

自然言語処理の最新動向'22

自然言語処理って何を連想しますか？

どうやって出来ているかの話です

- 小難しい話が沢山出できます
- が, ぼんやりとでも良いのでこれだけ覚えていってください
- **分布仮説**
- 単語の**分散表現**
- 文脈を考慮したテキスト(文)の**分散表現**

自然言語処理とは

- ・自然言語処理 (Natural Language Processing)
 - ・人間の言語 (自然言語) を機械で処理すること
- ・研究分野としての自然言語処理
 - ・古来, 記号列(単語や文)が持つ意味をどう扱うかを中心に研究してきた

分布仮説 [Harris+, 1954]

- 単語の意味はその周囲の単語から形成されるという仮説
→ 似た文脈で出現する単語は意味が似ている

文1: 昨日,りんごを食べた。りんごジュースを飲んだ。りんごの皮をむいた。

文2: 昨日,りんごを食べた。ぶどうジュースを買った。ぶどうの皮をむいた。

文3: 昨日,自転車に乗った。自転車を修理した。自転車を買った。

分布仮説 [Harris+, 1954]

- 单語の意味はその周囲の単語から形成されるという仮説
→ 似た文脈で出現する単語は意味が似ている

文1: 昨日,りんごを食べた。りんごジュースを飲んだ。りんごの皮をむいた。

文2: 昨日,りんごを食べた。ぶどうジュースを買った。ぶどうの皮をむいた。

文3: 昨日,自転車に乗った。自転車を修理した。自転車を買った。

共起による
ベクトル表現
[Lin, 2002]

	…	食べる	…	飲む	…	修理	…
りんご	0	1	0	1	0	0	0
ぶどう	0	1	0	1	0	0	0
自転車	0	0	0	0	0	1	0
:							

← 語彙数(数万～数十万)分の疎なベクトルになる →

似てる
似てない

分布仮説と単語の分散表現

- ・単語の意味はその周囲の単語から形成されるという仮説 (=分布仮説)
→ 似た文脈で出現する単語は意味が似ている
- ・各意味を複数の次元で分散して表現する (=分散表現)
→ 次元は低次元(例えば100次元)で, 値は実数値

単語埋め込み
(word embedding)
とも呼ばれる

これらの実数値をニューラルネットワークで求める

共起による
ベクトル表現
[Lin, 2002]

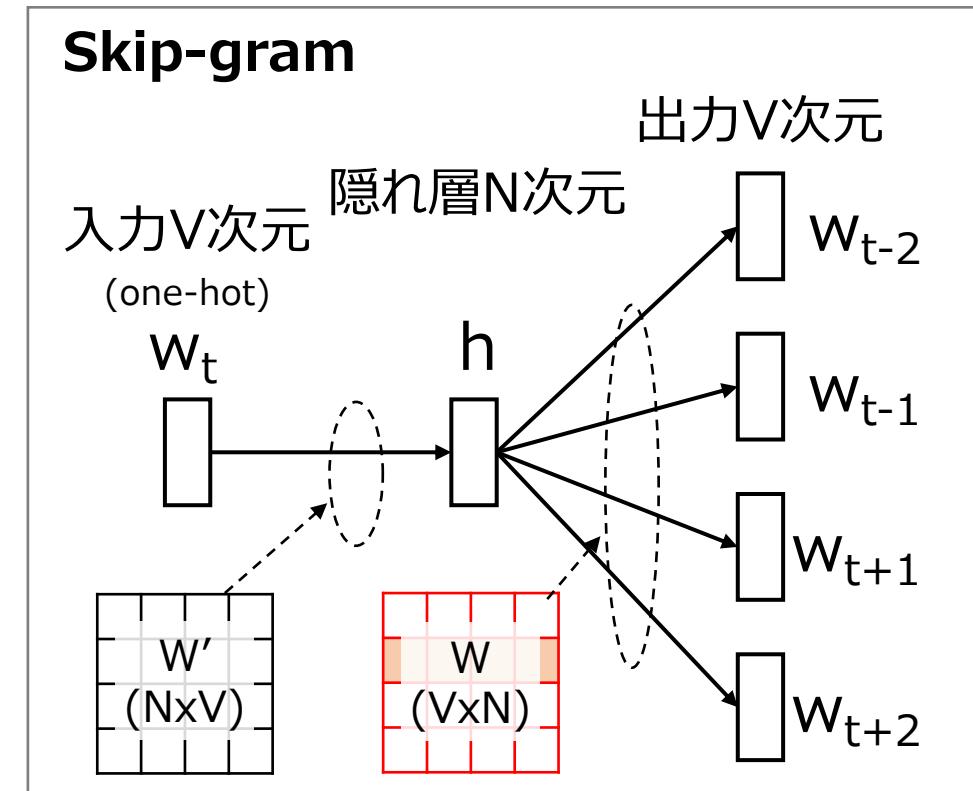
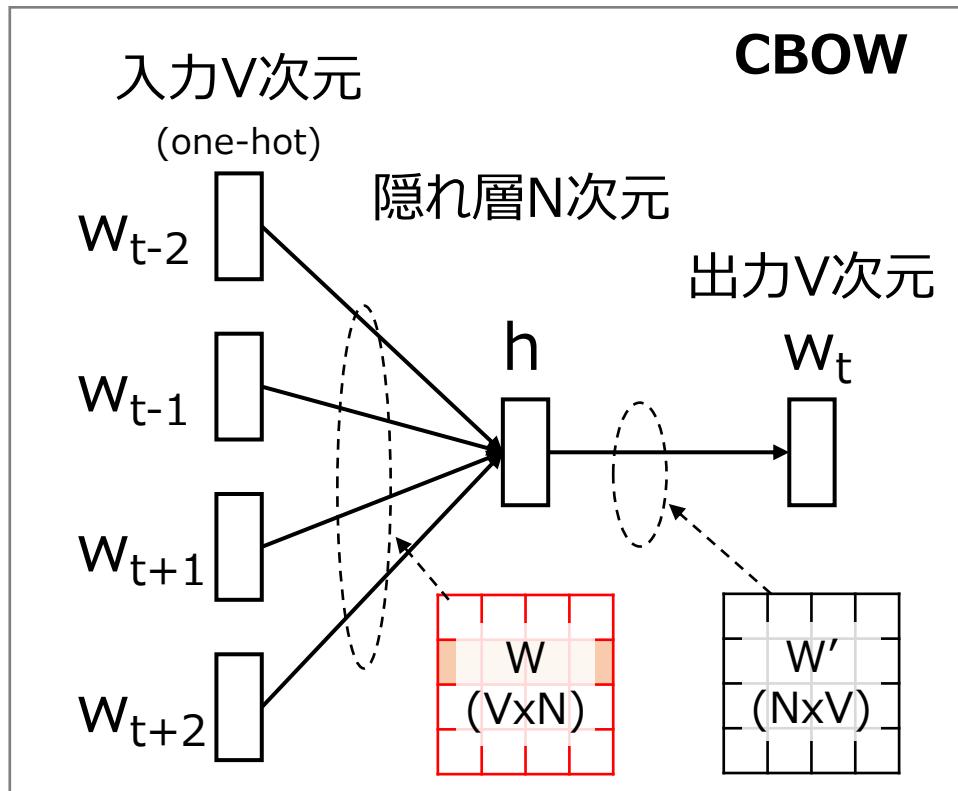
次元→	0	1	…	50	…	98	99
りんご	1.07	-1.08		1.48		0.46	0.48
ぶどう	1.95	-1.53		0.36		-0.61	-0.44
自転車	0.67	1.44		-1.50		0.10	0.67
:							

← 高々数百次元の密なベクトル →

似てる
似てない

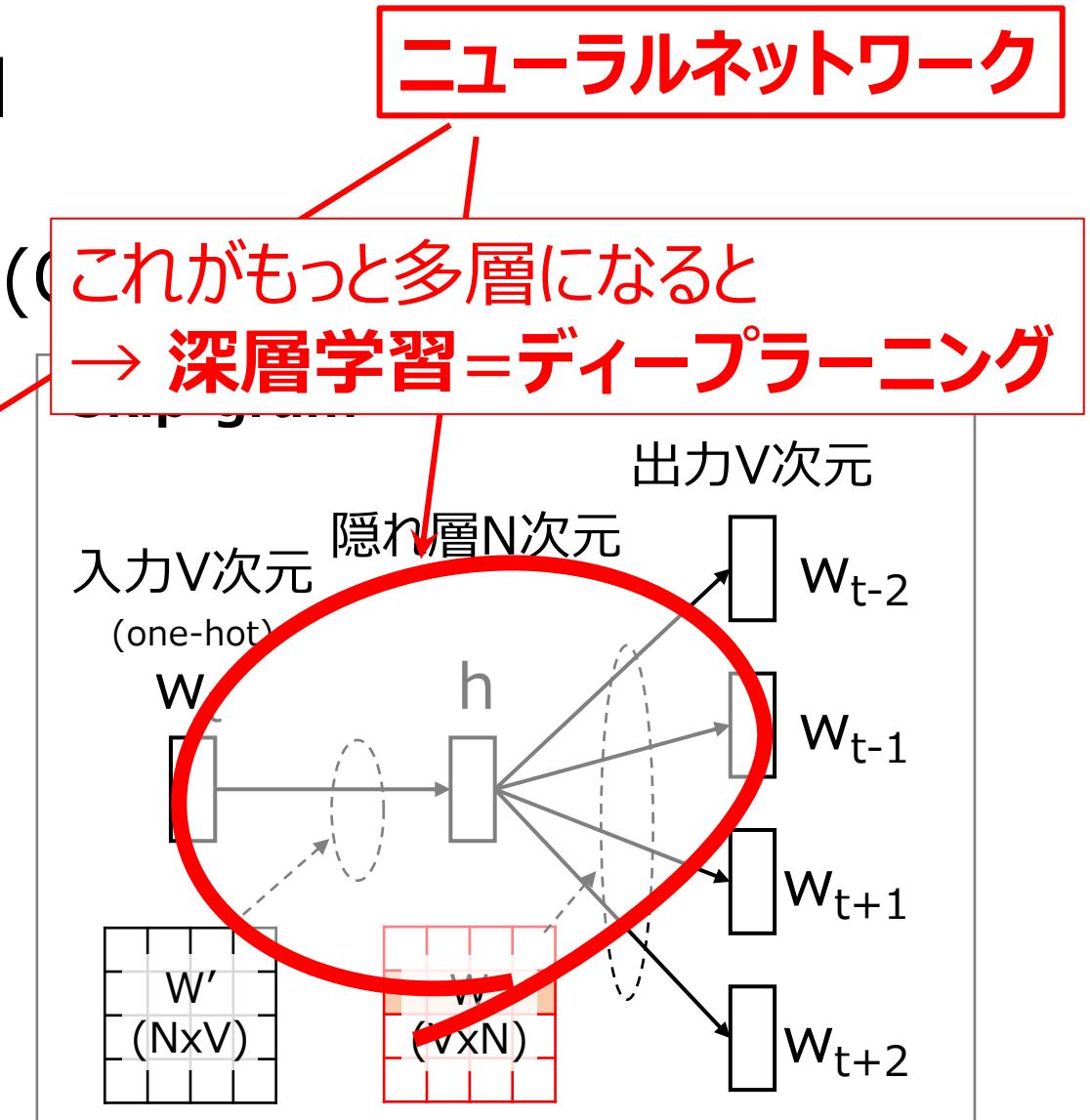
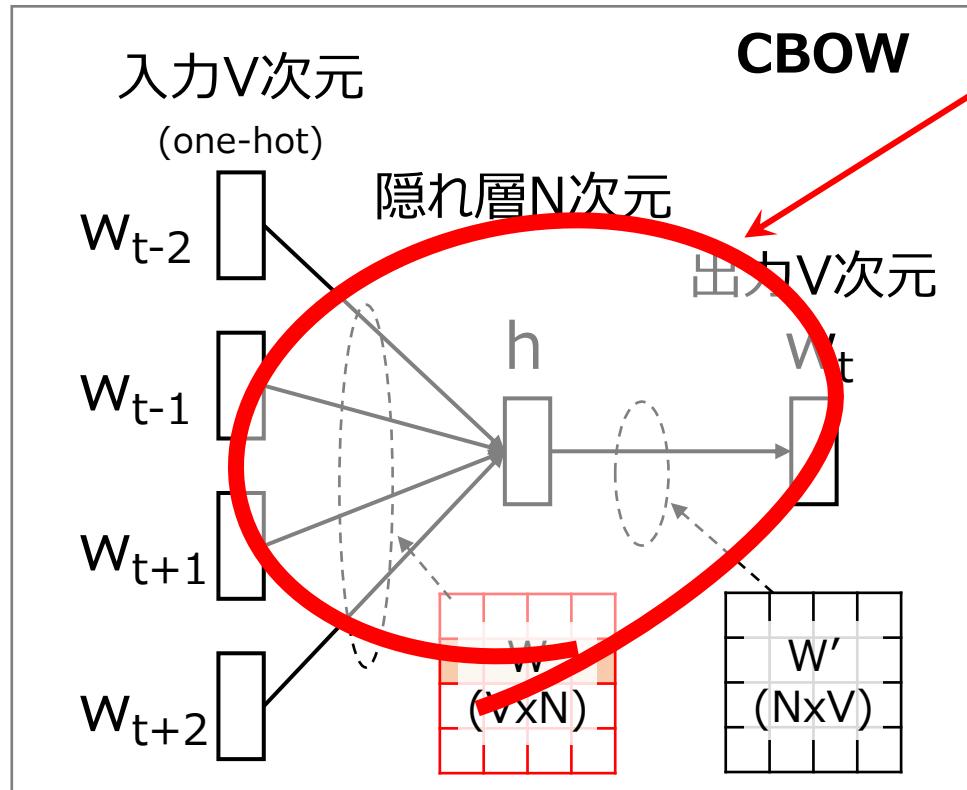
word2vec [Mikolov+, 2013]

- 周辺の単語から中心の単語を予測(CBOW) or その逆(Skip-gram)



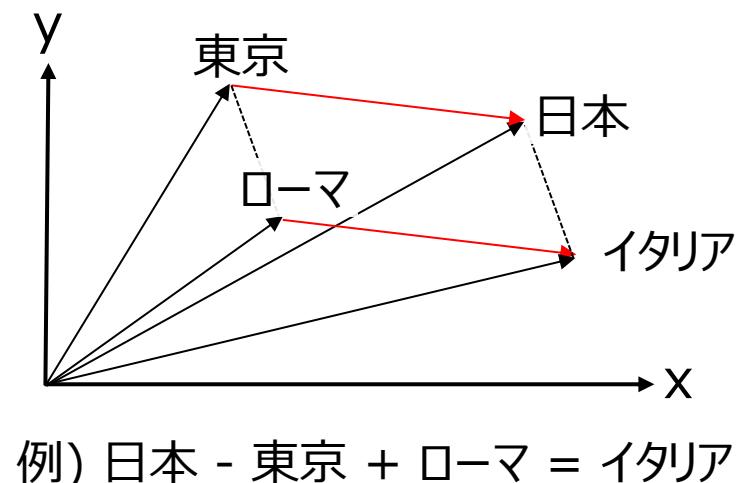
word2vec [Mikolov+, 2013]

- 周辺の単語から中心の単語を予測(CBOW)

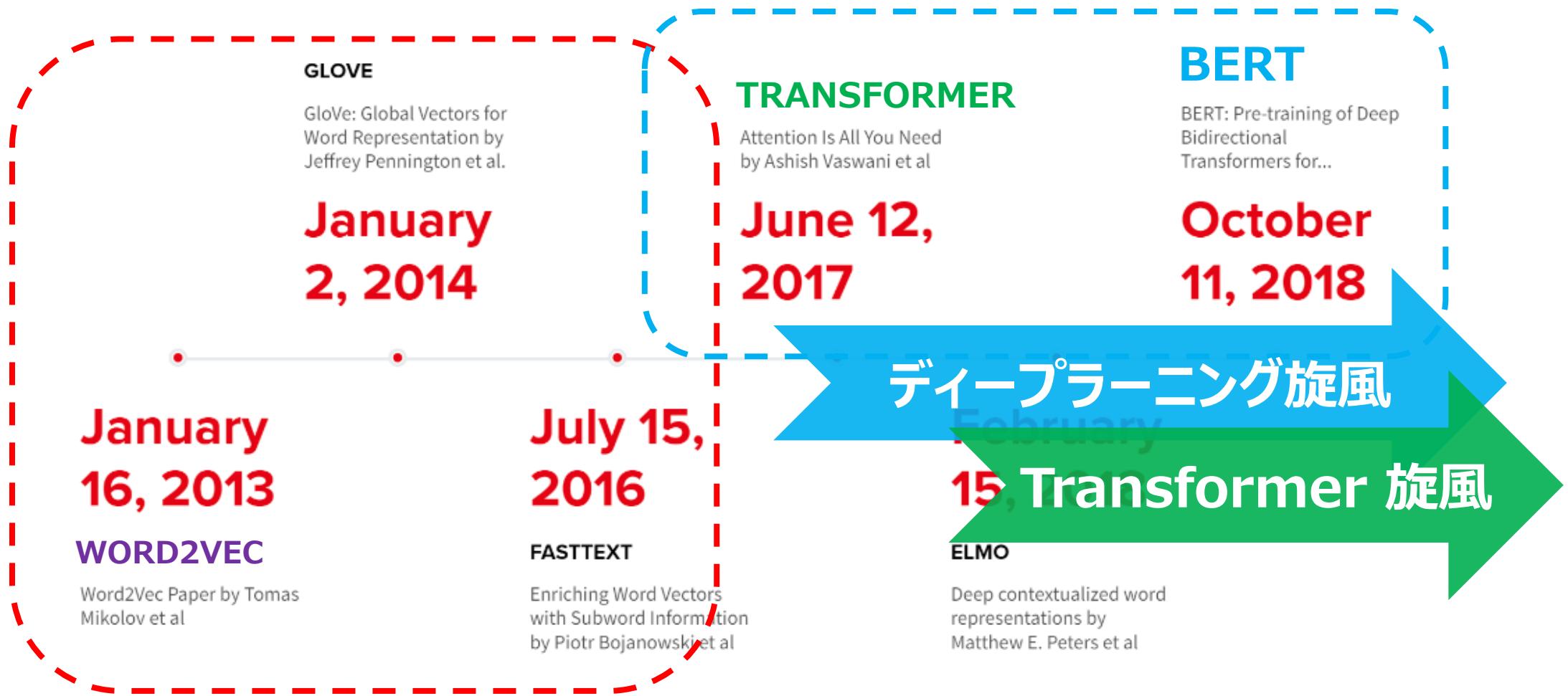


word2vec が何が凄いのか？

- 単語のベクトル表現は前からあった
 - 以前の方法: **TF-IDF**, Okapi BM25 など (分布的, 高次元, スパース)
 - 分散表現: **word2vec**, GloVe, fastText など (分散的, 低次元, 密)
- 類似度の比較はできていたが, **足したり引いたり**できなかった
 - $\text{king} - \text{man} + \text{woman} = \text{queen}$ で有名
- **学習済みモデル**が公開されるようになった
 - Wikipedia など**大規模コーパス**(テキスト)で学習したモデルが公開 → エコシステム
 - エコシステム化の加速で利用障壁が低下
→ メジャーなライブラリにも組み込まれるようになった



学習済みモデルのタイムライン



ここまでまとめ

- **分布仮説**
 - 似た文脈で出現する単語は意味が似ているという仮説
- **分散表現**
 - 単語の意味を複数次元の実数値で分散して表現したもの
 - ニューラルネットワークで学習し, 次元は**低次元** (例えば100次元)
 - word2vec が有名
- **学習済みモデル**
 - 学習済みモデル(大規模データで学習)の公開が加速し, 利用障壁が低下

(参考) ニューラルネットワーク研究の系譜

- 黎明～終焉を繰り返し,近年は 3度目のブーム

第 1 期	1940～	• McCullochとPitts が形式ニューロンモデルを発表 [McCulloch-Pitts,43]
	1950～	• Rosenblatt がパーセプトロンを発表 [Rosenblatt,57]
	1960～	• MinskyとPapert が単純パーセプトロンの(線形分離不可能問題への)限界を指摘 [Minsky-Papert,69]
冬	1970～	冬の時代 (階層的構造の学習方法が未解決)
第 2 期	1980～	• Fukushima らがネオコグニトロンを提案 [Fukushima,80]
		• Rumelhart らが誤差逆伝播法を提案 [Rumelhart+,86]
		• LeCun らが畳み込みニューラルネット Conv.net を提案 [LeCun,89]
冬	1990～	冬の時代 (学習時間や過学習に課題, 一方でSVMが流行)
第 3 期	2000～	• Hinton らが事前学習とオートエンコーダを導入した多層NNを提案 [Hinton+,06]
	2010～	• Seide らが音声認識のベンチマークで圧勝 [Seide+,11] • Krizhevsky らがReLU を提案し画像認識コンペで圧勝 [Krizhevsky,12]

(参考) 音声認識での成功 [Seide+, 2011]

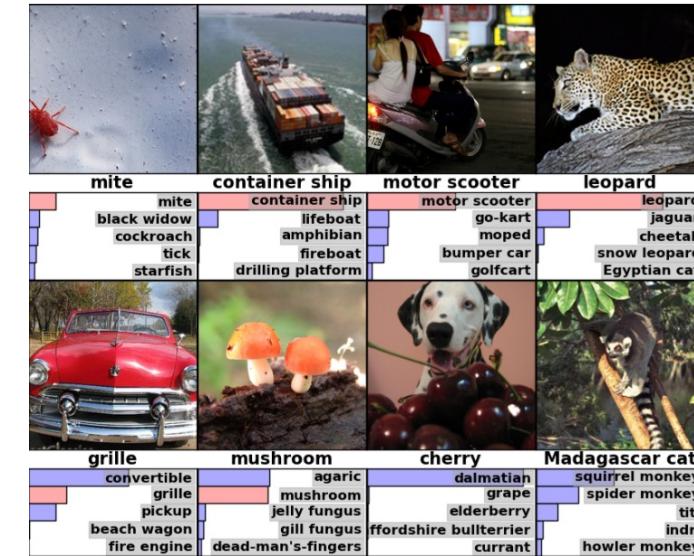
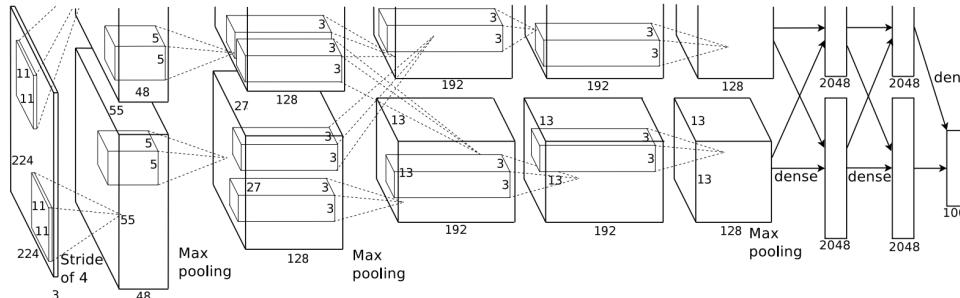
- Microsoft Research のグループ
 - 電話での会話音声の標準データセット
 - 入力(MFCC)-出力(HMM状態変数)の関係をDNNで学習
 - 従来 GMM-HMM → DNN-HMM (全結合7層, 事前学習あり)
 - 単語誤認識率で 10%前後の大幅な精度改善

acoustic model & training	recognition mode	RT03S		Hub5'00 SWB	voicemails		tele- conf
		FSH	SW		MS	LDC	
GMM 40-mix, ML, SWB 309h	single-pass SI	30.2	40.9	26.5	45.0	33.5	35.2
GMM 40-mix, BMMI, SWB 309h	single-pass SI	27.4	37.6	23.6	42.4	30.8	33.9
CD-DNN 7 layers x 2048, SWB 309h, this paper (rel. change GMM BMMI → CD-DNN)	single-pass SI	18.5 (-33%)	27.5 (-27%)	16.1 (-32%)	32.9 (-22%)	22.9 (-26%)	24.4 (-28%)

F. Seide, G. Li and D. Yu, "Conversational Speech Transcription Using Context-Dependent Deep Neural Networks." *Interspeech*. 2011.

(参考) 画像認識での成功 [Krizhevsky+, 2012]

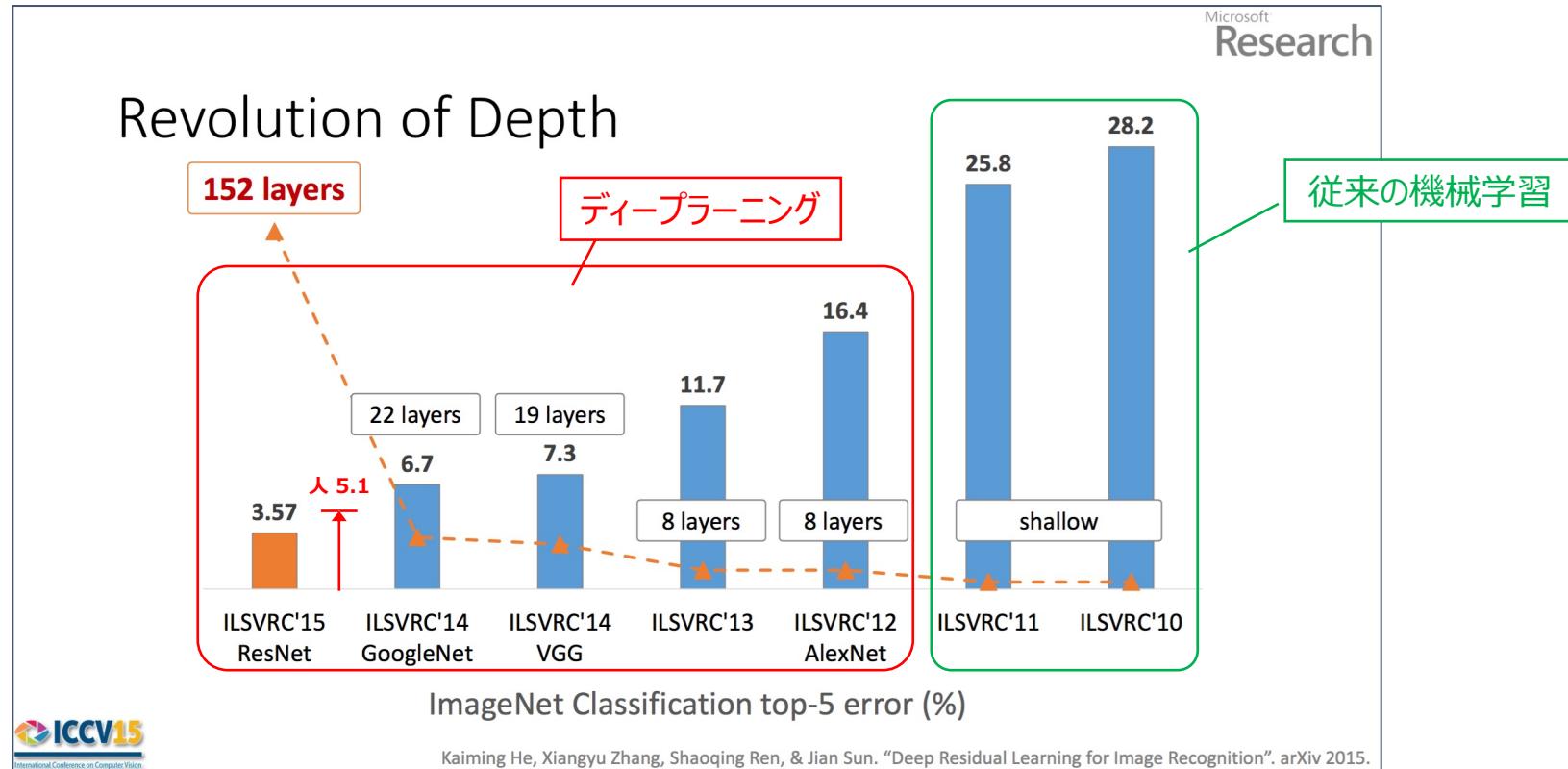
- 一般物体認識 (Hintonのグループ)
 - ImageNet Large-scale Visual Recognition Challenge 2012
 - 1000カテゴリ×約1000枚 = 100万枚 の訓練画像
 - 畳込み層5, 全結合層3, 2つのGPUで2週間 (AlexNet)
 - 誤識別率が10%以上減少 (過去数年間での向上は1~2%)



Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton.
"Imagenet classification with deep convolutional neural networks."
Advances in neural information processing systems. 2012.
<http://image-net.org/challenges/LSVRC/2012/supervision.pdf>

(参考) 画像認識で人を超える

- 2015年,人の認識精度(5.1%)を超えたことが話題に



word2vec の問題点

- 分布仮説に起因した以下の問題がある
 - 反義語: 共起する単語が似ているので,似たベクトルになりやすい
例)
 - 最終試験は**合格**だった
 - 最終試験は**不合格**だった
 - 多義語: 意味の異なる単語を同じベクトルにしようとする
例)
 - 昨日タイヤが**パンク**した
 - すきな音楽は**パンク**だ

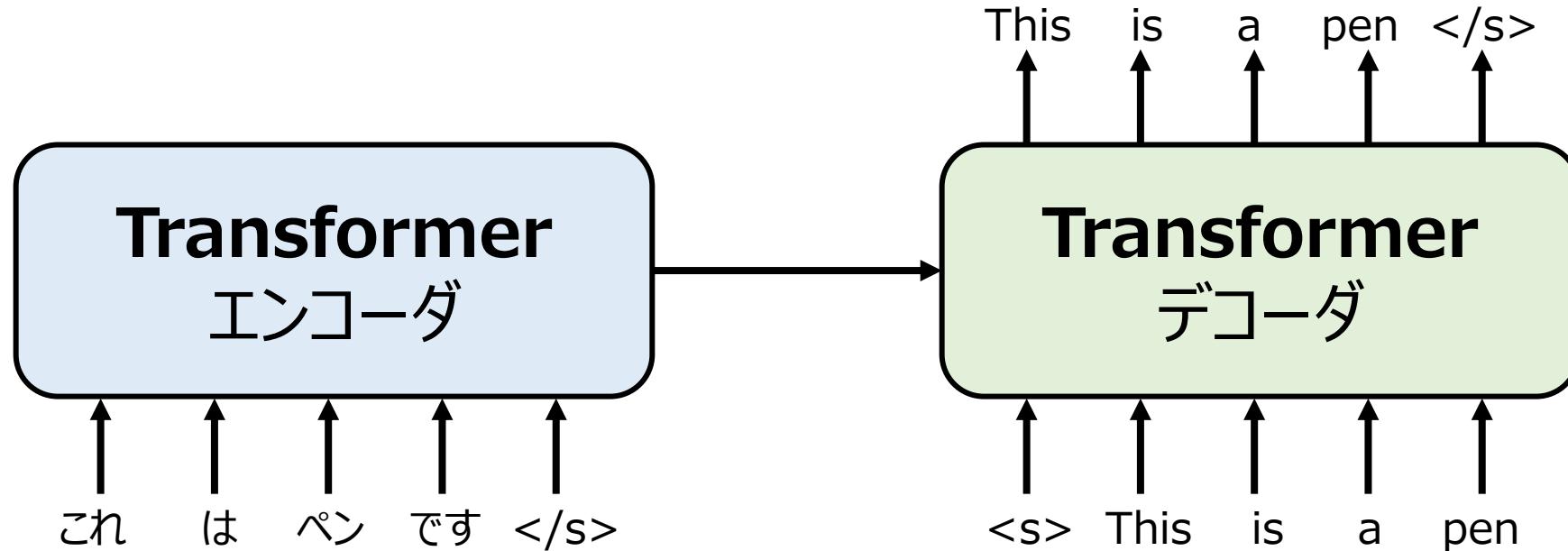
テキスト(文)の分散表現

- 文脈を考慮することで、様々なタスクの性能が向上していく

従来: 文脈に関係なく 一つの単語には一つのベクトルが割り当てられる	その後: 周りの文脈によって 同じ単語でも異なるベクトルが割り当てられる
<p>首を痛める</p> <p>首 </p> <p>会社を首になる</p> <p>首 </p>	<p>首を痛める</p> <p>首 </p> <p>会社を首になる</p> <p>首 </p>

Transformer の登場 [Vaswani+, NIPS2017]

- 単語間の関係をRNNやCNNを用いずアテンションのみを用いて表現したエンコーダデコーダ型モデルにより,機械翻訳で圧倒的な SOTA を達成

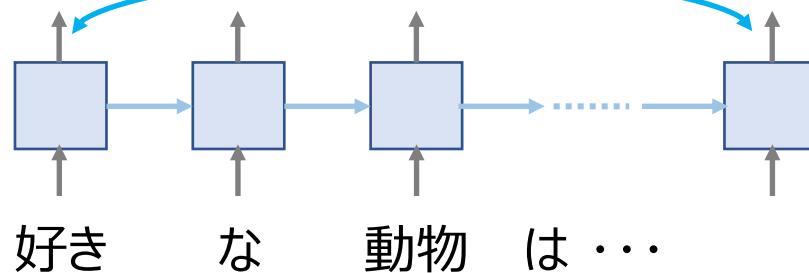


Transformer [Vaswani+, NIPS2017]

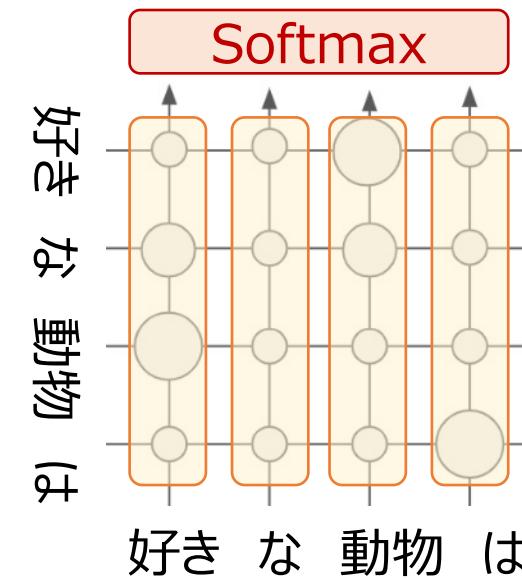
- ・従来の**文脈理解**は、長期依存性の理解に限界があった
- ・離れた単語の関係性も直接考慮できる Self-Attention が性能向上に大きく寄与した

従来(LSTM)

遠く離れた単語の関係性
を捕まえにくい



Self-Attention →
遠く離れた単語も直接
関係性を考慮できる



これまでの自然言語処理

- 基礎タスク

- 言語を応用タスクで利用しやすい形式に変換する

形態素解析

固有表現抽出

構文解析

照応解析

意味解析

談話解析

など

- 応用タスク

- 自然言語処理を応用したアプリケーション

テキスト検索

テキスト分類

テキスト要約

情報抽出

機械翻訳

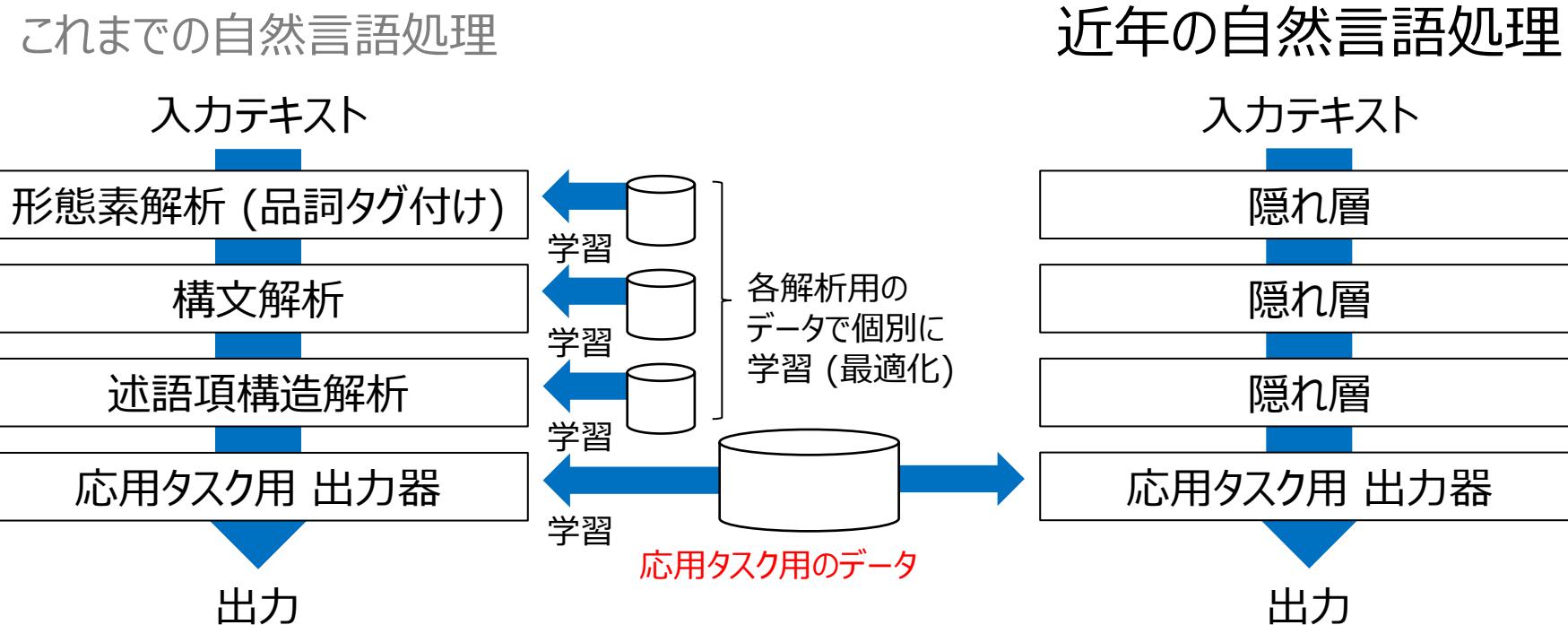
質問応答

対話

など

これまで→近年の自然言語処理

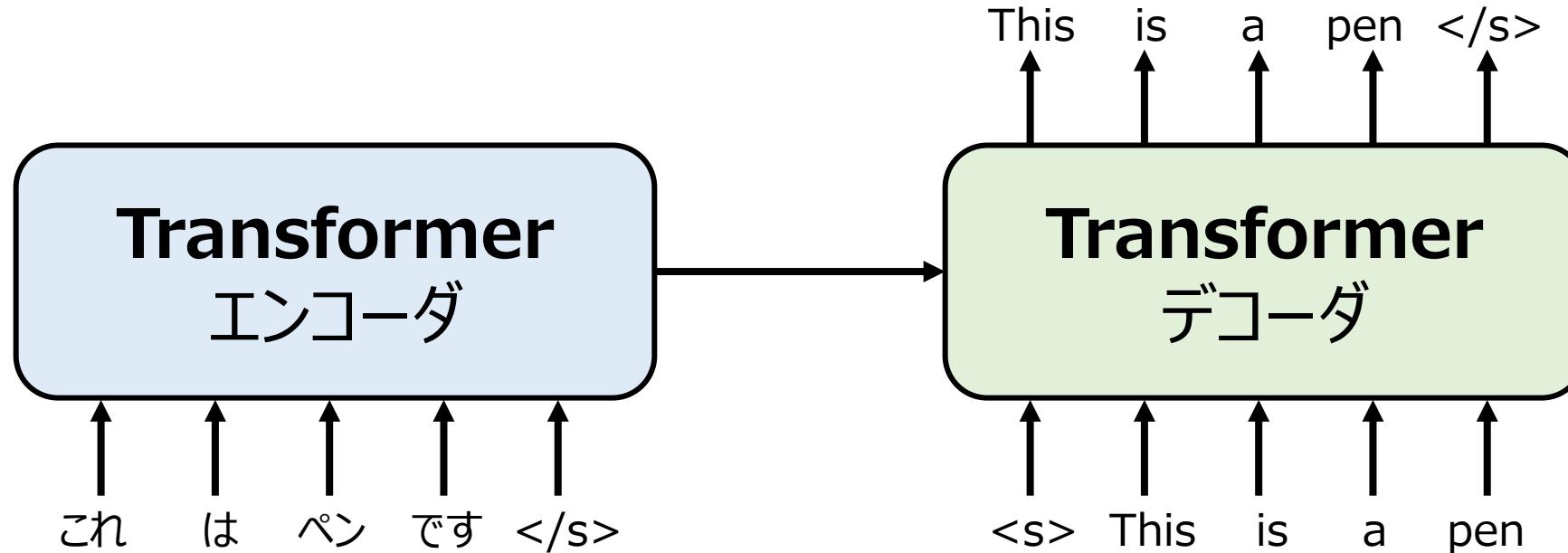
- 大規模な訓練データで応用タスク全体を学習 → End-to-end 学習



坪井, 海野, 鈴木. 深層学習による自然言語処理. 講談社, 2017, p.4 の図を一部修正

(再掲) Transformer [Vaswani+, NIPS2017]

