Figure 3) were devised in 2015. The memory cell in LSTM can keep information for longer period of time, thus capturing long term dependency of words. In LSTM-RNN, sequential words in the context are fed into the network one by one taking several time steps. LSTM cell consciously decides what to forget and what to retain. The LSTM-RNN is used for large number of tasks like time series forecasting, speech recognition, machine translation and image captioning etc. The drawback of any RNN based models is that they are slow and cannot be effectively parallelized for GPU architecture.

A uni-directional LSTM can only capture word dependency in one direction i.e. next word depends on previous word sequence. But word relation can exists in both directions i.e. prior word may depend on future word. As for example to fill in the blank in this sentence "The women went to the store and bought a __ of shoes", one has to read left to right and right to left both. ELMO (embeddings from language models) [20] uses bidirectional LSTM architecture devised in 2017, for context aware representation of words. ELMO is a language model which can predict the next word as well as previous word. ELMO out-performed all other language models that came before ELMO, in question answering, sentiment analysis and named entity recognition tasks.

A different approach to handle long term dependency was published in 2017's paper "Attention is all you need" by Google Brain research team [21]. The simple architecture in this paper is a transformer (Figure 4) for mass parallel computation on Google GPU and TPU. The transformer is based solely on attention mechanism, abolishing previous state of the art LSTM-RNN based architecture. Transformer has an encoder block and a decoder block to map input sequence to predicted output sequence. Here all words in the context are passed to the encoder block simultaneously after position encoding. The encoder and decoder blocks both consist of multi-head attention mechanism and feed forward network and are stacked ***Nx*** times as shown in Figure 4. The multi-head attention mechanism considers different parts of input sequence simultaneously. In deep learning, normalization is very important and added in every step after aggregation, so that all calculated numbers remain in the scale of zero to one. The encoder produces an internal representation of each word. The decoder block takes this internal representation and previous output to produce the current output one by one. This transformer architecture handles long-term dependency better than LSTM-RNN. Also this transformer architecture takes less time to train and perform better in wide variety of downstream NLP tasks than all other previous breakthroughs.

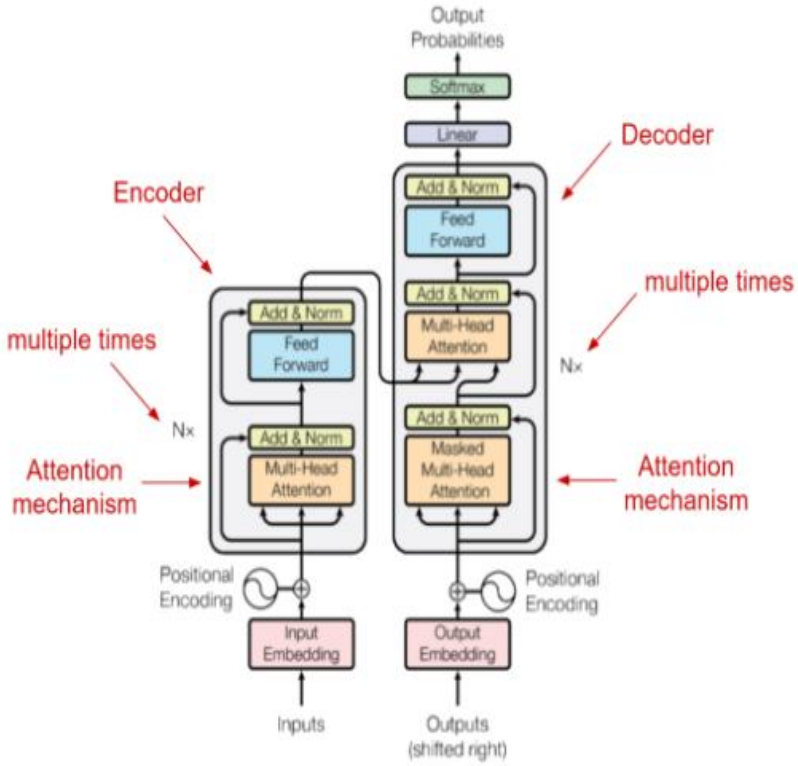

Figure 4: Transformer Model Architecture [21]

