

Building a Language Model (AI) from Scratch

Natarajan Ramasamy

* *As a software architect working with Cloud AI and local LLMs, I felt a gap between *using* these powerful tools and truly *understanding* them. So, I decided to close that gap by building one myself, from the ground up, using only Python and PyTorch on my personal Linux laptop.*

1. The Goal: From "Black Box" to "Glass Box"

* My objective wasn't to compete with large-scale models. It was to demystify the "magic" of Generative AI.

* I wanted to build a **Small Language Model (SLM)** that was powerful and non-toy, but highly specialized in a single domain.

* The Challenge: Could this be done on a standard CPU, without GPUs, by carefully managing the scope of knowledge, not the quality of the architecture?

2. The Project: A Specialist AI on the "Bhagavad Gita"

* I created a character-level language model trained exclusively on the text of the Bhagavad Gita.

* This means its entire "world" and knowledge base is contained within this single, profound philosophical text.

* The result is not a generalist chatbot, but a deep expert, capable of generating text that is stylistically and thematically perfect for its domain.

3. The Architecture: A Look Under the Hood (in Plain English)

My SLM's "brain" is a Transformer architecture, which works in a few key stages:

- * **Stage 1: The Input - Turning Language into Numbers**

- * **What it does:** A simple tokenizer converts each character of a prompt (e.g., 'A', 'r', 'j', 'u', 'n', 'a') into a unique number. The model only ever sees these numbers.

- * **Stage 2: The Embedding - Giving Numbers Meaning**

- * **What it does:** This layer acts like a rich dictionary, converting each number into a dense vector (a list of 128 numbers). This vector represents the character's "personality" and, crucially, its *position* in the sentence.

- * **Stage 3: The "Thinking" Core - The Transformer Blocks**

- * **The Collaboration Phase (Self-Attention):** This is the magic. Each word looks at all the previous words in the sentence to understand its context. It's like people in a meeting listening to what's been said before they speak.

- * **The Introspection Phase (Feed-Forward Network):** After gathering context, each word "thinks" individually to process what it has learned.

- * My model stacks 6 of these "thinking" blocks, allowing it to understand deeper and more complex patterns with each layer.

- * **Stage 4: The Output - Turning Thought back into Language**

- * **What it does:** After passing through all the layers, the model generates a list of scores for every possible next character. It then uses these scores to predict the most likely next character, appends it to the sequence, and repeats the entire process, generating the text one character at a time.

4. The Demonstration: Putting My SLM to the Test

I tested the model with different kinds of prompts to see how it "thinks":

* **The Scholar:** When given the first half of a real verse, it completed it with a thematically perfect (though not always identical) conclusion, proving it's not just a lookup table.

* **Prompt:** `You have a right to perform your prescribed duties, but`

* **Result:** `...but those soul Who tread the path celestial, worship Me With hearts unwandering...`

* **The Creative Poet:** When given a brand new sentence in the right style, it generated a completely original and authentic-sounding passage.

* **Prompt:** `He who is free from the grip of desire`

* **Result:** `...Burned clean in act by the white fire of truth, The wise call that man wise; and such an one...`

* **The Oracle:** When given a single core concept, it expounded on it in a way that demonstrated deep contextual understanding.

* **Prompt:** `karma`

* **Result:** `...He that, being self-contained, hath vanquished doubt, Disparting self from service, soul from works, Enlightened and emancipate, my Prince! Works fetter him no more!`

5. Key Learnings & Takeaways

- * **Feasibility:** Building a powerful, custom LLM from scratch on a CPU is absolutely possible if you scope the domain correctly.
- * **The Power of Specialization:** A smaller, specialized model can outperform a giant, generalist model within its narrow field of expertise.
- * **Prompting is Pattern-Matching:** I learned firsthand that prompting isn't about giving "commands." It's about providing the beginning of a statistical pattern that the model knows how to complete based on its training.
- * **The "Why" Matters:** This journey from using to understanding has been invaluable for me as an architect and has fundamentally deepened my grasp of AI.

What specialized domain do you think would be most interesting for a custom SLM? Let me know in the comments!

