

Next Token Generation

Nipun Batra

IIT Gandhinagar

August 2, 2025

Inspiration and Relevance

Inspired by the great lecture from Andrej Karpathy

- Search for "Neural Networks: Zero to Hero" to find the original lecture

Inspiration and Relevance

Inspired by the great lecture from Andrej Karpathy

- Search for "Neural Networks: Zero to Hero" to find the original lecture

Inspiration and Relevance

Inspired by the great lecture from Andrej Karpathy

- Search for "Neural Networks: Zero to Hero" to find the original lecture
- This approach is fundamental to modern language models

Inspiration and Relevance

Inspired by the great lecture from Andrej Karpathy

- Search for "Neural Networks: Zero to Hero" to find the original lecture
- This approach is fundamental to modern language models

Inspiration and Relevance

Inspired by the great lecture from Andrej Karpathy

- Search for "Neural Networks: Zero to Hero" to find the original lecture
- This approach is fundamental to modern language models
- **Direct connection to ChatGPT:**

**Understanding this simple version helps grasp
ChatGPT's foundation!**

Inspiration and Relevance

Inspired by the great lecture from Andrej Karpathy

- Search for "Neural Networks: Zero to Hero" to find the original lecture
- This approach is fundamental to modern language models
- **Direct connection to ChatGPT:**
 - Same core principle: predict the next token

**Understanding this simple version helps grasp
ChatGPT's foundation!**

Inspiration and Relevance

Inspired by the great lecture from Andrej Karpathy

- Search for "Neural Networks: Zero to Hero" to find the original lecture
- This approach is fundamental to modern language models
- **Direct connection to ChatGPT:**
 - Same core principle: predict the next token
 - Scaled up from characters to words/subwords

**Understanding this simple version helps grasp
ChatGPT's foundation!**

Inspiration and Relevance

Inspired by the great lecture from Andrej Karpathy

- Search for "Neural Networks: Zero to Hero" to find the original lecture
- This approach is fundamental to modern language models
- **Direct connection to ChatGPT:**
 - Same core principle: predict the next token
 - Scaled up from characters to words/subwords
 - Uses transformer architecture instead of MLP

**Understanding this simple version helps grasp
ChatGPT's foundation!**

What is the Next Character?

app

?

What is the next character?

Classification Task

app

?

We can pose this as a classification task

Classification Task

app

?

We can pose this as a classification task

Input:

app

Classification Task

app

?

We can pose this as a classification task

Input:
app

Output: Probability Distribution

Char	Prob	Char	Prob
a	0.01	n	0.01
b	0.01	o	0.01
c	0.01	p	0.01
...
l	0.45	z	0.01
m	0.01	-	0.05

Generate Indian Names

Specific Problem: Generate Indian Names

Dataset:

Generate Indian Names

Specific Problem: Generate Indian Names

Dataset:

Abid
Abhidha
Adesh
Aditya
Agam
...
:
Yash
Yogesh
Zara

Generate Indian Names

Specific Problem: Generate Indian Names

Dataset:

Abid
Abhidha
Adesh
Aditya
Agam
...
:
Yash
Yogesh
Zara

Generate Indian Names

Specific Problem: Generate Indian Names

Dataset:

Abid
Abhidha
Adesh
Aditya
Agam
...
:
Yash
Yogesh
Zara

- Collection of Indian names

Generate Indian Names

Specific Problem: Generate Indian Names

Dataset:

Abid
Abhidha
Adesh
Aditya
Agam
...
:
Yash
Yogesh
Zara

- Collection of Indian names
- Each name represents a sequence

Generate Indian Names

Specific Problem: Generate Indian Names

Dataset:

Abid
Abhidha
Adesh
Aditya
Agam
...
:
Yash
Yogesh
Zara

- Collection of Indian names
- Each name represents a sequence
- Goal: Learn to generate similar names

Assumptions

We'll make a few assumptions:

1. **Character set:** Only use 26 lowercase characters (a-z)

Assumptions

We'll make a few assumptions:

