# □□□□□□□□ □3: मल्टी-हेड लेटेंट अटेंशन और मल्टी-टोकन प्रेडिक्शन

इस पोस्ट में, मैं □□□□□□□□ □3 पर चर्चा करूंगा, जिसमें "□□□□□-□□□□ □□□□□□ □□□□□□□□□ □□□ □□□□□-□□□□□ □□□□□□□□□□ □□ □□□□□□□□ □3"वीडियो का संदर्भ दिया गया है □□□□□://□□□□□.□□/□□49□□□□□□□?□□=4□□2□□□-□□□□1□□□। मैंने वीडियो को ट्रांसक्राइब करने के लिए □□□□□□ □□□□□ □□□□□□-□□-□□□□ का उपयोग किया और ट्रांसक्रिप्ट को व्यवस्थित करने में मदद करने के लिए कुछ कोड का उपयोग किया।

---

□: □□□□ टैग में वापस स्वागत है। आज हम बड़े भाषा मॉडल्स की दुनिया में गहराई से जाएंगे। ठीक है, विशेष रूप से □□□□□□□□ □3।

□: अच्छा लगा। यह एक 671 बिलियन पैरामीटर वाला मॉडल है, जो अपनी दक्षता और प्रदर्शन के लिए चर्चा में है, है ना?

□: और आपने इसकी आर्किटेक्चर को विस्तार से समझाने वाला एक अकादमिक पेपर साझा किया।

□: हाँ।

□: और एक मशीन लर्निंग विशेषज्ञ के रूप में, आप यह समझना चाहते हैं कि □□□□□□□□ □3 कैसे उच्च प्रदर्शन और किफायती ट्रेनिंग दोनों को प्राप्त करता है।

□: हाँ, यह सही है।

□: ओह, हे, क्या हो रहा है?

□: □□□, विवरण, □□□ और यह कैसे काम करता है।

□: ओह, बिल्कुल। यह एक बढ़िया विचार है। हाँ, हम निश्चित रूप से मल्टी-हेड लेटेंट अटेंशन, या □□□, में गहराई से जा सकते हैं। तो आप □□□ के बारे में जानने के लिए उत्सुक हैं। चलिए इसे समझते हैं। हमने बताया कि □□□□□□□□ □3 की दक्षता की कुंजी इसके मिश्रण विशेषज्ञ, या □□□, आर्किटेक्चर में है, है ना? जहां प्रत्येक टोकन के लिए केवल एक अंश पैरामीटर सक्रिय होते हैं। और □□□□□□□□ □3 हमें □□□ और □□□□□□□□ □□ के साथ एक कदम आगे ले जाता है।

□: यह सही है। तो चलिए अभी □□□ पर ध्यान केंद्रित करते हैं।

□: ठीक है। तो रियल-टाइम एप्लिकेशन्स में, गति महत्वपूर्ण है।

□: हाँ। और इनफेरेंस के दौरान आवश्यक की-वैल्यू कैश एक प्रमुख बाधा हो सकती है।

□: बिल्कुल। यहीं पर □□□ काम आता है। ठीक है, तो पारंपरिक अटेंशन मैकेनिज्म को पिछले टोकन्स के बारे में बहुत सारी जानकारी स्टोर करने की आवश्यकता होती है।

□: हाँ, जैसा कि आप सोच सकते हैं, लंबे टेक्स्ट सीक्वेंस के साथ यह एक समस्या बन जाती है, है ना?

□: लेकिन □□□ इस जानकारी को चतुराई से संपीड़ित करता है, ठीक है, कैश फ्लो को काफी कम करने के लिए और इनफेरेंस को बहुत तेज बनाता है। तो यह एक भारी एनसाइक्लोपीडिया को लेकर उसे केवल मुख्य बिंदुओं तक संक्षिप्त करने जैसा है।

□: यह एक बढ़िया उदाहरण है। यह आवश्यक जानकारी को बिना अनावश्यक भार के बनाए रखता है। हाँ, तो यह रियल-टाइम एप्लिकेशन्स के लिए वास्तव में उपयोगी है।

□: हाँ। अब चलिए बात करते हैं कि यह वास्तव में कैसे काम करता है। ठीक है, तो □□□ यह संपीड़न कैसे प्राप्त करता है?