CONCEPTS

Diagram 1: The Input Layer

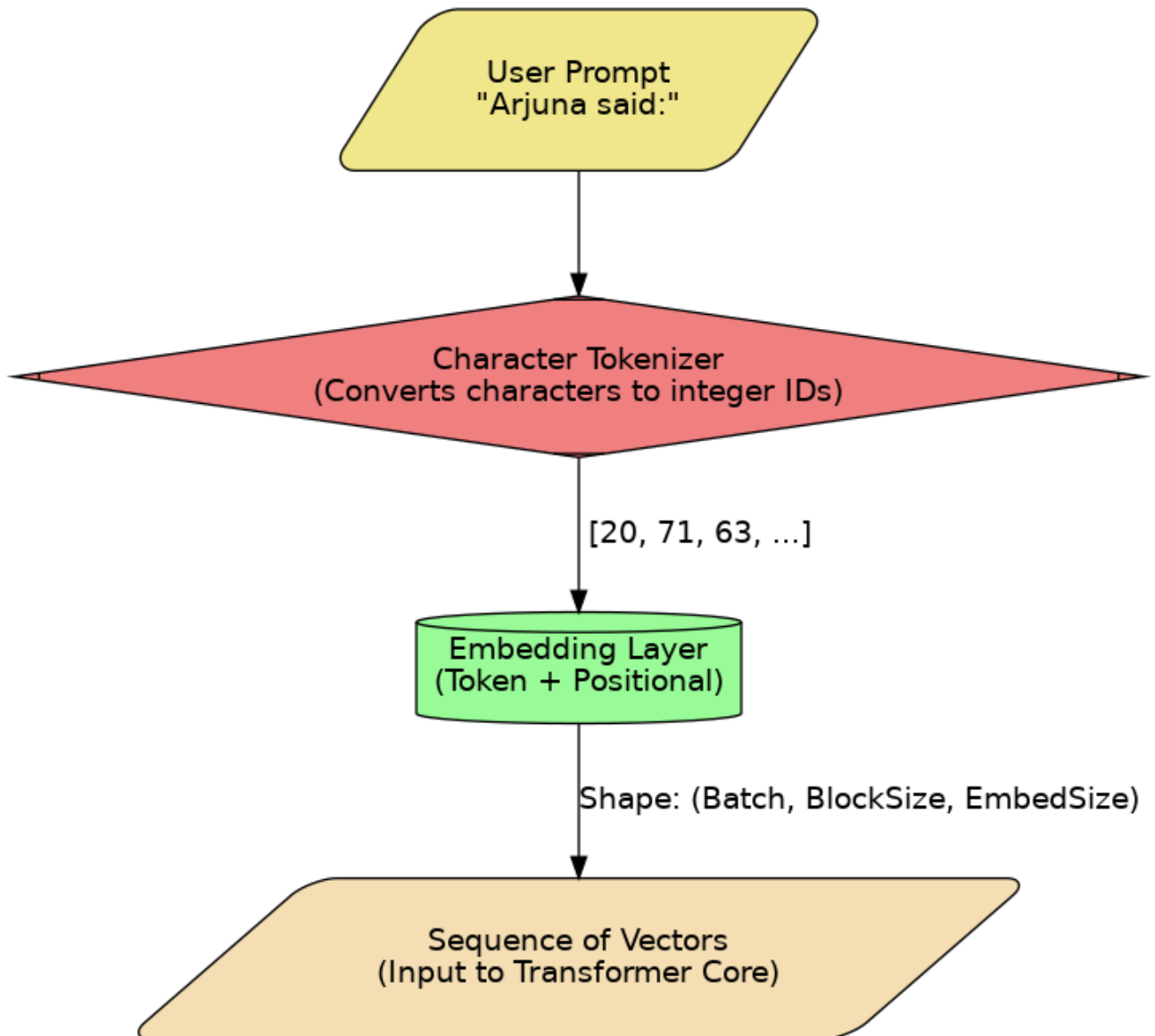


Diagram 2: The Transformer Core (One Block Shown, Repeated 6x)

