به نام خدا

# تمرین سری پنجم درس مبانی پردازش زبان و گفتار دکتر مرضیه داودآبادی فراهانی

# فرزان رحمانی 99521271

## سوال اول

پاسخگویی به سوالات دامنه بسته

تعریف: سیستم های کنترل کیفیت دامنه بسته برای پاسخ به سؤالات در یک دامنه یا حوزه موضوعی از پیش تعریف شده خاص طراحی شده اند. این سیستم ها از دانش خاص دامنه استفاده می کنند و معمولاً بر روی مجموعه داده ای آموزش می بینند که بسیار مرتبط با موضوع خاص است.

مشخصات:

* + دانش تخصصی: آنها از پایگاه های دانش تخصصی یا مجموعه داده های مربوط به حوزه خاص استفاده می کنند.
  + دقت بالاتر در دامنه: معمولاً به دلیل داده‌های آموزشی متمرکز و الگوریتم‌های متناسب، دقت بالاتری را برای سؤالات درون دامنه نشان می‌دهند.
  + محدوده محدود: سیستم ممکن است در ارائه پاسخ های دقیق به سوالات خارج از حوزه خود دچار مشکل شود یا شکست بخورد.

مثال ها:

* + یک سیستم QA پزشکی که به سوالات مربوط به بیماری ها، درمان ها و شرایط پزشکی پاسخ می دهد.
  + یک سیستم QA قانونی که به کاربران کمک می کند تا پاسخ های مربوط به قوانین، مقررات، و سوابق قانونی را بیابند.

پاسخگویی به سوالات دامنه باز

تعریف: سیستم های کنترل کیفیت دامنه باز برای پاسخ به سؤالات در مورد طیف گسترده ای از موضوعات بدون محدود شدن به یک دامنه خاص طراحی شده اند. این سیستم‌ها معمولاً از مجموعه داده‌های گسترده و متنوعی استفاده می‌کنند که اغلب دانش همه منظوره از وب، دایره‌المعارف‌ها یا سایر منابع دانش گسترده را در بر می‌گیرد.

مشخصات:

* + دانش عمومی: آنها از مجموعه داده های بزرگ مقیاس مانند ویکی پدیا، صفحات وب و سایر پایگاه های دانش عمومی استفاده می کنند.
  + دامنه وسیع: قادر به پاسخگویی به سؤالات تقریباً در هر موضوعی است، اگرچه دقت ممکن است بسته به ویژگی و پیچیدگی سؤال متفاوت باشد.
  + تطبیق پذیری: طراحی شده برای رسیدگی به طیف گسترده ای از سؤالات، آنها را برای موارد استفاده عمومی متنوع تر می کند.

مثال ها:

* + یک سیستم QA مانند جستجوی گوگل یا IBM Watson که می تواند به مجموعه وسیعی از سوالات در مورد موضوعات مختلف پاسخ دهد.

تفاوت های کلیدی

1. محدوده:

* + Closed-Domain: محدود به یک دامنه یا موضوع خاص.
  + دامنه باز: قادر به پاسخگویی به سؤالات در طیف وسیعی از موضوعات است.

2. پایگاه دانش:

* + Closed-Domain: متکی بر مجموعه داده های تخصصی و خاص دامنه است.
  + دامنه باز: از مجموعه داده های بزرگ و متنوعی که دانش عمومی را در بر می گیرد، استفاده می کند.

3. دقت:

* + Closed-Domain: به طور کلی در دامنه خاص به دلیل آموزش تخصصی دقیق تر است.
  + دامنه باز: دقت می تواند متغیر باشد. ممکن است به اندازه سیستم های دامنه بسته برای موضوعات تخصصی دقیق نباشد، اما هدف آن پوشش گسترده تر است.

4. پیچیدگی:

* + دامنه بسته: طراحی و بهینه سازی برای موارد استفاده خاص آسان تر است.
  + دامنه باز: طراحی پیچیده تر به دلیل نیاز به رسیدگی به انواع سؤالات و یکپارچه سازی حجم وسیعی از اطلاعات.

5. موارد استفاده:

* + دامنه بسته: مناسب برای برنامه هایی که به تخصص عمیق در زمینه خاصی نیاز دارند، مانند پشتیبانی تشخیص پزشکی یا سیستم های پشتیبانی فنی.
  + Open-Domain: مناسب برای برنامه های کاربردی همه منظوره که در آن کاربران ممکن است سؤالات مختلفی بپرسند، مانند دستیارهای مجازی یا موتورهای جستجوی عمومی.

درک این تفاوت ها به انتخاب نوع مناسب سیستم QA بر اساس نیازهای برنامه و ماهیت سوالات مطرح شده کمک می کند.

همچنین جواب claude هم که در ادامه آمده است خالی از لطف نیست و بیان دیگر جواب داده است.

Close-Domain Question Answering:

* These systems are designed to answer questions within a specific, limited domain or subject area.
* The knowledge base or corpus is restricted to a particular topic or set of topics.
* Examples include question answering for a specific encyclopedia, database, or technical manual.
* The language and terminology used in questions is constrained and well-defined within that domain.
* These systems can leverage domain-specific knowledge and rules to understand queries better.
* They typically achieve higher accuracy than open-domain systems for their focused topic area.

Open-Domain Question Answering:

* These systems attempt to answer questions on virtually any topic across a broad range of domains.
* The knowledge base is large and unconstrained, like the entire web or large text corpora.
* Users can ask free-form questions using natural language without domain restrictions.
* The language and terminology can be diverse and unconstrained.
* These systems rely more heavily on natural language processing and understanding techniques.
* They are generally more challenging than close-domain systems due to the breadth of topics and language variability.
* Open-domain QA is closer to how humans ask questions in the real world.

Key Differences:

1. Knowledge Base: Close-domain uses a limited, focused corpus, while open-domain has a broad, unconstrained knowledge base.
2. Language: Close-domain has constrained terminology, while open-domain handles diverse natural language.
3. Complexity: Open-domain QA is generally more complex due to the breadth of topics and language variability.
4. Accuracy: Close-domain systems can achieve higher accuracy within their focused domain compared to open-domain systems.

In summary, close-domain question answering deals with a specific subject area, while open-domain handles questions across any topic, making the latter more challenging but also more reflective of real-world question answering scenarios.

مراجع:

<https://chat.openai.com/>   
[https://gemini.google.com/](https://bard.google.com/)   
<https://claude.ai/chats>

## سوال دوم

درک ماشینی خواندن (MRC) وظیفه ای در پردازش زبان طبیعی است که شامل آموزش مدل هایی برای خواندن و درک داده های متنی و سپس پاسخ به سؤالات بر اساس اطلاعات موجود در متن است.

رابطه بین MRC و پاسخگویی به سؤال (QA) رابطه نزدیکی است، زیرا MRC می تواند نوع خاصی از وظیفه QA در نظر گرفته شود. در MRC، سیستم با یک متن (مواد خواندنی) و یک سوال مرتبط با آن متن ارائه می شود. هدف استخراج اطلاعات مربوطه از قسمت برای پاسخ صحیح به سوال است.

مدل‌های MRC معمولاً بر روی مجموعه داده‌های بزرگ متشکل از قسمت‌ها و سؤالات مرتبط آموزش داده می‌شوند، جایی که پاسخ‌ها نیز در طول آموزش ارائه می‌شوند. مدل یاد می گیرد که زمینه و معنای متن را درک کند، اطلاعات مربوطه را شناسایی کند و پاسخ مناسب را تولید یا انتخاب کند.

QA وظیفه گسترده‌تری است که می‌تواند شامل رویکردهای متفاوتی فراتر از درک مطلب باشد. برخی از سیستم‌های QA ممکن است به پایگاه‌های اطلاعاتی ساختاریافته، پایگاه‌های دانش، یا سایر منابع اطلاعاتی برای پاسخ به سؤالات، به جای خواندن و درک مستقیم متن‌ها، متکی باشند.

با این حال، MRC جزء حیاتی بسیاری از سیستم‌های QA مدرن است، به ویژه آنهایی که نیاز به درک و استدلال بر روی داده‌های متنی بدون ساختار دارند. با گنجاندن قابلیت های MRC، سیستم های QA می توانند به طور موثر اطلاعات مربوطه را از منابع متنی استخراج کنند و بر اساس محتوای آن متون پاسخ های دقیقی به سوالات ارائه دهند.

به طور خلاصه، درک مطلب ماشینی یک وظیفه QA خاص است که بر خواندن و درک متن برای پاسخ به سؤالات بر اساس اطلاعات موجود در آنها متمرکز است. این یکی از اجزای کلیدی بسیاری از سیستم‌های پیشرفته QA است که با داده‌های متنی بدون ساختار سروکار دارند، و آنها را قادر می‌سازد تا محتوای متنی را درک کرده و برای ارائه پاسخ‌های دقیق استدلال کنند.

همچنین جواب chatGPT هم که در ادامه آمده است خالی از لطف نیست و بیان دیگر جواب داده است.

**Machine Reading Comprehension (MRC)**

**Definition**: Machine Reading Comprehension (MRC) is a subfield of Natural Language Processing (NLP) that focuses on the ability of a machine to read, understand, and interpret human language texts to answer questions based on the information contained within those texts. MRC systems are designed to process a given text and answer questions about it, demonstrating an understanding of the content similar to human reading comprehension.