BERT (bidirectional encoder representation from transformers) [22] is the next breakthrough from Google research again as a pre-trained general purpose language model. Even though the internet is the repository of enormous amount of data they are not labeled data. To have task specific labeled data for example for sentiment analysis, is a problem. So BERT is pre-trained as general purpose language model with huge amount of unlabeled data. Later the model adapts quickly for fine tuning for a specific task with few labeled datasets. This is called transfer learning. Also as BERT is only a language representation model it needs only the encoder part of the transformer model shown in Figure 4, and not the decoder part. BERT is also bi-directional because to fill in the blank one has to read left to right and right to left both. So BERT is bi-directionally trained with few masked tokens to gain deeper sense of word context. BERT performs better in task like question answering and natural language inference (NLI). Now a day we see NLI implementation in smart phones and Google search window, when the next word is suggested automatically based on what we have typed so far. The trained BERT model can be downloaded free from GitHub, but it needs powerful hardware like GPU and TPU to run.

Recent research [22], [23], [24], [25], [26], [27] has shown that if the language model size is increased in number of parameters and training dataset size, then a task agnostic general purpose model learns better. Then the model can be fine tuned with few task specific examples for better transfer learning. This is similar to how human learns by seeing only a few examples of any particular task. So OpenAI, an AI research and deployment company has built a humongous auto regressive text generation language model named GPT-3 which has 175 billion parameters. Parameters are the weights of the 175 billion connections between the computing nodes and these numbers indicate the enormous complexity of the model. GPT-3 released in June 2020, has the same architecture but is 100 times bigger and is far better than its predecessor GPT-2 built in 2019. GPT-3 is again a transformer based general purpose unsupervised language model whose full name is generative pre-trained transformer. The model is trained with broad swath of topics comprising of 500 billion tokens from the Web Text. The enormous size of GPT- 3 enables it to tackle huge number of auto complete tasks. During pre-training the model learns broad set of skills of pattern recognition from input data. This pre-trained model adapt quickly to any downstream task with zero, one or few shot examples. The context window size in GPT-3 is 2046 tokens which can fit 10 to 100 examples for few shot learning. During text generation GPT-3 does not need the encoder block anymore, it only works with stack of decoder blocks from the pre-trained transformer.

For a question answering task, if a sentence like "The **trophy** does not fit into the brown **suitcase** because ***it*** is too **small**" is input to the GPT-3 model and is asked the question what the word "***it***" refers (attention) to, the model answer correctly with the word "suitcase". If the same sentence is slightly modified to "The **trophy** does not fit into the brown **suitcase** because ***it*** is too **large**" and the same question is asked, the model now correctly answers with the word "trophy". Human evaluator can be fooled to distinguish GPT-3 generated coherent writing samples, whether it is synthetic or written by human. The amazing capability of GPT-3 can be applied to many domains, from code generation to composing poems to simple arithmetic addition and subtraction as listed in Table 2. Gpt-3 is the first step towards steering development in artificial general intelligence (AGI) for machines.

Table 2: NLP Utilities

Domain	Use
Software Engineering	Code Auto completion
Writing	Grammar Assistance Auto completion – Assisted writing Language translation
Art	Creating or aiding literary art Poetry generation
Entertainment	Gaming Chat bot
Health	Medical – Question answering system
Mathematics	Upto 5 digit addition, subtraction

Left side of Figure 5 shows performance comparison of OpenAI's GPT-3 model of different size variants with several other model scores viz. i) random guess, ii) yester year's Google BERT language models and 3) human score on general language understanding evaluation (GLUE) task. It can be seen that when BERT is a fine tuned model, GPT-3 model is general purpose and has been fine tuned only with zero shot, one shot and few shot transfer learning. Even then the largest GPT-3 model with 175 B parameters performed better than the fine tune BERT-Large and BERT++. The figure also shows that as the GPT-3 model size increases its SuperGLUE score increases and also few shot transfer learning performs better than zero shot and one shot transfer learning.

Right side of Figure 5 shows accuracy score of various size GPT-3 models on two, three, four and five digits addition and subtraction tasks. Here also as the model size increases accuracy score increases for all tasks. Figure 5 also shows that two digit arithmetic's accuracy is more than that of five digit arithmetic.