1. **Character set:** Only use 26 lowercase characters (a-z)

Assumptions

We'll make a few assumptions:

1. **Character set:** Only use 26 lowercase characters (a-z)
2. **End marker:** A hyphen (-) indicates the end character

Assumptions

We'll make a few assumptions:

1. **Character set:** Only use 26 lowercase characters (a-z)
2. **End marker:** A hyphen (-) indicates the end character

Assumptions

We'll make a few assumptions:

1. **Character set:** Only use 26 lowercase characters (a-z)
2. **End marker:** A hyphen (-) indicates the end character
3. **Length constraint:** Names are between 4 and 10 characters

Total vocabulary size: $26 + 1 = 27$ characters

Generate Training Dataset

Creating Training Data from "abid"

Using history/context of 3 characters:

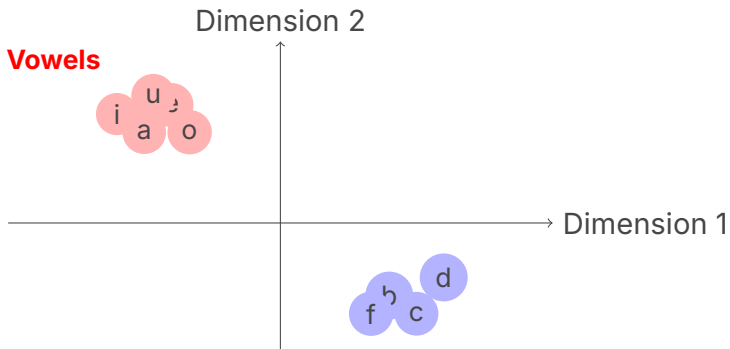
X (Input)		Y (Target)
[-, -, -]	→	a a>-, a
	→	b a, b>a, b
	→	i a, b, i>a, b, i
	→	d b, i, d>b, i, d
	→	-

Result: 5 training examples from one name "abid"

Representation Learning

Important Idea: Representation Learning

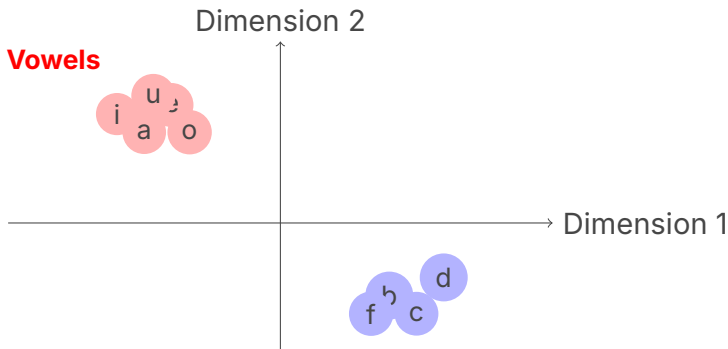
- Learn a vector representation for each character



Representation Learning

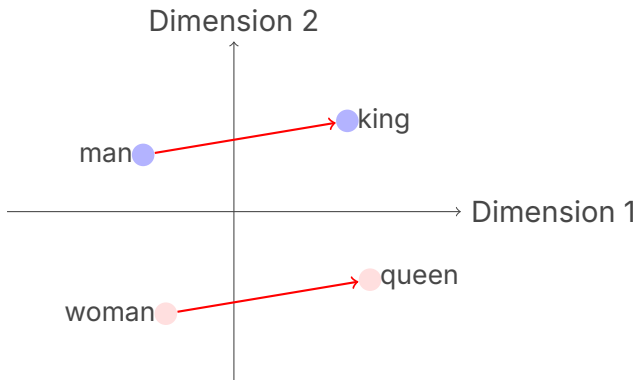
Important Idea: Representation Learning

- Learn a vector representation for each character
- Hope that similar characters will be closer in vector space