**Relation to Question Answering (QA)**

**Question Answering and MRC**: QA systems can be seen as a practical application of MRC. While MRC focuses on understanding and interpreting a given text to extract meaningful information, QA systems leverage this ability to provide direct answers to user queries. Here’s how they are related:

1. **Text-Based QA**: In text-based QA, the system is provided with a text passage and a question about that passage. The system’s task is to comprehend the text and generate an accurate answer based on it. This is a direct application of MRC, where the system’s understanding of the text is tested by how well it can answer questions related to the text.
2. **Core Components**:
   * **Text Processing**: MRC is essential for processing and understanding the text. This involves parsing the text, identifying key entities, relationships, and main ideas.
   * **Question Understanding**: For QA, the system must also understand the question. This involves parsing the question to determine what information is being requested.
   * **Answer Extraction/Generation**: After understanding both the text and the question, the system uses MRC techniques to extract or generate the correct answer from the text.
3. **Types of QA Based on MRC**:
   * **Extractive QA**: The system extracts a span of text from the given passage as the answer. For example, if asked "What is the capital of France?" given a passage about France, the system would extract "Paris".
   * **Abstractive QA**: The system generates an answer in its own words, rather than extracting it directly from the text. This involves more complex understanding and natural language generation.

**Integration of MRC in QA Systems**

* **Data Sources**: MRC-powered QA systems can work with various data sources such as documents, web pages, and structured databases. The system must read and comprehend these sources to answer questions accurately.
* **Knowledge Representation**: Effective MRC involves representing the knowledge in a way that is accessible for answering questions. This includes creating embeddings or other representations of the text that capture its meaning.
* **Inference and Reasoning**: Advanced QA systems may need to perform inference and reasoning over the text to answer questions that are not directly addressed. This requires deeper comprehension and the ability to connect different pieces of information within the text.

**Example**

* **Passage**: "The Eiffel Tower, located in Paris, France, was completed in 1889 and is one of the most famous landmarks in the world."
* **Question**: "When was the Eiffel Tower completed?"
* **Answer**: "1889."

In this example, an MRC system would read the passage, understand the context, and extract the relevant information to answer the question. The QA system uses the comprehension capabilities of the MRC system to provide a precise answer.

**Conclusion**

Machine Reading Comprehension is a foundational technology for Question Answering systems. By enabling machines to understand and interpret text, MRC allows QA systems to provide accurate and contextually relevant answers to user questions, demonstrating a level of understanding similar to human reading comprehension.

مراجع:

<https://chat.openai.com/>   
[https://gemini.google.com/](https://bard.google.com/)   
<https://claude.ai/chats>

## سوال سوم

در زمینه Question Answering، سؤالات را می توان به طور کلی به دو نوع سؤالات factoid و سؤالات non-factoid طبقه بندی کرد. در اینجا توضیحی در مورد هر یک آمده است:

1. سؤالات factoid: سؤالات factoid سؤالاتی هستند که پاسخ های کوتاه و دقیقی دارند که معمولاً می توانند به عنوان حقایق در متن یا پایگاه دانش یافت شوند. این سؤالات معمولاً با کلمات سؤالی مانند «چه کسی»، «چه»، «زمان»، «کجا»، «چند» و غیره شروع می‌شوند.

نمونه هایی از سوالات factoid:

* + پایتخت فرانسه کجاست؟
  + اولین رئیس جمهور آمریکا چه کسی بود؟
  + اعلامیه استقلال کی امضا شد؟
  + چند سیاره در منظومه شمسی ما وجود دارد؟

2. سؤالات non-factoid: سؤالات غیر واقعی که به سؤالات پیچیده یا سؤالات باز نیز گفته می شود، سؤالاتی هستند که نیاز به پاسخ های دقیق و دقیق تری دارند. این سوالات را نمی توان با یک واقعیت ساده یا یک عبارت کوتاه پاسخ داد. در عوض، آنها نیاز به استدلال، تجزیه و تحلیل، یا ترکیب اطلاعات از منابع متعدد دارند.

سؤالات non-factoid اغلب با کلمات سؤالی مانند «چرا»، «چگونه»، «توضیح دادن»، «توصیف» و غیره شروع می‌شوند. پاسخ‌های این سؤالات معمولاً بخش‌ها یا پاراگراف‌هایی طولانی‌تر هستند که توضیحات، نظرات یا گام به گام را ارائه می‌کنند. دستورالعمل های مرحله

نمونه هایی از سوالات non-factoid:

* + فرآیند فتوسنتز در گیاهان چگونه انجام می شود؟
  + علل و اثرات گرمایش زمین را توضیح دهید.
  + طرح کتاب مورد علاقه خود را شرح دهید.
  + چرا ورزش منظم برای حفظ سلامتی مهم است؟

تفاوت اصلی بین سوالات factoid و non-factoid در ماهیت و پیچیدگی پاسخ مورد انتظار نهفته است. سؤالات factoid ی را می توان با حقایق مختصر پاسخ داد، در حالی که سؤالات non-factoid ی نیازمند پاسخ های دقیق تر و دقیق تر است که شامل درک، استدلال و ترکیب اطلاعات از منابع مختلف است.

سیستم‌های پاسخ‌گویی به سؤال باید به گونه‌ای متفاوت طراحی شوند تا بتوانند به طور مؤثر این دو نوع سؤال را مدیریت کنند. سؤالات factoid ی اغلب می توانند با بازیابی و استخراج اطلاعات مرتبط از پایگاه دانش یا مجموعه متن پاسخ داده شوند، در حالی که سؤالات non-factoid ی ممکن است برای ارائه پاسخ های جامع و منسجم نیاز به درک زبان طبیعی، استدلال و توانایی های تولید پیشرفته داشته باشند.

همچنین جواب chatGPT هم که در ادامه آمده است خالی از لطف نیست و بیان دیگر جواب داده است.

In the area of Question Answering (QA), questions can generally be categorized into two types: factoid questions and non-factoid questions. Each type has distinct characteristics and typically requires different approaches for answering.

**Factoid Questions**

**Definition**: Factoid questions are questions that seek factual information, often requiring a specific, concise answer such as a name, date, location, or numerical value.

**Characteristics**:

* **Short and Specific Answers**: The answers are usually short and can be found directly in a text. Examples include names, dates, and numbers.
* **Objective**: The answers are based on objective facts rather than opinions or interpretations.
* **Common Examples**:
  + "What is the capital of France?" (Answer: "Paris")
  + "Who wrote 'Pride and Prejudice'?" (Answer: "Jane Austen")
  + "When did the Apollo 11 mission land on the moon?" (Answer: "July 20, 1969")

**Challenges**:

* **Named Entity Recognition (NER)**: Identifying entities such as people, places, and organizations within a text.
* **Precision**: Ensuring the model pinpoints the exact piece of information required.

**Approaches**:

* **Extractive Models**: These models typically extract the answer directly from a given context, relying on techniques such as token classification and span prediction.
* **Knowledge Bases**: Factoid questions can also be answered using structured data from knowledge bases like Wikidata or DBpedia.

**Non-Factoid Questions**

**Definition**: Non-factoid questions are questions that require more elaborate, often subjective, answers. These answers can be explanatory, descriptive, or opinion-based and usually require a deeper understanding of the context.

**Characteristics**:

* **Longer and Detailed Answers**: The answers are typically longer and more detailed, often requiring synthesis of information from multiple sources or in-depth understanding.
* **Subjective or Interpretative**: These questions can involve subjective judgments, opinions, or detailed explanations.
* **Common Examples**:
  + "Why is the sky blue?"
  + "How does the process of photosynthesis work?"
  + "What are the causes and effects of the Great Depression?"

**Challenges**:

* **Context Understanding**: Requires a deep understanding of context and often multiple sentences or paragraphs to form a complete answer.
* **Natural Language Generation (NLG)**: Generating coherent, fluent, and contextually appropriate text.
* **Subjectivity**: Handling questions that have multiple correct answers or require nuanced, subjective responses.

**Approaches**:

* **Abstractive Models**: These models generate answers in natural language, summarizing information and generating new sentences rather than extracting text verbatim.
* **Conversational AI**: Engaging in dialogue to clarify questions and provide detailed responses.
* **Comprehensive Summarization**: Techniques that aggregate and summarize information from multiple sources to form a coherent response.

**Differences Between Factoid and Non-Factoid Questions**

1. **Answer Length**:
   * **Factoid**: Short, specific answers.
   * **Non-Factoid**: Longer, detailed, and often explanatory answers.
2. **Answer Source**:
   * **Factoid**: Directly found in text or structured data.
   * **Non-Factoid**: Synthesized from context or generated using multiple pieces of information.
3. **Nature of Information**:
   * **Factoid**: Objective and factual.
   * **Non-Factoid**: Can be subjective or interpretative.
4. **Complexity**:
   * **Factoid**: Simpler, requiring basic retrieval and extraction.
   * **Non-Factoid**: More complex, often needing understanding and generation capabilities.

Understanding these distinctions helps in designing and deploying appropriate QA systems tailored to the nature of the questions they are expected to answer.

