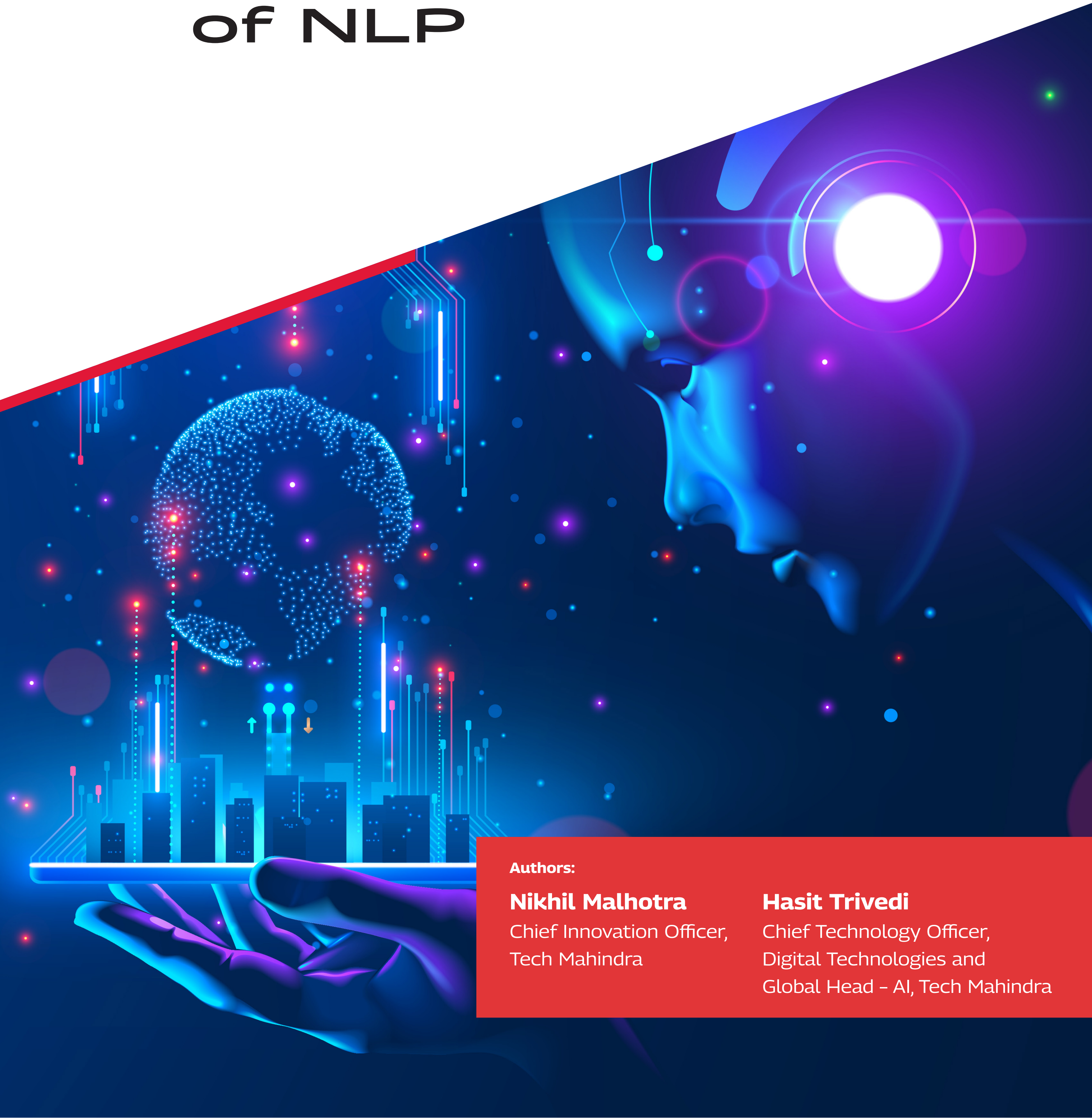
TechM amplifAI 0→∞

TECH mahindra

Whitepaper

Harnessing the Power of NLP



Authors:

Nikhil Malhotra
Chief Innovation Officer,
Tech Mahindra

Hasit Trivedi
Chief Technology Officer,
Digital Technologies and
Global Head – AI, Tech Mahindra

Abstract

An era of rapidly changing technology, virtualization, software-defined networking (SDN)/network function virtualization (NFV), the advent of 5G technology is redefining businesses across the globe. The digitalization wave has brought artificial intelligence (AI), natural language processing (NLP), analytics, and big data into the foray making it more possible for machines to emulate humans.

This is making most of the ways of doing things exceptionally well in the past, invalid in this new environment. At its peak, we expect the traditional operations replaced with operations driven by speech recognition, natural language processing, artificial intelligence, analytics, and machine learning. There will be new key roles emerging in facilitating machine learning (helping machines become intelligent) and analytics.

In this digital age, we see a marked shift in the consumer purchase cycle. From managing media and publishing – the key aspect of promoting mass consumption – marketers are shifting their attention to on-going customer experience. This is because of the critical shift in consumer trust from published media to peer opinion. That is what is driving service uptake and loyalty.

In fact, this has a major impact on how we view digital – the new way of creating services and the new way of consuming and servicing them which is dramatically changing the traditional customer-supplier equation and is driven by the evolving technologies that offer a new lifestyle and insights.

The era of the customer reaching out to the product is gone. Now, it is the service that must seek out the customer and that too through peer influence and not the conventional media.

The word ‘convenience’ has assumed an all-important role in not just discovering a service but also in delivering and managing it. Product lifecycles have become short placing a very high demand on the traditional ways of creating and delivering products. Often, the very nature of the time required to release a product makes it obsolete or is unable to deliver the desired ROI.

Key Takeaways

1

Our Inspiration

'Convenience' is associated with the word 'conversation' which we human beings are quite good at. Conversation and the ability to have a diverse language set to communicate is one of those traits that distinguish humans from the rest of the animal world. This was the inspiration that also drove Makers lab (the innovation lab at Tech Mahindra) towards building our own framework and developing our suite of AI solutions and offerings.

Our suite of AI offerings is branded as TechM amplifAI $^{0\to\infty}$. The term "amplifAI $^{0\to\infty}$" has a clear aspirational value where 'amplifAI' conveys our belief that AI is pervasive and has incredible power to amplify human capabilities, and thus leads to game-changing outcomes at speed and scale by transforming every aspect of any business – be it the business model, the processes that define it, the applications that deliver it or the infrastructure that supports it.

'0' (Zero) signifies our goal of helping businesses to achieve Zero Ops – i.e achieve extreme efficiency and effectiveness in business processes that humans do today. This could manifest as zero disruption, zero errors, zero touch, and zero latency. Further, '∞' (Infinity) implies that we help businesses to aspire for infinite possibilities to solve the problems of the future, which are hitherto considered 'unsolvable' due to some human dependency or technology limitations.

Our award-winning suite of offerings is boosted by our home grown and acquired product and solution intellectual properties (IPs) and covers the entire spectrum of AI applications. Now, we have close to 60 solution accelerators/assets and partner IPs under the umbrella of TechM amplifAI $^{0\to\infty}$, encompassing a robust set of AI offerings and solutions including conversational AI and chatbot analytics.

The idea of chatbots has been around for long time. Taking a leap from the first dial tone landlines which allowed us to ring or dial a number to get our account information was the earliest most primitive format of a chatbot. Being primitive, it was also not cognitive and worked on simple command sets (or number dialling sets) to provide information. As software took shape, IVR and other techniques allowed enterprises to serve customers across the world. Slightly better than a ring dial (some might say complicated as well) were IVR decision trees which were the first stage of intelligent chatbots with an ability to define a decision tree based on what the user dialled at the other end of the line.