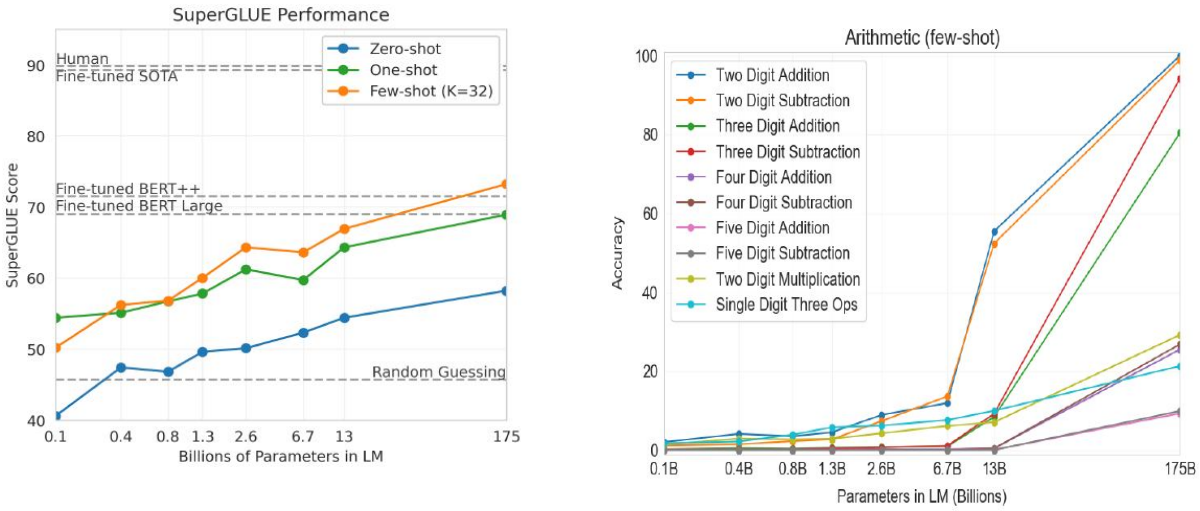

Figure 5: GPT-3 Performance Comparison on NLP Tasks [26]

The GPT-3 model size is order of magnitude bigger than all previously built state of the art models of NLP as shown in Figure 6. The GPT-3 model is too big, requires several days of parallel processing on most powerful GPUs and TPUs to train. The model is very expensive to run and needs very high expertise in AI domain to develop and deploy. As small and medium size organization will not be able to handle this big model, and to prevent misuse of the powerful impact of the model, instead of releasing the model in public domain the company has released the API in public domain. The interface is very simple and very flexible so that anyone can use it for variety of tasks.

Now the question is, eventhough GPT-3 like language model is a very useful tool for accomplishing so many different tasks; does it have true intelligence of a human? For example if GPT- 3 is asked the question "How many eyes a spider has?", it can give the correct answer as "A spider has eight eyes". But if GPT-3 is asked the question "How many eyes a foot has?", it gives the answer "A foot has one eye". So GPT-3 model clearly does not understand that the question itself is meaningless. Currently GPT-3 model has 175 billion parameters, where as human brain has 100 billion neurons and 100 to 500 trillion synapses. But the rate at which deep learning artificial neural network language model size is increasing it will not take many years to build even more powerful language models that mimic human intelligence more closely.

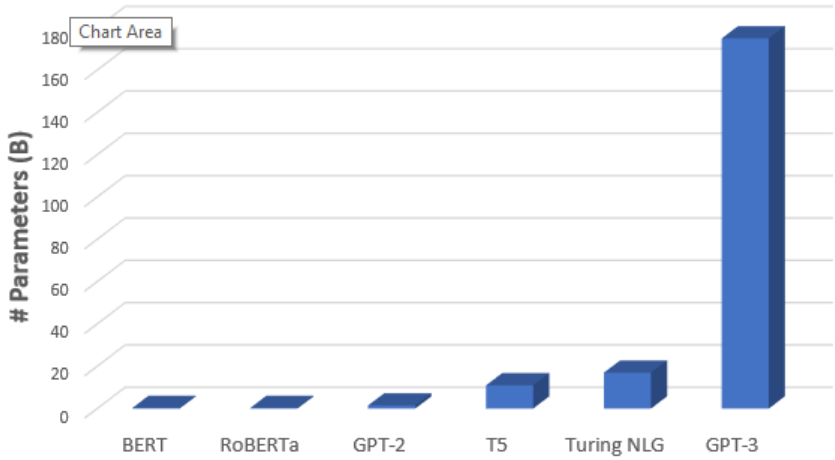

Figure 6: Language Model Size Comparison [27]

