Sequence Models

Jordan Boyd-Graber

University of Maryland

RNNs

Slides adapted from Jon Hewitt

Log-linear Language Models

Equation:

$$P(w \mid c) = \frac{\exp(\sum_i \lambda_i f_i(w, c))}{\sum_{w'} \exp(\sum_i \lambda_i f_i(w', c))} \tag{1}$$

where:

- $P(w \mid c)$ is the probability of word w given context c,
- $f_i(w, c)$ is a feature function representing some property of the word w and context c,
- λ_i is the (learned) weight of the corresponding feature.

Examples of Features:

- Presence of specific words in the context: $f_1(w, c) = 1$ if a particular word appears in c.
- Word length: $f_2(w, c) = \text{len}(w)$.
- Part of Speech: $f_3(w, c) = 1$ if w is a noun.

Bigram Language Model as a Log-Linear Model

Objective: Predict next word $w=$ bageling given previous word $c=$ go.

Log-Linear Model Equation:

$$P(w \mid c) = \frac{\exp\left(\sum_i \lambda_i f_i(w,c)\right)}{\sum_{w'} \exp\left(\sum_i \lambda_i f_i(w',c)\right)}$$

Example Features:

- $f_1(w,c) = \mathbb{1}(c = \text{go}, w = \text{bageling})$
- $f_2(c) = \mathbb{1}(c = \text{go})$
- $f_3(w,c) = \mathbb{1}(c = \text{go}, w = \text{to})$

Weights:

- $\lambda_1 = 2.5$
- $\lambda_2 = -10.0$
- $\lambda_3 = 0.5$

Prediction:

$$P(\text{bageling} \mid \text{go}) = \frac{\exp(2.5 - 10)}{\exp(-7.5) + \exp(3.0) + \exp(1.5)} \tag{2}$$

Normalizing over all possible next words.

The Backpack Language Model

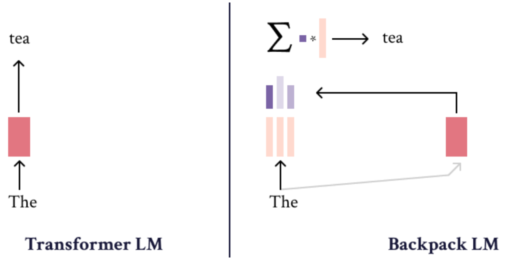
Backpack Language Models

John Hewitt **John Thickstun** **Christopher D. Manning** **Percy Liang**
Department of Computer Science, Stanford University
{johnhew,jthickstun,manning,pliang}@cs.stanford.edu

The Backpack Language Model

Backpack Language Models

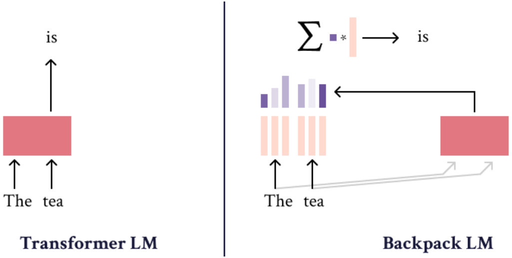
John Hewitt John Thickstun Christopher D. Manning Percy Liang
Department of Computer Science, Stanford University
{johnhew,jthickstun,manning,pliang}@cs.stanford.edu



The Backpack Language Model

Backpack Language Models

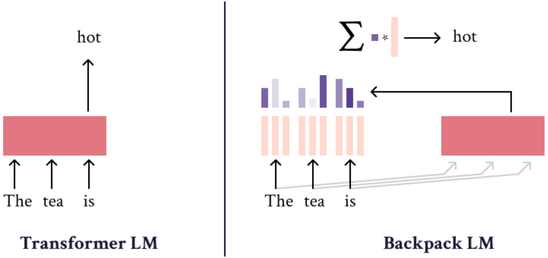
John Hewitt **John Thickstun** **Christopher D. Manning** **Percy Liang**
Department of Computer Science, Stanford University
{johnhew,jthickstun,manning,pliang}@cs.stanford.edu



The Backpack Language Model

Backpack Language Models

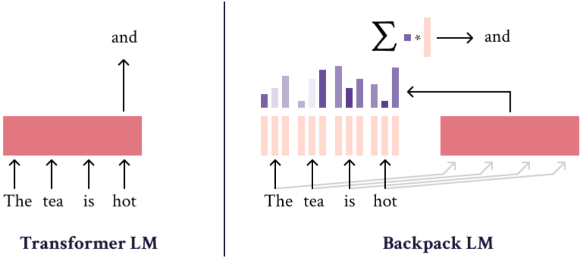
John Hewitt **John Thickstun** **Christopher D. Manning** **Percy Liang**
Department of Computer Science, Stanford University
{johnhew,jthickstun,manning,pliang}@cs.stanford.edu



The Backpack Language Model

Backpack Language Models

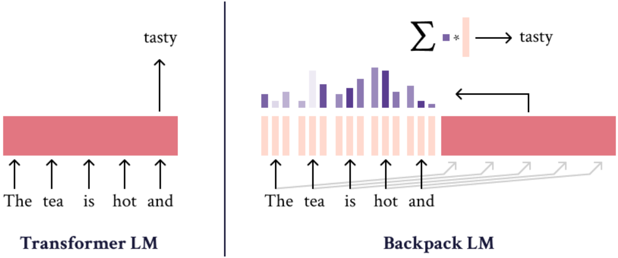
John Hewitt **John Thickstun** **Christopher D. Manning** **Percy Liang**
Department of Computer Science, Stanford University
{johnhew,jthickstun,manning,pliang}@cs.stanford.edu



The Backpack Language Model

Backpack Language Models

John Hewitt **John Thickstun** **Christopher D. Manning** **Percy Liang**
Department of Computer Science, Stanford University
{johnhew,jthickstun,manning,pliang}@cs.stanford.edu



The Backpack Language Model

Backpack Language Models

John Hewitt **John Thickstun** **Christopher D. Manning** **Percy Liang**
Department of Computer Science, Stanford University
{johnhew,jthickstun,manning,pliang}@cs.stanford.edu

But we're not talking about transformers today!

- However, general structure of Backpack Models is fairly simple
- Next time, we can reuse this framework to explain the transformer
- Shows the effect of non-linear context

Loglinear Language Models without Features

Equation 1: Given input $x_1 \ldots x_n$, sense vectors for the sequence:

$$C(x) = \{C(x)_1, \ldots, C(x)_k\}$$

Equation 2: Weighted sum of sense vectors:

$$o_i = \sum_{j=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell,i,j} C(x_j)_\ell$$

Equation 4: Probability of the next word given the sequence:

$$p(y \mid o_{1:n}) = \text{softmax}(E(o_{1:n}))$$

- $C(x)_1$ are sense vectors: e.g., a different vector for "dog" that barks and "dog" you serve on a bun.
- α^{ℓ}_{ij} contextualization weights: which sense is relevant.
- $o_{1:n}$ is new representation for tokens
- That becomes input to predict next word through loglinear E

Loglinear Language Models without Features

Equation 1: Given input $x_1 \dots x_n$, sense vectors for the sequence:

$$C(x) = \{C(x)_1, \dots, C(x)_k\}$$

Equation 2: Weighted sum of sense vectors:

$$o_i = \sum_{j=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell,i,j} C(x_j)_\ell$$

Equation 4: Probability of the next word given the sequence:

$$p(y \mid o_{1:n}) = \text{softmax}(E(o_{1:n}))$$

- $C(x)_1$ are sense vectors: e.g., a different vector for “dog” that barks and “dog” you serve on a bun.
- $\alpha^{\ell}_{i\,j}$ contextualization weights: which sense is relevant.
- $o_{1:n}$ is new representation for tokens
- That becomes input to predict next word through loglinear E