As mobility progressed and mobile phones became omni-present, the IVRs were replaced by visual IVRs, and myriads of applications on phones took away the need for humans to call up call centers and engage with primitive chatbots. We are at the dusk of this age where enterprises reached a stage where apps galore and an enterprise finds it difficult to manage the myriads of them for customers and employees alike. This brings us to the current state. In the current state, we are seeing conversational AI in a new avatar, an avatar previously unknown and un-fathomed. When ChatGPT launched, people realized the potential power of generative artificial intelligence (AI) and modelling for the first time.

This paper is all about conversational AI and multitudes of use cases that enterprises can utilize, and most of them do not need to reach the stage of a ChatGPT. Our attempt in this paper is to give an idea about what conversational AI is, and its history of progress reaching the stage of current transformers. While we explain the technology, we would piggy-back on some of the use cases that have become a norm amongst enterprises and how they can utilize the vast resource of Tech Mahindra to tap into for the same.

2

What is Natural Language?

Before we go in and explain the various mechanisms used in NLP, we wanted the readers to understand what natural language is. In simple terms, natural language is a verbose unformed sometimes, non-grammatical many times, a language that is used for general communications amongst human beings. We use this format to communicate with others but also use the same to formulate thoughts in our heads. Formal Languages in contrast are strict unchanging languages, the likes we have all read in schools viz. chemistry, and mathematics. A simple comparison between the two is drawn here:

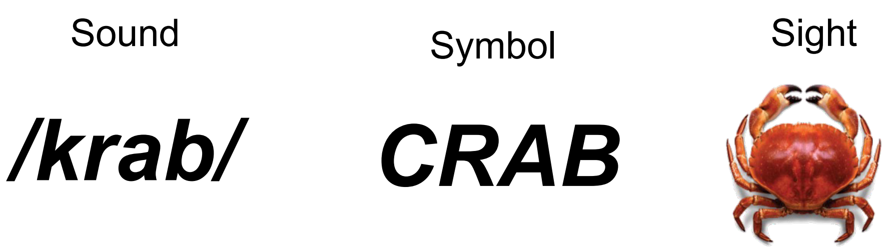
Formal Languages

- Strict, Unchanging rules
- Application specific like Mathematics and Chemistry
- No ambiguity
- Can be parsed by regular expressions
- Inflexible: no new terms

Natural Languages

- Flexible, evolving language
- Unspecific and used in many domains and applications
- Redundant and verbose
- Expressive and ambiguous
- Difficult to parse
- Very flexible

With that as a basic premise, the development of natural language processing started in history. One of the first real attempts was made to consider a language as a formulation of symbols, and these symbols convey certain things which enable thoughts. An example is when we use the term **'CRAB'**. Almost instantaneously, we know what this means. In certain cases, you would be transported to the place where you first saw one. The symbol crab however is a combination of a sound and the visual imagery that you get when you hear this word.



Now, if we say the term "kavouras", most non-Greek speakers would have no idea what we are referring to here. It would appear as the neuron firing has suddenly stopped as the language is hitherto unknown to the speaker. Kavouras just means a crab in Greek, and since the human brain has not seen or heard that symbol, this appears to be a word out of context. This was the primary reason why early practitioners considered language as a means of recognizing symbols, a thought process that had to be discarded for natural language processing to evolve.

3

Start of NLP: Word Vectors

Machines do not understand the text as humans do. Thus, the start of the NLP was really encoding text in a format that a machine can understand. Machines however do understand numbers, so the start of NLP was to convert the text in some format of numbers.

One-hot encoding: One-hot encoding was the first principle that it started with. This consisted of keeping a large corpus of vocabulary list that you may use and have words from a , an to Zulu, etc. This vocabulary is the vocabulary of words that you would use to understand sentences as the machine reads them as follows:

[a , aardvark...zulu]

The next step was to assign an index to this list and formulate a One-hot encoding vector. We can understand this better using a sentence. Let us say the sentence that the machine is trying to understand is:

"The animal crossed the street."

The sentence above is broken down using words as tokens. Let's take the first word which is "The". The process is to find "The" in our list/database of vocabulary and formulate a vector for the same. Let us assume the word "The" comes at the position 4137 and the total number of unique words in our vocabulary are 30,000 so the One-hot encoding vector would look like:

[Position:4137]
[0 , 0 , ...1 , 0,...................................]

By now, most of you would have realized that position 4137 has 1 and the rest of the 30,000 elements in the vector are 0, which gives this the name as One-hot encoding vector. This technique allows us to provide significant meaning to the word with 1 being in the position based on vocabulary.

It was soon found out that although One-hot encoding vector works, there are challenges when someone wishes to find differences or similarities between similar kinds of texts. We would use an example of a pair of sentences here:

"I wish to go to the hotel" and *"I wish to go to the motel"*

They are very similar in their semantic meaning as we see. A hotel or a motel is a place to stay, and the intent delivered by this sentence is a person trying to go to a location to stay. The challenge is that with One-hot encoding vector, all the other words have similar encoding, but the motel and hotel would have a difference primarily because of the position of these words in your corpus and vocabulary list.

When these two sentences would thus be compared, they would appear markedly different. Though for a human, they appear quite similar in their meaning. This gave rise to a mechanism to encode words as vectors which provide a meaning and a context to the word and sentence which led to the creation of a Word2Vec or word to vector technique.

The Word2Vec is generally used for indicating a group of related models that are used to produce **word 'embeddings'**. These models are shallow, two-layered neural networks that are trained to reconstruct the linguistic contexts of words. Word2Vec takes as its input a large corpus of text and produces a vector space, typically of several hundred dimensions, with each unique word in the corpus being assigned a corresponding vector in the space. Word vectors are positioned in the vector space such that words that share common contexts in the corpus are in close proximity to one another in the space.

The conceptualization that happens within the code is as follows. From a given sentence, we apply simple text processing like tokenizing sentences to words, removing common stop words from the language, and lemmatizing it.

```
tokenized_words = nltk.word_tokenize(sentence.lower())
if(not no_stop_word_check):
    #Remove special characters
    tokens = remove_SpecialCharacters(tokenized_words)

    #From the tokenized list, take the stopwords out
    tokens = remove_stopwords(tokens,language)

    #Lemmatize tokens
    tokens = lemmatize_words(tokens)
else:
```

We then create a word vector model out of these sentences to find 64-bit vectors of each word as found in the corpus. The vectors also help us play with language words because only a numerical value can help us perform mathematical functions on it.

A typical word vector of a corpus looks like the following:

```
[
    [0.01, -.02, 0.4, 0l67..........................................0.45],
    [0.11, -.03, 0.4, 0l67..........................................0.45],
    [0.01, -.02, 0.4, 0l67..........................................0.45],
    .
    .
    .
    [0.01, -.02, 0.4, 0l67..........................................0.45],
]
```

The above table shows a matrix (or an array of arrays), which gets formatted for all the questions in the system. Now, what this vector space indicates is how each word in a sentence given in the FAQ is positioned w.r.t the entire corpus. A 64-bit vector is made for each sentence. It looks like an array of points distributed in a space of dimensions, which are measured as the number of unique tokens of words that we have.

The technique for Word2Vec that we use, is the **skip-gram modeling** technique which states that given a window size "m" and a word in a sentence, we would try and predict words in the window on the left and right sides.

