गहन खोज □3: बहु-शीर्ष लेटेंट ध्यान और बहु-टोकन भविष्यवाणी

□□□□□□□□□ □3 की इस जगह पर खोज की गई है, जिसमें "□□□□□-□□□□ □□□□□□□ □□□□□□□□□□ □□□ □□□□□-□□□□□ □□□□□□□□□□□ □□ □□□□□□□□□ □3"वीडियो का संदर्भ दिया गया है □□□□□://□□□□□.□□/□□49□□□□□□□□?□□=4□□2□□□-□□□□1□□□। □□□□□□□ □□□□□ □□□□□□□-□□-□□□□ का उपयोग वीडियो को ट्रांसक्राइब करने के लिए किया गया था, साथ ही कुछ कोड भी था जो ट्रांसक्रिप्ट को संगठित करने में मदद करता था।

□: □□□□ □□□ पर वापस आओ। आज हम बड़े भाषा मॉडलों की दुनिया में गहरी डाइव करेंगे। ठीक है, विशेष रूप से □□□□□□□□□ □3।

□: सुनता है। यह 671 अरब पैरामीटर मॉडल है, जो अपनी दक्षता और प्रदर्शन के लिए एक अनोखी रणनीति के लिए जागरूक है, ठीक है?

□: और आपने एक अकादमिक पेपर साझा किया है जिसमें इसका आर्किटेक्चर वर्णित है।

□: हाँ।

□: और एक मशीन लर्निंग विशेषज्ञ के रूप में, आप □□□□□□□□□ □3 को कैसे उच्च प्रदर्शन और आर्थिक ट्रेनिंग दोनों प्राप्त करता है, समझने की कोशिश कर रहे हैं।

□: हाँ, ठीक है।

□: ओह, हेयर, क्या चल रहा है?

□: □□□, □□□ और इसका काम कैसे होता है।

□: ओह, बिल्कुल। यह एक अच्छा विचार है। हाँ, हम निश्चित रूप से □□□ में और गहरे उतर सकते हैं। तो आप □□□ के नट्स और बोल्ट्स में रुचि रखते हैं। ठीक है, तो इसे खोलते हैं। हमने कहा कि □□□□□□□□□ □3 की दक्षता का एक प्रमुख कारण उसका मिश्रित विशेषज्ञों, या □□□, आर्किटेक्चर है, ठीक है? जहां प्रत्येक टोकन के लिए केवल पैरामीटरों का एक छोटा हिस्सा सक्रिय होता है। और □□□□□□□□□ □3 हमें □□□ और □□□□□□□□□ □□ के साथ एक और कदम आगे ले जाता है।

□: बिल्कुल। तो हम अभी □□□ पर ध्यान केंद्रित करते हैं।

□: ठीक है। तो वास्तविक समय के अनुप्रयोगों में, गति महत्वपूर्ण है।

□: है। और नफरत के दौरान आवश्यक की-वैल्यू कैश एक प्रमुख बॉटलनेक हो सकता है।

□: बिल्कुल। यही □□□ का काम है। ठीक है, तो पारंपरिक ध्यान रणनीति में पूर्व टोकनों के बारे में बहुत सारी जानकारी को स्टोर करने की आवश्यकता होती है।

□: हाँ, जो आप सोच सकते हैं, लंबे पाठ के अनुक्रमों के साथ यह एक समस्या बन जाता है, ठीक है?

□: लेकिन □□□ चतुरतापूर्वक इस जानकारी को संक्षिप्त करता है, ठीक है, ताकि कैश फ्लो को काफी कम कर सके और नफरत को बहुत तेज कर सके। तो यह एक बुल्की एन्साइक्लोपीडिया को केवल मुख्य बिंदुओं तक संक्षिप्त करने जैसा है।

□: यह एक अच्छा तुलना है। यह आवश्यक जानकारी को बिना अनावश्यक भार के रखता है। हाँ, तो यह वास्तविक समय के अनुप्रयोगों के लिए बहुत उपयोगी है।

□: हाँ। अब हम इसे कैसे काम करता है, इसके बारे में बात करें। ठीक है, तो □□□ यह संक्षिप्तन कैसे प्राप्त करता है?

