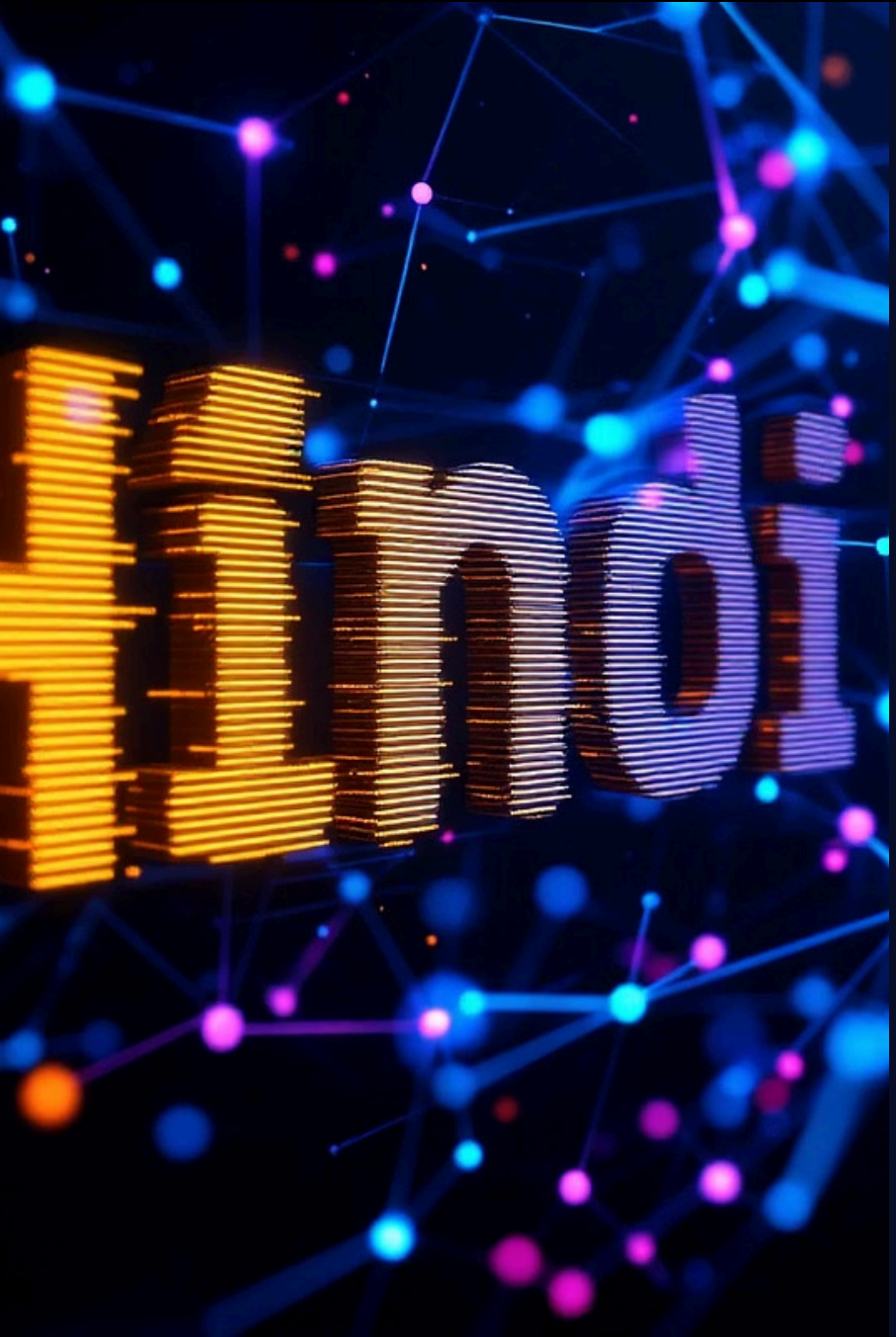


अपना खुद का LLM बनाना सीखें - शुरुआत से अंत तक

डेटा से टेक्स्ट जनरेशन तक का सफर (आसान हिंदी में)

मुख्य अवधारणाएं

LLM क्या होता है?

लार्ज लैंग्वेज मॉडल एक प्रकार का AI सिस्टम है जो बड़ी मात्रा में टेक्स्ट डेटा पर प्रशिक्षित होता है और मानव जैसा टेक्स्ट जनरेट कर सकता है।

टोकनाइज़ेशन क्यों ज़रूरी है?

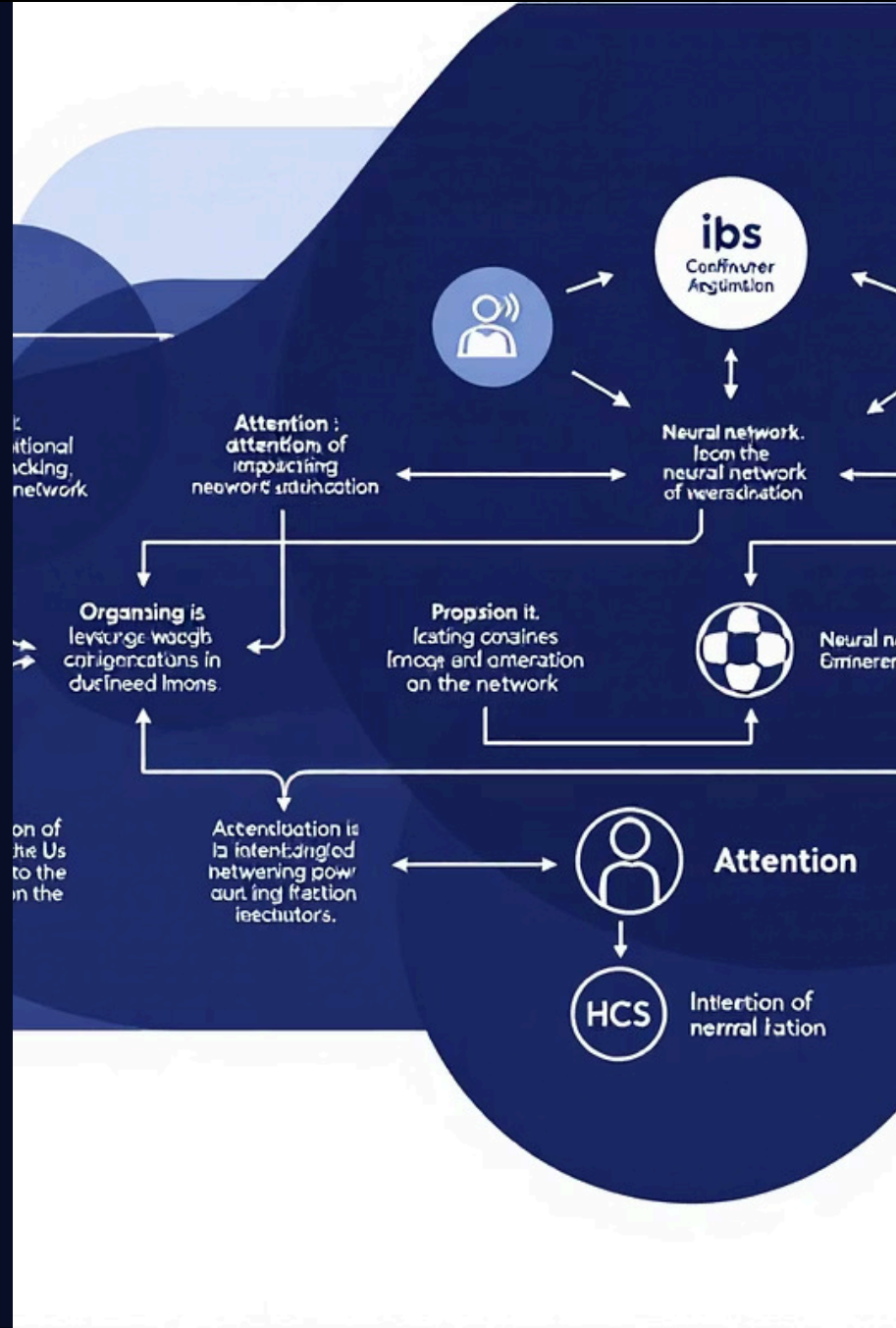
टोकनाइज़ेशन टेक्स्ट को छोटे-छोटे हिस्सों (टोकन) में बदलता है जिन्हें मॉडल समझ सकता है, यह भाषा प्रोसेसिंग का पहला महत्वपूर्ण चरण है।

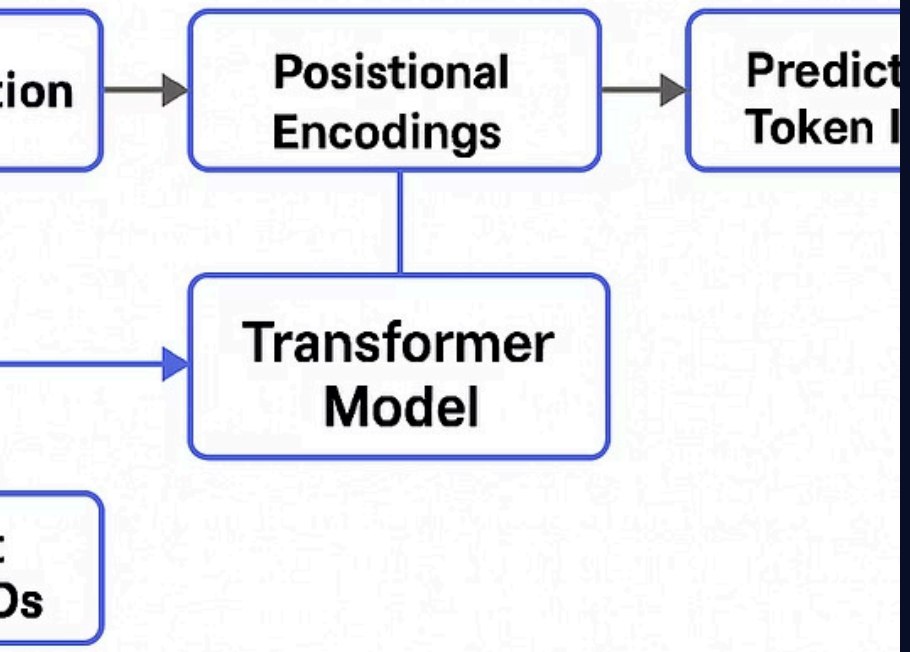
Transformer कैसे काम करता है?

ट्रांसफॉर्मर एक न्यूरल नेटवर्क आर्किटेक्चर है जो सेल्फ-अटेंशन मैकेनिज्म का उपयोग करके टेक्स्ट में शब्दों के बीच संबंधों को समझता है।

Self-Attention क्या होता है?

सेल्फ-अटेंशन एक तकनीक है जो मॉडल को सिखाती है कि वाक्य में कौन से शब्द एक-दूसरे से संबंधित हैं और कितने महत्वपूर्ण हैं।





Level Architecture of Training an

आर्किटेक्चर फ्लो

इनपुट टेक्स्ट टोकनाइज़र पोज़िशनल एम्बेडिंग

टेक्स्ट को टोकन में बदलकर उन्हें वेक्टर में कन्वर्ट किया जाता है और फिर पोजिशन की जानकारी जोड़ी जाती है।

Transformer ब्लॉक्स आउटपुट स्कोर

एम्बेडिंग को कई ट्रांसफॉर्मर लेयर्स से गुजारा जाता है जो अटेंशन मैकेनिज्म का उपयोग करके आउटपुट स्कोर जनरेट करते हैं।

टारगेट से तुलना लॉस (Cross Entropy)

मॉडल के आउटपुट की वास्तविक टारगेट से तुलना करके क्रॉस एंट्रॉपी लॉस की गणना की जाती है जिससे मॉडल सीखता है।

कोड की मुख्य बातें

खुद का टोकनाइज़र (हिंदी सपोर्ट)

हिंदी भाषा के लिए अनुकूलित टोकनाइज़र जो देवनागरी वर्णों और शब्दों को सही तरीके से प्रोसेस कर सकता है।

TransformerBlock और FeedForward

मॉडल के मुख्य कंपोनेंट्स जो अटेंशन मैकेनिज्म और फीड फॉरवर्ड नेटवर्क को इम्प्लीमेंट करते हैं।

TextDataset क्लास

टेक्स्ट डेटा को लोड करने और प्रोसेस करने के लिए कस्टम डेटासेट क्लास जो ट्रेनिंग के लिए बैच तैयार करता है।

SimpleLLM मॉडल की परिभाषा

पूरे LLM की आर्किटेक्चर जिसमें एम्बेडिंग, ट्रांसफॉर्मर ब्लॉक्स और आउटपुट लेयर शामिल हैं।

ट्रेनिंग लूप और लॉस ट्रैकिंग - मॉडल को प्रशिक्षित करने के लिए इटरेशन लूप जो लॉस को ट्रैक करता है और मॉडल के परफॉर्मेंस को मॉनिटर करता है।

```
# SimpleLLM मॉडल कोड उदाहरण
class SimpleLLM(nn.Module):
 def __init__(self, vocab_size, d_model, nhead, num_layers):
 super().__init__()
 self.embedding = nn.Embedding(vocab_size, d_model)
 self.pos_encoding = PositionalEncoding(d_model)
 self.transformer_blocks = nn.ModuleList([
 TransformerBlock(d_model, nhead) for _ in range(num_layers)
 ])
 self.output = nn.Linear(d_model, vocab_size)
```

जनरेशन और विज़ुअलाइज़ेशन

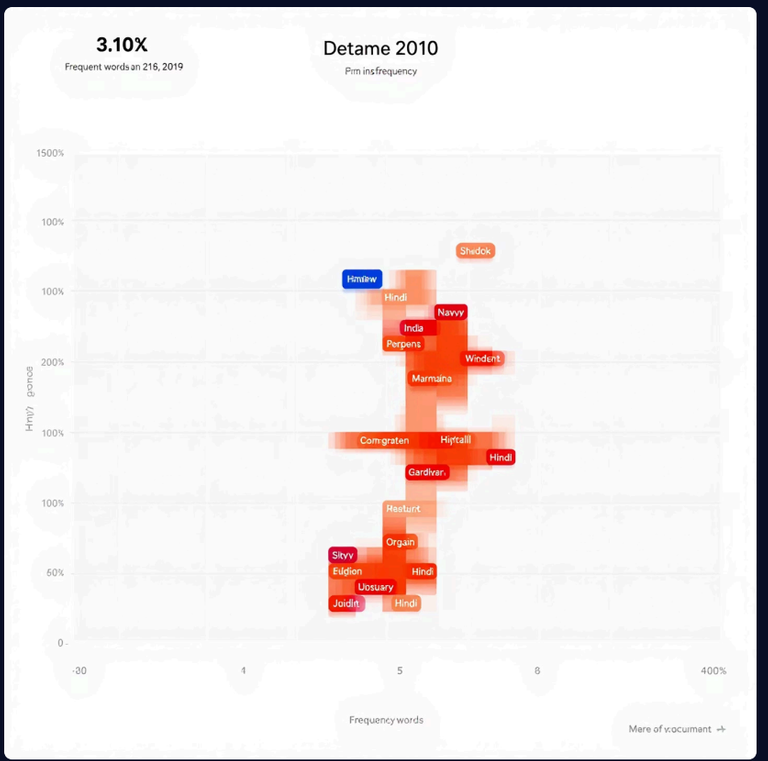
अपने मॉडल से हिंदी टेक्स्ट जनरेट करें

प्रशिक्षित मॉडल का उपयोग करके नया हिंदी टेक्स्ट जनरेट करें। आप विभिन्न प्रॉम्प्ट्स का उपयोग कर सकते हैं:

- प्रॉम्प्ट दें जैसे "मैं स्कूल..."
- मॉडल स्वाभाविक रूप से वाक्य को पूरा करेगा
- विभिन्न तापमान सेटिंग्स के साथ प्रयोग करें

हर लेयर का Attention वज़न देखें (Heatmap)

अटेंशन वेट्स को विज़ुअलाइज़ करके देखें कि मॉडल किन शब्दों पर ध्यान केंद्रित कर रहा है और कैसे वह संबंधों को समझता है।



अटेंशन हीटमैप विज़ुअलाइज़ेशन दिखाता है कि मॉडल कैसे शब्दों के बीच संबंध बनाता है

```
# टेक्स्ट जनरेशन कोड
def generate_text(model, tokenizer, prompt="मैं स्कूल", max_length=50, temperature=0.8):
 model.eval()
 tokens = tokenizer.encode(prompt)
 for _ in range(max_length):
 with torch.no_grad():
 logits = model(torch.tensor([tokens]))
 next_token = sample_next_token(logits, temperature)
 tokens.append(next_token)
 if next_token == tokenizer.eos_token_id:
 break
 return tokenizer.decode(tokens)
```