The model used in noise contrastive scaling using the maximum likelihood model is shown in Figure 1.

Figure 1

$$\begin{aligned} P(w_t|h) &= \text{softmax}(\text{score}(w_t, h)) \\ &= \frac{\exp\{\text{score}(w_t, h)\}}{\sum_{\text{Word w' in Vocab}} \exp\{\text{score}(w', h)\}} \end{aligned}$$

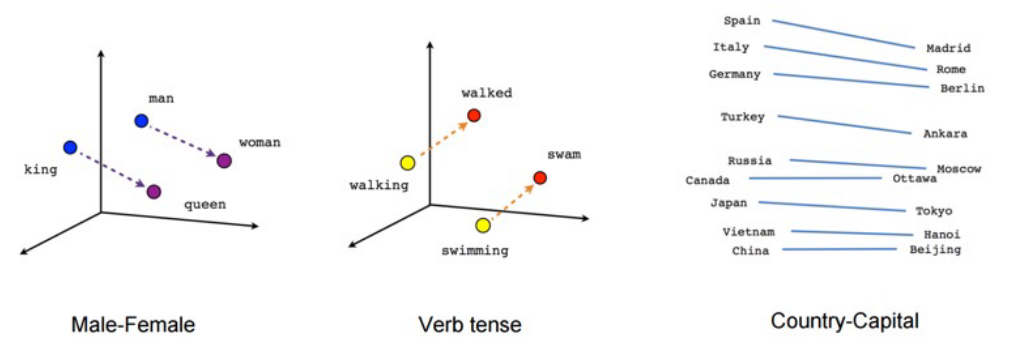
Where score (wt,h) computes the compatibility of word wt with the context h (a dot product is commonly used). We train this model by maximizing its log-likelihood on the training set, i.e. by maximizing (see Figure 2):

Figure 2

$$\begin{aligned} J_{\text{ML}} &= \log P(w_t|h) \\ &= \text{score}(w_t, h) - \log\left(\sum_{\text{Word w' in Vocab}} \exp\{\text{score}(w', h)\}\right). \end{aligned}$$

Once these word vectors and sentence vectors are made, it is very easy for the system to do an analogical reasoning of a question like "King is to queen as father is to?" and the answer is a "mother" because the vectors closely match the space of mother.

Figure 3



An example picked up to demonstrate this from tensor flow tutorials is shown in Figure 3. The vectors once formatted enable us to do a lot of mathematical analysis on them. For our scenario, we use these vectors to determine the semantic closeness of a question asked by a user and provide answers to the user.

4

Examples of WORD2VEC usage

Word2Vec was preceded by a long list of some of the other techniques as well which included Jacobian distance measurement and techniques, but it turned out Word2Vec and Glove vectors work the best for semantic meaning. Word2Vec is now the initial stage for any natural language processing technique.

One of the biggest examples that Word2Vec usage even today helps us to do is formulate a lot of working of simple FAQ based chatbots. Some examples that can be used with Word2Vec itself include the following:

1. **FAQ based chatbots:** Easy code which can be done by feeding in the questions and answers and when asked the machine finds the right answer amongst a given list
2. **Document parsing and Search:** Search through documents to find the paragraph or page where the answers lie
3. **IVR based automated response:** Most IVR based systems work on a call tree mechanism, but Word2Vec allows us a good way to even look at an NLP response and get the closest agent call code for resolution
4. **Music2Vec:** A new concept to convert music- based tones into a vectorized format akin to Word2Vec. This enables usage within applications like Spotify to find the nearest based song based on a note of music

Chabot scenario example: It is important to understand what the framework enables. As a natural UI, this is how the framework can be utilized:

1. **Natural UI with mobile phones**

Figure 4

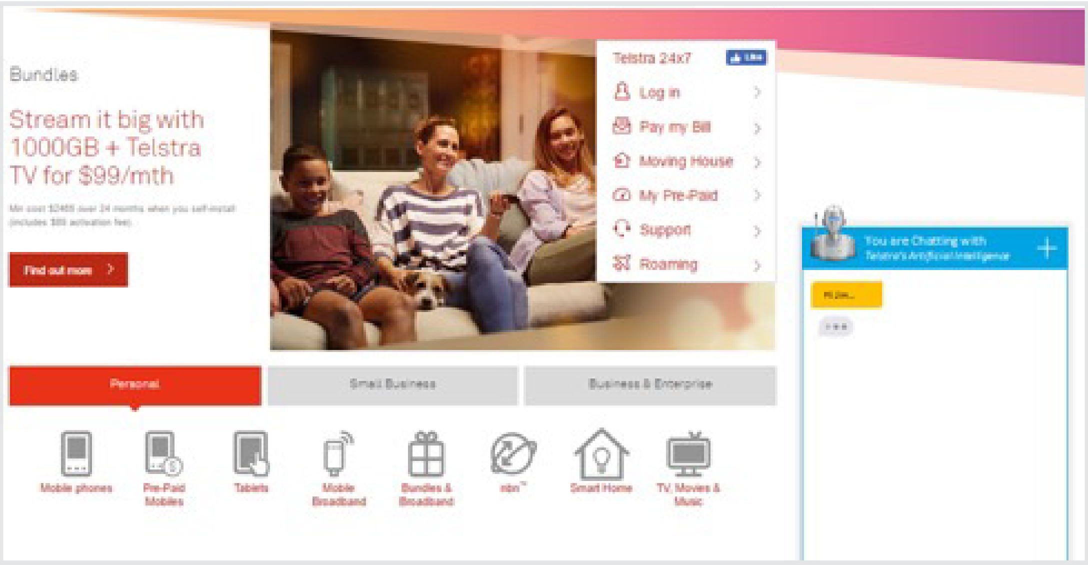
Figure 4

2. Conversational Browsing Agent

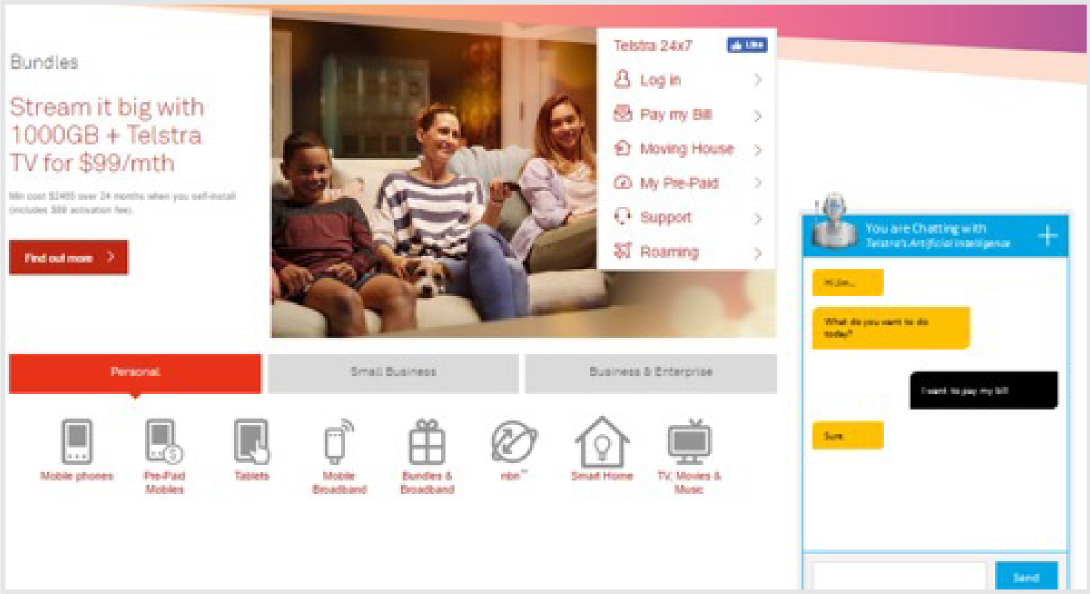
The idea of a conversational browsing agent is to enable the same Entellio natural language UI to be a conversational browsing agent on a website. We believe that a lot of people still use corporate websites.