□: तो यह ध्यान की-वैल्यू के लिए एक लो-रैंक संयुक्त संक्षिप्तन का उपयोग करता है।

□: ठीक है, तो यह की-वैल्यू को संक्षिप्त करता है, लेकिन यह बिल्कुल क्या मतलब है? तो थोड़ा तकनीकी हो जाएँ। ठीक है, □□□ रणनीति एक इनपुट छिपे प्रतिनिधित्व को लेती है, जो फिर क्वेरी, की, और वैल्यू वेक्टर में प्रोजेक्ट होती है। ठीक, अब यहाँ यह रोमांचक होता है। □□□ क्वेरी को दो हिस्सों में विभाजित करता है।

□: दो हिस्सों में?

□: हाँ। एक हिस्सा सामग्री के लिए उपयोग किया जाता है, और दूसरा हिस्सा रोप का उपयोग करके पोजिशनल जानकारी के लिए उपयोग किया जाता है।

□: रोप? यह बहुत तकनीकी लगता है।

□: यह रोटरी पोजिशन एम्बेडिंग का संक्षिप्त रूप है, और यह मॉडल को टोकनों के अनुक्रम में टोकनों की स्थिति समझने में मदद करता है। ठीक, फिर की और वैल्यू को एक कम आयामी छिपे स्थान में संक्षिप्त किया जाता है। तो यह डेटा को छोटा कर रहा है, जो मेमोरी में बचत करता है।

□: बिल्कुल। तो सबसे महत्वपूर्ण जानकारी को बचाया जाता है, लेकिन अनावश्यक भार को फेंक दिया जाता है। हाँ, और यह संक्षिप्त प्रतिनिधित्व एक छोटे □□ कैश के लिए अनुमति देता है, जिससे चीजें तेज हो जाती हैं।

□: और यह बहु-हेड प्रोसेसिंग का उपयोग करता है।

□: हाँ, जैसा कि पारंपरिक ध्यान, □□□ भी बहु-हेड का उपयोग करता है।

□: ओह, आगे बढ़ो।

□: तो इसलिए, दो छिपे स्थान और एक छिपी इनपुट।

□: यह एक अच्छा अवलोकन है। हाँ, आप बिल्कुल सही हैं। वास्तव में दो छिपे स्थान हैं। ठीक है, तो हम सामग्री छिपे स्थान और की-वैल्यू छिपे स्थान के बारे में बात कर रहे हैं।

□: बिल्कुल। और इन छिपे स्थानों को रोप, या रोटरी पोजिशन एम्बेडिंग के माध्यम से प्रोसेस किया जाता है।

□: ठीक है, तो रोप यही है कि वे पोजिशनल जानकारी प्राप्त करते हैं।

□: हाँ, यह दोनों सामग्री और की-वैल्यू छिपे स्थानों पर लागू किया जाता है, जैसा कि आपने बताया। तो यह संक्षिप्त प्रतिनिधित्व को प्रोसेस करता है, फिर इसे फिर से एक साथ मिलाता है।

□: हाँ, और कैश ऑप्टिमाइजेशन अनुक्रमिक प्रोसेसिंग के दौरान ओवरहेड को और कम करता है। ठीक है, तो यही है कि □□□ चीजों को तेज करता है।

□: बिल्कुल। यह दक्ष ध्यान प्राप्त करने का एक चतुर तरीका है बिना प्रदर्शन को कम करने के।

□: ठीक है, यह एक बहुत ही चतुर ट्रिक है। लेकिन आप जानते हैं क्या?

□: क्या?

□: अब हम □□□□□□□□ □□ पर चलते हैं। यह पारंपरिक □□□ मॉडलों से कैसे अलग है?

□: ठीक है, □□□□□□□□ □□...ओह, वापस हमारे श्रोता को, क्या?

□: और हम और छिपे स्थान के बारे में बात करते हैं। ठीक, छिपे स्थान से, क्या है?

□: बिल्कुल...तो देखते हैं आप क्या कह रहे हैं। छिपे स्थान वास्तव में बहुत दिलचस्प हैं। हाँ, आप छिपे स्थान, छिपे स्थान के बारे में पूछ रहे हैं, जिसे हम अभी बात कर रहे थे, ठीक है? आप उस गुफा के बारे में जानना चाहते हैं, जो वहां हो रहा है।