- ・単語間の関係をRNNやCNNを用いずアテンションのみを用いて表現したエンコーダデコーダ型モデルにより、機械翻訳で圧倒的な SOTA を達成



最近の自然言語処理

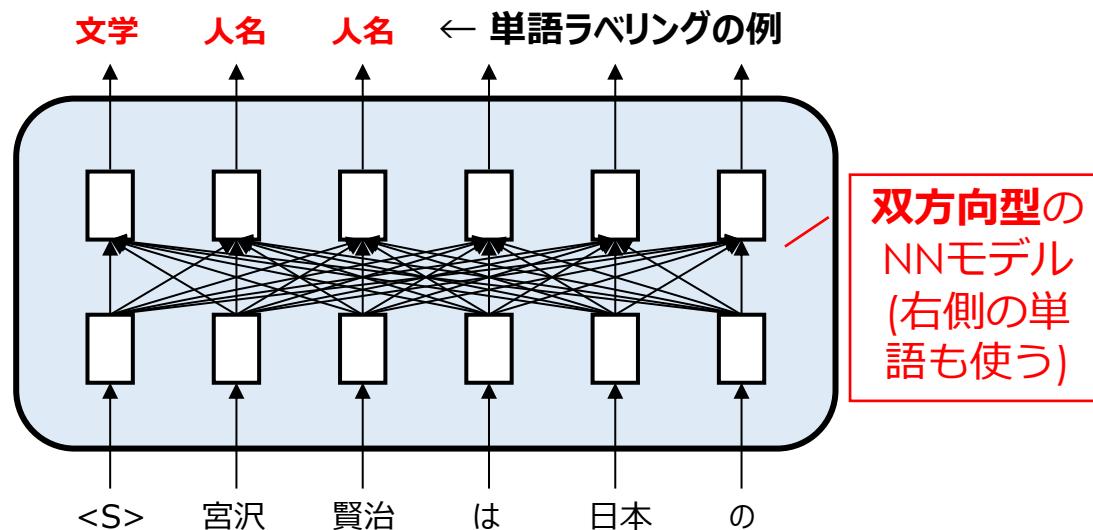
[西田,2022] JSAI2022 チュートリアル
講演資料の一部を修正して作成

- Transformerベースの事前学習モデルで,様々なタスクの性能を向上

エンコーダ型:

テキスト(単語系列)の**クラス分類**や
テキスト(単語系列)**単語ラベリング**など

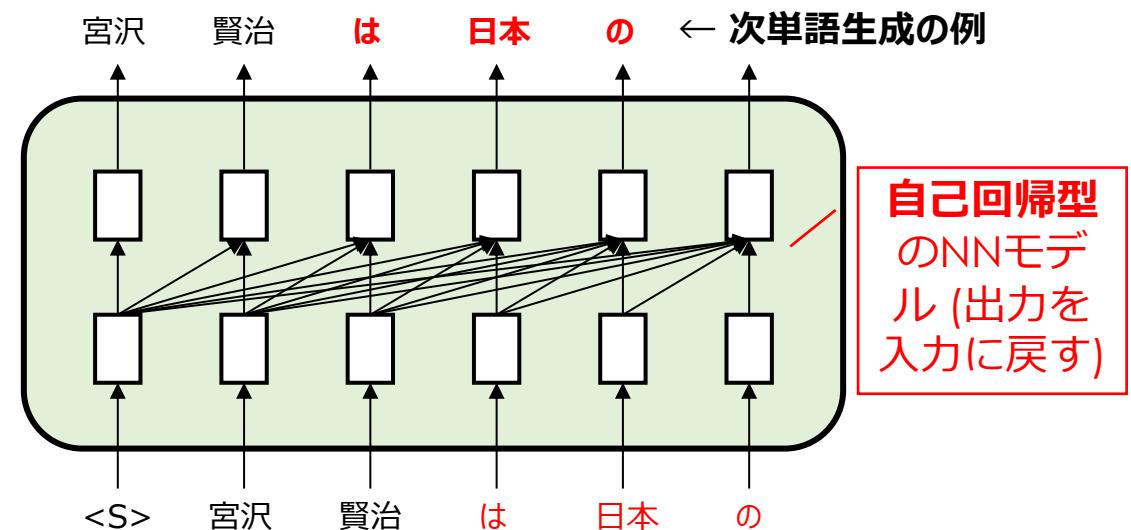
代表モデル: **BERT** [Devlin+, NAACL'19]



デコーダ型:

テキスト(単語系列)の続きを**生成**したり
テキストAからテキストBへの**変換(翻訳)**を行う

代表モデル: **GPT-3** [Brown+, NeurIPS'20]



これまでのまとめ

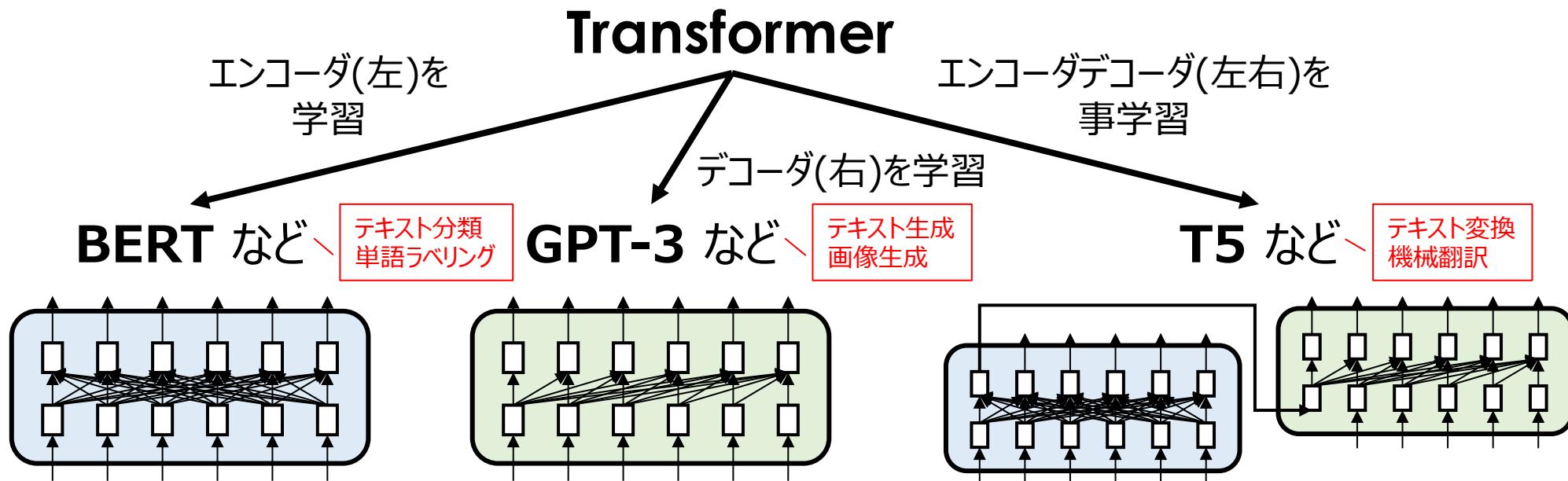
- 分布仮説
 - 似た文脈で出現する単語は意味が似ているという仮説
- 分散表現
 - 単語の意味を複数次元の実数値で分散して表現したもの
 - ニューラルネットワークで学習し, 次元は低次元 (例えば100次元)
 - word2vec が有名 → 反義語や多義語の問題がある
- **テキスト(文)の分散表現**
 - 文脈を考慮することで様々な自然言語処理タスクの性能が向上
 - Transformer ベースの BERT や GPT-3 が有名

最近の事前学習モデルに触れる

- **BERT**
 - [リクルート ELECTRA デモ](#) 
- **GPT-3**
 - <https://openai.com/api/> ※アカウント登録必要,90日過ぎると有料
 - <https://studio.ai21.com/> ※アカウント登録必要,無料で使える
- **DALL-E**
 - <https://huggingface.co/spaces/dalle-mini/dalle-mini>
- **CLIP**
 - <https://github.com/rinnakk/japanese-clip>
 - [日本語 CLIP デモ](#) 

Transformer=事前学習済みベースモデル

- ・近年の事前学習済みモデルの殆どが Transformer ベース
- ・コンピュータビジョンの分野にも Transformer の事前学習が派生



2018年10月：BERT の衝撃

- タスクに特化した構造を持たずに、人間のスコアを大きく超えた

SQuAD1.1 Leaderboard

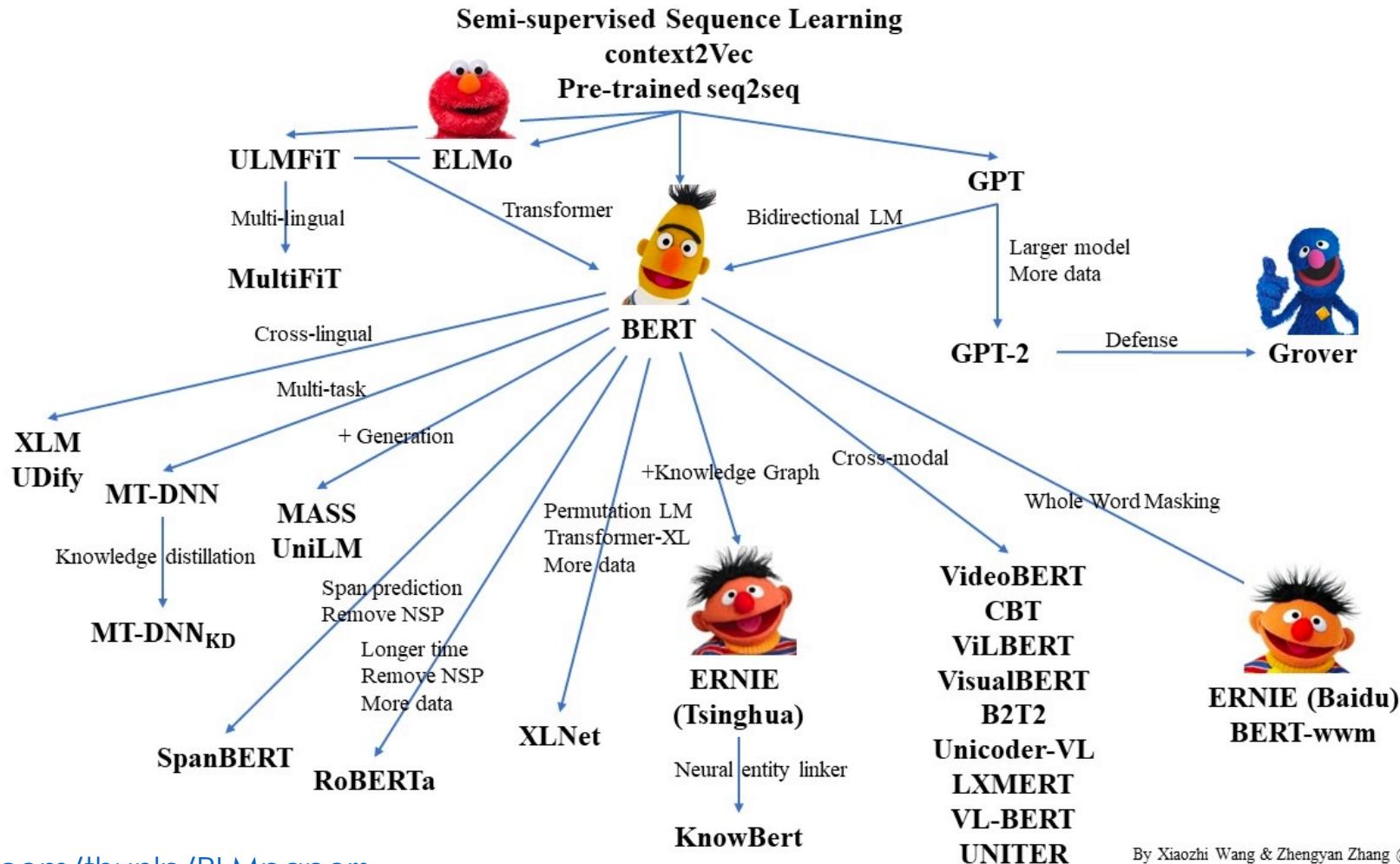
Since the release of SQuAD1.0, the community has made rapid progress, with the best models now rivaling human performance on the task. Here are the ExactMatch (EM) and F1 scores evaluated on the test set of SQuAD v1.1.

