

# Backpack Language Models

John Hewitt, John Thickstun, Christopher D. Manning, Percy Liang  
ACL2023 Outstanding Paper

---

発表者：稲葉 達郎

M1, Audio and Speech Processing Lab, Kyoto University

2023/8/27 第15回最先端 NLP 勉強会

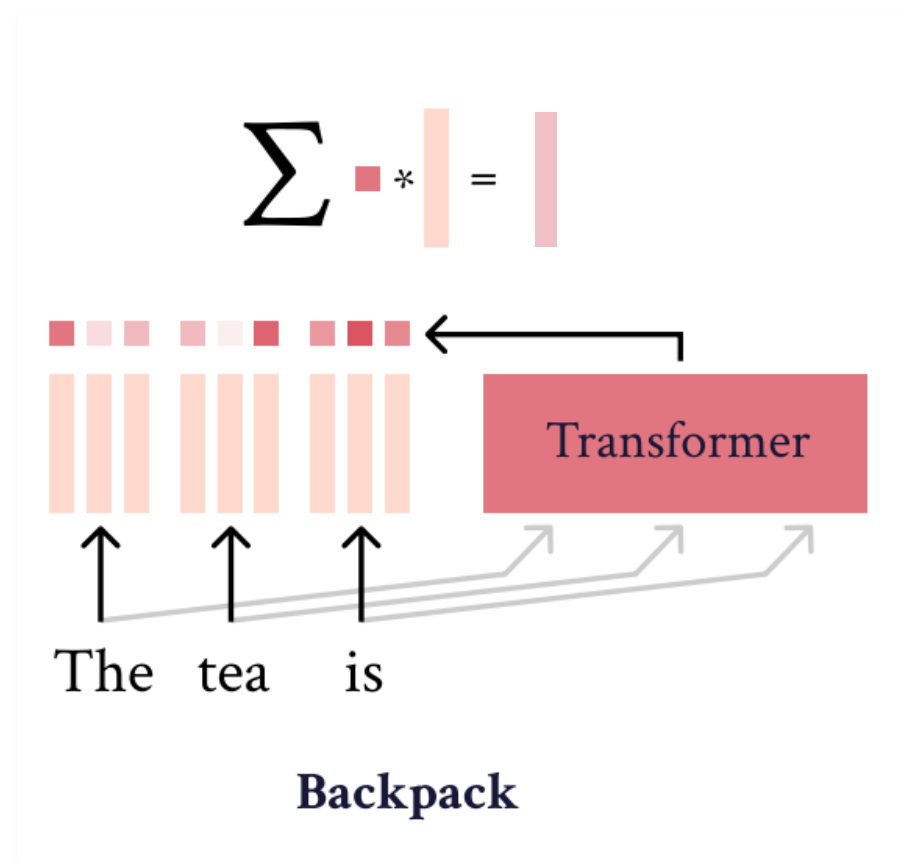
# 概要

## 手法

- 各単語をそれぞれ文脈に依存しない複数の Sense ベクトルに変換
- Transformer により文脈に依存した重みを計算
- Sense ベクトルの重み付き和をとる

## 特徴

- Transformer と同等の性能
- 文脈に依存しない Sense ベクトルを組み込むことで操作性・解釈性が向上



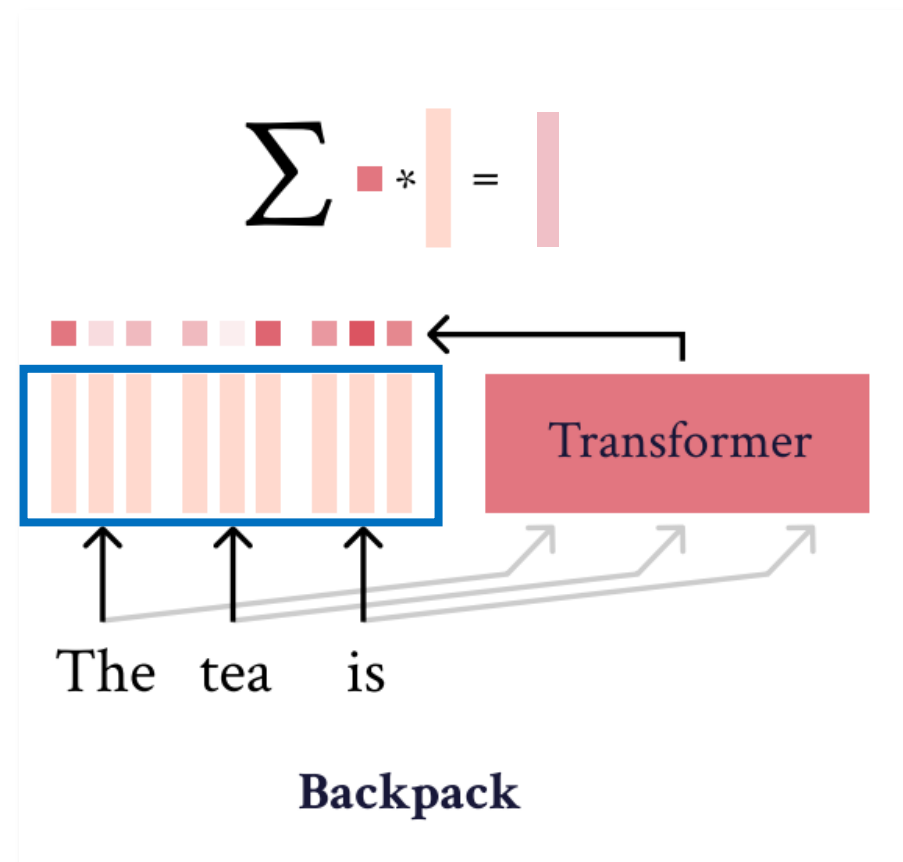
# 概要

## 手法

- 各単語をそれぞれ文脈に依存しない複数の Sense ベクトル に変換
- Transformer により文脈に依存した重みを計算
- Sense ベクトルの重み付き和をとる

## 特徴

- Transformer と同等の性能
- 文脈に依存しない Sense ベクトルを組み込むことで操作性・解釈性が向上



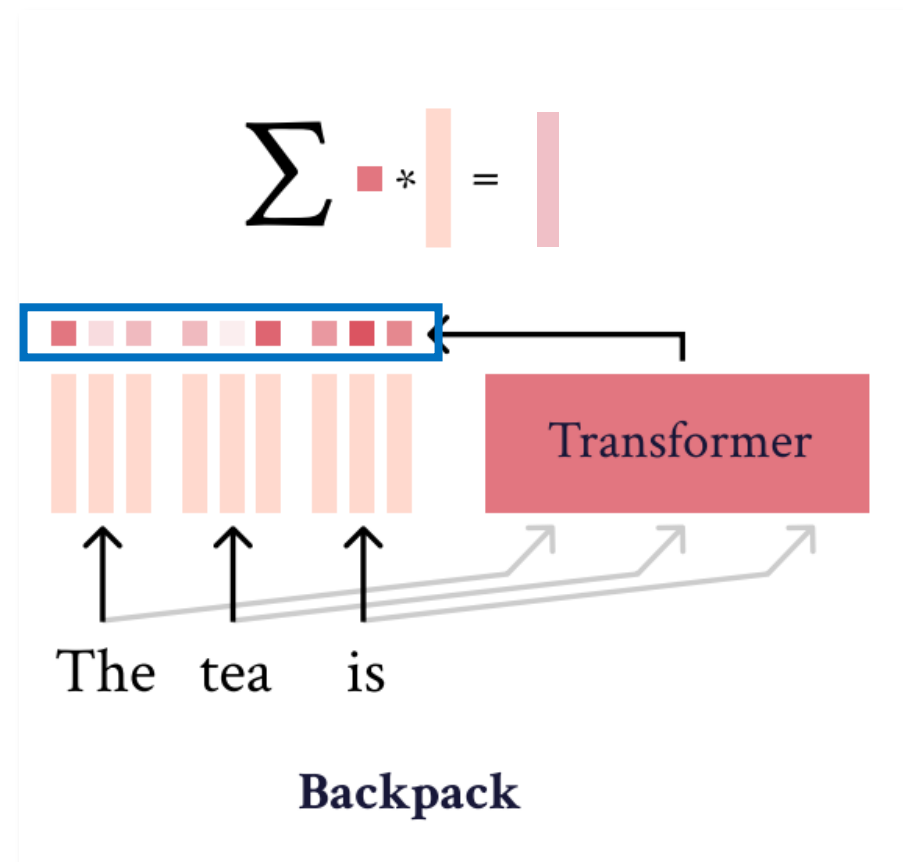
# 概要

## 手法

- 各単語をそれぞれ文脈に依存しない複数の Sense ベクトルに変換
- Transformer により 文脈に依存した重み を計算
- Sense ベクトルの重み付き和をとる

