Seq2Seq in Action: Column Segmentation

Part 1

Bill Watson

S&P Global

January 23, 2020

Types of Recurrent Models

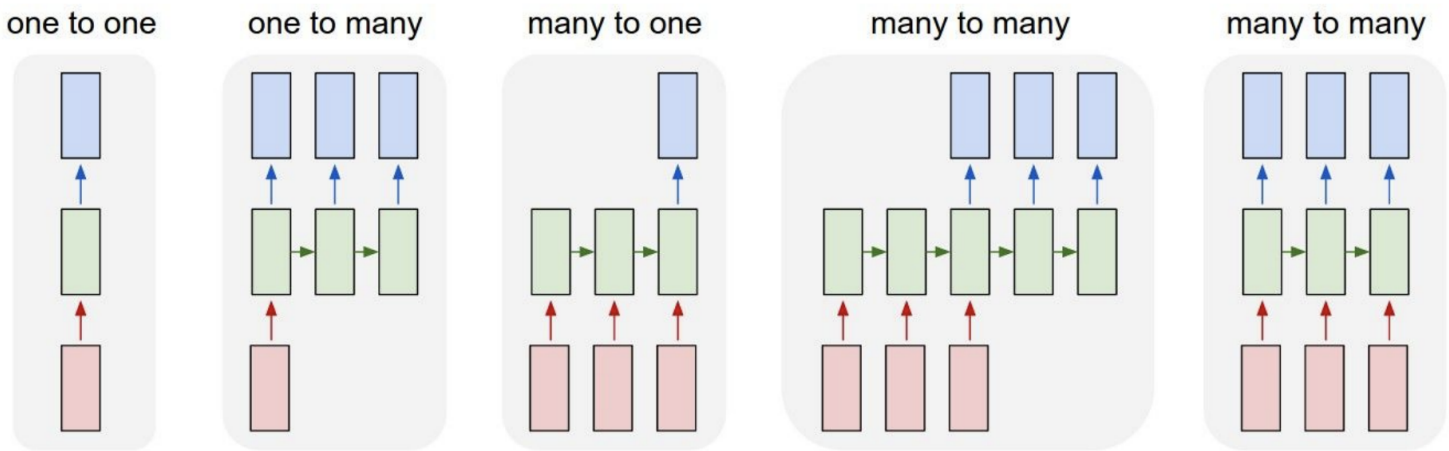


Figure: Source: Fei-Fei Li & Justin Johnson & Serena Yeung, Standford CS231, Recurrent Neural Networks

Type	Input T_x	Output T_y	Example
One-to-One	$T_x = 1$	$T_y = 1$	Vanilla NN
One-to-Many	$T_x = 1$	$T_y > 1$	Image Captioning
Many-to-One	$T_x > 1$	$T_y = 1$	Sentiment Classification
Many-to-Many	$T_x \neq T_y$	$T_x \neq T_y$	Machine Translation
Many-to-Many	$T_x = T_y$	$T_x = T_y$	Name Entity Recognition

What are Sequence to Sequence Models?

- Generally used to convert one set of tokens into another
  - Many - to - Many RNN
- Bleak view: map a sequence of indexes to another independent set of indexes