□: खैर, यह अटेंशन कीज़ और वैल्यूज़ के लिए लो-रैंक जॉइंट कंप्रेशन का उपयोग करता है।

□: ठीक है, तो यह कीज़ और वैल्यूज़ को संपीड़ित कर रहा है, लेकिन इसका वास्तव में क्या मतलब है? तो चलिए थोड़ा तकनीकी हो जाएं। ठीक है, □□□ मैकेनिज्म एक इनपुट हिडन रिप्रेजेंटेशन लेता है, जिसे फिर क्वेरी, की, और वैल्यू वेक्टर्स में प्रोजेक्ट किया जाता है। ठीक है, अब यहां यह दिलचस्प हो जाता है। □□□ क्वेरी को दो भागों में विभाजित करता है।

□: ठीक है, दो भाग?

□: हाँ। एक भाग कंटेंट के लिए उपयोग किया जाता है, और दूसरा भाग पोजिशनल इंफॉर्मेशन के लिए □□□□ नामक कुछ का उपयोग करता है।

□: □□□□? यह बहुत टेक्निकल लगता है।

□: यह रोटरी पोजिशन एम्बेडिंग्स के लिए खड़ा है, और यह मॉडल को सीक्वेंस में टोकन्स की स्थिति को समझने में मदद करता है। ठीक है, फिर कीज़ और वैल्यूज़ को एक लोअर डायमेंशनल लेटेंट स्पेस में संपीड़ित किया जाता है। तो यह डेटा को सिकोड़ने जैसा है, जो मेमोरी बचाता है।

□: बिल्कुल। तो सबसे महत्वपूर्ण जानकारी सहेजी जाती है, लेकिन अनावश्यक भार को हटा दिया जाता है। हाँ, और यह संपीड़ित रिप्रेजेंटेशन इनफेरेंस के दौरान एक बहुत छोटे □□ कैश की अनुमति देता है, जो चीजों को तेज करता है।

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

□: हाँ, पारंपरिक अटेंशन की तरह, □□□ कई हेड्स का उपयोग करता है।

□: ओह, आगे बढ़ो।

□: तो इस तरह, दो लेटेंट स्पेस और एक हिडन इनपुट होते हैं।

□: यह एक बढ़िया अवलोकन है। हाँ, आप सही हैं। वास्तव में दो लेटेंट स्पेस होते हैं। ठीक है, तो हम एक कंटेंट लेटेंट स्पेस और एक की-वैल्यू लेटेंट स्पेस की बात कर रहे हैं।

□: बिल्कुल। और इन लेटेंट स्पेस को □□□□, या रोटरी पोजिशन एम्बेडिंग्स के माध्यम से प्रोसेस किया जाता है।

□: ठीक है, तो यह □□□□ है जो उन्हें पोजिशनल इंफॉर्मेशन प्राप्त करने में मदद करता है।

□: हाँ, यह कंटेंट और की-वैल्यू लेटेंट स्पेस दोनों पर लागू होता है, जैसा कि आपने बताया। तो यह इस संपीड़ित रिप्रेजेंटेशन को लेता है, इसे प्रोसेस करता है, और फिर इसे वापस एक साथ जोड़ता है।

□: हाँ, और कैशिंग ऑप्टिमाइजेशन सीक्वेंशियल प्रोसेसिंग के दौरान ओवरहेड को और कम करता है। ठीक है, तो यह है कि □□□ चीजों को कैसे तेज करता है।

□: बिल्कुल। यह प्रदर्शन को त्यागे बिना दक्षता प्राप्त करने का एक चतुर तरीका है।

□: ठीक है, यह एक बहुत ही साफ-सुथरी चाल है। लेकिन आप जानते हैं क्या?

□: क्या है?

□: चलिए □□□□□□□□ □□ पर चलते हैं। यह पारंपरिक □□□ मॉडल्स से कैसे अलग है?

□: ठीक है, □□□□□□□□ □□ का उपयोग करता है...ओह, हमारे श्रोता, क्या हो रहा है?

□: और हम और अधिक हिडन स्पेस के बारे में बात करते हैं। ठीक है, हिडन स्पेस से, वह क्या है?