## 特徴

- Transformer と同等の性能
- 文脈に依存しない Sense ベクトルを組み込むことで操作性・解釈性が向上



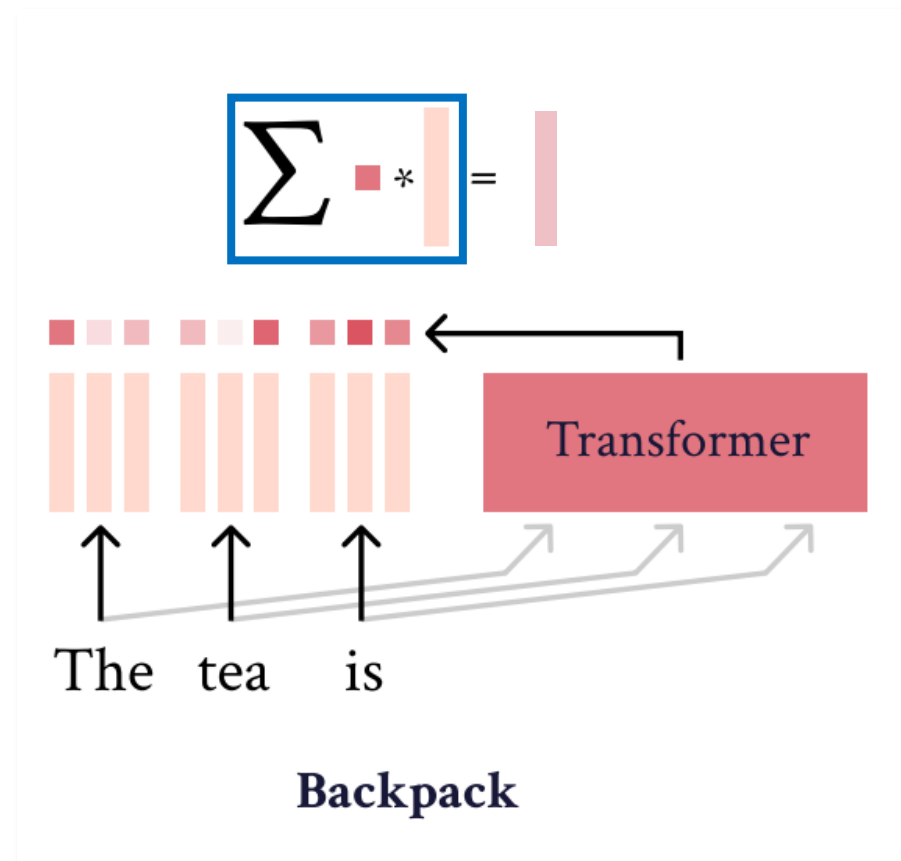
# 概要

## 手法

- 各単語をそれぞれ文脈に依存しない複数の Sense ベクトルに変換
- Transformer により文脈に依存した重みを計算
- Sense ベクトルの重み付き和をとる

## 特徴

- Transformer と同等の性能
- 文脈に依存しない Sense ベクトルを組み込むことで操作性・解釈性が向上



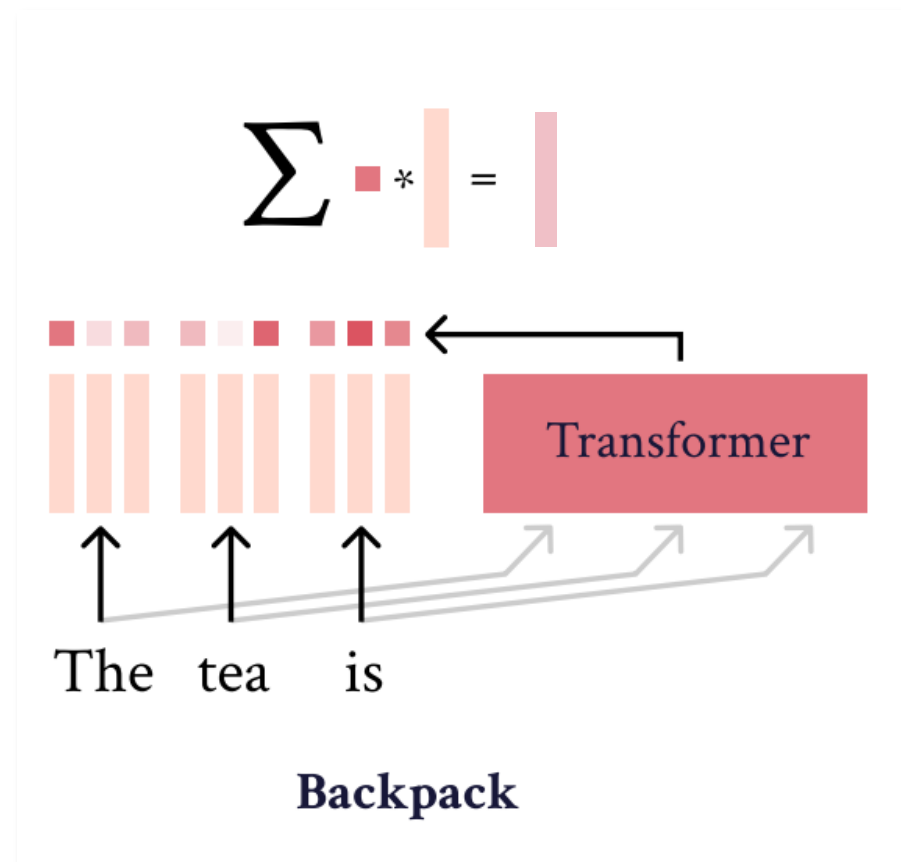
# 概要

## 手法

- 各単語をそれぞれ文脈に依存しない複数の Sense ベクトルに変換
- Transformer により文脈に依存した重みを計算
- Sense ベクトルの重み付き和をとる