مراجع:

<https://chat.openai.com/>   
[https://gemini.google.com/](https://bard.google.com/)   
<https://claude.ai/chats>

## سوال چهارم

ترانسفورمرها و شبکه‌های عصبی بازگشتی (RNN) دو معماری متفاوت هستند که برای وظایف پردازش متوالی داده در پردازش زبان طبیعی (NLP) و سایر حوزه‌ها استفاده می‌شوند. در اینجا برخی از مزایا و معایب کلیدی ترانسفورمرها در مقایسه با RNN آورده شده است:

مزایای ترانسفورمرها:

* 1. پردازش موازی: ترانسفورمرها می توانند توالی های ورودی را به صورت موازی پردازش کنند و آموزش و استنتاج سریع تری را در مقایسه با RNN ها که توالی ها را به صورت متوالی پردازش می کنند، ممکن می سازد.
  2. وابستگی های دوربرد: ترانسفورمرها می توانند به طور موثر وابستگی های دوربرد را به دلیل مکانیسم توجه به خود، که امکان اتصال مستقیم بین توکن ها را بدون در نظر گرفتن موقعیت آنها در دنباله، فراهم می کند، در دنباله ها ثبت کنند. RNN ها با گرفتن وابستگی های دوربرد به دلیل مشکل گرادیان ناپدید/انفجار مشکل دارند.
  3. بدون محاسبات متوالی: ترانسفورمرها به محاسبات متوالی نیاز ندارند، و برای کارهایی که نیازی به ترتیب دقیق ورودی‌ها ندارند، مانند ترجمه ماشینی یا مدل‌سازی زبان، کارآمدتر می‌شوند.
  4. عملکرد بهتر: ترانسفورمرها عملکرد بهتری را در وظایف مختلف NLP در مقایسه با RNN ها نشان داده اند، به ویژه برای توالی های طولانی تر.

معایب ترانسفورمرها:

* 1. پیچیدگی درجه دوم: مکانیزم توجه به خود در ترانسفورمرها دارای پیچیدگی درجه دوم نسبت به طول دنباله است، که آنها را از نظر محاسباتی برای توالی های بسیار طولانی گران می کند.
  2. عدم مفهوم صریح زمان: ترانسفورمرها مفهوم صریحی از زمان یا ترتیب توالی ندارند، که می تواند محدودیتی برای کارهایی باشد که نیاز به مدل سازی زمانی دقیق دارند، مانند تشخیص گفتار یا پیش بینی سری های زمانی.
  3. رمزگذاری موقعیتی: ترانسفورمرها برای گنجاندن اطلاعات متوالی به رمزگذاری موقعیتی متکی هستند، که می تواند کمتر از پردازش موقت در RNN ها بصری باشد.
  4. نیازهای حافظه بزرگتر: ترانسفورمرها به دلیل تعداد پارامترهای بیشتر و نیاز به ذخیره فعال سازی های میانی در طول آموزش و استنتاج، معمولاً به حافظه بیشتری نسبت به RNN ها نیاز دارند.

مزایای RNN:

* 1. پردازش متوالی: RNN ها برای کارهایی که نیاز به پردازش متوالی دقیق دارند، مانند تشخیص گفتار یا تجزیه و تحلیل سری های زمانی، مناسب هستند.
  2. مدل سازی زمانی صریح: RNN ها مفهوم صریحی از زمان دارند و می توانند وابستگی های زمانی را به طور طبیعی تر از ترانسفورمرها مدل کنند.
  3. پیچیدگی محاسباتی کمتر: RNN ها پیچیدگی محاسباتی کمتری برای وظایف با ورودی های متوالی دارند، زیرا آنها یک توکن را در یک زمان پردازش می کنند.
  4. نیازهای حافظه کوچکتر: RNN ها معمولاً به حافظه کمتری نسبت به ترانسفورمرها نیاز دارند، به خصوص برای توالی های طولانی تر.

معایب RNN ها:

* + 1. پردازش متوالی: در حالی که پردازش متوالی در برخی موارد یک مزیت است، می تواند برای کارهایی که نیازی به ترتیب دقیق متوالی ندارند نیز گلوگاه باشد.
    2. وابستگی‌های دوربرد: RNN‌ها با گرفتن وابستگی‌های دوربرد به دلیل مشکل گرادیان ناپدید/انفجار، که می‌تواند عملکرد آن‌ها را در کارهای با دنباله‌های طولانی محدود کند، با مشکل مواجه هستند.
    3. چالش های موازی سازی: RNN ها ذاتاً متوالی هستند، و موازی کردن محاسبات آنها به طور موثر، به ویژه در طول آموزش، چالش برانگیز است.
    4. آموزش و استنتاج کندتر: به دلیل ماهیت ترتیبی، RNNها می توانند برای آموزش و استنتاج در مقایسه با ترانسفورمرها که می توانند توالی ها را به صورت موازی پردازش کنند کندتر عمل کنند.

در سال‌های اخیر، ترانسفورمرها در NLP و سایر حوزه‌ها به دلیل عملکرد برتر در بسیاری از وظایف، به‌ویژه مواردی که شامل توالی‌های طولانی هستند، رایج‌تر شده‌اند. با این حال، RNN ها هنوز هم مزایای خود را برای کارهایی دارند که نیاز به مدل سازی زمانی دقیق دارند یا طول توالی کوتاه تری دارند. در عمل، انتخاب بین ترانسفورمرها و RNN ها اغلب به الزامات وظیفه خاص، منابع محاسباتی و مبادلات بین عملکرد و کارایی بستگی دارد.

همچنین جواب chatGPT هم که در ادامه آمده است خالی از لطف نیست و بیان دیگر جواب داده است.

Transformers and Recurrent Neural Networks (RNNs) are two popular architectures used in Natural Language Processing (NLP). Each has its advantages and disadvantages. Here's a detailed comparison:

**Transformers**

**Advantages**

1. **Parallelization**:
   * **Parallel Processing**: Transformers can process all elements of a sequence simultaneously due to their self-attention mechanism, allowing for significant parallelization and faster training times.
   * **Efficiency**: This parallelism leads to more efficient use of modern hardware like GPUs and TPUs.
2. **Long-Range Dependencies**:
   * **Effective Handling**: Transformers can model long-range dependencies in sequences more effectively than RNNs. The self-attention mechanism allows direct connections between any two positions in the sequence, irrespective of their distance.
   * **Context Capture**: This ability is crucial for tasks requiring understanding of the entire context, such as machine translation and text generation.
3. **Scalability**:
   * **Scalable Architecture**: Transformers can easily scale up by increasing the number of layers or attention heads, improving performance on larger datasets and more complex tasks.
   * **Flexibility**: They are versatile and can be adapted for various NLP tasks, including translation, summarization, and question answering.
4. **State-of-the-Art Performance**:
   * **Pretrained Models**: Transformers, especially large pretrained models like BERT, GPT, and T5, have achieved state-of-the-art results across a wide range of NLP benchmarks.

**Disadvantages**

1. **Computationally Intensive**:
   * **High Resource Usage**: The self-attention mechanism has a quadratic time and memory complexity concerning the sequence length, making transformers computationally expensive for very long sequences.
   * **Hardware Requirements**: Requires substantial computational resources for training, which can be a limitation for smaller organizations or individual researchers.
2. **Large Memory Footprint**:
   * **Memory Consumption**: Due to their complex architecture and parallel processing requirements, transformers consume a significant amount of memory, especially during training.
3. **Complexity**:
   * **Model Complexity**: Transformers are more complex to implement and require careful tuning of many hyperparameters, such as the number of attention heads, layers, and learning rates.

**Recurrent Neural Networks (RNNs)**

**Advantages**

1. **Sequential Data Handling**:
   * **Natural Fit for Sequences**: RNNs are inherently designed to handle sequential data, making them a natural choice for tasks involving time series or any data where order matters.
   * **Time-Step Processing**: They process sequences one step at a time, maintaining a hidden state that captures information from previous steps.
2. **Lower Computational Requirements**:
   * **Less Intensive**: Generally, RNNs require less computational power and memory compared to transformers, making them more accessible for simpler tasks and smaller datasets.
   * **Simpler Architecture**: Their simpler architecture can be an advantage in applications where large-scale models are unnecessary.
3. **Memory Efficiency for Short Sequences**:
   * **Efficiency**: For tasks involving short sequences, RNNs can be more memory-efficient and faster than transformers.

**Disadvantages**

1. **Difficulty with Long-Range Dependencies**:
   * **Vanishing Gradient Problem**: RNNs struggle with capturing long-range dependencies due to issues like vanishing and exploding gradients, especially when using standard RNNs without modifications.
   * **Context Limitation**: They may forget important context from earlier in the sequence, impacting their ability to model long-range dependencies effectively.
2. **Sequential Processing**:
   * **Slower Training**: RNNs process data sequentially, making it harder to parallelize computations and resulting in slower training times compared to transformers.
   * **Inference Speed**: Similarly, inference can also be slower because each time step depends on the previous one.
3. **Model Complexity for Long Sequences**:
   * **Increased Complexity**: While RNNs are simpler in architecture, making them work well for long sequences often requires adding complexity through architectures like LSTM (Long Short-Term Memory) or GRU (Gated Recurrent Unit), which aim to mitigate some of the limitations but add to the computational burden.

**Conclusion**

**Transformers** are powerful for their ability to handle long-range dependencies, parallelize computations, and achieve state-of-the-art results in many NLP tasks. However, they require significant computational resources and are complex to implement. **RNNs**, on the other hand, are naturally suited for sequential data and are computationally less intensive, but they struggle with long sequences and are slower due to their sequential processing nature. The choice between the two often depends on the specific requirements of the task, available computational resources, and the length and nature of the input sequences.