□: यह कूल है।

□: बिल्कुल। □□□ में वास्तव में दो अलग-अलग छिपे स्थान हैं, एक सामग्री और एक की-वैल्यू के लिए। यह जैसे दो अलग-अलग स्टोरेज यूनिट के लिए जानकारी रखने जैसा है। और इन छिपे स्थानों, जैसा कि हमने चर्चा की, रोप ऑपरेशन से गुजरते हैं, ठीक है? रोटरी पोजिशन एम्बेडिंग, जो पोजिशनल जानकारी को ध्यान रणनीति में एम्बेड करता है। यह बहुत महत्वपूर्ण है। तो फिर से, क्वेरी को विभाजित किया जाता है, और की और वैल्यू भी संक्षिप्त किए जाते हैं।

□: हाँ, और ये दो अलग-अलग छिपे स्थानों में रखे जाते हैं, एक सामग्री और एक की-वैल्यू जोड़ों के लिए। और ये छिपे स्थान □□□ के हिस्से के रूप में दक्षता और सब कुछ के लिए बहुत महत्वपूर्ण हैं।

□: बिल्कुल। अब हम इन ऑपरेशनों के बारे में थोड़ा और विस्तार से बात करें। ठीक है, तो □□□ इन छिपे स्थान परिवर्तनों को कैसे करता है?

□: तो इनपुट को सामग्री और की-वैल्यू प्रतिनिधित्वों के लिए समानांतर प्रोसेसिंग के लिए गुजरता है। ठीक है, तो यह जैसे दो पथों के साथ गुफा में है।

□: हाँ, प्रत्येक छिपे स्थान के लिए। और उन स्थानों में, जानकारी को रोप के साथ प्रोसेस किया जाता है।

□: बिल्कुल। यह सुनिश्चित करता है कि मॉडल पोजिशनल जानकारी को रखता है, जैसे कि वे गुफा के अंदर हैं। तो मॉडल जानता है कि पाठ का कौन सा हिस्सा कौन सा है, जैसे कि यह गुफा के अंदर है।

□: बिल्कुल। और यह प्रोसेसिंग अगले चरण के संयोजन से पहले किया जाता है। ठीक है, तो गुफा के छिपे स्थान के माध्यम से क्या संयोजित किया जाता है?

□: रणनीति दो प्रमुख संयोजन ऑपरेशन करता है। क्वेरी प्रतिनिधित्व संयोजित किए जाते हैं, और की प्रतिनिधित्व भी संयोजित किए जाते हैं। तो यह जैसे कि गुफा के छिपे स्थान के अंदर सभी महत्वपूर्ण टुकड़ों को एक साथ लाना।

□: हाँ, और ये संयोजन सामग्री को पोजिशनल जानकारी के साथ मिलाते हैं। और ये संयोजित प्रतिनिधित्व फिर ध्यान गणना के लिए उपयोग किए जाते हैं, ठीक है?

□: बिल्कुल। और शुरुआती संक्षिप्तन के कारण, यह गुफा के अंदर और बाहर बहुत तेज हो जाता है। तो □□□ बड़े मॉडलों जैसे □□□□□□□□ □3 के लिए ध्यान रणनीति को ऑप्टिमाइज करता है। यह एक बहुत अच्छा सवाल है। अब, जब हमने गुफा से गुजार लिया, तो अब हम □□□□□□□□ □□ पर चलते हैं।

□: ठीक है, □□□□□□□□ □□। बिल्कुल, मैं समझ गया। हाँ, □□□ में वास्तव में दो अलग-अलग छिपे स्थान हैं, एक सामग्री और एक की-वैल्यू के लिए।

□: बिल्कुल। और यह अलगाव वास्तव में इसके काम करने का एक महत्वपूर्ण हिस्सा है। यह जैसे दो अलग-अलग स्टोरेज यूनिट के लिए जानकारी रखने जैसा है। और इन छिपे स्थानों, जैसा कि हमने चर्चा की, रोप ऑपरेशन से गुजरते हैं, ठीक है? रोटरी पोजिशन एम्बेडिंग, जो पोजिशनल जानकारी को ध्यान रणनीति में एम्बेड करता है। तो फिर से, क्वेरी को विभाजित किया जाता है, और की और वैल्यू भी संक्षिप्त किए जाते हैं।

□: हाँ, और ये दो अलग-अलग छिपे स्थानों में रखे जाते हैं, एक सामग्री और एक की-वैल्यू जोड़ों के लिए। और ये छिपे स्थान □□□ के हिस्से के रूप में दक्षता और सब कुछ के लिए बहुत महत्वपूर्ण हैं।

□: बिल्कुल। अब हम इन ऑपरेशनों के बारे में थोड़ा और विस्तार से बात करें। ठीक है, तो □□□ इन छिपे स्थान परिवर्तनों को कैसे करता है?