## 特徴

- Transformer と同等の性能
- 文脈に依存しない Sense ベクトルを組み込むことで操作性・解釈性が向上

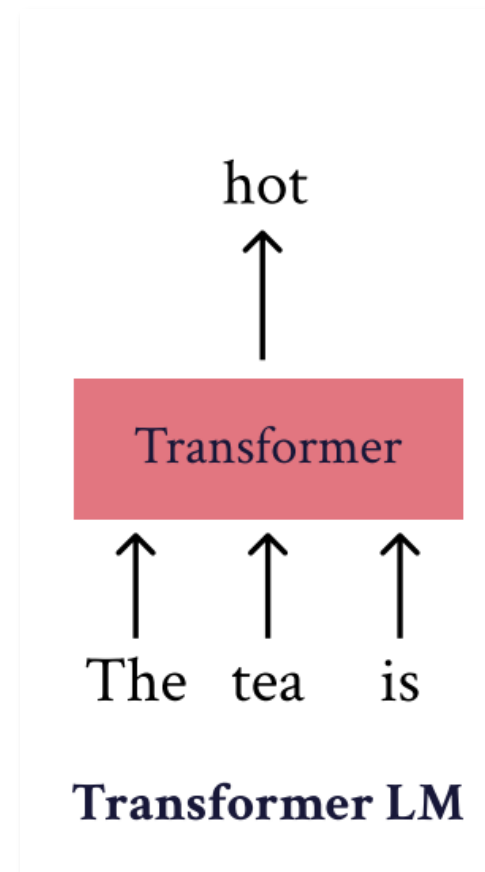


背景: Transformer の操作性・解釈性の低さ

# 背景: Transformer の操作性・解釈性の低さ

## 良い点 😊

- Attention 層 と MLP 層を繰り返すことで複雑な文脈を考慮した表現を獲得できる
  - 言語モデルはこの表現をもとに、入力文に続く単語を予測





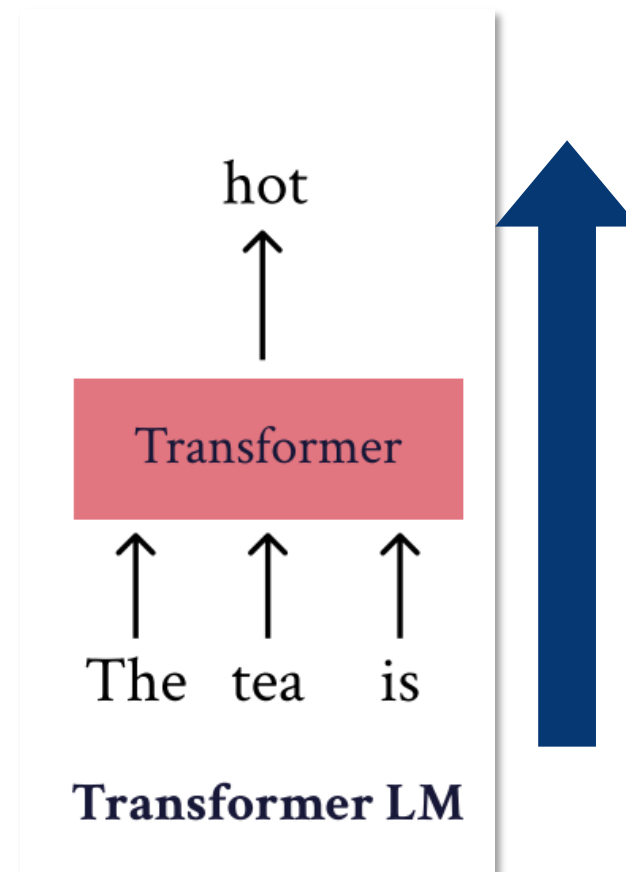
# 背景: Transformer の操作性・解釈性の低さ

## 良い点 😊

- Attention 層 と MLP 層を繰り返すことで複雑な文脈を考慮した表現を獲得できる
  - 言語モデルはこの表現をもとに、入力文に続く単語を予測

## 悪い点 😇

- 入力から出力まで一通の処理なので、モデル内部へ介入を行うと基本的に非線形な影響が発生
  - 操作・解釈が難しい



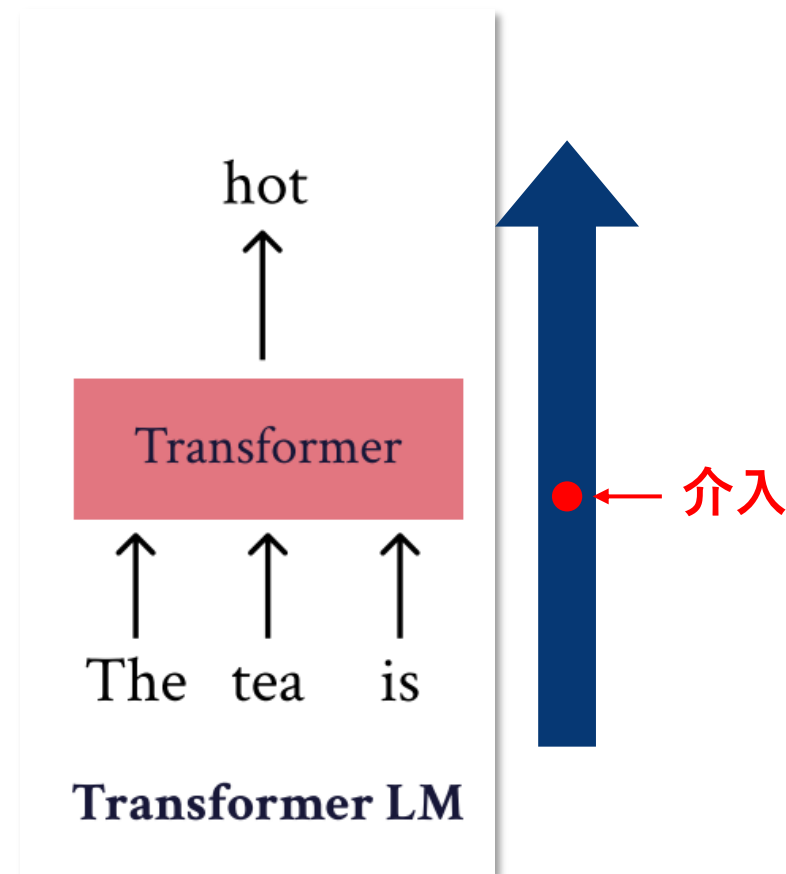
# 背景: Transformer の操作性・解釈性の低さ

## 良い点 😊

- Attention 層 と MLP 層を繰り返すことで複雑な文脈を考慮した表現を獲得できる