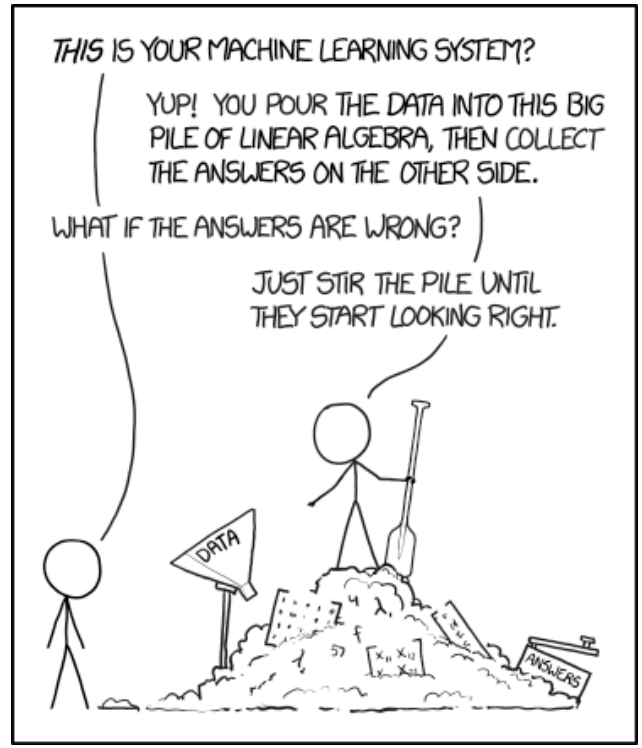


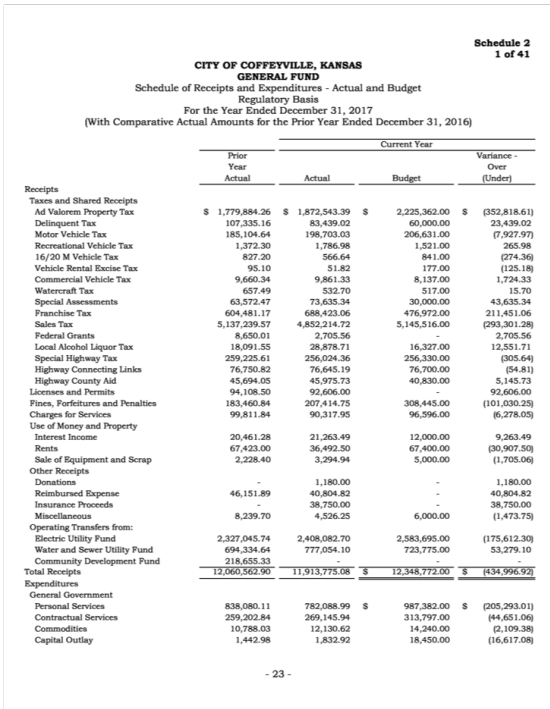
Figure: The pile gets soaked with data and starts to get mushy over time, so it's technically recurrent.

Use Case: Extracting Tables from Images

Primer: What are we trying to accomplish

- Tabular data is locked in PDFs
  - Either as a typeset document
  - Or as a scanned image
- Sometimes copy & paste won't suffice (or impossible with images)
- Let's take a quick look at the full pipeline to see where Seq2Seq models can fit in

What is our data?



Schedule 2
1 of 41

CITY OF COFFEYVILLE, KANSAS
GENERAL FUND
Schedule of Receipts and Expenditures - Actual and Budget
Regulatory Basis
For the Year Ended December 31, 2017
(With Comparative Actual Amounts for the Prior Year Ended December 31, 2016)