مراجع:

<https://chat.openai.com/>   
[https://gemini.google.com/](https://bard.google.com/)   
<https://claude.ai/chats>

## سوال پنجم

رمزگذاری موقعیتی یک مؤلفه مهم در مدل‌های مبتنی بر ترانسفورمر است که به مدل اجازه می‌دهد ترتیب یا موقعیت توکن‌ها را در توالی ورودی ثبت و ترکیب کند. این مهم است زیرا مکانیسم توجه به خود در ترانسفورمرها، توالی ورودی را به عنوان مجموعه ای از نشانه ها، بدون هیچ گونه مفهوم ذاتی ترتیب یا موقعیت در نظر می گیرد.

در زبان طبیعی و بسیاری از حوزه‌های داده‌های متوالی دیگر، ترتیب یا موقعیت نشانه‌ها معنا و زمینه قابل توجهی دارد. به عنوان مثال، جملات «سگ تعقیب گربه را تعقیب کرد» و «گربه سگ را تعقیب کرد» با وجود اینکه مجموعه کلمات یکسانی دارند، معانی کاملا متفاوتی دارند. برای گرفتن این اطلاعات موقعیتی، مدل های ترانسفورمر از رمزگذاری موقعیتی استفاده می کنند.

رمزگذاری موقعیتی یک نمایش برداری است که به جاسازی‌های ورودی هر نشانه در دنباله اضافه می‌شود. این بردارهای رمزگذاری موقعیتی برای هر موقعیت در دنباله منحصر به فرد هستند و برای رمزگذاری اطلاعات موقعیت نسبی یا مطلق توکن ها طراحی شده اند. دو نوع اصلی از استراتژی های رمزگذاری موقعیتی وجود دارد:

1. رمزگذاری موقعیت سینوس و کسینوس:

* + این رایج ترین و اصلی ترین رویکرد مورد استفاده در مقاله ترانسفورمر سمینال است.
  + از مجموعه ثابتی از توابع سینوسی و کسینوس با فرکانس های مختلف برای تولید بردارهای رمزگذاری موقعیتی استفاده می کند.
  + بردارهای رمزگذاری موقعیتی به دست آمده برای هر موقعیت منحصر به فرد هستند و دارای یک الگوی خاص هستند که می توان با مکانیسم توجه به خود یاد گرفت.

2. رمزگذاری موقعیتی آموخته شده:

* + در این رویکرد، بردارهای رمزگذاری موقعیتی به عنوان بخشی از پارامترهای مدل در طول آموزش یاد می‌گیرند.
  + مدل بردارهای رمزگذاری موقعیتی بهینه را می آموزد که اطلاعات موقعیت را به طور موثر برای کار و داده خاص دریافت می کند.

اهمیت رمزگذاری موقعیتی در مدل‌های مبتنی بر ترانسفورمر در توانایی آن در ارائه مکانیسم توجه به خود با اطلاعات موقعیتی لازم برای ثبت صحیح ترتیب متوالی و زمینه توکن‌ها نهفته است. بدون رمزگذاری موقعیتی، مکانیسم توجه به خود، توالی ورودی را به عنوان کیسه ای از نشانه ها در نظر می گیرد، ترتیب آنها را نادیده می گیرد و به طور بالقوه منجر به خروجی های نادرست یا بی معنی می شود.

با ترکیب کدگذاری موقعیتی، مدل‌های ترانسفورمر می‌توانند به طور موثر وابستگی‌های دوربرد و اطلاعات زمینه‌ای را ضبط کنند، در حالی که همچنان توانایی پردازش توالی‌ها را به صورت موازی حفظ می‌کنند، که یک مزیت کلیدی نسبت به مدل‌های متوالی مانند شبکه‌های عصبی بازگشتی (RNNs) است.

رمزگذاری موقعیتی نقش مهمی در موفقیت مدل‌های مبتنی بر ترانسفورمر در کارهای مختلف پردازش زبان طبیعی، مانند ترجمه ماشینی، مدل‌سازی زبان، تولید متن و پاسخ‌گویی به سؤال، و همچنین در حوزه‌های دیگر مانند بینایی رایانه و پردازش گفتار ایفا می‌کند. گرفتن اطلاعات متوالی ضروری است.

همچنین جواب chatGPT هم که در ادامه آمده است خالی از لطف نیست و بیان دیگر جواب داده است.

**Positional Encoding in Transformer Models**

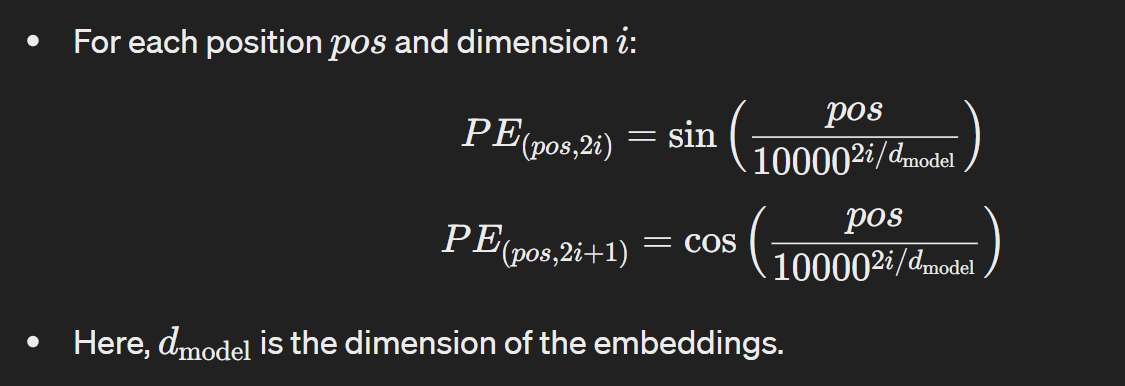
**Definition**: Positional encoding is a technique used in transformer models to inject information about the position of each token in a sequence. Unlike Recurrent Neural Networks (RNNs) that process input tokens sequentially and naturally maintain the order of tokens, transformer models process the entire sequence in parallel. As a result, they lack a built-in mechanism to capture the positional information of tokens in a sequence. Positional encoding addresses this issue by adding information about the positions of tokens to their embeddings.

**Importance of Positional Encoding**

1. **Maintaining Sequence Order**:
   * **Order Information**: Positional encoding helps the model understand the order of tokens in the sequence, which is crucial for understanding the context and meaning of the text.
   * **Sequential Context**: Without positional encodings, the model would treat the input tokens as a bag of words, ignoring their sequential nature and losing critical context.
2. **Enabling Self-Attention**:
   * **Self-Attention Mechanism**: The self-attention mechanism in transformers calculates the relationship between each pair of tokens in a sequence. Positional encoding ensures that these calculations take the positions of tokens into account, allowing the model to differentiate between tokens based on their positions.
   * **Contextual Relationships**: By incorporating positional information, the self-attention mechanism can effectively capture the dependencies between tokens that are far apart in the sequence.
3. **Improving Model Performance**:
   * **Enhanced Understanding**: Positional encoding improves the model's ability to understand and generate coherent and contextually appropriate responses by providing a sense of order and structure to the input data.
   * **Better Results**: Models with positional encodings tend to perform better on tasks requiring an understanding of the sequential nature of text, such as translation, summarization, and text generation.

**How Positional Encoding Works**

1. **Encoding Functions**:
   * **Sinusoidal Functions**: The most common method of positional encoding uses sinusoidal functions of different frequencies. This method generates a unique positional encoding for each position in the sequence.
   * **Formula**:

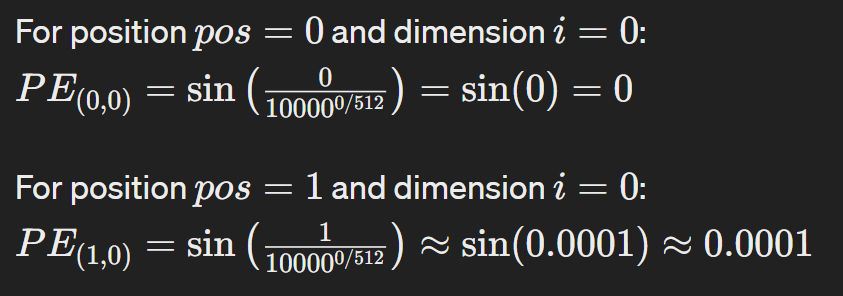


1. **Properties of Sinusoidal Encoding**:
   * **Smooth Transitions**: The use of sine and cosine functions creates smooth transitions between positions, which helps the model generalize better to unseen lengths.
   * **Relative Position**: Sinusoidal encodings allow the model to compute relative positions, which is useful for capturing the distance between tokens.
2. **Alternative Methods**:
   * **Learnable Positional Embeddings**: Some transformer models use learnable positional embeddings, where the positional information is learned during training rather than being fixed. This approach allows the model to adapt the positional encodings based on the data.
   * **Absolute vs. Relative**: Besides absolute positional encoding, relative positional encoding can be used, where the model learns to encode the relative distances between tokens rather than their absolute positions.

**Example of Sinusoidal Positional Encoding**

For a given sequence length 𝐿*L* and embedding dimension 𝑑model*d*model​, positional encodings are added to the input embeddings before feeding them into the transformer model.