2.1 PROGRAMMING IN THE AGE OF ARTIFICIAL INTELLIGENCE

Any programming language is a set of vocabulary and a set of rules or syntax to write a command using variable names, mathematical operations, data structures, data flow and control flow rules to instruct a computer to accomplish some tasks. A set of such instructions make a program or software.

Computer scientists write software programs so that computers can perform a task more quickly and effectively than a human. There are almost seven hundred programming languages that have evolved since the time we have started using digital computers from 1950s. Some of the programming languages are Fortran (1957), LISP (1958), Cobol (1962), Basic (1964), C (1972), Ada (1984), C++ (1985), Python (1991), Java (1995), JavaScript (1995), and R (2000). These languages can be grouped in several categories as per their commonalities and are called programming paradigms. The programming paradigms are i) procedural language ii) object oriented language iii) functional language iv) scripting programming language and v) declarative language. Choosing a programming language for a project primarily depends on the type of applications i.e. whether it is a web application or mobile app or embedded application. Other factors are size and complexity of the application, company's preference, maintainability and security issues etc.

Knowing the philosophy behind emergence of so many programming languages makes things clearer. Programming languages have evolved to make them more programmer friendly and to manage software development lifecycle more efficiently. From first generation to fifth generation (1GL to 5GL), programming languages have become closer to human language and easier to program but at the cost of slower execution time. Low level, 1GL was machine code in terms of binary 1s and 0s. Intermediate level 2GL was assembly code which can handle very small program but can fine tune performance as per underlying hardware. 1GL and 2GL languages are machine dependent.

3GL programming languages like C, C++, Java are called high level programming language as they are more English like natural language. 3GL programming languages are general purpose, more machine independent, more programmer friendly and write a series of instructions to instruct the computer what to do and how to do it both. Thus 3GL is also called procedural language. Major system software written in C language was UNIX operating system. C++ and Java are also object oriented programming languages for modularity, scalability, code reuse, portability and maintenance superiority. JavaScript, Node JS, Ruby, Python, Perl are scripting languages; used to integrate client and server side applications written in different languages. Scripting languages are for run -time environment and they don't need to be precompiled. Scripting languages help in creating dynamic web pages with interactive visualization and they are easy to learn also.

In contrast to 3GL, 4GL programming languages are domain specific like SQL for database programming and PROLOG for logic programming in artificial intelligence. 4GL is non procedural as it gives instruction to the computer about what to do but not how to do it, so 4GLs are called declarative programming languages. 4GL is also called very high level programming language as they are more close to human language.

Perl, PHP, Python are mix of 3GL and 4GL. SQL compiler and Python interpreter are written in C language. Python is very flexible and general purpose programming language. Python has a huge inbuilt library functions organized in packages like NumPy, Pandas, SciPy, SciKitLearn, MatPlotLib, TensorFlow, PyTorch, Teano, NLTK, SpaCy etc. for doing data analysis, scientific computing, statistics, machine learning, predictive analytics, natural language processing and visualization very easily with few lines of code. Best programming languages for machine learning and natural language processing are Python, R, Java, JavaScript and Scala.

Programming languages Clojure, LISP, Haskell and Scala etc. take functional programming (FP) approach which is also declarative programming. Functional programs contain only functions where there is no order of execution for the functions. They are suitable for parallel and distributed computing. FP is also easy to debug. Disadvantage of FP is that it may be easy to write a single function but combining many functions to create an application is a difficult concept for programmers to learn.

5GL is developed to eliminate the need of a programmer altogether. Any user should be able to get their task done in the computer without having any programming skills. A user only needs to worry about what problems need to be solved and what constraints need to be met. Examples of 5GL are Mercury, KL-One, OPS5 etc., used for artificial intelligence research. Needless to say again, from 1GL to 5GL, programming has become easier and for the same task, size of the program has reduced manifold but execution time has increased.