	Prior Year Actual	Current Year Actual	Current Year Budget	Current Year Variance - Over (Under)
Receipts				
Taxes and Shared Receipts				
Ad Valorem Property Tax	$ 1,779,884.26	$ 1,872,543.39	$ 2,225,362.00	$ (352,818.61)
Delinquent Tax	107,335.16	83,439.02	60,000.00	23,439.02
Motor Vehicle Tax	185,104.64	198,703.03	206,631.00	(7,927.97)
Recreational Vehicle Tax	1,372.30	1,786.98	1,521.00	265.98
16/20 M Vehicle Tax	827.20	566.64	841.00	(274.36)
Vehicle Rental Excise Tax	95.10	51.82	177.00	(125.18)
Commercial Vehicle Tax	9,660.34	9,861.33	8,137.00	1,724.33
Watercraft Tax	657.49	532.70	517.00	15.70
Special Assessments	63,572.47	73,635.34	30,000.00	43,635.34
Franchise Tax	604,481.17	688,423.06	476,972.00	211,451.06
Sales Tax	5,137,239.57	4,852,214.72	5,145,516.00	(293,301.28)
Federal Grants	8,650.01	2,705.56	-	2,705.56
Local Alcohol Liquor Tax	18,091.55	28,878.71	16,327.00	12,551.71
Special Highway Tax	259,225.61	256,024.36	256,330.00	(305.64)
Highway Connecting Links	76,750.82	76,645.19	76,700.00	(54.81)
Highway County Aid	45,694.05	45,975.73	40,830.00	5,145.73
Licenses and Permits	94,108.50	92,606.00	-	92,606.00
Fines, Forfeitures and Penalties	183,460.84	207,414.75	308,445.00	(101,030.25)
Charges for Services	99,811.84	90,317.95	96,596.00	(6,278.05)
Use of Money and Property				
Interest Income	20,461.28	21,263.49	12,000.00	9,263.49
Rents	67,423.00	36,492.50	67,400.00	(30,907.50)
Sale of Equipment and Scrap	2,228.40	3,294.94	5,000.00	(1,705.06)
Other Receipts				
Donations	-	1,180.00	-	1,180.00
Reimbursed Expense	46,151.89	40,804.82	-	40,804.82
Insurance Proceeds	-	38,750.00	-	38,750.00
Miscellaneous	8,239.70	4,526.25	6,000.00	(1,473.75)
Operating Transfers from:				
Electric Utility Fund	2,327,045.74	2,408,082.70	2,583,695.00	(175,612.30)
Water and Sewer Utility Fund	694,334.64	777,054.10	723,775.00	53,279.10
Community Development Fund	218,655.33	-	-	-
Total Receipts	12,060,562.90	11,913,775.08	$ 12,348,772.00	$ (434,996.92)
Expenditures				
General Government				
Personal Services	838,080.11	782,088.99	$ 987,382.00	$ (205,293.01)
Contractual Services	259,202.84	269,145.94	313,797.00	(44,651.06)
Commodities	10,788.03	12,130.62	14,240.00	(2,109.38)
Capital Outlay	1,442.98	1,832.92	18,450.00	(16,617.08)

5. CAPITAL LEASE OBLIGATIONS

The City has entered into a capital lease agreement in order to finance the purchase of a Blackstart Generator. Payments are made of $77,729.23 semi-annually, including interest at approximately 5.00%. Final maturity for the lease is in 2026. Future minimum lease payments are as follows:

Year Ended December 31	Totals
2018	$ 155,458.46
2019	155,458.46
2020	155,458.46
2021	155,458.46
2022	155,458.46
2023-2026	505,870.61
	1,283,162.91
Less imputed interest	(174,862.64)
Net Present Value of Minimum Lease Payments	1,108,300.27
Less: Current Maturities	(118,236.87)
Long-Term Capital Lease Obligations	$ 990,063.40

The City has entered into a capital lease agreement in order to finance the purchase of a Fire Truck. Payments are made of $30,023.18 semi-annually, including interest at approximately 3.27%. Final maturity for the lease is in 2027. Future minimum lease payments are as follows:

Year Ended December 31	Totals
2018	$ 60,046.36
2019	60,046.36
2020	60,046.36
2021	60,046.36
2022	60,046.36
2023-2027	300,110.21
	600,342.01
Less imputed interest	(91,679.72)
Net Present Value of Minimum Lease Payments	508,662.29
Less: Current Maturities	(43,858.90)
Long-Term Capital Lease Obligations	$ 464,803.39

6. OPERATING LEASES

As of December 31, 2017 the City has entered into an operating lease for office equipment. Rent expense for the year ended December 31, 2017, was $18,799.36. Under the current lease agreements, the future minimum rental payments are as follows:

2018	$ 4,216.68
2019	3,162.51

Schedule 1

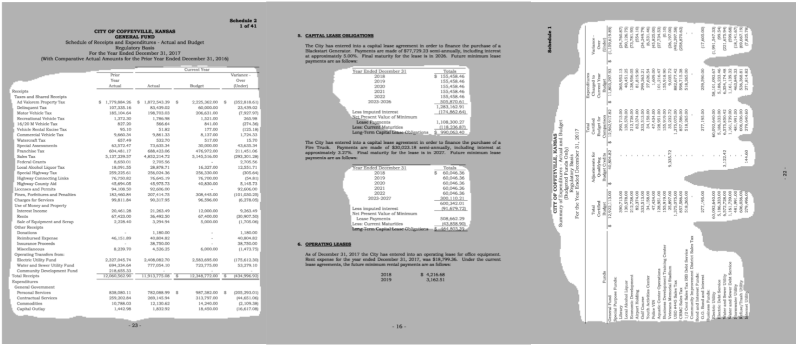
CITY OF COFFEYVILLE, KANSAS
Summary of Expenditures - Actual and Budget
(Budgeted Funds Only)
Regulatory Basis
For the Year Ended December 31, 2017

Funds	Total Certified Budget	Adjustments for Qualifying Budget Credits	Total Certified Budget for Comparison	Expenditures Charged to Current Year	Variance - Over (Under)
General Fund	$ 12,922,113.00	$ 40,804.82	$ 12,962,917.82	$ 11,803,297.93	$ (1,159,619.89)
Special Purpose Funds:					
Library	390,713.00	-	390,713.00	365,952.13	(24,760.87)
Local Alcohol Liquor	193,578.00	-	193,578.00	49,451.25	(80,126.75)
Economic Development	212,794.00	-	212,794.00	138,965.05	(73,828.95)
Airport Holding	82,374.00	-	82,374.00	81,819.90	(554.10)
Golf Course	333,513.00	-	333,513.00	299,263.21	(34,249.79)
Youth Activities Center	34,158.00	-	34,158.00	27,626.54	(6,531.46)
Police VIPS	47,434.00	-	47,434.00	1,609.00	(45,825.00)
Aquatic Center Operations	128,345.00	-	128,345.00	101,214.67	(27,130.33)
Business Development Training Center	155,919.00	-	155,919.00	155,918.90	(0.10)
Veterans Memorial Stadium	25,897.00	9,335.72	35,232.72	9,035.72	(26,197.00)
USD #445 Sales Tax	1,375,075.00	-	1,375,075.00	882,677.42	(492,397.58)
CBMC Sales Tax	857,286.00	-	857,286.00	598,715.38	(258,570.62)
CBMC Special Sales Tax	518,265.00	-	518,265.00	518,265.00	-
Community Improvement District Sales Tax	-	-	-	-	-
Bond and Interest Funds:					
G.O. Bond and Interest	277,195.00	-	277,195.00	259,590.00	(17,605.00)
Business Funds:					
Electric Debt Service	60,092,640.00	-	60,092,640.00	58,101,092.67	(1,991,547.33)
Water and Sewer Utility	5,186,333.00	-	5,186,333.00	5,186,233.46	(99.54)
Water and Sewer Debt Service	6,572,728.00	3,122.42	6,575,850.42	6,354,174.48	(221,675.94)
Stormwater Utility	1,161,739.00	-	1,161,739.00	1,161,139.32	(599.68)
Refuse Collection Utility	481,991.00	-	481,991.00	463,849.33	(18,141.67)
Internet Utility	646,000.00	-	646,000.00	628,354.61	(17,645.39)
	279,496.00	144.60	279,640.60	271,814.82	(7,825.78)

Current Pipeline for Table Extraction

- Convert to an Image
- Binary Page Classification: Is there a table on this page?
- Table Segmentation: U-Net Model
- Optical Character Recognition via Tesseract
- **Column Segmentation**
- Table Alignment
- Dump to CSV