* **Input Sequence**: ["The", "quick", "brown", "fox"]
* **Position Indices**: [0, 1, 2, 3]
* **Positional Encoding**: Generate a unique encoding for each position using the sinusoidal formula.

**Example Calculation**:   


These positional encodings are then added to the word embeddings to provide the model with both the content and positional information.

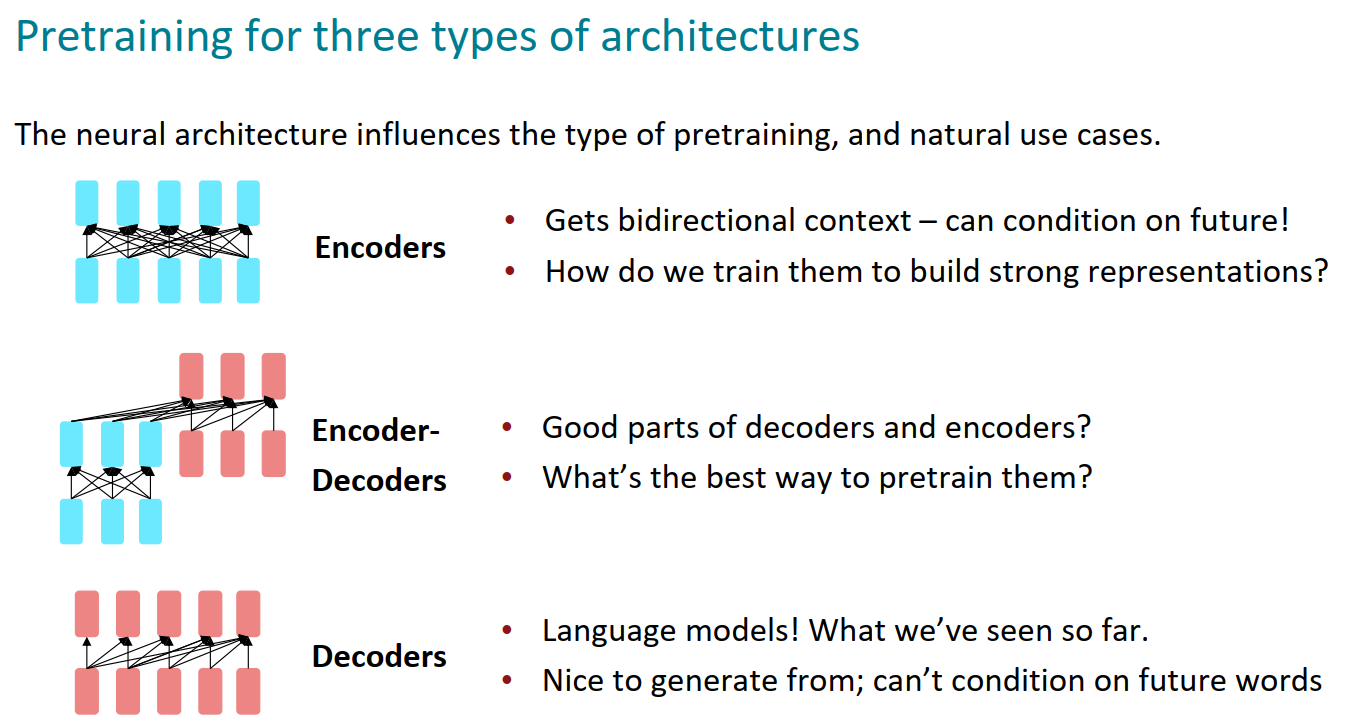
**Conclusion**

Positional encoding is a crucial component of transformer models, enabling them to capture the order of tokens in a sequence. By incorporating positional information, transformers can leverage their powerful parallel processing capabilities while maintaining an understanding of the sequential nature of the input data, leading to improved performance on various NLP tasks.

مراجع:

<https://chat.openai.com/>   
[https://gemini.google.com/](https://bard.google.com/)   
<https://claude.ai/chats>

## سوال ششم



در مدل های مبتنی بر ترانسفورماتور، معماری را می توان به طور کلی به سه نوع طبقه بندی کرد: رمزگذار-رمز، فقط رمزگشا و فقط رمزگذار. هر معماری برای انواع خاصی از وظایف طراحی شده است و ویژگی های منحصر به فرد خود را دارد. در اینجا توضیح مفصلی در مورد هر نوع و تفاوت آنها آورده شده است:

معماری رمزگذار-رمزگشا

توضیحات: معماری رمزگذار-رمزگشا از دو جزء اصلی تشکیل شده است: رمزگذار و رمزگشا. این معماری معمولاً برای کارهایی استفاده می شود که نیاز به تبدیل ورودی-خروجی دارند، مانند ترجمه ماشینی.

اجزاء:

* 1. رمزگذار: توالی ورودی را پردازش می کند و آن را به مجموعه ای از نمایش های پیوسته تبدیل می کند. رمزگذار از چندین لایه تشکیل شده است که هر یک شامل شبکه های عصبی خود توجه و پیشخور است.
  2. رمزگشا: دنباله خروجی را بر اساس نمایش های تولید شده توسط رمزگذار ایجاد می کند. رمزگشا همچنین از چندین لایه تشکیل شده است که هر لایه شامل توجه به خود، توجه رمزگذار-رمزگشا و شبکه های عصبی پیشخور است.

عملکرد:

* + رمزگذار یک دنباله ورودی می گیرد و یک نمایش آگاه از زمینه تولید می کند.

رمزگشا این نمایش را همراه با توکن های تولید شده قبلی می گیرد تا توکن بعدی را در دنباله خروجی تولید کند.

مثال ها:

* + ترجمه ماشینی (به عنوان مثال، ترجمه انگلیسی به فرانسوی)
  + خلاصه سازی متن
  + پاسخ به سؤال (هنگام ایجاد پاسخ بر اساس یک زمینه معین)

مزایای:

* + تطبیق پذیری: می تواند وظایف پیچیده توالی به دنباله را انجام دهد.
  + درک متنی: رمزگذار یک نمایش جامع از توالی ورودی را به تصویر می‌کشد که به رمزگشا کمک می‌کند خروجی‌های دقیق تولید کند.

معایب:

* + پیچیدگی: به هر دو مؤلفه رمزگذار و رمزگشا نیاز دارد که مدل را پیچیده تر و محاسباتی فشرده تر می کند.

معماری فقط رمزگشا

توضیحات: معماری فقط رمزگشا فقط از بخش رمزگشای مدل ترانسفورماتور تشکیل شده است. این معماری در درجه اول برای کارهای مولد استفاده می شود که در آن مدل بر اساس برخی از اعلان های ورودی متن تولید می کند.

اجزاء:

* + رمزگشا: مشابه رمزگشا در معماری رمزگذار-رمزگشاست اما به صورت مجزا استفاده می شود. این شامل چندین لایه با شبکه های عصبی خود توجه و پیشخور است.

عملکرد:

* + این مدل متن را به روش خود رگرسیون تولید می کند، و یک نشانه را در هر زمان بر اساس دستور ورودی و نشانه های تولید شده قبلی پیش بینی می کند.

مثال ها:

* + تولید متن (به عنوان مثال، مدل‌های GPT)
  + مدل سازی زبان
  + عوامل گفتگو

مزایای:

* + سادگی: پیچیدگی کمتری نسبت به معماری رمزگذار-رمزگشا دارد زیرا فقط شامل رمزگشا می شود.
  + کارایی در تولید: برای کارهایی که نیاز به تولید توالی از یک اعلان دارند، مناسب است.

معایب:

* + محدود به وظایف مولد: اساساً برای تولید به جای وظایفی که نیاز به تغییرات ورودی-خروجی دقیق دارند طراحی شده است.

معماری فقط رمزگذار

توضیحات: معماری فقط رمزگذار فقط از بخش رمزگذار مدل ترانسفورماتور تشکیل شده است. این معماری معمولاً برای کارهایی استفاده می شود که شامل درک و طبقه بندی توالی های ورودی به جای تولید توالی های جدید است.

اجزاء:

* + رمزگذار: متشکل از لایه های متعدد با شبکه های عصبی خود توجه و پیشخور.

عملکرد:

* + مدل کل توالی ورودی را پردازش می کند تا بازنمایی های متنی تولید کند، که سپس می تواند برای کارهای مختلف طبقه بندی یا رگرسیون استفاده شود.

مثال ها:

* + طبقه بندی متن (به عنوان مثال، تجزیه و تحلیل احساسات)
  + شناسایی نهاد نامگذاری شده (NER)
  + تولید جاسازی متن (به عنوان مثال، BERT)

مزایای:

* + Simplicity for Understanding Tasks: بر درک و پردازش توالی های ورودی بدون نیاز به یک جزء مولد تمرکز می کند.
  + کارایی برای طبقه بندی: برای کارهایی که نیاز به درک دقیق توالی ورودی دارند، موثر است.

معایب:

* + برای نسل مناسب نیست: نمی‌تواند دنباله‌های جدیدی ایجاد کند، و استفاده از آن را به درک و طبقه‌بندی وظایف محدود می‌کند.

خلاصه تفاوت ها

1. اجزاء:

* + رمزگذار - رمزگشا: هر دو رمزگذار و رمزگشا وجود دارند.
  + Decoder-Only: فقط رمزگشا موجود است.
  + Encoder-Only: فقط رمزگذار موجود است.