To write a program a programmer should take a modular approach so that code reuse is possible, code maintenance is easy and the program has minimum memory foot print. Now a day there is many open source software that can be freely accessed and whose source code can be modified and shared. There are many programming repositories like GitHub for free code and communities like Stack Oveflow for debugging help. Programmers especially students just copy and paste open source codes to accomplish their tasks quickly without much knowledge about the code. They tweak the code to the extent that the program runs without optimizing the code in fear of that the program will not run. As a result what they create is called spaghetti code which is very difficult to maintain and this culture is called cargo cult programming. Moreover from the late 1990's Web Services [28], [29], [30], [31], [32] hosted on remote computer or server can be accessed by clients on different computers over the internet for data exchange. This allows distributed computing and composite application development by another web server or mobile app.

Web services can be written on different platform and programming language from that of client applications. But they are interoperable because the data exchange format is standard JSON or XML over HTTP web protocol. Protocols like SOAP (simple object access protocol, 2003) and REST (representational state transfer, 2000) application programming interface (API) guidelines are there for client server communication. REST API is more popular because it takes less band width whereas SOAP is more robust and secured.

Also to eliminate software development complexity like errors, defects, test automation and development time etc. there are several AI ML and NLP applications have emerged that are generating code automatically, viz. Bayou (2018), Sketch2Code (2018), DeepCode (2016), Embold, PyCharm (2010), PyLint, CleverCommit (2019) and Kite etc. Bayou has trained artificial neural network with millions of lines of human written Java code from GitHub. The trained model just needs few keywords of a program and a query to predict the right program code a user wants. DeepCode uses symbolic AI for code review to find bugs and vulnerability in the code. Sketch2Code will generate working HTML solution from hand-written notes and drawings of website design. PyCharm is an intelligent IDE which integrates web technologies like HTML, CSS, JavaScript, CoffeScript etc. and database technologies like SQL Oracle, PostgreSQL in Python to create web application using database. The IDE helps debugging code easily and give auto complete code suggestion and simplify Python application testing.

So we can see that programming and software development is very different now than it was 20 or 30 years ago. AI automation in programming is for better quality economical software development and improved user experience. AI products and services need a lot of real time embedded software for autonomous IOT systems these days. So programmers have to brace this disruptive technology and they have to be adaptive to learn the new technology as they evolve. Programmers new focus expertise area should be in AI components viz. linear algebra, probability, statistics, calculus, data science, machine learning algorithms and natural language processing to make AI products and services at present and in future. Following Luddite fallacy, victory of AI, ML and NLP may not increase unemployment but it will surely need different skill sets for the programmers and software developers in very near future. In a nutshell AI requires smarter developers for smarter applications.

2.2 SOFTWARE ENGINEERING METHODOLOGIES IN THE AGE OF AI

In this section past and present of software engineering is followed and future is speculated. Software engineering is defined as a systemic and disciplined approach to translate user requirements to a computer program using scientific knowledge and creative design that is built economically and works efficiently and reliably. Software development life cycle (SDLC) comprises the following steps viz. requirement gathering, planning, design, coding, testing, deployment and maintenance. As during development software is intangible software engineers create associated documents of every step of development. Traditionally following project management manifesto, software developers take linear and sequential approach called Water Fall model (Figure 7). In Water Fall model software development life cycle starts from requirement gathering, then design, then coding and then deployment of the whole project. But for a large and complex project involving many developers this approach failed to deliver quality software as customer dissatisfaction detected at last is hard to repair in upstream steps. Developers today (late 1990s onwards) favour Agile method which is an iterative process of developing smaller chunk of use cases at a time and taking customers approval immediately. Agile method is more flexible in responding to change demanded by customers in design, coding and testing.

Figure 7: Comparison between WaterFall, Agile and DevOps Model

ISSN: 2393-9842
https://www.irjcs.com/archives

In Agile model, software development is incremental and in rapid cycles with smaller chunk of functionalities but it still creates a gap between development team and operation team as the software is deployed at the end as a whole. To bridge this gap most modern methodology named DevOps [33] is created which also deploys smaller chunk of functionality as soon as it is developed and tested. Continuous integration (CI) and continuous delivery (CD) is the key process in DevOps methodology. Although almost same as Agile, DevOps has added benefit of less time to market, improved customer satisfaction, better service quality and increased productivity. Figure 7 depicts difference among the three models viz. Water Fall, Agile and DevOps.