□: तो इनपुट को सामग्री और की-वैल्यू प्रतिनिधित्वों के लिए समानांतर प्रोसेसिंग के लिए गुजरता है। ठीक है, तो यह जैसे दो पथों के साथ गुफा में है।

□: हाँ, प्रत्येक छिपे स्थान के लिए। और उन स्थानों में, जानकारी को रोप के साथ प्रोसेस किया जाता है।

□: बिल्कुल। यह सुनिश्चित करता है कि मॉडल पोजिशनल जानकारी को रखता है, ठीक है? और दक्षता को बढ़ाने के लिए, यह साझा विशेषज्ञों का उपयोग करता है। ठीक है, तो विशेषज्ञ जो कई कार्यों के लिए उपयोग किए जा सकते हैं।

□: हाँ, तो यह पुनरावृत्ति को रोकता है और प्रणाली को और अधिक साफ करता है।

□: हाँ, यह जैसे एक टीम है जहां लोगों के पास विशेषताएं होती हैं, लेकिन वे अन्य चीजें भी कर सकते हैं।

□: हाँ, यह एक बहुत ही चतुर तरीका है। हाँ, लेकिन इतनी विशेषज्ञों के साथ, वे कैसे सुनिश्चित करते हैं कि कोई भी ओवरलोड नहीं हो जाए?

□: हाँ, जबकि अन्य लोग खाली बैठे हैं।

□: यही उनकी नवीनिकरक ऑक्सिलरी लॉस-फ्री लोड बैलेंसिंग का काम है।

□: यह बहुत रोमांचक हो जाता है, ठीक है? तो वे कैसे करते हैं?

□: पारंपरिक □□□ मॉडल ट्रेनिंग के दौरान एक ऑक्सिलरी लॉस फंक्शन का उपयोग करते हैं, ठीक है, ताकि समान विशेषज्ञ उपयोग को प्रोत्साहित किया जा सके, लेकिन यह वास्तव में प्रदर्शन को कम कर सकता है।

□: हाँ, यह जैसे कि सबको एक ही चेकआउट लाइन का उपयोग करने के लिए मजबूर करना है।

□: बिल्कुल, चाहे कुछ तेज चल रहे हों, ठीक है? तो यह अनावश्यक देरी पैदा करता है।

□: हाँ। तो □□□□□□□□ □3 इसको टालता है, एक बायस टर्म को प्रत्येक विशेषज्ञ के लिए डायनामिक रूप से समायोजित करके, ठीक है, उसके लोड के आधार पर। ठीक है, तो अगर एक विशेषज्ञ बहुत सारे अनुरोध प्राप्त कर रहा है, तो प्रणाली इसे रूटिंग रणनीति के लिए थोड़ा कम आकर्षक बना देती है, जिससे कुछ ट्रैफिक कम बिजी विशेषज्ञों की ओर मोड़ दिया जाता है।

□: ठीक है, तो यह लंबे अनुक्रमों को कुशलतापूर्वक संभालने के लिए, नफरत के दौरान आवश्यक □□ कैश की आकार को कम करने के लिए, सभी को उपयोग करता है, ठीक है? तो यह प्रदर्शन को ऊंचा रखते हुए ओवरहेड को कम करने के बारे में है।

□: बिल्कुल। यह एक बहुत ही चतुर तरीका है एक महत्वपूर्ण बॉटलनेक को संबोधित करने का।

□: बिल्कुल। अब हम भी □□□□□□□□ □3 को कैसे लोड बैलेंसिंग संभालता है, इसके बारे में बात करनी चाहिए।

□: हाँ, हम निश्चित रूप से करेंगे। यह भी एक बहुत ही महत्वपूर्ण पजल का हिस्सा है। हम अगले चरण में इसे छू सकते हैं।

□: सुनता है। तो मैं सोचता हूँ कि यह आपको □□□ और उसके छिपे स्थान के बारे में एक अच्छा अवलोकन देता है।

□: हाँ, सभी विवरणों के साथ बात करने के लिए धन्यवाद। हम अगली बार और गहरे उतरेंगे।

□: बिल्कुल, यह जैसे एक ट्रैफिक प्रबंधन प्रणाली है, ठीक है, विशेषज्ञों के लिए, हमेशा फ्लो को निगरानी करते हुए और बॉटलनेक्स को टालने के लिए समायोजन करते हुए।

□: और यह ऑक्सिलरी लॉस के प्रदर्शन हिट को टालता है।

□: बिल्कुल। और ओह, आगे बढ़ो।

□: हाँ, हम □□□ के बारे में बात कर सकते हैं, कैसे...कैसे □□□ मॉड्यूल अपने एम्बेडिंग और सब कुछ को साझा करते हैं।

□: बिल्कुल। यह एक अच्छा सवाल है। हाँ, तो हम □□□ मॉड्यूलों के सेटअप और उनके संसाधनों को साझा करने के बारे में बात कर सकते हैं। ठीक है, तो आप □□□ मॉड्यूलों के बारे में जानना चाहते हैं, और कैसे वे अपने संसाधनों को साझा करते हैं। ठीक है, तो प्रत्येक □□□ मॉड्यूल में एक साझा एम्बेडिंग लेयर, ठीक है, और एक साझा आउटपुट हेड शामिल है। ठीक है, तो वे मुख्य मॉडल से उसी एम्बेडिंग और आउटपुट हेड का उपयोग करते हैं।

□: बिल्कुल। तो यह जैसे कि वे सभी एक ही ज्ञान के पूल से ले रहे हैं। हाँ, और यह गणनात्मक लागत को बचाता है।

□: हाँ। अब यह अपने ट्रांसफॉर्मर ब्लॉक का उपयोग करता है। ठीक है, तो यह मुख्य मॉडल के साथ वही ट्रांसफॉर्मर ब्लॉक नहीं साझा करता।

□: बिल्कुल। प्रत्येक □□□ मॉड्यूल अपने ट्रांसफॉर्मर ब्लॉक के लिए प्रोसेसिंग के लिए अपना ट्रांसफॉर्मर ब्लॉक रखता है। ठीक है, तो यही है कि वे प्रत्येक टोकन के लिए अलग-अलग प्रेक्षणों को बनाए रखते हैं।

□: हाँ, और जानकारी को संयोजित करने के लिए, ये लिनियर प्रोजेक्शंस और संयोजन...

□: ठीक है, तो यह जैसे कि कई जगहों से टुकड़े लेना है ताकि पूरा चित्र बनाया जा सके।

□: हाँ, और सभी □□□ मॉड्यूल एक साथ काम करते हैं, लेकिन वे अपने एम्बेडिंग लेयर और आउटपुट हेड साझा करते हैं, ठीक है?

□: हाँ, जो इस डिजाइन की दक्षता का एक महत्वपूर्ण हिस्सा है। ठीक है, तो यह जैसे कि एक प्रणाली है जिसमें सभी हिस्से एक दूसरे पर निर्भर हैं, ठीक है?

□: और यह संसाधनों का दक्ष साझा करने से तेज ट्रेनिंग और बेहतर प्रदर्शन की अनुमति देता है।

□: ठीक है, यह एक बहुत ही चतुर ट्रिक है। आप जानते हैं क्या?

□: क्या?

□: अब हम एक बड़े चित्र के दृष्टिकोण पर चलते हैं। यह मॉडल लोड बैलेंसिंग कैसे संभालता है? और वे कैसे चुने जाते हैं?

□: हाँ, हम निश्चित रूप से इसके बारे में बात कर सकते हैं। ठीक है, अब हम □□□□□□□□ □3 की लोड बैलेंसिंग रणनीति में गहरे उतरते हैं।

□: सुनता है। ठीक है, तो □□□□□□□□ □3 को वे कहते हैं, बहु-टोकन प्रेक्षण, या □□□। हम अभी □□□ के बारे में बात कर रहे थे, तो अब हम लोड बैलेंसिंग के बारे में बात करें, ठीक है?

□: हाँ, हम अभी इसके बारे में बात कर रहे थे। अब यह संसाधनों को साझा करता है, और आप जानना चाहते हैं कि यह संसाधनों को कैसे साझा करता है। हम इसके बारे में बात कर चुके हैं।

□: बिल्कुल। तो केवल अगले टोकन को प्रेक्षण करने के बजाय, ठीक है, यह कई भविष्य के टोकनों को एक साथ प्रेक्षण करता है, जैसा कि हम अभी बात कर रहे थे। तो यह जटिलता को बढ़ाता है?

□: यह ऐसा लगता है, लेकिन यह कई फायदे देता है। ठीक है, एक रूट को योजना बनाएं। अगर आप केवल अगले मोड़ को देखते हैं, तो आप एक अधिक कुशल...ठीक है, आगे देखना और कई मोड़ों को योजना बनाना आपको सबसे अच्छा रूट चुनने में मदद करता है।

□: हाँ। □□□□□□□□ □3 एक नवीनिकरक रणनीति का उपयोग करता है जिसे ऑक्सिलरी लॉस-फ्री लोड बैलेंसिंग कहा जाता है, तो यह ट्रेनिंग के दौरान एक अलग लॉस फंक्शन के लिए निर्भर नहीं करता।

□: बिल्कुल। पारंपरिक □□□ मॉडल ट्रेनिंग के दौरान एक ऑक्सिलरी लॉस फंक्शन का उपयोग करते हैं, ताकि समान विशेषज्ञ उपयोग को प्रोत्साहित किया जा सके, ठीक है? लेकिन हमने पहले बताया कि यह वास्तव में प्रदर्शन को कम कर सकता है।

□: हाँ, यह जैसे कि सबको एक ही चेकआउट लाइन का उपयोग करने के लिए मजबूर करना।

□: ठीक है, तो कई टोकनों को प्रेक्षण करके, मॉडल को संदर्भ को बेहतर पकड़ने में मदद मिलती है।

□: हाँ, और यह अधिक सुसंगत और सटीक जवाबों को जनरेट कर सकता है। यह जैसे कि मॉडल अपने प्रतिनिधित्वों को भविष्य के लिए योजना बनाता है, जैसा कि मैंने पहले बताया था, बेहतर भविष्य के प्रेक्षण के लिए। ठीक है, और यह एक साफ ट्रेनिंग सिग्नल और बेहतर डेटा दक्षता में परिणाम देता है।

□: हाँ, तो इसके बजाय, □□□□□□□□ □3 प्रत्येक विशेषज्ञ के लिए एक बायस टर्म को डायनामिक रूप से समायोजित करता है, ठीक है, उसके लोड के आधार पर। ठीक है, तो अगर एक विशेषज्ञ बहुत सारे अनुरोध प्राप्त कर रहा है, तो प्रणाली इसे रूटिंग रणनीति के लिए थोड़ा कम आकर्षक बना देती है, जिससे कुछ ट्रैफिक कम बिजी विशेषज्ञों की ओर मोड़ दिया जाता है।

□: हाँ, जैसे कि एक ट्रैफिक प्रबंधन प्रणाली, विशेषज्ञों के लिए, हमेशा फ्लो को निगरानी करते हुए और बॉटलनेक्स को टालने के लिए समायोजन करते हुए। तो □□□ और क्या कर सकता है?

□: □□□ मॉड्यूल ट्रेनिंग के दौरान उपयोग किए जाते हैं, तो वे सामान्य नफरत के दौरान फेंक दिए जा सकते हैं, ठीक है, या चतुरतापूर्वक कुछ के लिए पुनः उपयोग किए जा सकते हैं, जिसे विशेष डिकोडिंग कहा जाता है।

□: विशेष डिकोडिंग। क्या है?

□: तो नफरत के बजाय, मॉडल भी अगले टोकन के साथ-साथ संभव विकल्पों को भी प्रेक्षण करता है।

□: ओह वाह, तो यह पाठ को तेज बनाने में मदद करता है क्योंकि यह पहले से ही कई विकल्पों को सोच रहा है, एक बैकअप योजना तैयार रखने के लिए।

□: हाँ, तो मॉडल को हर बार रोकना और फिर से गणना करने की आवश्यकता नहीं होती।

□: ठीक है, यह समझ में आता है। हाँ, अब दक्षता के बारे में बात करते हैं, और बॉटलनेक्स को टालने के लिए, और यह ऑक्सिलरी लॉस के प्रदर्शन हिट को टालता है।

□: बिल्कुल। और वे एक सहयोगी अनुक्रम-वाइज बैलेंस लॉस भी शामिल करते हैं, ठीक है, ताकि व्यक्तिगत...प्रक्रियाओं के भीतर गंभीर असंतुलन को रोक सके।

□: ...प्रक्रियाओं में। और प्रत्येक टोकन को चार नोड्स तक सीमित करके, वे नेटवर्क संचार को भी कम करते हैं। ठीक है, तो यह चीजों को भी साफ करता है।