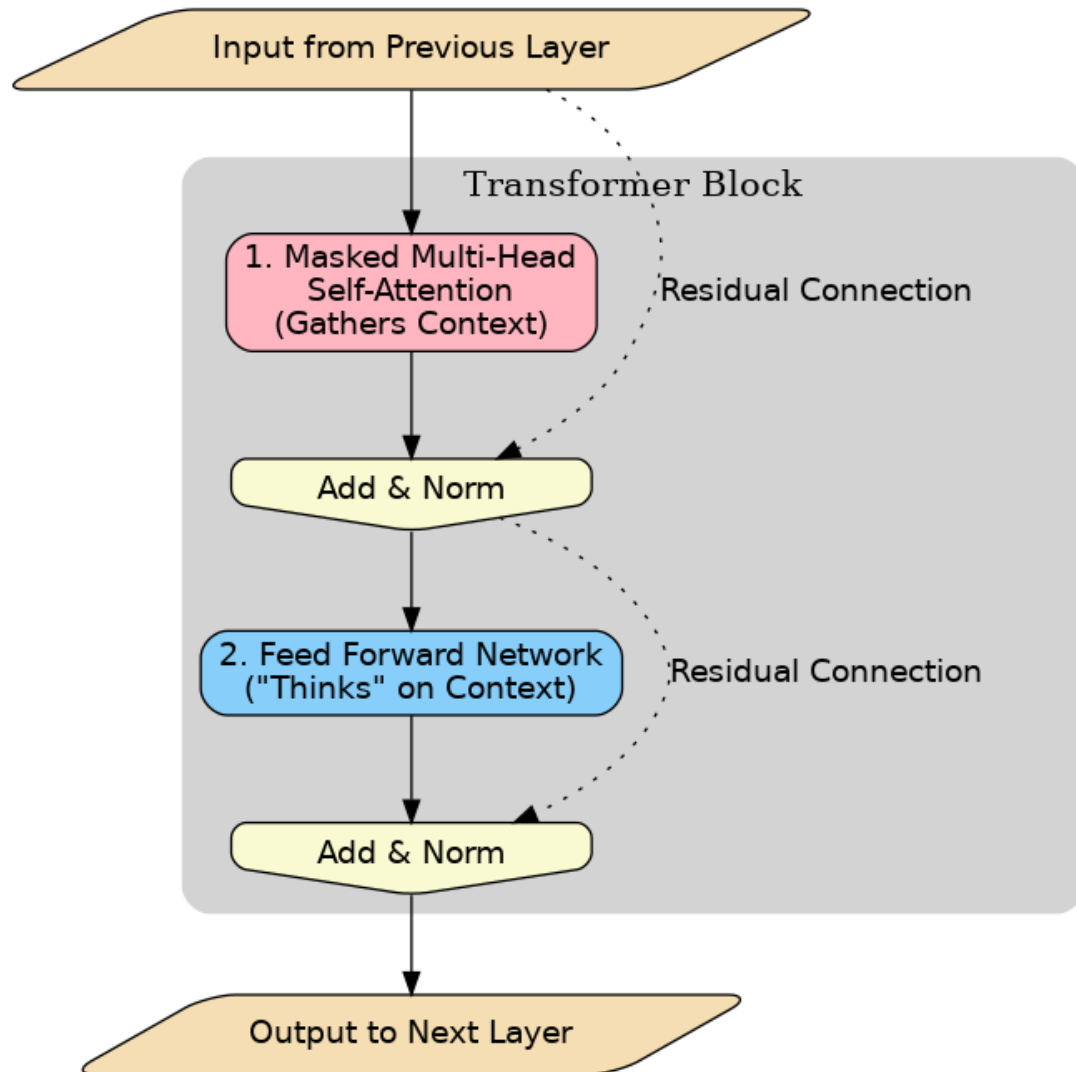


Diagram 3: The Output Layer & Generation Loop

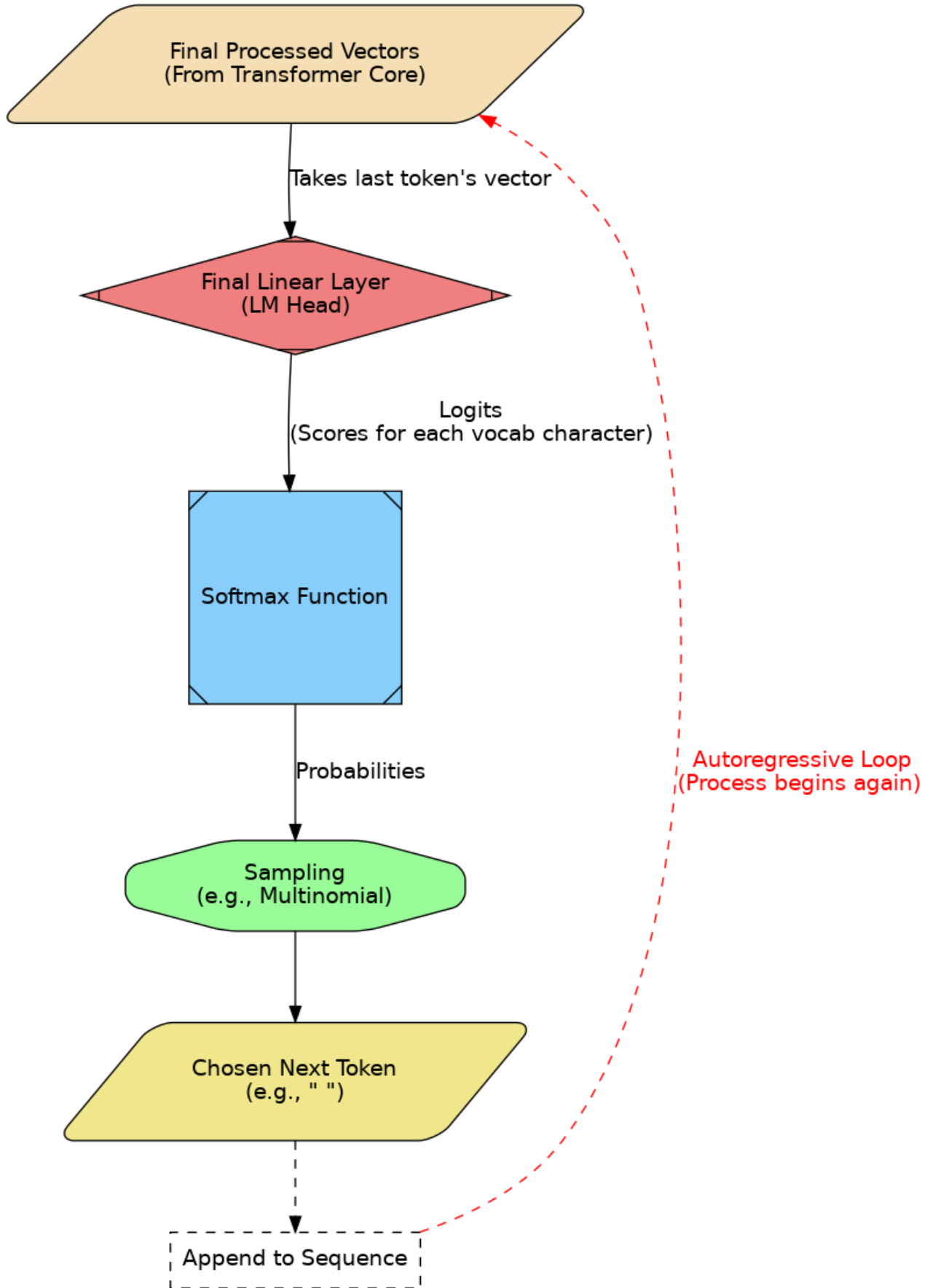


Diagram 4: Training Sequence Diagram