Word2Vec Reference

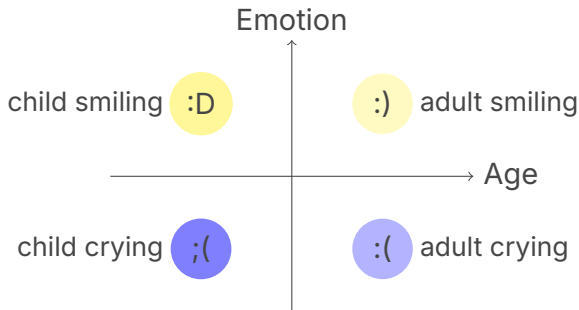
Classic Word2Vec Relationship



Relationship: $\text{queen} \approx \text{king} - \text{man} + \text{woman}$

Analogy with Smileys

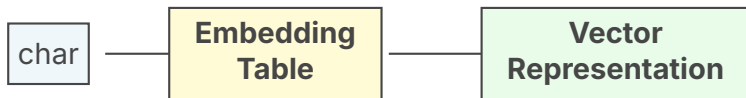
Emotional Expression Analogy



Relationship: child crying = child smiling + adult crying - adult smiling

Embedding Matrix/Table

Main Idea: Embedding Matrix/Table



Process: Character → Lookup in Embedding Table → Dense Vector

27 × K Embedding Matrix

Embedding Table Structure

Char	D1	D2	...	DK
a	0.2	-0.1	...	0.8
b	-0.3	0.5	...	-0.2
c	0.1	0.3	...	0.4
⋮	⋮	⋮	⋮	⋮
z	0.7	-0.4	...	0.1
-	0.0	0.9	...	-0.5

This overall becomes a 27 × K dimensional matrix

Learnable Matrix

This matrix is learnable!

- **Initially:** Random values

Learnable Matrix

This matrix is learnable!

- **Initially:** Random values

Learnable Matrix

This matrix is learnable!

- **Initially:** Random values
- **During training:** Updated via backpropagation

Learnable Matrix

This matrix is learnable!

- **Initially:** Random values
- **During training:** Updated via backpropagation

Learnable Matrix

This matrix is learnable!

- **Initially:** Random values
- **During training:** Updated via backpropagation
- **After training:** Contains meaningful character representations

Learnable Matrix

This matrix is learnable!

- **Initially:** Random values
- **During training:** Updated via backpropagation
- **After training:** Contains meaningful character representations

Learnable Matrix

This matrix is learnable!

- **Initially:** Random values
- **During training:** Updated via backpropagation
- **After training:** Contains meaningful character representations
- **Similar characters:** Will have similar embedding vectors

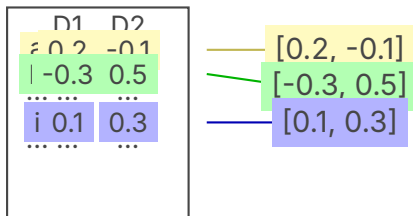
The network learns both the embeddings AND the classification weights!

Overall Architecture (2D Example)

Example with $X = \text{"abi"}$ and 2D embeddings

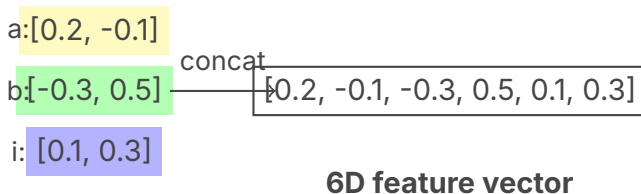
Input: $X = [\text{"a"}, \text{"b"}, \text{"i"}]$

Embedding Matrix (27×2)



Concatenate the Embeddings

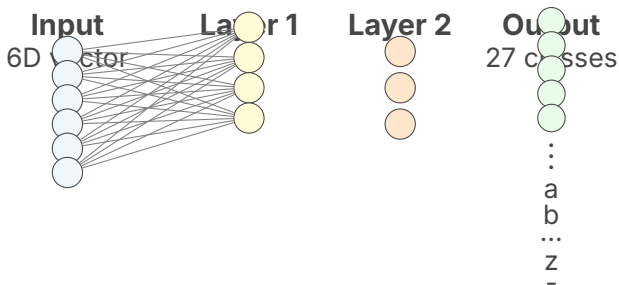
Feature Vector Creation for X = "abi"