□: मैं बिल्कुल...देखते हैं कि आप क्या कह रहे हैं। हिडन स्पेस वास्तव में दिलचस्प हैं। हाँ, आप हिडन स्पेस, लेटेंट स्पेस के बारे में पूछ रहे हैं जिसके बारे में हम अभी बात कर रहे थे, है ना? आप उत्सुक हैं कि उन लेटेंट स्पेस के भीतर क्या हो रहा है, वह गुफा। हाँ, यह सिर्फ लेटेंट स्पेस की संख्या के बारे में नहीं है, बल्कि वहां क्या हो रहा है।

□: यह अच्छा है।

□: बिल्कुल। □□□ के भीतर वास्तव में दो अलग-अलग लेटेंट स्पेस होते हैं, एक कंटेंट के लिए और एक की वैल्यूज़ के लिए। यह जानकारी के लिए दो अलग-अलग स्टोरेज यूनिट्स होने जैसा है। और ये लेटेंट स्पेस, जैसा कि हमने चर्चा की है, □□□□ ऑपरेशन्स से गुजरते हैं, है ना? रोटरी पोजिशनल एम्बेडिंग, जो अटेंशन

मैकेनिज्म में पोजिशनल इंफॉर्मेशन को एम्बेड करता है। यह उनके लिए बहुत महत्वपूर्ण है। तो संक्षेप में, क्वेरी को विभाजित किया जाता है, और कीज़ और वैल्यूज़ को भी संपीड़ित किया जाता है।

: हाँ, और इन्हें दो अलग-अलग लेटेंट स्पेस में रखा जाता है, एक कंटेंट के लिए और एक की-वैल्यू पेयर्स के लिए। और ये लेटेंट स्पेस □□□ के हिस्से के रूप में दक्षता और सभी के लिए वास्तव में महत्वपूर्ण हैं।

: बिल्कुल। अब चलिए इस गुफा के भीतर इन ऑपरेशन्स के बारे में थोड़ा और विस्तार से बात करते हैं। ठीक है, तो □□□ वास्तव में इन लेटेंट स्पेस ट्रांसफॉर्मेशन को कैसे करता है?

: खैर, इनपुट कंटेंट और की-वैल्यू रिप्रेजेंटेशन दोनों के लिए समानांतर प्रोसेसिंग से गुजरता है। ठीक है, तो यह उस गुफा के भीतर दो पथों की तरह है।

: हाँ, प्रत्येक लेटेंट स्पेस के लिए एक। और इन स्पेस के भीतर, जानकारी को □□□□ का उपयोग करके प्रोसेस किया जाता है।

: यह सही है। यह सुनिश्चित करता है कि मॉडल पोजिशनल इंफॉर्मेशन को बनाए रखता है क्योंकि वे उस गुफा के भीतर जाते हैं। तो मॉडल जानता है कि टेक्स्ट का कौन सा हिस्सा कौन सा है जब वह उस गुफा के भीतर होता है।

: बिल्कुल। और यह प्रोसेसिंग अगले स्टेज ऑफ कंकैटनेशन से पहले की जाती है। ठीक है, हिडन स्पेस गुफा के माध्यम से जाने पर क्या कंकैटनेट किया जा रहा है?

: मैकेनिज्म दो प्रमुख कंकैटनेशन ऑपरेशन करता है। क्वेरी रिप्रेजेंटेशन को कंकैटनेट किया जाता है, और की रिप्रेजेंटेशन को भी कंकैटनेट किया जाता है। तो यह उस हिडन स्पेस गुफा के भीतर सभी महत्वपूर्ण टुकड़ों को एक साथ लाने जैसा है।

: हाँ, और ये कंकैटनेशन कंटेंट को पोजिशनल इंफॉर्मेशन के साथ जोड़ने में मदद करते हैं। और फिर इन कंकैटनेटेड रिप्रेजेंटेशन का उपयोग अटेंशन कैलकुलेशन के लिए किया जाता है, है ना?

: सही है। और प्रारंभिक संपीड़न के कारण, यह उस गुफा के माध्यम से बहुत तेज होता है जिसका आपने उल्लेख किया है। तो □□□ उस हिडन गुफा के अंदर और बाहर कम्प्यूटेशनल लागत को काफी कम कर देता है।

: बिल्कुल। यह □□□□□□□□ □3 जैसे बड़े मॉडल्स के लिए अटेंशन मैकेनिज्म को ऑप्टिमाइज़ करता है। यह एक बढ़िया सवाल है। अब, जब हम गुफा से गुजर चुके हैं, तो चलिए □□□□□□□□ □□ पर चलते हैं।

: