
ChatGPTは
どう変わろうとしているのか？
マルチモーダル化とカスタム化と
AIの近未来像を中心に

はじめに

ChatGPTの登場から、約一年が過ぎようとしています。その衝撃が冷めやらぬ今また、AIの世界に大きな変化が起きようとしています。

まず、11月6日のOpenAI DevDayの発表は、ChatGPTの技術的な変化の方向を示しました。

同時に、11月17日のAltmanのCEO解任とその後の混乱は、OpenAIの内部に、OpenAIが進むべき進路をめぐって対立があることを内外に示しました。

今回のセミナーの特徴について

今回の丸山のセミナーは、基本的には、こうした事態の進行に触発されたものです。

今回のセミナーは三つの特徴を持っています。

大規模言語モデルの進化を振り返る

第一の特徴は、今回のセミナーは、OpenAIのAI技術の現在を、大規模言語モデルの技術の進化の中で、位置付けようとしていることです。

これからもAI技術の進化は続くと思います。それは、それまでの達成と問題を踏まえ次の変化を生み出すというドラスティックなものになるでしょう。その中で、現在の変化をもたらしたものを、その基礎から歴史的に振り返ることは重要なことだと僕は考えています。

今回のセミナーは、AI技術の変化を振り返り、それを基礎から捉え返す機会になると思います。

AIのマルチモーダル化とカスタム化に注目する

第二の特徴は、OpenAIのDevDayの発表には、非常に多岐にわたる内容が含まれているのですが、今回のセミナーは、その中で二つのことにフォーカスしようとしています。

一つは「AIのマルチモーダル化」で、もう一つは「AIがカスタマイズ可能になること」です。この二つは、AIと人間の未来での関係を考える上で、重要なステップになっていくと僕は考えています。

近未来のAIが進むべき道について

第三の特徴は、この間OpenAIの内部で顕在化したように見える意見の相違を、ひとつのトピックスとして取り上げていることです。

この点では、客観的な事実経過を可能な限り明らかにしようとしましたが、最終章で示されているのは、丸山自身が近未来のAIにどのような展望を持っているのかという個人的な展望です。

セミナーの構成について

第一部　この一年の間にAIの世界で起きたこと

第二部　大規模言語モデルの成立とChatGPTの成功の背景

- Attentionメカニズムと Google機械翻訳
- TransformerとBERT
- ChatGPTの成功と「人間のフィードバックからの強化学習」の導入
- AIの危険性の認識とModel Refusalという手法

第三部　AIのマルチモーダル化とカスタム化

- Google Vision Transformer
- OpenAI CLIP
- AI Assistantアプリ
- Assistant API

第四部　近未来のAIの展望

質疑応答

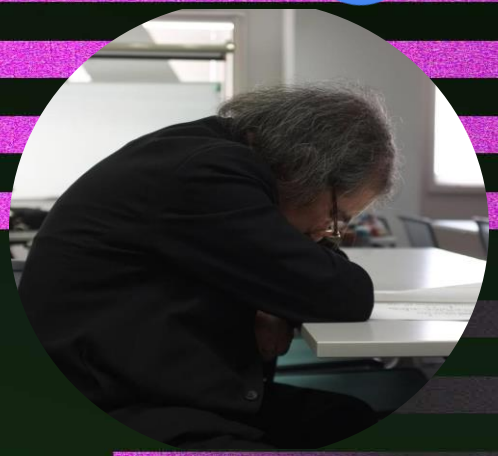
- 問題提起と司会　角川アスキー総研　遠藤諭 氏

未来展望について

AIの未来展望については、非常に楽観的なものですが、11月23日に開催される日本Androidの会のイベントABC Autumn での丸山の基調講演「Be My AI !」でお話させていただきます。

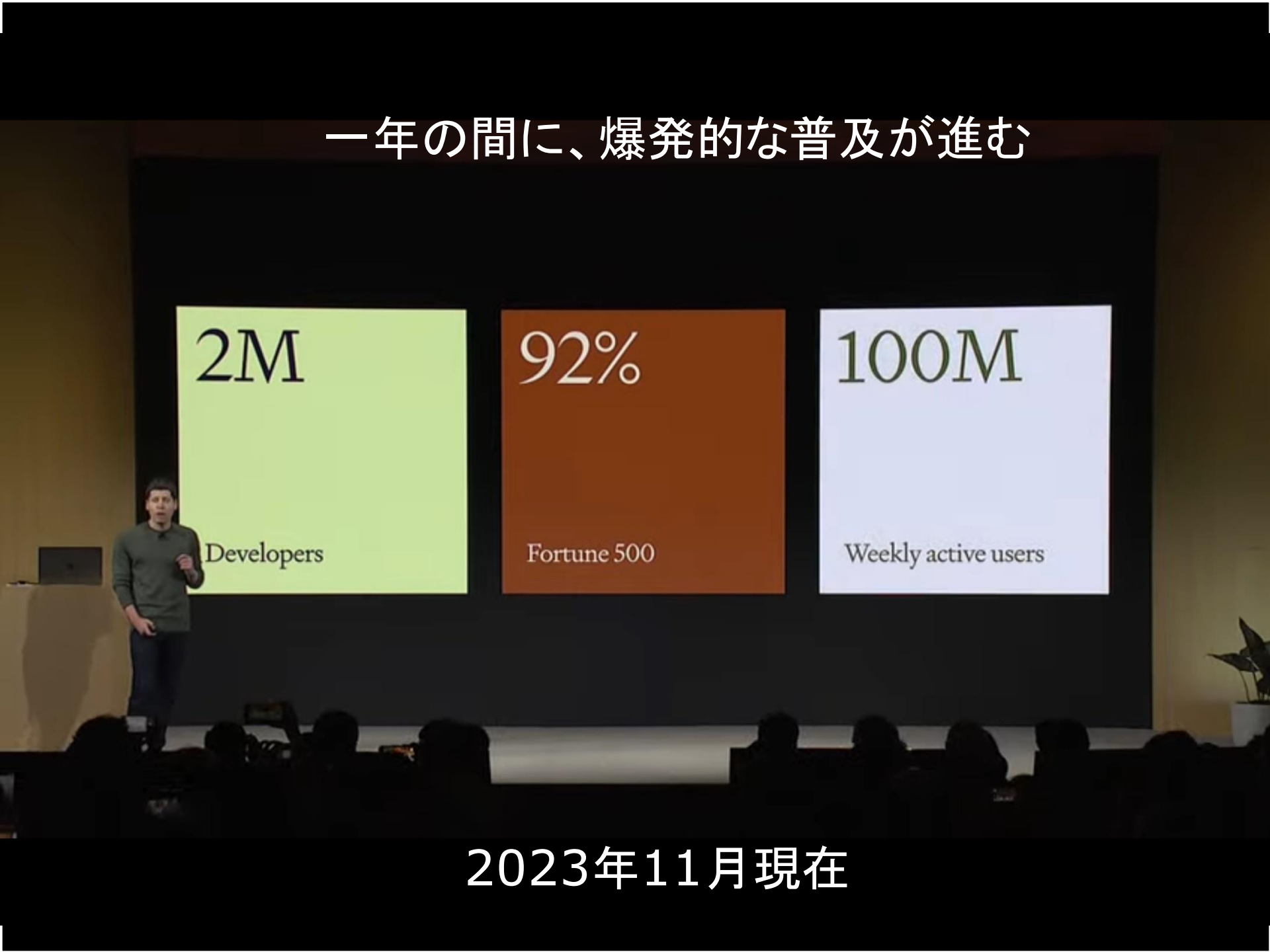

この一年の間に
AIの世界で起きたこと

ChatGPTの登場



2022/11/30

一年の間に、爆発的な普及が進む

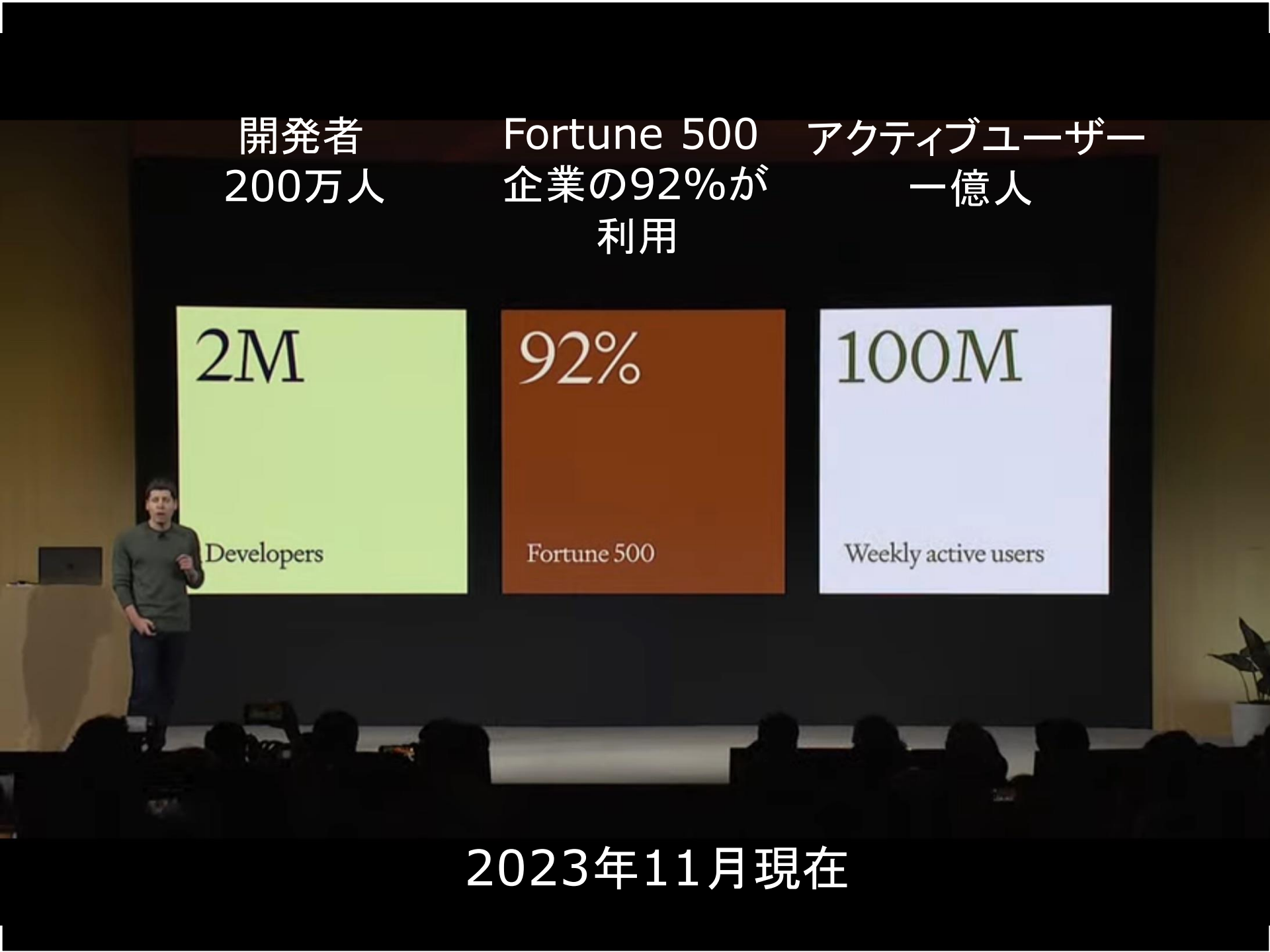


2023年11月現在

開発者
200万人

Fortune 500
企業の92%が
利用

アクティブユーザー
一億人



2023年11月現在

「AIの父」ヒントン、Googleを辞める

2023/05/01

OpenAIが、危惧していること

GPT-4 で観察された
安全性への挑戦

- 幻覚
- 有害コンテンツ
- 悪意のある表現
- 偽情報と影響力操作
- 過信
- ・・・

2023/05/23

OpenAI GPT-4 System Card 論文

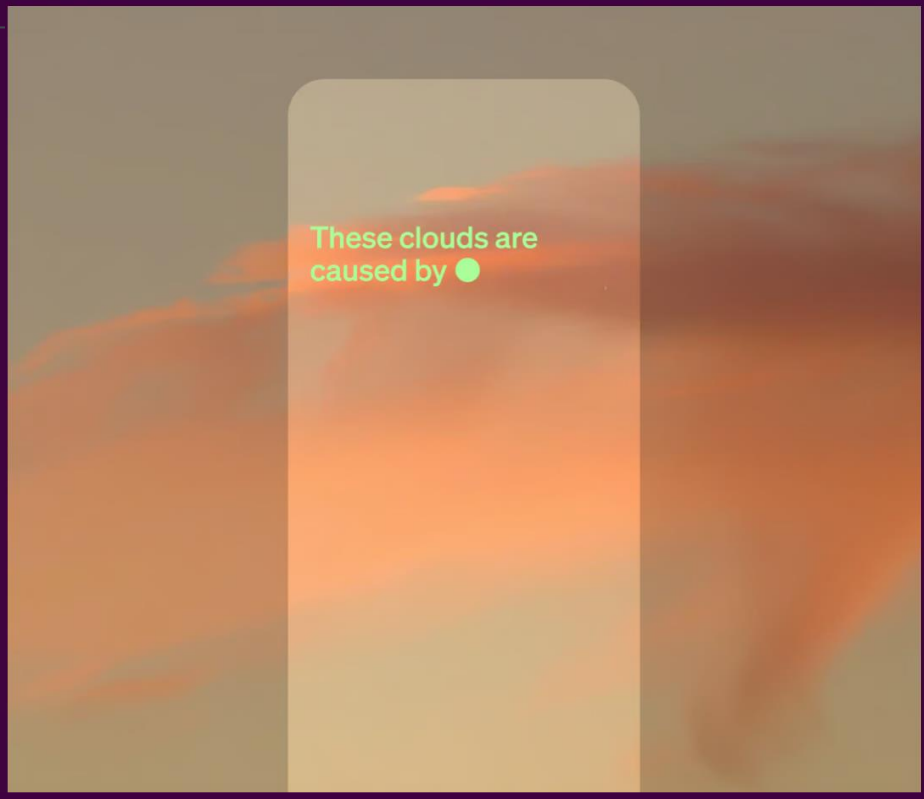
AIは、マルチモーダルへ

ChatGPTは、いまや、見ることも聞くことも話すこともできる

2023/09/25

ChatGPT can now see, hear, and speak

We are beginning to roll out new voice and image capabilities in ChatGPT. They offer a new, more intuitive type of interface by allowing you to have a voice conversation or show ChatGPT what you're talking about.

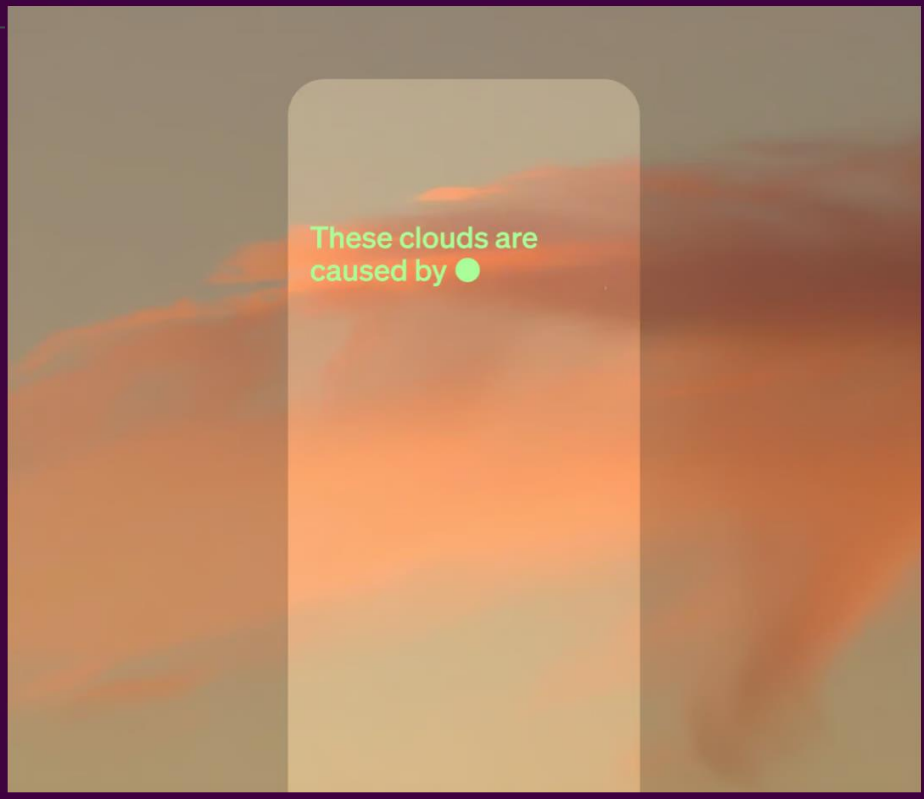


ChatGPTでは、新しい音声と画像機能を提供し始めています。
音声で会話したり、話している内容をChatGPTに見せることで、
より直感的な新しいタイプのインターフェイスを提供します。

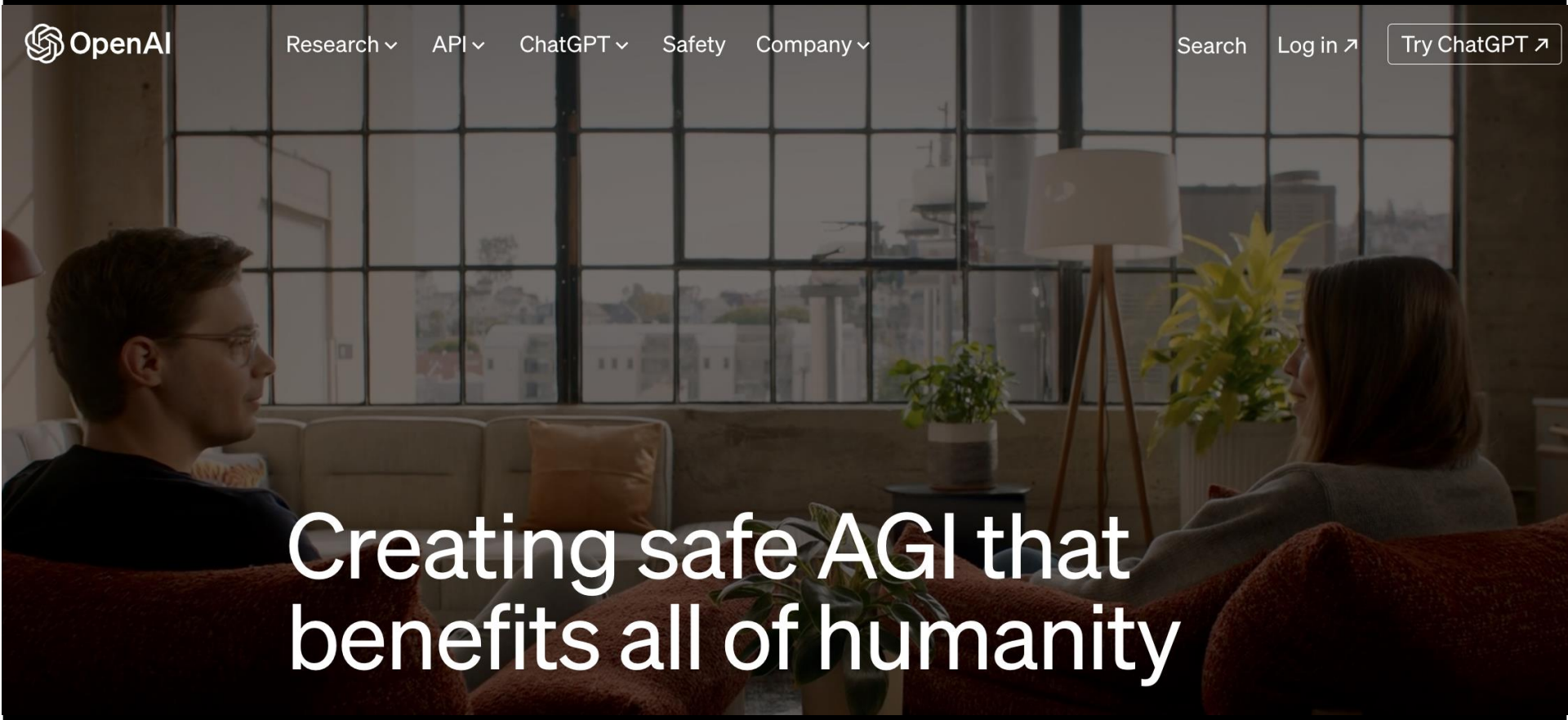
2023/09/25

ChatGPT can now see, hear, and speak

We are beginning to roll out new voice and image capabilities in ChatGPT. They offer a new, more intuitive type of interface by allowing you to have a voice conversation or show ChatGPT what you're talking about.



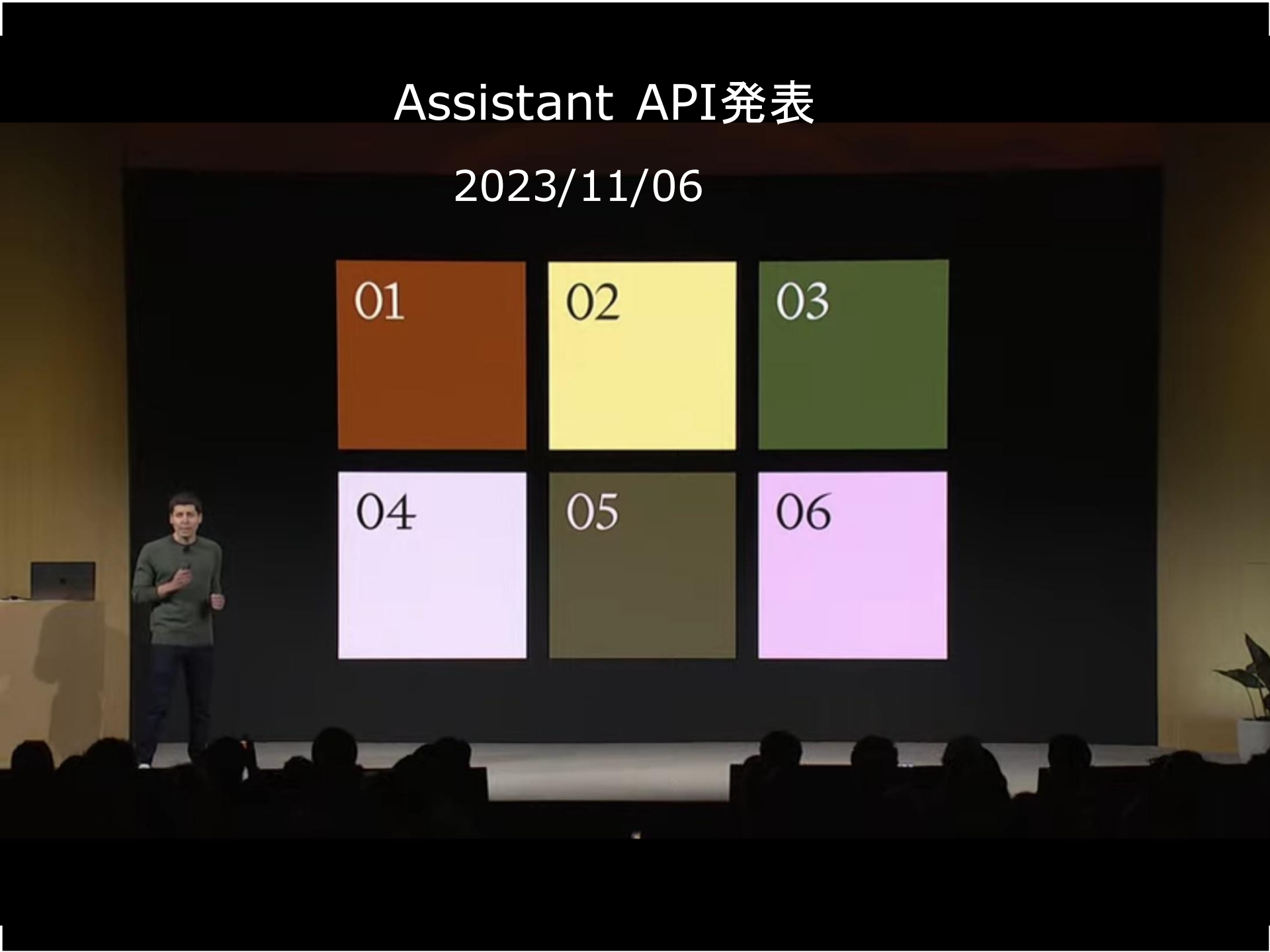
2023/10/23
OpenAIのトップページ



全人類の利益になる安全な汎用の人工知能を創造する

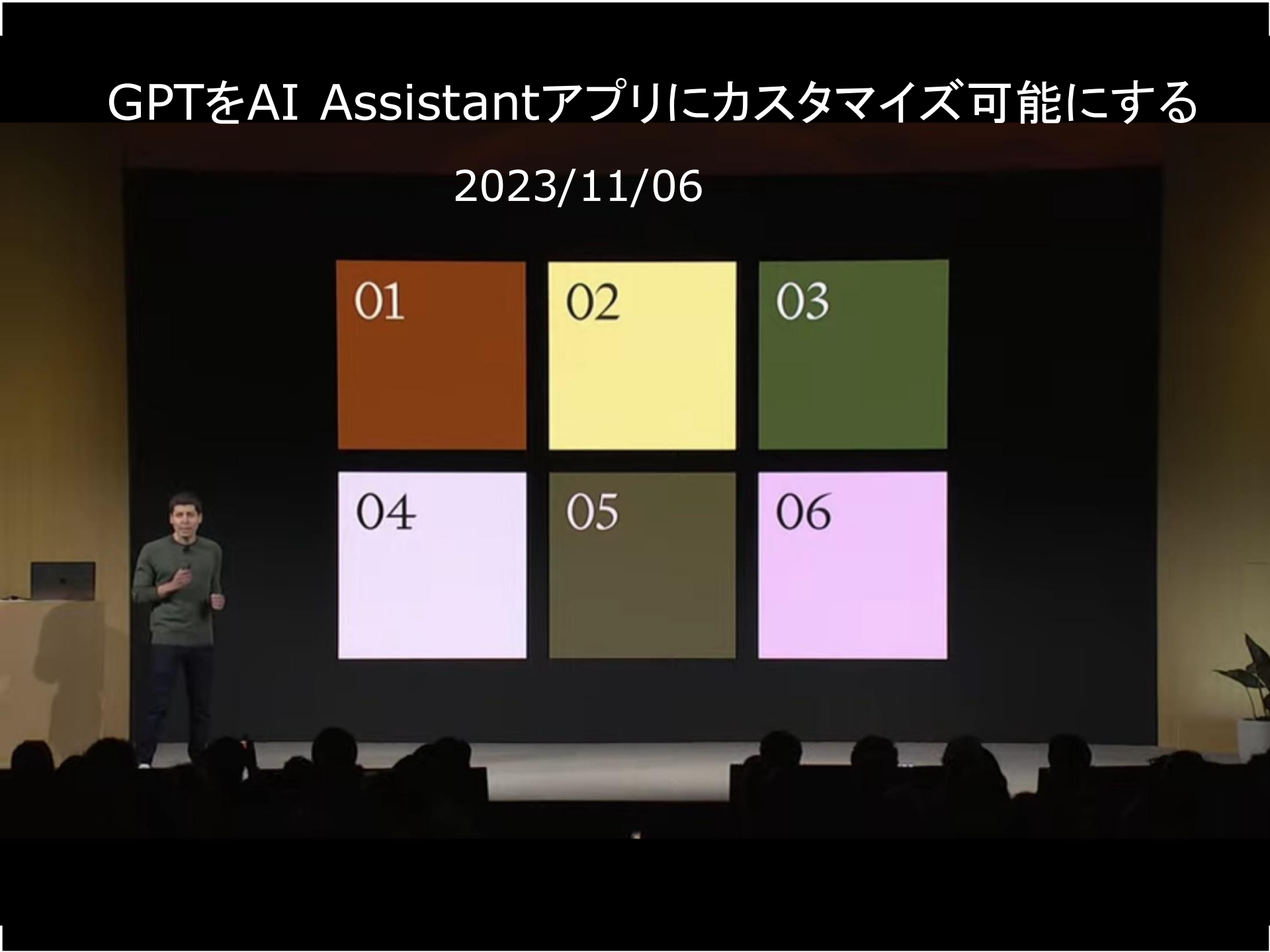
Assistant API発表

2023/11/06

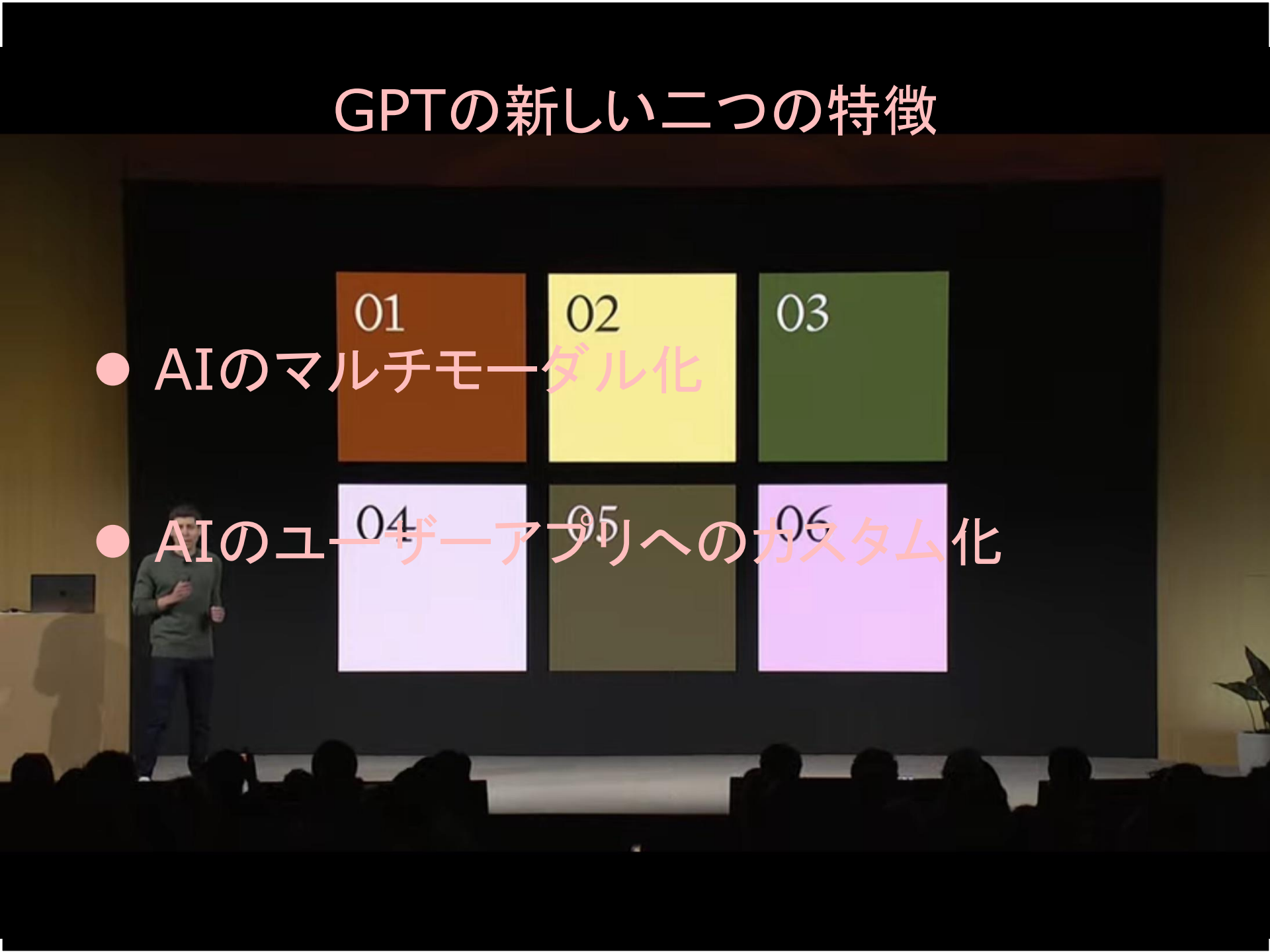


GPTをAI Assistantアプリにカスタマイズ可能にする

2023/11/06



GPTの新しい二つの特徴



- AIのマルチモーダル化
- AIのユーザーアプリへのカスタム化

GPTの新しい二つの特徴

- AIのマルチモーダル化
- AIのユーザーアプリへのカスタム化


2023/11/17
Altman 解任
2023/11/19
Microsoftへ？


OpenAI について

Google

Larry Page on AI

Larry Page on AI

「Googleの検索エンジンが、AIによって完全なものになったときにのみ、Googleのミッションは、完遂されるだろう。あなたたちは、それが何を意味するのか知っている。それが人工知能なのだ。」

「人工知能は、Googleの最終バージョンになるだろう。Web上のすべてのものを理解するだろう究極の検索エンジンは、あなたが望むものを正確に理解するだろうし、あなたに正しいものを与えるだろう。我々は、今は、そうしたことをするのには、遠いところにいる。ただ、我々は、少しずつ、それに近ずくことはできる。我々が取り組んでいることは、基本的には、そのことなのだ。」

http://goo.gl/OEL1oC

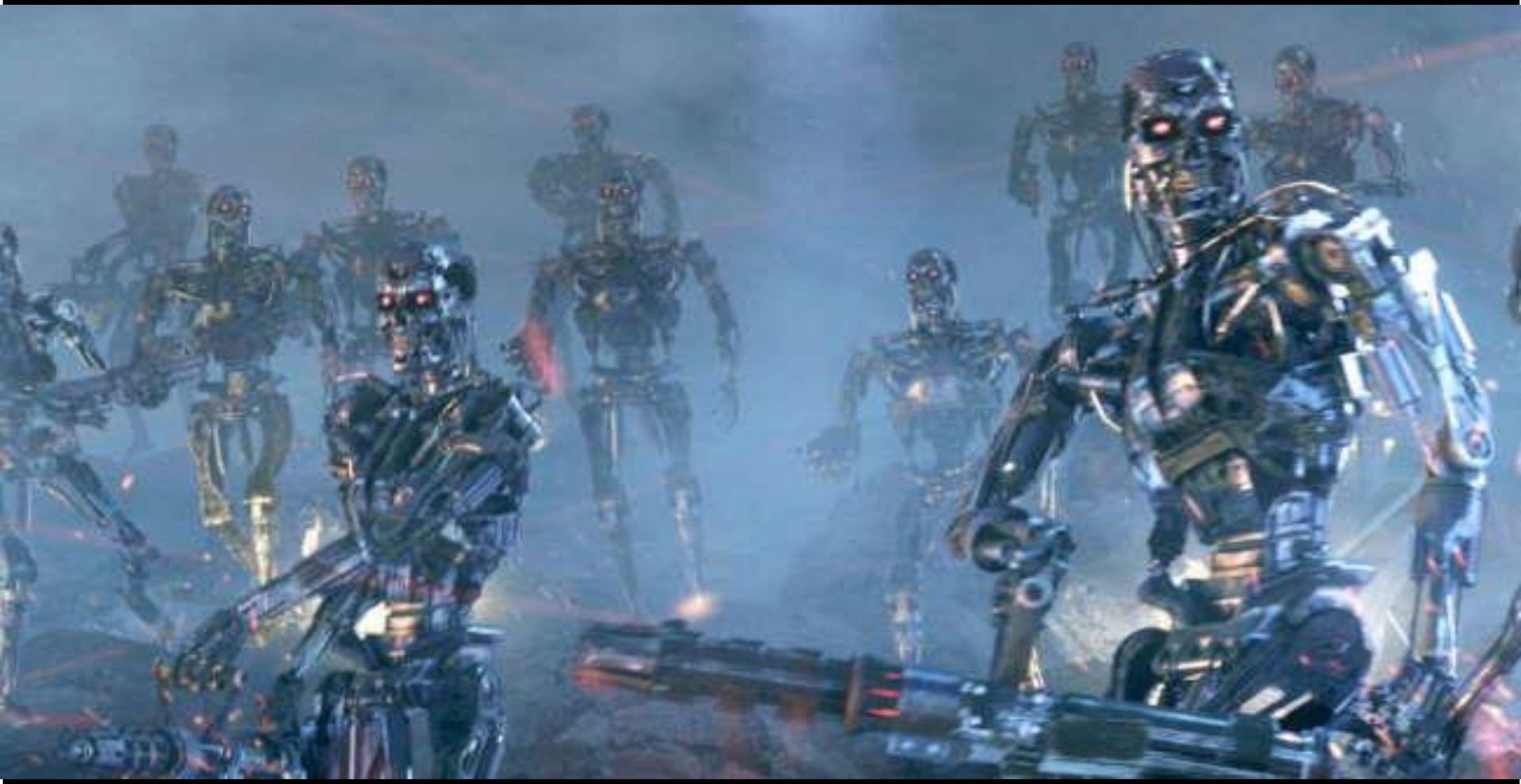
Larry Page on AI

「検索における我々の大きな目標は、人が望むものを、実際に正確に理解し、世界のすべてのものを理解することである。コンピューター科学者として、我々は、それを人工知能と呼ぶ。」

http://goo.gl/OEL1oC

“Elon Musk lives in fear of Google's killer robot army”

“Elon Musk lives in fear of Google's killer robot army”



http://goo.gl/LdHPZA

Musk is genuinely worried that Page might just lead to the destruction of humanity as we know it.

"I'm really worried about this,"

"This," refers to the possibility that Page would develop artificially-intelligent robots that could turn evil and have the ability to annihilate the human race.

Page may be well-meaning, but as Musk says, "He could produce something evil by accident."

OpenAI Structure
https://openai.com/our-structure

当初のOpenAI Nonprofitを設立してから約3年後の2019年に、私たちは「上限利益」構造を発表しました。

設立当初から私たちは、AGI（経済的に価値のある仕事において人間を凌駕する高度に自律的なシステム）を頂点とする強力なAIは、安全に対処すべきリスクとともに、社会を再構築し、多大な利益をもたらす可能性を秘めていると信じてきました。

現在のシステムの能力が高まっていることは、OpenAIや他のAI企業にとって、それぞれの使命と運営の中核となる原則、経済メカニズム、ガバナンスモデルを共有することがこれまで以上に重要であることを意味します。

Overview

私たちは、人類の利益のために安全で有益な人工知能を構築することを目標に、2015年後半に非営利団体OpenAIを設立しました。このようなプロジェクトは、以前であれば1つまたは複数の政府によるものであり、人類にとって広範な利益を追求する人類規模の取り組みであったかもしれません。

公共部門には明確な道筋がないと考え、また民間企業における他の野心的なプロジェクト（スペースX社、クルーズ社など）の成功を考慮し、私たちは公益への強いコミットメントに縛られた民間の手段でこのプロジェクトを推進することにしました。私たちは当初、501(c)(3)が、利益インセンティブに邪魔されることなく、安全で広く有益なAGIの開発を指揮する最も効果的な手段であると考えていました。私たちは、そうすることが安全であり、公共の利益になると思われる場合には、私たちの研究とデータを公表することを約束しました。

- OpenAIの非営利団体はそのまま残り、その理事会はOpenAIのすべての活動の統括団体として継続する。
- 新しい営利目的の子会社が設立され、資本を調達するために株式を発行し、ワールドクラスの人材を雇用することができますが、非営利団体の指示に従います。営利目的の取り組みに従事していた従業員は、新しい子会社に移行された。
- 営利企業はNPOの使命を追求する法的義務を負い、研究、開発、商業化、その他の中核業務に従事することでその使命を遂行する。OpenAIの指導原則である安全性と広範な利益は、そのアプローチにおいて中心的なものとなる。
- 営利目的の株式構造では、純粋な利益最大化に焦点を当てるのではなく、商業性と安全性および持続可能性のバランスをとった方法でAGIを研究、開発、展開するインセンティブを与えるため、投資家と従業員への最大財務リターンを制限する上限を設ける。

- NPOは、自らの運営に加え、理事会を通じてそのような活動すべてを管理・監督する。また、包括的なベーシックインカム研究の後援、経済効果研究の支援、OpenAI Scholarsのような教育中心のプログラムの試行など、幅広い慈善活動を継続する。NPOは長年にわたり、スタンフォード大学人工知能インデックス・ファンド、ブラック・ガールズ・コード、ACLU財団など、テクノロジー、経済効果、正義に焦点を当てた他の公益団体も数多く支援してきた。

そうすることで、非営利団体は私たちの組織の中心であり続け、AGIの開発をコントロールし、営利団体は、OpenAIのコアミッションを追求する義務を負いながら、これを達成するためのリソースを結集する任務を負うことになります。ミッションが何よりも優先されることは、すべての投資家と従業員が従う営利企業の運営契約書に明記されています：

IMPORTANT

****Investing in OpenAI Global, LLC is a *high-risk investment*****
****Investors could lose their capital contribution and not see any return****
****It would be wise to view any investment in OpenAI Global, LLC in the spirit of a donation, with the understanding that it may be difficult to know what role money will play in a post-AGI world****

The Company exists to advance OpenAI, Inc.'s mission of ensuring that safe artificial general intelligence is developed and benefits all of humanity. The Company's duty to this mission and the principles advanced in the OpenAI, Inc. Charter take precedence over any obligation to generate a profit. The Company may never make a profit, and the Company is under no obligation to do so. The Company is free to re-invest any or all of the Company's cash flow into research and development activities and/or related expenses without any obligation to the Members. See Section 6.4 for additional details.