2. موارد استفاده اولیه:

* + رمزگذار-رمزگشا: کارهای ترتیب به دنباله (به عنوان مثال، ترجمه، خلاصه سازی).
  + فقط رمزگشا: وظایف تولیدی (به عنوان مثال، تولید متن، هوش مصنوعی مکالمه).
  + Encoder-Only: درک و طبقه بندی وظایف (به عنوان مثال، تجزیه و تحلیل احساسات، NER).

3. پیچیدگی:

* + Encoder-Decoder: پیچیده ترین به دلیل وجود هر دو جزء.
  + Decoder-Only: ساده تر از رمزگذار-رمزگشا، اما متمرکز بر تولید.
  + Encoder-Only: ساده ترین از نظر معماری، متمرکز بر درک.

4. موازی سازی و کارایی:

* + رمزگذار-رمزگشا: به دلیل اجزای دوگانه، از نظر محاسباتی فشرده تر است.
  + فقط رمزگشا: کارآمد برای تولید، استفاده از ویژگی های خودبازگشت.
  + Encoder-Only: کارآمد برای درک وظایف، با انجام تمام پردازش ها به صورت موازی.

هر معماری برای انواع مختلف وظایف NLP بهینه شده است و انتخاب معماری به نیازهای خاص کار در دست بستگی دارد.

همچنین جواب claude هم که در ادامه آمده است خالی از لطف نیست و بیان دیگر جواب داده است.

Transformer-based models can have different architectural configurations, including encoder-decoder, decoder-only, and encoder-only architectures. Each of these configurations is suited for different types of tasks and has its own strengths and limitations. Here's an explanation of the differences between them:

1. Encoder-Decoder Architecture:
   * This is the original and most common configuration introduced in the seminal Transformer paper.
   * It consists of two main components: an encoder and a decoder.
   * The encoder processes the input sequence and generates a contextual representation, which is then passed to the decoder.
   * The decoder attends to the encoder's output and generates the output sequence token by token.
   * This architecture is commonly used for sequence-to-sequence tasks, such as machine translation, summarization, and dialogue systems.
   * Examples: Transformer (Vaswani et al., 2017), T5 (Raffel et al., 2019), BERT (Devlin et al., 2018) for sequence-to-sequence fine-tuning.
2. Decoder-Only Architecture:
   * This configuration consists only of the decoder component, without an encoder.
   * The input sequence is typically prepended with a special start token and fed directly into the decoder.
   * The decoder generates the output sequence token by token, attending to the previously generated tokens and the input sequence.
   * This architecture is well-suited for language generation tasks, such as open-ended text generation, dialogue response generation, and story completion.
   * Examples: GPT (Radford et al., 2018), GPT-2 (Radford et al., 2019), GPT-3 (Brown et al., 2020).
3. Encoder-Only Architecture:
   * This configuration consists only of the encoder component, without a decoder.
   * The input sequence is processed by the encoder, and the final contextual representation is used for various tasks, such as text classification, sentiment analysis, or sequence labeling.
   * The encoder-only architecture is often used for transfer learning, where the pre-trained encoder is fine-tuned on downstream tasks.
   * Examples: BERT (Devlin et al., 2018), RoBERTa (Liu et al., 2019), DistilBERT (Sanh et al., 2019) for transfer learning tasks.

The choice of architecture depends on the specific task and requirements:

* Encoder-decoder is suitable for tasks that involve generating an output sequence from an input sequence, such as translation or summarization.
* Decoder-only is well-suited for open-ended generation tasks, where the input is optional or minimal.
* Encoder-only is often used for transfer learning and fine-tuning on various natural language understanding tasks, such as text classification, named entity recognition, or question answering.

It's worth noting that some models, like BART (Lewis et al., 2019) and T5 (Raffel et al., 2019), can be configured as either encoder-decoder or encoder-only architectures, providing flexibility for different tasks.

The architectural choice also influences the pretraining tasks and objectives used to train these models. For example, encoder-decoders are often pretrained on tasks like machine translation or summarization, while decoder-only models are pretrained on language modeling or generative objectives, and encoder-only models are pretrained on tasks like masked language modeling or next sentence prediction.

مراجع:

<https://chat.openai.com/>   
[https://gemini.google.com/](https://bard.google.com/)   
<https://claude.ai/chats>

## سوال هفتم

پاسخ سوال استخراجی(Extractive Question Answering):

* + در QA استخراجی، سیستم عبارات یا جملات کلمه به کلمه را از متن یا پایگاه دانش داده شده به عنوان پاسخ به سوال شناسایی و استخراج می کند.
  + سیستم متن جدیدی تولید نمی کند. به سادگی قطعات مربوطه را به کلمه کلمه از منبع منبع پیدا کرده و کپی می کند.
  + این رویکرد معمولاً برای سؤالات فاکتوئیدی استفاده می شود که پاسخ آن به صراحت در متن بیان شده است.
  + سیستم‌های QA استخراجی بر تکنیک‌هایی مانند بازیابی اطلاعات، شناسایی موجودیت نام‌گذاری شده و پیش‌بینی دامنه پاسخ برای شناسایی گستره پاسخ مربوطه تکیه دارند.

پاسخ انتزاعی به سوال(Abstractive Question Answering):

* + در QA انتزاعی، سیستم به جای استخراج عبارات کلمه به کلمه از منبع منبع، متن جدیدی را برای ترکیب و ارائه پاسخ تولید می کند.
  + سیستم باید سوال را درک کند، اطلاعات مربوطه را از منابع متعدد استدلال کند، و یک پاسخ زبان طبیعی ایجاد کند که ممکن است کلمه به کلمه در متن ورودی وجود نداشته باشد.
  + این رویکرد بیشتر برای سوالات غیر واقعی و باز که نیاز به استدلال، تجزیه و تحلیل و خلاصه سازی دارند مناسب است.
  + سیستم های QA انتزاعی اغلب از تکنیک های تولید زبان طبیعی، مانند مدل های دنباله به دنباله یا مدل های زبان، برای تولید پاسخ های منسجم و روان استفاده می کنند.

تفاوت های کلیدی:

1. تولید پاسخ: سیستم‌های استخراجی پاسخ‌ها را به‌لفظ استخراج می‌کنند، در حالی که سیستم‌های انتزاعی متن جدیدی را به عنوان پاسخ تولید می‌کنند.
2. انواع سؤال: QA استخراجی برای سؤالات فاکتوئیدی مناسب تر است، در حالی که QA انتزاعی برای سؤالات غیر واقعی و باز مناسب تر است.
3. تکنیک ها: QA استخراجی بر بازیابی اطلاعات و پیش بینی دامنه پاسخ متکی است، در حالی که QA انتزاعی از تولید زبان طبیعی و قابلیت های استدلال استفاده می کند.
4. پیچیدگی: QA انتزاعی به طور کلی پیچیده تر و چالش برانگیزتر از QA استخراجی است، زیرا به درک، استدلال و قابلیت های تولید نیاز دارد.

توجه به این نکته مهم است که برخی از سیستم‌های QA پیشرفته ممکن است هر دو روش استخراجی و انتزاعی را با هم ترکیب کنند و از تکنیک‌های استخراجی برای شناسایی تکه‌های مرتبط و تکنیک‌های انتزاعی برای ایجاد پاسخ‌های طبیعی و منسجم‌تر با ترکیب و بازنویسی اطلاعات استخراج‌شده استفاده کنند.

همچنین جواب chatGPT هم که در ادامه آمده است خالی از لطف نیست و بیان دیگر جواب داده است.

Extractive and abstractive question answering (QA) are two fundamental approaches used in QA systems to generate answers. Here's an explanation of each type along with their differences:

**Extractive Question Answering**

**Definition**: Extractive QA systems identify and extract a span of text directly from a given document or passage as the answer to a question. The answer is a contiguous sequence of words from the source text.

**Characteristics**:

* **Text Span Selection**: The system selects a portion of the text that directly answers the question.
* **Source Reliance**: The answers are always verbatim excerpts from the source material.
* **Implementation**: Typically involves techniques like text span prediction where the start and end positions of the answer within the text are identified.

**Example**:

* **Passage**: "The capital of France is Paris."
* **Question**: "What is the capital of France?"
* **Answer**: "Paris."

**Abstractive Question Answering**

**Definition**: Abstractive QA systems generate answers using natural language generation techniques. The answers are not limited to the exact wording of the source text but are instead generated based on the system's understanding and synthesis of the information.

**Characteristics**:

* **Paraphrasing**: The system can rephrase information from the source text to form a coherent and concise answer.
* **Flexibility**: Allows for more flexible and potentially more informative answers that may combine information from multiple parts of the text or add explanatory context.
* **Implementation**: Involves advanced natural language generation models, often based on sequence-to-sequence architectures or transformers like GPT or BERT.

**Example**:

* **Passage**: "The Eiffel Tower, an iconic structure located in Paris, France, was completed in 1889 and stands as a symbol of French architectural innovation."
* **Question**: "When was the Eiffel Tower completed and what does it symbolize?"
* **Answer**: "The Eiffel Tower was completed in 1889 and symbolizes French architectural innovation."