Loglinear Language Models without Features

Equation 1: Given input $x_1 \dots x_n$, sense vectors for the sequence:

$$C(x) = \{C(x)_1, \dots, C(x)_k\}$$

Equation 2: Weighted sum of sense vectors:

$$o_i = \sum_{j=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell,i,j} C(x_j)_\ell$$

Equation 4: Probability of the next word given the sequence:

$$p(y \mid o_{1:n}) = \text{softmax}(E(o_{1:n}))$$

- $C(x)_1$ are sense vectors: e.g., a different vector for “dog” that barks and “dog” you serve on a bun.
- $\alpha^{\ell}_{i\,j}$ contextualization weights: which sense is relevant.
- $o_{1:n}$ is new representation for tokens
- That becomes input to predict next word through loglinear E

Loglinear Language Models without Features

Equation 1: Given input $x_1 \dots x_n$, sense vectors for the sequence:

$$C(x) = \{C(x)_1, \dots, C(x)_k\}$$

Equation 2: Weighted sum of sense vectors:

$$o_i = \sum_{j=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell,i,j} C(x_j)_\ell$$

Equation 4: Probability of the next word given the sequence:

$$p(y \mid o_{1:n}) = \text{softmax}(E(o_{1:n}))$$

- $C(x)_1$ are sense vectors: e.g., a different vector for "dog" that barks and "dog" you serve on a bun.
- α^{ℓ}_{ij} contextualization weights: which sense is relevant.
- $o_{1:n}$ is new representation for tokens
- That becomes input to predict next word through loglinear E

Loglinear Language Models without Features

Equation 1: Given input $x_1 \dots x_n$, sense vectors for the sequence:

$$C(x) = \{C(x)_1, \dots, C(x)_k\}$$

Equation 2: Weighted sum of sense vectors:

$$o_i = \sum_{j=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell,i,j} C(x_j)_\ell$$

Equation 4: Probability of the next word given the sequence:

$$p(y \mid o_{1:n}) = \text{softmax}(E(o_{1:n}))$$

- $C(x)_1$ are sense vectors: e.g., a different vector for “dog” that barks and “dog” you serve on a bun.
- α^{ℓ}_{ij} contextualization weights: which sense is relevant.
- $o_{1:n}$ is new representation for tokens
- That becomes input to predict next word through loglinear E

Compare and Contrast

Traditional Feature-based Loglinear Models

- Dimension of features is gigantic (but sparse)
- Fitting those weights is hard
- Very interpretable

Backpack Models

- Less interpretable than features
- More interpretable than RNN
- Comparable to GPT

Compare and Contrast

Traditional Feature-based Loglinear Models

- Dimension of features is gigantic (but sparse)
- Fitting those weights is hard
- Very interpretable

Backpack Models

- Less interpretable than features
- More interpretable than RNN
- Comparable to GPT2

Example: Sense and Sentimentabilities

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix}$$

Example: Sense and Sentimentabilities

$$\mathbf{x} = \begin{pmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{pmatrix} \longleftarrow \begin{array}{c} \text{Positive Sentiment} \\ \text{Negative Sentiment} \\ \text{Skateboarding Context} \\ \text{Health Context} \end{array}$$

Backpack Language Model Example: "That trick was sick"

Sentence: "That trick was sick"

that	**trick**	**was**	**sick**
$C(\text{that})_0$	$C(\text{trick})_0$	$C(\text{was})_0$	$C(\text{sick})_0$
$\begin{pmatrix}0 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 0 & 1 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}1 & 0 & 1 & 0\end{pmatrix}$
$C(\text{that})_1$	$C(\text{trick})_1$	$C(\text{was})_1$	$C(\text{sick})_1$
$\begin{pmatrix}0 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 1 & 0 & 1\end{pmatrix}$

Explanation:

- Each word has two sense vectors: $C(x_i)_1$ and $C(x_i)_2$.
- "that" and "was" have zero vectors for both senses.
- "trick" has a non-zero vector for $C(\text{trick})_1$ at the third position (evokes scateboarding), while the second vector is zero.
- "sick" has two non-zero vectors: $C(\text{sick})_1$ and $C(\text{sick})_2$: positive for skateboard, negative for health.