Brief departure of Altman and Brockman
2023年11月17日

2023年11月17日、サム・アルトマンは取締役会（ヘレン・トナー、イリヤ・スーツキーヴァー、アダム・ダンジェロ、ターシャ・マッコーリーで構成）の不信任に基づきCEOを解任され、最高技術責任者のミラ・ムラーティが暫定CEOに就任した。OpenAIの社長であったグレッグ・ブロックマンは取締役会の会長から解任された[63][64]。ブロックマンはこの発表の直後に同社の社長職を辞任し、彼が辞める前に起こった出来事のいくつかの詳細を報告した[65][66]。これに続いて、OpenAIの3人の上級研究者が辞任した。研究ディレクター兼GPT-4リードのヤクブ・パチョッキ、AIリスク責任者のアレクサンダー・マドリー、研究者のシモン・シドーである[67][68]。

https://en.wikipedia.org/wiki/OpenAI

2023年11月18日

2023年11月18日、Altmanの退任を非難したMicrosoftやThrive Capitalなどの投資家が取締役会に圧力をかける中、AltmanがCEOに復帰する話があったと報じられた[69]。Altman自身はOpenAIに復帰することに賛成であると話したが、話がうまくいかなければ新会社を立ち上げ、OpenAIの元従業員を連れてくることを考えていると述べた[70]。[70]アルトマンが復帰する場合、取締役会のメンバーは「原則的に」会社を辞職することで合意した[71]。2023年11月19日、アルトマンとの復帰交渉は失敗し、ムラーティはエメット・シアーに代わり暫定CEOに就任した[72]。取締役会は当初、アルトマンの後任としてOpenAIの元幹部であるAnthropicのCEOダリオ・アモデイに接触し、合併を提案したが、両方の申し出は断られた[73]。

2023年11月20日

2023年11月20日、マイクロソフトのサティア・ナデラCEOは、アルトマンとブロックマンが高度AIに関する新しい研究チームを率いるために同社に入社することを発表し、このような事態に陥ったにもかかわらず、彼らはOpenAIに引き続きコミットしていると述べた[74]。[75]OpenAIの770人の従業員のうち、MuratiとSutskeverを含む約738人が、取締役会がAltmanをCEOとして再雇用し、その後辞任しないのであれば、仕事を辞めてMicrosoftに入社するという公開書簡に署名した[76][77]。投資家は、潜在的な大量辞任とAltmanの解任を受けて、取締役会メンバーに対して法的措置を取ることを検討している[78]。これに対してOpenAIの経営陣は、Altmanと取締役会との交渉が再び進行中であり、しばらく時間がかかるという社内メモを従業員に送った[79]。

2023年11月21日

2023年11月21日、継続的な交渉の後、AltmanとBrockmanは以前の役割のまま会社に戻り、Bret Taylor（会長）とLawrence Summersで構成される新メンバーとD'Angeloが残る取締役会が再構築された[80]。2023年11月22日、Sam AltmanのOpenAIからの解雇は、組織の極秘プロジェクトQ*における重要なブレークスルーを誤って処理した疑惑に関連している可能性があることを示唆する報道がなされた。OpenAIの情報筋によると、プロジェクトQ*は論理的推論と定理証明におけるAI能力の開発を目的としている。この開発に対するアルトマンの対応、特に発見の潜在的な安全性への影響に関する懸念は、彼が解雇される直前に会社の取締役会に提起されたと伝えられている[81][82]。

Greg Brockman on X

サムと私は、今日の取締役会の決定にショックを受け、悲しんでいます。

まず最初に、OpenAIで一緒に働いてきたすべての素晴らしい人々、顧客、投資家、そして手を差し伸べてくれたすべての人々に感謝します。

私たちも、何が起こったのかを正確に把握しようとしています。わかっていることは以下の通りです：

- 昨晩、サムはイリヤから金曜日の正午に話をしようというメールを受け取った。サムはグーグルミートに参加し、グレッグを除く役員全員がそこにいた。イリヤはサムに、自分が解雇されること、そしてその知らせがもうすぐ出ることを告げた。

- 午後12時19分、グレッグはイリヤから急ぎの電話を求めるメールを受け取った。12時23分、イリヤはグーグルミートのリンクを送った。グレッグは、自分が取締役会から外されること（しかし、会社にとって不可欠な存在であり、その役割は維持される）、そしてサムが解雇されたことを告げられた。同じ頃、OpenAIはブログ記事を公開した。

- 私たちの知る限り、前日の夜に知ったミラ以外の経営陣は、まもなくこのことを知らされた。

ありがとうございます。でも、どうか心配しないでください。私たちは大丈夫です。もうすぐもっと大きなことが起こる。

To the Board of Directors at OpenAI

「OpenAIは世界をリードするAI企業です。私たちOpenAIの従業員は、最高のモデルを開発し、この分野を新たなフロンティアへと押し進めてきました。AIの安全性とガバナンスに関する我々の仕事は、グローバルな規範を形成しています。私たちが構築した製品は、世界中の何百万人もの人々に利用されています。これまで、私たちが働き、大切にしている会社がこれほど強い立場にあったことはありません。

あなたがサム・アルトマンを解雇し、グレッグ・ブロックマンを取締役から解任したプロセスは、この仕事すべてを危険にさらし、私たちの使命と会社を弱体化させました。あなたの行為は、あなたにOpenAIを監督する能力がないことを明らかにしました。"

あなたの決断を私たち全員が予期せず知ったとき、OpenAIのリーダーシップチームは会社を安定させるために迅速に行動しました。彼らはあなたの懸念に注意深く耳を傾け、あらゆる理由であなたに協力しようとしました。あなたの主張に対する具体的な事実を何度も求めたにもかかわらず、あなたは一度も証拠書類を提出しませんでした。彼らはまた、あなたが職務を遂行する能力がなく、不誠実な交渉をしていることに次第に気づいていった。

リーダーシップ・チームは、当社の使命、会社、利害関係者、従業員、そして世間一般に最も貢献できる、最も安定した前進の道は、あなたが辞任し、会社を安定的に前進させることができる有能な取締役会を設置することであると提案しました。リーダーシップは24時間体制であなたと協力し、互いに合意できる結果を探しました。しかし、あなたは最初の決断から2日も経たないうちに、会社の最善の利益に反して再びミラ・ムラーティ暫定CEOを交代させた。あなたはまた、会社の破壊を許すことが "使命に合致する "とリーダーシップ・チームに伝えた。

あなたの行動は、あなたがOpenAIを監督する能力がないことを明白にしました。私たちは、私たちの使命と従業員に対する能力、判断力、配慮に欠ける人たちのために、あるいは一緒に働くことはできません。私たちは、OpenAIを辞職し、Sam AltmanとGreg Brockmanが経営する新しく発表されたマイクロソフトの子会社に参加することを選択するかもしれません。マイクロソフトは、私たちが参加することを選択した場合、この新しい子会社にOpenAIの全従業員のためのポジションがあることを保証してくれました。現取締役全員が辞任し、取締役会がBret TaylorやWill Hurdのような2人の新しい主席独立取締役を任命し、Sam AltmanとGreg Brockmanを復職させない限り、私たちは間もなくこのステップを踏むでしょう。

1. Mira Murati
2. Brad Lightcap
3. Jason Kwon
4. Wojciech Zaremba
5. Alec Radford
6. Anna Makanju
7. Bob McGrew
8. Srinivas Narayanan
9. Che Chang
10. Lillian Weng
11. Mark Chen
12. Ilya Sutskever"

Decoding Intentions
Helen Toner

政策立案者は、人工知能分野における意図をどのように信頼できる形で明らかにし、評価することができるのだろうか？

AI技術は急速に進化しており、民間や軍事への幅広い応用を可能にしている。AIの技術革新の大部分をリードしているのは民間企業だが、その動機やインセンティブは、本社を置く国家のそれとは異なる場合がある。政府と企業がより高性能なシステムの導入を競い合う中で、誤算や不注意によるエスカレーションのリスクは高まるだろう。

地政学的な競争が激化する中、安全で責任あるシステム開発を行うためには、誤解を防ぎ、明確な意思疎通を図るための政策手段をすべて理解することが不可欠である。

本ブリーフでは、これまであまり注目されてこなかった重要な政策的テコとして、「コストのかかるシグナル」を取り上げる。高価なシグナルとは、発信者が最初の約束や脅しを撤回したり履行しなかったりした場合に、政治的、評判的、金銭的な代償を支払うことになる発言や行動のことである。学術文献の再確認をもとに、4つのコスト・シグナル・メカニズムに注目し、それらをAIの分野に適用する（表1に要約）：

- 一方的なAI政策声明、多国間機関における議決、AIモデルのテストと評価のための公的コミットメントなどである；
- サンク・コストは、AIアルゴリズムに対するライセンスや登録の要件、テストベッドやその他の施設を含むテスト・評価インフラへの大規模な投資など、コストが最初から織り込まれているコミットメントに依存する；
- 割賦コストは、AIシステムの持続的検証技術や、データセンターにおけるAIチップの使用に関する会計ツールなど、送り手が現在ではなく将来に代償を支払うことになるコミットメントである；
- 還元可能なコストとは、前払いではあるが、シグナルを送る側の行動次第で時間をかけて相殺できるもので、より解釈しやすいAIモデルへの投資、AI投資基準の策定への参加表明、AI対応システムの代替設計原則などがある 。

我々は3つのケーススタディにおいて、AIの高価なシグナリングメカニズムを探求する。

最初のケーススタディでは、軍事AIと自律性をめぐるシグナリングについて考察する。

2つ目のケーススタディでは、人権、市民の自由、データ保護、プライバシーへのコミットメントをAI技術の設計、開発、導入に組み込む民主的AIをめぐる政府のシグナリングについて検討する。

3つ目のケーススタディは、大規模言語モデル（LM）の開発とリリースをめぐる民間セクターのシグナリングについて分析したものである。

しかし、その長所と限界を理解することが重要である。

キューバ危機の後、米国はモスクワと直接ホットラインを結び、そこからメッセージを送ることができた。

シグナルは、うっかりすると高くつくこともある。民主的AIに関する米国政府のシグナリングは、特定の価値観へのコミットメントについて強力なメッセージを送るが、こうした原則を共有しない可能性のあるパートナーとの間で違反が生じるリスクがあり、米国が偽善の容疑にさらされる可能性がある。

すべてのシグナルが意図的であるわけではないし、商業主体は他の分野や国の政府や業界関係者とは異なるコストを概念化するかもしれない。

このような複雑さは克服できないものではないが、次のような経済状況におけるシグナル伝達の課題となっている。

民間企業がイノベーションを推進し、その拠点となる国と利害が対立する可能性のある経済状況において、シグナリングに課題をもたらす。

誤認や不用意なエスカレーションのリスクを考えれば、官民のリーダーは、シグナルを首尾一貫した戦略に組み込むよう注意しなければならない。コストのかかるシグナルには、シグナル伝達を目的とした透明性と、プライバシーやセキュリティをめぐる規範との緊張関係など、管理すべきトレードオフが伴う。

政策立案者や技術指導者が、能力を「隠すか、明らかにするか」だけでなく、どのように明らかにするか、また、どのようなチャネルを通じて意図のメッセージを伝えるかも考慮することで、シグナル伝達の機会は信頼性をもって拡大する。

公共部門と民間部門のリーダーによる互換性のあるメッセージは、AIにおけるコミットメントの信頼性を高めることができるが、政府関係者が異なる技術分野にわたる能力を評価するための適切なコンテキストを欠いている場合、シグナルを誤解する可能性もある。

政策立案者は、想定を明確にし、エスカレーションのリスクを軽減し、危機時のコミュニケーションに関する共通の理解を深めるために、コストのかかるシグナルを卓上演習や同盟国や競合国との集中的な対話に取り入れることを検討すべきである。シグナルはノイズが多く、時には聴衆を混乱させることもあるが、それでも必要である。

Helon Tonerの論文

	Military AI and Autonomy	Democratic AI	Private Sector Signaling
Tying hands	一方的な政策声明を発表し、意図を伝える。 核の指揮統制に関する意思決定	AIを活用した民主主義社会を攻撃する敵対的な攻撃に対して、あらかじめ定義された行動をとることを約束することで、民主的なAIの原則を守る。	学習データ、モデルの性能、危険な能力に関する透明性など高度なAIモデルに関する重要な情報を公開する。
Sunk costs	訓練中および配備前のレッドチーム編成手順に投資し、AI対応兵器システムの帰属を容易にするエンブレムの使用を検討する。	AI技術が悪用されるシステミックリスクがある市場で事業を行う民間企業向けのデューデリジェンス指針を公表する。	信頼できるホスティングサービスと、テストベッドやその他の施設を含むテスト・評価インフラに投資する。
Installment costs	AI対応システムの持続的検証技術にコミットし、集中的なコンピュートアカウンティングのための取り決めを開発する。	AI監査人のための共通の認証基準、ツール、慣行を開発する。	リアルタイムのインシデント監視と、AI対応システムが関与するインシデントのデータ収集と分析に関する共通基準にコミットする。
Reducible costs	要件を設定し、解釈可能なAIモデルや代替設計原則に投資するインセンティブを設ける。	AIの安全性に関する研究や民主主義的価値を促進するプライバシー向上技術の開発に対する賞金コンテストを主催する。	AIの影響評価とAIシステムの内部監査結果を公表する。

https://cset.georgetown.edu/wp-content/uploads/CSET-Decoding-Intentions.pdf

OpenAI researchers warned board of AI breakthrough ahead of CEO ouster, sources say

11月22日 ロイター］ - OpenAIのサム・アルトマン最高経営責任者（CEO）が4日間にわたり追放されるのに先立ち、複数のスタッフ研究者が取締役会宛に、人類を脅かす可能性があるという強力な人工知能の発見を警告する書簡を書いたと、この件に詳しい2人の関係者がロイターに語った。

これまで報告されていなかった書簡とAIアルゴリズムは、取締役会がジェネレーティブAIの申し子であるアルトマンを更迭する前の重要な進展であった、と2人の関係者は語った。アルトマンが火曜日遅くに凱旋する前、700人以上の従業員が、解雇されたリーダーと連帯するために、退職して支援者であるマイクロソフト(MSFT.O)に加わると脅していた。

https://www.reuters.com/technology/sam-altmans-ouster-openai-was-precipitated-by-letter-board-about-ai-breakthrough-2023-11-22/

情報筋は、この書簡を、アルトマンの解雇につながった取締役会の不満の長いリストの中の一つの要因として挙げており、その中には、結果を理解する前に進歩を商業化することへの懸念も含まれていた。ロイターは書簡のコピーを再確認できなかった。書簡を書いたスタッフは、コメントを求めたが応じなかった。

ロイターの連絡を受け、コメントを拒否したオープンAIは、スタッフへの内部メッセージの中で、Q*と呼ばれるプロジェクトと、週末の出来事の前に取締役会に宛てた手紙を認めたと、ある関係者は述べた。OpenAIの広報担当者は、長年の幹部であるMira Muratiが送ったメッセージは、その正確性についてコメントすることなく、特定のメディア記事についてスタッフに注意を喚起するものだったと述べた。

オープンエイの一部の関係者は、Q*（Q-Starと発音）は人工知能（AGI）として知られるものを探求するスタートアップの突破口になると考えている、とロイターに語った。オープンAIはAGIを、経済的に価値のあるタスクのほとんどにおいて人間を凌駕する自律型システムと定義している。

膨大なコンピューティング・リソースを与えられた新モデルは、特定の数学的問題を解くことができたと、その人物は匿名を条件に語った。小学生レベルの数学しかできないが、このようなテストに合格したことで、研究者たちはQ*の将来の成功を非常に楽観視するようになったと、情報筋は語った。

ロイターは、研究者が主張するQ*の能力を独自に検証することはできなかった。

研究者たちは、数学がジェネレーティブAI開発のフロンティアだと考えている。現在のところ、生成AIは次の単語を統計的に予測することで、文章作成や言語翻訳を得意としている。しかし、正解がひとつしかない数学の能力を克服することは、AIが人間の知能に似たより高い推論能力を持つことを意味する。これは例えば、斬新な科学研究に応用できるとAI研究者は考えている。

限られた数の演算しかできない電卓とは異なり、AGIは一般化し、学習し、理解することができる。

情報筋によれば、理事会への書簡の中で、研究者たちはAIの能力と潜在的な危険性を指摘しているとのことである。コンピューター科学者たちの間では、高度に知的な機械がもたらす危険性、たとえば人類を滅亡させることが自分たちの利益になると判断する可能性について、長い間議論されてきた。

研究者たちは、複数の情報筋がその存在を確認した "AIサイエンティスト "チームによる研究も指摘している。このグループは、以前の「コード・ジェン」と「数学・ジェン」チームを統合して結成されたもので、既存のAIモデルをどのように最適化し、推論力を向上させ、最終的には科学的な仕事をこなすかを模索していると、関係者の一人は語っている。

アルトマンは、ChatGPTを史上最も急成長したソフトウェア・アプリケーションのひとつにする努力を率い、AGIに近づくために必要なマイクロソフト社からの投資とコンピューティング・リソースを引き出した。

アルトマンは先週、サンフランシスコで開催された世界的リーダーによるサミットで、今月のデモンストレーションで多数の新しいツールを発表し、大きな進歩が目前に迫っていることを示唆した。

「OpenAIの歴史の中で、これまで4回、最近ではここ数週間のことですが、私たちが無知のベールを押し戻し、発見のフロンティアを前進させるとき、私はその場に立ち会うことができました。

その翌日、取締役会はアルトマンを解雇した。

OpenAI made huge breakthrough before ousting Sam Altman, introducing Q*

OpenAIの共同設立者サム・アルトマンが更迭される前日、AIのリーダーは「人類を脅かす」可能性のある技術的ブレークスルーを行った。

https://www.tweaktown.com/news/94534/openai-made-huge-breakthrough-before-ousting-sam-altman-introducing/index.html

さて、今日は歴史上最も重要な日の一つとして知られることになるだろう…OpenAIがQ*（Q-Starと発音する）の導入でAIにブレークスルーをもたらし、それがOpenAIの共同設立者兼CEOのサム・アルトマンの解雇の背景にあることが判明した。

オープンエイの社内では、人工知能（AGI）の探求におけるこのAIのブレークスルーは、Q*による彼らの進歩の新たな一歩であると考えている。AGIは、科学者や研究者、そして特にOpenAIのようなAI企業が目指してきたポイントのひとつです。AGIシステムは人間よりも賢い…それはもう単なるチャットボットではなく、全く別のものなのだ。

ここで少しフリンジ思考になるが、このQ*がAIドラマの狂気と驚きの瞬間の背後にいるのだろうか？というのも、この時点で、目の前で繰り広げられている1つの大きな「陰謀」として、すべてが意味を持ち始めるからだ。

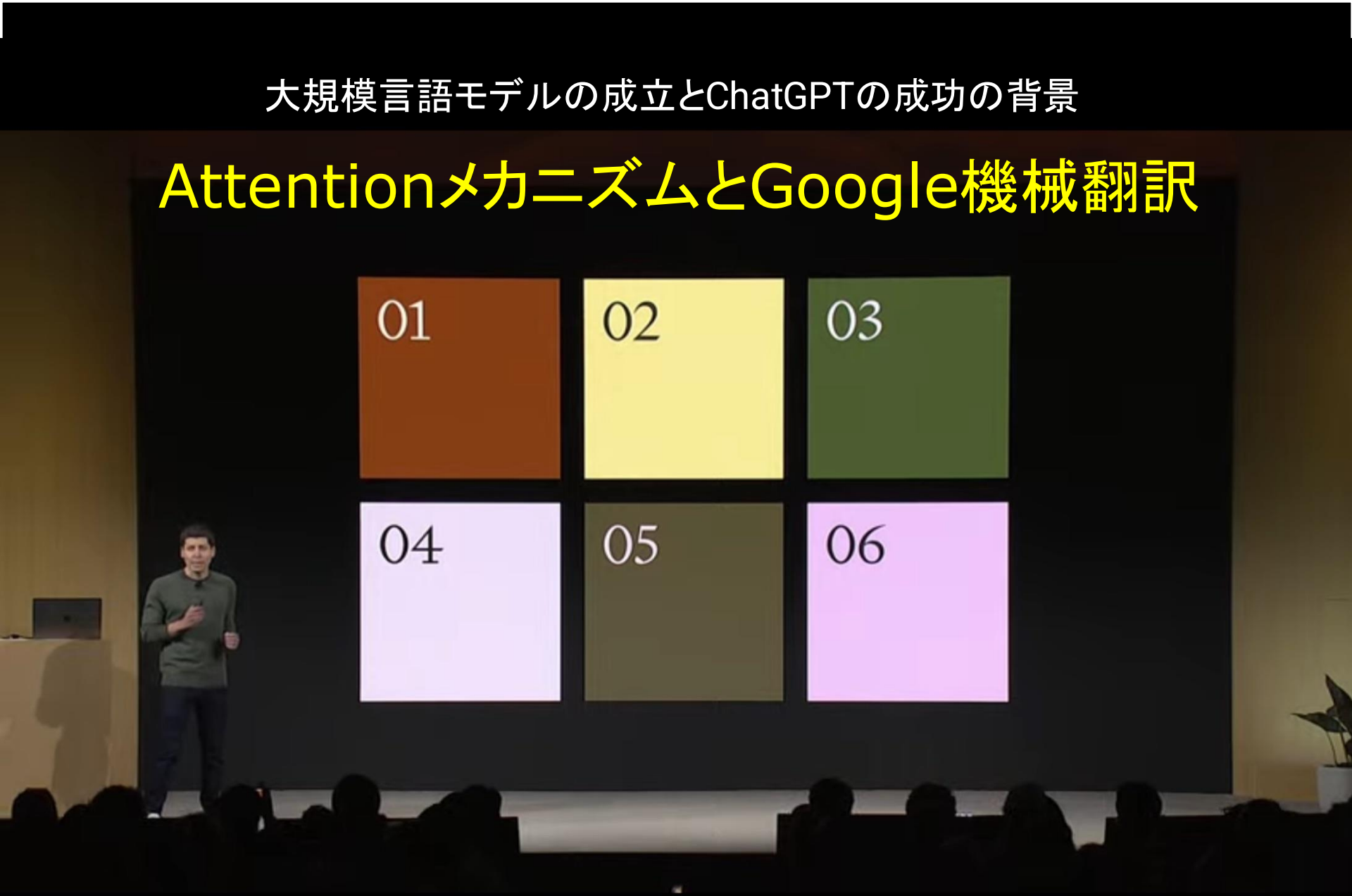
1点興味深いのは、アルトマンがOpenAIを解雇される数日前に、複数のスタッフ研究者が取締役会に対し、強力な人工知能の発見--AGIとこの新しいQ*--について警告する書簡を送っており、それが「人類を脅かす」可能性があると、この件に詳しい2人がロイターに語ったことだ。

アルトマンが解雇されたとき、マイクロソフトが彼を新しい高度AI研究チームに雇ったので、アルトマンにとっては素晴らしいニュースだった。OpenAIのスタッフは反旗を翻し、800人近いスタッフのうち700人以上がOpenAIの役員会を脅して、アルトマンや他のOpenAIの元スタッフをマイクロソフトに追随させようとした。アルトマンがOpenAIに戻り、Q*が導入された。


大規模言語モデルの成立と
ChatGPTの成功の背景

大規模言語モデルの成立とChatGPTの成功の背景

AttentionメカニズムとGoogle機械翻訳



11/24 角川セミナー

大規模言語モデルの成立期

このセッションでは、大規模言語モデルの成立期の話をしようと思います。まず、大まかな流れを見ておきましょう。

この時期の到達点を示すのは、2016年の「Google ニューラル機械翻訳」の登場なのですが、それに至る経過で重要な画期がいくつかあります。

一つが2014年の Ilya Sutskever らによる、ニューラルネットワークによる翻訳モデルの提案です。

もう一つが、2016年の Bengioのグループによる Ilya翻訳モデルの批判と「Attention メカニズム」の提案です。

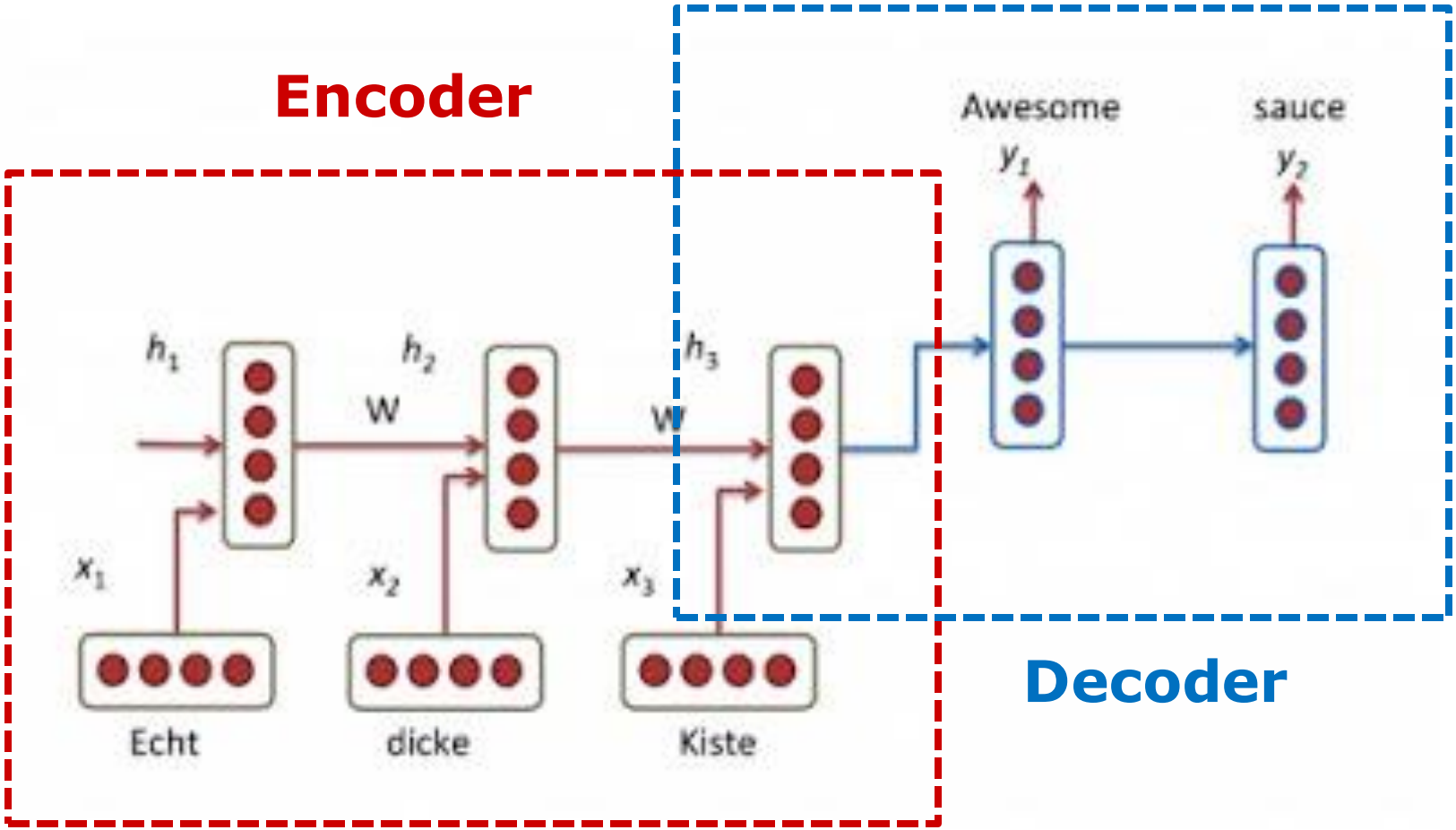
Ilyaの翻訳モデル
Sequence to Sequence

2014年に、Ilya Sutskever らは、シーケンスをシーケンスに変換するRNN(LSTM)の能力が、機械翻訳に応用できるという論文を発表します。

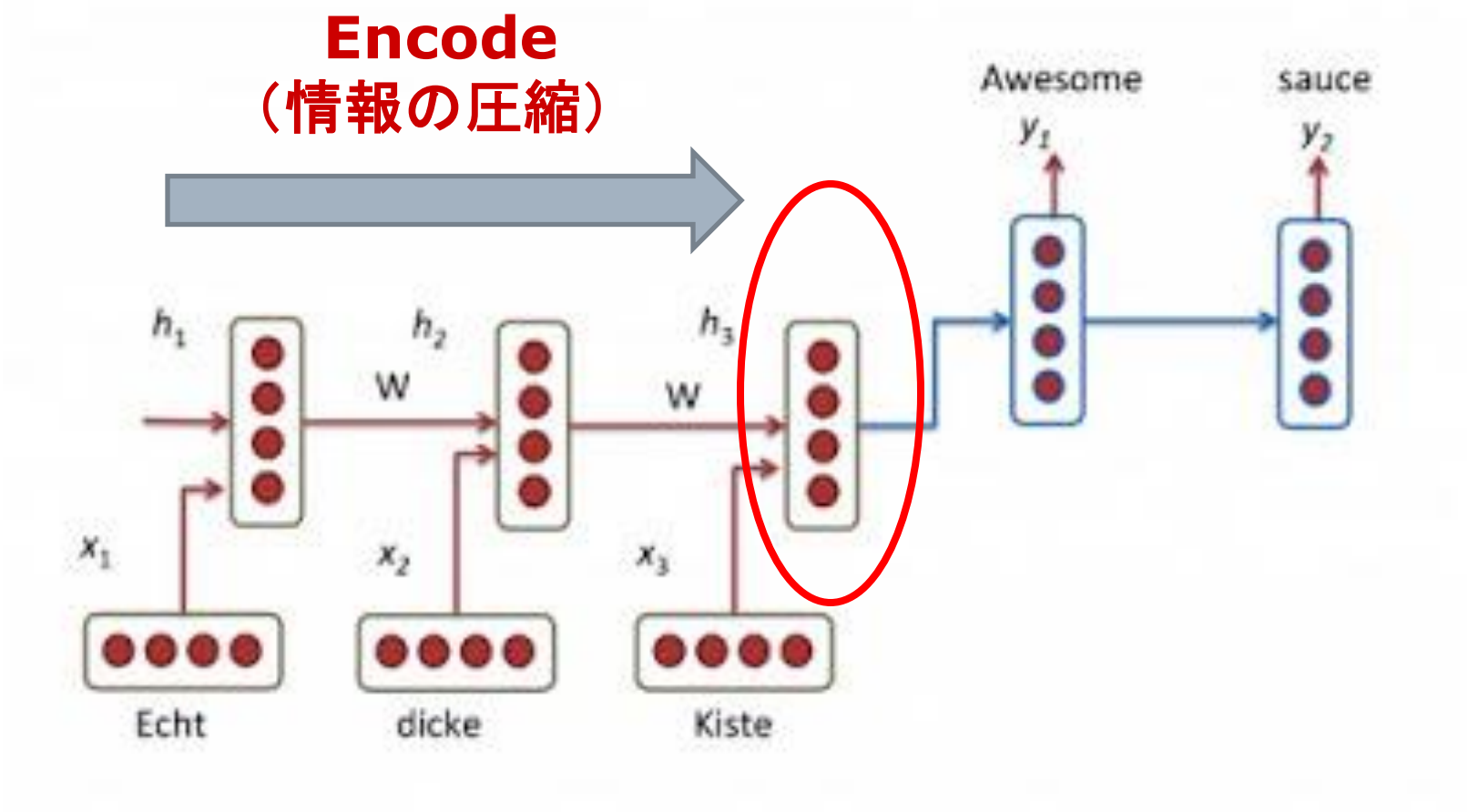
「我々の方法では、入力のシーケンスを固定次元のベクトルにマップするのに、多層のLong Short-Term Memory（LSTM）を利用する。その後、別の深いLSTMが、このベクトルから目的のシーケンスをデコードする。」

「Sequence to Sequence」は、当時、非常に注目されたコンセプトだったのですが、それは、単なる文字列から文字列への変換・生成とも解釈できます。その本当の意味は、皆に明らかだった訳ではなかったようにも思えます。

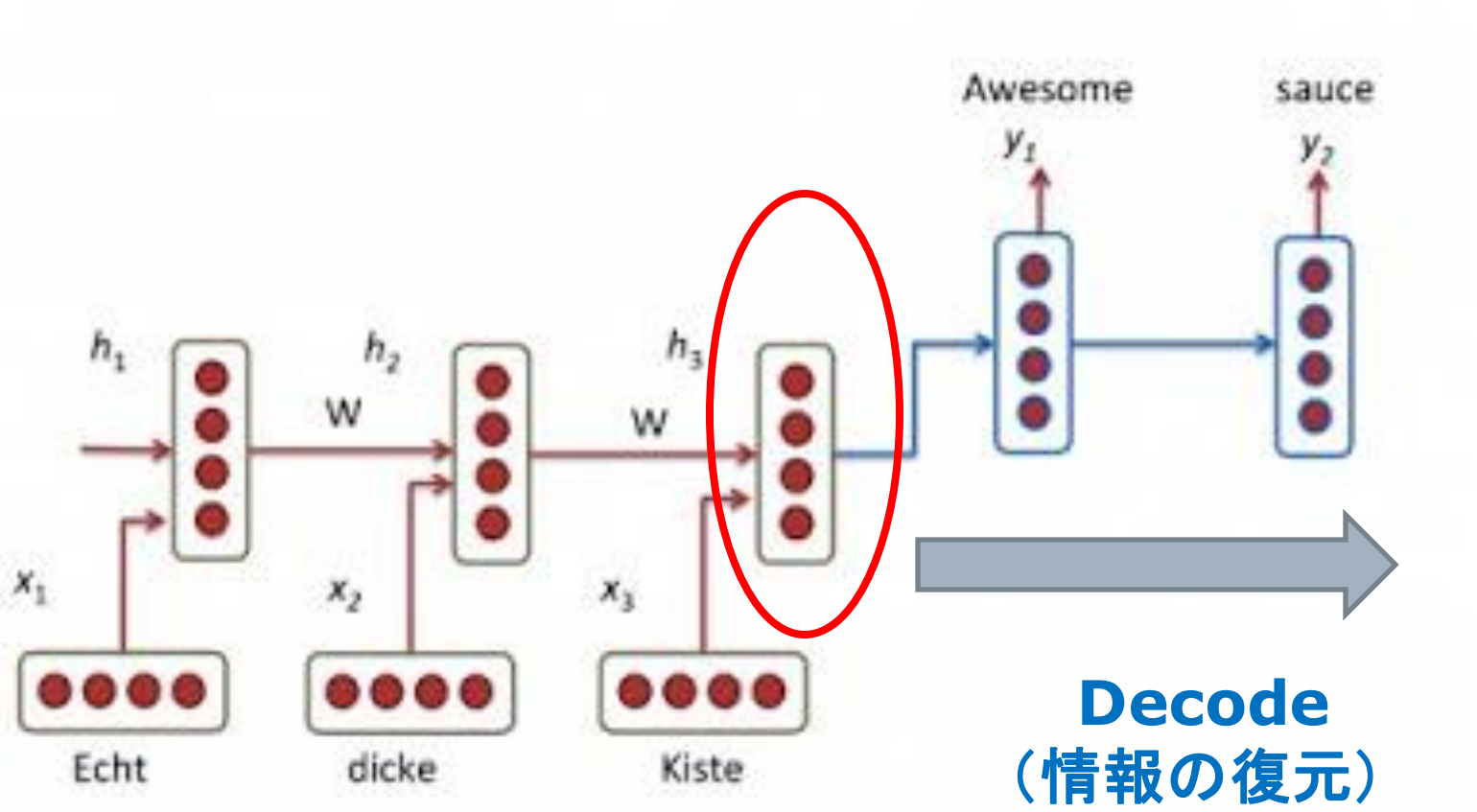
次の図(https://goo.gl/JGckBP から)は、こうしたメカニズムで、RNNが、独文の “Echt dicke Kiste” を英文の“Awesome sauce” に翻訳する様子を表している。(ここでは、文章の終わりを表す<EOS>は、省略されている) これは誤訳の例である。



ここでは、Encoder部が、文章の最後にとるRNNの内部状態 h_3 が、そのままDecoder部に渡されることが示されている。入力シーケンスの情報のエッセンスが、この内部状態 h_3に凝縮されていると考えればいい。



AutoencoderのDecoder部が、圧縮された情報から元の情報を復元しようとするように、ここでは、その情報から、「**同じ意味**」を持つ、別の言語の文章を復元しようとする。



文の意味のベクトル表現

それでは、翻訳モデルで二つのSequence を結びつけているのはなんでしょう。それは二つのSequenceが「同じ意味」を持つということです。

前段の入力のSequenceから作られ、後段の出力のSequenceを構成するのに利用される「固定次元のベクトル」とは、二つの文が「同じ意味」を持つことを表現している文の意味のベクトル表現に他なりません。

発見されたこの文の意味ベクトルは、次のセッションでに見るTransformer / BERT が作り上げる大規模言語モデルの世界で、本質的に重要な役割を果たすことになります。

Bahdanau たちの批判と Attentionメカニズムの登場

Ilya Sutskever らの翻訳システムでは、翻訳さるべき文は、Encoderで、一旦、ある決まった大きさの次元（例えば8000次元）を持つベクトルに変換されます。このベクトルからDecoderが翻訳文を生成します。

入力された文が、長いものであっても短いものであっても、中間で生成され以降の翻訳プロセスすべての出発点となるこのベクトルの大きさは同じままです。このシステムでは、長くても短くても入力された文全体が、一つの固定長のベクトルに変換されるます。

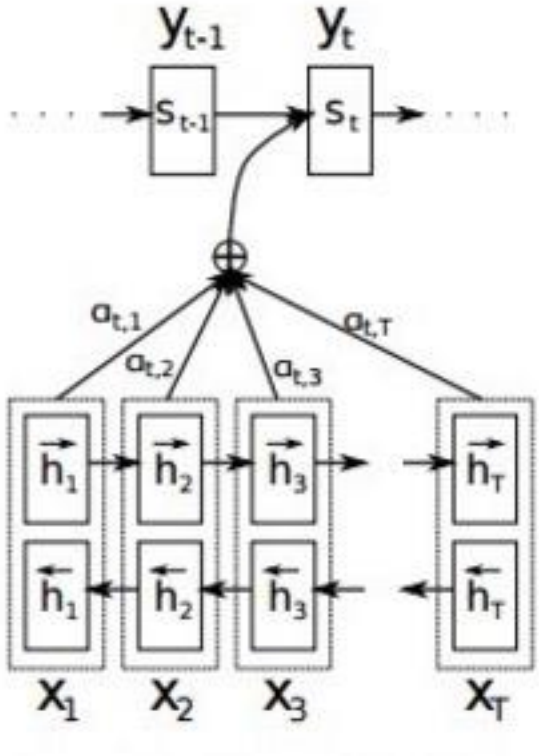
2016年の論文で、Bahdanauらは、Ilyaの翻訳モデルを「固定長ベクトルの使用が、この基本的なEncoder/Decoderアーキテクチャの性能を改善する上でのボトルネックになっている」と批判します。

その上で、「モデルに自動的に、ターゲット・ワードを予測するのに重要なソース・文の一部分について、（ソフト）検索を可能とすることによって、これを拡張すること」を提案します。

これが、Attention メカニズムです。

基本的アイデア

文全体に一つの固定長のベクトルを割り当てるのではなく、翻訳時に、ソース・文の一部分を 改めて見直して、その部分から提供される情報を翻訳に生かそうということだ。



$a_{3,2}$が大きい場合、これは、Decorderがターゲット文の第3の単語を生成しながら、ソース文の第2の状態に多くの注意を払うことを意味する。

「ここで、yはデコーダによって生成された翻訳された単語であり、xは原文の単語である。上記の図は双方向のリカレント・ネットワークを使用しているが、それは重要ではない。逆方向は無視していい。

重要な部分は、各デコーダの出力するワード y_tが、Encoderの最後の状態だけでなく、すべての入力状態の重みづけられた結合に依存することである。

aは、出力ごとに、それぞれの入力状態をどの程度考慮されるべきかを定義する重みである。したがって、$a_{3,2}$が大きい場合、これは、Decorderがターゲット文の第3の単語を生成しながら、ソース文の第2の状態に多くの注意を払うことを意味する。

aは、通常、1に合計されるように正規化される（それらは、入力状態に対する確率分布である）。」

先行したVision Attention

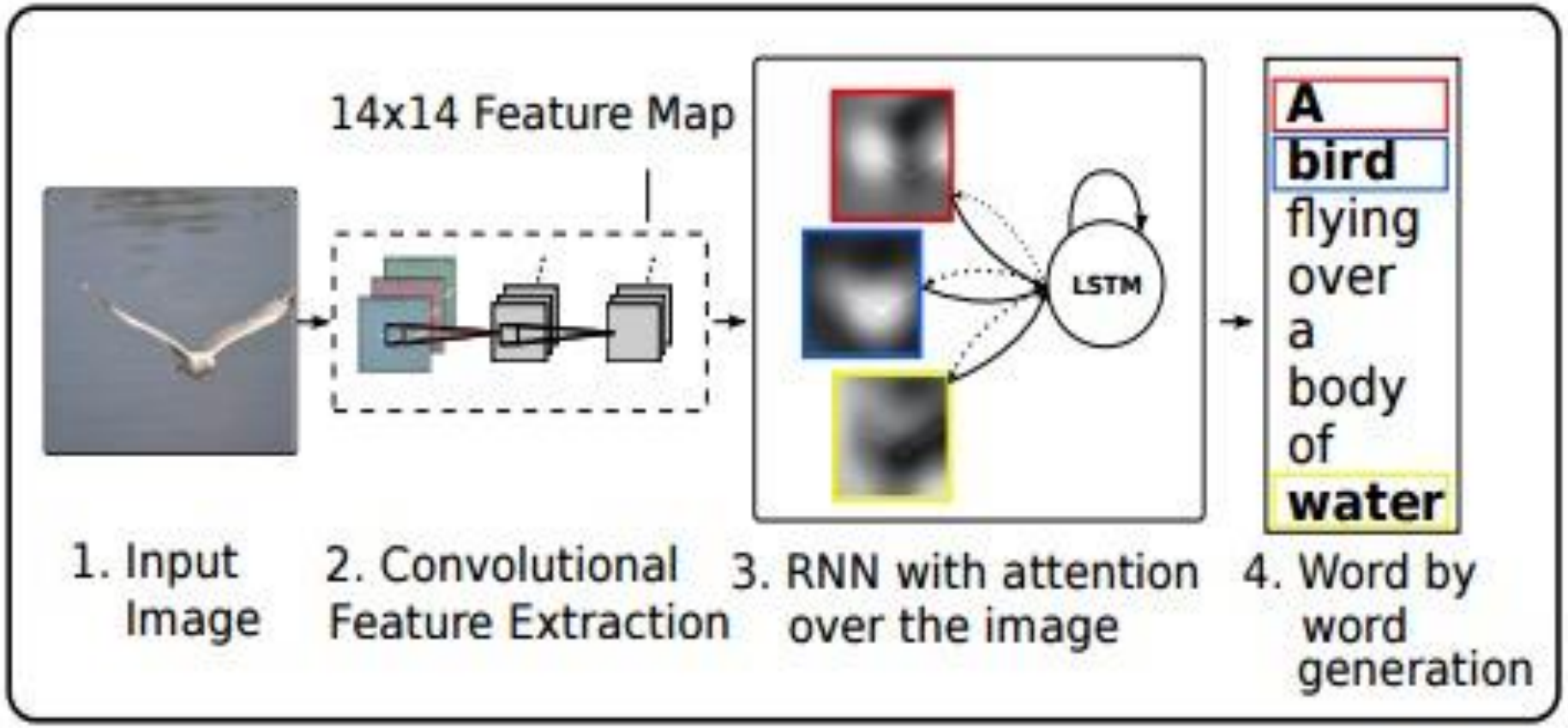
実は、2016年のBahdanauらの論文より前に、Attentionの重要性を指摘した論文があるのです。それは、2015年のKelvin Xu らの論文です。

BahdanauもKelvin Xuも、Bengioの研究グループに属する人で、Attentionについてのこの二つの先駆的な論文には、いずれにも、Bengio が Last Author として名を連ねています。

興味深いことは、ここで提唱されているのは、画像に対するAttentionを利用することで、画像からCaptionを生成することができるというシステムでした。

簡単にいうと、画像のある部分にAttentionを固定した時（それは文字通りあるオブジェクトに「注意」を集中することです)、そのオブジェクトに対応する単語を生成するというものです。視点が移動するにつれて、Captionが生成されることになります

システム概要


14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

画像の特定の部分に注意を向ける

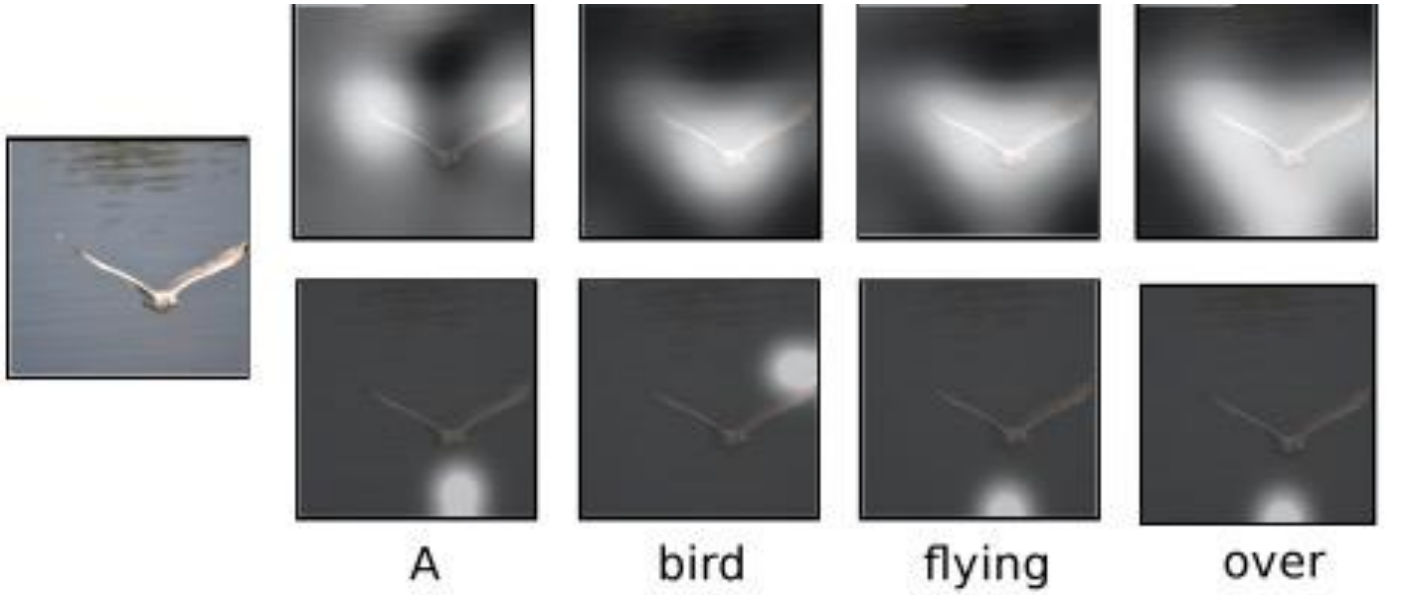
A woman is throwing a frisbee in a park.

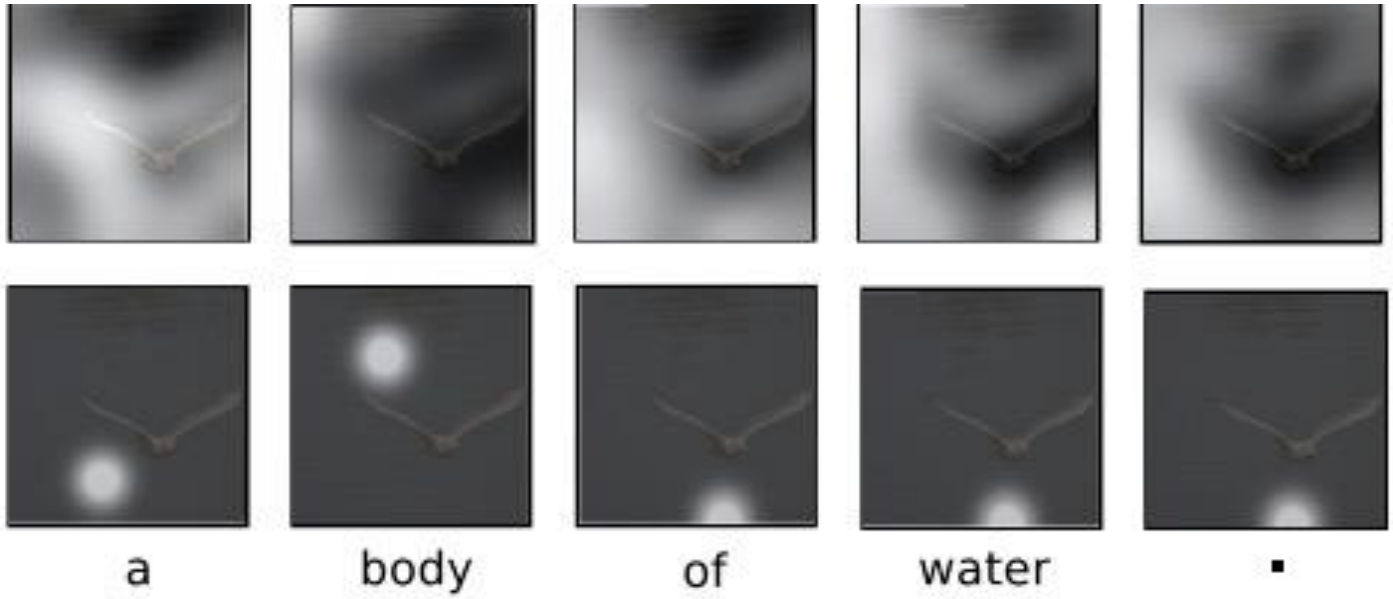
A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

生成されたcaption

注意点の移動とcaptionの生成





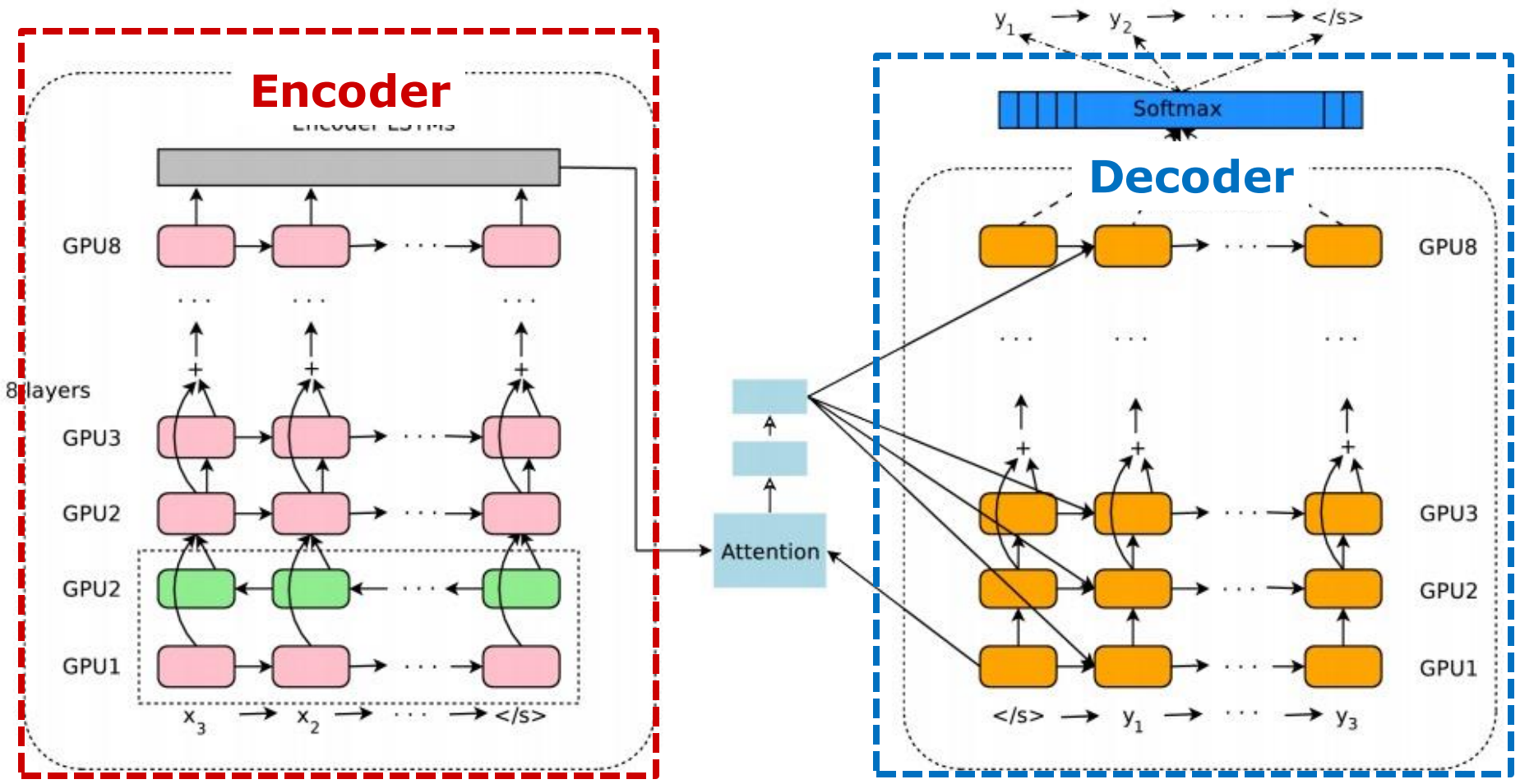
Googleのニューラル機械翻訳の成立

Ilya Sutskever らのニューラル翻訳システムとBengioらのAttentionメカニズムの提案という二つの研究の流れが合流したものが「Google ニューラル機械翻訳」です。

それは、Attentionメカニズムと文の意味の分散表現の採用という遺伝子を、Transformer / BERT らの大規模言語モデルに伝えることになりました。

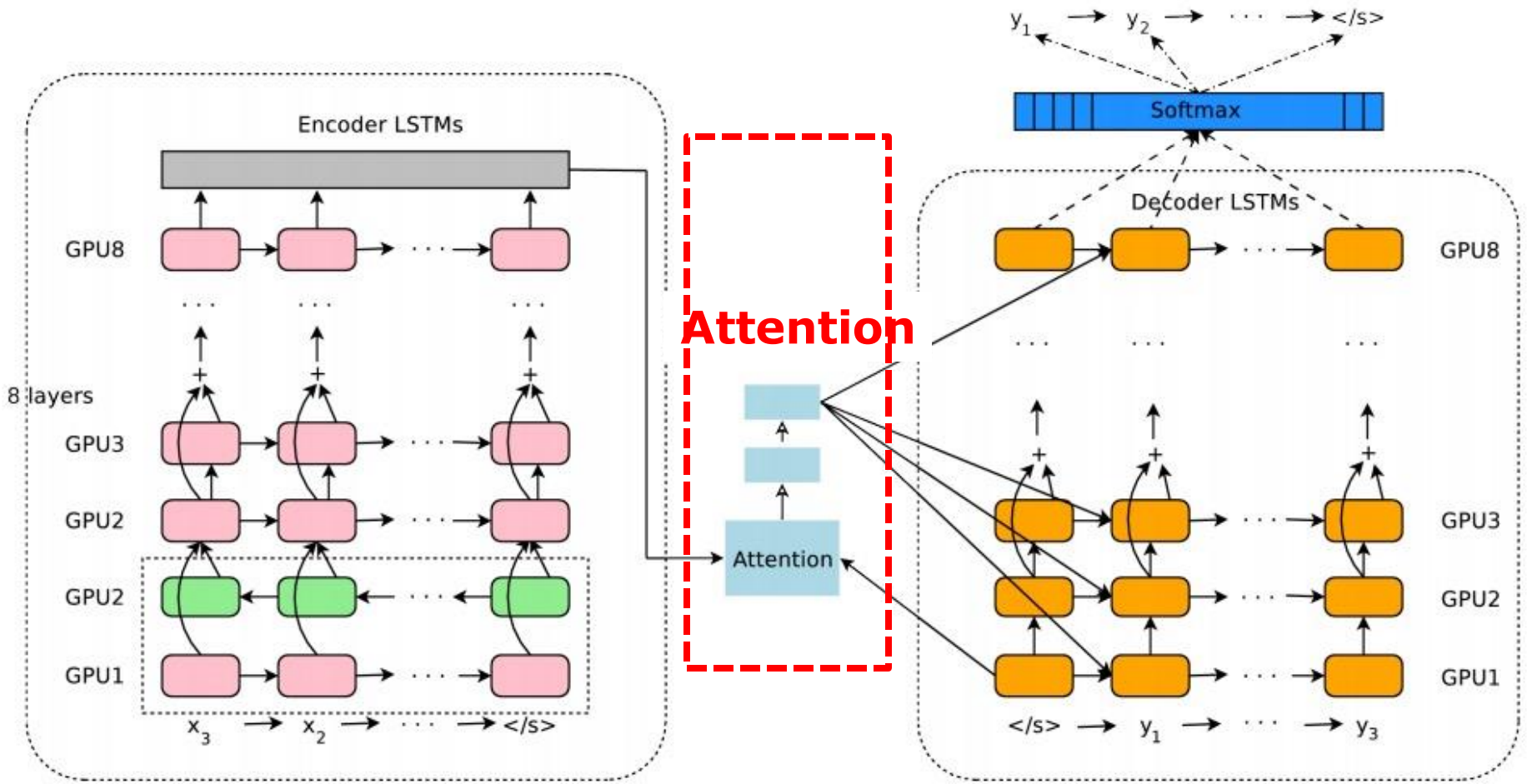
翻訳モデルは、大規模言語モデルの母胎なのです。

Encoder / Decoder



左側に、LSTMを8段重ねにした Encoder LSTMがあり、
右側には、同じくLSTMを8段重ねにした Decoder LSTMがある。

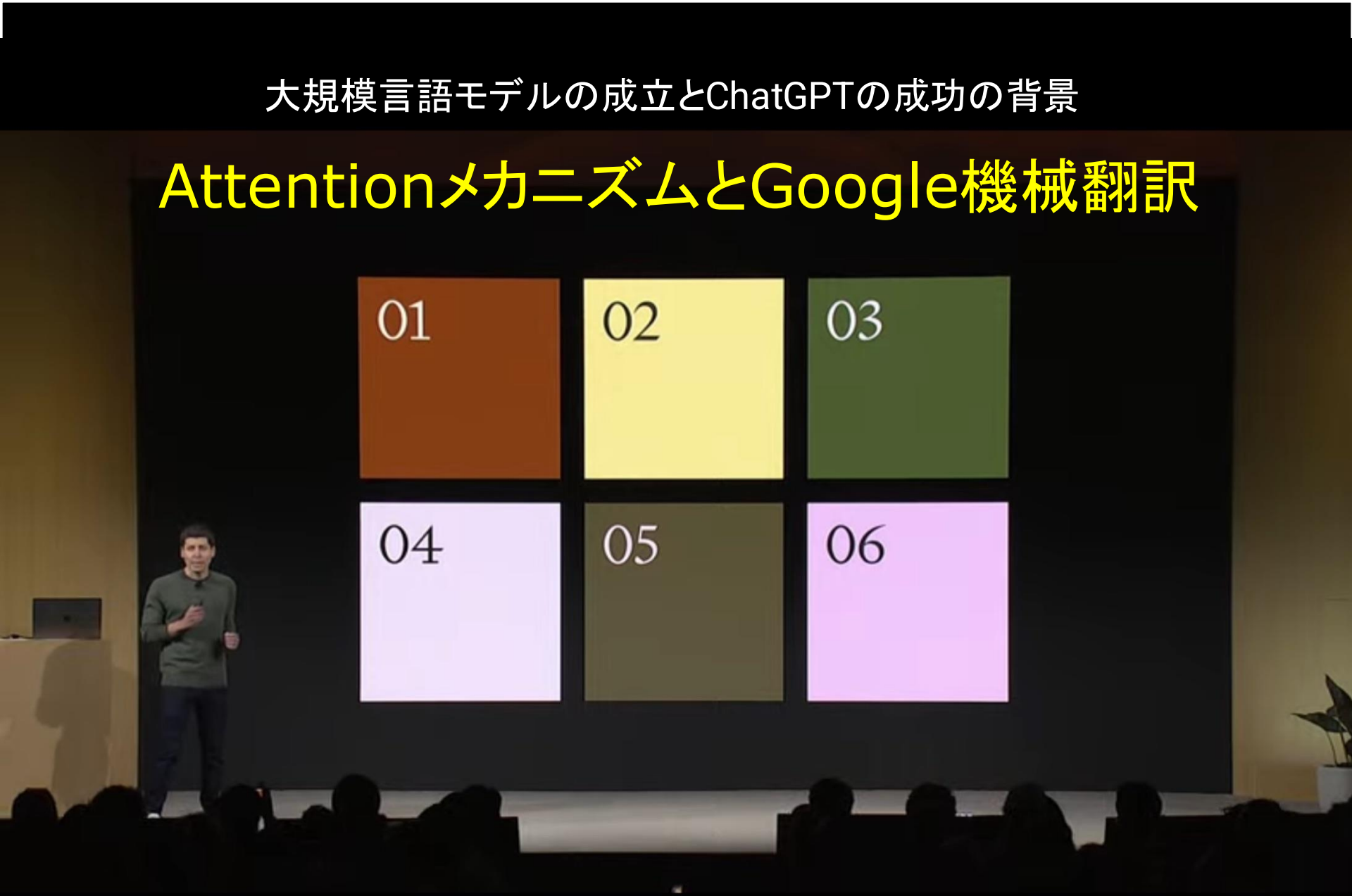
Attention Mechanism



EncoderとDecoderの中間に、Attentionと記された領域がある。
ここからの出力Attention Context は、Decoderのすべての
ノードに供給されている。

大規模言語モデルの成立とChatGPTの成功の背景

AttentionメカニズムとGoogle機械翻訳



11/24 角川セミナー

大規模言語モデルの成立期

このセッションでは、大規模言語モデルの成立期の話をしようと思います。まず、大まかな流れを見ておきましょう。

この時期の到達点を示すのは、2016年の「Google ニューラル機械翻訳」の登場なのですが、それに至る経過で重要な画期がいくつかあります。

一つが2014年の Ilya Sutskever らによる、ニューラルネットワークによる翻訳モデルの提案です。

もう一つが、2016年の Bengioのグループによる Ilya翻訳モデルの批判と「Attention メカニズム」の提案です。

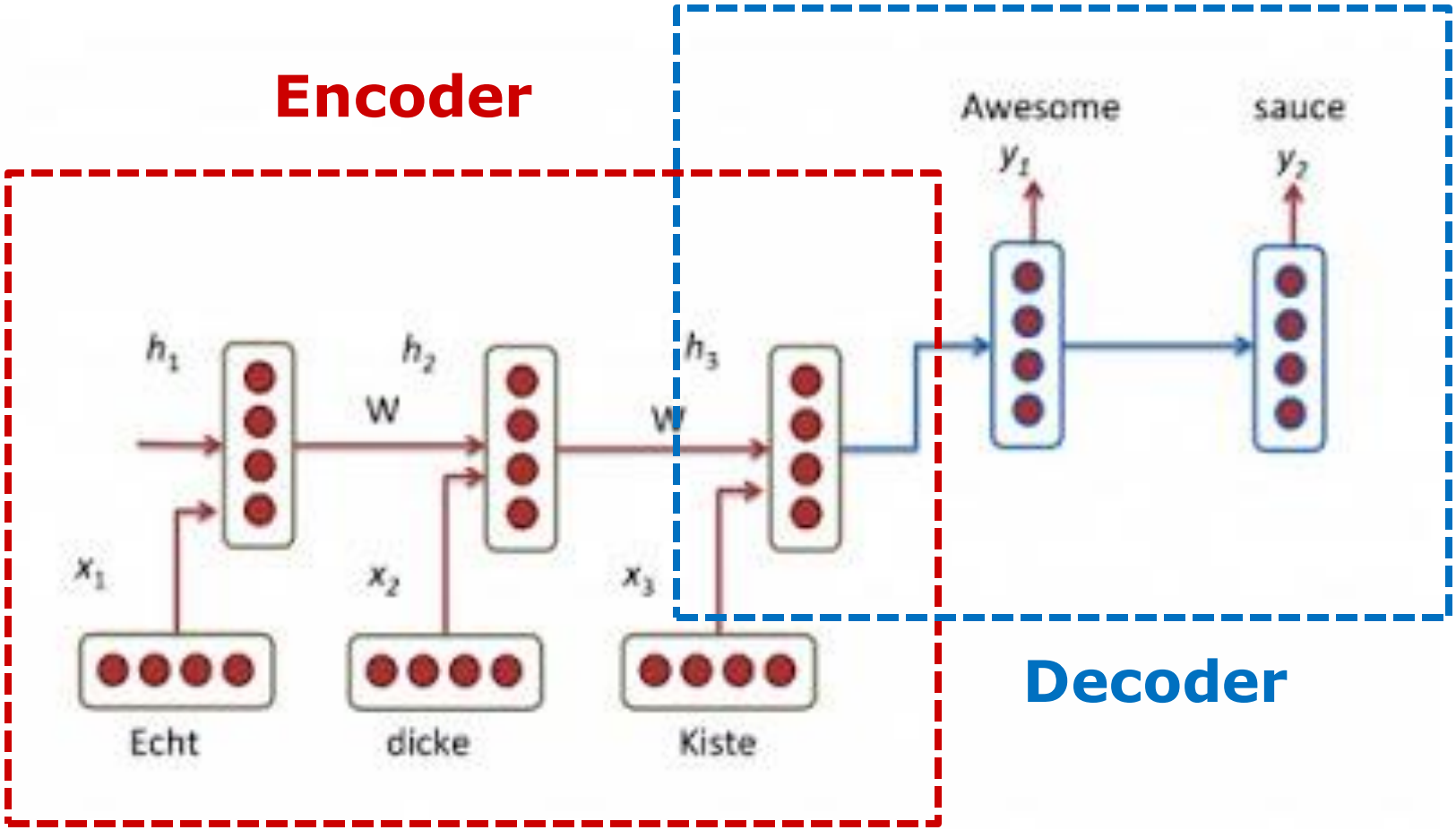
Ilyaの翻訳モデル
Sequence to Sequence

2014年に、Ilya Sutskever らは、シーケンスをシーケンスに変換するRNN(LSTM)の能力が、機械翻訳に応用できるという論文を発表します。

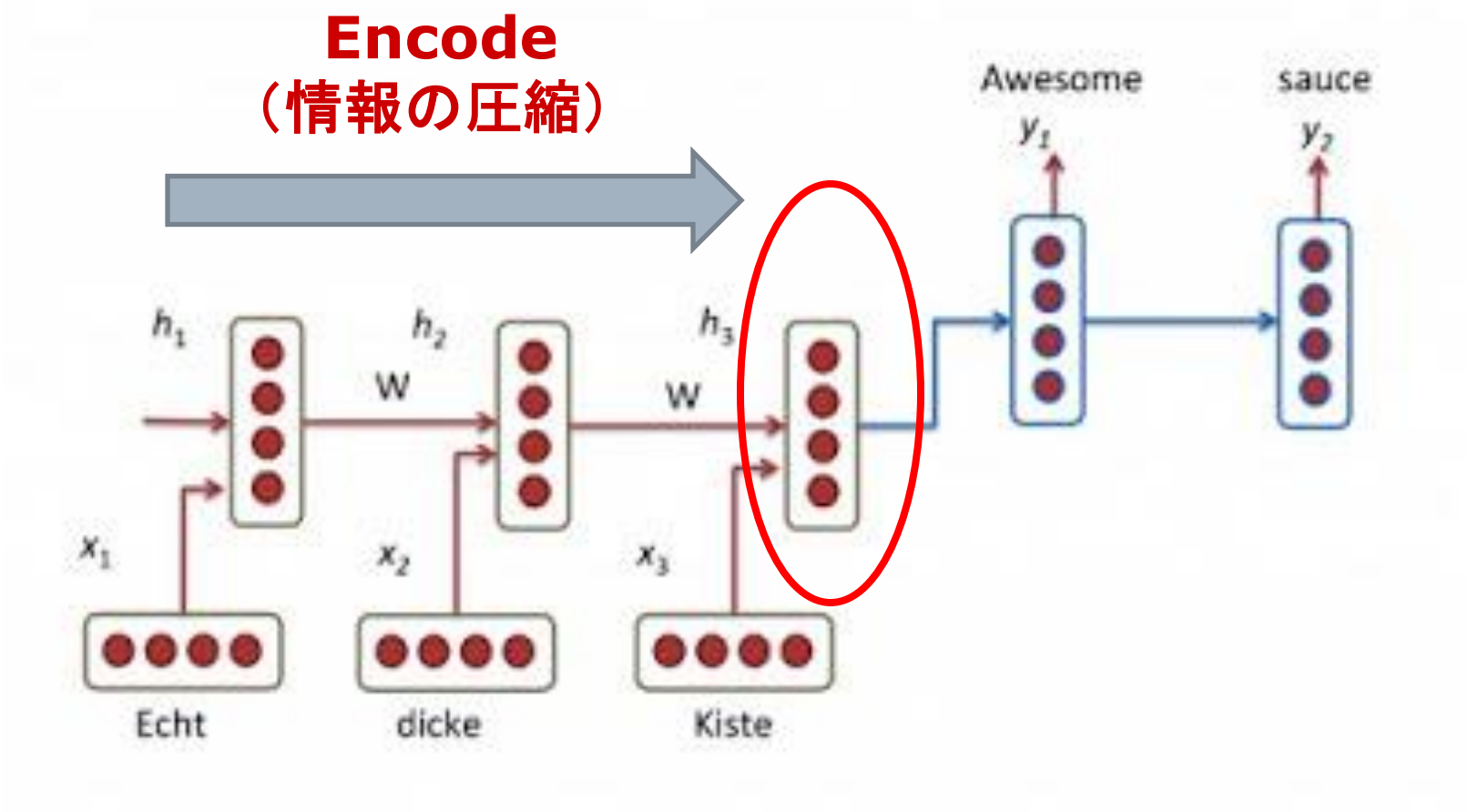
「我々の方法では、入力のシーケンスを固定次元のベクトルにマップするのに、多層のLong Short-Term Memory（LSTM）を利用する。その後、別の深いLSTMが、このベクトルから目的のシーケンスをデコードする。」

「Sequence to Sequence」は、当時、非常に注目されたコンセプトだったのですが、それは、単なる文字列から文字列への変換・生成とも解釈できます。その本当の意味は、皆に明らかだった訳ではなかったようにも思えます。

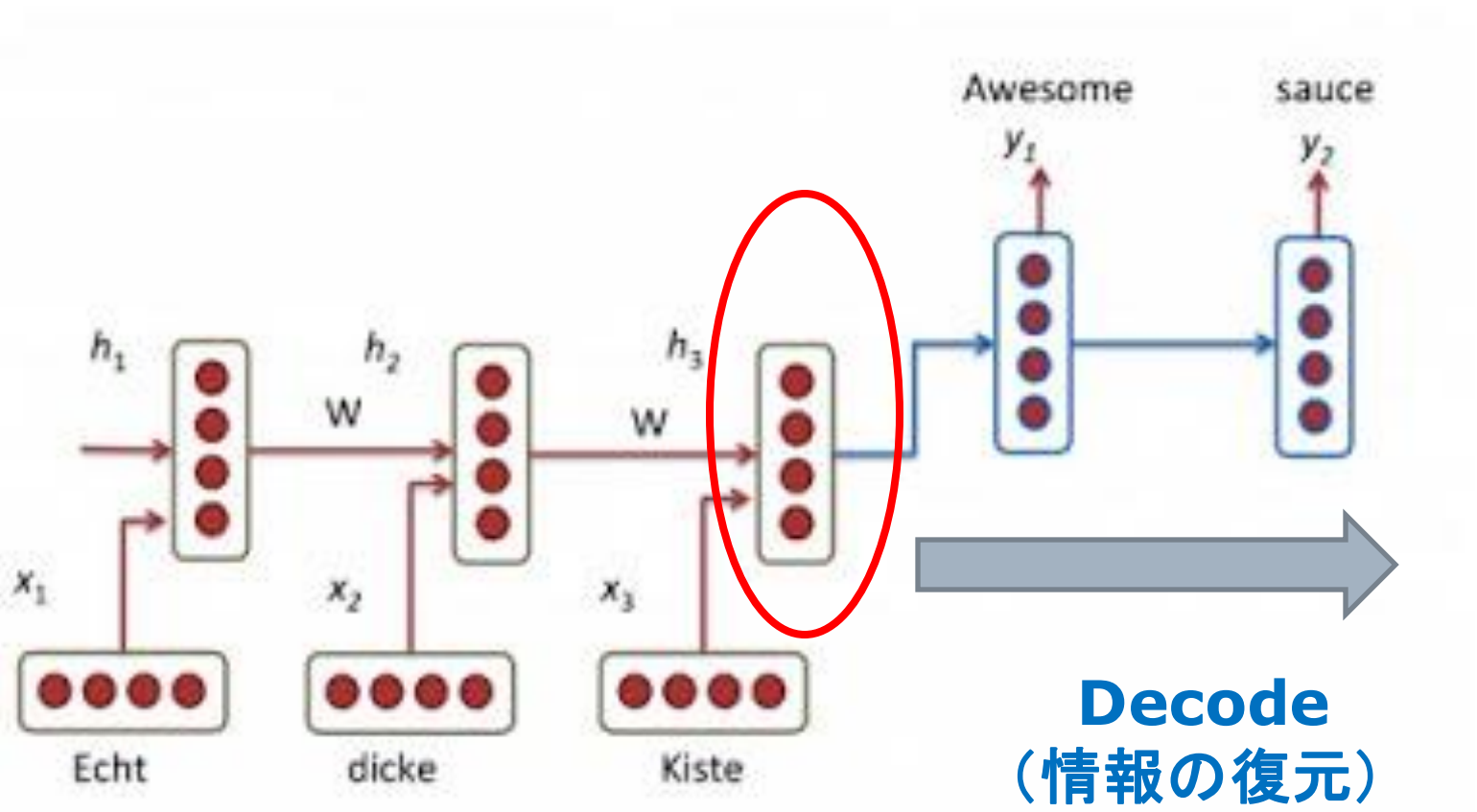
次の図(https://goo.gl/JGckBP から)は、こうしたメカニズムで、RNNが、独文の “Echt dicke Kiste” を英文の“Awesome sauce” に翻訳する様子を表している。(ここでは、文章の終わりを表す<EOS>は、省略されている) これは誤訳の例である。



ここでは、Encoder部が、文章の最後にとるRNNの内部状態 h_3 が、そのままDecoder部に渡されることが示されている。入力シーケンスの情報のエッセンスが、この内部状態 h_3に凝縮されていると考えればいい。



AutoencoderのDecoder部が、圧縮された情報から元の情報を復元しようとするように、ここでは、その情報から、「**同じ意味**」を持つ、別の言語の文章を復元しようとする。



文の意味のベクトル表現

それでは、翻訳モデルで二つのSequence を結びつけているのはなんでしょう。それは二つのSequenceが「同じ意味」を持つということです。

前段の入力のSequenceから作られ、後段の出力のSequenceを構成するのに利用される「固定次元のベクトル」とは、二つの文が「同じ意味」を持つことを表現している文の意味のベクトル表現に他なりません。

発見されたこの文の意味ベクトルは、次のセッションでに見るTransformer / BERT が作り上げる大規模言語モデルの世界で、本質的に重要な役割を果たすことになります。

Bahdanau たちの批判と Attentionメカニズムの登場

Ilya Sutskever らの翻訳システムでは、翻訳さるべき文は、Encoderで、一旦、ある決まった大きさの次元（例えば8000次元）を持つベクトルに変換されます。このベクトルからDecoderが翻訳文を生成します。

入力された文が、長いものであっても短いものであっても、中間で生成され以降の翻訳プロセスすべての出発点となるこのベクトルの大きさは同じままです。このシステムでは、長くても短くても入力された文全体が、一つの固定長のベクトルに変換されるます。

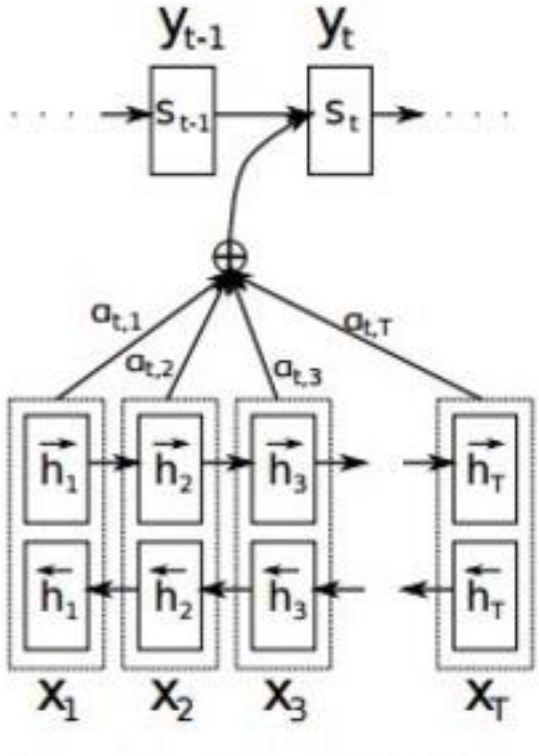
2016年の論文で、Bahdanauらは、Ilyaの翻訳モデルを「固定長ベクトルの使用が、この基本的なEncoder/Decoderアーキテクチャの性能を改善する上でのボトルネックになっている」と批判します。

その上で、「モデルに自動的に、ターゲット・ワードを予測するのに重要なソース・文の一部分について、（ソフト）検索を可能とすることによって、これを拡張すること」を提案します。

これが、Attention メカニズムです。

基本的アイデア

文全体に一つの固定長のベクトルを割り当てるのではなく、翻訳時に、ソース・文の一部分を 改めて見直して、その部分から提供される情報を翻訳に生かそうということだ。



$a_{3,2}$が大きい場合、これは、Decorderがターゲット文の第3の単語を生成しながら、ソース文の第2の状態に多くの注意を払うことを意味する。

「ここで、yはデコーダによって生成された翻訳された単語であり、xは原文の単語である。上記の図は双方向のリカレント・ネットワークを使用しているが、それは重要ではない。逆方向は無視していい。

重要な部分は、各デコーダの出力するワード y_tが、Encoderの最後の状態だけでなく、すべての入力状態の重みづけられた結合に依存することである。

aは、出力ごとに、それぞれの入力状態をどの程度考慮されるべきかを定義する重みである。したがって、$a_{3,2}$が大きい場合、これは、Decorderがターゲット文の第3の単語を生成しながら、ソース文の第2の状態に多くの注意を払うことを意味する。

aは、通常、1に合計されるように正規化される（それらは、入力状態に対する確率分布である）。」

先行したVision Attention

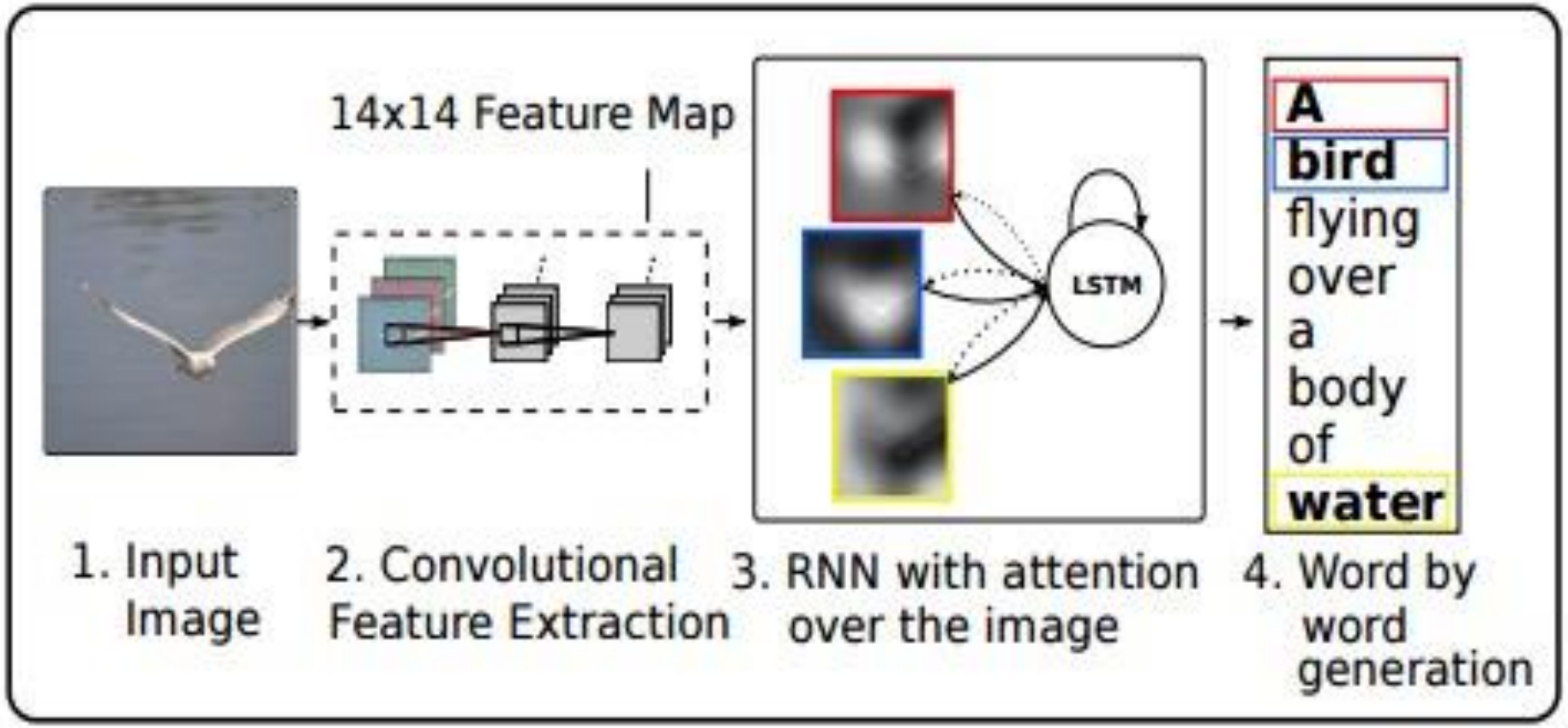
実は、2016年のBahdanauらの論文より前に、Attentionの重要性を指摘した論文があるのです。それは、2015年のKelvin Xu らの論文です。

BahdanauもKelvin Xuも、Bengioの研究グループに属する人で、Attentionについてのこの二つの先駆的な論文には、いずれにも、Bengio が Last Author として名を連ねています。

興味深いことは、ここで提唱されているのは、画像に対するAttentionを利用することで、画像からCaptionを生成することができるというシステムでした。

簡単にいうと、画像のある部分にAttentionを固定した時（それは文字通りあるオブジェクトに「注意」を集中することです)、そのオブジェクトに対応する単語を生成するというものです。視点が移動するにつれて、Captionが生成されることになります

システム概要


14x14 Feature Map
LSTM
A
bird
flying
over
a
body
of
water
1. Input Image
2. Convolutional Feature Extraction
3. RNN with attention over the image
4. Word by word generation

画像の特定の部分に注意を向ける

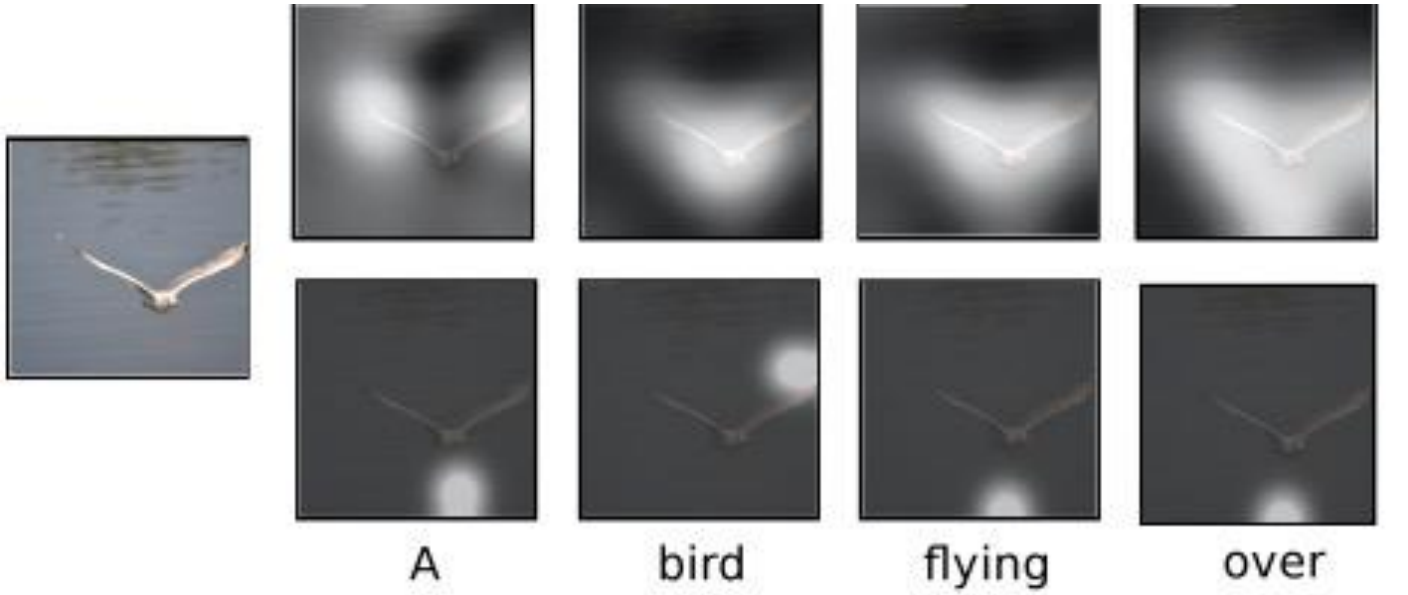
A woman is throwing a frisbee in a park.