□: ठीक है, अब हम □□□□□□□□ □3 को कैसे इसकी गणनात्मक मांगों को संभालता है, इसके बारे में बात करें। और मैं जानता हूँ कि आप विशेष रूप से लागत-ऑप्टिमाइजेशन और कैसे वे आर्थिक रूप से चीजें कर रहे हैं, के बारे में जानना चाहते हैं।

□: हाँ, और यह मॉडल कुछ अद्भुत चीजें उस क्षेत्र में करता है।

□: बिल्कुल। हाँ, औसतन 3.2 विशेषज्ञ प्रति टोकन चुने जाते हैं, जो ओवरहेड को कम करने के लिए एक अच्छा संतुलन है।

□: बिल्कुल। तो यह एक बहुत ही दक्ष और प्रभावी तरीका है।

□: हाँ, यह एक बहुत ही चतुर तरीका है, एक इस तरह के जटिल मॉडल को इतना अच्छा काम करने के लिए।

□: बिल्कुल। और वे इस तरीके से विशेषज्ञ विशेषता प्राप्त करते हैं। ठीक है, तो इसका मतलब है कि अलग-अलग डोमेन में अलग-अलग विशेषज्ञ सक्रिय होते हैं। तो वे क्या हैं?

□: □□□□□□□□ □3 एक □□□ मिश्रित-प्रिसिजन ट्रेनिंग फ्रेमवर्क का उपयोग करता है। ठीक है, एक मॉडल इस पैमाने के लिए एक महत्वपूर्ण प्रगति। मुझे फिर से □□□ क्या है, याद दिलाएं?

□: बिल्कुल, यह 8-बिट फ्लोटिंग पॉइंट है।

□: ठीक है, और यह कम बिट्स का उपयोग करके संख्याओं को प्रतिनिधित्व करता है, जैसा कि पारंपरिक प्रारूप। ठीक है, तो यह कम मेमोरी और तेज गणना में परिणाम देता है।

□: बिल्कुल। यह जैसे कि एक बड़ी छवि फाइल को संक्षिप्त करना, लेकिन आप अभी भी छवि के मूल का पालन करते हैं। यह बस कम जगह लेता है, ठीक है?

□: बिल्कुल। तो प्रत्येक विशेषज्ञ सिर्फ सामान्य रूप से सक्रिय नहीं होता, बल्कि विशेष डोमेन में सक्रिय होता है। तो यह फाइनली ट्यून किया जाता है और कार्य के लिए तैयार है।

□: बिल्कुल। अब यह बैच-वाइज रणनीति बहुत चतुर है।

□: बिल्कुल, मैं सहमत हूँ। यह लोड बैलेंसिंग की इस डायनामिक रणनीति में बहुत रोमांचक है। यह सब दक्षता और संतुलन के बारे में है।

□: बिल्कुल, यह सब □□□□□□□□ □3 की प्रदर्शन और संसाधन उपयोग के प्रति प्रतिबद्धता का हिस्सा है।

□: बिल्कुल। तो आज हमने बहुत कुछ कवर किया। यह बहुत रोमांचक है, लेकिन कम बिट्स का उपयोग करना तो □□□□□□□□ को प्रभावित कर सकता है?

□: यह एक वैध चिंता है, और यह कुछ तकनीकों को लागू करके सावधानी से संबोधित किया गया है, जिसमें फाइन-ग्रेन क्वांटाइजेशन शामिल है।

□: हाँ, यह □□□ में संख्याओं को कैसे प्रतिनिधित्व किया जाता है, इसके बारे में बहुत सटीक नियंत्रण देता है। हाँ, बहु-हेड लेटेंट एटेंशन से □□□□□□□□ □□ और लोड बैलेंसिंग तक, यह □□□□□□□□ □3 मॉडल एक बहुत ही जटिल प्रणाली है, और यह एक उदाहरण है कि कैसे नवीनिकरण हमारे...के सीमाओं को बढ़ा रहा है।

□: बिल्कुल, आज की गहरी डाइव में एक अच्छा समय बिताया।

□: बिल्कुल, मैं सोचता हूँ कि यह आपको □□□□□□□□ □3 के बारे में एक ठोस अवलोकन देता है।

□: बिल्कुल। हमें इसके साथ खोजने के लिए धन्यवाद।

□: बिल्कुल, धन्यवाद। और यह आज की गहरी डाइव के लिए है। तो हम जल्द ही एक और के साथ वापस आएंगे।

□: तो वे आपसे संतुलन को बनाए रखते हैं।