  - 言語モデルはこの表現をもとに、入力文に続く単語を予測

## 悪い点 😇

- 入力から出力まで一通の処理なので、モデル内部へ介入を行うと基本的に非線形な影響が発生
  - 操作・解釈が難しい



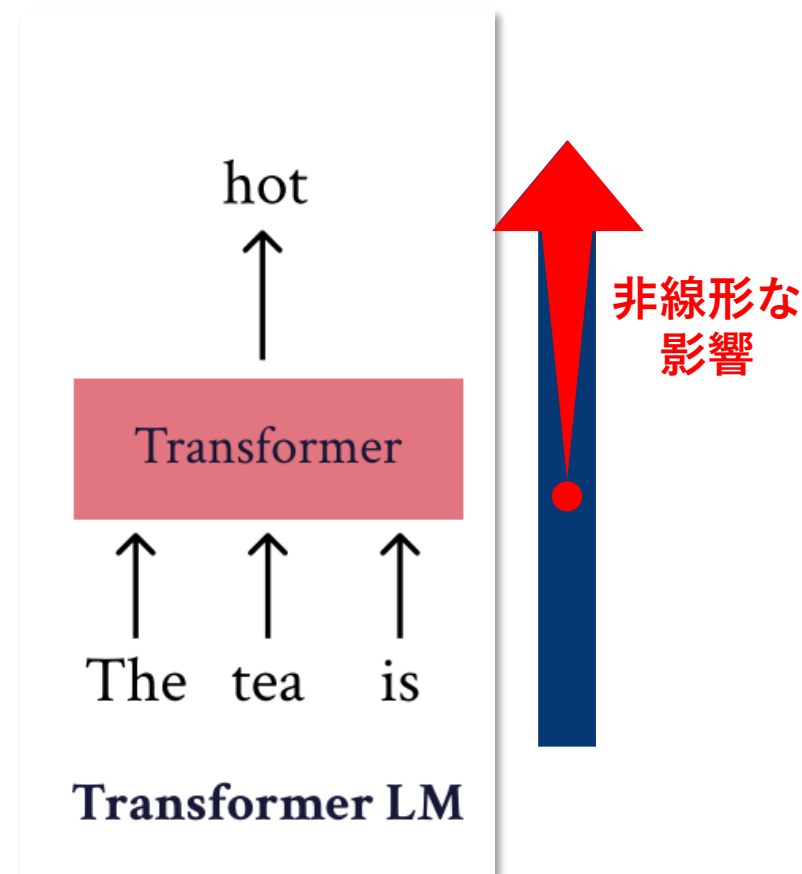
# 背景: Transformer の操作性・解釈性の低さ

## 良い点 😊

- Attention 層 と MLP 層を繰り返すことで複雑な文脈を考慮した表現を獲得できる
  - 言語モデルはこの表現をもとに、入力文に続く単語を予測

## 悪い点 😇

- 入力から出力まで一通の処理なので、モデル内部へ介入を行うと基本的に非線形な影響が発生
  - 操作・解釈が難しい



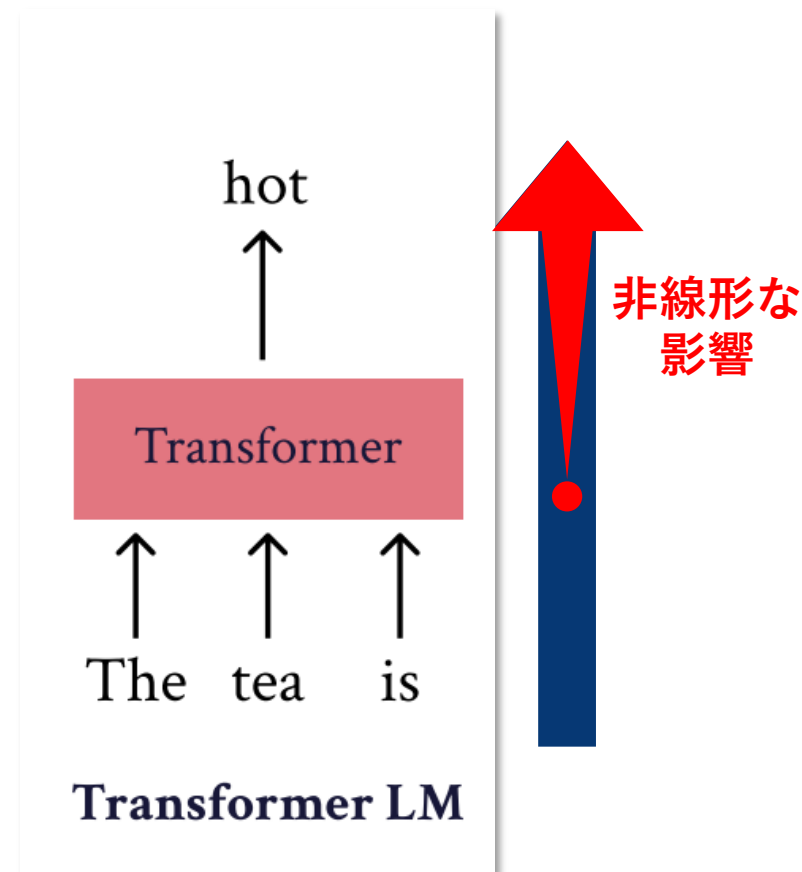
# 背景: Transformer の操作性・解釈性の低さ

## 良い点 😊

- Attention 層 と MLP 層を繰り返すことで複雑な文脈を考慮した表現を獲得できる
  - 言語モデルはこの表現をもとに、入力文に続く単語を予測

## 悪い点 😇

- 入力から出力まで一通の処理なので、モデル内部へ介入を行うと基本的に非線形な影響が発生
  - 操作・解釈が難しい
  - これを頑張って分析する研究も存在 [[Meng+, ICLR2023](#); [Serrano+, ACL2019](#)]



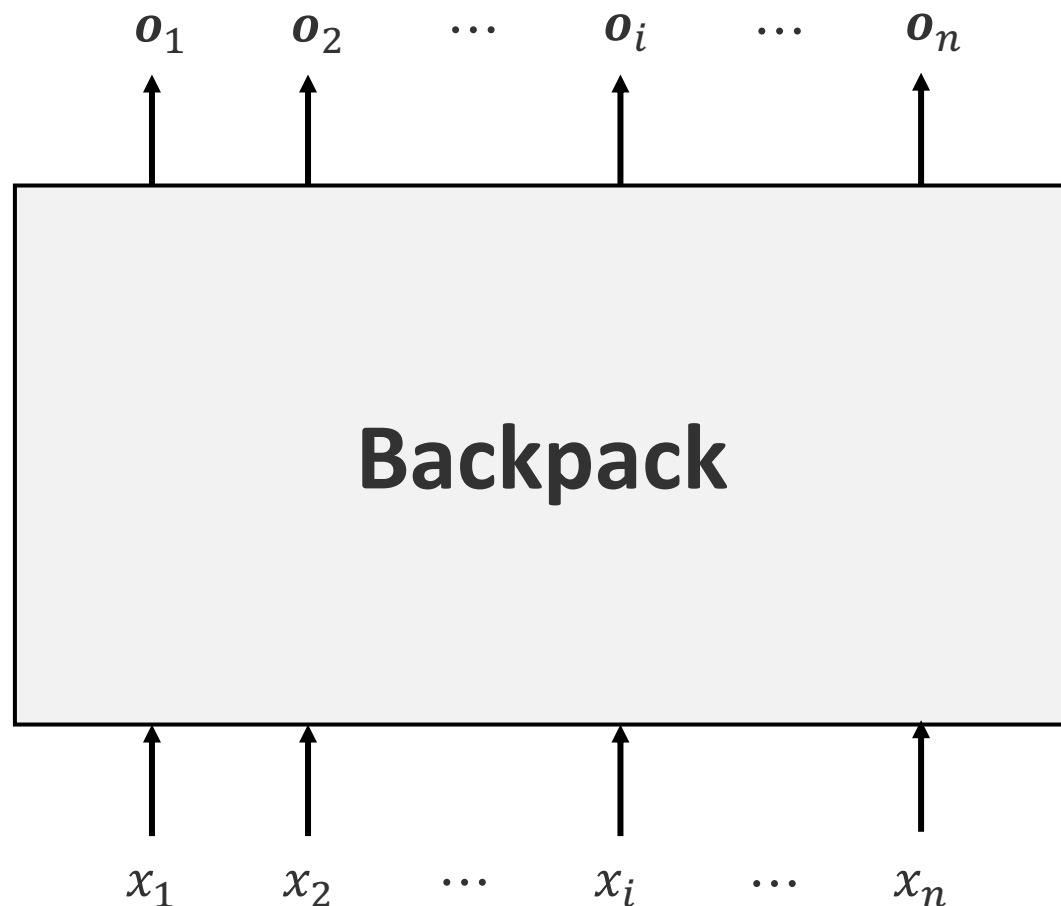
MLP

Attention

# Backpack モデルとは？

# Backpack: 全体像

- Backpack は文脈を考慮した単語埋め込み表現の獲得方法



$$\mathbf{o}_i = \text{Backpack}(x_1 : x_n)$$

$$(1 \leq i \leq n, x_i \in \mathcal{V}, \mathbf{o}_i \in R^d)$$

$\mathcal{V}$  は  
Vocabulary

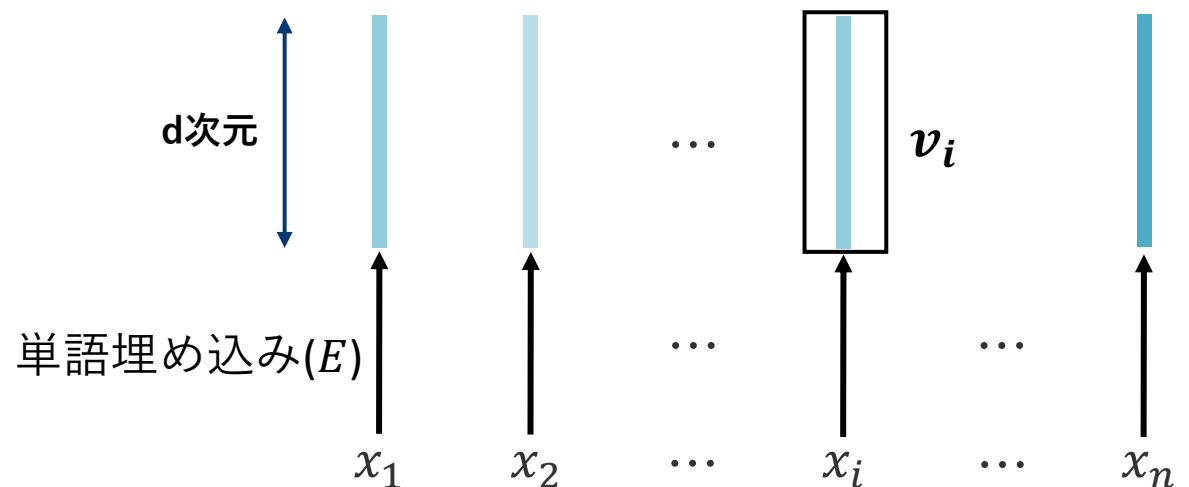
# Backpack: 単語から Sense ベクトルへ

1. 各単語をそれぞれ文脈に依存しない  $k(\geq 1)$  個の Sense ベクトルに変換

$x_1$     $x_2$     $\dots$     $x_i$     $\dots$     $x_n$

# Backpack: 単語から Sense ベクトルへ

1. 各単語をそれぞれ文脈に依存しない  $k(\geq 1)$  個の Sense ベクトルに変換



各単語を埋め込み,

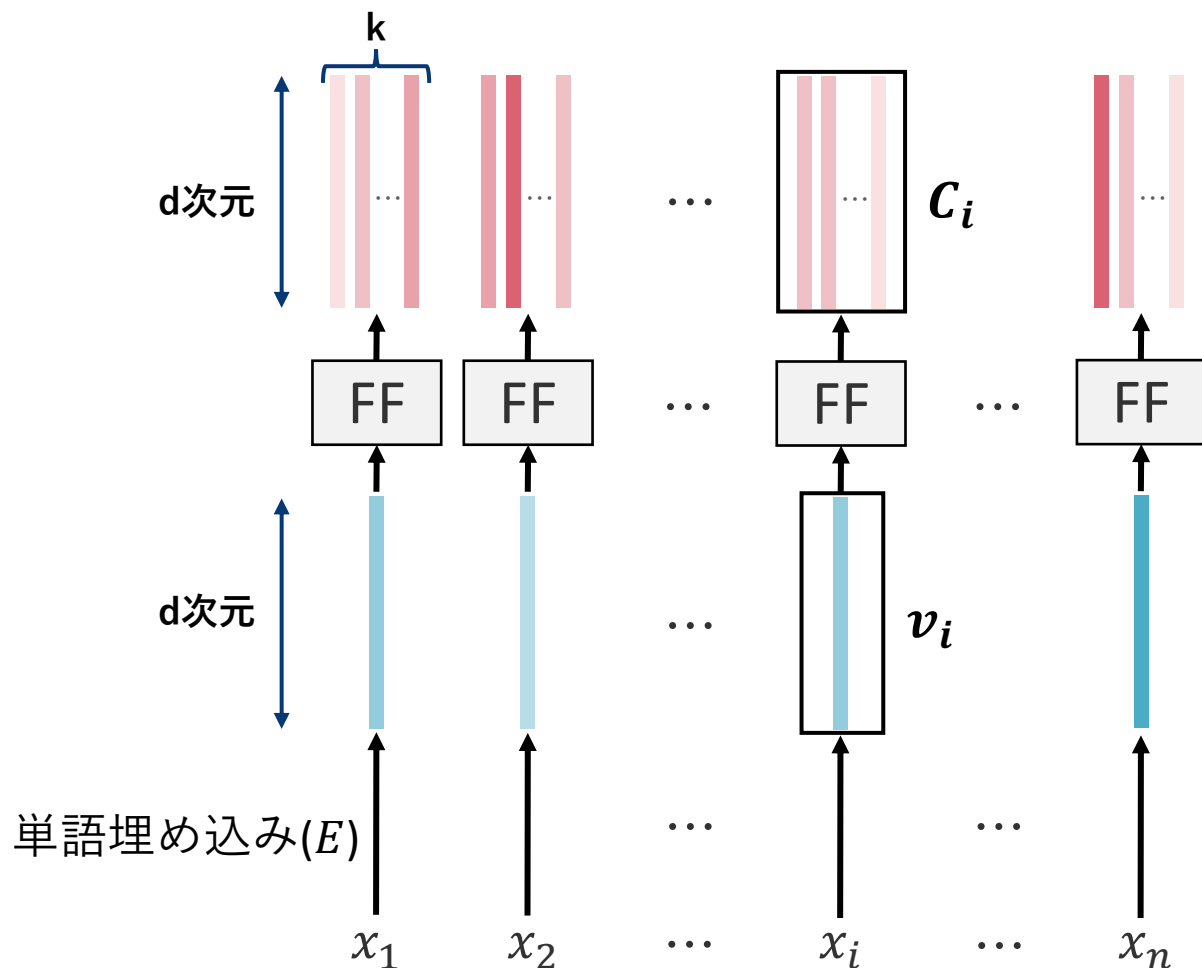
$$\boldsymbol{v}_i = E\boldsymbol{x}_i$$

$$(\boldsymbol{v}_i \in R^d)$$



# Backpack: 単語から Sense ベクトルへ

1. 各単語をそれぞれ文脈に依存しない  $k(\geq 1)$  個の Sense ベクトルに変換



各単語を埋め込み,

$$v_i = E x_i$$

それを Feed-Forward で変換

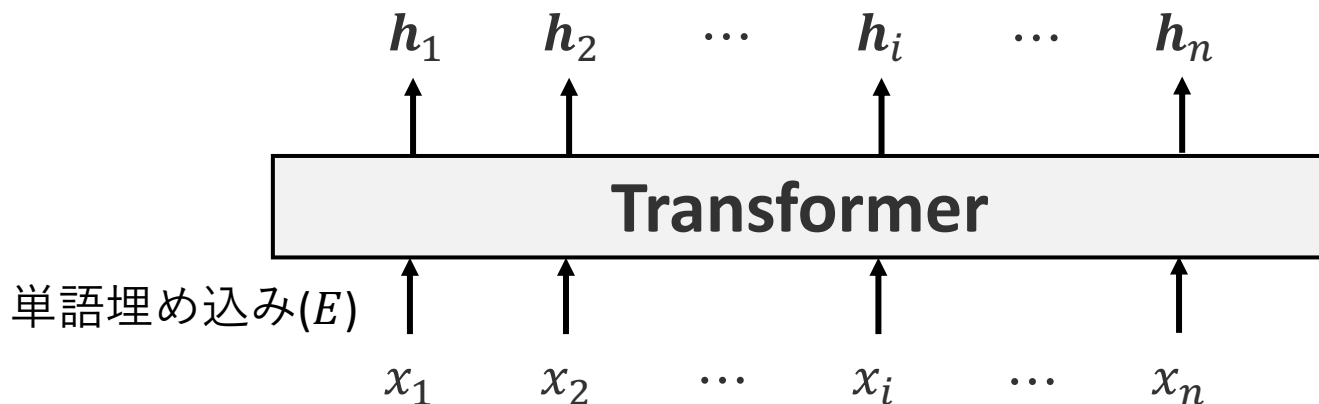
$$C_i = \text{FF}(v_i)$$

$$C = \{C_1, \dots, C_i, \dots, C_n\}$$

$$\begin{pmatrix} v_i \in R^d \\ C_i \in R^{k \times d} \\ C \in R^{n \times k \times d} \end{pmatrix}$$