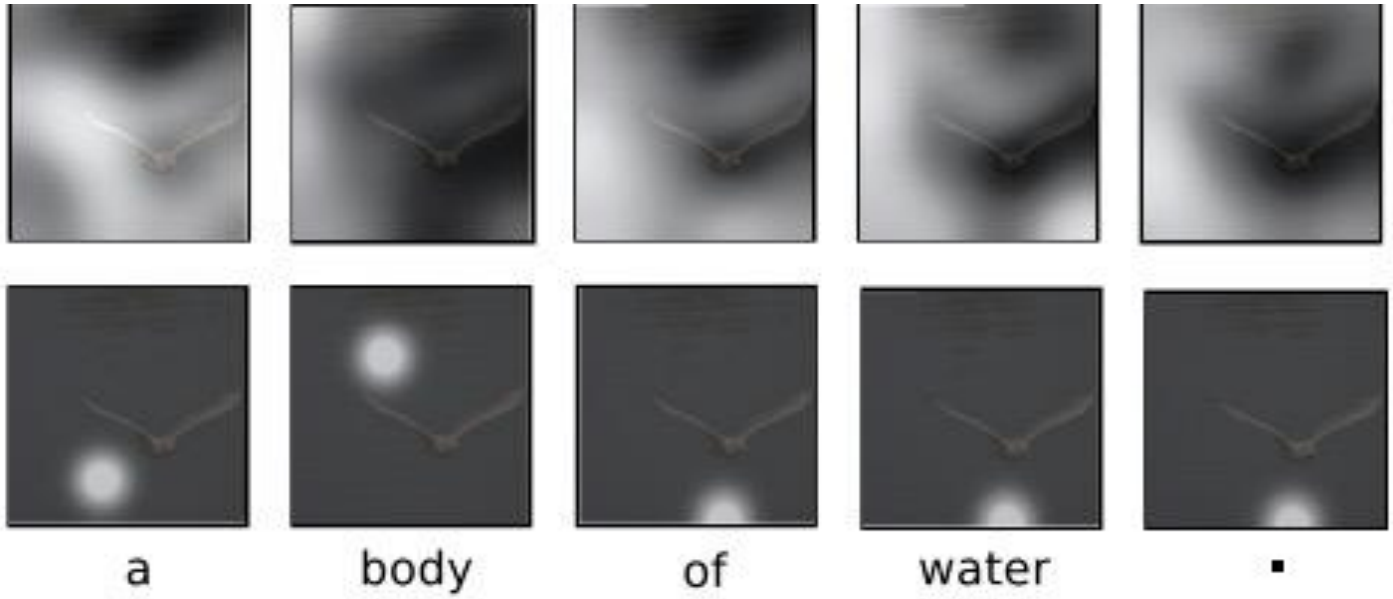
A dog is standing on a hardwood floor.

A stop sign is on a road with a mountain in the background.

生成されたcaption

注意点の移動とcaptionの生成





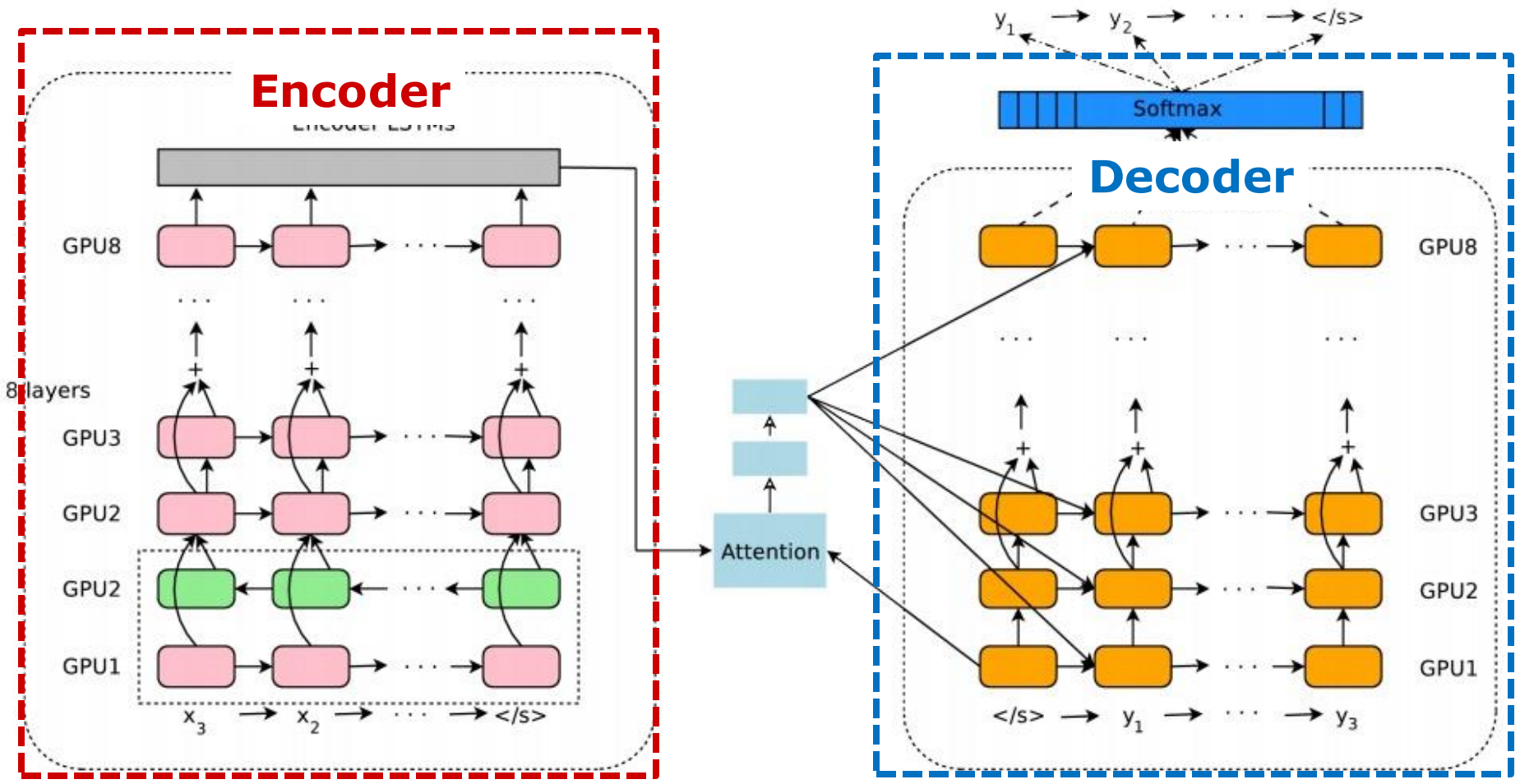
Googleのニューラル機械翻訳の成立

Ilya Sutskever らのニューラル翻訳システムとBengioらのAttentionメカニズムの提案という二つの研究の流れが合流したものが「Google ニューラル機械翻訳」です。

それは、Attentionメカニズムと文の意味の分散表現の採用という遺伝子を、Transformer / BERT らの大規模言語モデルに伝えることになりました。

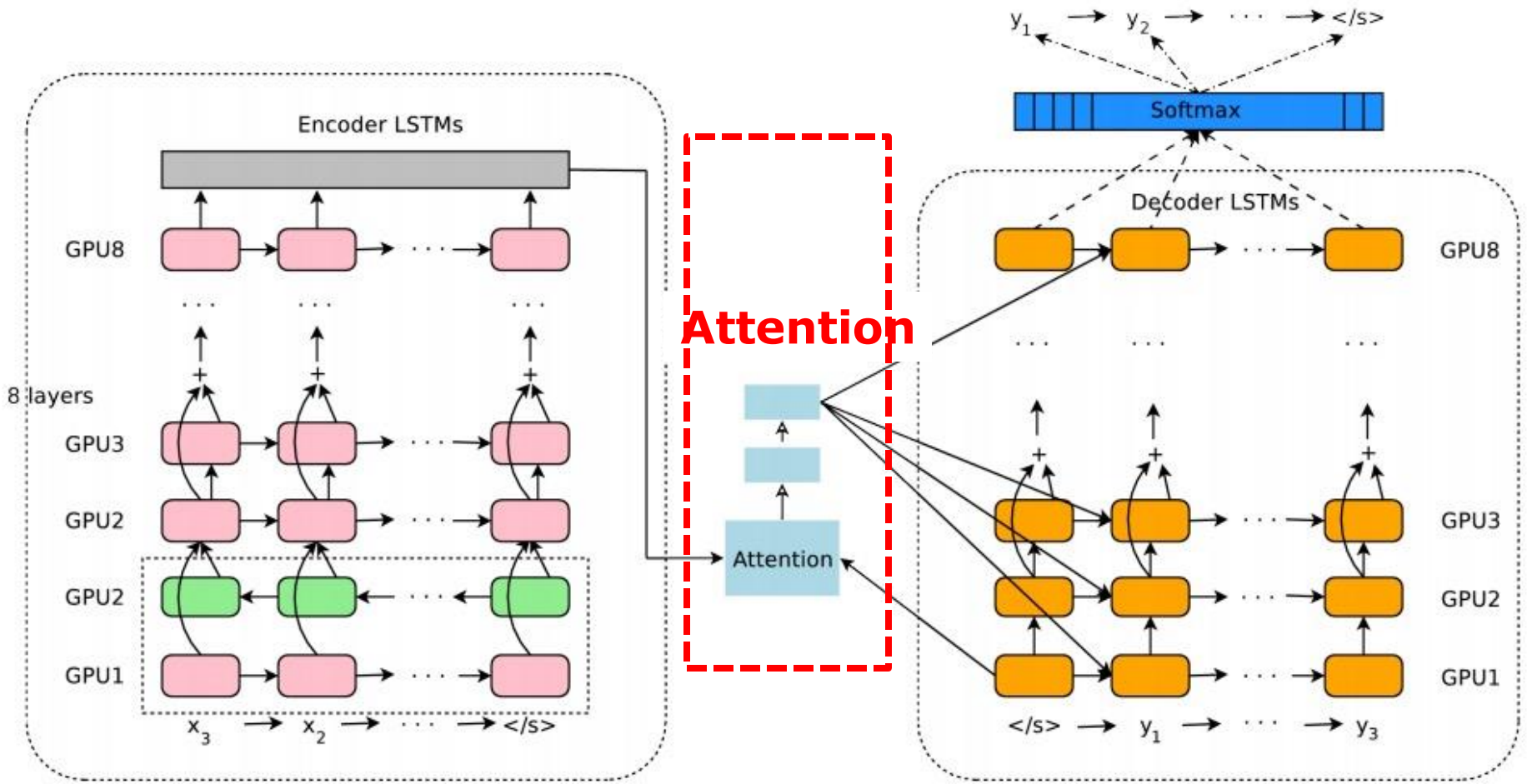
翻訳モデルは、大規模言語モデルの母胎なのです。

Encoder / Decoder



左側に、LSTMを8段重ねにした Encoder LSTMがあり、
右側には、同じくLSTMを8段重ねにした Decoder LSTMがある。

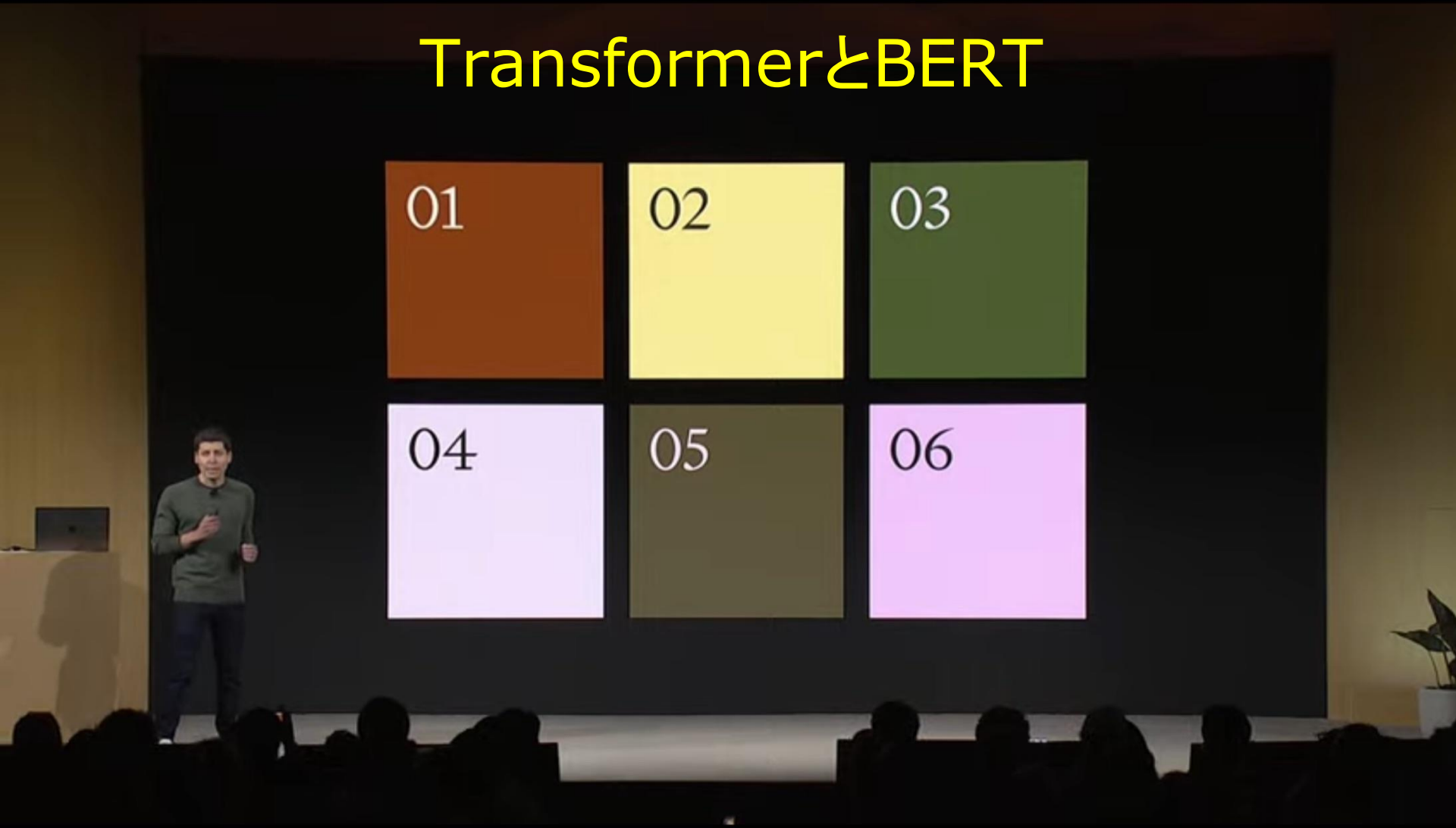
Attention Mechanism



EncoderとDecoderの中間に、Attentionと記された領域がある。
ここからの出力Attention Context は、Decoderのすべての
ノードに供給されている。

大規模言語モデルの成立とChatGPTの成功の背景

TransformerとBERT



11/24 角川セミナー

大規模言語モデルの基礎

このセッションでは、現在の大規模言語モデルの基礎となっている二つのアーキテクチャーを紹介します。

一つは、2017年にGoogleが発表したアーキテクチャーTransformerです。

もう一つは、2019年に Google が発表した「言語表現モデル」BERTです。

次回のセッションで紹介するChatGPTは、もちろんOpenAIのプロダクトですが、この時期のAI技術は、主要にGoogleによって推進されてきたことに留意ください。

TransformerとGoogleニューラル機械翻訳

Transformerは、GoogleのBERTやOpenAIのGPTといった現代の大規模言語モデルほとんど全ての基礎になっています。BERTの最後の文字 'T' も、GPTの'T'も"Transformer" アーキテクチャーを採用していることを表しています。

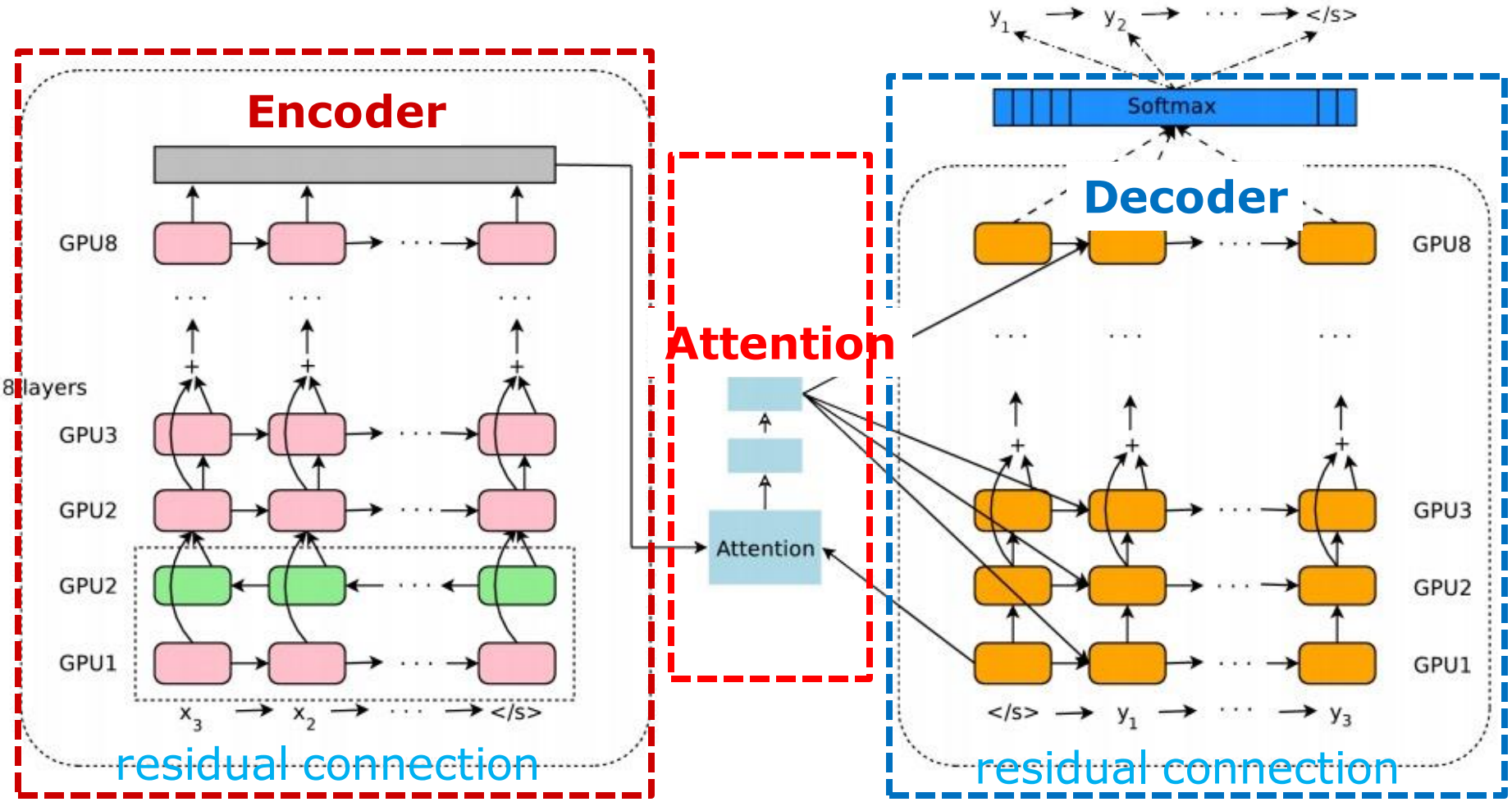
まず最初に確認したいのは、見かけはずいぶん違って見えますが、Transformer アーキテクチャーは、大きな成功を収めた2016年のGoogle ニューラル機械翻訳のアーキテクチャーから多くを学んでいるということです。

ポイントをあげれば、Encode-Decoder アーキテクチャーの採用、EncoderとDecoderの分離、両者をつなぐAttention Mechanismの採用、等々。

こうした、Google ニューラル機械翻訳のアーキテクチャーの特徴は、そのまま、Transformerのアーキテクチャーに引き継がれています。

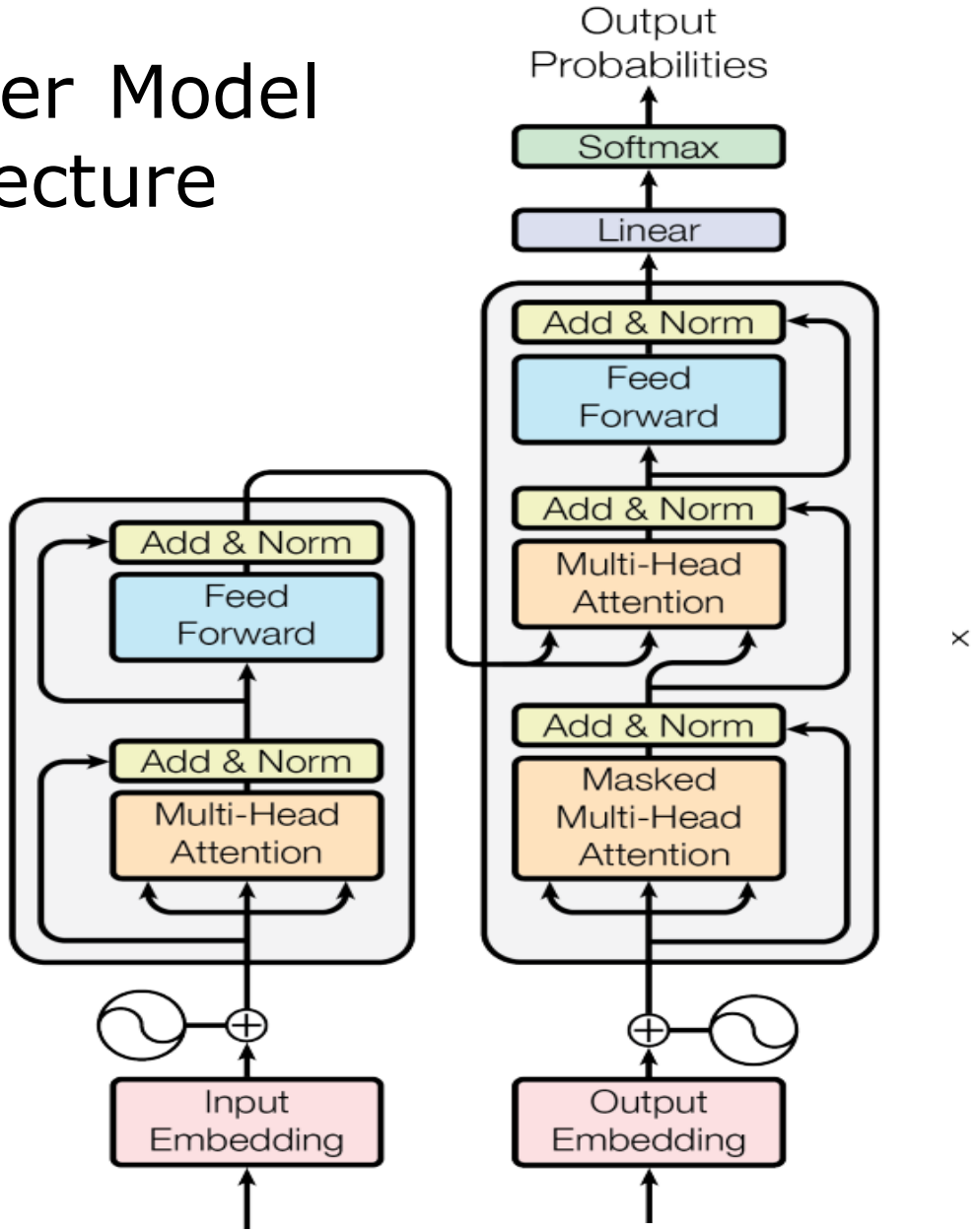
それらの特徴の中で、Attentionこそが一番重要なのだというのが、Transformerの提案者の分析なのだと思います。