**Key Differences**

1. **Answer Generation**:
   * **Extractive**: Directly pulls the answer from the text as a verbatim snippet.
   * **Abstractive**: Generates the answer by rephrasing or synthesizing information from the text.
2. **Flexibility and Expressiveness**:
   * **Extractive**: Limited to the exact wording present in the text, which can sometimes result in less natural or informative answers.
   * **Abstractive**: Can produce more natural, coherent, and contextually enriched answers.
3. **Complexity**:
   * **Extractive**: Generally simpler to implement because it involves selecting text spans.
   * **Abstractive**: More complex due to the need for natural language generation and ensuring the generated answer is accurate and coherent.
4. **Information Integration**:
   * **Extractive**: Restricted to information that is explicitly and contiguously stated in the text.
   * **Abstractive**: Can integrate and synthesize information from different parts of the text or even infer information that is not explicitly stated.
5. **Use Cases**:
   * **Extractive**: Suitable for tasks where precise excerpts are required, such as legal document review or factual question answering where the exact wording is important.
   * **Abstractive**: Suitable for tasks requiring more natural and human-like responses, such as conversational agents or summarization.

**Conclusion**

Both extractive and abstractive QA systems play crucial roles depending on the requirements of the task at hand. Extractive QA is useful for straightforward, fact-based questions where exact text matches are needed. Abstractive QA is beneficial for generating more natural, comprehensive answers, especially when combining or interpreting information is necessary.

مراجع:

<https://chat.openai.com/>   
[https://gemini.google.com/](https://bard.google.com/)   
<https://claude.ai/chats>

## سوال هشتم

مدل: BERT-Base-Cased

نمونه جمله ورودی: "The quick brown fox jumps over the lazy dog."

Task: شناسایی نقش دستوری کلمات (POS Tagging: Part Of Speech Tagging)

چرا داشتن Multiple Head Attention مفید است؟

در مدل BERT (Bidirectional Encoder Representations from Transformers)، مکانیسم توجه چند سر(multi-head attention mechanism) نقش مهمی در گرفتن جنبه‌های مختلف توالی ورودی و مدل‌سازی وابستگی‌های پیچیده بین نشانه‌ها ایفا می‌کند. داشتن سرهای توجه متعدد، به جای یک مکانیسم توجه واحد، به چند دلیل مفید است:

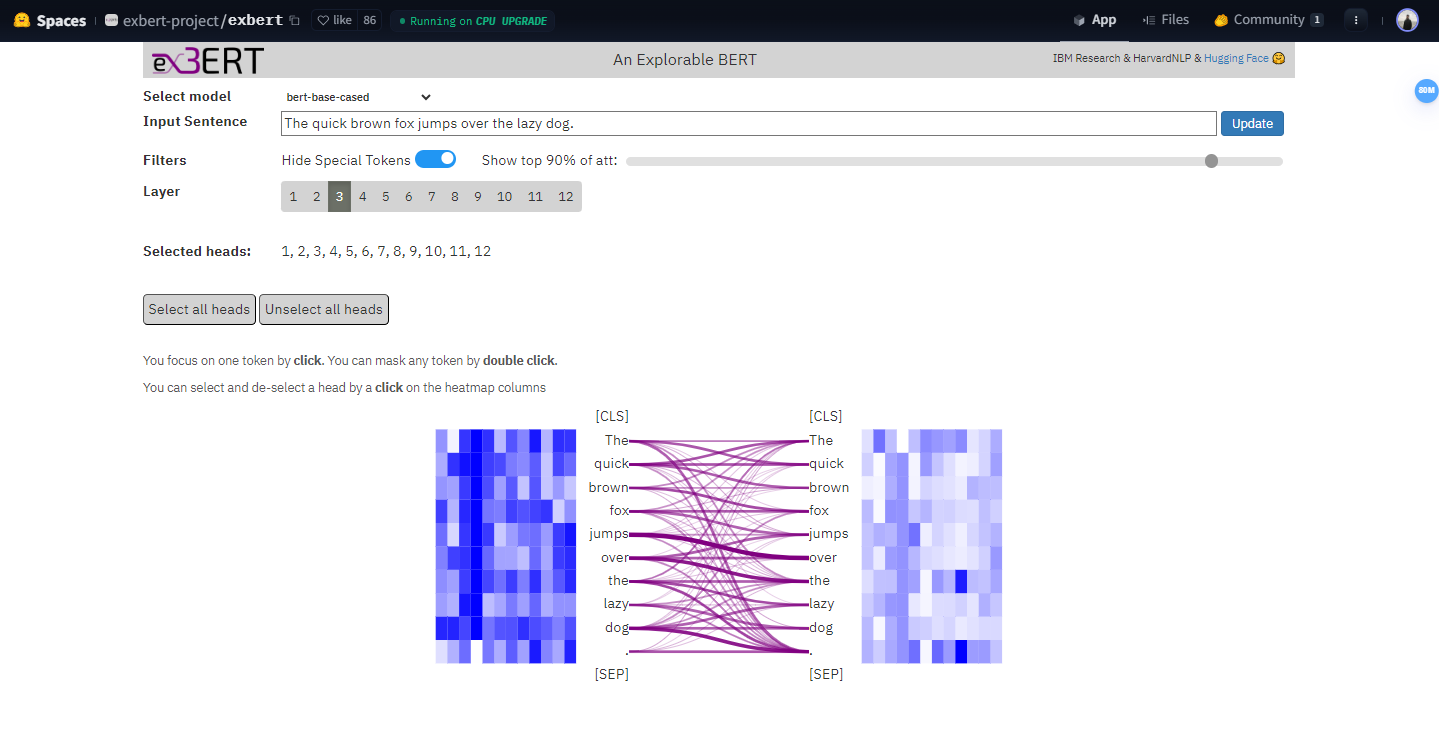
* 1. موازی سازی و کارایی(Parallelization and Efficiency): هر سر توجه می تواند به صورت موازی عمل کند و امکان محاسبه کارآمد و گرفتن انواع مختلف روابط را به طور همزمان فراهم کند. این موازی سازی کارایی محاسباتی کلی مدل را بهبود می بخشد.
  2. در نظر گرفتن روابط مختلف و متنوع(Capturing Different and Diverse Relationships): هر سر توجه به طور بالقوه می تواند انواع مختلفی از روابط یا وابستگی های بین نشانه ها را در دنباله ورودی یاد بگیرد و بگیرد. به عنوان مثال، یک سر ممکن است بر روابط نحوی، دیگری بر روابط معنایی، و دیگری بر روابط موقعیتی یا متنی تمرکز کند.
  3. قدرت بیان و بازنمایی(Expressive Power and Representation): با ترکیب خروجی های سرهای توجه متعدد، مدل می تواند نمایش گویاتر و جامع تری از توالی ورودی ایجاد کند. این نمایش ترکیبی می‌تواند تفاوت‌های ظریف و پیچیدگی‌های زبان طبیعی را بهتر به تصویر بکشد، که منجر به بهبود عملکرد در کارهای مختلف پایین‌دستی می‌شود.
  4. استحکام و تعمیم(Robustness and Generalization): با سرهای توجه متعدد، مدل قوی تر می شود و می تواند بهتر به ورودی ها و وظایف مختلف تعمیم دهد. اگر یک سر توجه نتواند رابطه یا جنبه خاصی را به تصویر بکشد، سرهای دیگر می توانند آن را جبران کنند و آسیب پذیری مدل را نسبت به حالت های خرابی خاص کاهش دهند.
  5. تفسیرپذیری(Interpretability): توجه چند سر می‌تواند بینش‌هایی را در مورد فرآیند تصمیم‌گیری مدل با اجازه دادن به محققان برای تجزیه و تحلیل انواع مختلف روابط جذب شده توسط هر سر توجه ارائه دهد. این می تواند به درک رفتار مدل کمک کند و به طور بالقوه منجر به بهبود یا تعدیل شود. به عنوان مثال، در POS، برخی از سران ممکن است در شناسایی افعال تخصص داشته باشند، در حالی که دیگران ممکن است بر سرنخ‌های زمینه‌ای که صفت‌ها یا قید‌ها را نشان می‌دهند تمرکز کنند.
  6. ارائه افزونگی(Providing Redundancy): استفاده از هدهای متعدد افزونگی را فراهم می کند، که در سناریوهایی که بخش های خاصی از داده ها نویز دارند یا از دست رفته اند کمک می کند. هدهای مختلف می توانند بر روی بخش های مختلف داده تمرکز کنند و اطمینان حاصل کنند که اطلاعات مهم نادیده گرفته نمی شوند.
  7. مدیریت وابستگی های دوربرد(Handling Long-Range Dependencies): هدهای متعدد به مدل اجازه می دهد تا وابستگی های دوربرد را به طور موثرتری در نظر بگیرد. در جمله « The quick brown fox jumps over the lazy dog.»، درک اینکه «fox» فاعل است و «jumps» فعل است، ممکن است نیاز به در نظر گرفتن کلماتی باشد که در دنباله از هم فاصله دارند. پس داشتن چندین سر به انجام وظیفه POS کمک شایانی می کند.

تجزیه و تحلیل مثالی با BERT-Base-Cased در تسک NER

با استفاده از ابزار BERT Explorer (https://huggingface.co/spaces/exbert-project/exbert)، میخواهیم visualize کنیم که چگونه چندین سر به بخش‌های مختلف جمله نمونه توجه می‌کنند.