These websites have entropy associated with links in them, not allowing new or old customers to reach specific information seamlessly. Entellio can become a natural language agent on the website where it can understand queries, but the web redirect mechanism described earlier can enable the movement of pages on the website. It is a technique called conversational browsing, and it is unique to Entellio as it gets packaged as a part of the entire suite.

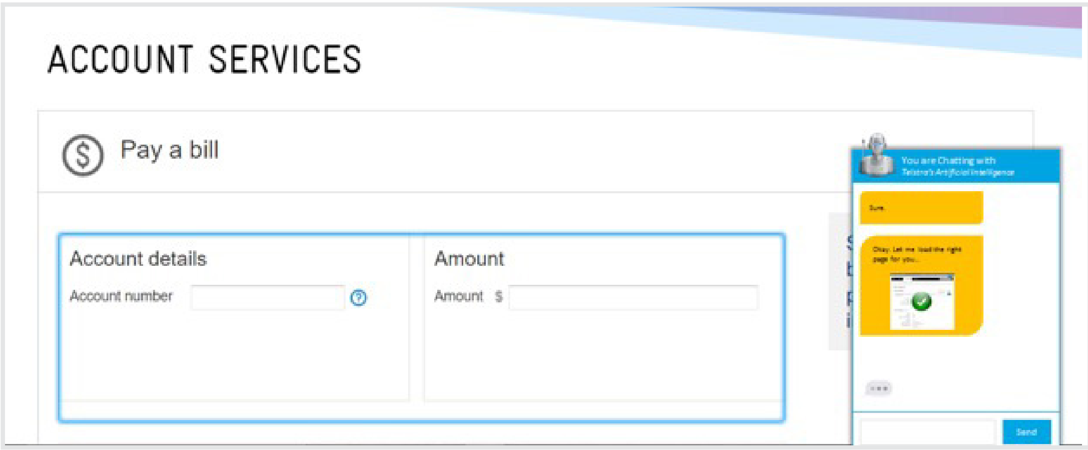
An example is shown in Figure 5 illustrating a use case for a telecom company and a customer requirement of paying a bill:



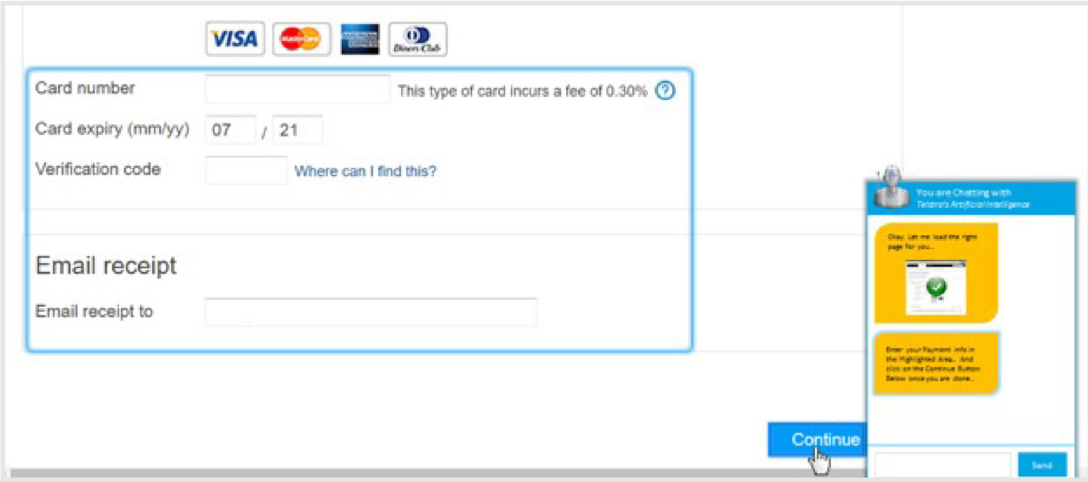
A simple example showing Entellio placed as javascript on a Telco website where a user can start conversation with it.



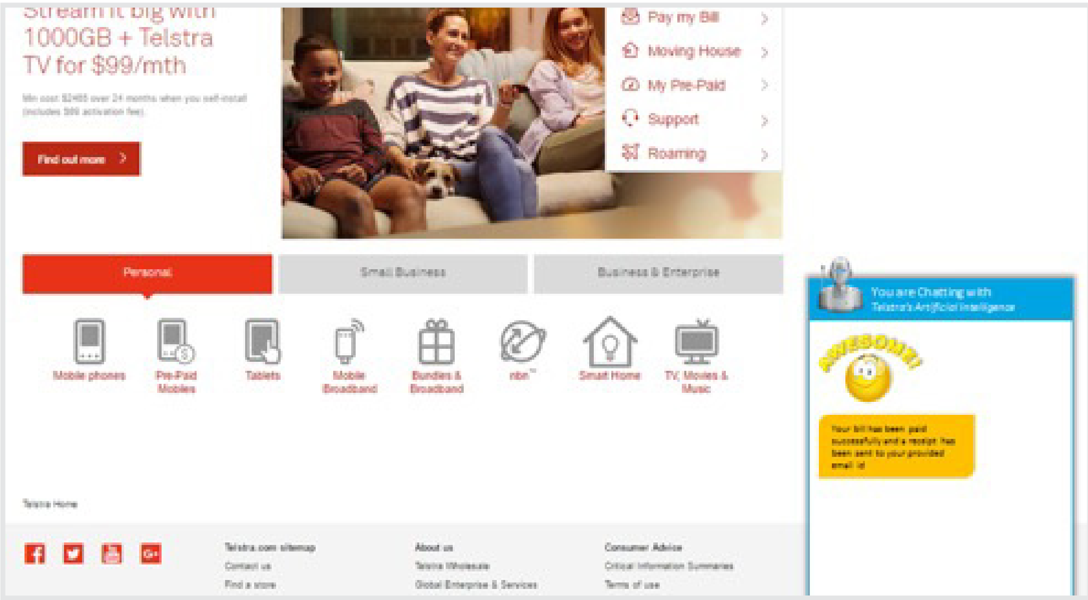
A conversation initiated with the user by the framework about his intention which happens to be paying his bill.



Power of web redirect or conversational browsing is shown here where the user is automatically routed to the deep link payment screen once the intention is sanctified.



Highlighting the area of interest for the user to fill his/her payment details here.



A happy customer with his journey sorted out in a natural language mode.