The feature vector pulls up embeddings and concatenates them

Multi-Layer Perceptron

Neural Architecture



Eventually shows 27-class output vector

Cross-Entropy Loss

Learning Process

- **Loss Function:** Use cross-entropy loss to learn

Cross-Entropy Loss

Learning Process

- **Loss Function:** Use cross-entropy loss to learn

Cross-Entropy Loss

Learning Process

- **Loss Function:** Use cross-entropy loss to learn
- **We are learning two things:**

Cross-Entropy Loss

Learning Process

- **Loss Function:** Use cross-entropy loss to learn
- **We are learning two things:**
 1. The embedding matrix ($27 \times K$ parameters)

Cross-Entropy Loss

Learning Process

- **Loss Function:** Use cross-entropy loss to learn
- **We are learning two things:**
 1. The embedding matrix ($27 \times K$ parameters)
 2. The MLP weights (neural network parameters)

Cross-Entropy Loss

Learning Process

- **Loss Function:** Use cross-entropy loss to learn
- **We are learning two things:**
 1. The embedding matrix ($27 \times K$ parameters)
 2. The MLP weights (neural network parameters)

Cross-Entropy Loss

Learning Process

- **Loss Function:** Use cross-entropy loss to learn
- **We are learning two things:**
 1. The embedding matrix ($27 \times K$ parameters)
 2. The MLP weights (neural network parameters)
- **Training Process:**

Cross-Entropy Loss

Learning Process

- **Loss Function:** Use cross-entropy loss to learn
- **We are learning two things:**
 1. The embedding matrix ($27 \times K$ parameters)
 2. The MLP weights (neural network parameters)
- **Training Process:**
 - Forward pass: Input \rightarrow Embeddings \rightarrow Concatenate \rightarrow MLP \rightarrow Probabilities

Cross-Entropy Loss

Learning Process

- **Loss Function:** Use cross-entropy loss to learn
- **We are learning two things:**
 1. The embedding matrix ($27 \times K$ parameters)
 2. The MLP weights (neural network parameters)
- **Training Process:**
 - Forward pass: Input \rightarrow Embeddings \rightarrow Concatenate \rightarrow MLP \rightarrow Probabilities
 - Compute cross-entropy loss against true next character

Cross-Entropy Loss

Learning Process

- **Loss Function:** Use cross-entropy loss to learn
- **We are learning two things:**
 1. The embedding matrix ($27 \times K$ parameters)
 2. The MLP weights (neural network parameters)
- **Training Process:**
 - Forward pass: Input \rightarrow Embeddings \rightarrow Concatenate \rightarrow MLP \rightarrow Probabilities
 - Compute cross-entropy loss against true next character
 - Backward pass: Update both embeddings and MLP weights

Generate/Sample from Learned Model

Test Input: "abi"

Probability vector for next character:

Next Char	Probability	Next Char	Probability
a	0.01	n	0.05
b	0.01	o	0.02
c	0.03	p	0.01
d	0.60
...	...	z	0.01

- ABIA would be 1%

Generate/Sample from Learned Model

Test Input: "abi"

Probability vector for next character:

Next Char	Probability	Next Char	Probability
a	0.01	n	0.05
b	0.01	o	0.02
c	0.03	p	0.01
d	0.60
...	...	z	0.01

- ABIA would be 1%
- ABIB would be 1%

Generate/Sample from Learned Model

Test Input: "abi"

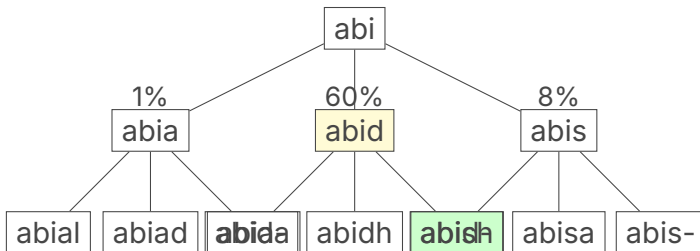
Probability vector for next character:

Next Char	Probability	Next Char	Probability
a	0.01	n	0.05
b	0.01	o	0.02
c	0.03	p	0.01
d	0.60
...	...	z	0.01

- ABIA would be 1%
- ABIB would be 1%
- **ABID would be 60%**

Tree Structure

Generation as Tree Structure



Had we chosen A, it starts a new branch. Had we chosen D, it starts a new branch, etc.