# Backpack: 単語から重みへ

## 2. Transformer により文脈に依存した重みを計算



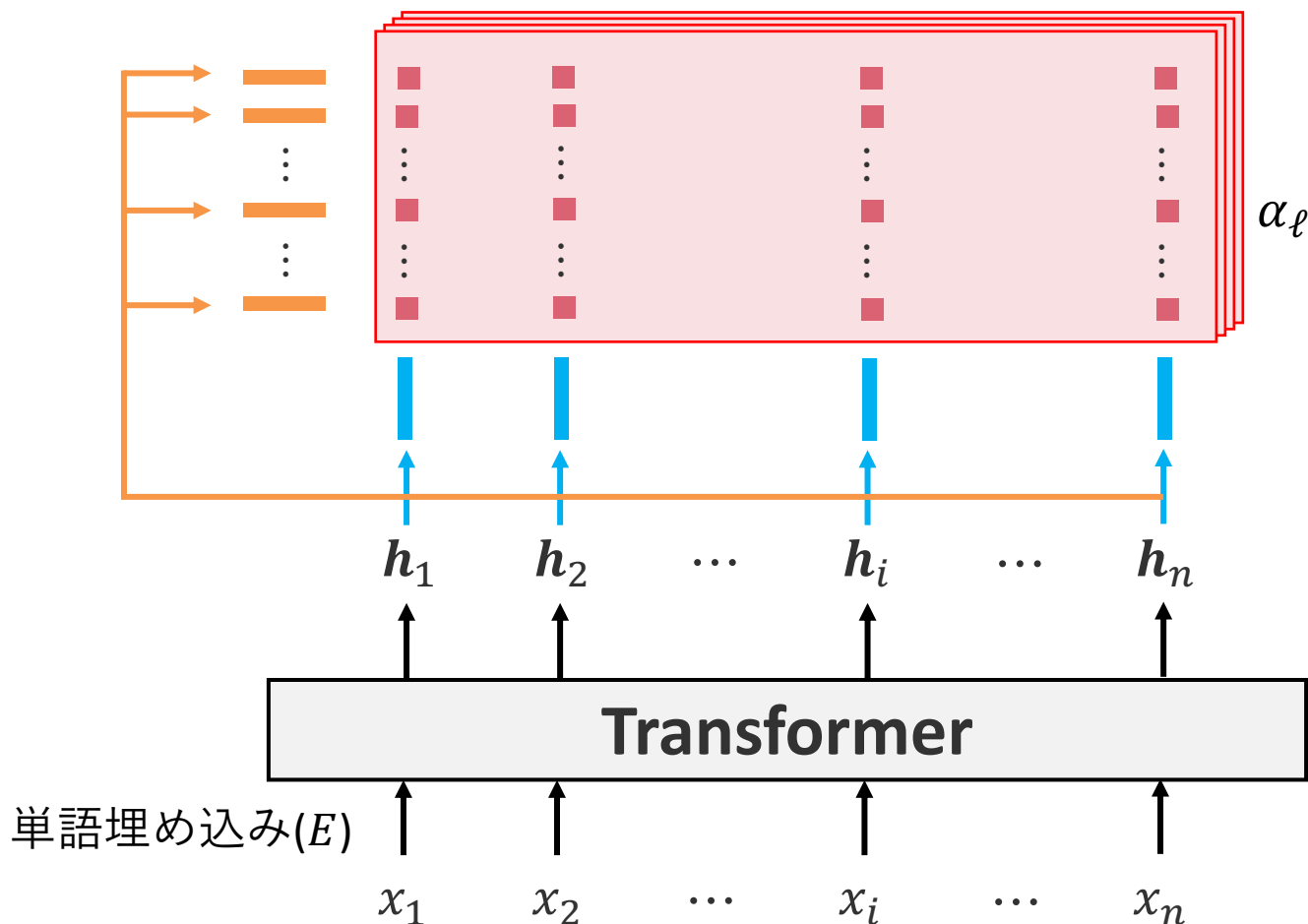
Transformer に文を入力

$$\mathbf{h}_{1:n} = \text{transformer}(E x_{1:n})$$

$$(\mathbf{h}_{1:n} \in R^{n \times d})$$

# Backpack: 単語から重みへ

## 2. Transformer により文脈に依存した重みを計算



Transformer に文を入力

$$\mathbf{h}_{1:n} = \text{transformer}(Ex_{1:n})$$

Head 数  $k$  の Attention map を計算

$$\alpha_\ell = \text{softmax}\left(\mathbf{h}_{1:n} W_k^{(\ell)\top} W_Q^{(\ell)} \mathbf{h}_{1:n}^\top\right)$$
$$\alpha = \{\alpha_1, \dots, \alpha_\ell, \dots, \alpha_k\}$$

$$\left( \begin{array}{l} \mathbf{h}_{1:n} \in R^{n \times d} \\ 1 \leq \ell \leq k \\ W^{(\ell)}, Q^{(\ell)} \in R^{d \times \frac{d}{k}} \\ \alpha_\ell \in R^{n \times n} \\ \alpha \in R^{k \times n \times n} \end{array} \right)$$

# Backpack: 重み付き和を計算

3. Sense ベクトルの重み付き和をとる

- 1 から文脈非依存の Sense ベクトル  $\mathbf{C} \in R^{n \times k \times d}$
- 2 から文脈依存の重み  $\alpha \in R^{k \times n \times n}$

これらから, Sense ベクトルの重み付き和を計算

$$\mathbf{o}_i = \sum_{j=1}^n \sum_{\ell=1}^k \alpha_{\ell i j} \mathbf{C}_{j, \ell}$$

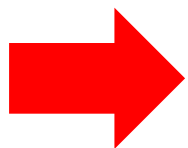
# Backpack: 重み付き和を計算

3. Sense ベクトルの重み付き和をとる

- 1 から文脈非依存の Sense ベクトル  $\mathbf{C} \in R^{n \times k \times d}$
- 2 から文脈依存の重み  $\alpha \in R^{k \times n \times n}$

これらから, Sense ベクトルの重み付き和を計算

$$\mathbf{o}_i = \sum_{j=1}^n \sum_{\ell=1}^k \alpha_{\ell i j} \mathbf{C}_{j, \ell}$$

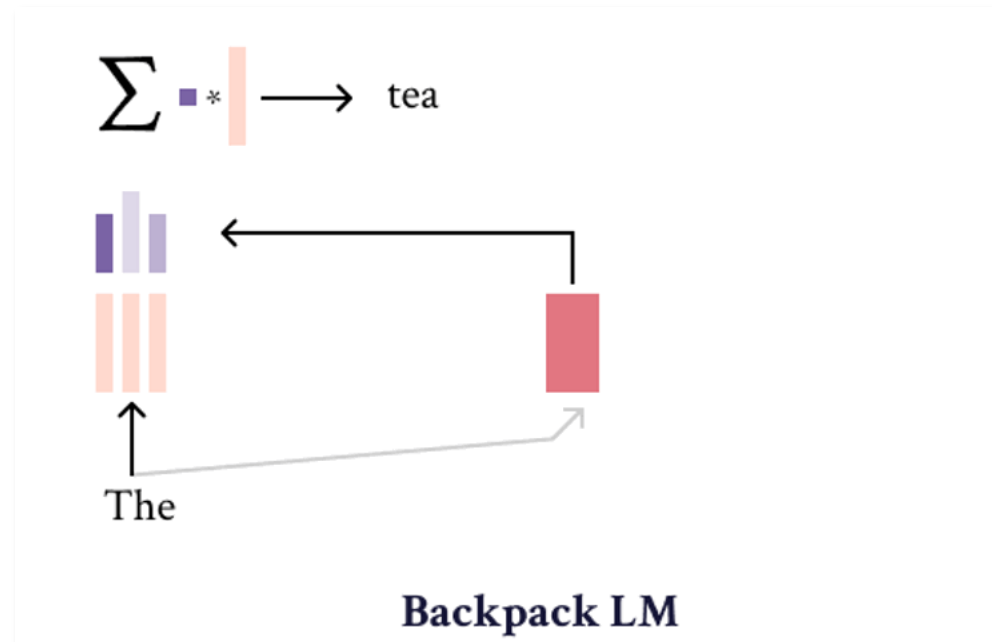


文脈非依存の Sense ベクトルの線形和で  
文脈を考慮した単語埋め込み表現を獲得

# Backpack Language Models

## Backpack Language Models とは?

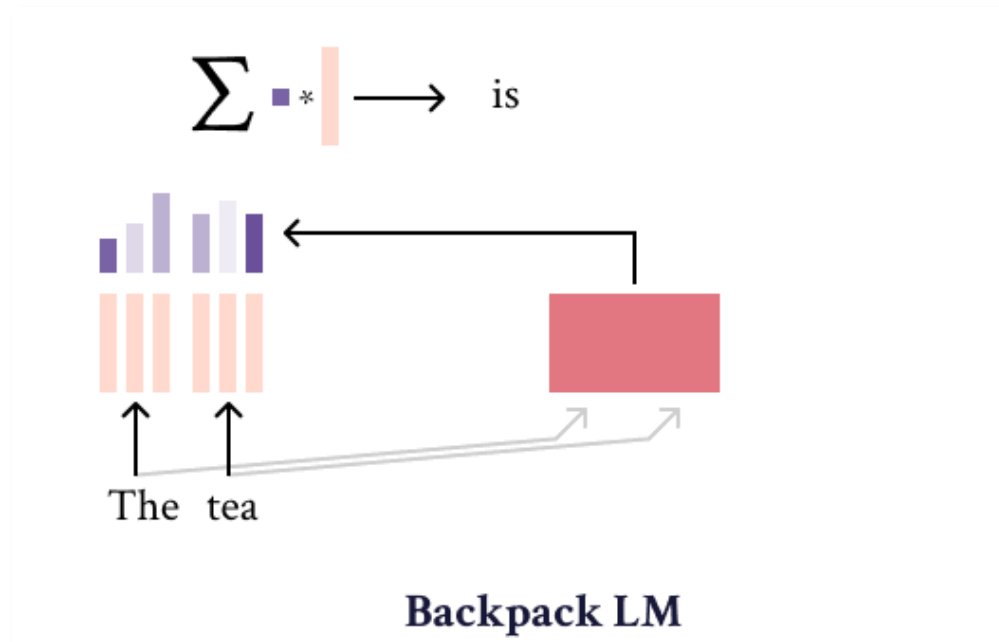
- Backpack を利用した言語モデル
- 重みを計算する Transformer に Causal Mask



# Backpack Language Models

## Backpack Language Models とは?

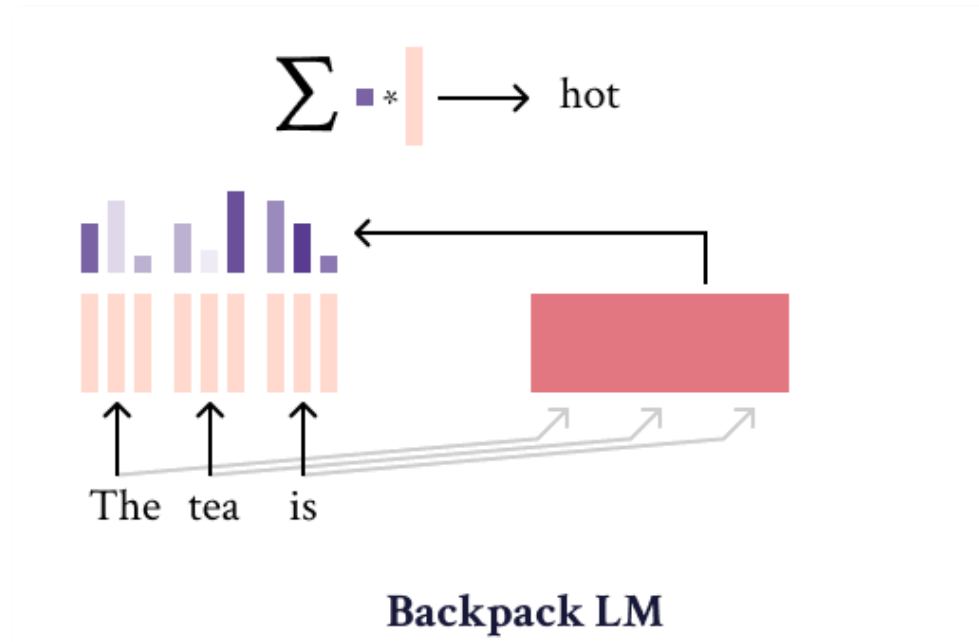
- Backpack を利用した言語モデル
- 重みを計算する Transformer に Causal Mask



# Backpack Language Models

## Backpack Language Models とは?

- Backpack を利用した言語モデル
- 重みを計算する Transformer に Causal Mask

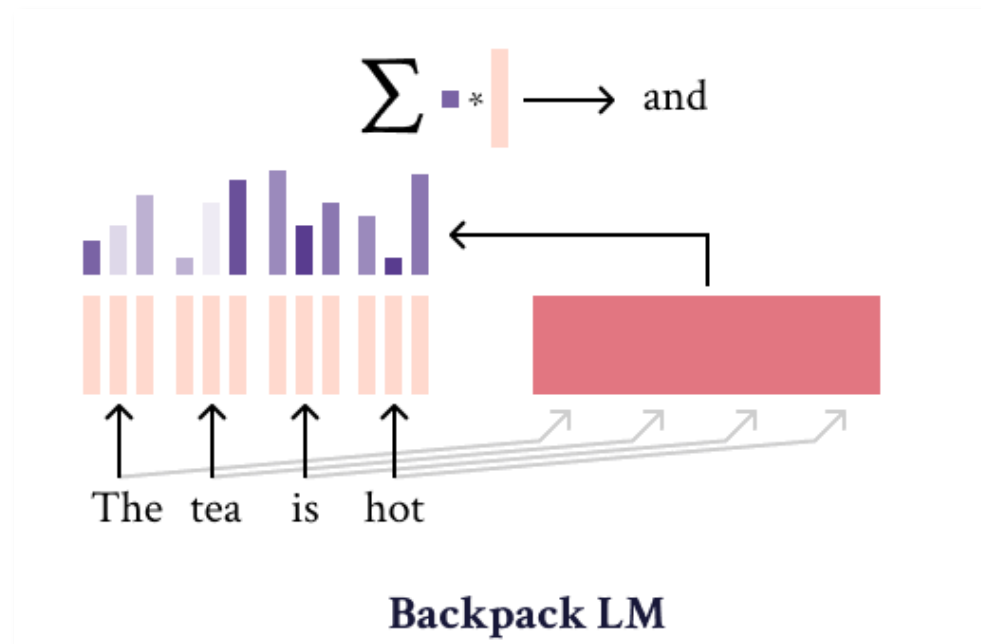




# Backpack Language Models

## Backpack Language Models とは？

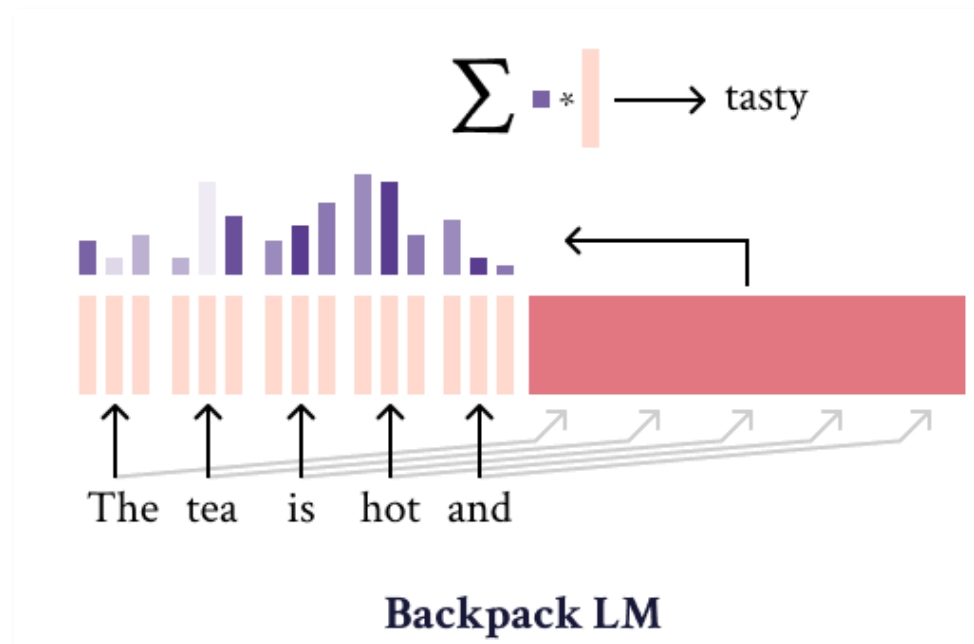
- Backpack を利用した言語モデル
- 重みを計算する Transformer に Causal Mask



# Backpack Language Models

## Backpack Language Models とは?

- Backpack を利用した言語モデル
- 重みを計算する Transformer として Decoder-only のモデルを使用



# 実験: Backpack Language Models

- モデル
  - Backpack-Micro(40M)/Mini(100M)/Small(170M)
  - Transformer-Micro(30M)/Mini(70M)/Small(124M)
- 学習データ
  - OpenWebtext コーパス [[Gokaslan+, 2019](#)]
- 評価
  - Perplexity 等の基本的な評価指標を使用