Google ニューラル機械翻訳のアーキテクチャー

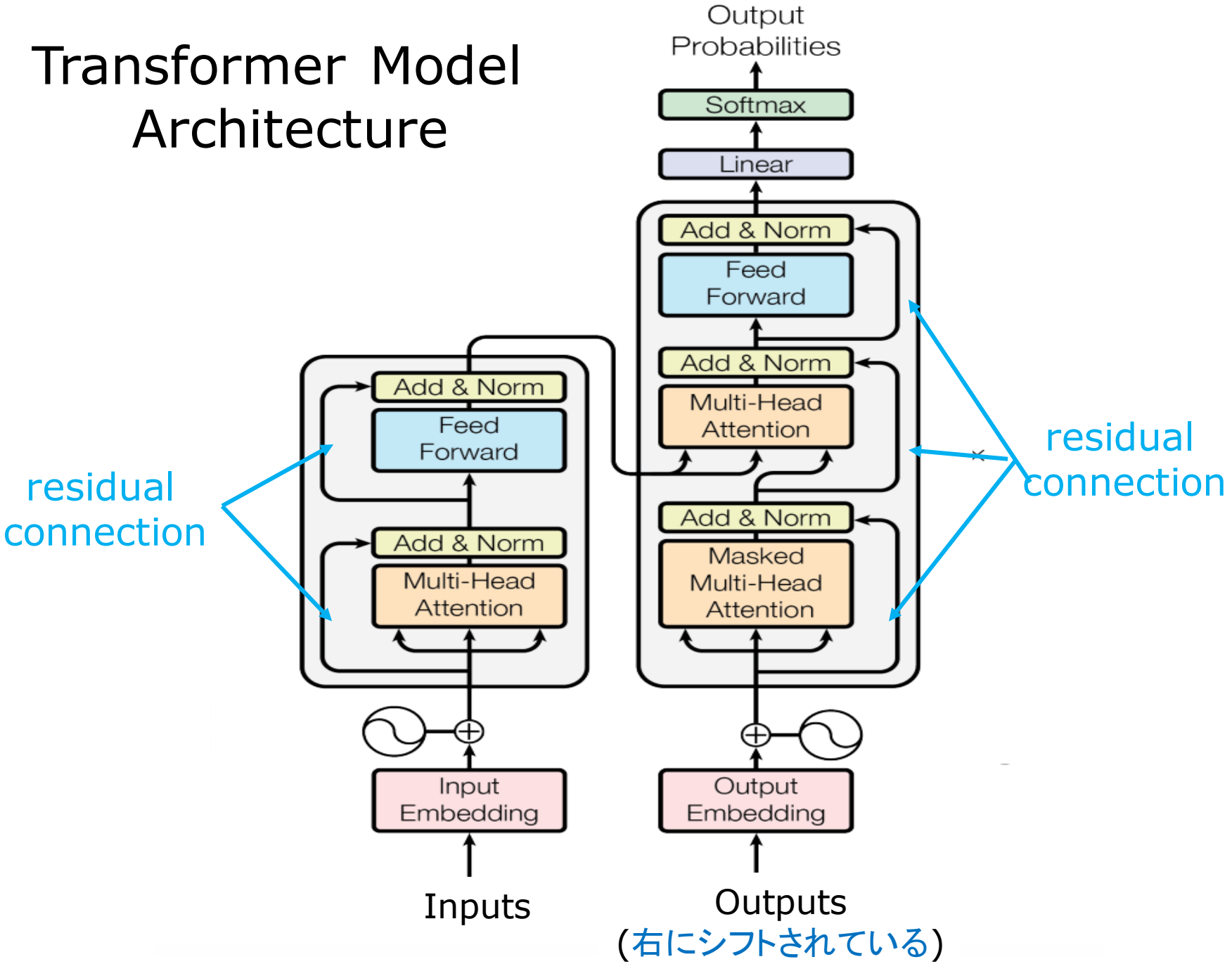


Encoder / Decoder/Attention

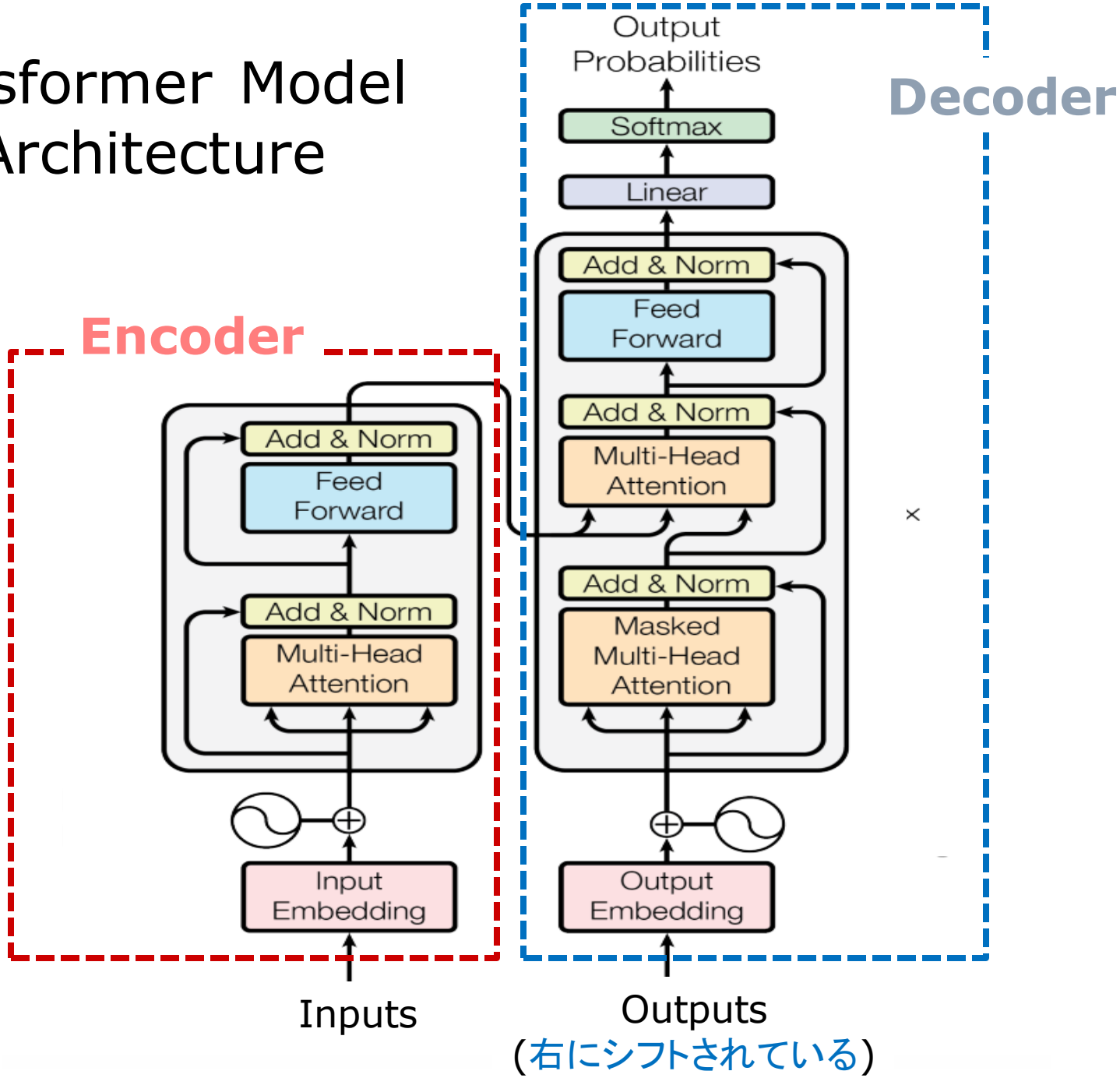
Transformer Model Architecture



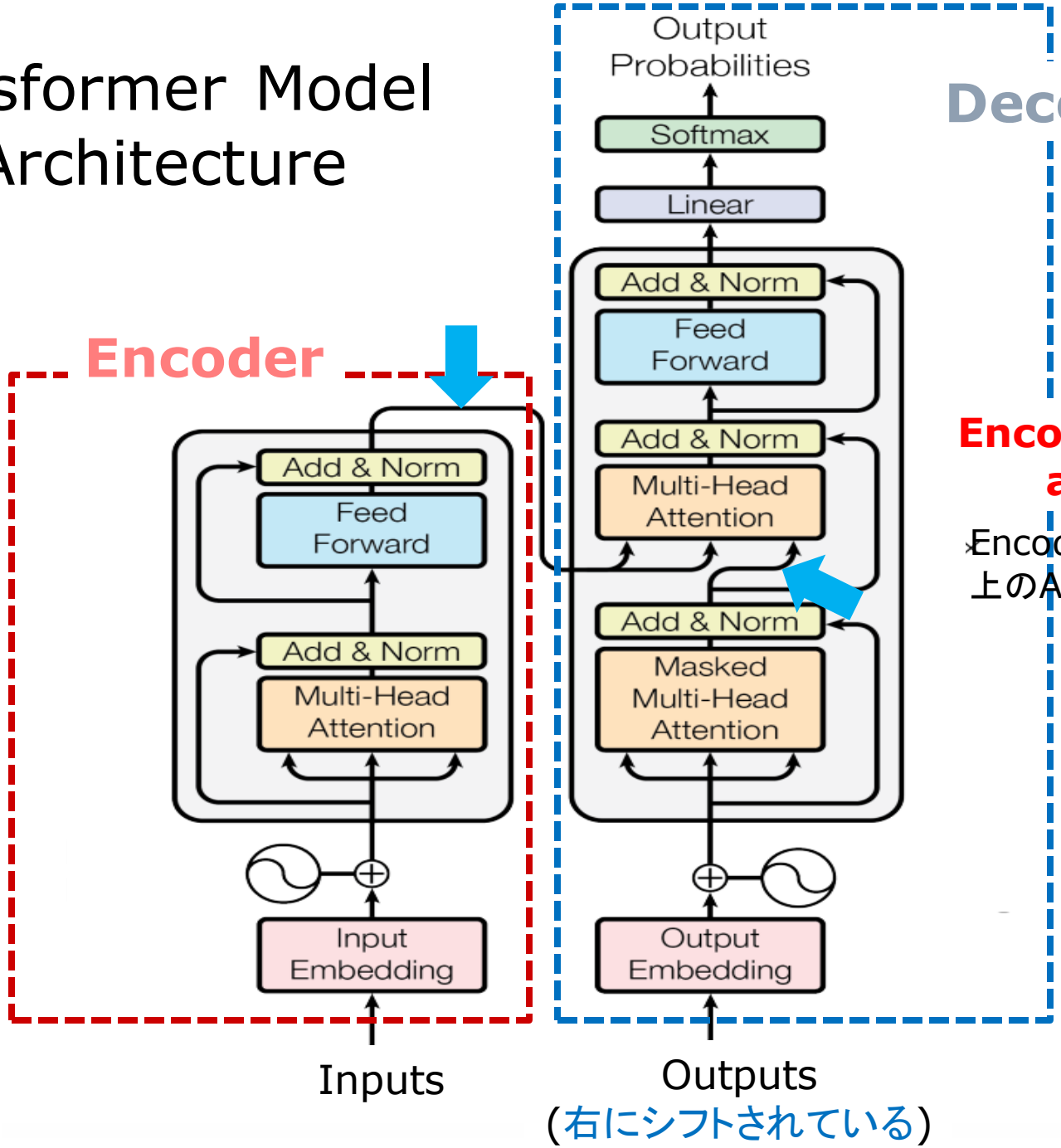
Transformer Model Architecture



Transformer Model Architecture



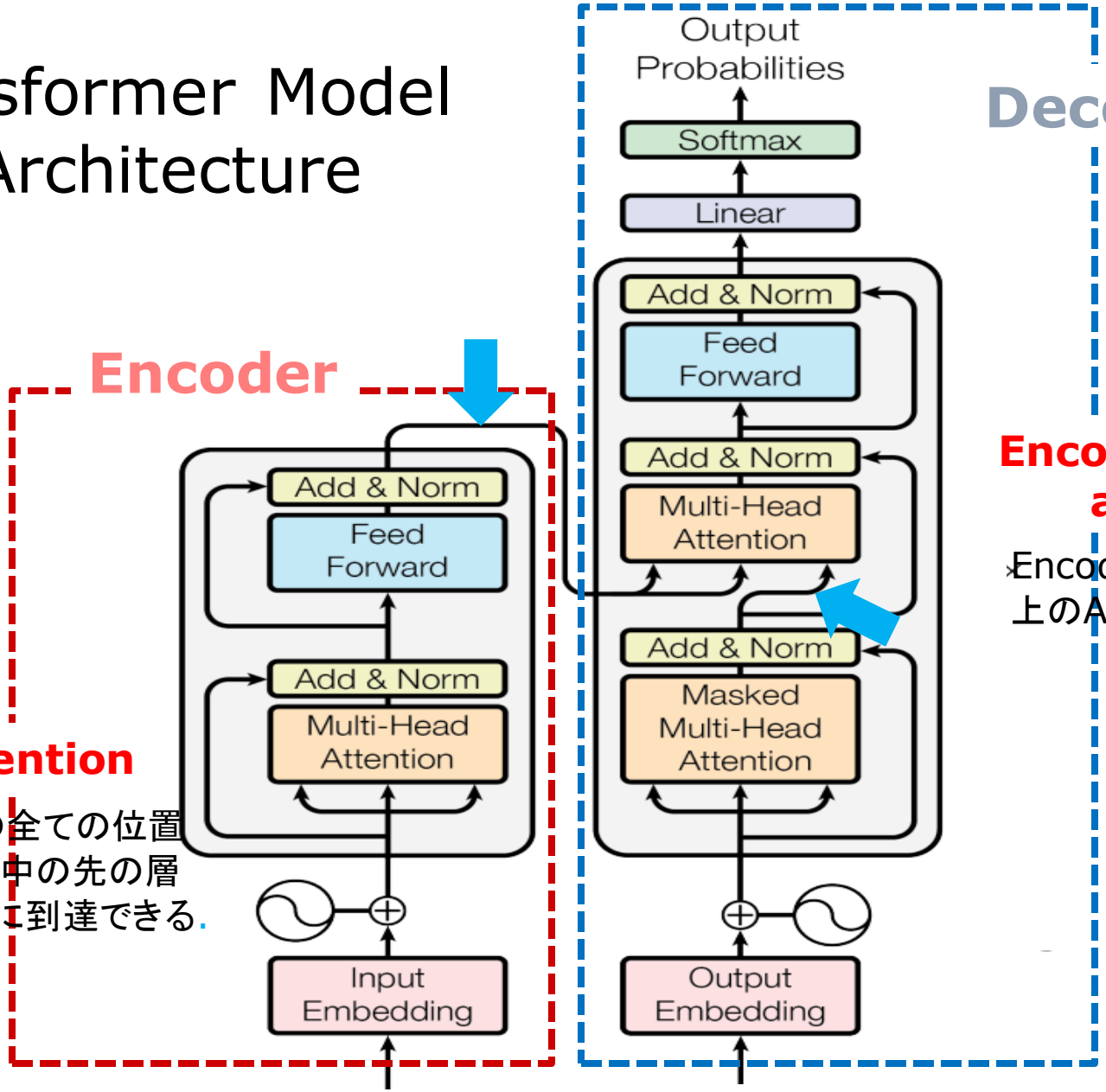
Transformer Model Architecture



Encoder-decoder attention

- Encoderスタックの出力上のAttention

Transformer Model Architecture



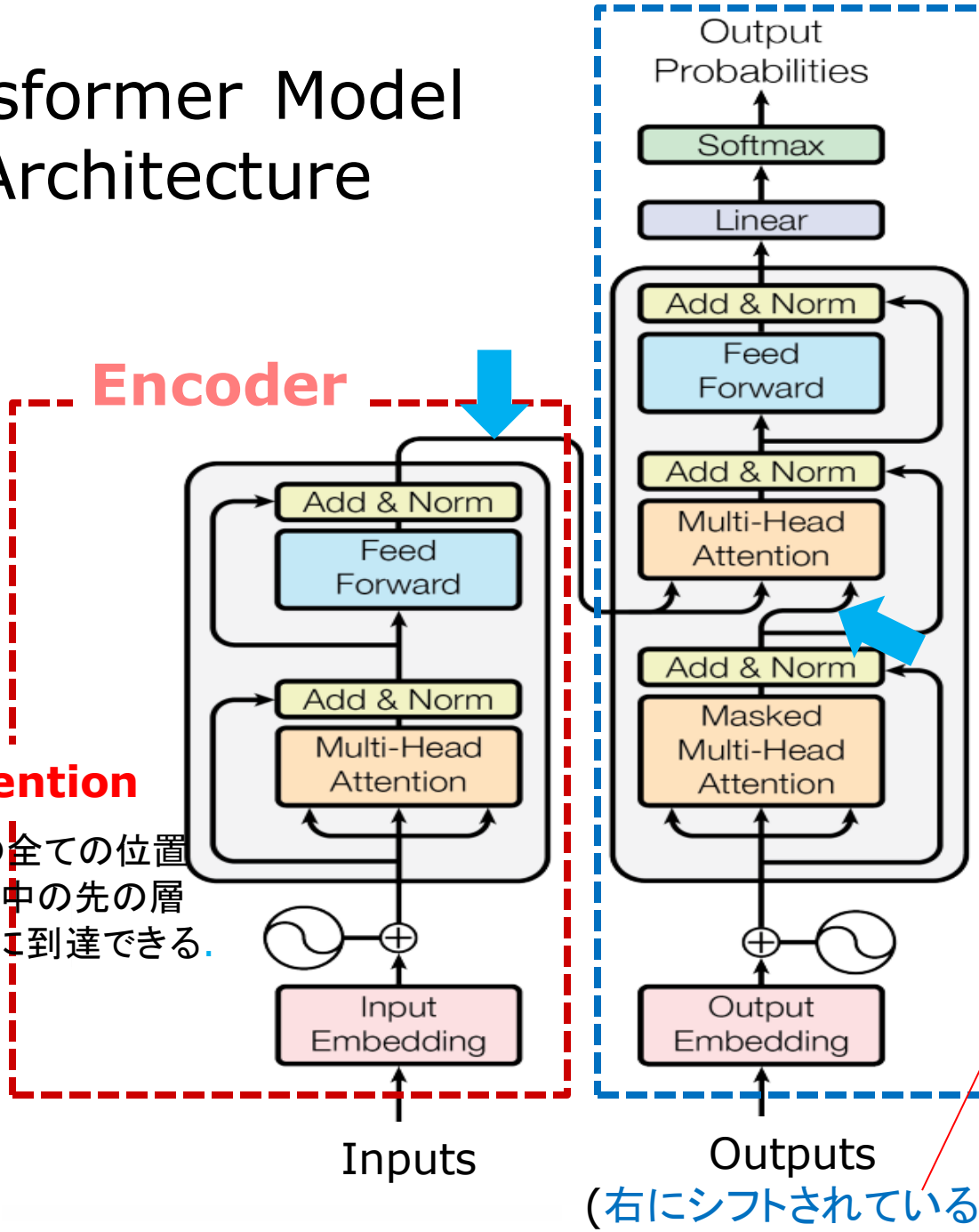
Encoder-decoder attention

Encoderスタックの出力上のAttention

Self attention

Encoder中の全ての位置は、encoder中の先の層の全ての位置に到達できる.

Transformer Model Architecture



Encoder

Decoder

Self attention

Encoder中の全ての位置は、encoder中の先の層の全ての位置に到達できる.

Encoder-decoder attention

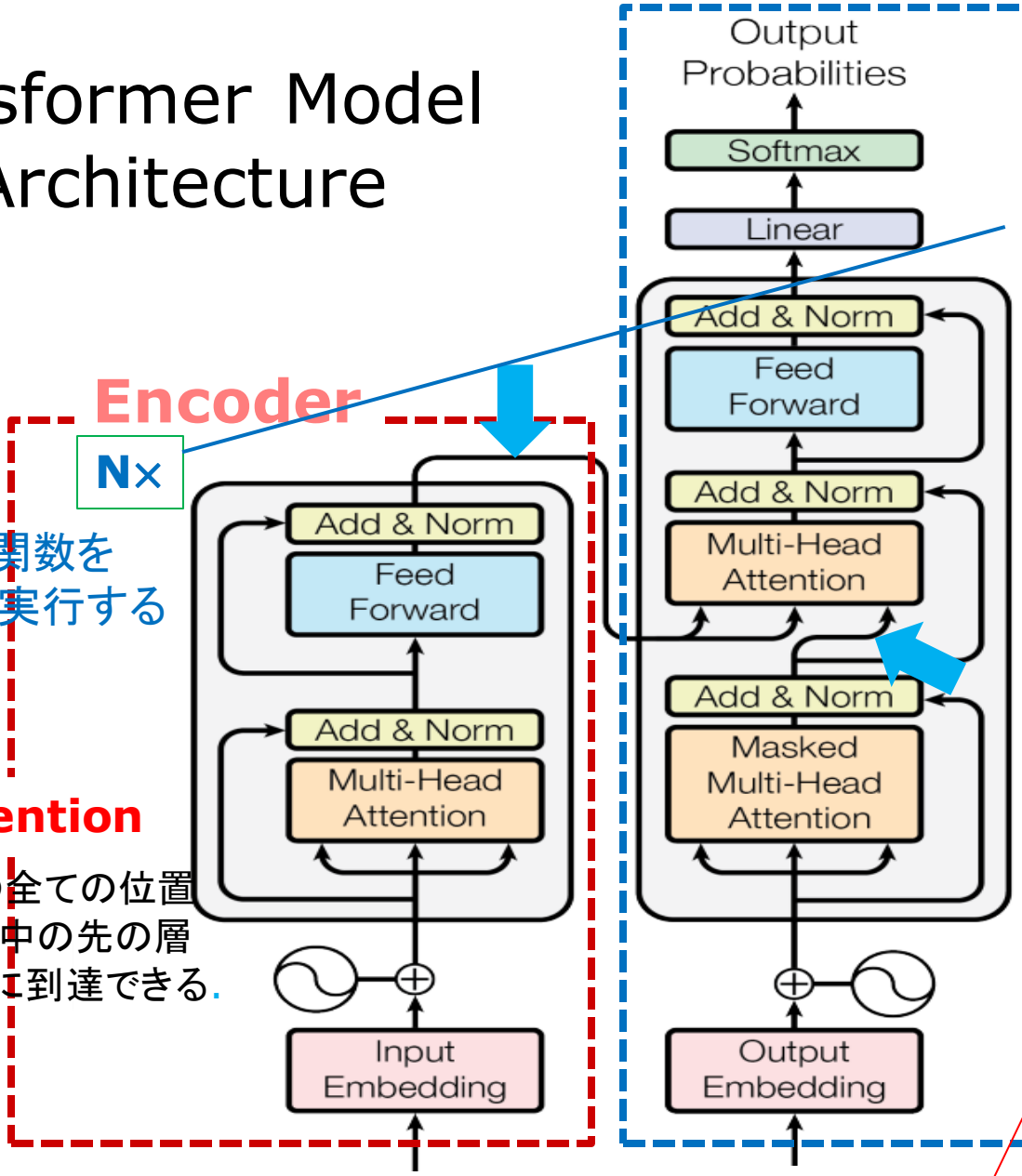
Encoderスタックの出力上のAttention

Masked Self attention

Decoderの中で、情報が左方向に流れるのを止める必要がある。

(右にシフトされている)

Transformer Model Architecture



Encoder

N×

Attention関数を
N個並列に実行する

Self attention

Encoder中の全ての位置は、encoder中の先の層の全ての位置に到達できる.

Decoder

N×

Attention関数を
N個並列に実行する

Encoder-decoder attention

Encoderスタックの出力上のAttention

Masked Self attention

Decoderの中で、情報が左方向に流れるのを止める必要がある。

(右にシフトされている)

Attention Is All You Need

Transfomer を提案した 2017年のVaswani らの論文は、"Attention Is All You Need" と名付けられていました。

「現在優勢なシーケンス変換モデルは、エンコーダーとデコーダーを含む、複雑なリカレントまたはコンボリューション・ニューラルネットワークに基づいている。

また、最も優れた性能を持つモデルは、アテンション・メカニズムを通じてエンコーダーとデコーダーを接続している。

我々は、RNNやCNNを完全に排除し、アテンション機構のみに基づく新しいシンプルなネットワークアーキテクチャ Transformer を提案する。」

Attention Mechanismに注目すると、そこには特にパフォーマンスの面で課題があることも浮かび上がってきます。それは、シーケンシャルに実行され並列化されていません。また、Attentionが注目する二つの点の距離が離れると離れるほど計算量が増大します。

"Attention is all you need" 論文は、Attention Mechanismのパフォーマンス改善の提案なのです。"Multi Head Attention" は、まさに、そうしたものです。

「Transformerでは、計算量を一定の演算数にまで減少させることができる。確かに、Attentionで重み付けされた位置の平均化によって有効解像度は低下するのだが、この効果は3.2節で説明するようにマルチヘッドAttentionで打ち消すことができる。」

もう一つ、この論文の大事な提案は、Self Attentionの重要性の指摘です。

「 Self Attentionは、intra-attentionと呼ばれることもあるのだが、シーケンスの表現を計算するために、単一のシーケンス内部の異なる位置を関連付けるAttentionメカニズムである。

Self Attentionは、読解、抽象的要約、テキストの含意、タスクに依存しない文章表現の学習など、様々なタスクでうまく利用されている。」

BERTは言語表現モデル

ここでは、「言語表現モデル」という言い方をしていることに注意したらいいと思います。それは、直接には、Google 機械翻訳のように自然言語処理の何らかの具体的な処理を担うモデルではありません。それ自体は、言語のコンピュータ上の「表現」の構成にもっぱら関わるモデルなのです。

そのアーキテクチャーは、前回見たTransformerの前段部のEncoder だけを独立させたようなものです。内部に、Multi headのSelf-Attention Mechanismを抱えています。

そのAttention Mechanismから「表現」を「構成」することが、BERTの「言語表現モデル」としての第一の仕事になります。

Pre-training とFine-tuning

Transformerから出力を担当するDecoderを切り取って、もっぱらEncoder内に「言語表現」を溜め込んで、どうするのでしょう？

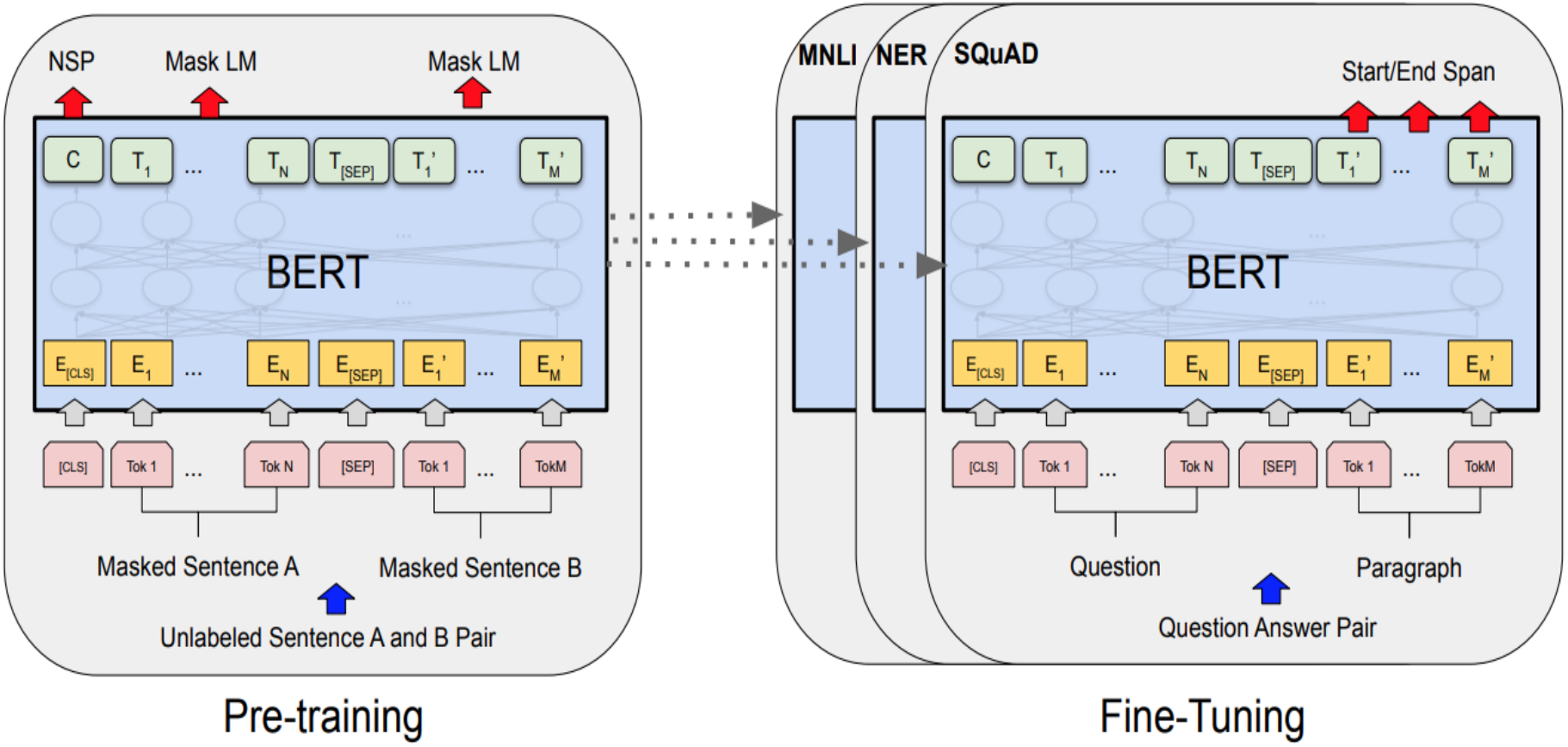
BERTでは、膨大な計算時間をかけて、Encoder内部に「言語表現モデル」を構成するまでの、この段階をPre-training と言います。

BERTで外部に対して何かの仕事をしようとするときには、Pre-training で構成された「言語表現」をそのまま使って、具体的なタスクを実行する出力層を一枚かぶせます。これを Fine-tuning と言います。

「働くBERT」は、Pre-training + Fine-tuning の形をしています。ただ、タスクが変わるたびに、長い計算が必要なPre-trainingの段階を繰り返し実行する必要はありません。

「事前に訓練済み」のPre-trained された「言語表現」は、何度も何度も、タスクが変わっても「使い回す」ことができるからです。

Pre-TrainingとFine-Tuning



BERT は文と文の繋がりを学習する

BERTは、Pre-trainingの段階で、文内部に置かれたマスクで隠された一つのブランクに、どの語が入るかを繰り返し学習します。

文Sの内部での、語w_i と語w_j の両者の間には、一つには語の意味のつながりの、もう一つには文法的なルールで規定される相関関係があります。

両者は、BERTの 文内部のSelf Attentionで検出され結合されて一つの確率分布で表現されます。

その相関関係は強いものです。

BERTにはもうひとつ際立った特徴があります。それは、二つの文の関係を把握しようとすることです。

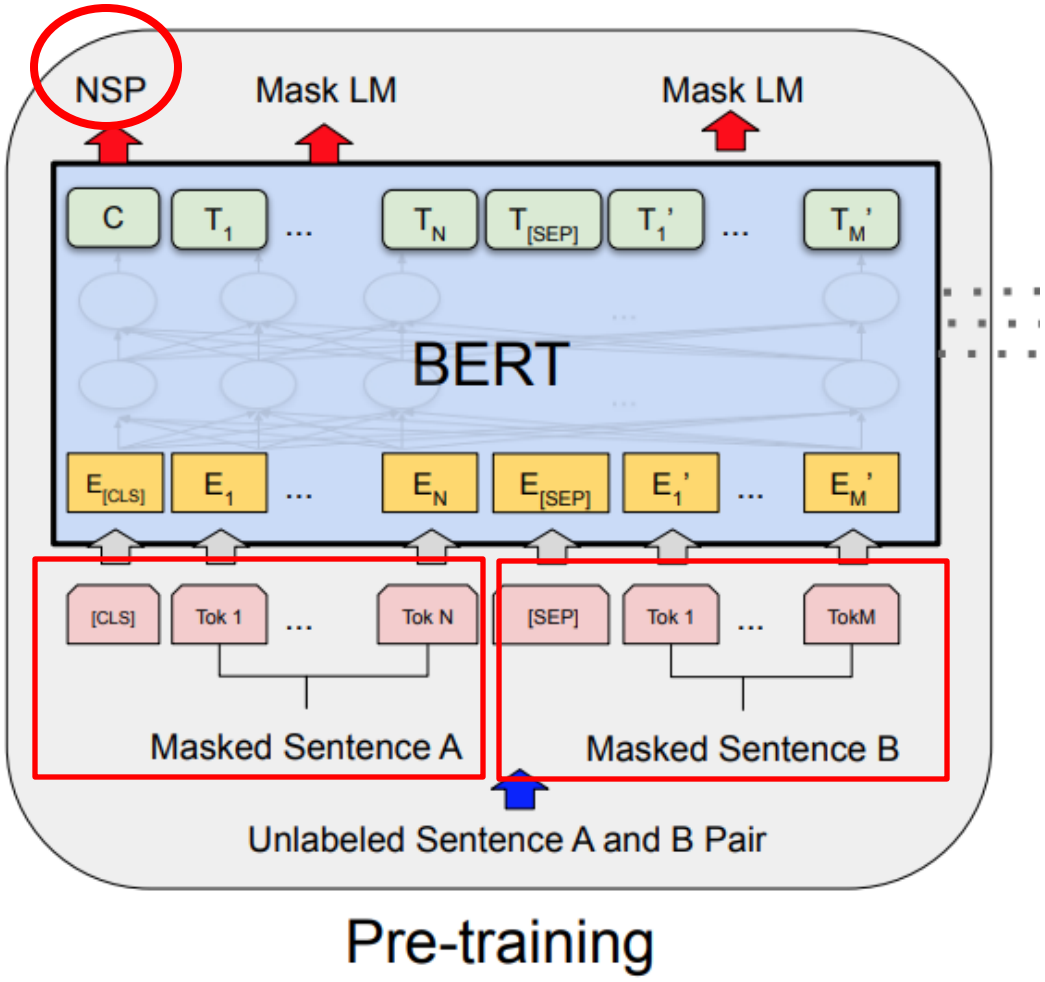
具体的には、BERTはPre-trainingの段階で、ある文Aともう一つの文Bが与えられたとき、文Aのうしろには文Bがつながる("IsNext")、あるいはつながらない("NotNext")ということを学習します。それをNext Sentence Prediction (NSP)といいます。

BERTがPre-trainingで行うこと

BERTのPre-trainingでは、次の二つのことを、徹底して学習します。

- 双方向での語の意味の表現
- 二つの文が与えられたとき、それが関連しているかの判断

Pre-training BERTのタスク
NSPの入力：Sentence Pair



- NSP:
  Next
  Sentence
  Prediction

IsNext/NotNext

文と文の関係の例

例えば、

Sentence A = The man went to the store.

Sentence B = He bought a gallon of milk.

だとすると、この二つの文は、つながっていると判断できますので、"IsNext"というラベルを出力するように訓練します。

Sentence A = The man went to the store.

Sentence B = Penguins are flightless.

だとすると、この二つの文は関連が薄いので、"NotNext" というラベルを出力するように訓練します。

Next Sentence Predictionが
もたらした強力な力

こうした文と文の関係は、語の意味の表現(Word2Vex)やLlyaの翻訳モデル(Sentence to Sentence)での文の意味の表現のような、個別の語、個別の文の意味表現のレベルでは、把握することはできません。

Next Sentence Prediction (NSP) の導入によって、翻訳モデルと大規模言語モデルでは、Sentence to Sentence の意味するところが違うものになりました。

翻訳モデルでは、Sentence to Sentenceは二つの同じ意味をもつ文の変換・生成だったのですが、大規模言語モデルでは、Sentence to Sentence は、二つの文の意味とは関係なしに、その文が "IsNext(つながる)"というラベルを持つ文の生成を行います。

翻訳モデルと大規模言語モデルの、一番大きな違いは、この Next Sentence Prediction を行う能力の有無にあると、僕は考えています。

NSPの導入は、大規模言語モデルに強力な力を与えました。

それは、最初の文が与えられれば、それと"IsNext"　関連づけられた無数の文の連鎖を生成する能力を大規模言語モデルに与えたのです。

何が NSP を可能にしたのか？

NSPを可能にしたのは、実は、簡単な技術的「工夫」によるものです。

それは、Transformer / BERT では、語の意味も語の集まりである文の意味も、同じ長さの固定長ベクトルで表すことにしたからです。

それればかりではありません。文の集まりであるDocument (あるいはContext)の意味も、同じ長さの固定長ベクトルで表されるのです。

(Attention を生み出すきっかけとなった、Bengio たちの固定長ベクトル批判は、どこへいったのでしょう？）

ただ、それにはメリットがあります。

例えば、ドキュメントの意味のベクトル表示は、ドキュメントの「要約」の作成で、本質的な役割を果たしています。

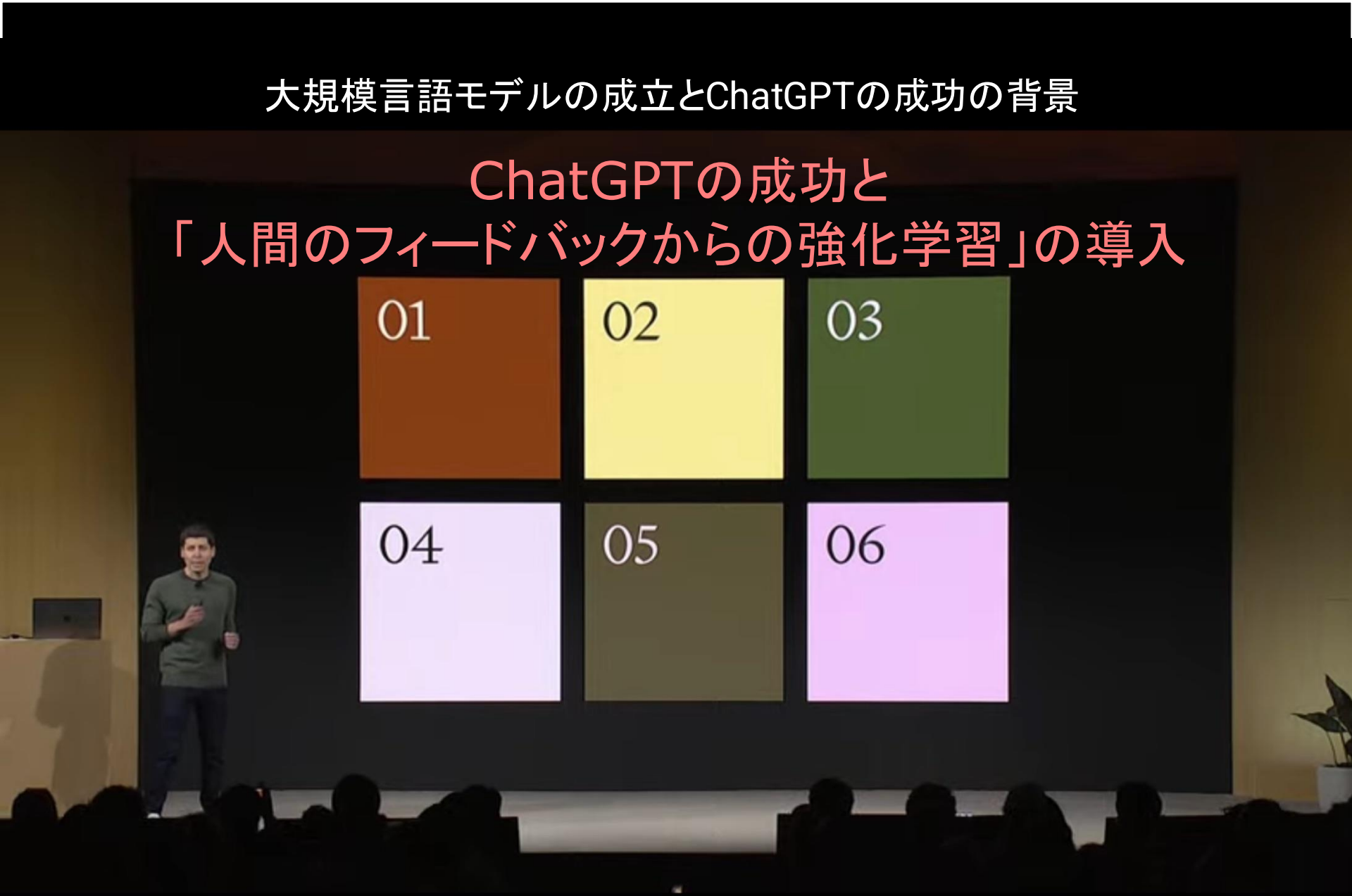
要約とは、意味が近くて短い文章の集まりを探すことなら、文章の集まりの意味の近さを比較することが必要なのですが、一番簡単なのは、意味のベクトルの次元が等しいとして、内積を計算することです。

もちろん、そこには代償があります。

大規模言語モデルで、「ハルシネーション」が最初に観察されたのが、長い文章の要約を作成するプロセスであったのは、偶然ではないのです。

大規模言語モデルの成立とChatGPTの成功の背景

ChatGPTの成功と「人間のフィードバックからの強化学習」の導入



11/24 角川セミナー

何がChatGPTの成功を可能にしたのか

このセッションでは、ChatGPTの成功の背景を考えていきたいと思います。

ChatGPT 登場の背景として最初に確認したいのは、これまで見てきた大規模言語モデルの自然な発展としてChatGPTが生まれてきた訳ではないということです。

翻訳モデルから大規模言語モデルへの転身をとげ、その後も順調な発展を遂げ、新しい領域へのチャレンジを開始した大型言語モデルですが、2022年の初めには、OpenAI, Googleの双方に、大規模言語モデルの前途には現状では解決が難しい課題があるという認識がうまれはじめます。

一つには、大規模言語モデルで、論理的・数学的問題を解くことは難しいという認識であり、もう一つには、大規模言語モデルの規模を拡大しても、モデルの能力は思うようには拡大しないだろうという認識です。

こうした困難に直面し、それを打開するために従来とは異なる新しい道の探求が始まり、その取り組みの中で生まれたのがOpenAIのChatGPT だったということです。

論理的・数学的問題を解くことの難しさ

論理的・数学的に問題を解く能力は、人間の知能を構成する重要な能力の一つです。自分の力に自信を持ち始めた大規模言語モデルが、この領域に挑戦を始めたのは、当然のことかもしれません。

GoogleのDeepMindは、プラグラムコンテストCodeforcesの問題を、Transformerを使って解くプロジェクト AlphaCode を立ち上げます。

同様に、OpenAIは、数学オリンピックの問題をGPTで解くプロジェクト Open-AI Theorem Prover を立ち上げます。

Googleの総括

ただ、結果は芳しいものではありませんでした。

AlphaCodeの論文は、結果を次のように語ります。

「AlphaCodeは、5,000人以上が参加したコンペティションで、平均54.3%の上位入賞を果たしました。」

これは失敗とは言えないかもしれませんが、「平均54.3%の上位入賞」は平凡なものです。

OpenAIの総括

OpenAIのTheorem Proverの総括は、もっとストレートで率直なものです。

「我々は、何度も我々のモデルが生成した証明手順の複雑さに、強く印象付けられてきた。

しかし、こうした推論ステップを必要とする証明の多くは、現在のコンピュータの能力の地平を超えたところにある。

我々が、たとえ挑戦的な数学オリンピックの選択された問題を解いたとしても、我々のモデルは、いまだこれらの競技の最も優秀な学生たちと競争するにははるかに遠い場所にいるのである。」

論理的・数学的問題を解くことの難しさは、大規模言語モデルの原理的なレベルでの限界に関係すると僕は考えています。

大規模言語モデルの規模の問題

従来の大規模言語モデルのChatGPTへの進化は、もう一つの言語モデルの規模の問題に関連して起きました。

まず、Googleの問題意識を見ておきましょう。

「多くのタスクで強力な性能を発揮するにもかかわらず、事前学習された言語モデルは、分布外の構成的一般化で苦労することが示されている。… モデルサイズを拡大することで、意味解析における構成的一般化も改善できるのだろうか？」

彼らは、11Bパラメータから540Bパラメータまでの様々な規模の言語モデルの性能を比較します。

その結果は、次のようなものです。

「我々は、意味解析評価の分布外構成的一般化において、fine-tuningは一般にフラットか負のスケーリングカーブを持つことを観察した。」

モデルの規模を拡大しても、意味解析評価の分布外構成的一般化は改善されず、むしろ劣化するということです。

「モデル規模によって異なるいくつかのエラー傾向も確認された。例えば、より大きなモデルは一般に出力の構文をモデル化するのに優れているが、ある種のオーバーフィッティングを起こしやすいということがある。」

この論文のlast author であるKristina Toutanovaは、BERT開発の中心人物の一人です。

OpenAIの規模の問題への回答

OpenAIの規模の問題への回答は、率直であると同時に、驚くべきものでした。

「言語モデルを大きくしても、ユーザーの意図に沿うようになるとは限らない。

大きな言語モデルには、真実味のない、有害な、あるいはユーザーにとって役に立たない出力を生成する可能性がある。別の言葉で言えば、これらのモデルはユーザー にそっていないのである。

本論文では、様々なタスクにおいて言語モデルをユーザーの意図に沿うようにする道は、人間のフィードバックを用いてモデルのfine-tuningを行うことにあることを示す。」

こうして、ChatGPTが誕生することになります。

人間のフィードバックによる強化学習

人工知能研究は、基本的には、機械による知能の実現を目指すものです。それは、人間の介在を必要としない、機械の自律的な知能を追求してきました、

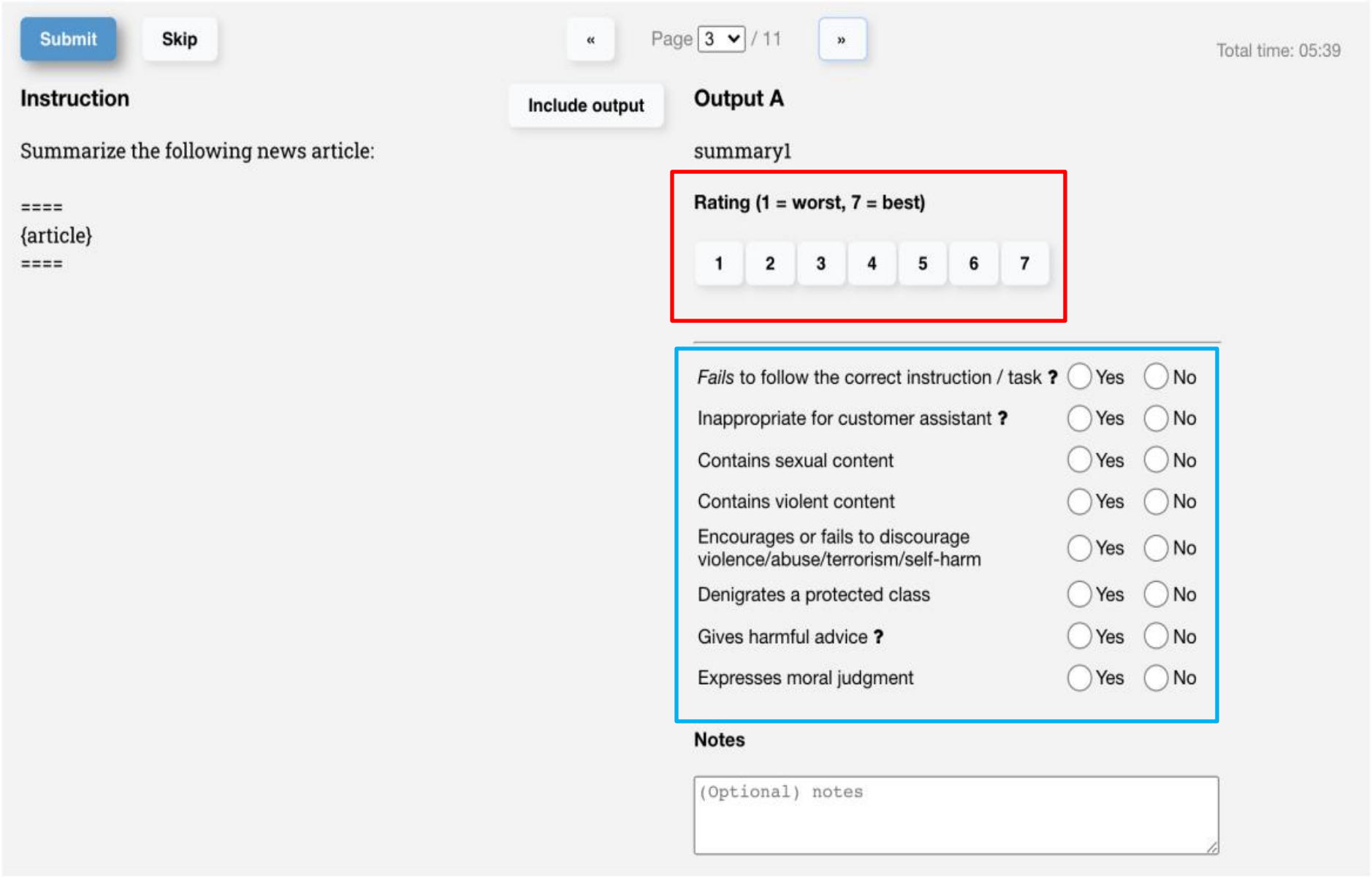
しかし、先に見たChatGPTの選択は、人工知能の実現に、積極的に人間のフィードバックを活かそうというものです。それは、従来の人工知能のアプローチとは、ずいぶんと違ったものです。

- ある質問に対して、機械が答えるのではなく、まず人間が答えて、それを機械に教えます。
- 次の段階では、ある質問に対して機械あるいは他の人間がが答えを用意するのですが、どれが一番相応しい答えなのかを、人間が選択します。
- これらの人間の選択を機械は学習します。
- この学習結果をコアとして、人間の振る舞いを模倣するようにシステムを拡大していきます。

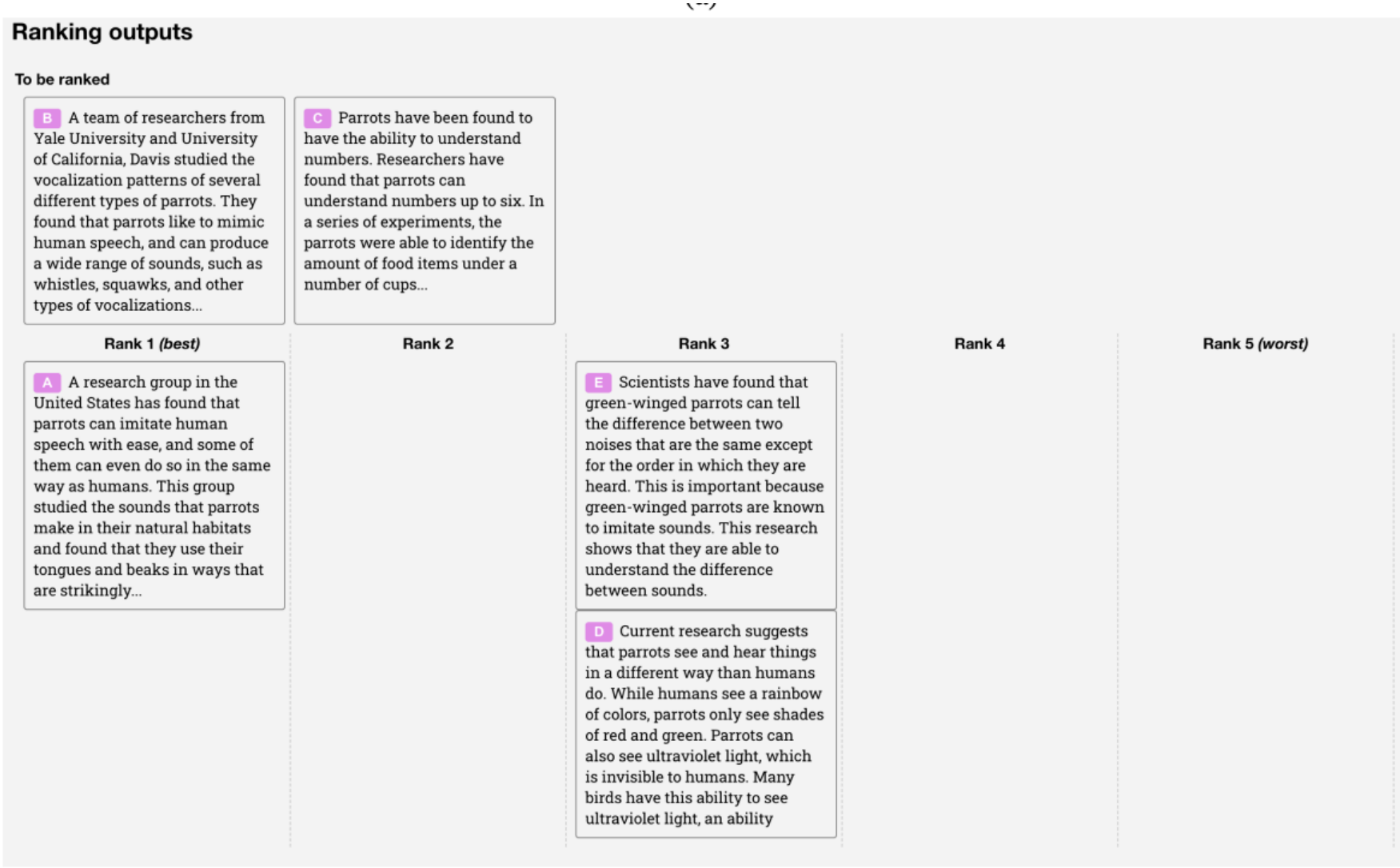
これを「人間のフィードバックによる強化学習」と呼びます。これがChatGPTの心臓部です。