Temperature Term

Temperature in Softmax

- **Standard Softmax:**

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{27} e^{z_j}} \quad (1)$$

Temperature Term

Temperature in Softmax

- **Standard Softmax:**

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{27} e^{z_j}} \quad (1)$$

Temperature Term

Temperature in Softmax

- **Standard Softmax:**

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{27} e^{z_j}} \quad (1)$$

- **Temperature-scaled Softmax:**

$$P(y_i) = \frac{e^{z_i/T}}{\sum_{j=1}^{27} e^{z_j/T}} \quad (2)$$

Temperature Term

Temperature in Softmax

- **Standard Softmax:**

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{27} e^{z_j}} \quad (1)$$

- **Temperature-scaled Softmax:**

$$P(y_i) = \frac{e^{z_i/T}}{\sum_{j=1}^{27} e^{z_j/T}} \quad (2)$$

Temperature Term

Temperature in Softmax

- **Standard Softmax:**

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{27} e^{z_j}} \quad (1)$$

- **Temperature-scaled Softmax:**

$$P(y_i) = \frac{e^{z_i/T}}{\sum_{j=1}^{27} e^{z_j/T}} \quad (2)$$

- **Temperature Effects:**

Temperature Term

Temperature in Softmax

- **Standard Softmax:**

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{27} e^{z_j}} \quad (1)$$

- **Temperature-scaled Softmax:**

$$P(y_i) = \frac{e^{z_i/T}}{\sum_{j=1}^{27} e^{z_j/T}} \quad (2)$$

- **Temperature Effects:**

- $T = 1$: Default/standard probabilities

Temperature Term

Temperature in Softmax

- **Standard Softmax:**

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{27} e^{z_j}} \quad (1)$$

- **Temperature-scaled Softmax:**

$$P(y_i) = \frac{e^{z_i/T}}{\sum_{j=1}^{27} e^{z_j/T}} \quad (2)$$

- **Temperature Effects:**

- $T = 1$: Default/standard probabilities
- $T \rightarrow 0$: Very low temperature \rightarrow more peaked (deterministic)

Temperature Term

Temperature in Softmax

- **Standard Softmax:**

$$P(y_i) = \frac{e^{z_i}}{\sum_{j=1}^{27} e^{z_j}} \quad (1)$$

- **Temperature-scaled Softmax:**

$$P(y_i) = \frac{e^{z_i/T}}{\sum_{j=1}^{27} e^{z_j/T}} \quad (2)$$

- **Temperature Effects:**

- $T = 1$: Default/standard probabilities
- $T \rightarrow 0$: Very low temperature \rightarrow more peaked (deterministic)
- $T \rightarrow \infty$: Very high temperature \rightarrow more uniform (random)

Temperature Variations

How sampling differs across temperatures

Next Char	Default T=1.0	Low T T=0.5	High T T=2.0
a	0.01	0.001	0.08
d	0.60	0.95	0.25
s	0.08	0.01	0.12
h	0.03	0.005	0.09
-	0.05	0.02	0.11
others	0.23	0.015	0.35

- **Low Temperature:** Conservative, predictable generation

Temperature Variations

How sampling differs across temperatures

Next Char	Default T=1.0	Low T T=0.5	High T T=2.0
a	0.01	0.001	0.08
d	0.60	0.95	0.25
s	0.08	0.01	0.12
h	0.03	0.005	0.09
-	0.05	0.02	0.11
others	0.23	0.015	0.35

- **Low Temperature:** Conservative, predictable generation
- **High Temperature:** Creative, diverse generation