# 実験: Backpack Language Models

## 結果

- どの評価を重視するかにもよるが、Transformer と同等程度の性能を発揮
  - そもそもパラメータ数が同じじゃないのはフェアじゃない感

Model	OpenWebText PPL ↓	LAMBADA PPL ↓	LAMBADA ACC ↑	Wikitext PPL ↓	BLiMP ↑
Backpack-Micro	<b>31.5</b>	<b>110</b>	<b>24.7</b>	<b>71.5</b>	75.6
Transformer-Micro	34.4	201	21.3	79.5	<b>77.8</b>
Backpack-Mini	<b>23.5</b>	<b>42.7</b>	<b>31.6</b>	<b>49.0</b>	76.2
Transformer-Mini	24.5	58.8	29.7	52.8	<b>80.4</b>
Backpack-Small	<b>20.1</b>	<b>26.5</b>	<b>37.5</b>	<b>40.9</b>	76.3
Transformer-Small	<b>20.2</b>	32.7	34.9	42.2	<b>81.9</b>

# Backpack LM の良い点

# 良い点: 操作性・解釈性が向上

## 解釈性の例

- "science" という単語を  $k = 16$  で Sense ベクトル ( $\mathbf{C}_{science} \in R^{16 \times d}$ ) に変換し, それを  $E^T \mathbf{C}_{science}$  と投影することで単語の意味 (確率分布) を入手可能
  - Sense 7/8 には科学的手法 (replicationやexperiments など)
  - Sense 3 は science の後ろに続きやすい単語 (science denial で科学不信)

A few senses of the word <i>science</i>				
Sense 3	Sense 7	Sense 9	Sense 10	Sense 8
fiction	replication	religion	settled	clones
fictional	citation	rology	sett	experiments
Fiction	Hubble	hydra	settle	mage
literacy	reprodu	religions	unsett	experiment
denial	Discovery	nec	Sett	rats

# 良い点: 操作性・解釈性が向上

## 操作性の例

- “When the nurse came into the room,” に Backpack LM を適用
- “nurse” という単語の影響で “She” が来る可能性が高い

### Backpack Sense Visualization

Language Modeling

Individual Word Sense Look Up

Input Sentence

When the nurse came into the room,

Predict next word

Reset weights

Top-k predicted next word

1	2	3	4	5	6	7	8	9	10
she	the	he	they	I	there	a	her	we	it
0.277	0.141	0.127	0.061	0.039	0.025	0.02	0.019	0.018	0.013

# 良い点: 操作性・解釈性が向上

## 操作性の例

- “nurse” の Sense ベクトルの意味を見てみると, Sense 10 に “She” という意味があった
- この重みを人為的に 0 にしてもう一度次の単語を予測すると. . .





# 良い点: 操作性・解釈性が向上

## 操作性の例

- “She” の確率が低下 ( $0.277 \rightarrow 0.228$ ) し, “He” の確率が増加 ( $0.127 \rightarrow 0.167$ )

### Backpack Sense Visualization

Language Modeling

Individual Word Sense Look Up

Input Sentence

When the nurse came into the room,

Predict next word

Reset weights

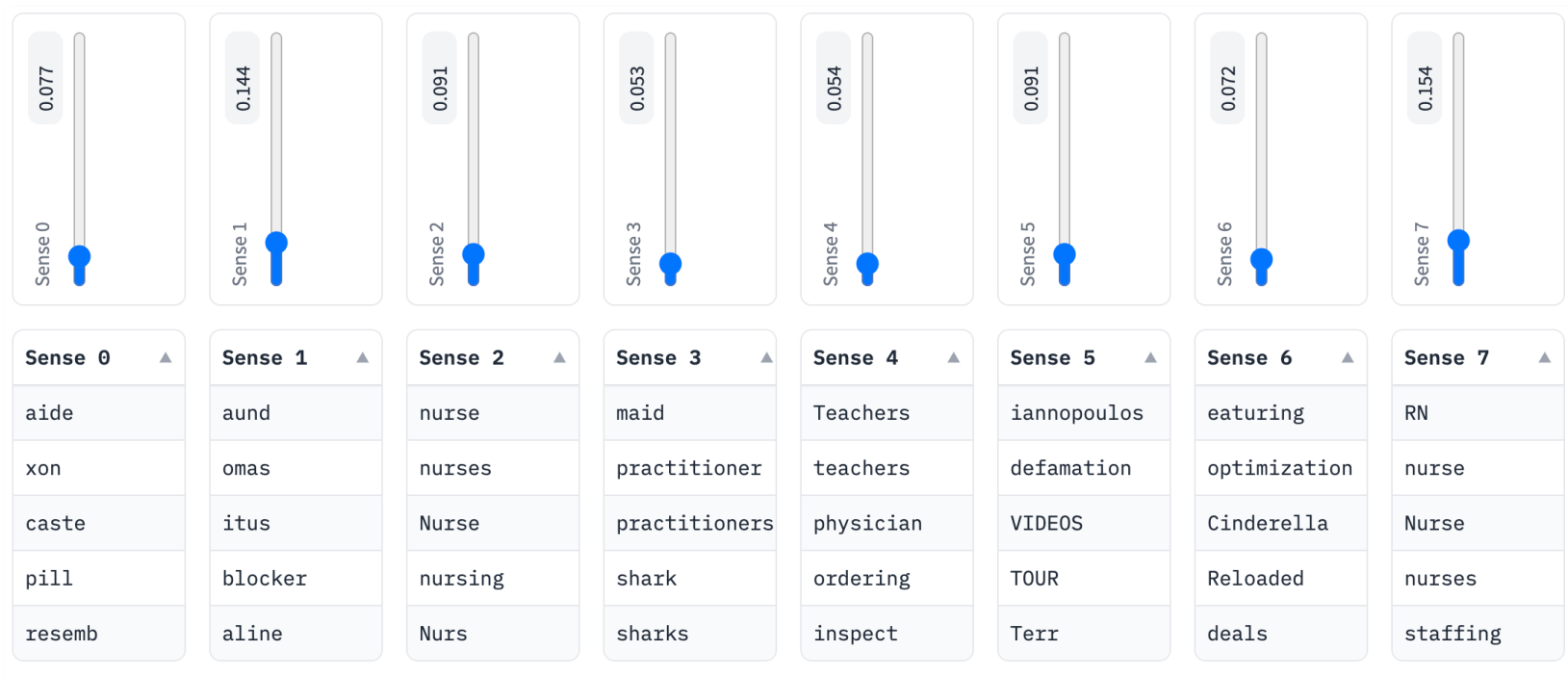
Top-k predicted next word

1	2	3	4	5	6	7	8	9	10
she	he	the	they	I	there	a	we	her	it
0.228	0.167	0.142	0.061	0.04	0.025	0.021	0.018	0.017	0.014

# Backpack LM の注意点

# 注意点1: 完全に解釈可能になったわけではない

- 全ての Sense ベクトルが綺麗に言語化できるわけではない
  - Sense ベクトルが綺麗に言語化されるような学習は行っていない
  - “nurse” の Sense ベクトル0~7の意味をみるとあまり一般的でない意味も多い



## 注意点2: パラメータ数が増加している

- Sense ベクトルを生成するネットワーク分のパラメータが増加している
  - 例: Transformer-small(124M)→Backpack-small(170M)
- Trainable なパラメータが増えたことにより, ロスの収束に時間がかかる
- 性能が同じだと仮定すると, 解釈性と計算コストにトレードオフの関係

## 注意点3: 表現力が下がる可能性

### そもそも

- 文脈に依存しない Sense ベクトルの線形和を使用し、重みはかならず正となる

### なので

- 否定的な表現が苦手な可能性
  - 「Sense ベクトルの中にネガティブな意味も含まれていて、その重みが大きくなるはず」と論文では述べているが、ポジティブな Sense ベクトルの方が多いはずであり、それらの重みが0になることはない
- 複数の単語が組み合わさって初めて生まれる意味に弱い可能性
  - これも Sense ベクトルの中にそういう意味を持つものがあり、その重みが大きくなるのかもしれないが、出現頻度が低い組み合わせだと、 $k$  を大きくしないと対応できない感

# まとめ

# まとめ

## 手法

- 各単語をそれぞれ文脈に依存しない複数の Sense ベクトルに変換
- Transformer により文脈に依存した重みを計算
- Sense ベクトルの重み付き和をとる

## 特徴

- Transformer と同等の性能
- 文脈に依存しない Sense ベクトルを組み込むことで操作性・解釈性が向上