Rank	Model	EM	F1
	Human Performance <i>Stanford University</i> (Rajpurkar et al. '16)	82.304	91.221
1	BERT (ensemble) <i>Google AI Language</i> https://arxiv.org/abs/1810.04805	87.433	93.160
2	BERT (single model) <i>Google AI Language</i> https://arxiv.org/abs/1810.04805	85.083	91.835
2	nlnet (ensemble) <i>Microsoft Research Asia</i>	85.356	91.202

<https://rajpurkar.github.io/SQuAD-explorer/>

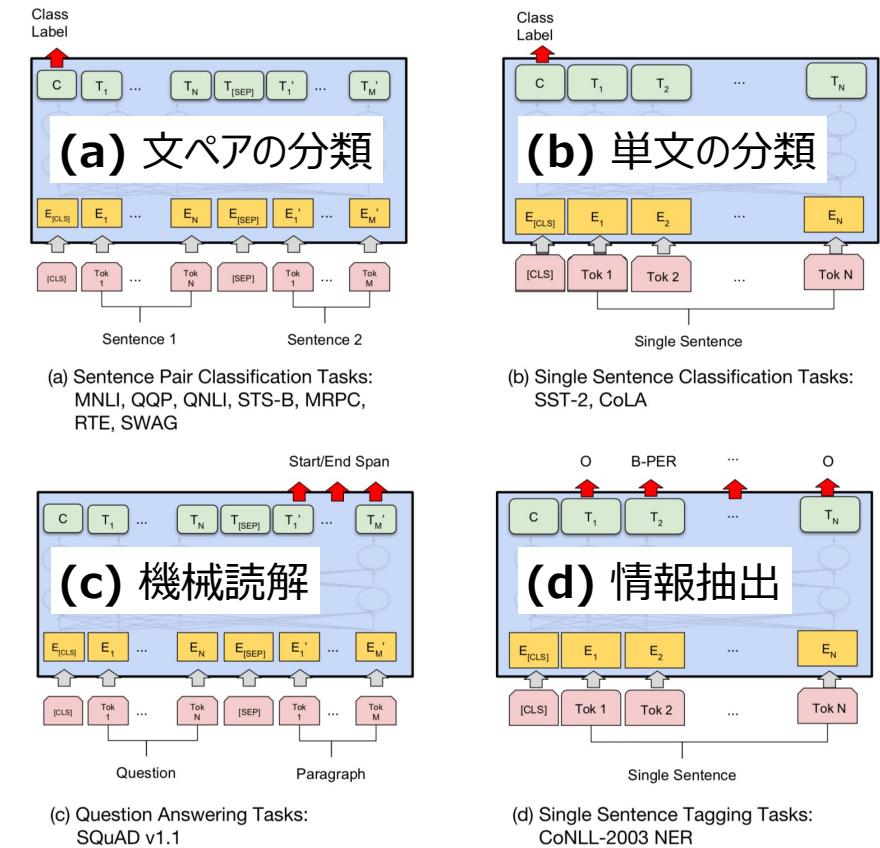
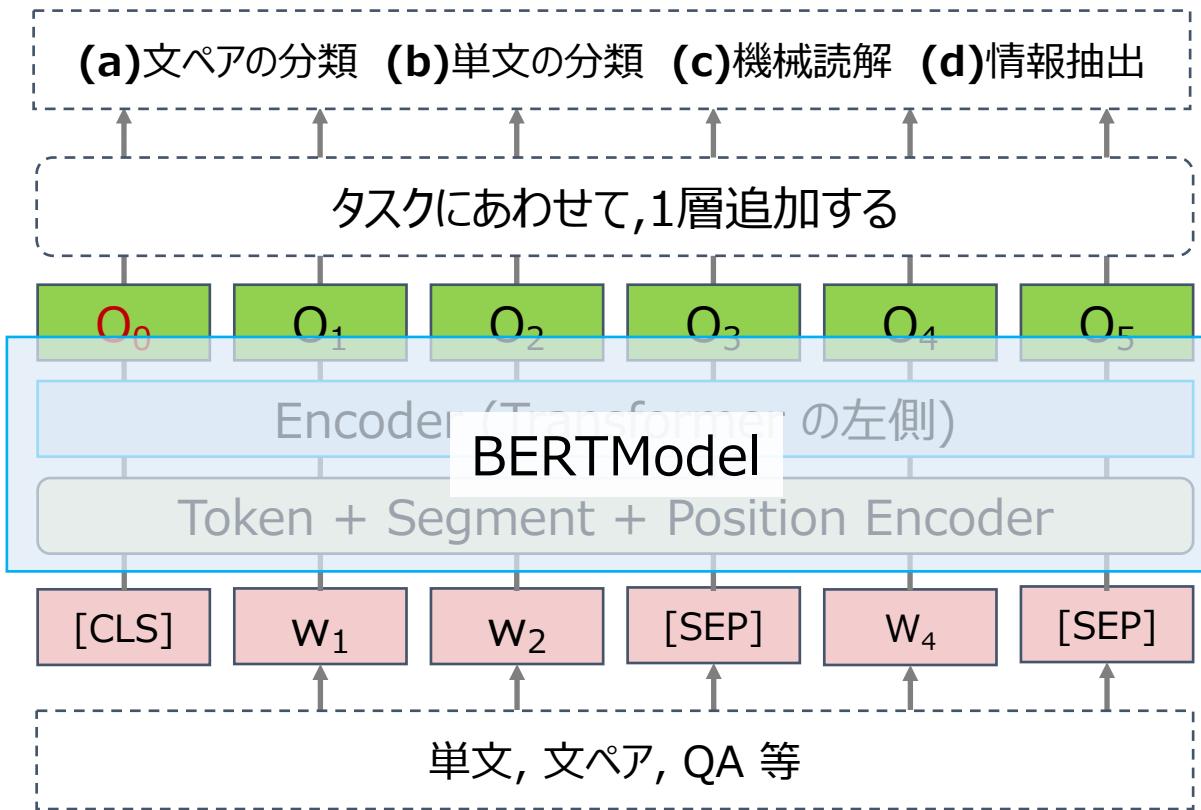
- 双向向 **Transformer** ブロックを24層重ねた言語モデル
- 機械読解タスク(左)で、完全一致と部分一致の両指標で最高精度(2018/10/5)
- 様々な自然言語理解タスクでSOTA(QA,含意,言い換え,NER等)

1年以内に BERT 改良モデルが続々登場



BERT fine-tuning

- 出力層をタスク毎に1層のみ追加して、様々なタスクに適応できる



事前学習モデルが公開

公開元	Google Research	京大 黒橋・河原・村脇研	NICT	東北大 乾・鈴木研
日/英	英語	日本語	日本語	日本語
コーパス	14GB (Book Corpus, Wikipedia)	3GB (Wikipedia)	3GB (Wikipedia)	3GB (Wikipedia)
単語数	30K (BPE)	32K (JUMAN & BPE)	32K (MeCab+JUMAN & BPE)	32K (MeCab+Neologd & BPE)
入力長 *1	最大512トークン	最大128トークン	最大512トークン	最大512トークン
パラメータ	24層, 各層1024次元	24層, 各層1024次元	12層, 各層768次元	12層, 各層768次元
学習時間	16Cloud TPUs で 4日間(≈100時間)	1GPU (GTX 1080 Ti) で 約 30日間(≈750時間)*2	32GPU (V100) で約 7日 間(≈175時間)	8Cloud TPUs で 約14日間(≈350時間)

*1 入力できるシーケンスの長さに制限があることに注意

*2 表中のパラメタは LARGEモデル, 学習時間のみ BASEモデル(12層, 768次元)の場合

区分	名前	モデル	事前学習テキスト	開発元	HuggingFace *1
汎用	京大BERT	BERT (base, large)	日本語 Wikipedia (約1,800万文)	京大 黒橋研	△
	東北大BERT	BERT (base, large)	日本語 Wikipedia (base (v1): 約1,700万文 (2.6GB), base (v2) & large: 約3,000万文 (4.0GB))	東北大 乾研	○ (base (v1), base (v2), large)
	NECT BERT	BERT (base)	日本語 Wikipedia (約2,000万文 (2.9GB))	NECT	△
	東大BERT	BERT (small)	日本語 Wikipedia (約2,000万文 (2.9GB))	東大 和泉研	○
	chiTra (Sudachi Transformers)	BERT (base)	国語研日本語ウェブコーパス (NWJC) (148GB)	NINJAL & ワークス徳島人工知能NLP研	△
	日本語DistilBERT	DistilBERT	- (東北大BERT(base) を親モデルとして知識蒸留)	BANDAI NAMCO Research	○
	rinna RoBERTa	RoBERTa (base)	日本語 Wikipedia + Japanese CC-100	rinna	○
	早大RoBERTa	RoBERTa (base, large)	日本語 Wikipedia + Japanese CC-100	早大 河原研	○ (base, large, large (seq512)) *2
	インフォマティクスRoBERTa	RoBERTa (base)	日本語 Wikipedia + Web 上の記事 (計25GB)	インフォマティクス	△
	シナモンELECTRA	ELECTRA (small)	日本語 Wikipedia	シナモン	○
	リクルートELECTRA	ELECTRA (base)	mC4 データセット内の日本語 (約2億文)	Megagon Labs (リクルート)	○
	東大ELECTRA	ELECTRA (small, base)	日本語 Wikipedia (約2,000万文 (2.9GB))	東大 和泉研	○ (small, base)
テキスト生成	rinna GPT/GPT-2	GPT/GPT-2 (xsmall, small, medium, gpt-1b)	日本語 Wikipedia + Japanese CC-100	rinna	○ (xsmall, small, medium, gpt-1b)
	早大GPT-2	GPT-2 (small)	日本語 Wikipedia + Japanese CC-100	早大 河原研	○
	日本語BART	BART (base, large)	日本語 Wikipedia (約1,800万文)	京大 黒橋研	
	日本語T5	T5 (base)	mC4 データセット内の日本語 (87,425,304 ページ (782 GB)) + wiki40b データセット内の日本語 (828,236 記事 (2 GB))	Megagon Labs (リクルート)	○
	日本語対話Transformer	Transformer	Twitter 上の日本語リプライのペア	NTT	
言語と画像	日本語CLIP	CLIP *3	CC12M のキャプションを日本語に翻訳したもの	rinna	○
	日本語CLOOB	CLOOB *3	CC12M のキャプションを日本語に翻訳したもの	rinna	

*1 ○: HuggingFace の Model Hub にモデルがあり AutoModel.from_pretrained() 等で読み込める △: Model Hub にはモデルがないが HuggingFace の形式に対応している

*2 nlp-waseda/roberta-base-japanese , nlp-waseda/roberta-large-japanese は最大トークン長を128で事前学習, nlp-waseda/roberta-large-japanese-seq512 は512で事前学習

*3 画像エンコーダは google/vit-base-patch16-224 で重みが初期化された ViT-B/16、テキストエンコーダは rinna RoBERTa で重みが初期化された RoBERTa(base))

HuggingFace's Transformers

<https://huggingface.co/>

- Huggingface が提供する Pytorch によるフレームワーク
- 簡単にBERTなどの汎用言語モデルを動かせる

The image shows three screenshots of the Hugging Face website:

- Left Screenshot:** A search bar at the top is highlighted with a red box and a red arrow pointing to it. The text "①モデルを検索" (Search for models) is overlaid on the left side.
- Middle Screenshot:** A search result for "japanese" is shown. One result, "cl-tohoku/bert-base-japanese-whole-word-masking", is highlighted with a red box and a red arrow pointing to it. The text "②サンプルコード" (Sample code) is overlaid on the right side.
- Right Screenshot:** A detailed view of the "cl-tohoku/bert-base-japanese-whole-word-masking" model card. It includes a "README.md" file link, a "Pretrained Japanese BERT models" section, and a "Features" section listing various model configurations and training details.

リクルート ELECTRA デモ

<https://github.com/haradatm/lecture/tree/master/gssm-202207/05-colab>

The image shows two screenshots. On the left is a screenshot of a README.md file. It contains a table with two rows. The first row has a 'file name' column with 'japanese_electra.ipynb' and a 'memo' column with 'Try the Japanese ELECTRA model.'. Below the table is a large red box highlighting the 'Open in Colab' button for the first row. A red arrow points from this button to the Colab screenshot on the right. The Colab screenshot shows a browser window titled 'japanese_electra.ipynb'. The code cell contains the following Python code:

```
!pip install -U ginza ja-ginza ja-ginza-electra > /dev/null
```

RESTART RUNTIME

```
[ ] import spacy
[ ] # ja-ginza モデルのロード
nlp = spacy.load('ja_ginza')

# 文の解析
doc = nlp('今日はいい天気です')

# doc はトークンを要素に持つイテレータになる
print([token for token in doc])

print(doc.text)      # doc 内のテキストの表示
print(doc.has_vector) # doc にベクトルが定義されているか (bool)
print(doc.vector.shape) # doc に対するベクトル

[今日, は, いい, 天気, です]
今日はいい天気です
True
(300,)

[ ] # doc を構成する単語を順番にイテレート
for token in doc:
    # sum(token.vector).shape ベクトルの要素の合計
```

GPT-3 [Brown+ (OpenAI), NeurIPS2020]

- GPT-1(1億パラメタ), GPT-2(15億パラメタ)と同じ自己回帰モデルだが**超大規模**
- 超大量(3000億トークン)のテキスト, 超巨大(96層)の Transformer デコーダで **1750億パラメタを学習** (例: BERTは, 3.3億トークン, 24層, 3.4億パラメタ)
- タスクの説明もテキストとして入力し, 様々なタスク(マルチタスク)を実現
 - **Zero-shot**: タスク説明のみ与え全くサンプルを与えない
 - **One-shot**: タスク説明と1つのサンプルのみを与える
 - **Few-shot**: タスク説明と少数(10から100)のサンプルを与える

The three settings we explore for in-context learning

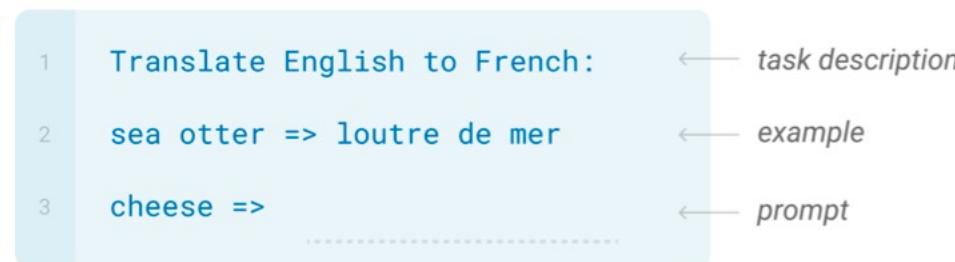
Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.



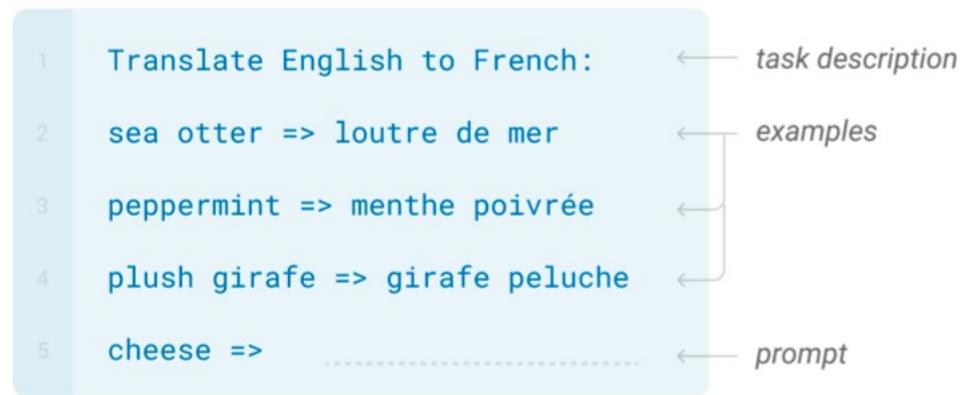
One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

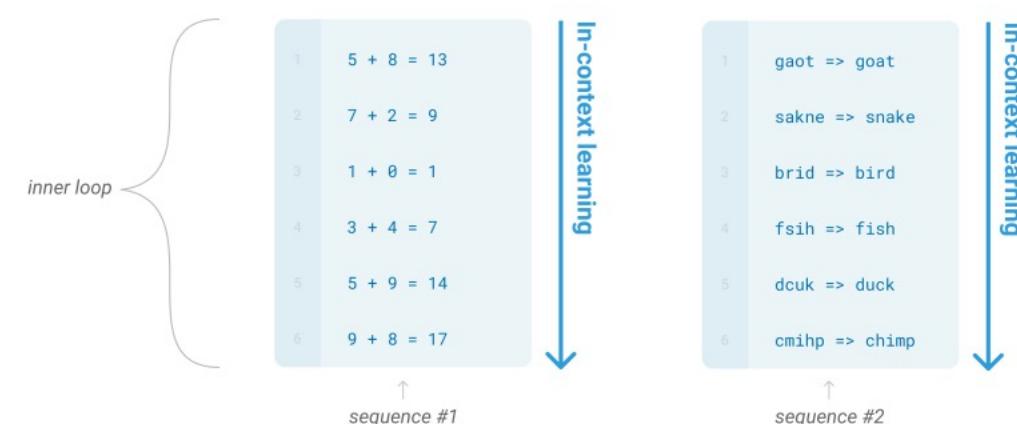


Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.



文脈内学習：同じタスクの繰り返しを含む系列を大量テキストの中から学習



GPT-3 デモ

<https://studio.ai21.com/>

イスラエルのスタートアップ企業
AI21からリリースされたGPT-3
と同サイズ(1780億)のモデル

The screenshot shows the AI21 Studio playground interface. On the left, a sidebar lists various examples like Chatbot, Twitter agent, and Product description generator. The main canvas displays a sequence of arithmetic operations:
 $5 + 8 = 13$
 $7 + 2 = 9$
 $1 + 0 = 1$
 $3 + 4 = 7$
 $5 + 9 = 14$
 $9 + 8 =$

On the right, the model configuration panel is visible, showing settings for the j1-jumbo (178B) model. The configuration includes:

- Max completion length: 64 (slider from 1 to 2048)
- Temperature: 0.7 (slider from 0 to 1)
- Top P: 1 (slider from 0.01 to 1)
- Stop sequences: Press Tab ↴ to apply
- Repetition penalties: (checkbox)
- Alternative tokens: (button)

GPT-3 デモ

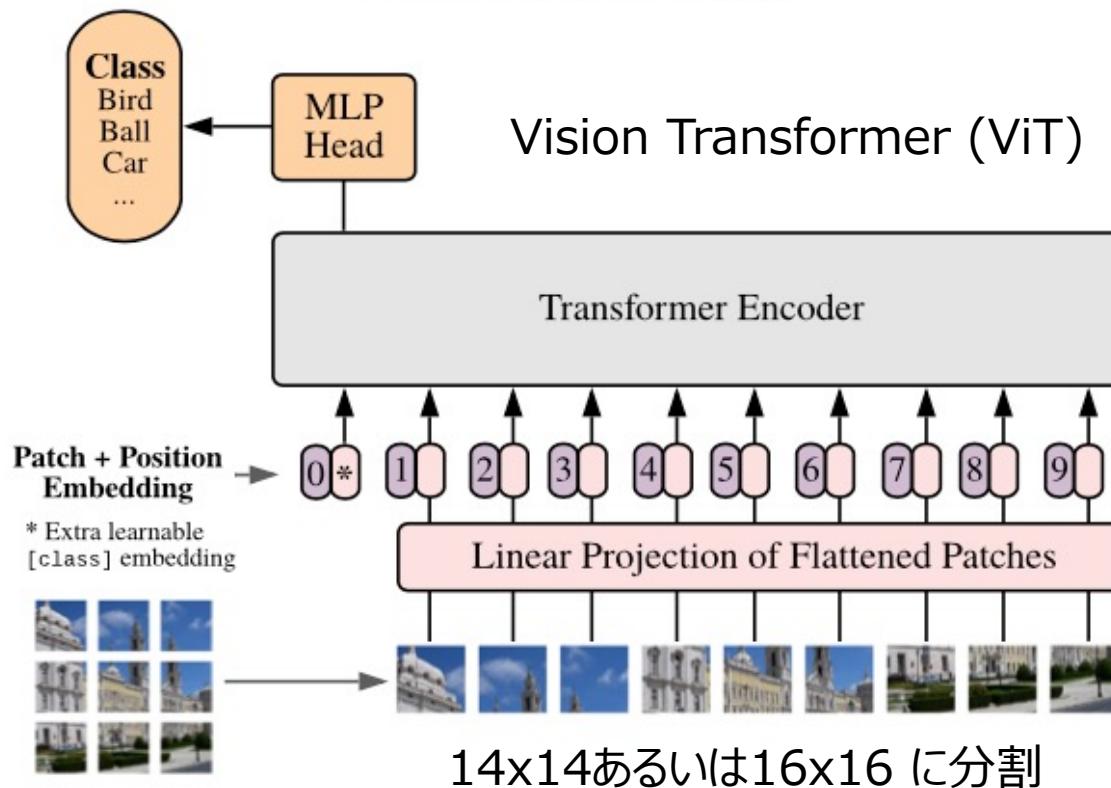
<https://studio.ai21.com/>

イスラエルのスタートアップ企業
AI21からリリースされたGPT-3
と同サイズ(1780億)のモデル

The screenshot shows the AI21 Studio interface. On the left, there's a sidebar with various examples like Chatbot, Twitter agent, Ads copywriter, etc. The main canvas displays the text "西暦から和暦に変換します。" followed by three examples of conversion: "西暦2021年 => 令和3年", "西暦1967年 => 昭和42年", and "西暦1900年 =>". To the right, there are model selection and configuration options. The model selected is "j1-jumbo (178B)". The configuration includes sliders for "Max completion length" (set to 64), "Temperature" (set to 0.7), "Top P" (set to 1), and a "Stop sequences" input field. A large blue button labeled "▶ Generate" is at the bottom.

Vision Transformer (ViT) [Dosovitskiy+, 2021]

- Transformer は画像認識などの NLP以外でも成果を発揮



- 画像パッチを単語とみなす6.32 億パラメタの Transformerエンコーダ
- 画像は最初にパッチに分割した後、線形変換で埋め込み
- 3億枚以上の画像分類で事前学習し、ImageNet 等で SOTA

https://github.com/google-research/vision_transformer

DALL·E [Ramesh+ (OpenAI), 2021]

- OpenAI が発表した文章に忠実な画像を生成するモデル

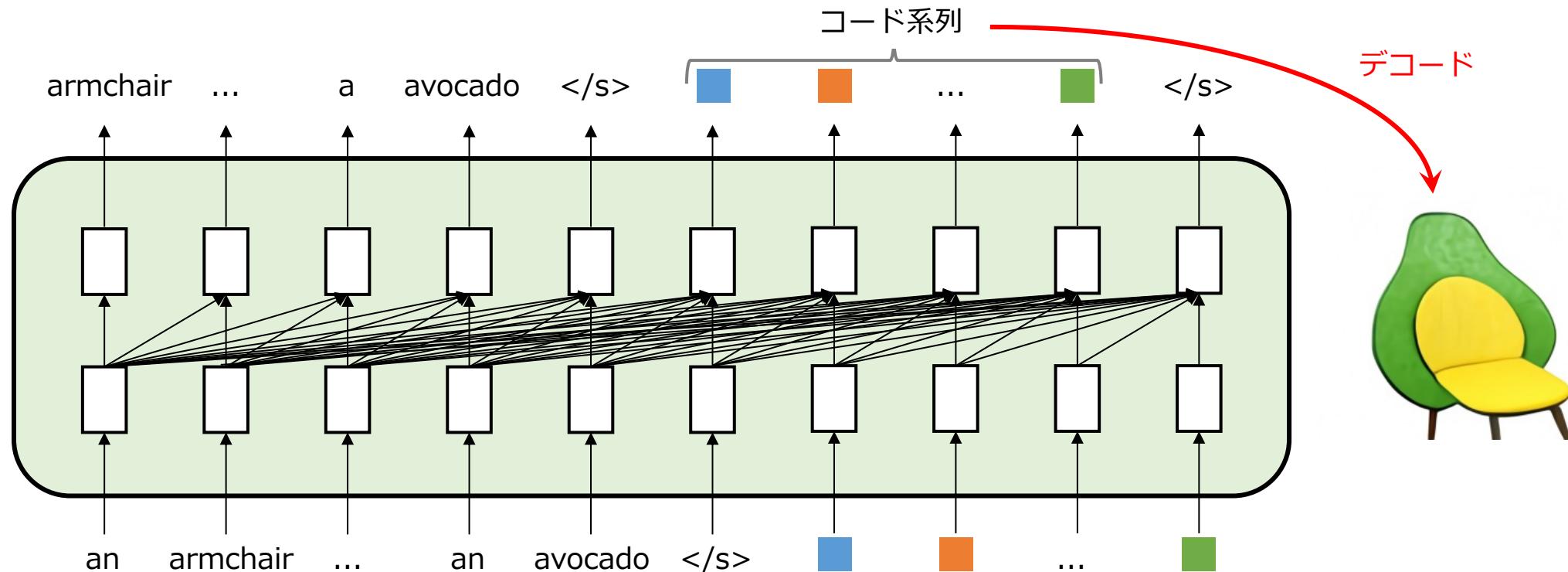


- 巨大な Transformer デコーダによる Text-to-image モデル
 - 最大 120億パラメタ (ViTの約20倍)
- 大量の画像と説明文ペアから学習、生成画像のレベルが高い
- 画像は1024(32x32)のコード系列(8192種)として扱う

<https://openai.com/blog/dall-e/>

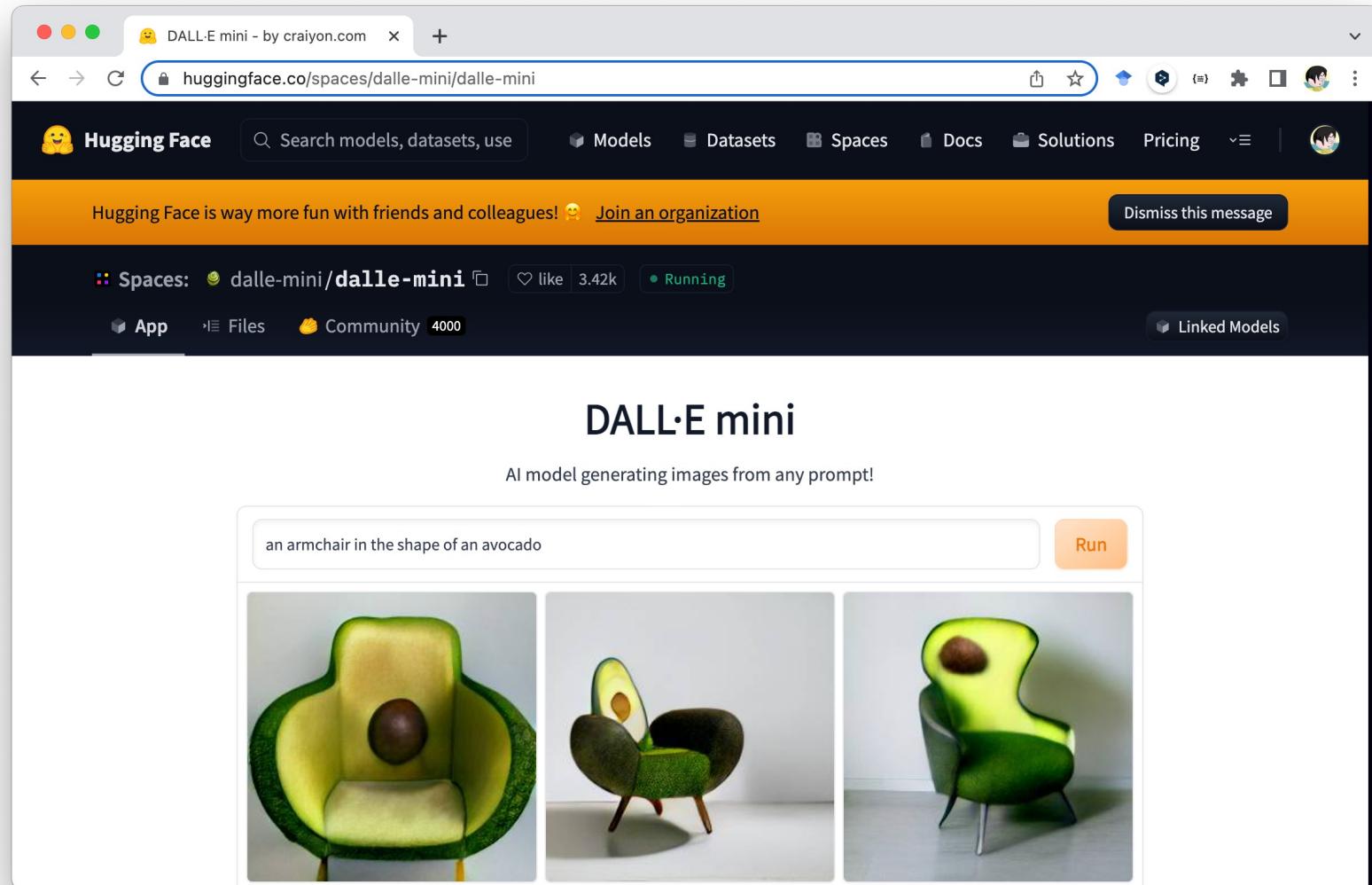
DALL·E [Ramesh+ (OpenAI), 2021]

- 画像をコード系列に変換し,テキストからコード系列を生成するよう学習
 - 入力画像(256x256)→コード系列(32x32,8192種) 変換は VAEベースの変換器



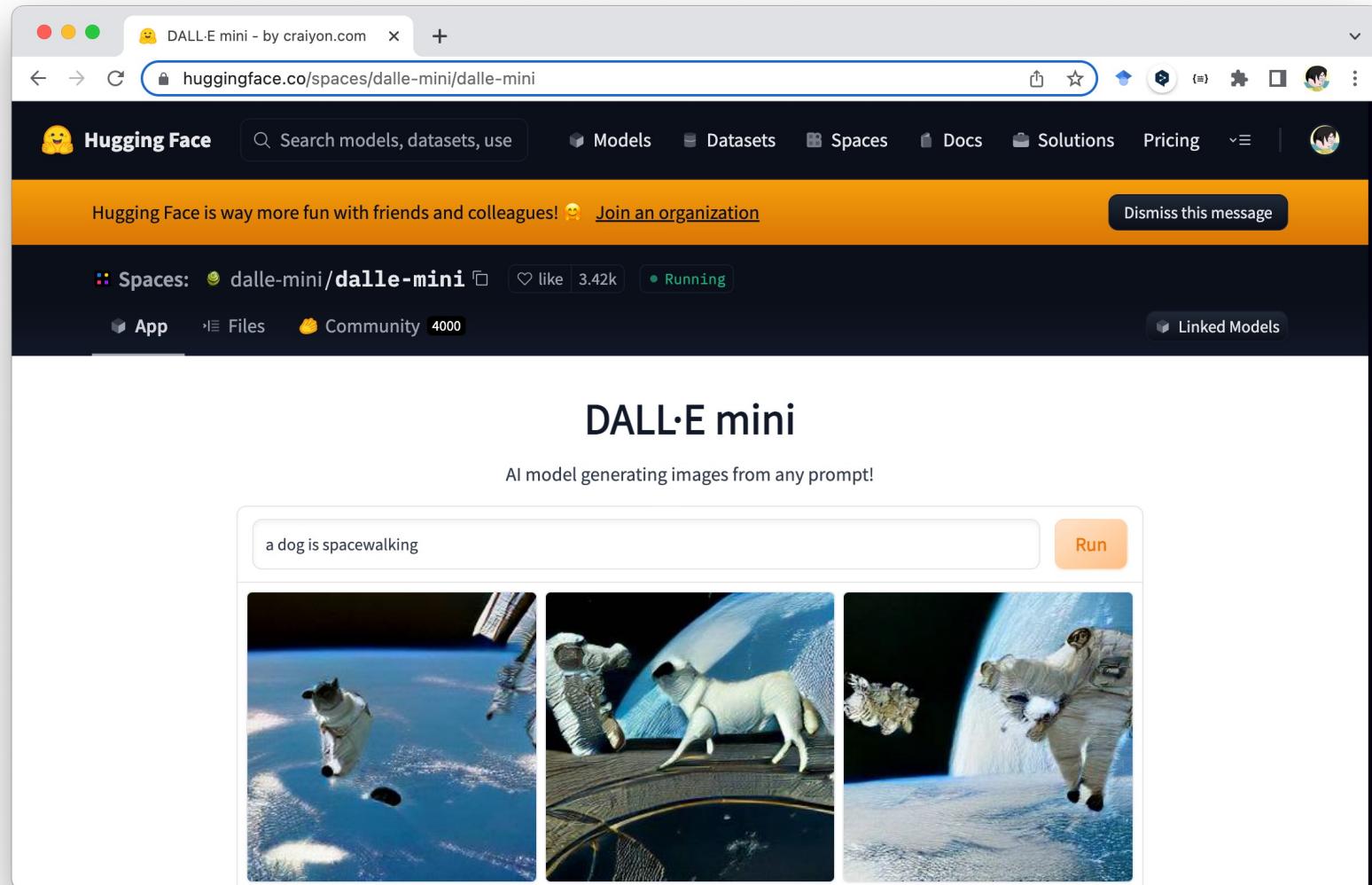
DALL·E デモ

<https://huggingface.co/spaces/dalle-mini/dalle-mini>



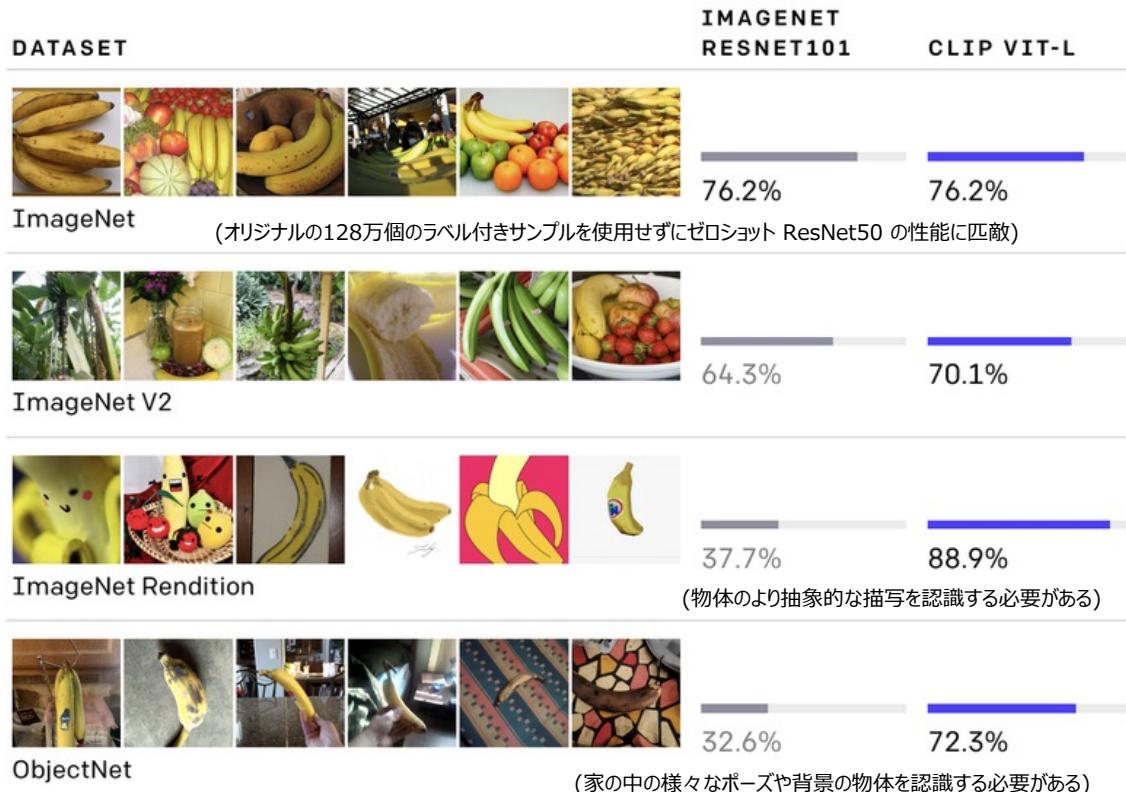
DALL·E デモ

<https://huggingface.co/spaces/dalle-mini/dalle-mini>



CLIP [Radford+ (OpenAI), ICML2021]

- 大規模な画像とテキストのペアで zero-shot の画像認識を実現



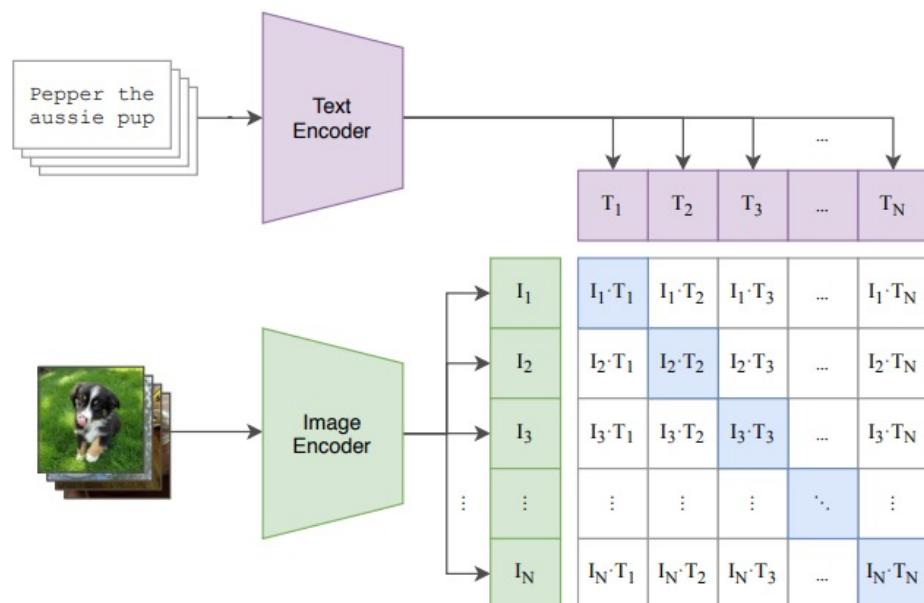
- Webから収集した画像とテキストのマッチングを4億ペアから事前学習
- 正しい画像と説明文のペアが近く(内積が大きくなる)ように対照学習
- 事前学習後は zero-shot で画像とマッチするテキスト候補を選ぶ
- DALL·E の生成画像のランキングにも使われている

<https://openai.com/blog/clip/>

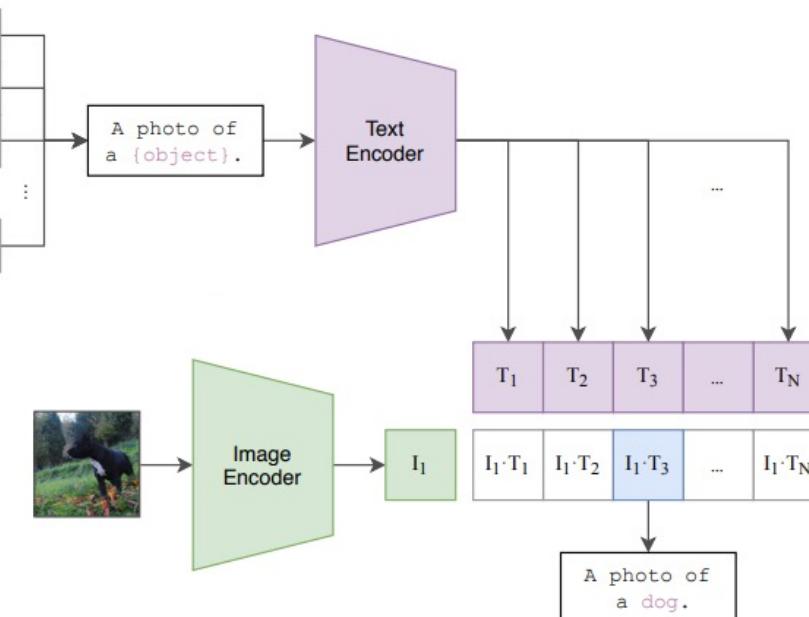
CLIP [Radford+ (OpenAI), ICML2021]

- ・テキストと画像の表現との内積が大きくなるように両方のエンコーダを学習
 - ・推論時には、任意のテキストの候補の中から画像にマッチするものを選ぶ

左: 学習時



右: 推論時



日本語 CLIP <https://github.com/rinnakk/japanese-clip>

- rinna社 CC12M データセット※の1200万の言語・画像ペアを日本語に翻訳して学習した 日本語CLIP を公開

The screenshot shows the `README.md` file from the GitHub repository. It includes a "Usage" section with two steps: 1. Install package (using pip) and 2. Run a Python script. The script demonstrates how to load the Japanese CLIP model, preprocess an image of a dog, and get image and text features, then calculate and print the label probabilities.

```
## README.md



### Usage



1. Install package  
$ pip install git+https://github.com/rinnakk/japanese-clip.git
2. Run  


```
from PIL import Image
import torch
import japanese_clip as ja_clip

device = "cuda" if torch.cuda.is_available() else "cpu"
ja_clip.available_models()
['rinna/japanese-clip-vit-b-16', 'rinna/japanese-cloob-vit-b-16']
model, preprocess = ja_clip.load("rinna/japanese-clip-vit-b-16", cache_dir="/tmp/japanese_clip", device=device)
tokenizer = ja_clip.load_tokenizer()

image = preprocess(Image.open("./data/dog.jpeg")).unsqueeze(0).to(device)
encodings = ja_clip.tokenize(
 texts=["犬", "猫", "象"],
 max_seq_len=77,
 device=device,
 tokenizer=tokenizer, # this is optional. if you don't pass, load tokenizer each time
)

with torch.no_grad():
 image_features = model.get_image_features(image)
 text_features = model.get_text_features(**encodings)

 text_probs = (100.0 * image_features @ text_features.T).softmax(dim=-1)

print("Label probs:", text_probs) # prints: [[1.0, 0.0, 0.0]]
```

```

- 言語モデルに rinna社公開の日本語 BERT(1.1億パラメタ) を利用
- 8つの NVIDIA Tesla A100 GPU (80GBメモリ) で学習
- Hugging Face に商用利用可能な Apache-2.0 Licenseで公開 (→左図)

日本語 CLIP デモ

<https://github.com/haradatm/lecture/tree/master/gssm-202207/05-colab>

The image shows a GitHub README.md page on the left and a Colab session on the right. The README.md page lists two scripts: 'japanese_electra.ipynb' and 'japanese_clip.ipynb'. The 'japanese_clip.ipynb' row has a red box around the 'Open in Colab' button. A red arrow points from this button to the Colab session on the right. The Colab session shows the execution of commands to install the Japanese CLIP model and print available models.

README.md

Sample code for [Open in Colab](#) [Open](#) [Studio Lab](#)

Tiny scripts to trial latest NLP models (introduced in the)

file name	memo
japanese_electra.ipynb	Try the Japanese ELECTRA model.
japanese_clip.ipynb	Try the Japanese CLIP model.

Open in Colab

クリックすると Colab にジャンプ

japanese_clip.ipynb - Colaboratory

pip install -U git+https://github.com/rinnakk/japanese-clip.git > /dev/null

Running command git clone -q https://github.com/rinnakk/japanese-clip.git /tmp/pip-req-build-xhdvmzau

[] import torch
import japanese_clip as ja_clip

[] device = "cuda" if torch.cuda.is_available() else "cpu"
print(device)

cpu

[] ja_clip.available_models()

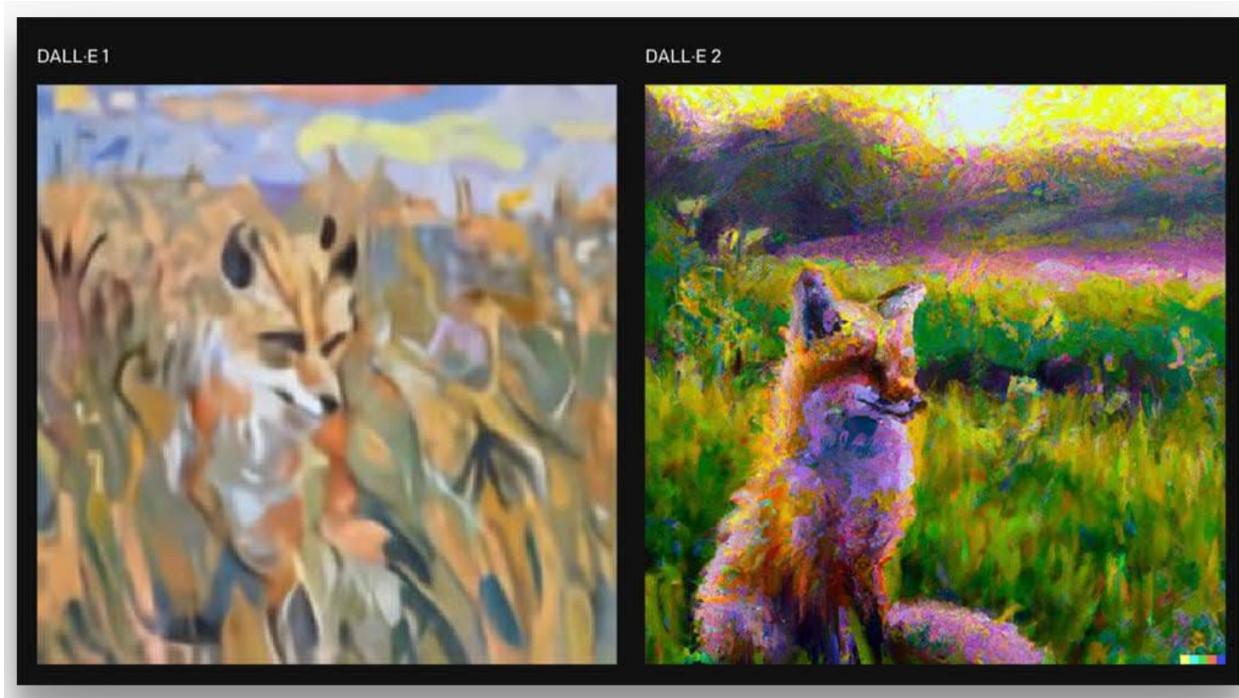
[] ['rinna/japanese-clip-vit-b-16', 'rinna/japanese-cloob-vit-b-16']

[] %mkdir -p ./data
!wget https://github.com/rinnakk/japanese-clip/raw/master/data/dog.jpeg -P ./data

--2022-06-26 11:41:33-- <https://github.com/rinnakk/japanese-clip/raw/master/data/dog.jpeg>
Resolving github.com (github.com)... 140.82.113.3
Connecting to github.com (github.com)|140.82.113.3|:443... connected.
HTTP request sent, awaiting response... 302 Found
Location: <https://raw.githubusercontent.com/rinnakk/japanese-clip/master/data/dog.jpeg> [following]
--2022-06-26 11:41:33-- <https://raw.githubusercontent.com/rinnakk/japanese-clip/master/data/dog.jpeg>
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.110.133, 185.199.1
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK

DALL·E2 [Ramesh+ (OpenAI), 2022/04]

- CLIP+拡散モデル[Ho+, NeurIPS2020]によるテキストからの画像生成
 - 拡散モデルにより,**さらに高品質**の画像が生成できるようになっている

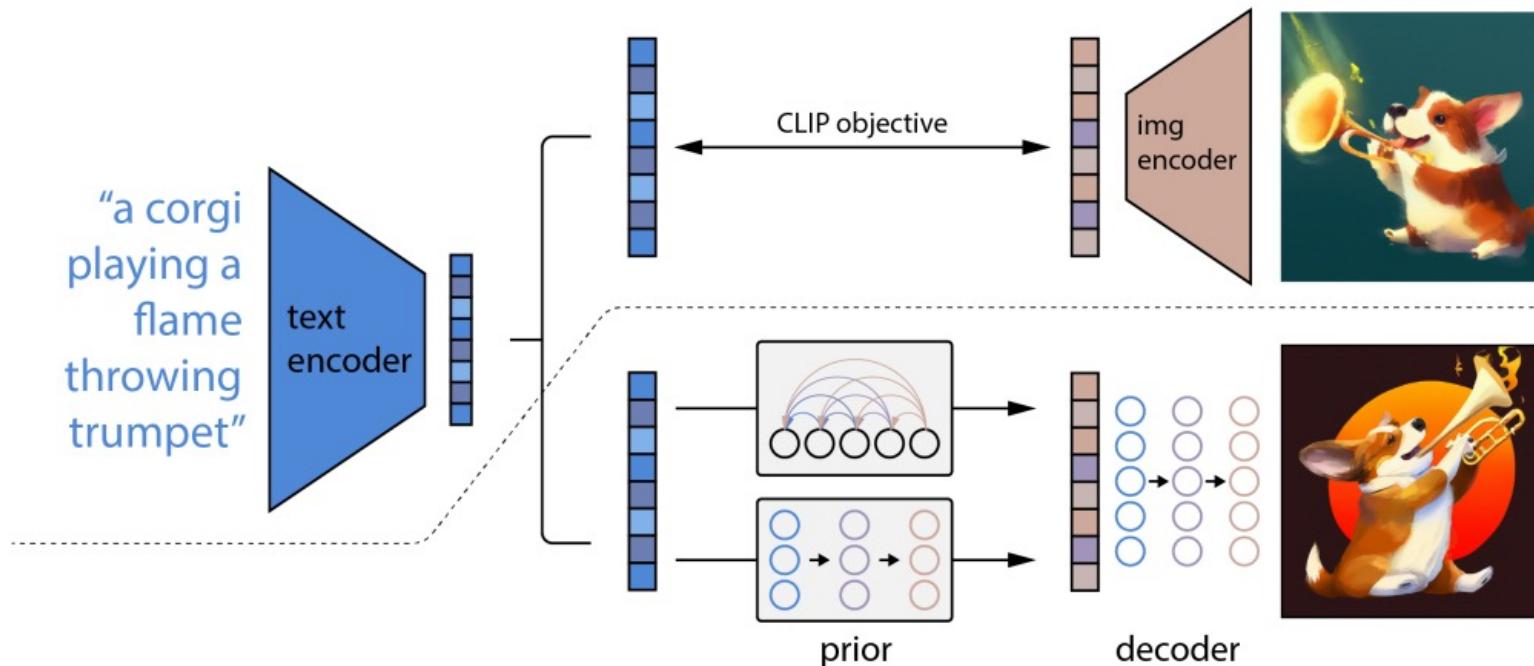


- DALL·E2 では,解像度を4倍に高め,よりリアルで正確な画像を生成できる
 - 左図は “A painting of a fox sitting in a field at sunrise in the style of Claude Monet” (クロード・モネ風に描かれた、朝日に照らされながら野原に座るキツネの絵)

<https://cdn.openai.com/papers/dall-e-2.pdf>

DALL·E2 [Ramesh+ (OpenAI), 2022/04]

- 上は、テキスト表現から画像表現を生成する CLIP のプロセス
- 下は CLIP のテキスト表現から画像を生成するプロセス



- CLIP のテキスト表現は prior (自己回帰モデル or 拡散モデル) を介して decoder に送られる
- decoder の学習中は、CLIP モデルは凍結

<https://cdn.openai.com/papers/dall-e-2.pdf>

Imagen [Saharia+ (Google), 2022/05]

- DALL-E2 の1ヶ月後に,同じく拡散モデルによるテキストからの画像生成
 - DALL-E2 がテキスト埋め込みに CLIP を使っていったのに対して,T5 を使用



A storefront with Text to Image written on it.

- DALL-E2 が苦手としている文字列をふくんだ画像を適切に生成できる
- テキスト埋め込みを生成する言語モデルのサイズを大きくしたほうが,拡散モデルのサイズを大きくするより生成画像の品質が向上すると主張

<https://imagen.research.google/>

最近の事前学習モデルに触れる のまとめ

- Transformer の事前学習がコンピュータビジョンの分野にも派生
- CLIP は, **画像とテキスト**(任意の言語)を結びつけ,**視覚と言語**の融合による最近の急速なモデル進化につながっている
- 最近では, 画像・テキスト検索のみならず, テキストからの画像生成など**様々なタスクでCLIPの導入が進みそ**う

(参考) CO Colaboratory とは

- 機械学習の教育・研究を目的とした研究用ツール



- 設定不要** (最初から Python や機械学習に必要なものが入っている)
- 無料で使える** (Googleアカウントさえあれば良い)
- ブラウザで動作する** (PCのスペックが低くても関係なし)
- GPUが無料で使える** (計算時間を大幅に短縮できる)
- ただし、90分&12時間ルールあり** ^{*1}

*1 Colab Pro (1,072円/月)にすることで各種制限を緩和できます