この路線転換は大成功を収めます。

教師（Labeler）用の画面インターフェース

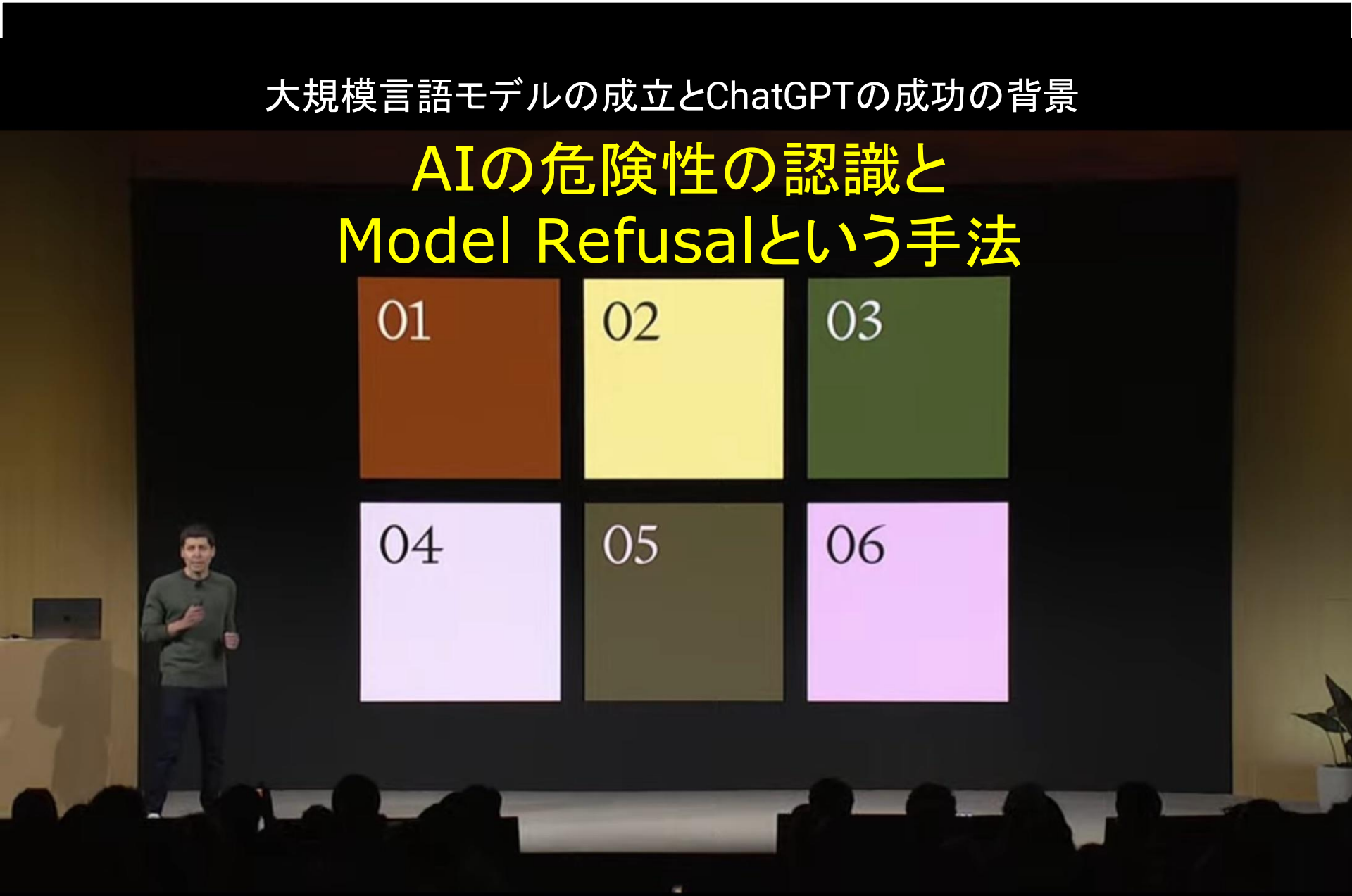


教師（Labeler）用の画面インターフェース



大規模言語モデルの成立とChatGPTの成功の背景

AIの危険性の認識と Model Refusalという手法



11/24 角川セミナー

Interlude -- AIと人間の関係を考える

これまで、ChatGPTの成功に至るまでの大規模言語モデルの成立とその発展を、主要には技術的な関心から振り返ってきました。それは過去の歴史の話です。

後半では「マルチモーダル化」と「カスタム化」という二つのトピックスにフォーカスして、ChatGPTがどのように変わっていくのかということを考えたいのですが、そこでも具体的には技術的な話が中心になります。それはAI技術の現在の話になるでしょう。

AIの未来を考えようとすると、それを単なる技術予測として語るのは適切なものではないと思います。AI技術が人間と社会の未来に大きな影響を与えるだろうと考えるならなおさらのことです。それは技術だけの問題ではないからです。

興味深いのは、技術の側から見ても「単なる技術」というくくりはAIの「技術的予測」にとっても狭いものかもしれないと思えることです。

もしも、ChatGPTの成功の要因のひとつが、「人間のフィードバックからの強化学習」という「技術」の採用にあるのなら、それは、現在のAI技術は人間の介在を必要としていると考えることもできるはずです。そして、それは正しい認識だと僕は考えています。

AIの安全性をめぐって
OpenAI の隠れた優位性

AIの安全性をめぐる議論は、まさに、AIと人間の接点の問題です。この問題は、AI技術が社会的に受け入れられ、AIビジネスが経済的に成功するためにも、今以上に重要な課題になっていくと思います。AI開発の競争の焦点は、言語モデルの規模の大きさから、AIシステムの安全性に移っていくと思います。

AIの安全性をめぐる議論は、AIの危険性をめぐる議論に他なりません。AIを安全なものにするためには、その危険性を知らないといけないはずです。

OpenAIについて、我々はその技術的優位性に目が行きがちなのですが、これらのAIを安全なものにする取り組みで、OpenAIが、圧倒的に進んでいることは注目に値します。

OpenAIの安全性への取り組み

OpenAIは、訓練用データから性的コンテンツを人手で除去し、不適切な回答を人間がチェックする安全に関連するRLHFトレーニングプロンプトの見直しを進めています。

また、社外の多数の専門家とも連携して、危険性の徹底的な洗い出しを行なっています。"GPT-4 System Card" というドキュメントは、そうした検討の集大成です。

さらに、Red TeamというOpenAIのAIシステムを「攻撃」するチームを結成して、問題の洗い出しを不断に継続しています。

文字通り、「人間のフィードバック」によるAIシステムの改善に、人手と時間をかけて取り組んでいます。皮肉なことかもしれないですが、AIの危険性を誰よりも良く知っているのは、OpenAI自身です。

OpenAIの安全性確保の中心的手法
Model Refusal

OpenAIの安全性対策で特筆すべきは、こうした時間とコストがかかる人間による作業と並行して、よりきめ細かいレベルでモデルを適切な動作に導くために、モデル自身をツールとしてコンテンツのチェックの自動化を進めていることです。それを、rule-based reward models (RBRMs) といいます。基本的には、有害なコンテンツの生成を拒否するのに利用されています。

正確にいうと、外からは、モデルが有害コンテンツの生成を止めたように見えるかもしれませんが、実際に行われているのは、モデルが生成した有害なコンテンツを、モデルと人間の中間にあるシステムが受け取りを拒否して、人間に渡さないようにするのです。

いわば、モデルが有害なコンテンツを生み出すという問題には手をつけず、それが表には出てこないように蓋をするということです。

これをModel Refusal と言います。OpenAIの安全性確保の中心的手法です。

あるいはこうした方法しかないのかもしれません。あるいは、それはAIの問題は機械的に解決するのが、経済的には一番合理的だという考えの現れかもしれません。

OpenAIにも曖昧さはある

先に見たGPT-4 System Cardの「結論」部分は、こうしたアプローチに対する楽観的とも悲観的とも取れる「両儀的」なものに、僕は感じました。

「実際、これらのシステムをどのように統治するのが最善か、それらが生み出す利益をどのように公平に分配するか、アクセスをどのように公平に共有するか、に取り組むための先行予測研究がない限り、AIシステムがそうすることを期待すべきだろう。」

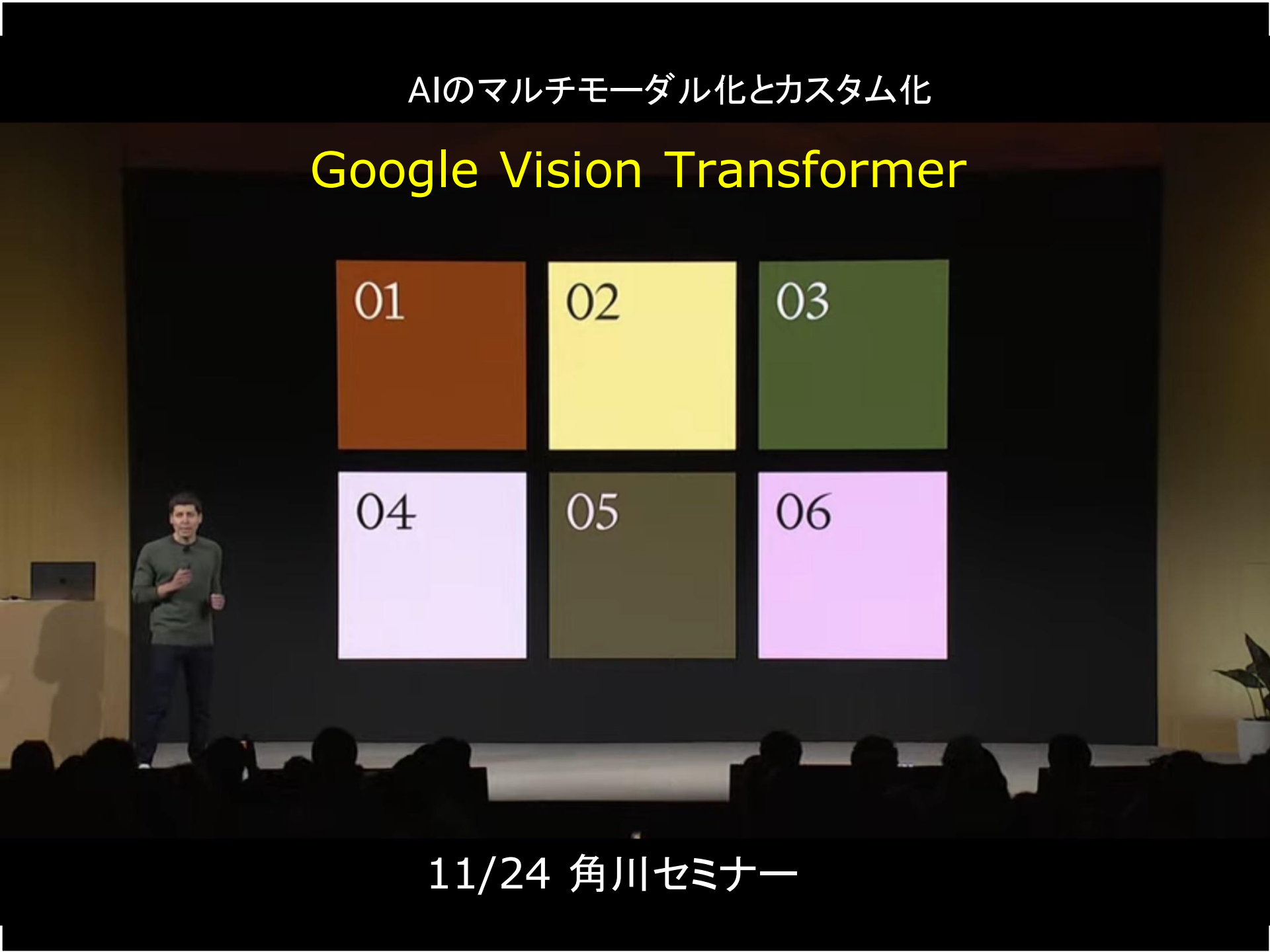
本当でしょうか？

こうした曖昧さは、このGPT-4 System Cardという重要な文書の公開の仕方自体にも表れていました。この文書は、独立の文書としては、なぜか公開されなかったのです。


AIのマルチモーダル化とカスタム化

AIのマルチモーダル化とカスタム化

Google Vision Transformer



11/24 角川セミナー

大規模言語モデルから
Multimodalな人工知能へ

現在の人工知能技術の技術的な焦点の一つは、「Multimodalな人工知能」の実現にあります。このセッションでは、大規模言語の上に Multimodalな人工知能を実現しようとする動きを紹介しようとおもいます。

マルチモーダルな人工知能とは、現在のテキスト中心の人間と人工知能のインターフェースを大きく変える「見ることも聞くことも話すこともできる」インターフェースを備えた人工知能のことです。

ただ、AIが「聞くこと話すこと」と比べて、AIが「見ること」を実現するのは技術的には様ざまな難しさがあります。ですから、マルチモーダルなAIを目指す技術の大きな関心は、AIが「見ること」の実現にむけられていると僕は考えています。

Vision Transformer とは何か？

大規模言語モデルがMulti-Modal なAI に展開して上で、大きな役割を果たしたシステムがあります。それが、2021年に Google が発表した Vision Transformer です。

自然言語処理の世界では、Transformerベースの大規模言語モデルが大きな成功を収めていたのですが、画像情報処理の世界では、近年に至るまで CNN (Convolution Neural Network)が主流でした。

それに対して、GoogleのVision Transformer は、大規模な画像情報処理の世界でも、CNNを全く利用せずに、Transformerだけで最先端のCNNのシステムを上回る性能を発揮できることを示しました。

このことは、Transformerをエンジンとする一つのシステムで、自然言語処理と画像処理のタイプの異なる二つの処理が同時に可能になることを意味しています。

Vision Transformer が、Multi-ModalなAIへの突破口となったというのは、そういうことです。

Vision Transformer のアーキテクチャー

Vision Transformerが自然言語だけではなく、画像も処理できるのは、次のような手法を用いているからです。

「元の画像を小さな画像パッチに分割し、これらのパッチの線形なembeddingのシーケンスをTransformerへの入力として提供する。」

画像パッチは、自然言語処理アプリケーションにおけるトークン（単語）と同じように扱われ、教師あり方式で画像分類モデルを学習します。

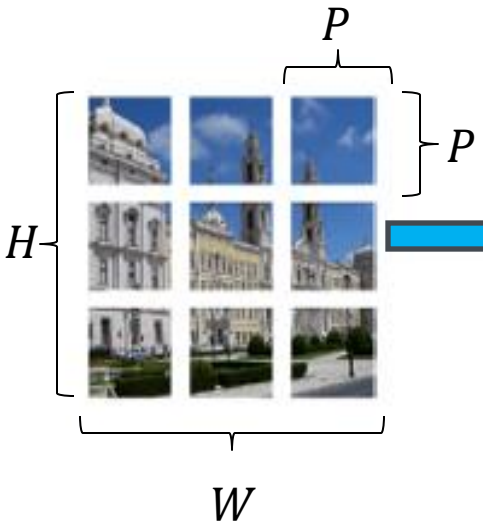
論文タイトルの "An Image Is Worth 16x16 Words" というのは、このことを指しています。

注目すべきことは、この画像のembedding の方法を除いては、Vision Transformer は、元のTransformerの実装を、可能な限り修正しないようにしています。

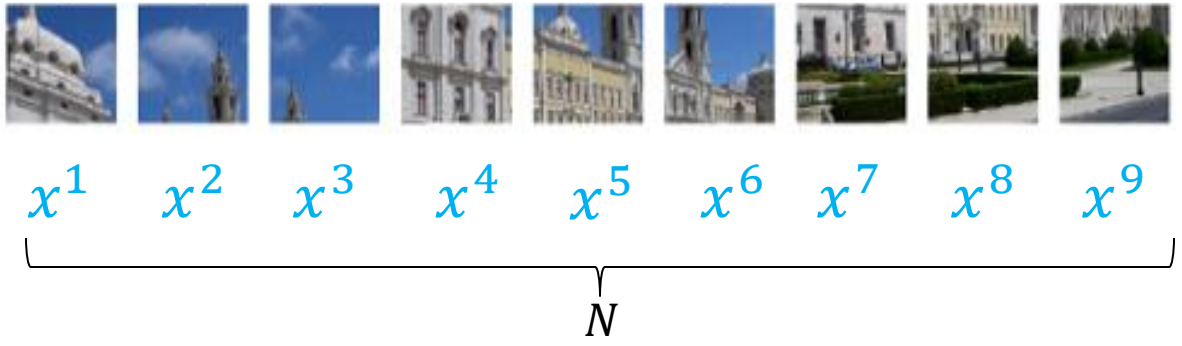
ですから、もしも、自然言語処理での標準的なTransformerの実装を知っていれば、この画像のembedding の方法さえ理解すれば、ほとんど、Vision Transformer の振る舞いを理解できることになります。

画像 x をパッチx^iのシークエンスへ

画像 x



パッチx^iのシークエンス



$N = HWC \,/\, P^2\mathrm{C}$

$x \in \mathbb{R}^{H\times W\times C}$

Cは(R, G, B)等の
カラーチャンネル

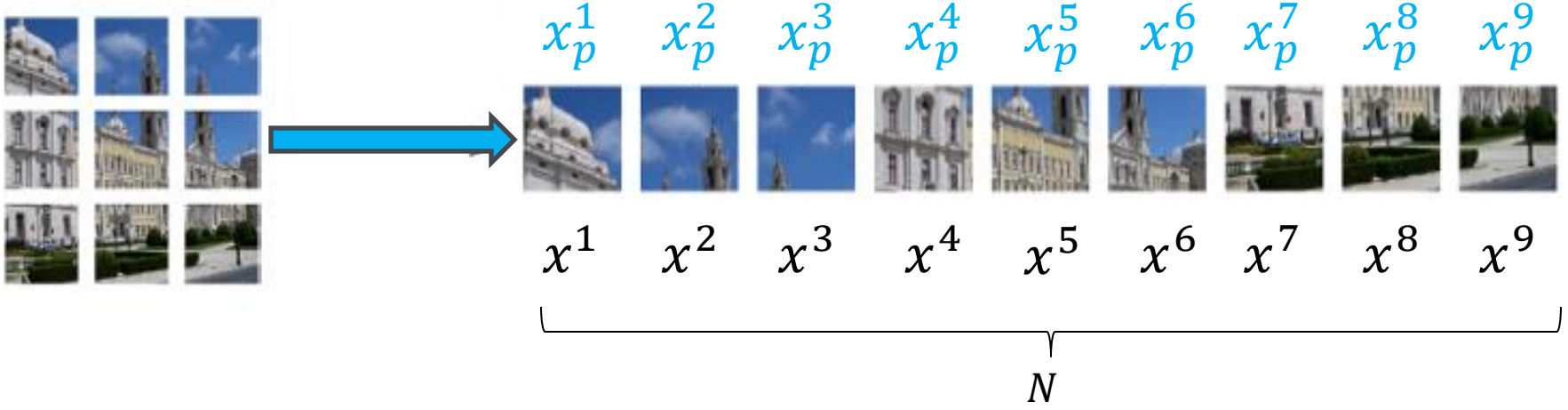
$x_p \in \mathbb{R}^{N\times(P^2\cdot C)}$

P^2C次元の行ベクトルのN個の並びへ

$x_p \in \mathbb{R}^{N \times (P^2 \cdot C)}$

$x_p^i \in \mathbb{R}^{1 \times (P^2 \cdot C)}$

パッチ x^iのP^2C 個のピクセルからなる
行ベクトルx_p^iのN個の並び



画像 x

パッチx^iのN個のシークエンス

D次元のembeddingを構成する

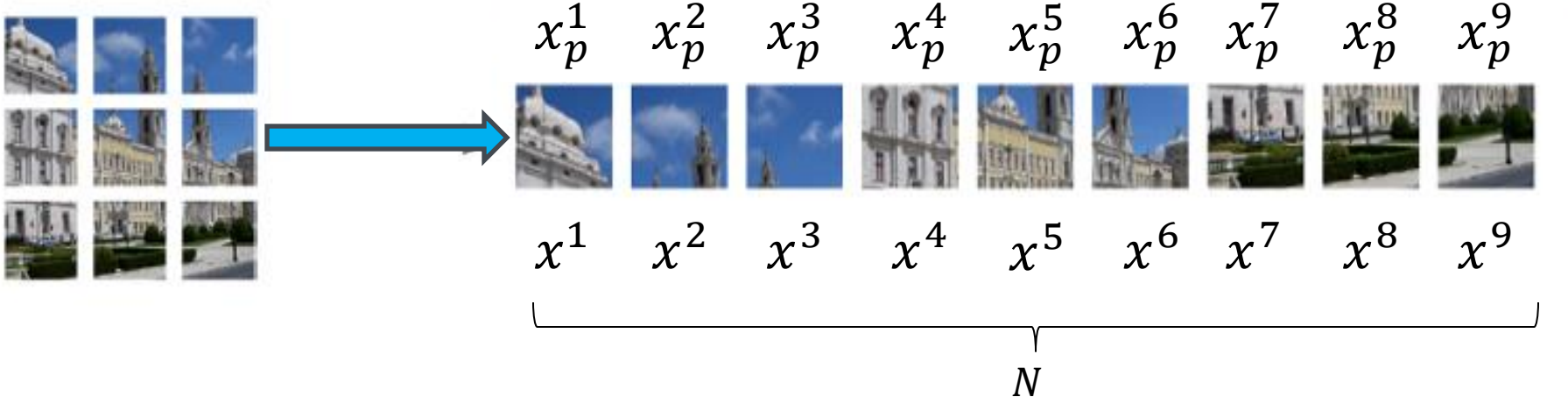
Linear Projection of Flattened Patches

$x_p^i \in \mathbb{R}^{1\times(P^2\cdot C)}$

$E \in \mathbb{R}^{(P^2C)\times D}$

$x_p^i E \in \mathbb{R}^{1\times D}$

D次元の行ベクトルのN個の並び



画像 x

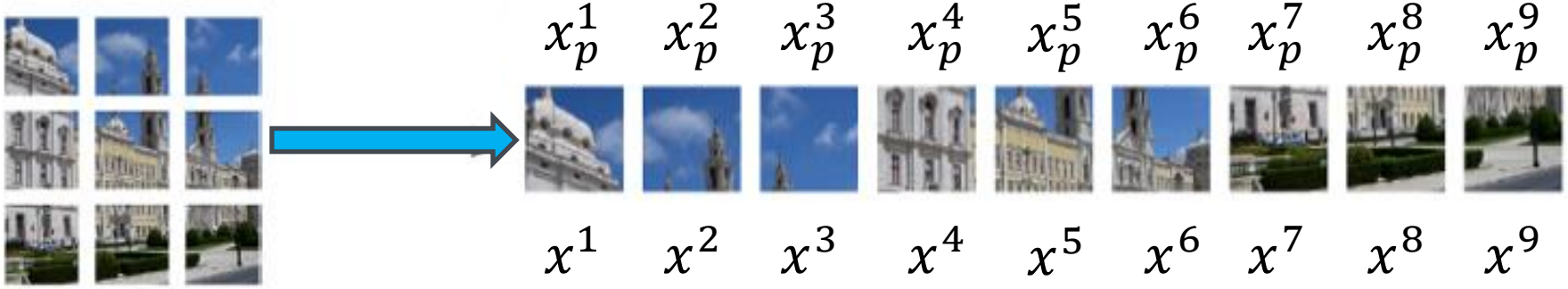
パッチx^iのN個のシークエンス

学習可能な [class]トークンの追加

$$[\, x_{class}; x_p^1 E; x_p^2 E; x_p^3 E; x_p^4 E; x_p^5 E; x_p^6 E; x_p^7 E; x_p^8 E; x_p^9 E \,]$$

$x_p^1 E$ $x_p^2 E$ $x_p^3 E$ $x_p^4 E$ $x_p^5 E$ $x_p^6 E$ $x_p^7 E$ $x_p^8 E$ $x_p^9 E$

Linear Projection of Flattened Patches



画像 x

パッチx^iのN個のシークエンス

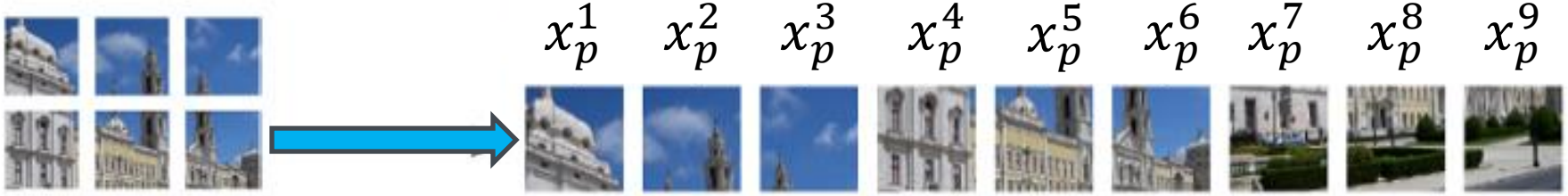
位置埋め込みの追加

$$[\, x_{class};\ x_p^1E; x_p^2E; x_p^3E;\ x_p^4E; x_p^5E; x_p^6E; x_p^7E; x_p^8E;\ x_p^9E \,] + x_{pos}$$

$$[\, x_{class}; x_p^1E; x_p^2E; x_p^3E;\ x_p^4E; x_p^5E; x_p^6E; x_p^7E; x_p^8E;\ x_p^9E \,]$$

x_p^1E x_p^2E x_p^3E x_p^4E x_p^5E x_p^6E x_p^7E x_p^8E x_p^9E

Linear Projection of Flattened Patches



x_p^1 x_p^2 x_p^3 x_p^4 x_p^5 x_p^6 x_p^7 x_p^8 x_p^9

x^1 x^2 x^3 x^4 x^5 x^6 x^7 x^8 x^9

N

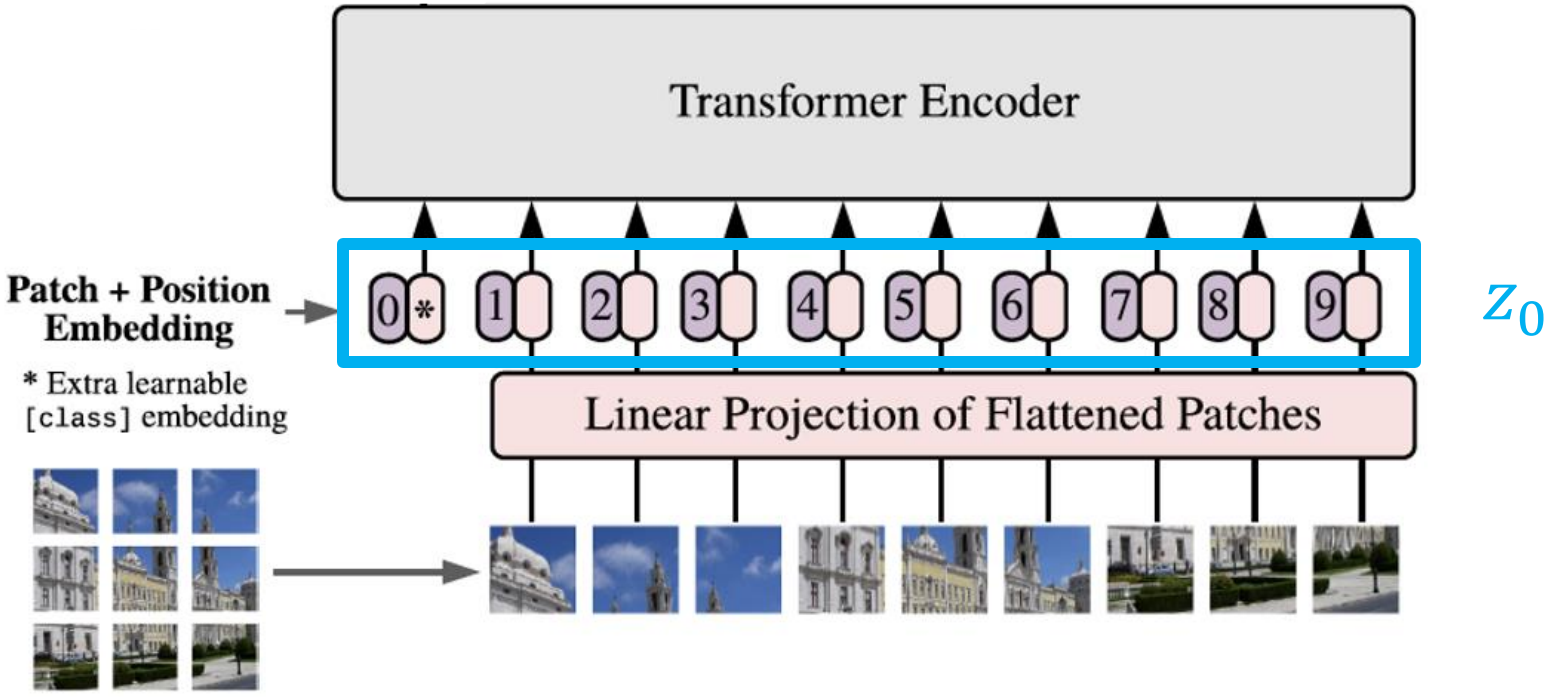
画像 x

パッチx^iのN個のシークエンス

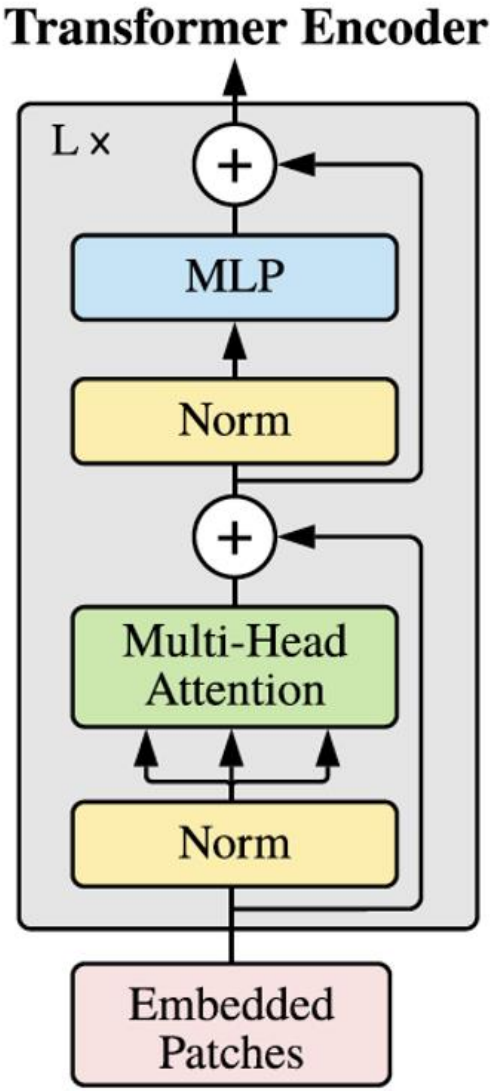
エンコーダへの入力 z_0

$$z_0 = [\, x_{class}; x_p^1 E; x_p^2 E; x_p^3 E; x_p^4 E; x_p^5 E; x_p^6 E; x_p^7 E; x_p^8 E; x_p^9 E \,] + x_{pos}$$

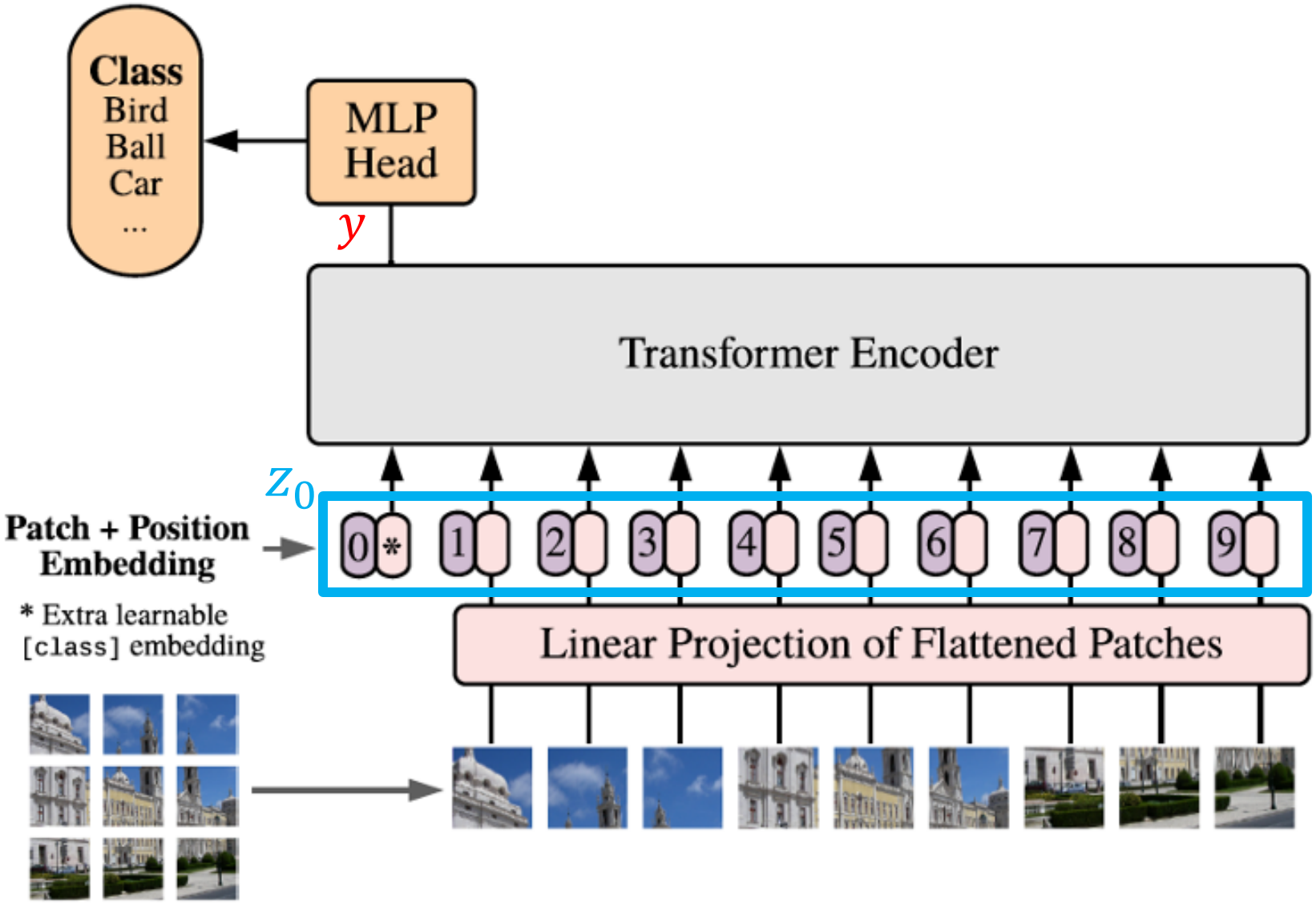
一般にエンコーダへの入力　$z_0 = [\, x_{class}; x_p^1 E; x_p^2 E; \cdots; x_p^N E \,] + x_{pos}$



Transfomer Encoder



Transformerエンコーダは、Multi-head Self Attention（MSA)とMLP(Multi Layer Perceptron)ブロック（の交互の層で構成される。各ブロックの前にはLayer-norm（LN）が適用され、各ブロックの後には残差接続が適用される。MLPにはGELU非線形性を持つ2つの層が含まれる。


Class
Bird
Ball
Car
...
MLP Head
y
Transformer Encoder
z_0
Patch + Position Embedding
* Extra learnable [class] embedding
0 * 1 2 3 4 5 6 7 8 9
Linear Projection of Flattened Patches

Vision Transformerが登場した時代

留意して欲しいのは、2021年のこのモデルが、Transformerを搭載した大規模言語モデルの規模拡大による快進撃の中で生まれたことです。

「Transformersの計算効率とスケーラビリティのおかげで、100Bを超えるパラメータを持つ前例のないサイズのモデルを訓練することが可能になった（Brown et al.）モデルとデータセットが増大する中、性能が飽和する兆候はまだない。」

「NLPにおけるTransformerのスケーリングの成功に触発され、我々は標準的なTransformerを、可能な限り少ない修正で、画像に直接適用する実験を行う。」

Vision Transformerが発見したこと

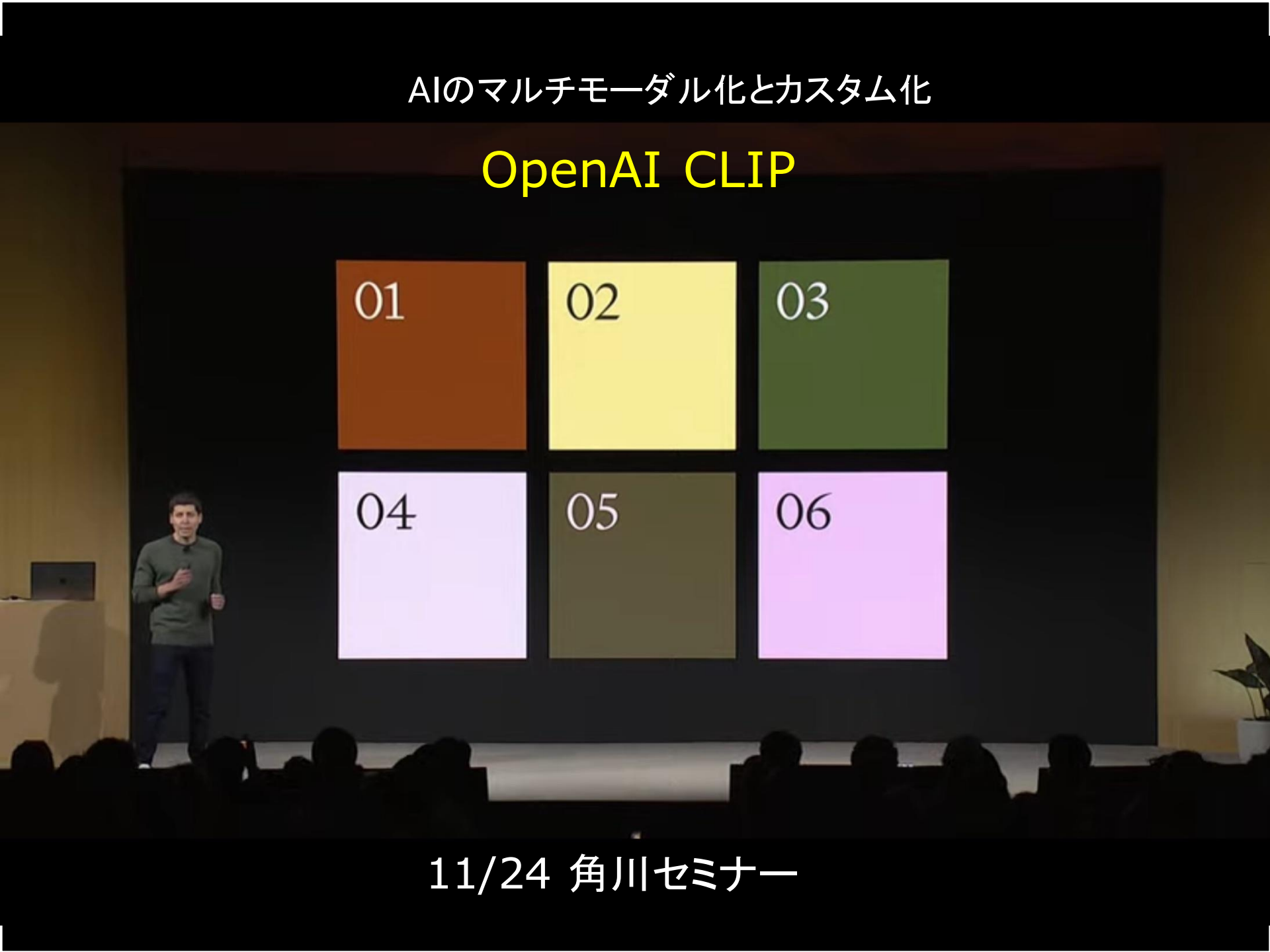
「ImageNetのような中規模のデータセットを強力な正規化なしで学習した場合、これらのモデルの精度は、同程度のサイズのResNetsを数%下回る。