Visual: Table Segmentation

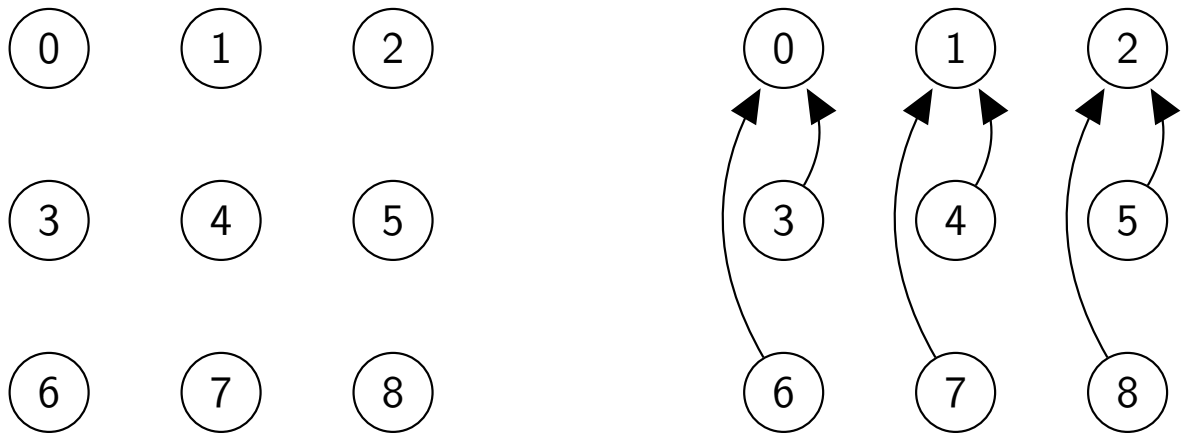


Visual: Tesseract OCR Output

level	page_num	block_num	par_num	line_num	word_num	left	top	width	height	conf	text
5	1	1	1	1	1	1092	100	62	27	95	For
5	1	1	1	1	2	1165	100	55	26	95	the
5	1	1	1	1	3	1239	100	82	25	96	Year
5	1	1	1	1	4	1335	100	114	24	96	Ended
5	1	1	1	1	5	1463	100	82	24	93	June
5	1	1	1	1	7	1630	100	88	22	36	ZU18
5	1	1	1	2	1	1295	146	219	43	96	(Continued)
5	1	1	1	3	1	460	263	128	32	96	NOTE
5	1	1	1	3	2	605	263	16	31	90	1
5	1	1	1	3	3	637	280	13	5	87	-
5	1	1	1	3	4	665	260	245	34	96	SUMMARY
5	1	1	1	3	5	925	260	61	32	95	OF
5	1	1	1	3	6	1002	257	303	34	96	SIGNIFICANT
5	1	1	1	3	7	1319	254	318	34	95	ACCOUNTING
5	1	1	1	3	8	1651	252	220	34	95	POLICIES
5	1	1	1	3	9	1886	251	232	42	96	(Continued)
5	1	1	1	4	1	537	372	39	31	93	H.
5	1	1	1	4	2	612	369	206	34	96	Inventories
5	1	1	1	4	3	833	368	64	33	96	and
5	1	1	1	4	4	911	367	141	43	96	Prepaid
5	1	1	1	4	5	1066	368	98	31	96	Items
5	1	1	1	5	1	611	477	207	34	96	Inventories
5	1	1	1	5	2	831	487	54	22	96	are
5	1	1	1	5	3	896	476	123	32	96	valued
5	1	1	1	5	4	1031	480	32	27	96	at
5	1	1	1	5	5	1073	475	55	32	96	the
5	1	1	1	5	6	1141	474	101	33	96	lower
5	1	1	1	5	7	1253	473	41	33	96	of
5	1	1	1	5	8	1301	477	73	28	93	cost
5	1	1	1	5	9	1385	471	147	42	90	(first-in,
5	1	1	1	5	10	1545	470	163	41	96	first-out)