Backpack Language Model Example: "That trick was sick"

Sentence: "That trick was sick"

that	**trick**	**was**	**sick**
$C(\text{that})_0$	$C(\text{trick})_0$	$C(\text{was})_0$	$C(\text{sick})_0$
$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}$
$C(\text{that})_1$	$C(\text{trick})_1$	$C(\text{was})_1$	$C(\text{sick})_1$
$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}$

Explanation:

- Each word has two sense vectors: $C(x_i)_1$ and $C(x_i)_2$.
- "that" and "was" have zero vectors for both senses.
- "trick" has a non-zero vector for $C(\text{trick})_1$ at the third position (evokes scateboarding), while the second vector is zero.
- "sick" has two non-zero vectors: $C(\text{sick})_1$ and $C(\text{sick})_2$: positive for skateboard, negative for health.

Backpack Language Model Example: "That trick was sick"

Sentence: "That trick was sick"

that	**trick**	**was**	**sick**
$C(\text{that})_0$	$C(\text{trick})_0$	$C(\text{was})_0$	$C(\text{sick})_0$
$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}$
$C(\text{that})_1$	$C(\text{trick})_1$	$C(\text{was})_1$	$C(\text{sick})_1$
$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}$

Explanation:

- Each word has two sense vectors: $C(x_i)_1$ and $C(x_i)_2$.
- "that" and "was" have zero vectors for both senses.
- "trick" has a non-zero vector for $C(\text{trick})_1$ at the third position (evokes scateboarding), while the second vector is zero.
- "sick" has two non-zero vectors: $C(\text{sick})_1$ and $C(\text{sick})_2$: positive for skateboard, negative for health.

Backpack Language Model Example: "That trick was sick"

Sentence: "That trick was sick"

that	**trick**	**was**	**sick**
$C(\text{that})_0$	$C(\text{trick})_0$	$C(\text{was})_0$	$C(\text{sick})_0$
$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 1 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 1 & 0 & 1 & 0 \end{pmatrix}$
$C(\text{that})_1$	$C(\text{trick})_1$	$C(\text{was})_1$	$C(\text{sick})_1$
$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 0 & 0 & 0 \end{pmatrix}$	$\begin{pmatrix} 0 & 1 & 0 & 1 \end{pmatrix}$

Explanation:

- Each word has two sense vectors: $C(x_i)_1$ and $C(x_i)_2$.
- "that" and "was" have zero vectors for both senses.
- "trick" has a non-zero vector for $C(\text{trick})_1$ at the third position (evokes scateboarding), while the second vector is zero.
- "sick" has two non-zero vectors: $C(\text{sick})_1$ and $C(\text{sick})_2$: positive for skateboard, negative for health.

Backpack Language Model Example: "That trick was sick"

Sentence: "That trick was sick"

that	**trick**	**was**	**sick**
$C(\text{that})_0$	$C(\text{trick})_0$	$C(\text{was})_0$	$C(\text{sick})_0$
$\begin{pmatrix}0 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 0 & 1 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}1 & 0 & 1 & 0\end{pmatrix}$
$C(\text{that})_1$	$C(\text{trick})_1$	$C(\text{was})_1$	$C(\text{sick})_1$
$\begin{pmatrix}0 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 0 & 0 & 0\end{pmatrix}$	$\begin{pmatrix}0 & 1 & 0 & 1\end{pmatrix}$

Explanation:

- Each word has two sense vectors: $C(x_i)_1$ and $C(x_i)_2$.
- "that" and "was" have zero vectors for both senses.
- "trick" has a non-zero vector for $C(\text{trick})_1$ at the third position (evokes scateboarding), while the second vector is zero.
- "sick" has two non-zero vectors: $C(\text{sick})_1$ and $C(\text{sick})_2$: positive for skateboard, negative for health.