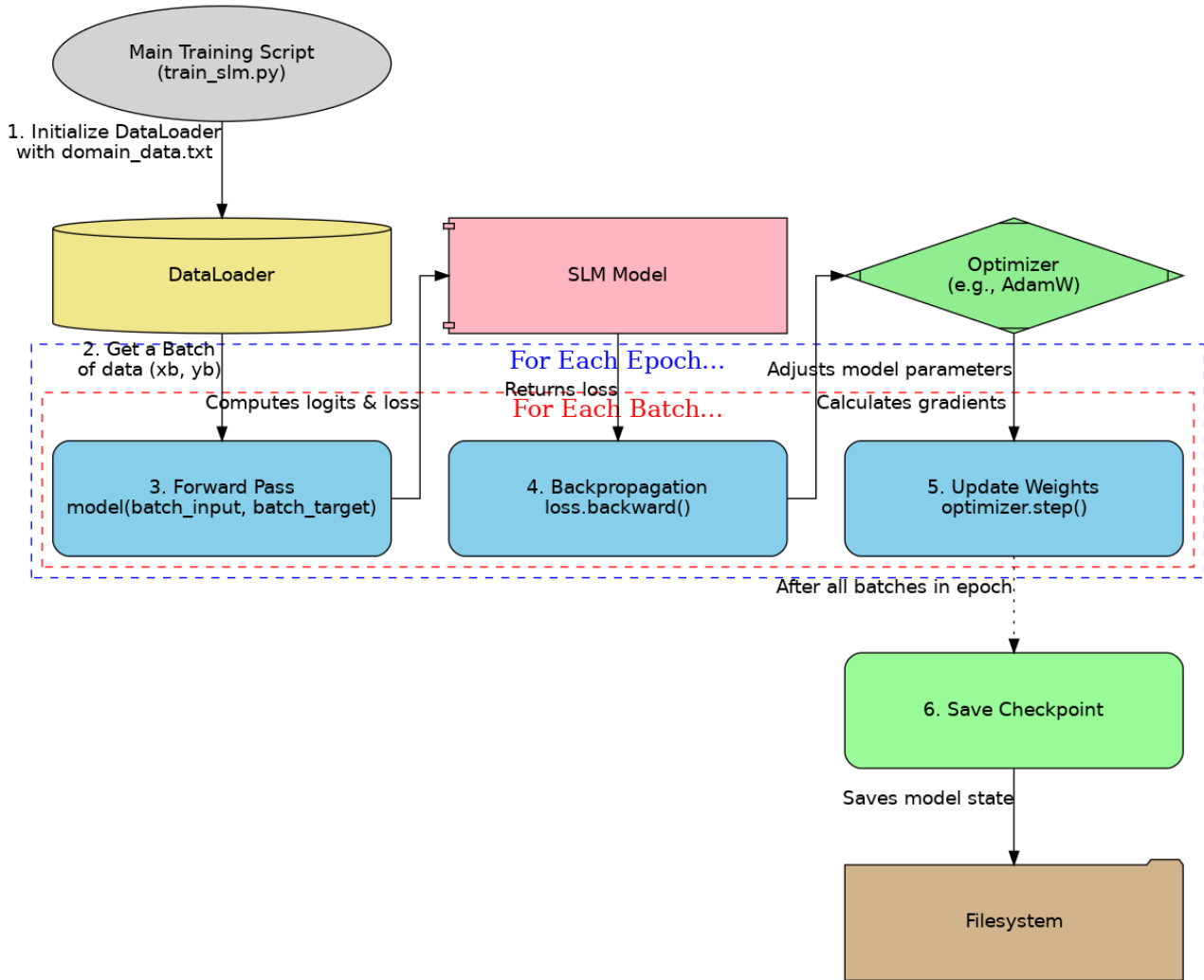
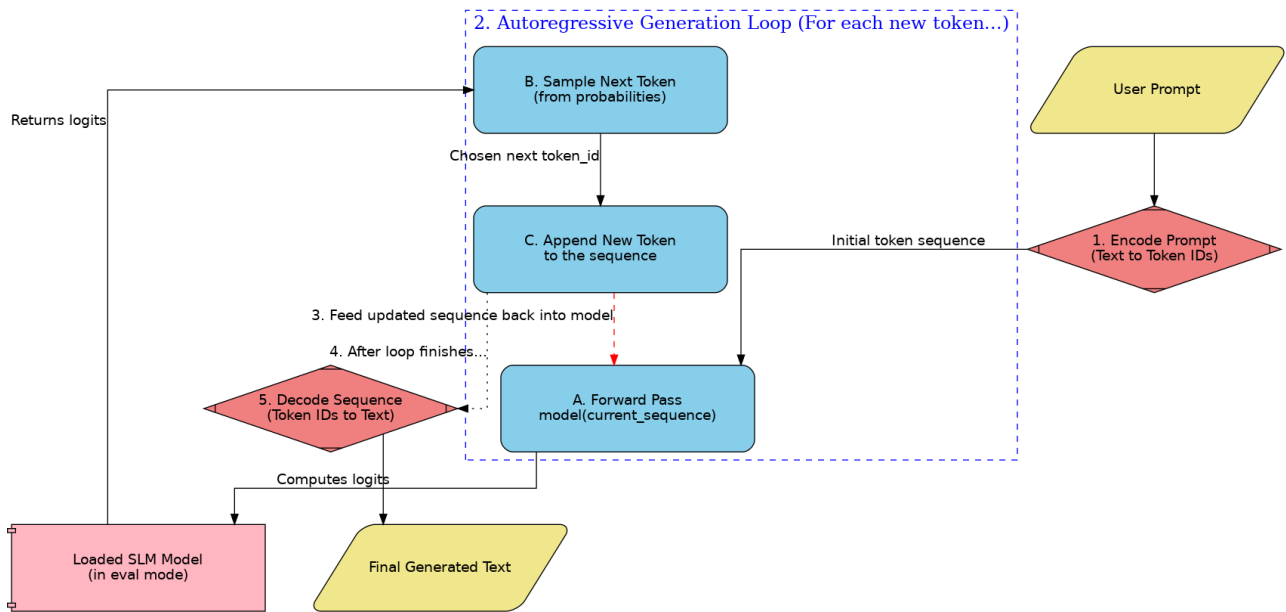


Diagram 5: Inference (Text Generation) Sequence Diagram



Hands – On

Dataset Text is copied from this book.