In recent years AI and ML have progressed rapidly specially through deep learning neural network and more parallel processing power of GPUs and TPUs, The origin of AI being in computer science discipline, initially its applications have been in computer science domain applications viz. face recognition, language translation, robotics and consumer products like Alexa and Siri etc.. But the latest invention of unsupervised general purpose deep learning in GPT-3 has the capability of quick invention or innovation in wide range of other domains like chemistry, physics, biology, medicine, neurology, space research, agriculture, architecture, retail, finance and law and order [34], [35], [36], [37], [38], [39]. These innovations will have profound impact on society and lead to a smarter world. AI will boost product and service quality, it will help businesses to increase productivity and innovation, thus having huge impact on economic growth. Companies that will pave way to AI enabled transition to smarter world will change many job descriptions and will give rise to competition to acquire the big data that has the hidden treasure in it and to use AI enabled tools. It is clear that few tech giants like Google, Microsoft, OpenAI will make these expensive AI models and other companies will use them from cloud through RESTful API or use the open sourced model.

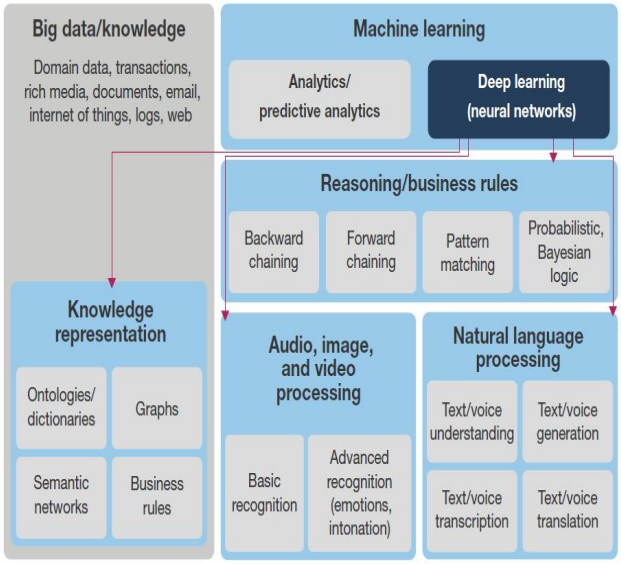

Figure 8: AI Scope in Application Development [34]

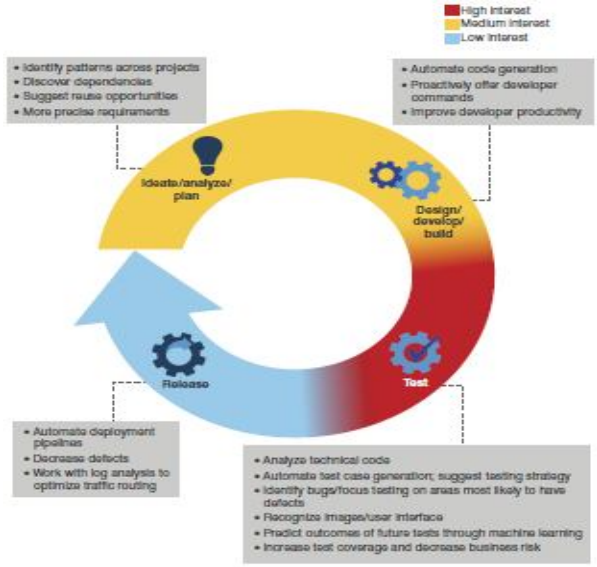

Figure 9: Use of AI at Different Stages of SDLC [34]

Figure 8 taken from Forrester research shows scope of various AI, ML and NLP algorithms to process big data and to impact business growth. Right now USA and China are leading countries in AI research and investment. Market research shows $2.1trillion values of AI enabled tools will be marketed by 2021. Figure 9 also taken from Forrester Research, shows how AI going to upgrade software development at its various stages. Some AI impact is low, some are medium and some are high impact. AI automation can be applied to software development process or as software product and service. After requirement gathering the machine learning algorithm can be applied for requirement categorization. Then AI natural language processor like GPT-3 can be used for automatic code generation from software module design. AI can help in generating automatic test cases, in detecting bugs in code for the modules and can predict defects in module integration. AI and ML can be applied to products that can perceive scene, understand human speech and speak in natural language. AI and ML can create intelligent service that can predict future and help in decision making.