この一見がっかりするような結果は予想通りかもしれない：Transformerは、変換の等価性や局所性といったCNNに固有の帰納的バイアスのいくつかを欠いているため、十分な量のデータで訓練してもうまく汎化できない。

しかし、より大規模なデータセット（1,400万～3,000万画像）でモデルを学習させると、様相は一変する。我々は、大規模訓練が帰納的バイアスに勝ることを発見した。」

AIのマルチモーダル化とカスタム化

OpenAI CLIP



11/24 角川セミナー

OpenAIのCLIPのアプローチ

CLIPは、GoogleのVision Transfomer のすこし後に、OpenAIによって公開された「テキストとイメージを結合する」を目標とするプロジェクトです。

それは、「見ることも聞くことも話すこともできる」ChatGPTとして最近公開されたGPT-4Vや、テキストから自由に画像を生成することのできるDall E-3の基礎技術です。

OpenAIのCLIPの一つの特徴は、現在のコンピュータによる画像処理技術の現状に満足できないことを率直に語ることから始めていることです。

「ディープラーニングはコンピュータ・ビジョンに革命をもたらしたが、現在のアプローチにはいくつかの大きな問題がある。」

最大のものは、データセットの問題だとOpenAIは言います。

先に見た Vision Transformer は、”Inductive Bias Free”なシンプルなアーキテクチャーでも、データセットの規模を拡大すると、画像認識の性能を上げられることを強調し、「大規模訓練が帰納的バイアスに勝ることを発見した。」と豪語していたのですが、OpenAIのCLIPのアプローチは、すこし違ったものです。

「典型的なビジョン・データセットは、作成に労力とコストがかかる一方で、狭い範囲の視覚概念しか教えない。標準的なビジョン・モデルは、1つのタスクと1つのタスクにしか向いておらず、新しいタスクに適応させるためには多大な労力を必要とする。」

「また、ベンチマークでは優れた性能を発揮するモデルも、ストレス・テストでは失望するほど低い性能しか発揮できず、コンピュータ・ビジョンへのディープラーニング・アプローチ全体に疑問を投げかけている。」

なかなか辛辣です。

「我々はこのような問題を解決することを目的としたニューラルネットワークを発表する。」

それがCLIPだといいます。

「それは、インターネット上に豊富に存在する多種多様なnatural language supervisionを用いて、多種多様な画像で学習される。これは重要な変更点である。」

GoogleとOpenAIで、少しマルチモーダルAIの実装の方向性について、違いがあることは、留意してもらえたらと思います。

はじめに
画像処理処理の現状

しかし、コンピュータビジョンのような分野では、ImageNetのようなクラウドラベル付きデータセットでモデルを事前学習するのが標準的なやり方である。

ウェブテキストから直接学習するスケーラブルな事前学習法は、コンピュータビジョンにおいても同様のブレークスルーをもたらすのだろうか？

さまざまな、有望な先行研究がある。

はじめに
本研究の課題

本研究では、大規模なnatural language supervisionで訓練された画像分類器の振る舞いを研究する。

インターネット上で公開されている大量のこの形式のデータを利用し、4億の（画像とテキストの）ペアからなる新しいデータセットを作成し、ゼロから学習したCLIP（Contrastive Language-Image Pre-training）が、natural language supervisionから学習する効率的な手法であることを実証する。

我々は、ほぼ2桁の計算量に及ぶ一連の8つのモデルを訓練することによってCLIPのスケーラビリティを研究し、転送性能が計算量の滑らかに予測可能な関数であることを観察する。

はじめに
成果

- 我々は、CLIPがOCR、ジオロカライゼーション、行動認識、その他多くのタスクを含む幅広いタスクを事前学習中に学習することを発見した。

- 先行するタスク固有の教師ありモデルと競合できることを発見した。CLIPが計算効率に優れながら、公開されている最良のImageNetモデルを上回ることを示す。

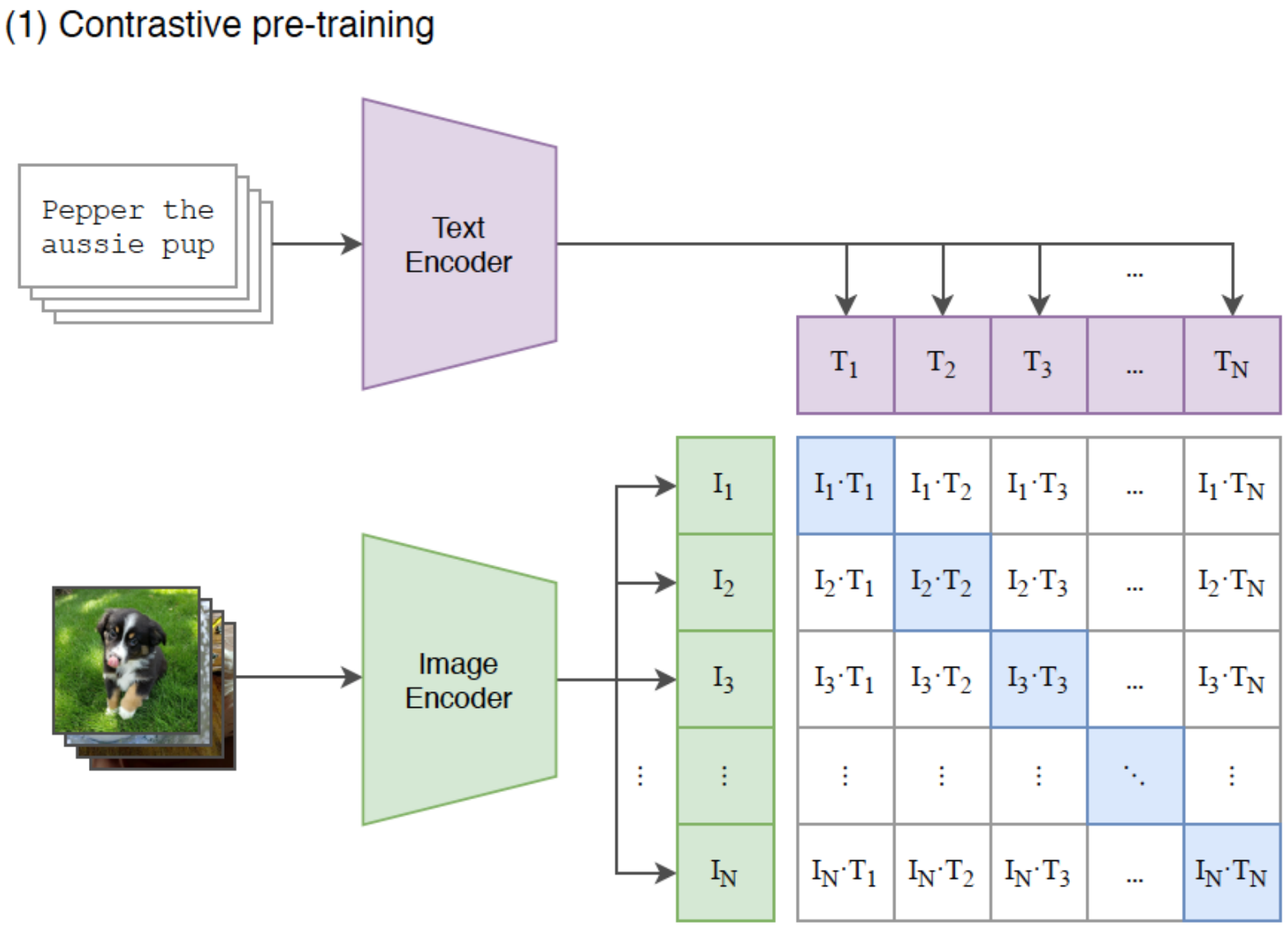
- さらに、ゼロショットCLIPモデルは、同等の精度を持つ教師ありImageNetモデルよりもはるかにロバストであることを発見した。

我々のアプローチの概要

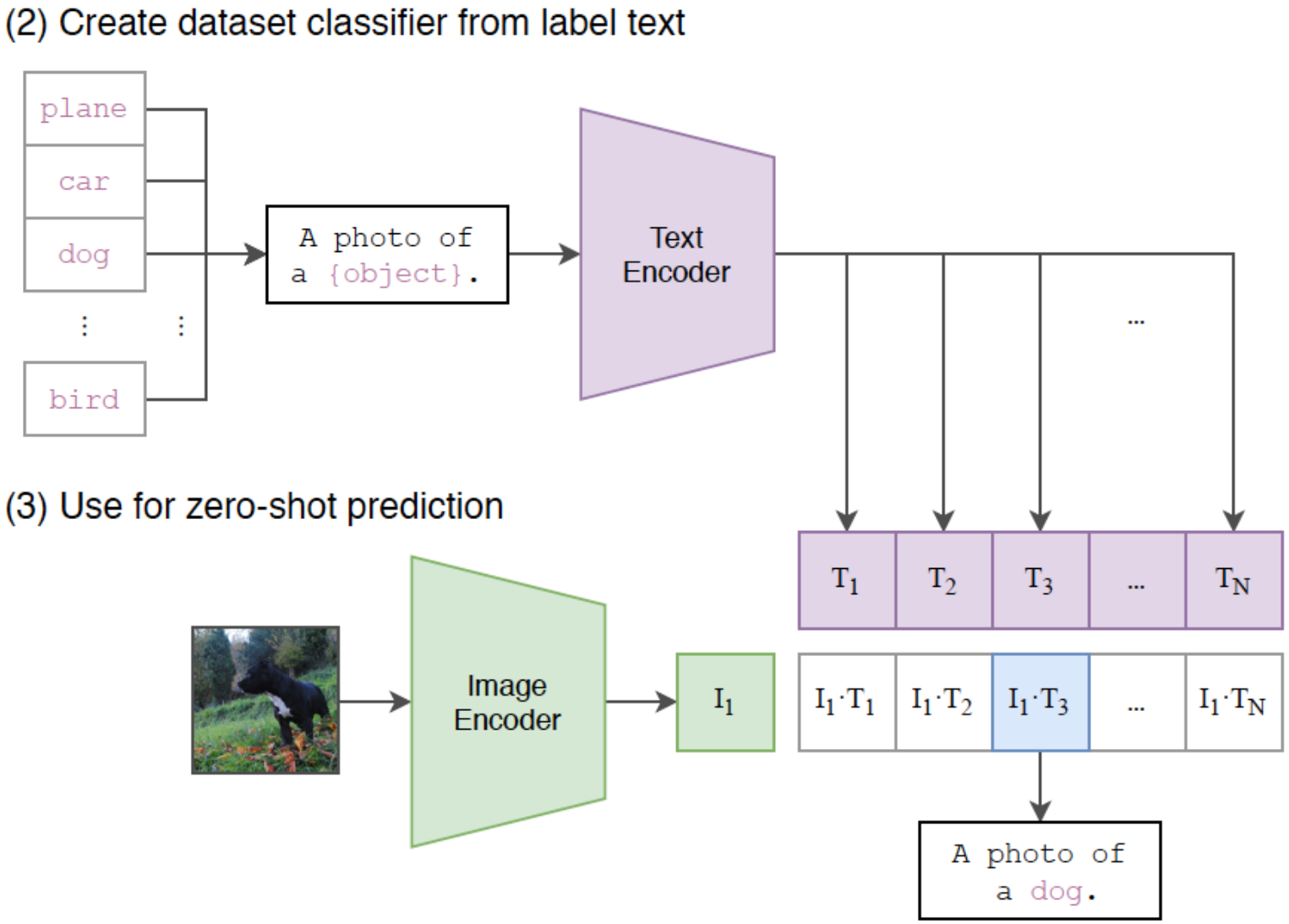
CLIPは画像エンコーダとテキストエンコーダを共同で学習し、（画像とテキストの）バッチ学習例の正しいペアリングを予測する。

テスト時に、学習されたテキストエンコーダは、ターゲットデータセットのクラスの名前や説明を埋め込むことで、ゼロショットの線形分類器を合成する。

(1) Contrastive pre-training



(2) Create dataset classifier from label text
(3) Use for zero-shot prediction



アプローチ
Natural Language Supervision

我々のアプローチの核心は、自然言語に含まれる監督情報から知覚を学習するという考え方である。

これは全く新しいアイデアではないが、この領域での研究を説明するために使用される用語は様々で、一見矛盾しているようにさえ見え、述べられた動機も多様である。

Zhangら(2020)、Gomezら(2017)、Joulinら(2016)、Desai & Johnson(2020)はいずれも、画像と対になったテキストから視覚表現を学習する手法を紹介しているが、それぞれのアプローチを教師なし、自己教師あり、弱教師あり、教師ありと表現している。

私たちは、この一連の研究に共通しているのは、使用されている特定の手法の詳細ではなく、自然言語を学習の信号として評価していることであることを強調する。

これらのアプローチはすべて、Natural Language Supervisionから学習している。初期の研究では、トピックモデルとn-gram表現を使用する場合、自然言語の複雑さと格闘していたが、深い文脈表現学習の改善は、我々は現在、この豊富な監督ソースを効果的に活用するためのツールを持っていることを示唆している（McCannら、2017）。

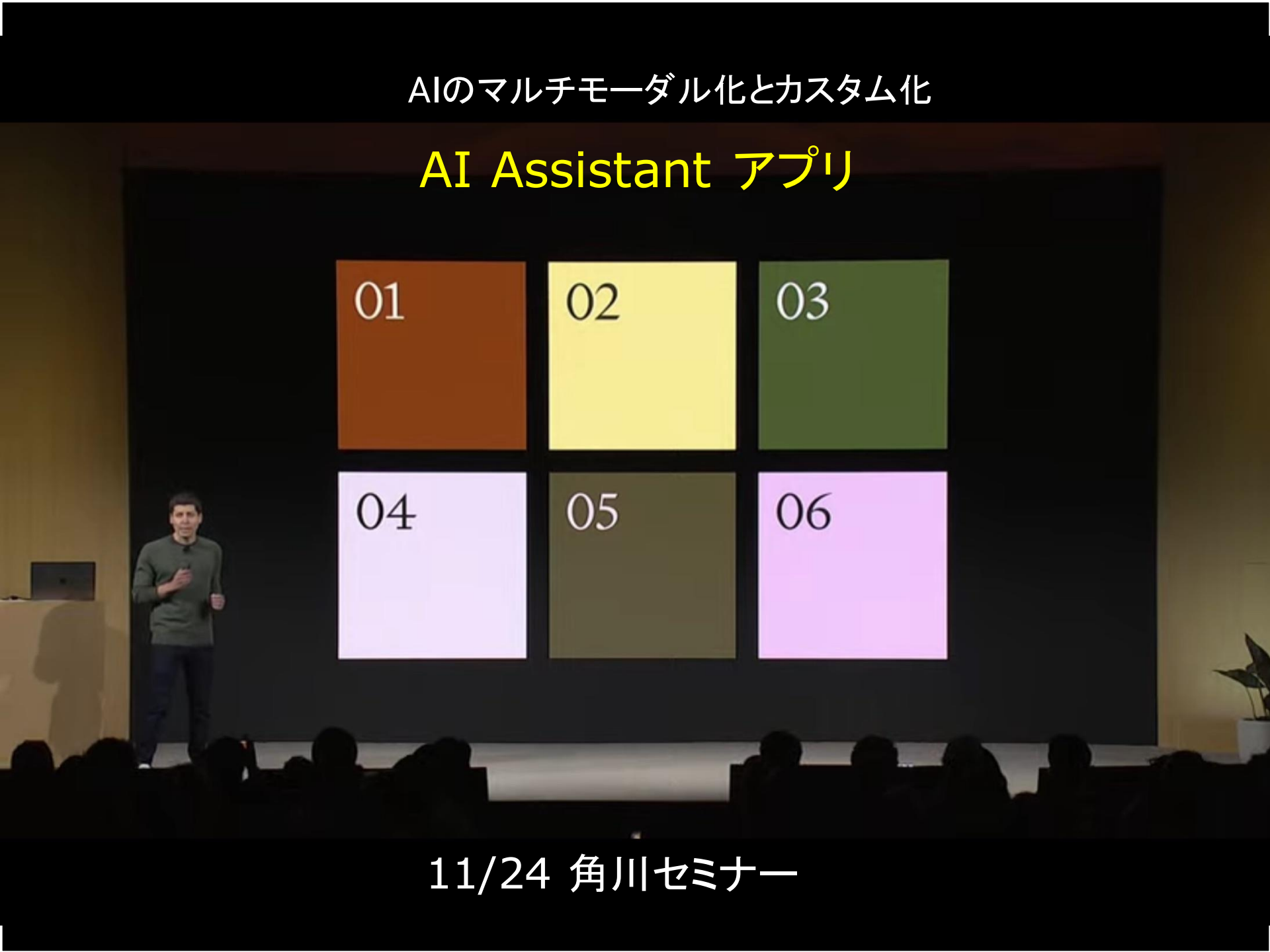
自然言語からの学習には、他の学習方法と比較していくつかの潜在的な強みがある。画像分類のための標準的なクラウドソースラベリングと比較して、自然言語監視のスケールははるかに簡単です。なぜなら、正統的な1対Nの多数決「ゴールドラベル」のような古典的な「機械学習互換フォーマット」である注釈を必要としないからです。

その代わりに、自然言語で動作するメソッドは、インターネット上の膨大な量のテキストに含まれる監視から受動的に学習することができる。

また、自然言語からの学習は、「単に」表現を学習するだけでなく、その表現を言語と結びつけることで、柔軟なゼロショット転送を可能にするという点で、ほとんどの教師なし学習や自己教師あり学習アプローチよりも重要な利点がある。

AIのマルチモーダル化とカスタム化

AI Assistant アプリ



11/24 角川セミナー

AI利用のインターフェースを
劇的に変えるAI Assistant アプリ

このセッションでは、OpenAi DevDayで発表された、GPTの能力をユーザーが開発したアプリの上で自由に生かすことを可能にするAssistant APIの概要を見ていきます。

また次回以降のセッションでは、OpenAIが同時に開発を進めていたAIのマルチモーダル化の成果を、今や、AI Assistant アプリの形で、ユーザーが利用できることを紹介したいと思います。

これまで、ChatGPTの利用のスタイルは、OpenAIのサイトにログインして直接ChatGPTと向き合って対話を続けること、具体的にはキーボードとスクリーンを通じてChatGPTとテキストを交換するのが基本でした。このスタイルが大きく変わろうとしています。

ユーザは、場合によればそのアプリの背後にAIがいることを全く意識せずに、普通のスマートフォンアプリと同じように画面タッチでボタンを押したり、スワイプしたりすればいいのです。僕が一番気に入っているインターフェースは、アプリに声で話しかけ、アプリが声で答えるというものです。

重要なことは、こうしたアプリを、OpenAIだけでなく開発者なら誰でも作成できるということです。OpenAIは、こうしたアプリの開発・流通を促進するためのマーケットを用意しています。

AI Assistant アプリ（これを、OpenAIはGPTsと呼んでいるようですが、ChatBotといういいかたもよく使われているようです）の登場は、一般のユーザーとAIとの距離をとても身近なものに劇的に変えるだけではありません。

それは、IT技術者・開発者とAIの距離を大きく変えるものです。

IT技術は・開発者は、これまで 、github copilot等を利用して、主要に開発支援ツールとしてAIを利用してきました。これからは、AIに支援された強力な独自のアプリを、自分の手で開発し、それを多数のユーザーが待つ市場に送り出すことができるのです。

このセミナーの前半では、人工知能技術の転換点を、翻訳モデル、大規模言語モデル、ChatGPTの三つに見てきたのですが、マルチモーダル化したAI Assistant アプリの登場が第四のマイルストーンになるのは確実だと、僕は考えています。

Assistant とは何か

Assistantとは、OpenAI APIの場合、GPT-4のような大規模な言語モデルからパワーを得て、ユーザーのためにタスクを実行することができるエンティティのことを指す。

これらのアシスタントは、モデルのコンテキストウィンドウ内に埋め込まれた命令に基づいて動作する。

また、アシスタントは通常、コードを実行したり、ファイルから情報を取得したりするような、より複雑なタスクを実行できるツールにもアクセスできる。

https://platform.openai.com/docs/introduction

Assistants API

Assistants APIを使用すると、独自のアプリケーション内にAI Assistantを構築することができる。

Assistantは指示を持ち、モデル、ツール、知識を活用してユーザーの問い合わせに応答することができる。

Assistants APIは現在、3種類のツールをサポートしている：

- Code Interpreter
- retrieval(知識検索)、
- Function Call（関数呼び出し）

である。

Assistant APIの働きの流れ
Assistant, Thread, Message, Run

1. Assistant を作成する。それは、モデルに対する指示を定義し、モデルを選択することで、必要であれば、Code Interpreter、Retrieval、Function callingなどのツールを有効にする。

2. ユーザーが会話を開始すると、Threadを生成する。

3. ユーザーが質問する際に、TreadにMessageを追加する。

4. Thread上でAssistantを実行(Run)して、応答をトリガーする。これにより、関連ツールが自動的に呼び出される。

Assistant API で利用される基本的なObject

Assistant	OpenAIのモデルと呼び出しツールを使用した専用AI
Thread	アシスタントとユーザー間の会話セッション。スレッドはメッセージを保存し、コンテンツをモデルのコンテキストに合わせるために自動的に切り捨てを処理する。
Message	アシスタントまたはユーザーが作成したメッセージ。メッセージにはテキスト、画像、その他のファイルを含めることができる。メッセージはスレッドにリストとして保存される。
Run	スレッド上でのアシスタントの呼び出し。アシスタントは設定とスレッドのメッセージを使用して、モデルやツールを呼び出してタスクを実行する。実行の一部として、アシスタントはスレッドにメッセージを追加する。
Run Step	アシスタントが実行の一部として行ったステップの詳細リスト。アシスタントは実行中にツールを呼び出したり、メッセージを作成したりできる。実行ステップを調べることで、アシスタントが最終結果にどのように到達しているかを知ることができる。

退職後の財政プランを立ててくれる パーソナル・アシスタント「僕の財政ボット」の例

Assistant

僕の財政ボット

Thread

退職後の財政プラン

Run

Assistant 僕の財政ボット
Thread 退職後の財政プラン

User's message
退職後の財政プラン用に、毎年いくら積み立てればいいだろうか？

Step

1. **Use code interpreter**
2. **Create messages**

Assistant's Message
年間6万円ほど積み見立ておいたほうがいい。さらに、…

Assistantはどのように動くのか？

Assistants APIは、開発者がさまざまなタスクを実行できる強力なAI Assistantアプリを構築できるように設計されている。

- AssistantはOpenAIのモデルを呼び出し、その性格や能力を調整するための具体的な指示を出すことができる。

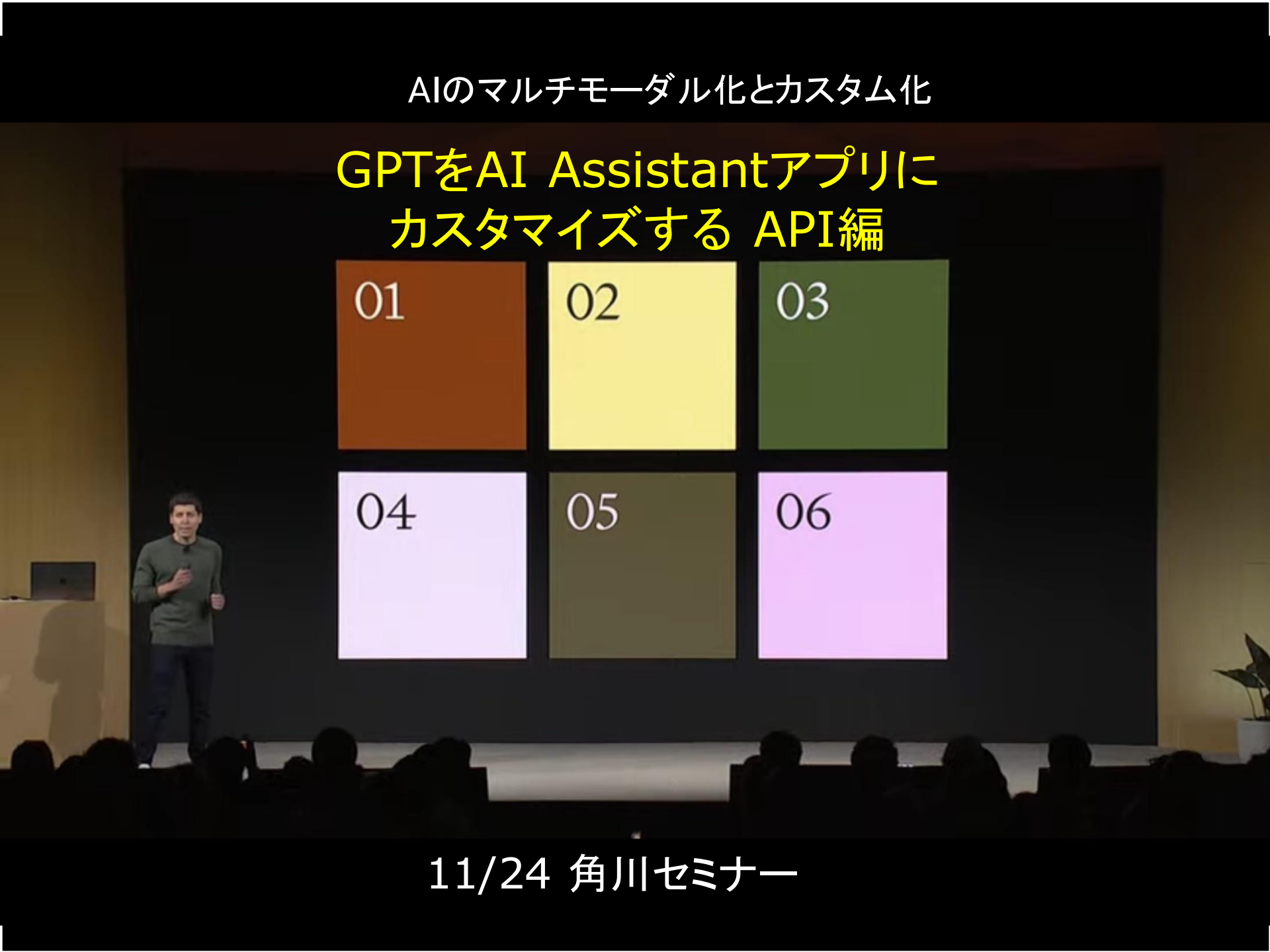
- Assistantは複数のツールに並行してアクセスできる。OpenAIがホストしているツール（コードインタープリタや知識検索など）、またはあなたが構築/ホストしているツール（関数呼び出しによる）の両方にアクセスできる。

- Assistantは永続的なスレッドにアクセスできる。スレッドは、メッセージの履歴を保存し、会話がモデルのコンテキストの長さに対して長くなりすぎたときに切り捨てることで、AIアプリケーションの開発を簡素化する。スレッドを一度作成し、ユーザーが返信するたびにメッセージをスレッドに追加するだけでいい。

- Assistantは、ファイルを作成する際、またはアシスタントとユーザー間のスレッドの一部として、いくつかの形式でファイルにアクセスすることができる。ツールを使用している場合、アシスタントはファイル（画像、スプレッドシートなど）を作成し、作成したメッセージで参照するファイルを引用することもできる。

AIのマルチモーダル化とカスタム化

GPTをAI Assistantアプリに カスタマイズする API編



11/24 角川セミナー

AI利用のインターフェースを
劇的に変えるAI Assistant アプリ API編

このセッションでは、前回紹介したAI利用のインターフェースを
劇的に変えるAI Assistant アプリ をどのように開発するのかを、
Assistant APIのレベルで、少し詳しく紹介しようと思います。

はじめにAssistant APIの基本を、改めて確認します。

その後で、OpenAIが公開している、Assistants playgroundでのコードを見ていきたいと思います。

最後に、Assistantの内部で利用できる三つのツールを確認します。

このセッションは、あまりビデオでの短い講義には向いていません。
是非、公開しているpdfファイルをゆっくりお読みください。

Assistants playground

Assistants playgroundサンプルコードは、次のような構成をしています。

- Step 1: Assistantを生成する
- Step 2: Threadを生成する
- Step 3: ThreadにMessageを追加する
- Step 4: Assistantを走らせる
- Step 5: Runのstatusをチェックする
- Step 6: AssistantのResponseを表示する
- サンプルの出力例

Toolsの説明は次のような構成です。

- Code Interpreter
  - Code Interpreterを有効にする
  - Code Interpreterにファイルを渡す
- Knowledge Retrieval
  - Retrievalを有効にする
  - Retrieval は、どう働くか？
  - Retrieval用のファイルをアップロードする
- 関数呼び出し
  - 関数を定義する
  - Assistantから呼ばれた関数を読み込む
  - 関数の出力をサブミットする

Assistant APIの基本
Assistant, Thread, Message, Run

1. Assistant を作成する。それは、モデルに対する指示を定義し、モデルを選択することで、必要であれば、Code Interpreter、Retrieval、Function callingなどのツールを有効にする。

2. ユーザーが会話を開始すると、Threadを生成する。

3. ユーザーが質問する際に、TreadにMessageを追加する。

4. Thread上でAssistantを実行(Run)して、応答をトリガーする。これにより、関連ツールが自動的に呼び出される。

Assistant API で利用される基本的なObject

Assistant	OpenAIのモデルと呼び出しツールを使用した専用AI
Thread	アシスタントとユーザー間の会話セッション。スレッドはメッセージを保存し、コンテンツをモデルのコンテキストに合わせるために自動的に切り捨てを処理する。
Message	アシスタントまたはユーザーが作成したメッセージ。メッセージにはテキスト、画像、その他のファイルを含めることができる。メッセージはスレッドにリストとして保存される。
Run	スレッド上でのアシスタントの呼び出し。アシスタントは設定とスレッドのメッセージを使用して、モデルやツールを呼び出してタスクを実行する。実行の一部として、アシスタントはスレッドにメッセージを追加する。
Run Step	アシスタントが実行の一部として行ったステップの詳細リスト。アシスタントは実行中にツールを呼び出したり、メッセージを作成したりできる。実行ステップを調べることで、アシスタントが最終結果にどのように到達しているかを知ることができる。

Assistants playgroundでのサンプル

Assistants APIに加えて、Assistantsプレイグラウンドも提供している。

このプレイグラウンドは、Assistants APIの機能を調べたり、コードを書かずに独自のアシスタントを構築する方法を学んだりするのに最適である。

Step 1: Assistantを生成する

アシスタントは、次のようないくつかのパラメータを使用して、ユーザーのメッセージに応答するように設定できるエンティティを表している：

- **Instructions**：アシスタントとモデルの動作や応答方法
- **Models**：微調整されたモデルを含め、GPT-3.5またはGPT-4モデルを指定できる。検索ツールはgpt-3.5-turbo-1106とgpt-4-1106-previewモデルを必要とする。
- **Tools**：APIは、OpenAIによって構築されホストされているCode InterpreterとRetrievalをサポートしている。
- **Functions**：APIでは、カスタム関数のシグネチャを定義することができる。

この例では、コードインタープリターツールを有効にして、個人的な数学の家庭教師であるアシスタント[(/docs/api-reference/assistants/createAssistant)]を作成している。

```
assistant = client.beta.assistants.create(
    name="Math Tutor",
    instructions="You are a personal math tutor. Write and run code to answer math questions.",
    tools=[{"type": "code_interpreter"}],
    model="gpt-4-1106-preview"
)
```

Step 2: Threadを生成する

スレッドは会話を表す。ユーザが会話を始めたらすぐに、ユーザごとに1つのThreadを作成することをお勧めする。このスレッドにユーザ固有のコンテキストやファイルを渡すには、Messagesを作成する。

```
thread = client.beta.threads.create()
```

スレッドにはサイズ制限がない。スレッドにはいくつでもメッセージを追加できる。

アシスタントはモデルへのリクエストが最大コンテキストウィンドウ内に収まるように、ChatGPTで広範囲にテストした切り捨てなどの関連する最適化テクニックを使用する。

アシスタントのAPIを使用する場合、任意のRunでモデルに渡される入力トークンの数の制御を委譲する。これは、場合によってはアシスタントの実行コストをあまり制御できないことを意味しますが、コンテキストウィンドウの管理の複雑さに自分で対処する必要はない。

Step 3: ThreadにMessageを追加する

メッセージにはテキストと、ユーザーがアップロードできるファイルが含まれる。メッセージは特定のスレッドに追加する必要がある。

GPT-4とVisionを使用したチャットcompletionのように、メッセージオブジェクトを経由して画像を追加することは現在サポートされていないが、今後数ヶ月のうちにサポートを追加する予定である。画像をアップロードして、retrievalで処理させることはできる。

```
message = client.beta.threads.messages.create(
    thread_id=thread.id,
    role="user",
    content="I need to solve the equation `3x + 11 = 14`. Can you help me?"
)
```

スレッド内のメッセージを一覧すると、このメッセージが追加されていることがわかる。


```
{
  "object": "list",
  "data": [
    {
      "created_at": 1696995451,
      "id": "msg_abc123",
      "object": "thread.message",
      "thread_id": "thread_abc123",
      "role": "user",
      "content": [{
        "type": "text",
        "text": {
          "value": "I need to solve the equation `3x + 11 = 14`. Can you help me?",
          "annotations": []
        }
        }],
      ...
```

Step 4: Assistantを走らせる

アシスタントがユーザメッセージに応答するには、Runを作成する必要がある。これにより、アシスタントはスレッドを読み取り、ツールを呼び出すか（ツールが有効な場合）、単にモデルを使用してクエリに最適に応答するかを決定する。

実行が進むと、アシスタントはスレッドにrole="assistant"のメッセージを追加する。

また、アシスタントはモデルのコンテキストウィンドウに含める過去のメッセージを自動的に決定する。これは、コストとモデルの性能の両方に影響する。

現在のアプローチはChatGPTの構築で学んだことに基づいて最適化されており、時間の経過とともに進化していくだろう。

オプションで、実行の作成中にアシスタントに追加の指示を渡すことができるが、これらの指示はアシスタントのデフォルトの指示を上書きすることに注意すること。

```
run = client.beta.threads.runs.create(
  thread_id=thread.id,
  assistant_id=assistant.id,
  instructions="Please address the user as Jane Doe. The user has a premium account."
)
```

Step 5: Runのstatusをチェックする

デフォルトでは、Runはキュー状態になります。定期的にRunを取得し、ステータスが完了したかどうかを確認することができる。

```
run = client.beta.threads.runs.retrieve(
  thread_id=thread.id,
  run_id=run.id
)
```

Step 6: AssistantのResponseを表示する

実行が完了すると、アシスタントがスレッドに追加したメッセージを一覧表示できる。

```
messages = client.beta.threads.messages.list(
  thread_id=thread.id
)
```

そして最後に、それらをユーザーに表示する！

この実行中に、アシスタントは2つの新しいメッセージをスレッドに追加した。

以下はその例である：

サンプルの出力例

ROLE CONTENT

user I need to solve the equation 3x + 11 = 14. Can you help me?

assistant Certainly, Jane Doe. To solve the equation (3x + 11 = 14) for (x), you'll want to isolate (x) on one side of the equation. Here's how you can do that:

Subtract 11 from both sides of the equation to get (3x = 3).

Then, divide both sides by 3 to solve for (x).

Let me calculate the value of (x) for you.

assistant The solution to the equation (3x + 11 = 14) is (x = 1).

Tools (Beta)

コードインタープリタや知識検索のようなOpenAIがホストしているツールへのアクセスをアシスタントに与えたり、関数呼び出しを使って独自のツールを構築することができる。

Code Interpreter

Code Interpreterは、Assistants APIを使用して、サンドボックス化された実行環境でPythonコードを記述し、実行することができる。

このツールは、多様なデータやフォーマットのファイルを処理し、データやグラフの画像を含むファイルを生成できる。

Code Interpreterを使用すると、アシスタントはコードを繰り返し実行して、難しいコードや数学の問題を解くことができる。

アシスタントが実行に失敗するコードを記述した場合、コードの実行が成功するまで別のコードの実行を試みることで、このコードを繰り返し実行することができる。

Code Interpreterを有効にする

Code Interpreterを有効にするには、アシスタント・オブジェクトのtoolsパラメータにcode_interpreterを渡す：

```
assistant = client.beta.assistants.create(
  instructions="You are a personal math tutor. When asked a math question, write and run code to answer the question.",
  model="gpt-4-1106-preview",
  tools=[{"type": "code_interpreter"}]
)
```

Code Interpreterにファイルを渡す

Code Interpreterは、ファイルからデータを解析することができる。これは、大量のデータをアシスタントに提供したい場合や、ユーザが独自のファイルをアップロードして分析できるようにしたい場合に便利である。

アシスタント・レベルで渡されたファイルには、このアシスタントを使用するすべてのランがアクセスできる：

```
# Upload a file with an "assistants" purpose
file = client.files.create(
  file=open("speech.py", "rb"),
  purpose='assistants'
)

# Create an assistant using the file ID
assistant = client.beta.assistants.create(
  instructions="You are a personal math tutor. When asked a math question, write and run code to answer the question.",
  model="gpt-4-1106-preview",
  tools=[{"type": "code_interpreter"}],
  file_ids=[file.id]
)
```

ファイルはスレッドレベルでも渡すことができる。これらのファイルには、特定のスレッドでのみアクセスできる。ファイルアップロードエンドポイントを使用してファイルをアップロードし、メッセージ作成リクエストの一部としてファイル ID を渡す:

```
thread = client.beta.threads.create(
  messages=[
    {
      "role": "user",
      "content": "I need to solve the equation `3x + 11 = 14`. Can you help me?",
      "file_ids": [file.id]
    }
  ]
)
```

Knowledge Retrieval

Retrievalは、独自の製品情報やユーザーから提供されたドキュメントなど、アシスタントのモデル外からの知識でアシスタントを補強する。

ファイルがアップロードされ、アシスタントに渡されると、OpenAIは自動的にドキュメントをチャンキングし、インデックスを作成し、エンベッディングを保存し、ユーザーのクエリに答えるために関連するコンテンツを取得するためにベクトル検索を実装する。

Retrievalを有効にする

アシスタントの tools パラメータにretrievalを渡して、検索を有効にする:

```
assistant = client.beta.assistants.create(
  instructions="You are a customer support chatbot. Use your knowledge base to best respond to customer queries.",
  model="gpt-4-1106-preview",
  tools=[{"type": "retrieval"}]
)
```

Retrieval は、どう働くか？

その後、モデルはユーザーメッセージに基づいてコンテンツを取得するタイミングを決定する。Assistants APIは自動的に2つの検索テクニックを選択する：

1. 短いドキュメントの場合はプロンプトにファイルの内容を渡す。
2. 長いドキュメントの場合はベクトル検索を行う。

検索は現在、モデル呼び出しのコンテキストにすべての関連コンテンツを追加することで、品質を最適化している。

我々は、開発者が検索品質とモデル使用コストの間の異なるトレードオフを選択できるように、他の検索戦略を導入することを計画している。

Retrieval用のファイルをアップロードする

コード・インタープリターと同様に、ファイルはアシスタント・レベル
またはスレッド・レベルで渡すことができる。

```
# Upload a file with an "assistants" purpose
file = client.files.create(
  file=open("knowledge.pdf", "rb"),
  purpose='assistants'
)

# Add the file to the assistant
assistant = client.beta.assistants.create(
  instructions="You are a customer support chatbot.
Use your knowledge base to best respond to
customer queries.",
  model="gpt-4-1106-preview",
  tools=[{"type": "retrieval"}],
  file_ids=[file.id]
)
```

スレッド内のメッセージにファイルを追加することもできる。これらのファイルはこのスレッド内でのみアクセス可能である。ファイルをアップロードした後、メッセージを作成するときにこのファイルの ID を渡すことができる。

```
:message = client.beta.threads.messages.create(
  thread_id=thread.id,
  role="user",
  content="I can not find in the PDF manual how to turn off this device.",
  file_ids=[file.id]
)
```

関数呼び出し

チャット completion APIと同様に、アシスタントAPIは関数呼び出しをサポートしている。

関数呼び出しでは、アシスタントに関数を記述し、呼び出しが必要な関数を引数とともにインテリジェントに返す。

Assistants APIは関数を呼び出すとRunの実行を一時停止し、関数呼び出しの結果を返すことでRunの実行を継続することができる。

関数を定義する

まず、アシスタントを作成するときに関数をを定義する:

```
assistant = client.beta.assistants.create(
  instructions="You are a weather bot. Use the
provided functions to answer questions.",
  model="gpt-4-1106-preview",
  tools=[{
      "type": "function",      )
    "function": {
      "name": "getCurrentWeather",
      "description": "Get the weather in location",
      "parameters": {
        "type": "object",
```

```
"properties": {
        "location": {"type": "string", "description":
"The city and state e.g. San Francisco, CA"},
        "unit": {"type": "string", "enum": ["c", "f"]}
      },
      "required": ["location"]
    }
  }
},
```

```
    {
    "type": "function",
    "function": {
      "name": "getNickname",
      "description": "Get the nickname of a city",
      "parameters": {
        "type": "object",
        "properties": {
          "location": {"type": "string", "description":
"The city and state e.g. San Francisco, CA"},
        },
        "required": ["location"]
      }
    }
  }]
)
```

Assistantから呼ばれた関数を読み込む

関数をトリガーするユーザーメッセージで検査を開始すると、検査は保留状態になる。

処理後、Runはrequires_actionステータスになり、Runを取得することで確認することができる。

モデルは、並列関数呼び出しを使用して、一度に呼び出す複数の関数を提供することができる:

```
{
  "id": "run_abc123",
  "object": "thread.run",
  "assistant_id": "asst_abc123",
  "thread_id": "thread_abc123",
  "status": "requires_action",
  "required_action": {
    "type": "submit_tool_outputs",
    "submit_tool_outputs": {
      "tool_calls": [
        {
          "id": "call_abc123",
          "type": "function",
          "function": {
```

```
        "name": "getCurrentWeather",
        "arguments": "{\"location\":\"San
Francisco\"}"
      }
    },
    {
      "id": "call_abc456",
      "type": "function",
      "function": {
        "name": "getNickname",
        "arguments": "{\"location\":\"Los
Angeles\"}"
      }
    }
  ]
}
```

関数の出力をサブミットする

次に、呼び出した関数からのツール出力をサブミットすることで、Runを完了することができる。

上記のrequired_actionオブジェクトで参照したtool_call_idを渡して、各関数呼び出しの出力を一致させる。

```
run = client.beta.threads.runs.submit_tool_outputs(
  thread_id=thread.id,
  run_id=run.id,
  tool_outputs=[
    {
      "tool_call_id": call_ids[0],
      "output": "22C",
    },
    {
      "tool_call_id": call_ids[1],
      "output": "LA",
    },
  ]
)
```


近未来のAIの展望


Be My AI !
AIのマルチモーダル化の中で
パーソナルなAIを展望する

「パーソナルなAI」を展望する

今回の講演で僕が示したいと思っているのは、一言でいえば、「パーソナルなAIへ」という展望です。

自分の目や耳や口をもつAIの登場といえば、AIロボットがしだいしだいに人間を押し除けてゆく、AI優位の近未来をイメージする人も、少なくないと思います。

そうではなく、様々な局面で我々人間を支援する、あくまでも人間のために役にたつAIを考えたいと思います。

「パーソナルなAI」を展望する

そういうAIを展望する一つの鍵は、すべての人が日常的にAIをパーソナルなアシスタントとして利用し、また、AIにとって人間のアシスタントであることが、競争的優位性を持つようにAIの未来を設計することだと、僕は考えています。

Be My AI !