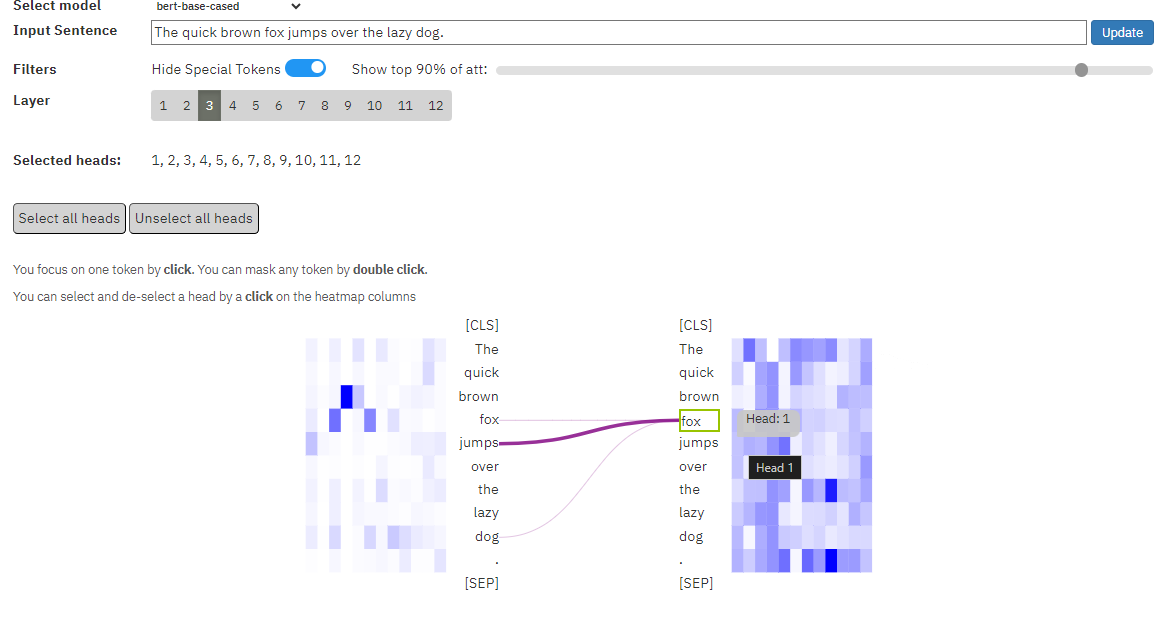
مراحل visualize:

1. مدل را بارگذاری میکنیم: مدل "bert-base-cased" را در ابزار BERT Explorer انتخاب میکنیم.
2. جمله را وارد میکنیم: جمله " The quick brown fox jumps over the lazy dog." را وارد میکنیم.
3. سرهای توجه را تجزیه و تحلیل میکنیم: مشاهده میکنیم که چگونه سرهای توجه مختلف بر روی نشانه های مختلف در جمله تمرکز می کنند.

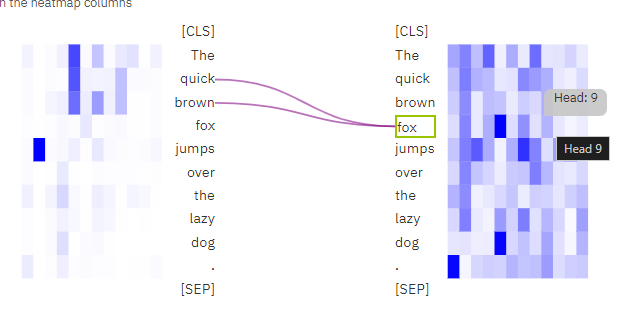


تفسیر سر های مختلف:

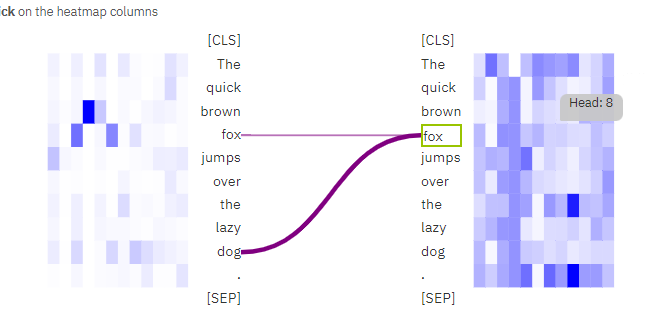
* **Head 1**: بر رابطه بین "fox" و "jumps" تمرکز میکند و رابطه فاعل و فعل را به تصویر می کشد.



* **Head 9**: به "quick" و "brown"، درک صفت های تغییر دهنده "fox" توجه میکند.



* **Head 8**: به رابطه معنایی بین «fox» و «dog» تمرکز میکند و ارتباط آنها و دنبال کردن سگ توسط روباه را درک میکند.



با استفاده از چندین سر(multiple heads)، مکانیزم توجه در BERT می تواند درک جامعی از ساختار جمله و روابط بین کلمات ایجاد کند، که برای تسک هایی مانند POS Tagging بسیار مهم است. این مکانیسم توجه چند سری به BERT اجازه می دهد تا با استفاده از نقاط قوت سرهای توجه مختلف برای به دست آوردن طیف گسترده ای از ویژگی ها و وابستگی های زبانی در وظایف مختلف NLP برتری یابد.

مراجع:

<https://chat.openai.com/>   
[https://gemini.google.com/](https://bard.google.com/)   
<https://claude.ai/chats>

## سوال نهم

مراحل کلیدی که من انجام داده ام عبارتند از:

1. توکنایزر DistilBERT را لود کردم.
2. تابع preprocess\_function را برای مدیریت قالب‌بندی مجموعه داده SQuAD کامل کردم.
3. یک نمونه از DefaultDataCollator ایجاد کردم.
4. مدل DistilBERT از پیش آموزش دیده را برای تسک پاسخگویی به سؤال دالود کردم.
5. TrainingArguments را راه اندازی کردم و یک نمونه از Trainer ایجاد کردم.
6. مدل را آموزش(fine-tune) دادم و همراه با توکنایزر ذخیره کردم.
7. یک نمونه سوال و context را برای استنباط ارائه کردم.
8. با استفاده از مدل آموزش دیده یک pipeline پاسخگویی به سؤال ایجاد کردم.
9. نحوه اجرای دستی استنتاج با استفاده از مدل و توکنایزر را تست کردم.

بخش‌های اصلی که شامل پر کردن بخش‌های # TODO، بارگذاری مدل و توکنایزر از پیش آموزش‌دیده، ایجاد DefaultDataCollator ، راه‌اندازی Trainer، و ارائه نمونه سوال/زمینه تستی برای استنتاج بود.

توضیحات بیشتر و با جزییات کامل تر در خصوص کارهایی که انجام دادم.

1. **Loaded the DistilBERT tokenizer**:

tokenizer = AutoTokenizer.from\_pretrained("distilbert-base-uncased")

The AutoTokenizer class from the Transformers library allows us to load a pre-trained tokenizer for a specific model. In this case, we're loading the tokenizer for the distilbert-base-uncased model, which will be used to tokenize the input text for the model.

1. **Defined the preprocess\_function**: This function handles the specific preprocessing requirements for the SQuAD dataset, which is used for the Question Answering task. The key steps in this function are:
   * Tokenize the question and context text using the tokenizer.
   * Truncate the context if it exceeds the maximum input length.
   * Map the start and end positions of the answer in the context text.
   * Handle cases where the answer is not fully contained in the context.
   * Add the start and end positions as input features for the model.
2. **Created a DefaultDataCollator instance**:

data\_collator = DefaultDataCollator()

The DefaultDataCollator is used to batch and collate the preprocessed data into a format that can be fed into the model during training and evaluation.

1. **Loaded the DistilBERT model for Question Answering task**:

model = AutoModelForQuestionAnswering.from\_pretrained("distilbert-base-uncased")

The AutoModelForQuestionAnswering class from the Transformers library allows us to load a pre-trained model for the Question Answering task. We're loading the distilbert-base-uncased model here.

1. **Set up the TrainingArguments and created a Trainer instance**:

training\_args = TrainingArguments(...)

trainer = Trainer(model=model, args=training\_args, train\_dataset=tokenized\_squad["train"], ...)

The TrainingArguments class is used to specify various hyperparameters and settings for the training process, such as the learning rate, batch size, number of epochs, and more.

The Trainer class is a high-level API that handles the training and evaluation of the model. We pass in the model, training arguments, datasets, tokenizer, and data collator to set up the training process.

1. **Trained the model and saved it along with the tokenizer**:

trainer.train()

trainer.save\_model("qa\_model")

tokenizer.save\_pretrained("qa\_model")

The trainer.train() method starts the training process using the specified configuration. Once the training is complete, we save the trained model and tokenizer to disk for later use.

1. **Provided a sample question and context for inference**: We define a sample question and context text to test the trained model during inference.
2. **Created a Question Answering pipeline using the trained model**:

question\_answerer = pipeline("question-answering", model="qa\_model")

The pipeline function from the Transformers library provides a convenient way to perform inference tasks like Question Answering. We pass in the task name ("question-answering") and the path to our trained model ("qa\_model").

1. **Showed how to manually run inference using the model and tokenizer**: In this part, we demonstrate how to perform inference manually by tokenizing the input text, passing it through the model, and decoding the predicted answer tokens.

Throughout the notebook, we leverage the Transformers library and the Datasets library from Hugging Face to load data, preprocess it, train a model, and perform inference. The key steps involve data preparation, model loading, training configuration, and inference using the trained model.

مراجع:

<https://chat.openai.com/>   
[https://gemini.google.com/](https://bard.google.com/)   
<https://claude.ai/chats>

پایان