Even though application of AI in software development process adapts well to customer requirement understanding, to faster code development and to better quality software design it also embeds new risk in the software. The promise of AI entices software developers to incorporate AI products and services as intelligent subsystems of the software without knowing how to fix them if they malfunction. It is said that deep learning AI models are like black box and it is not so evident how the input maps to output.

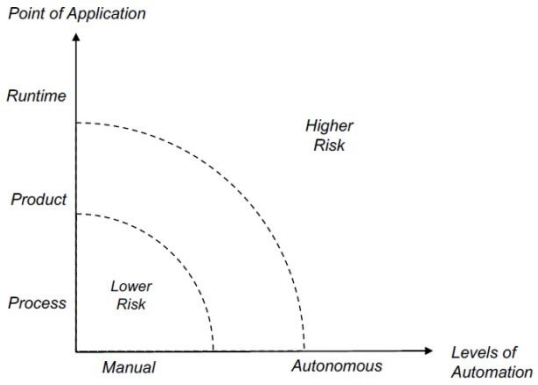

Figure 10: Risk Factor in Level of Automation and Point of Application [35]

It is also said that AI model can produce biased output if the input data is not right or contaminated. In reference [35] author has depicted risk level of applications where AI has been incorporated as shown in Figure 10. In Figure 10, ***x***-axis shows level of automation (LA) and ***y*** axis shows point of application (PA) of AI. Level of automation is in the scale of 1 to 10. Scale 1 is low level of automation where human has full control in deciding next level of action and scale 10 is highly automated where software automatically decides next level of action and informs human if it is needed. Point of application (PA) of AI can be either process, or product or runtime. The figure shows as PA and LA increases risk of automation increases. Thus software companies should take risk according to their AI expertise to leverage AI benefits.

III. CONCLUSION

In this paper changing trend of programming and software engineering is foreseen as AI and NLP are making very rapid strides. Following Luddite fallacy, victory of AI, ML and NLP may not increase unemployment but it will surely need different skill sets for the programmers and software developers in very near future. As AI NLP will generate programming code, automate software testing, automate business decision making, deploy chat bots everywhere, programmers need to know the AI landscape so that they can navigate through data analysis and visualization objectives to integrate required AI models. Programmers need to radically change themselves to be data scientist so that they will collect big data, curate raw data and pass clean data to the chosen AI models. Knowledge of big data processing platform of Hadoop, MapReduce, Spark and data analytics languages like Python, PyTorch and R will be essential. AI will be more disruptive in future and programmers have to quickly adopt the changes. Also AI and ML need to be incorporated in software engineering curriculum early on. In future author will explore impact of AI, ML and NLP on embedded system design on FPGA platform.

REFERENCES

1. Turing, A. Computing Machinery and Intelligence. Mind, 59, 433-460, 1950.
2. Reddy, R. Foundations and Grand Challenges of Artificial Intelligence. AI Magazine, 1988.
3. Bruce G Buchanan. A (Very) Brief History of Artificial Intelligence. AI Magazine, Vol. 26, No. 4, 2005.
4. Robert Kowalski. Computational Logic and Human Thinking – How to be Artificially Intelligent. Cambridge University Press, 2011.
5. Wikipedia. Timeline of Artificial Intelligence. Accessed on Aug. 2020.
6. Cambier, Hubert. The Evolutionary Meaning of World 3. Philosophy of the Social Sciences. 46 (3): 242–264. Jun. 2016.
7. Abadi, M. et al. Tensorflow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. arXiv preprint arXiv:1603.04467. 2016.
8. Ovtcharov, K. et al. Toward Accelerating Deep Learning at Scale Using Specialized Hardware in the Datacenter. 2015 IEEE Hot Chips 27 Symposium, 2015.
9. Norman P. Jouppi et al. In-Datacenter Performance Analysis of a Tensor Processing Unit. 44th International Symposium on Computer Architecture (ISCA), Toronto, Canada, Jun. 2017.
10. Antonio M. Ortiz et al. The Cluster Between Internet of Things and Social Networks: Review and Research Challenges. IEEE Internet of Things Journal, IEEE, 1 (3), pp.206-215, 2014.
11. Omar Said et al. Towards Internet of Things: Survey and Future Vision. International Journal of Computer Networks (IJCN), Volume (5) : Issue (1), 2013.
12. S. Deerwester et al. Indexing by Latent Semantic Analysis. In Journal of the American Society for Information Science, 1990.
13. D. Blei et al. Latent Dirichlet Allocation. In Journal of Machine Learning Research, 3, pp 993-1022, 2003.
14. Radha Guha, 2017. Exploring the Field of Text Mining. International Journal of Computer Applications, Vol. 975.