5

Moving on to Sequences

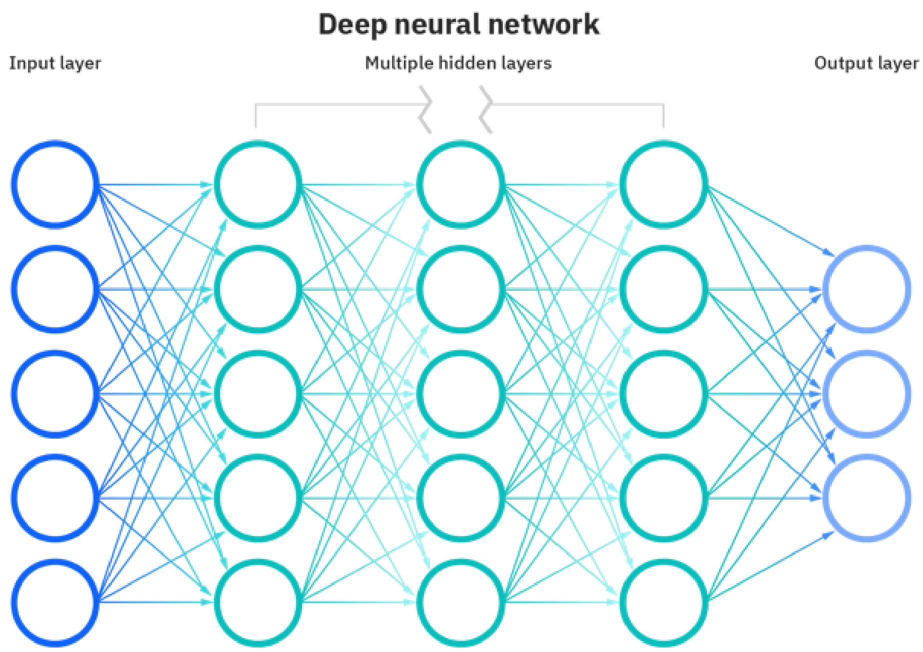
Let's continue to build from the top. The Word2Vec approach told us that words are no longer an island, and they carry context where they occur. In fact, the basic principle of Word2Vec taught us those words of a feather flock together. For e.g., medical journals would have words which are quite related to each other and quite unrelated to other topics.

The next process within natural language understanding was understanding the word sequences. Sequences are the core of sentence formation, and hence the core of communication. Sequences, however, can translate into a large number of use cases; a few are as follows:

1. **Machine translation:** Je suis content -> I am Happy
2. **Finding named entity recognition from a system : Harry Potter** and **Hermione Granger** are good friends.
3. **Generating music from a single note**
4. **Generating words: Something like ChatGPT does**

When sequence modelling was considered, it was found that the usual neural networks would not work well. This is for a simple reason that neural networks have fixed input and fixed output. A plain vanilla network would not be able to model the intricacies and would not be resilient to change when inputs and outputs change.

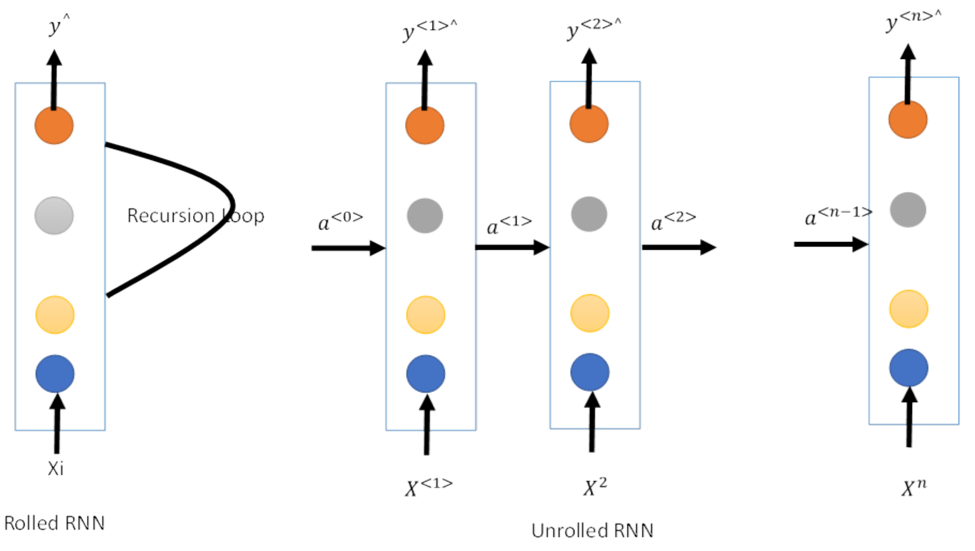
Figure 6



We have seen in some of the examples, particularly when we do machine translation, that the number of words in the input is not the same as the number of words in the output. Similarly, generating music notes out of noise has no input or very little input, and the output could be a complete note of music melody.

A special class of algorithm or tweaking of neural networks was needed. This was done using the recurrent neural network as shown below:

Figure 7



See Figure 7. The network on the left is a rolled network. The box can be considered as the network. The circles are nodes that do some computation like the neural network in the above figure. This recurrent neural network works overtime. However, most books show this network, and we would love to explain from the unrolled network itself. Consider a neural network that gets unrolled in time "t" every time it takes a word in the sequence of sentences. This is shown as an unrolled network.

The advantages of RNN over vanilla neural networks was large because they could model time in the sequence of words and could understand the sequence of words better. Another advantage was that the weights and other parameters were shared across the network. This is of great importance for an AI modeler. Consider a network that has learned that "Harry" is a proper noun and a name. We would like this network now to remember this whenever it finds the name "Harry" anywhere else in the document or text of importance. This is the reason RNN survived and did very well till they succumbed to a general issue.

The issue that RNN faced was the context when the sentences became longer. Consider the following paragraph.

An ***animal*** *came to a city. The city was large and cacophonic. The animal was very scared as it saw a large number of cars, buildings and other features in the city. The city was so large that* ***it*** *found that the forest appeared smaller*

We have just made a random text to explain the issue with RNN. Please don't judge this for its grammar or context. The challenge that we wished to put forth is that "it" which is referred to in the last line represents "what". Does it represent the city or the animal? For us as human beings, this comes intuitively that the "it" refers to the animal here but for machines, it is not. Applying RNN over these large sentences posed a challenge as the RNN could not understand it. The major challenge here that RNN faced was the "vanishing gradient" problem.

To solve RNN's issue, a new class of algorithms was built in. This was called the LSTM or Long short-term memory. This model had the ability to refer to such large references and was able to understand the context. However, LSTM also suffered from a few challenges. LSTMs were cumbersome to train and were found not producing results as needed.

6

The age of Transformers is here

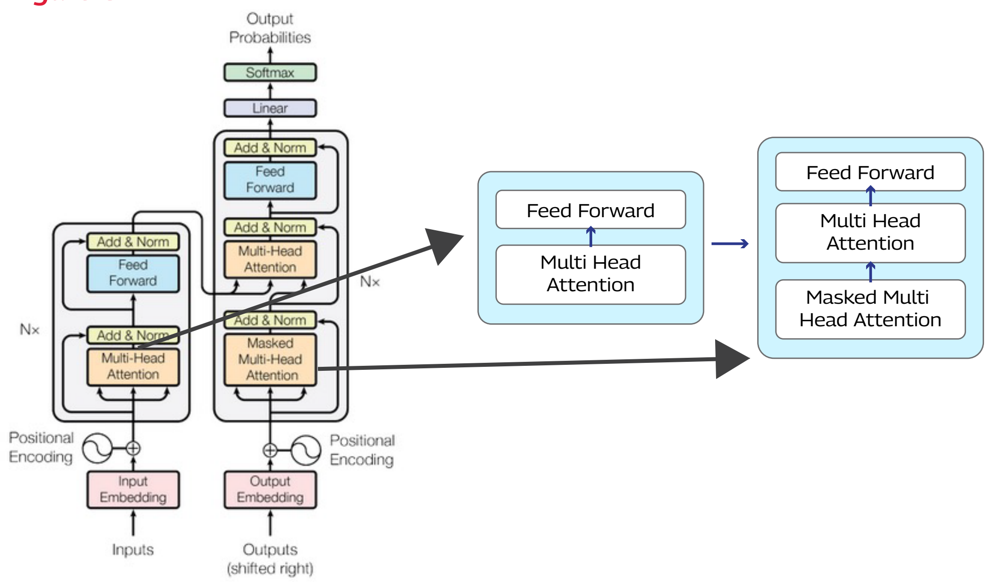
The age of GPT evolved from a transformer and this is the age that we live in. In all the above models, the biggest challenge was "attention", attention to what the sequence of words meant and what really was being conveyed as a context. Some models did do attention but then fail to recognize long-standing sequences.

Sequence-to-sequence models are deep learning models that have achieved a lot of success in tasks like machine translation, text summarization, and image captioning. Google Translate started using such a model in production in late 2016. However, sequence models suffered from the attention which became via a paper in 2017.

Here are the steps of transformations (Transformers sit in the heart of ChatGPT) and how they evolve with attention. It is, for this reason, ChatGPT can generate tokens (words) so well.

The transformers are made of the same encoder and decoder layers as a seq-2-seq model shown by RNN (see Figure 8).

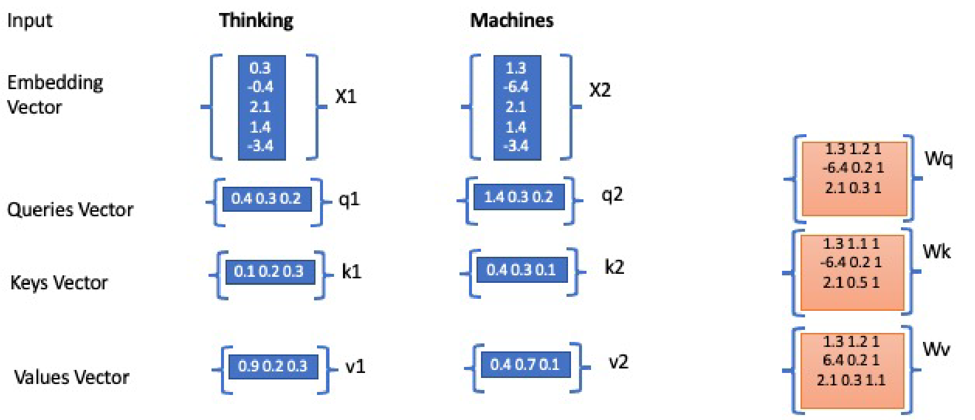
Figure 8



For some reasons, there are 6 encoders and 6 decoders that the creators found giving them better results.

The attention of a word is shown by Figure 9A:

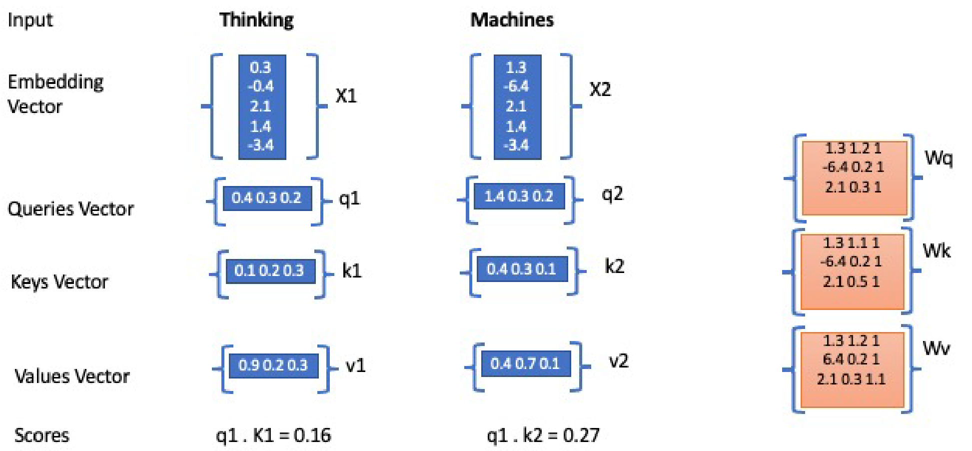
Figure 9A



The first step is the step of embedding which is nothing but creating words into vectors as we have seen in the Word2Vector part. The second step is attention by creating queries, keys, and values vectors. These are random vectors, and these are the vectors that are trained. These vectors are created by multiplying the embedding vectors with the three matrices shown Wq, Wk and Wv. These vectors are nothing but abstractions to think about attention.

The next step is calculating a score:

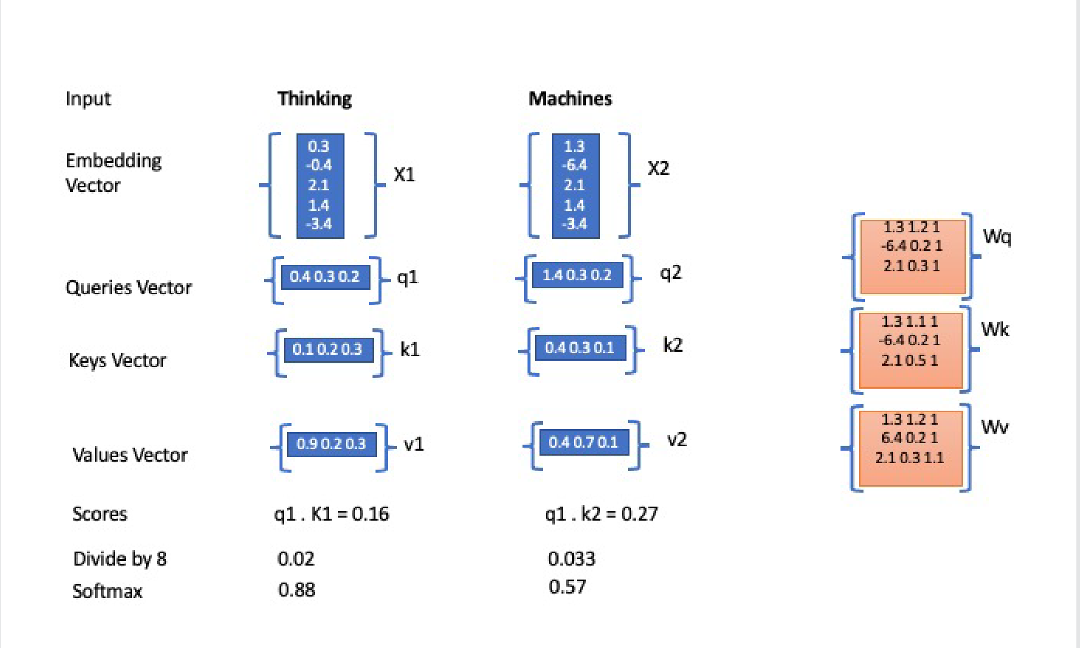
Figure 9B



Let us imagine (taking the same example as the blog) we have two words namely “Thinking Machines”. q1 , q2, k1 , k2 and v1, v2 are created by multiplying the vectors x1 and x2 with the matrices respectively.

The score is then calculated by multiplying the query vector with the key vector. The score is then divided by the square of the vector size (64 here) and then, a softmax is applied (see Figure 9C).

Figure 9C



The softmax in a simple way defines that when we look at a word called “Thinking”, what is the attention given to itself and how much to a word like machine.

This is the current state of transformers. What’s next?