**The
Song Celestial.
or
Bhagavad-Gita
(From the Mahabharata)**

Being a Discourse Between Arjuna,
Prince of India, and the Supreme Being
Under the Form of Krishna

Translated from the Sanskrit Text
by
Sir Edwin Arnold,
M.A., K.C.I.E., C.S.I.

New York
Truslove, Hanson & Comba, Ltd.
67 Fifth Avenue
1900

Dedication

TO INDIA

So have I read this wonderful and spirit-thrilling speech,
By Krishna and Prince Arjun held, discoursing each with each;
So have I writ its wisdom here,—its hidden mystery,
For England; O our India! as dear to me as She!

EDWIN ARNOLD

Training the Model

First Training is configured for 100 epoches. But learning is completed around 43 rd epoch. We can see that Validation loss reached to a bottom and after that it starts increasing steadily

```
Activities Terminal Jul 18 19:23 natarajan@natarajan-ZEB-NBC-SS: ~/slm

2025-07-18 15:44:49.919 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_30.pth
Epoch 31/100: 100%
2025-07-18 15:59:46.940 - INFO - Epoch 31 | Training Loss: 0.3279 | 6464/6464 [14:57:00:00, 7.21it/s, loss=0.326]
2025-07-18 16:00:04.402 - INFO - Epoch 31 | Validation Loss: 0.1981
2025-07-18 16:00:04.493 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_31.pth
Epoch 32/100: 100%
2025-07-18 16:15:24.723 - INFO - Epoch 32 | Training Loss: 0.3253 | 6464/6464 [15:20:00:00, 7.02it/s, loss=0.322]
2025-07-18 16:15:42.375 - INFO - Epoch 32 | Validation Loss: 0.1984
2025-07-18 16:15:42.404 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_32.pth
Epoch 33/100: 100%
2025-07-18 16:30:34.866 - INFO - Epoch 33 | Training Loss: 0.3225 | 6464/6464 [14:52:00:00, 7.24it/s, loss=0.322]
2025-07-18 16:30:52.703 - INFO - Epoch 33 | Validation Loss: 0.1964
2025-07-18 16:30:52.743 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_33.pth
Epoch 34/100: 100%
2025-07-18 16:46:27.923 - INFO - Epoch 34 | Training Loss: 0.3195 | 6464/6464 [15:35:00:00, 6.91it/s, loss=0.316]
2025-07-18 16:46:45.304 - INFO - Epoch 34 | Validation Loss: 0.1956
2025-07-18 16:46:45.334 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_34.pth
Epoch 35/100: 100%
2025-07-18 17:02:15.406 - INFO - Epoch 35 | Training Loss: 0.3173 | 6464/6464 [15:30:00:00, 6.95it/s, loss=0.303]
2025-07-18 17:02:33.093 - INFO - Epoch 35 | Validation Loss: 0.1956
2025-07-18 17:02:33.128 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_35.pth
Epoch 36/100: 100%
2025-07-18 17:17:55.564 - INFO - Epoch 36 | Training Loss: 0.3149 | 6464/6464 [15:22:00:00, 7.01it/s, loss=0.328]
2025-07-18 17:18:13.305 - INFO - Epoch 36 | Validation Loss: 0.1947
2025-07-18 17:18:13.336 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_36.pth
Epoch 37/100: 100%
2025-07-18 17:33:26.774 - INFO - Epoch 37 | Training Loss: 0.3128 | 6464/6464 [15:13:00:00, 7.08it/s, loss=0.315]
2025-07-18 17:33:43.901 - INFO - Epoch 37 | Validation Loss: 0.1949
2025-07-18 17:33:43.937 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_37.pth
Epoch 38/100: 100%
2025-07-18 17:48:25.364 - INFO - Epoch 38 | Training Loss: 0.3107 | 6464/6464 [14:41:00:00, 7.33it/s, loss=0.316]
2025-07-18 17:48:42.661 - INFO - Epoch 38 | Validation Loss: 0.1928
2025-07-18 17:48:42.694 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_38.pth
Epoch 39/100: 100%
2025-07-18 18:04:22.917 - INFO - Epoch 39 | Training Loss: 0.3086 | 6464/6464 [15:40:00:00, 6.87it/s, loss=0.295]
2025-07-18 18:04:40.225 - INFO - Epoch 39 | Validation Loss: 0.1920
2025-07-18 18:04:40.254 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_39.pth
Epoch 40/100: 100%
2025-07-18 18:19:52.832 - INFO - Epoch 40 | Training Loss: 0.3066 | 6464/6464 [15:12:00:00, 7.08it/s, loss=0.319]
2025-07-18 18:20:17.099 - INFO - Epoch 40 | Validation Loss: 0.1930
2025-07-18 18:20:17.127 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_40.pth
Epoch 41/100: 100%
2025-07-18 18:35:51.979 - INFO - Epoch 41 | Training Loss: 0.3050 | 6464/6464 [15:34:00:00, 6.91it/s, loss=0.315]
2025-07-18 18:36:07.958 - INFO - Epoch 41 | Validation Loss: 0.1923
2025-07-18 18:36:07.980 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_41.pth
Epoch 42/100: 100%
2025-07-18 18:51:36.306 - INFO - Epoch 42 | Training Loss: 0.3029 | 6464/6464 [15:28:00:00, 6.96it/s, loss=0.336]
2025-07-18 18:51:49.700 - INFO - Epoch 42 | Validation Loss: 0.1922
2025-07-18 18:51:49.722 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_42.pth
Epoch 43/100: 100%
2025-07-18 19:06:53.463 - INFO - Epoch 43 | Training Loss: 0.3011 | 6464/6464 [15:03:00:00, 7.15it/s, loss=0.31]
2025-07-18 19:07:10.025 - INFO - Epoch 43 | Validation Loss: 0.1909
2025-07-18 19:07:10.046 - INFO - Checkpoint saved to checkpoints v2/checkpoint_epoch_43.pth
Epoch 44/100: 100%
2025-07-18 19:22:42.078 - INFO - Epoch 44 | Training Loss: 0.2998 | 6464/6464 [15:32:00:00, 6.94it/s, loss=0.286]
2025-07-18 19:22:59.733 - INFO - Epoch 44 | Validation Loss: 0.1908
```

Once the training is over , we have to evaluate the model.