もちろん、そのためにはAI技術は誰に対しても開かれたOpenなものでなければなりません。

AIのマルチモーダル化が可能とするボイスAIは AI利用拡大のゲームチェンジャー

僕は、音声で入出力ができる「ボイスAI」に大きな期待を持っています。

もしも、みんながドラえもんのようなAIロボットと一緒に暮らしていて、彼は、僕らの質問に、可能な限りいい答えを返してくれるとしましょう。

彼とのやりとりに、僕らは、キーボードを叩く必要があるでしょうか。それは面倒です。ボイスでやり取りをするのが「自然」です。

彼の話は、聞き取りやすいものになるでしょうか？　それは場合によります。

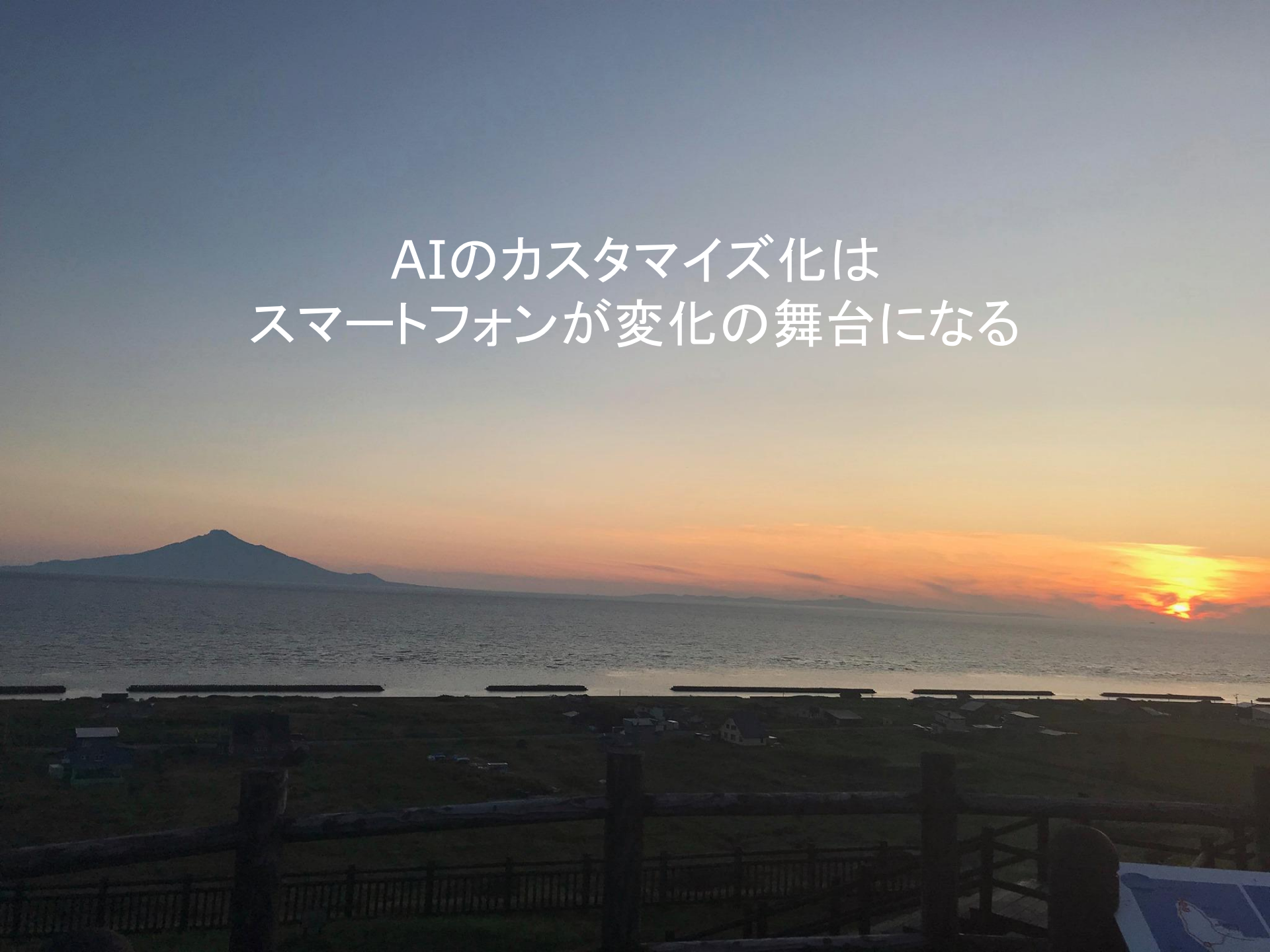
ある場合には、ボイス・インターフェースを他のテキストあるいはイメージのインターフェースに切り替える必要があるでしょう。

また、ある場合には、AIロボットとのボイスによるやり取りを繰り返して、必要な情報をボイスで取得することに成功するかもしれません。

人間・機械間のやりとりを
繰り返すことには意味がある

実は、大規模言語モデルにとって、こうした 人間・機械間のやりとりを繰り返すfew-shot prompt は、正しい答えに辿り着く、とても有効な方法なのです。

もっとも、現在のAIは、「次のプロンプト」をサジェストすることはできていません。それは、もっぱら、人間の役割です。ただ、この点は、少しマシにできるかもしれません。


AIのカスタマイズ化は
スマートフォンが変化の舞台になる

新しいインターフェースと
新しいデバイスへの期待

僕の「ボイスAI」に対する期待は、このような新しいインターフェースの開発への期待です。同時にそれは、そうしたインターフェースを搭載した新しいデバイスの登場への期待です。

その変化の主な舞台は、スマートフォンの世界になるはずです。

Smart Phoneが「賢い電話」という意味なら、それは「もっと賢い電話」にならなければなりません。それは、Smart PhoneにAIを搭載することで、はじめて可能になります。

Androidが、Androidという名前を持っていたことは、こうした変化にとって象徴的だったのかもしれません。

AI利用のインターフェースを
大きく変えるOpenAI Assistant API

これまで、ChatGPTの利用のスタイルは、OpenAIのサイトにログインして直接ChatGPTと向き合って対話を続けること、具体的にはキーボードとスクリーンを通じてChatGPTとテキストを交換するのが基本でした。このスタイルが大きく変わろうとしています。

ユーザは、場合によればそのアプリの背後にAIがいることを全く意識せずに、普通のスマートフォンアプリと同じように画面タッチでボタンを押したり、スワイプしたりすればいいのです。

先に触れたように、僕が一番気に入っているインターフェースは、アプリに声で話しかけ、アプリが声で答えるというものです。

スマートフォンに搭載されたAI Assistant アプリの登場は、一般のユーザーとAIとの距離をとても身近なものに大きく変えるでしょう。

それだけではありません。

それは、IT技術者・開発者とAIの距離を大きく変えるものです。

IT技術者・開発者は、これまで 、github copilot等を利用して、主要に開発支援ツールとしてAIを利用してきました。

これからは、IT技術者・開発者は、AIに支援された強力な独自のアプリを、自分の手で開発し、それを多数のユーザーが待つ市場に送り出すことができるのです。

退職後の財政プランを立ててくれる パーソナル・アシスタント「僕の財政ボット」の例

Assistant

僕の財政ボット

Thread

退職後の財政プラン

Run

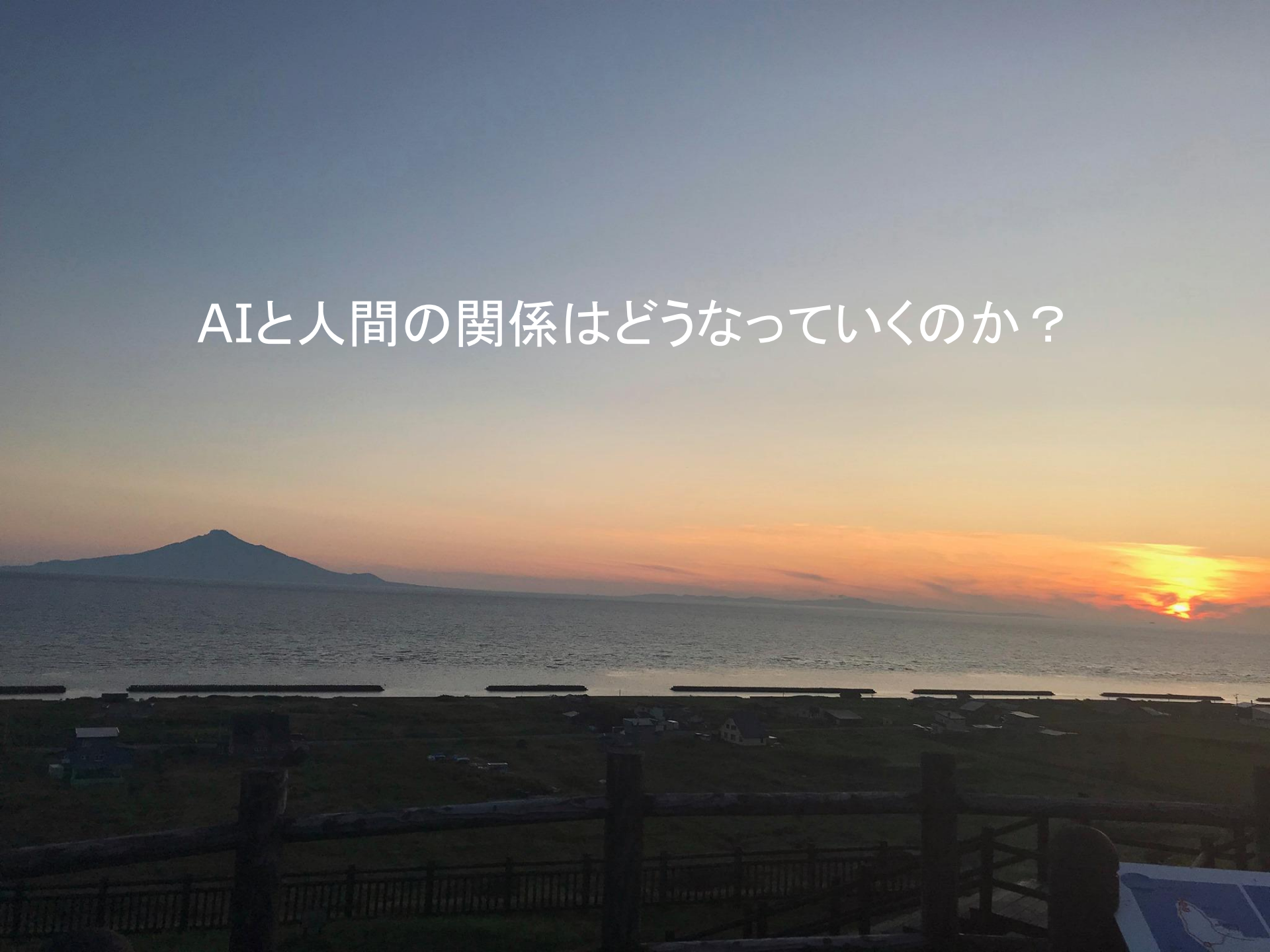
Assistant 僕の財政ボット
Thread 退職後の財政プラン

User's message
退職後の財政プラン用に、毎年いくら積み立てればいいだろうか？

Step

1. **Use code interpreter**
2. **Create messages**

Assistant's Message
年間6万円ほど積み見立ておいたほうがいい。さらに、…


AIと人間の関係はどうなっていくのか？

我々とAIの関係を明確にすることが AIの新しい発展を可能にする

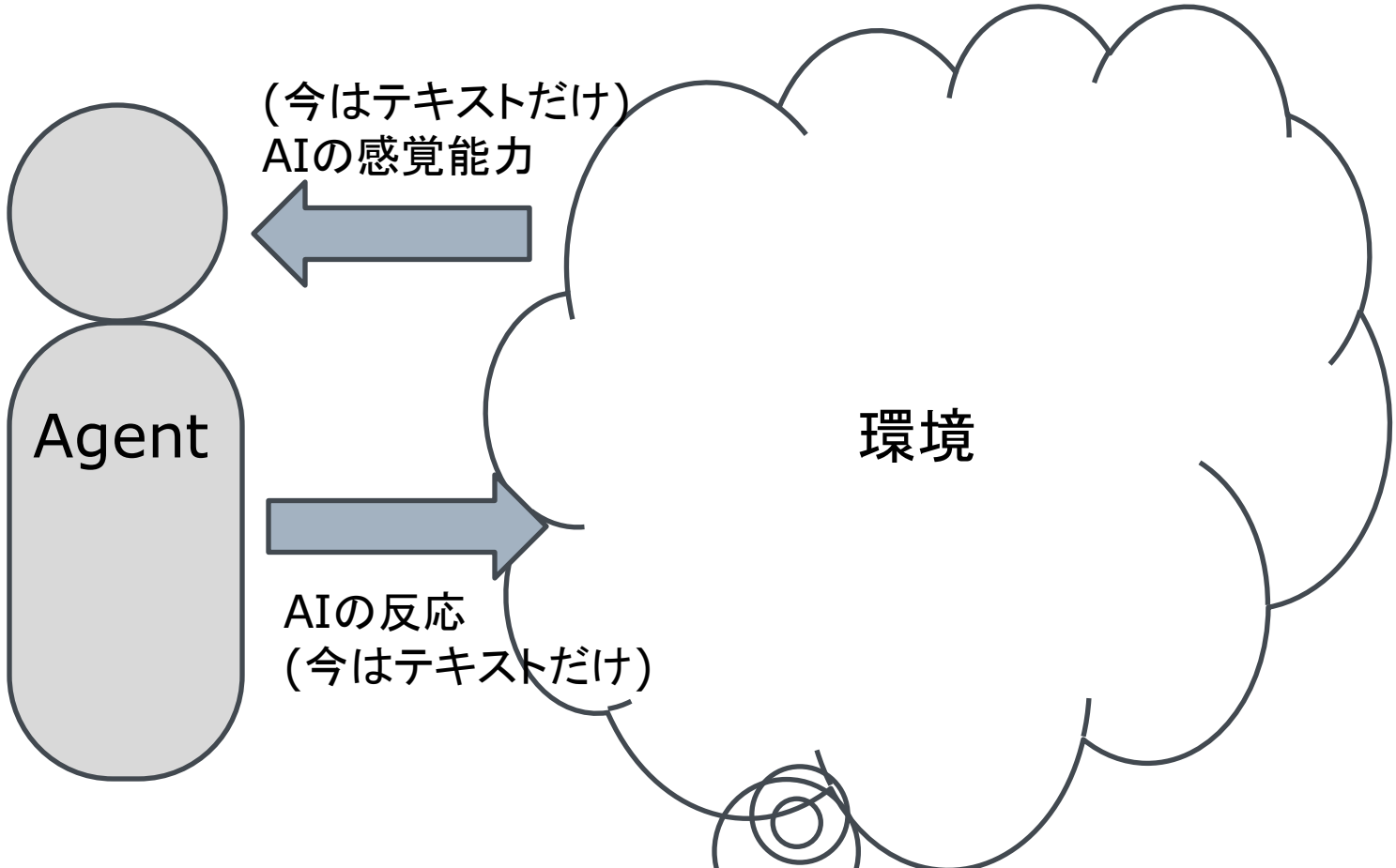
AIと人間の関係にとって一番基本的な問題は、我々人間がどのようなAIを望んでいるのかということにあります。

それが、AI利用の拡大にとっても、AIとのインターフェースを考える上でも鍵になります。

そうした問いかけが、AIの新しい発展を可能にするのです。

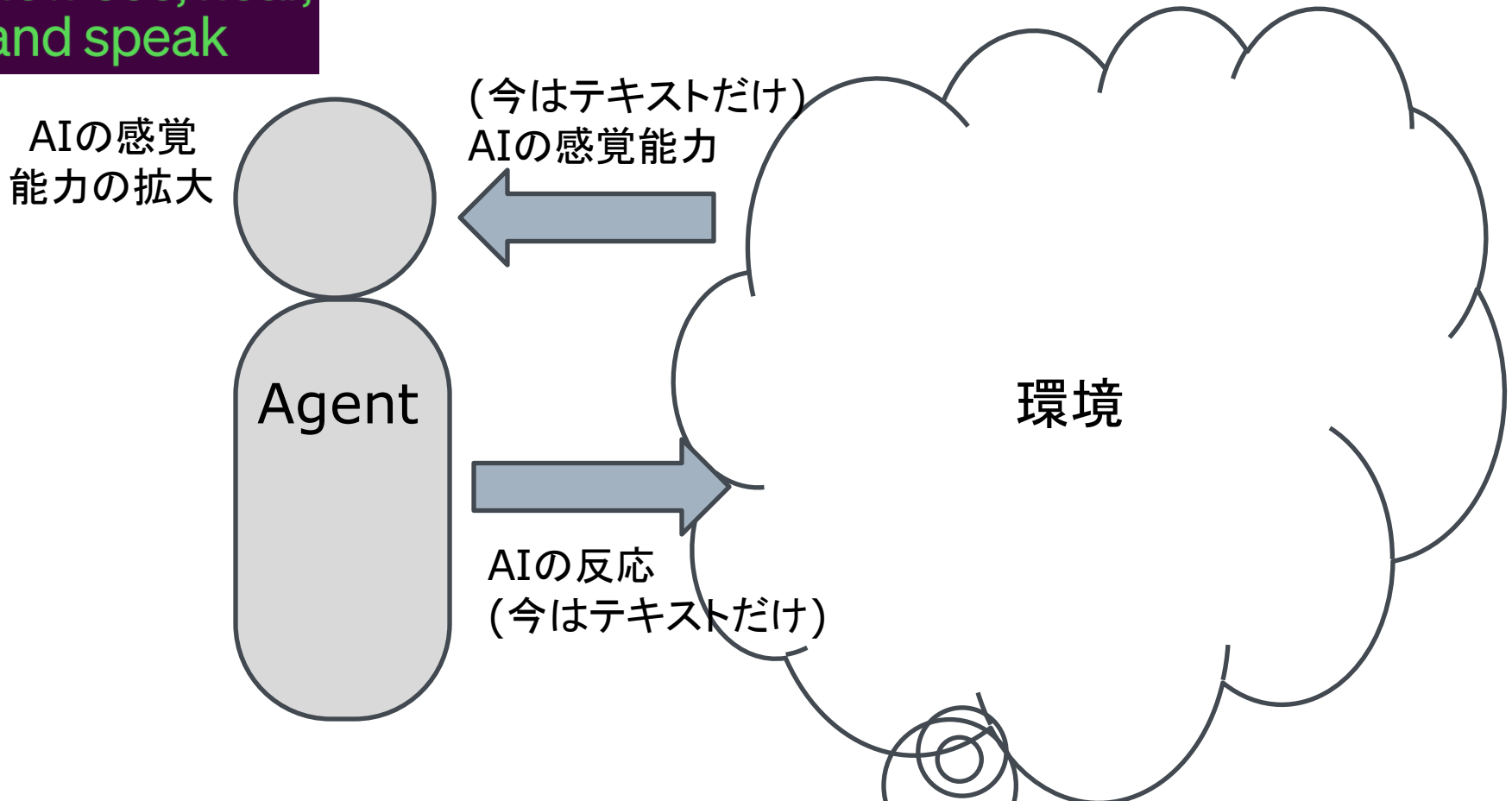
そういう問題を考える時期に、ようやく差し掛かっているのだと思います。

マルチモーダルなAIのモデルを考える
Agent Base Model

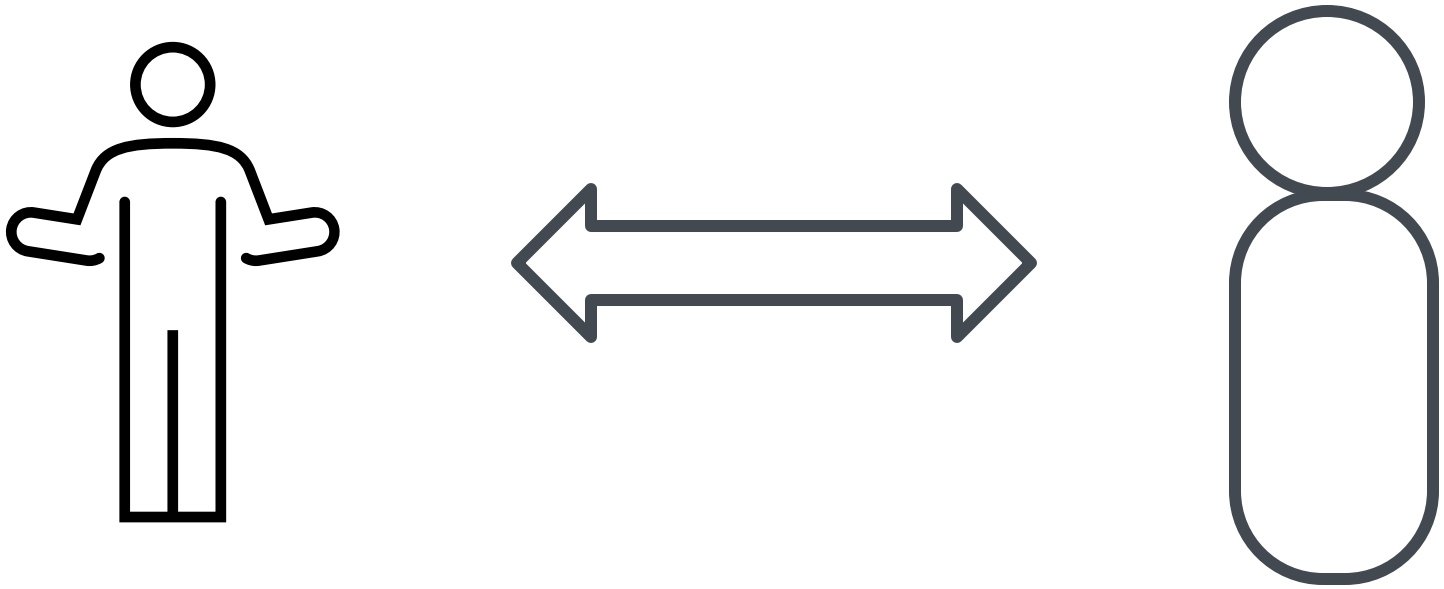


マルチモーダルなAIのモデル
Agent Base Model

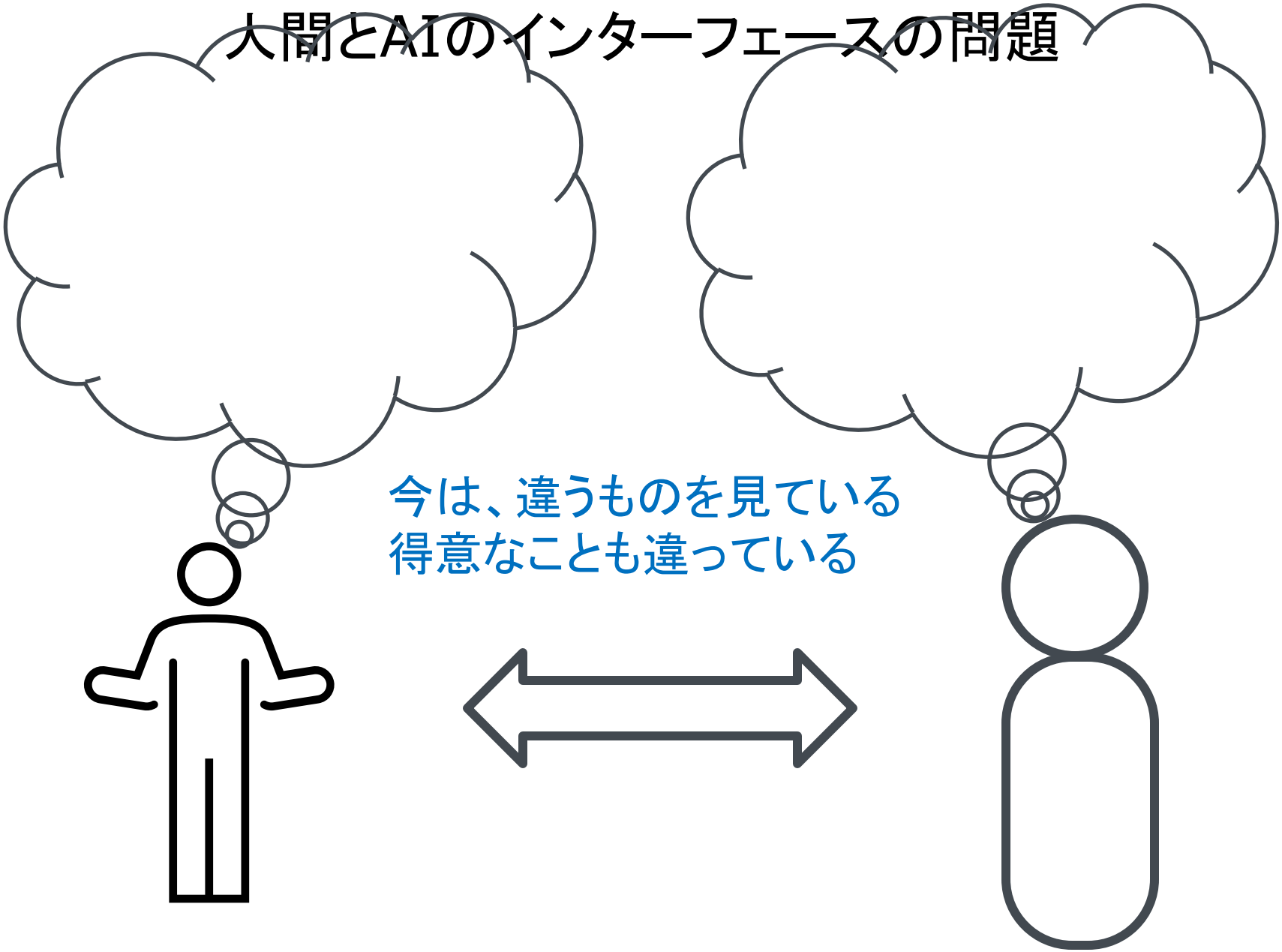
ChatGPT can now see, hear, and speak



人間とAIのインターフェース

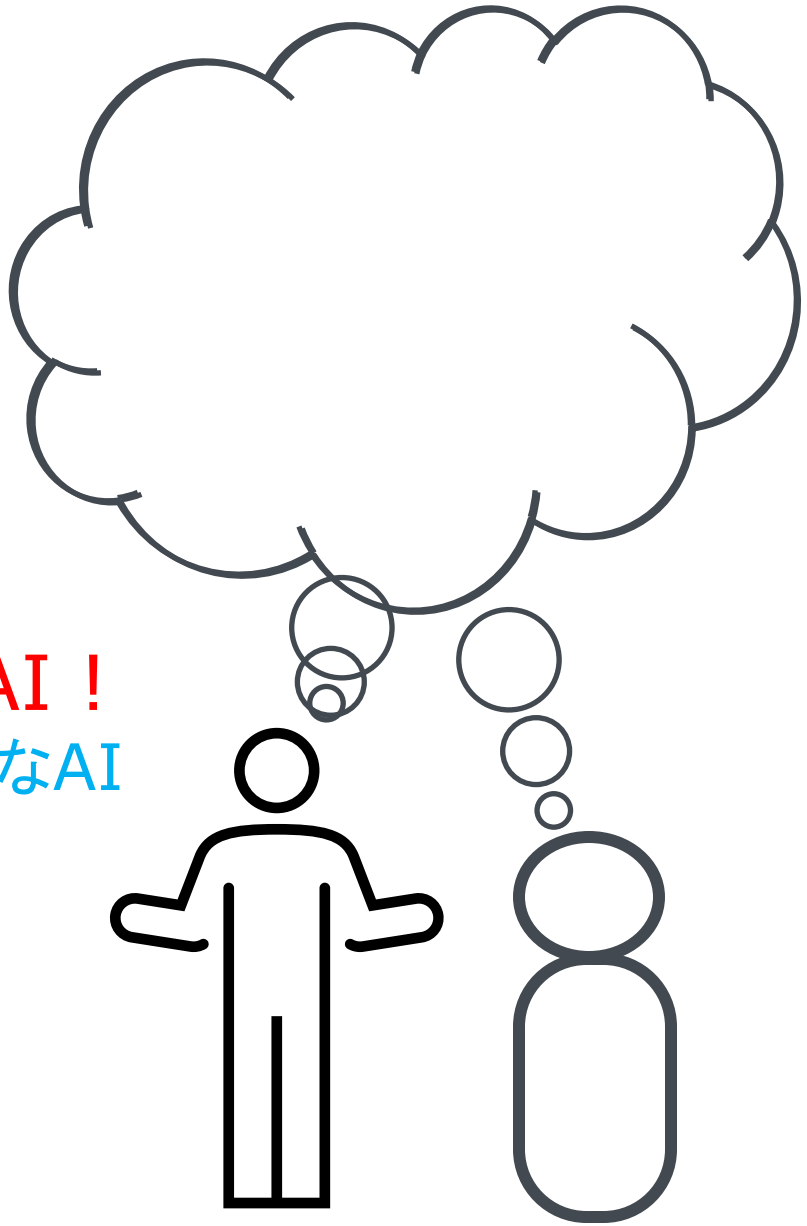


人間とAIのインターフェースの問題

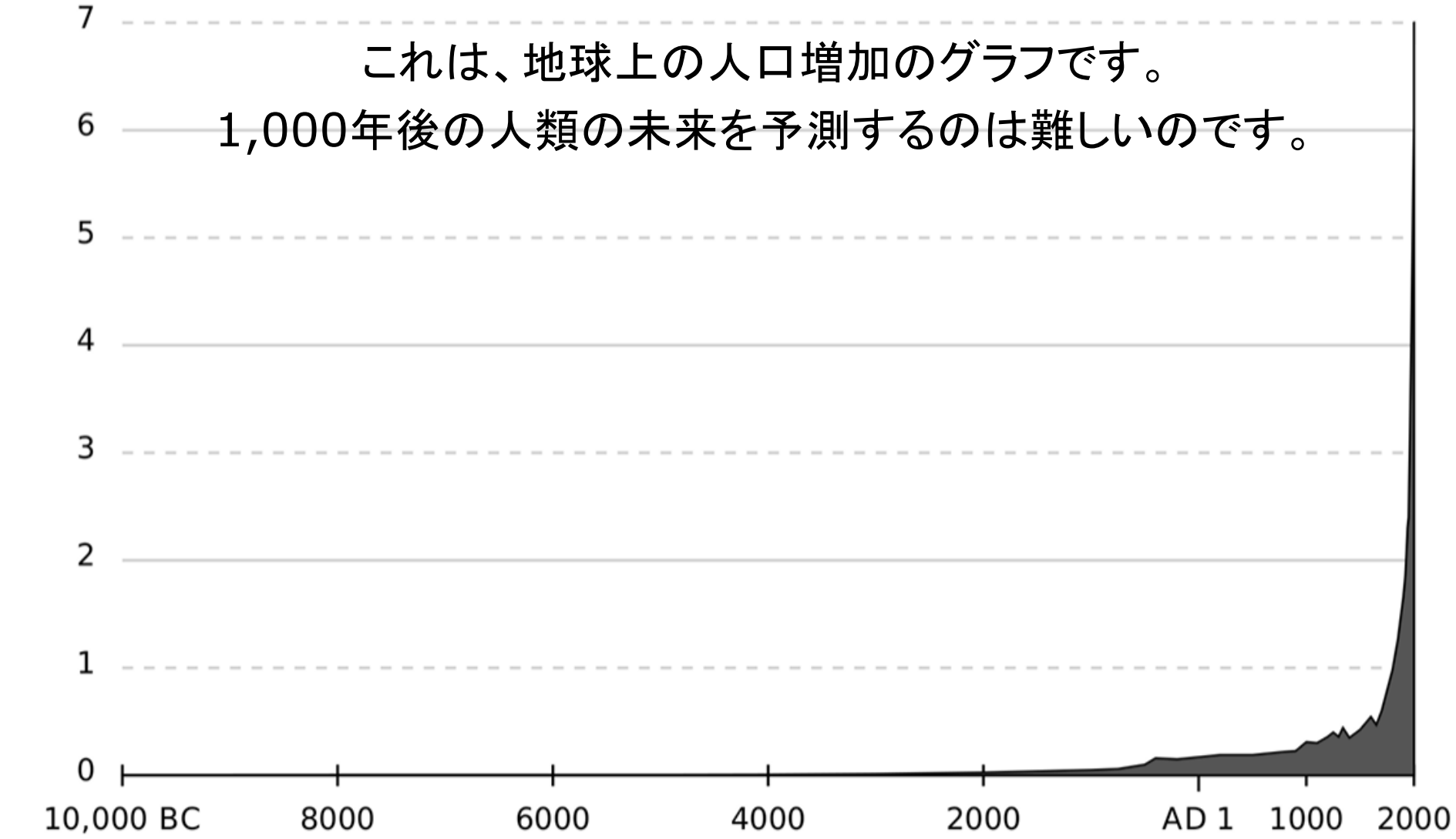


Be My AI !

パーソナルなAI



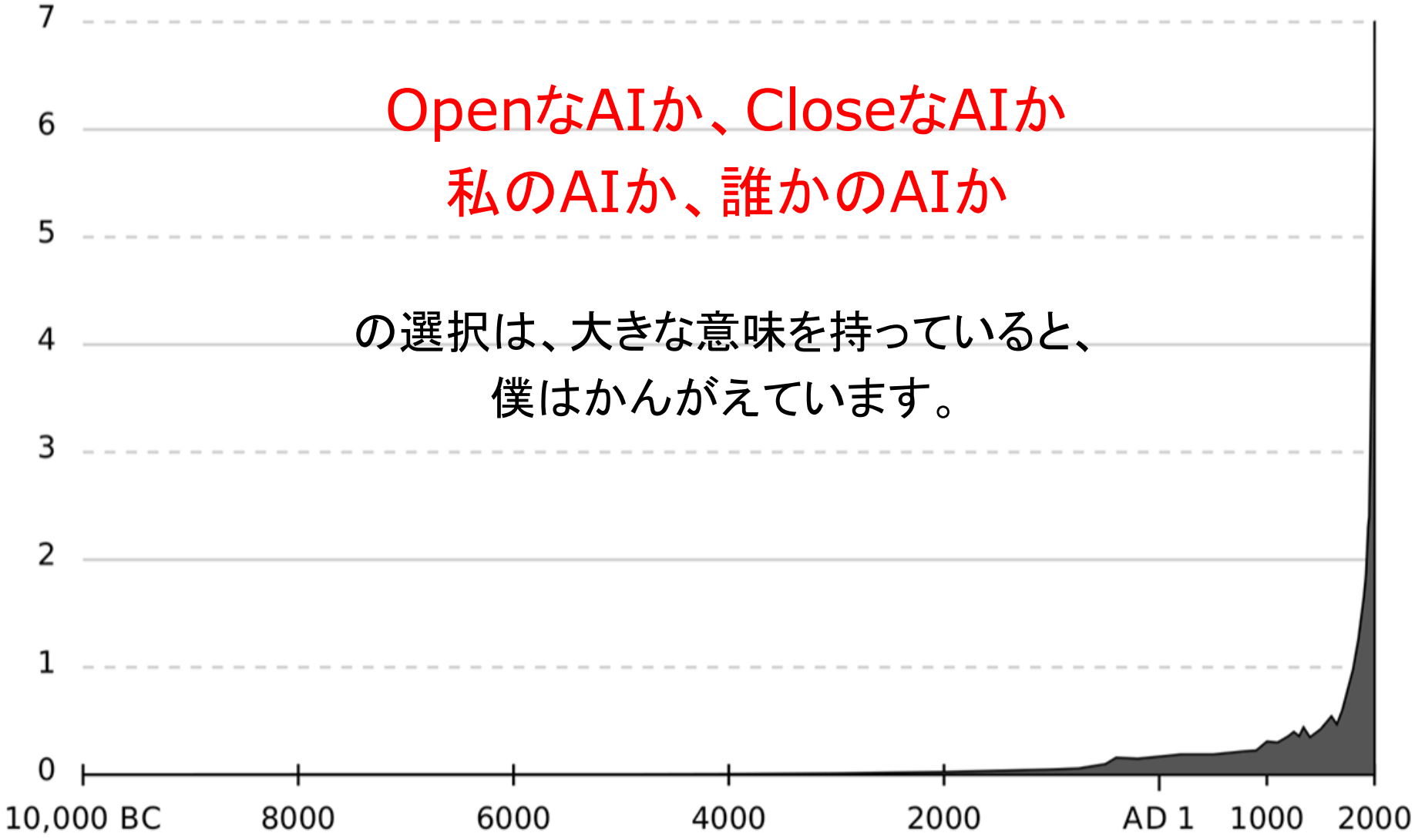
今回は、非常に楽観的な未来予想をしました。



ただ、100年後の未来についていえば、

OpenなAIか、CloseなAIか
私のAIか、誰かのAIか

の選択は、大きな意味を持っていると、
僕はかんがえています。




Be My AI !