Selecting the sense

Definition of $\alpha_{\ell,i,j}$:

$$\alpha_{\ell,i,j} = \begin{cases} 0 & \text{if } j \neq i \\ \sum_{r \neq i} \frac{C(\mathbf{x}_r)_{\ell+2}}{C(\mathbf{x}_r)_2 + C(\mathbf{x}_r)_3} & \text{if } j = i \end{cases} \tag{3}$$

Selecting the sense

Definition of $\alpha_{\ell,i,j}$:

$$\alpha_{\ell,i,j} = \begin{cases} 0 & \text{if } j \neq i \\ \sum_{r \neq i} \frac{C(\mathbf{x}_r)_{\ell+2}}{C(\mathbf{x}_r)_2 + C(\mathbf{x}_r)_3} & \text{if } j = i \end{cases} \tag{3}$$

Which sense does "sick" take?

$$\mathbf{o}_3 = \sum_{j=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell,i,j} C(\mathbf{x}_j)_\ell \tag{4}$$

Given the feature vectors:

$$C(\text{trick})_0 = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad C(\text{sick})_0 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix}, \quad C(\text{sick})_1 = \begin{pmatrix} 0 \\ 1 \\ 0 \\ 1 \end{pmatrix}$$

Selecting the sense

Definition of $\alpha_{\ell,i,j}$:

$$\alpha_{\ell,i,j} = \begin{cases} 0 & \text{if } j \neq i \\ \sum_{r \neq i} \frac{C(\mathbf{x}_r)_{\ell+2}}{C(\mathbf{x}_r)_2 + C(\mathbf{x}_r)_3} & \text{if } j = i \end{cases} \tag{3}$$

Which sense does "sick" take?

$$\mathbf{o}_3 = \sum_{j=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell,i,j} C(\mathbf{x}_j)_\ell \tag{4}$$

$$\mathbf{o}_3 = \frac{C(\text{trick})_0[2]}{C(\text{trick})_0[2] + \text{trick})_0[3]} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \frac{C(\text{trick})_0[3]}{C(\text{trick})_0[2] + C(\text{trick})_0[3]} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{5}$$

$$\tag{6}$$

Selecting the sense

Definition of $\alpha_{\ell,i,j}$:

$$\alpha_{\ell,i,j} = \begin{cases} 0 & \text{if } j \neq i \\ \sum_{r \neq i} \frac{C(\mathbf{x}_r)_{\ell+2}}{C(\mathbf{x}_r)_2 + C(\mathbf{x}_r)_3} & \text{if } j = i \end{cases} \tag{3}$$

Which sense does "sick" take?

$$\mathbf{o}_3 = \sum_{j=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell,i,j} C(\mathbf{x}_j)_\ell \tag{4}$$

$$\mathbf{o}_3 = \frac{1}{1+0} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} + \frac{0}{1+0} \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{5}$$

$$\tag{6}$$

Selecting the sense

Definition of $\alpha_{\ell,i,j}$:

$$\alpha_{\ell,i,j} = \begin{cases} 0 & \text{if } j \neq i \\ \sum_{r \neq i} \frac{C(\mathbf{x}_r)_{\ell+2}}{C(\mathbf{x}_r)_2 + C(\mathbf{x}_r)_3} & \text{if } j = i \end{cases} \tag{3}$$

Which sense does "sick" take?

$$\mathbf{o}_3 = \sum_{j=1}^{n} \sum_{\ell=1}^{k} \alpha_{\ell,i,j} C(\mathbf{x}_j)_\ell \tag{4}$$

$$\mathbf{o}_3 = \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{5}$$

(6)

Wrap-up, Next time

- Today: How context can shape the internal state of models
  - RNNs: linear evolution
  - Backpack models: selecting representations
- Next:
  - Attention: Scanning over entire sentence
  - Multiple representations: Transformer heads

Wrap-up, Next time

- Today: How context can shape the internal state of models
  - RNNs: linear evolution
  - Backpack models: selecting representations
- Next:
  - Attention: Scanning over entire sentence
  - Multiple representations: Transformer heads
  - How attention can produce the α patterns we asserted