All LLMs, it may be OpenAI, Gemini , Claude, but most of the LLMs main function is just generation of text. If we give some text as input to these LLMs , based on its training , the LLM will try to find the next probable word , add it to the given input, and then continue this in loop until the number of the words generated crosses the max limit configured .

So our small language model also expected to do the same thing. Ie whether it is able to generate text or not.

Here there is a major difference in between our small language model and the LLMs created by the industry giants.

Common LLMs are created by spending a large amount of money in training as they will be trained with almost all available data in internet and other sources. So , in the training, they learn most of the things , almost all.

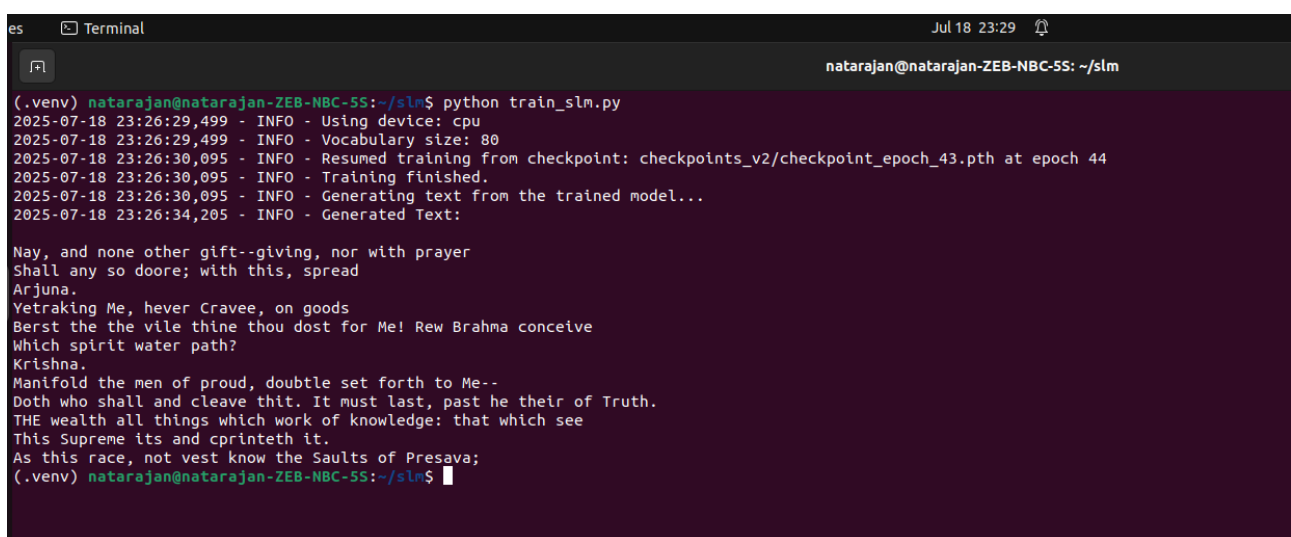
But our LLM is created with different purpose, I am not ready to spend huge money in training this small LLM. And I dont want to train my model with all kinds of knowledge , but I want to teach / train my model , just only one subject.

Ie it is not a generic model. It is subject matter expert in only one domain. For the current POC, I have choosen Gita as the domain.

It is created in my small laptop within 8 hours training using simple python scripts. It has learned in the training all Eighteen chapters of Gita. So it can generate text from Gita only.

So once the training is over, we use a test script to check its text generation capability.

We can see the text generated by our LLM in the screen shot below.