15. Radha Guha, 2020. Exploring Information Retrieval by Latent Semantic and Latent Dirichlet Allocation Techniques. International Research Journal of Computer Science, Vol. 7, Issue 5.
16.T. Mikolov et al., 2013. Distributed Representations of Words and Phrases and Their Compositionality [C]. Advances in Neural Information Processing Systems. 3111-3119, 2013.
17.J. Pennington et al. GloVe: Global Vector for Word Representation. Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pp 1532–1543., 2014.
18. Jianpeng Cheng et al. Long Short-Term Memory-Networks for Machine Reading. arXiv preprint arXiv:1601.06733, 2016.
19. Chris DyeUr et al. Recurrent Neural Network Grammars. In Proc. of NAACL, 2016.
20. Matthew E. Peters et al. Deep Contextualized Word Representations. arXiv:1802.05365. 2018.
21. Ashish Vaswani et al. Attention is All You Need. arXiv:1706.03762. 2017.
22. Jacob Devlin et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805. 2018.
23. Denny Britz et al. Massive Exploration of Neural Machine Translation Architectures. CoRR, abs/1703.03906, 2017.
24. Alec Radford et al. Improving Language Understanding by Generative Pre-Training. 2018.
25. Alec Radford et al. Language Models are Unsupervised Multitask Learners. 2020.
26. Tom B. Brown. Language Models are Few-Shot Learners. arXiv:2005.14165v4 [cs.CL] Jul 2020.
27. Moiz Saifee. GPT-3: The New Mighty Language Model from OpenAI. May 2020.
28.SOAP (2003). Simple Object Access Protocol 1.2, from http://www.w3.org/TR/soap/.
29.WU Nai-zhong. Dynamic Composition of Web Service Based on Cloud Computing. International Journal of Hybrid Information Technology Vol.6, No.6 (2013), pp.389-398, 2013.
30. Radha Guha. Impact of Semantic Web and Cloud Computing Platform on Software Engineering. Book: Software Engineering Frameworks for Cloud Computing Paradigm. Springer, 2013.
31. Radha Guha. Software Engineering on Semantic Web and Cloud Computing Platform. International Journal of Software Engineering, 2011.
32.A. Neumann et al. An Analysis of Public REST Web Service APIs. in IEEE Transactions on Services Computing, 2018.
33.F.M.A. Erich et al. A Qualitative Study of DevOps Usage in Practice. Journal of Software:Evolution and Process, 2017.
34. Diego Lo Guidice. How AI will Change Software Development and Applications. Forrester Research Inc., Nov. 2016.
35. Robert Feldt et al. Ways of Applying Artificial Intelligence in Software Engineering. arXiv:1802.02033v2[cs.SE]. Feb. 2018.
36. Bresnahan, T. et al. Information Technology, Workplace Organization, and the Demand for Skilled Labor: Firm-Level Evidence. The Quarterly Journal of Economics, 117(1),339-376, 2002.
37. Iain M. Cockburn et al. The Impact of Artificial Intelligence on Innovation. NBER Working Paper No. 24449, March 2018.
38. Pornsiri Muenchaisri. Literature Reviews on Applying Artificial Intelligence/Machine Learning to Software Engineering Research Problems: Preliminary.CEUR-WS.org/Vol-2506/Paper5-seed-2019.pdf
39. Mellisa Dsouza. 5 Ways Artificial Intelligence is Upgrading Software Engineering, 2018. Accessed on Aug. 2020.