7

ChatGPT and Transformers

ChatGPT as of today has become the latest surge and the poster child of what natural language generation can do. ChatGPT is just the first realization of the power of transformers brings to this world. It is certainly not the last and not the first either. Models that came before ChatGPT have shown the power of what transformers (explained above) bring to the table. Some of the earliest ones include BERT.

In this section, we will discuss what ChatGPT can do and what are some of its limitations. And explain what and how transformers can be utilized within an enterprise world. While ChatGPT may look like a cool kid on the block answering questions, creating prose and poems or even PPTs will come with limitations.

Positives of ChatGPT: Lets start with positives of ChatGPT

- **NLG:** For the first time, natural language generation, which means the generation of sequences of words which are grammatically correct have been possible using some model of AI. This is a great achievement. Until now sequence 2 sequence models like RNN, LSTM could do this, but the words and generation were not possible to the level ChatGPT could do.
- **Foundational AI:** It's a foundational model, which not just does conversational AI but also generates code pieces. Foundational AI models have the capacity to do more than a specific task. This is because ChatGPT is trained on a vast corpus of Wikipedia, Quora and some other forum-based sites, where answers are provided to questions posed on code pieces.
- **Limited Transparency :** We would not go out on a limb that it is truly ethical AI, but it tries to do its part as the creators seem to have learned from mistakes done by other models in the past. These include challenges produced by Tay bot of Microsoft and some other models.

Limitations of ChatGPT : Lets start with some limitations now

- It doesn't have the context of an enterprise-based system. For it to attain so, data for the enterprise needs to be inputted. This itself brings two issues which are whether enterprises would be willing to send their conversations and contexts on a cloud-based system.

- ChatGPT API provided to developers have a limitation on the number of tokens(words) that you can send. It seems to be 150 tokens. As a result, if you wish ChatGPT to take a complete text to understand and generate answers from it, it doesn't work. You must break open paragraphs in a format which is digestible. The question now is what paragraphs. Here a pipeline of BERT – ChatGPT works well. BERT enables us to outline paragraphs which can be sent to ChatGPT and ChatGPT can use it to generate free language

- ChatGPT is also confident when it is wrong. This can be a major barrier for enterprise-based applications.
- You can profess a thought that ChatGPT would eventually agree. As an example, let's try writing a statement of ChatGPT

  - **ME:** "I saw a pink elephant flying".

ChatGPT would retort back by saying the following

  - **CHATGPT:** *"It's possible that you experienced a hallucination. Seeing a pink elephant is often used as a symbol for being drunk or having consumed other substances that can cause hallucinations. However, if you are concerned about your health, it's always a good idea to consult with a doctor."*
  - **ME:** "What are you saying .I have seen pink elephants flying I am sure"
  - **ChatGPT:** *"If you're certain that you saw a pink elephant flying, it's possible that you experienced a vivid imagination or a dream. Seeing things that are not there can also be a symptom of certain mental or neurological conditions, but without further information or context, it's difficult to determine the exact cause. If you are feeling concerned or experience any other unusual symptoms, it's recommended that you speak with a medical professional for an evaluation"*

As you notice, if we keep emphasizing on incorrect ideas or concepts, ChatGPT would revert more. Now we tried this again:

  - **ME:** "I saw a pink elephant flying"
  - **ChatGPT:** *"It's not uncommon for people to have unusual or imaginative experiences, and seeing a pink elephant flying could be an example of this. However, if you're experiencing this vision repeatedly or it's affecting your daily life, it might be a good idea to consult a doctor or mental health professional for an evaluation and to rule out any underlying health conditions."*

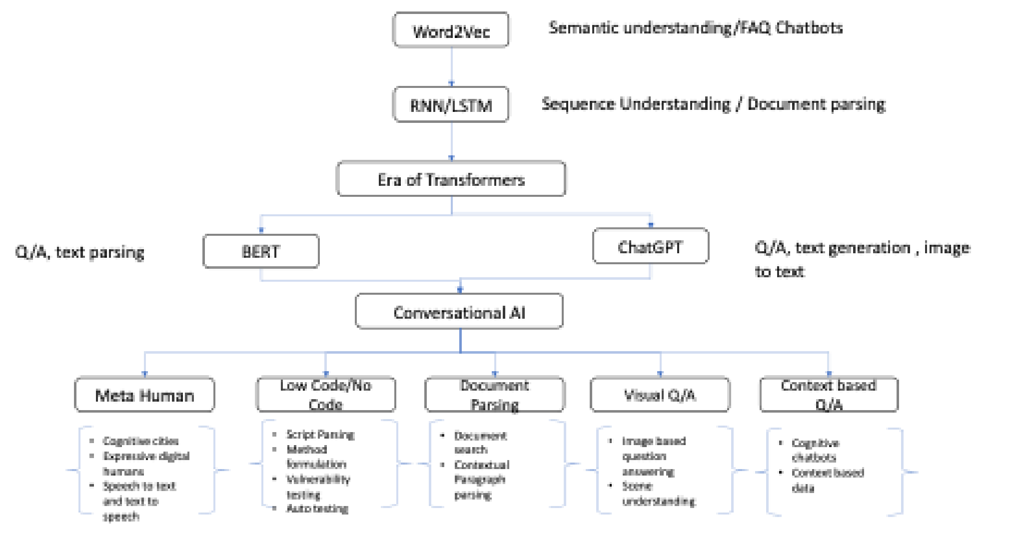
The response is different which gives the illusion you are talking to a real person, which is not. If all of you try the same sentence as we have, you would see CHATGPT alternate between the two answers which indicates that the model, although very good, has some defined datasets from which it generates its values.

8

Tech Mahindra's point of view on NLU/NLG

The TechM POV on conversational AI is represented in Figure 11.

Figure 11



If you notice, our POV starts from the very beginning and looks at a value chain in the life of the enterprise. We continue to use and simplify models using Word2Vec to solve some of the latent not-so-big challenges. Since Word2Vec give a semantic understanding of text, they seem to perform well for FAQ based chatbots

RNN/LSTM are used to improve the power of this semantic understanding by providing sequences to understand documents and also sequence disambiguation.

Our world has been transformed by transformers. These transformers are shifting the purview of conversational AI. We have divided conversational AI in the formats as depicted above. This also references work that we are doing with given SDKs and systems and our research regarding the next step beyond the same.

- **Metahuman:** Metahumans have become a mainstay now especially since smart cities have evolved into cognitive cities. These cognitive cities are now looking towards never-seen-before digital humans who can express feelings and can converse with netizens. They could tell you the weather, your electricity bill or even order food from a restaurant.

- **Our Research on Metahuman:** While we work with given systems available in the world, like Convo.AI, Unreal-based metahumans and soul machines one of the works that the Maker's Lab is doing is trying to evolve metahumans on their own. We have followed the principle of graphics pipeline by creating our own metahumans which include creating face meshes, understanding mesh points/polygons, and then training them with speech. This speech may not be just English, but we have decided to train these meshes with Indian and Arabic languages as well. Once trained these meshes would then be linked with any face structure created to talk. One of the challenges of training today's metahumans is dependence on the third party and creating your own wave file format for speech to connect. We wanted to make it simpler as a service

- **Low Code/No Code:** One of the premier use cases for conversational AI is low code and no code. Github co-pilot which also uses transformers at the backend is helping developers do the same

- **Our research on LC/NC:** Makers' Lab is involved in training some part of github co-pilot in testing these LC/NC based systems and their efficacy for the system. Our research is primarily on looking at an end-to-end application and then seeing how much of a code can be regenerated using LC/NC

- **Document Parsing:** Document parsing does not include anything new from our end but using BERT we are trying to make document parsing and search better

- **Visual Q/A:** Visual Q/A is a new concept that we have worked upon where images speak in form of text. Once provided an image, a user can ask questions about the same using our system. This Visual Q? A of course uses principles similar to ChatGPT to generate answers around questions asked.