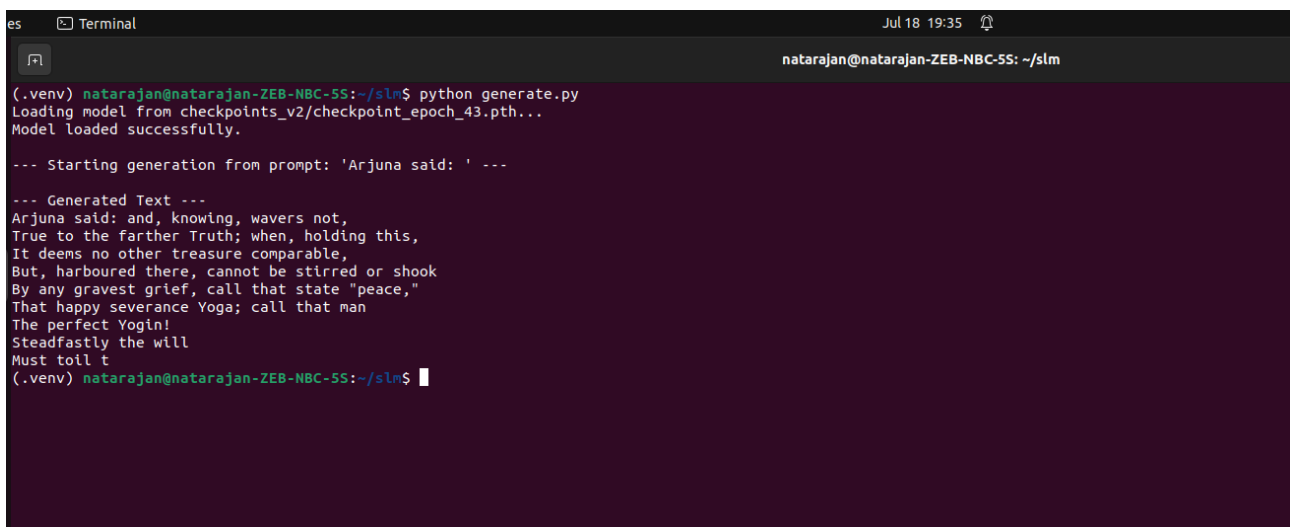
```
es Terminal Jul 18 23:29 natarajan@natarajan-ZEB-NBC-5S: ~/slm
(.venv) natarajan@natarajan-ZEB-NBC-5S:~/slm$ python train_slm.py
2025-07-18 23:26:29,499 - INFO - Using device: cpu
2025-07-18 23:26:29,499 - INFO - Vocabulary size: 80
2025-07-18 23:26:30,095 - INFO - Resumed training from checkpoint: checkpoints_v2/checkpoint_epoch_43.pth at epoch 44
2025-07-18 23:26:30,095 - INFO - Training finished.
2025-07-18 23:26:30,095 - INFO - Generating text from the trained model...
2025-07-18 23:26:34,205 - INFO - Generated Text:
Nay, and none other gift--giving, nor with prayer
Shall any so doore; with this, spread
Arjuna.
Yetraking Me, hever Cravee, on goods
Berst the the vile thine thou dost for Me! Rew Brahma conceive
Which spirit water path?
Krishna.
Manifold the men of proud, doubtle set forth to Me--
Doth who shall and cleave thit. It must last, past he their of Truth.
THE wealth all things which work of knowledge: that which see
This Supreme its and cprinteth it.
As this race, not vest know the Saults of Presava;
(.venv) natarajan@natarajan-ZEB-NBC-5S:~/slm$
```

Testing Models Response for various types of Prompts and our Models Responses for them.

I have not created web chat application for entering prompts to our model and displaying their responses like chatbots. For simplicity I am using python script to set the prompt , invoke the model, get the responses.

We can see the various prompts I have used and the responses generated by our model in the next three screenshots.

These screenshots makes you clear, how our small language model reads the prompt and generating responses for it.