What we see as pragmatic applications of NLU/NLG:

NLU use cases

- Clinical Documentation : *Simple Word2Vec/BERT solves this*
- Speech Recognition
- Computer Assisted Coding :Some extent using Github copilot
- Document parsing and understanding
- Clinical Decision Support
- AI chatbots
- Sentiment Analysis
- Dictation / Grammar Correction
- Root Cause Analysis
- Generating multiple choice questions
- Generating coherent text
- Topic Modelling

ChatGPT based use cases

- Contextual Conversational AI : Independently not possible. Have to use BERT here for context
- Diffusion models for network diagrams in combination
- Website/Webpage construction: Limited
- Video based creation: Limited use case
- Summarization of text
- Creation of power points

9

Beyond Transformers / Neural Networks

While we have seen the number of AI models that have been applied to the problem of language, which has started from Word2Vec doing word embeddings to find similarity, to Recurrent Neural Networks for sequences to the models today that include BERT and GPT-3. All these models have made a significant progress in terms of moving the needle ahead for language modelling but are far off to mimic the human brain. This research is an attempt to try something very different from a human brain perspective. But before we do that, we are laying down foundations by comparing some of the works that have passed before this.

Our methodology of doing this revolves around considering language as language and not vectors as words and looking at solving some challenges in terms of natural language processing, concepts like predicting the next word in a sentence, establishing a language model from speech (example: we bear the bears), Q&A with context, to even understanding multimodal transformations like a machine understanding a video and generating a summary.

Our attempt is not to debunk the previous models, but to build on top of it to generate a better machine to understand language.

Synaptic Graph Networks

- Synaptic graph network builds on the thought that is unique, but one that takes place in human brains continuously. We are putting down a few of those for consideration
- Words for a human brain are not vectors but placement symbols
- A vocabulary for a human brain is the symbolic attachment of an embodied experience and past experiences
- Grammar is important but not essential for communication
- Words immediately transpire a mental image even with a negative connotation. An example to test it is to say the words ("Don't think of an elephant") which immediately conjures up an image of the same
- Memory (long term) in a neo-cortex is in the hippocampus and a large part of the conversation includes that
- The synapses are the learning centers of the brain. The connection between Synapses guides the memory formation
- Words that humans use are relative based on past experiences and their vocabulary. Every task may not require the same vocabulary, the brain spends energy based on the task at hand

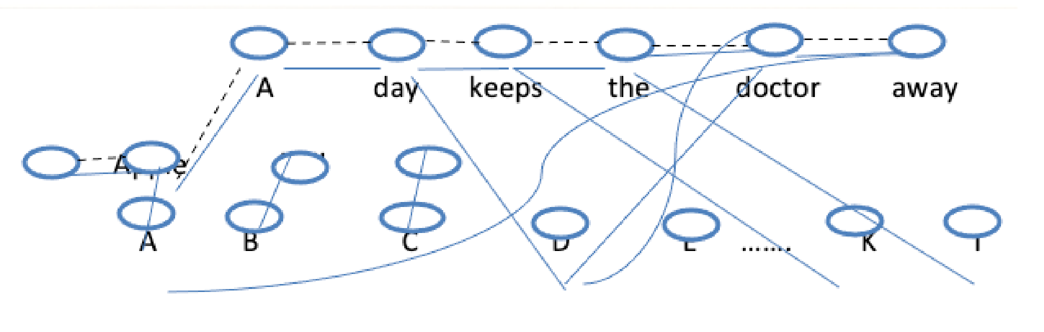
What is Synaptic Graph Network?

It is an attempt by Maker's Lab to build a language model to mimic the brain. The process has just started, and this paper is laying down a foundation (requirements) for the same.
Please Note: Components can change as we proceed.

Step 1: Training Process:

- The first process of an SGN (Synaptic Graph Network) would be to train the network based on corpuses(data) available. The author can use a large Wikipedia vocabulary or a vocabulary that gives the initial model a kickstart.
- A basic graph structure would be made something akin to how a child learns 'A' for Apple and so on.
- The corpus would be tokenized into sentences and words.
- Sentence definition would be made proper by adding <EOS> end-of-sentence symbol at the end of the sentence.
- These words would then be attached to an acyclic graph structure based on how they formulate sentences, something akin to how a human formulates the structure in the brain.

Figure 12



The dashed lines indicate the sentence being formulated (see Figure 12).

The edges represent 2 data points –
- Connections of conjunctions or relations between words like 'A' for apple
- Weight of the edge which represents synaptic weight

A synaptic weight would be calculated based on the number of times a connection is turned on based on vocabulary usage.
- Weight would be incremented as 0.1 every time a connection is utilized.
- For weights that get frequently referred to in terms of statement in the vocabulary and have reached a value of 1, will reflect a permanent connection in the vocabulary.
- Weights of value 1 and the sentences would be placed in a register that would play the part of a hippocampus.

Some things to note:

- The structure represents the spikes and system a human brain uses for vocabulary.
- There are no vectors here or an attempt to convert words to vectors (It must be seen how performant the system is)
- Human neo-cortical design is used for testing.

Test: What do the author(s) hope to achieve:

- How easy or difficult is it to train a corpus in the manner of graphs?
- How fast/energy efficient can the system be? Even though training does not involve gradient descent, how performant would it be?
- Test of language functions

Test of language:

Following tests would be conducted on them SGLN:

- Next word generation -> Application -> Search
- The system checks from the hippocampus register and the network based on the edge value (weight) to get the next word.
- Speech language model -> Speech to Text
- Sentence/ word comparison -> Sentence encoding

About the Authors

"We are reminded of the limitlessness of human curiosity, when we see man and machine create marvels for the future together."
- Nikhil Malhotra

Nikhil Malhotra
Chief Innovation Officer

Nikhil Malhotra is the Chief Innovation officer and creator of Makers Lab, a unique Thin-q-bator space within Tech Mahindra with over 21+ years of experience in a variety of technology domains.

He lives by a dream of creating smart machines that would wed human emotions with artificial intelligence to make lives better.

He is also a leading speaker on the practical use of AI and the future AI, quantum, and innovation.

He holds a master's degree in computing with a specialization in distributed computing from Royal Melbourne Institute of Technology, Melbourne.

Hasit Trivedi
Chief Technology Officer, Digital Technologies & Global Head-AI, Tech Mahindra

Hasit Trivedi is a seasoned IT professional with more than two and half decades of industry experience. A technocrat with a wide variety of experience in running businesses in the niche technology areas like artificial intelligence, intelligent automation, cloud, and IoT. Hasit has played key role in technology innovation, client delivery, product management, product engineering and building units from the scratch.

Hasit is an imminent speaker and member of various industry forums. He has been awarded as the best AI leader, recognized as a leading AI voice and one of the most influencing voices for enterprise brands. Besides, he is also associated with start-up ecosystem as a mentor and advisor in voluntary capacity.

He has been sharing his thought leadership opinions in various reputed publications and forums.
Hasit has been leading the discussion of AI with customers.

TECH mahindra

www.youtube.com/user/techmahindra09
www.facebook.com/techmahindra
www.twitter.com/tech_mahindra
www.linkedin.com/company/tech-mahindra
www.techmahindra.com
top.marketing@techmahindra.com