A terminal window with a dark background and light text. The title bar shows 'es' and 'Terminal'. The top right corner displays 'Jul 18 19:35' and a bell icon. The terminal content shows a user running a Python script 'generate.py'. The script loads a model from 'checkpoints_v2/checkpoint_epoch_43.pth'. It then starts generation from the prompt 'Arjuna said: '. The output is a poem-like text starting with 'Arjuna said: and, knowing, wavers not, True to the farther Truth; when, holding this, It deems no other treasure comparable, But, harboured there, cannot be stirred or shook By any gravest grief, call that state "peace," That happy severance Yoga; call that man The perfect Yogin! Steadfastly the will Must toil t'. The prompt is enclosed in single quotes in the terminal output.

```
es Terminal Jul 18 19:35
natarajan@natarajan-ZEB-NBC-55: ~/slm

(.venv) natarajan@natarajan-ZEB-NBC-55:~/slm$ python generate.py
Loading model from checkpoints_v2/checkpoint_epoch_43.pth...
Model loaded successfully.

--- Starting generation from prompt: 'Arjuna said: ' ---

--- Generated Text ---
Arjuna said: and, knowing, wavers not,
True to the farther Truth; when, holding this,
It deems no other treasure comparable,
But, harboured there, cannot be stirred or shook
By any gravest grief, call that state "peace,"
That happy severance Yoga; call that man
The perfect Yogin!
Steadfastly the will
Must toil t
(.venv) natarajan@natarajan-ZEB-NBC-55:~/slm$
```



```
es  Terminal  Jul 18 19:40  natarajan@natarajan-ZEB-NBC-55: ~/slm

(.venv) natarajan@natarajan-ZEB-NBC-55:~/slm$ python generate.py
Loading model from checkpoints_v2/checkpoint_epoch_43.pth...
Model loaded successfully.

--- Starting generation from prompt: 'Q: What is the main lesson of the Bhagavad Gita? A:' ---

--- Generated Text ---
Q: What is the main lesson of the Bhagavad Gita? A:
Yea! know Me Soul of Soul in thy own soul!
Fight! vanquish foes and doubts, dear Hero! slay
What haunts thee in fond shapes, and would betray!
HERE ENDETH CHAPTER III. OF THE BHAGAVAD-GITA,
Entitled "Jnana Yog,"
Or "The Book of Religion by Self-Restraint."
CHAPTER VII
Krishna.
Learn now, dear Princ
(.venv) natarajan@natarajan-ZEB-NBC-55:~/slm$ ^C
(.venv) natarajan@natarajan-ZEB-NBC-55:~/slm$
```

```
(.venv) natarajan@natarajan-ZEB-NBC-5S:~/slm$ python generate.py
Loading model from checkpoints_v2/checkpoint_epoch_43.pth...
Model loaded successfully.
```

```
--- Starting generation from prompt: 'Therefore, O Prince, the wise man acts without thought of reward, seeing' ---
```

```
--- Generated Text ---
```

```
Therefore, O Prince, the wise man acts without thought of reward, seeing the Lords of Hearts,
And seeking fasts, breathe from the bond
Between the senses and the sense-world. Sweet
As Amrit is its first taste, but its last
Bitter as poison. 'Tis of Rajas, Prince!
And foul and "dark" the Pleasure is which thou knowest not,
O slayer of thy Foes! Albeit I be
Unborn, undying
```

```
(.venv) natarajan@natarajan-ZEB-NBC-5S:~/slm$ python generate.py
Loading model from checkpoints_v2/checkpoint_epoch_43.pth...
Model loaded successfully.
```

```
--- Starting generation from prompt: 'The Self, which is unborn and eternal,' ---
```

```
--- Generated Text ---
```

```
The Self, which is unborn and eternal, which is
Fades back again to Him who sent it forth;
Yea! this vast company of living things--
Again and yet again produced--expires
At Brahma's Nightfall; and, at Brahma's Dawn,
Riseth, without its will, to life new-born.
But--higher, deeper, innermost--abides
Another Life, not like the life of men
```

```
(.venv) natarajan@natarajan-ZEB-NBC-5S:~/slm$ python generate.py
Loading model from checkpoints_v2/checkpoint_epoch_43.pth...
Model loaded successfully.
```

```
--- Starting generation from prompt: 'He who is free from the grip of desire' ---
```

```
--- Generated Text ---
```

```
He who is free from the grip of desire,
Burned clean in act by the white fire of truth,
The wise call that man wise; and such an one
Who worship Me, and with heart and mind! so shalt thou dost laud
Surcease of works, and, at another time,
Service through work. Of these twain plainly tell
Which is the better end?
Krishna.
```

```
I told thee, bl
```

```
(.venv) natarajan@natarajan-ZEB-NBC-5S:~/slm$
```

```
es  Terminal  Jul 18 19:58  natarajan@natarajan-ZEB-NBC-5S: ~/slm

(.venv) natarajan@natarajan-ZEB-NBC-5S:~/slm$ python generate.py
Loading model from checkpoints_v2/checkpoint_epoch_43.pth...
Model loaded successfully.

--- Starting generation from prompt: 'Dharma' ---

--- Generated Text ---
Dharmans, Viswest, Unmapas;
Maruts, and those great Twins
The heavenly, fair, Aswins,
Gandharvas, Rakshasas, Siddhas, and Asuras,[FN#22]--
These see Thee, and revere
In sudden-stricken fear;
Yea! the Worlds,--seeing Thee with form stupendous,
With faces manifold,
With eyes which all behold,
Unnumbered eye
(.venv) natarajan@natarajan-ZEB-NBC-5S:~/slm$ python generate.py
Loading model from checkpoints_v2/checkpoint_epoch_43.pth...
Model loaded successfully.

--- Starting generation from prompt: 'karma' ---

--- Generated Text ---
karmany doubts.
He that, being self-contained, hath vanquished doubt,
Disparting self from service, soul from works,
Enlightened and emancipate, my Prince!
Works fetter him no more! Cut then atwain
With sword of wisdom, Son of Bharata!
This doubt that binds thy heart-beats! cleave the bond
Born of thy ig
(.venv) natarajan@natarajan-ZEB-NBC-5S:~/slm$ python generate.py
Loading model from checkpoints_v2/checkpoint_epoch_43.pth...
Model loaded successfully.

--- Starting generation from prompt: 'karma yog' ---

--- Generated Text ---
karma yog,
Truthful and deed: and, ever with tender smile,
Such is the Saint!--not!--nor, with human eyes, Arjuna! ever mayest!
Therefore I give thee sense divine. Have other eyes, new light!
And, look! This is My glory, unveiled to mortal sight!
Sanjaya.
Then, O King! the God, so saying,
Stood, to Pritha's
(.venv) natarajan@natarajan-ZEB-NBC-5S:~/slm$
```

Model Files

es

Archive Manager

Jul 19 00:05

Extract

+

checkpoint_epoch_43.pth

<

>

Location: /checkpoint_epoch_43/

Name	Size	Type	Modified
.data	40 bytes	Folder	
data	15.0 MB	Folder	
.format_version	1 byte	unknown	30 November 1979, 00:00
.storage_alignment	2 bytes	unknown	30 November 1979, 00:00
byteorder	6 bytes	unknown	30 November 1979, 00:00
data.pkl	59.6 kB	unknown	30 November 1979, 00:00
version	2 bytes	unknown	30 November 1979, 00:00

So far we have seen What is the data set we have used to train our language model?

Once training is over , how our model generates text from Gita and how it responds for our various types of Prompts.

Also we have seen the files generated by our model training.

With this POC , a small exercise, but very powerful, as it explains us what is happening while we train our LLM models, what is happening when we are querying (doing inference) it.

We dont have to spend Million dollars as the current Industial giants LLMs, but with our standard hardware , with very less cost, we can create our own LLM for our own requirements.

Our small LLM s may not have elaborate general knowledge as the industrial giants LLMs, but we can produce subject matter expert or specialist LLM in our own domain .

One thing, if it is a block box, we can not understand it and utilise it fully , but once understood the core underlying concepts, either we can build our own LLM as per our requirements, or we can use the commercial LLM providers very effectively with this thorough understanding.

Git Hub Project Link

**[https://github.com/Natarajan-R/
SLM_from_SCRATCH](https://github.com/Natarajan-R/SLM_from_SCRATCH)**

THANK YOU

NATARAJAN RAMASAMY

natarajansr@gmail.com