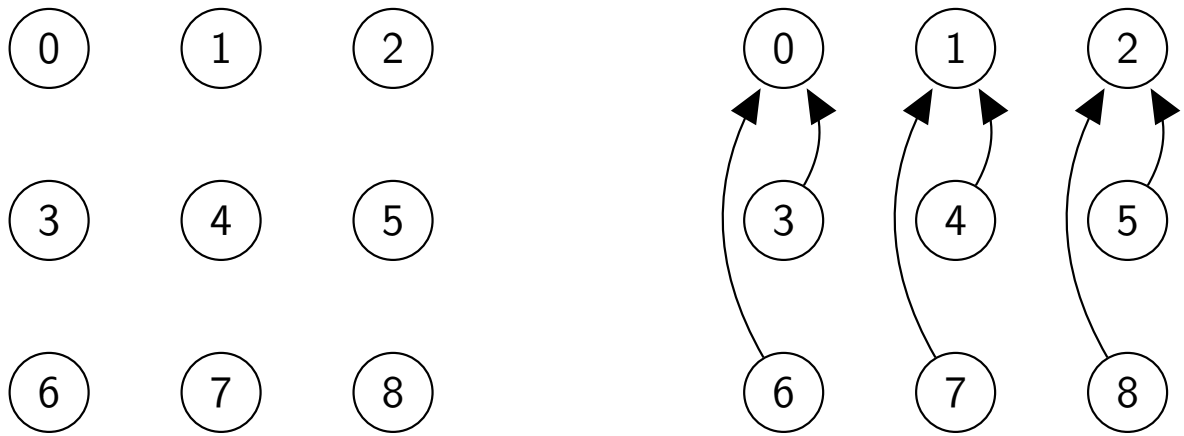
Column Segmentation

- Naive approach is to merge words close together
  - But spacing is **variable** in tables
  - Text justification (left, center, right)
- **Union-Find Disjoint Set (UFDS) Algorithm** to group columns by **Intersection Over Union (IOU)**



Column Segmentation

- Naive approach is to merge words close together
  - But spacing is **variable** in tables
  - Text justification (left, center, right)
- **Union-Find Disjoint Set (UFDS) Algorithm** to group columns by **Intersection Over Union (IOU)**



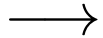
- **Idea:** Use a Seq2Seq to learn correct segmentation

Visual: Alignment

				——"——sSurrentYear t—('SOSCS~™S										
	Prior													Variance -
	Year													Over
	Actual				Actual				Budget					(Under)
Receipts														
Taxes and Shared Receipts														
Ad Valorem Property Tax	$1,779,884.26				$1,872,543.39			§§	2,225,362			$	-352,818.61	
Delinquent Tax	107,335.16				83,439.02				60,000				23,439.02	
Motor Vehicle Tax	185,104.64				198,703.03				206,631				-7,927.97	
Recreational Vehicle Tax	1,372.3				1,786.98				1,521				265.98	
16/20 M Vehicle Tax	827.2				566.64				841				-274.36	
Vehicle Rental Excise Tax	95.1				51.82				177				-125.18	
Commercial Vehicle Tax	9,660.34				9,861.33				8,137				1,724.33	
Watercraft Tax	657.49				532.7				517				15.7	
Special Assessments	63,572.47				73,635.34				30,000				43,635.34	
Franchise Tax	604,481.17				688,423.06				476,972				211,451.06	
Sales Tax	5,137,239.57				4,852,214.72				5,145,516				-293,301.28	
Federal Grants	8,650.01				2,705.56					-			2,705.56	
Local Alcohol Liquor Tax	18,091.55				28,878.71				16,327				12,551.71	
Special Highway Tax	259,225.61				256,024.36				256,330				-305.64	
Highway Connecting Links	76,750.82				76,645.19				76,700				-54.81	
Highway County Aid	45,694.05				45,975.73				40,830				5,145.73	
Licenses and Permits	94,108.5				92,606					-			92,606	
Fines, Forfeitures and Penalties	183,460.84				207,414.75				308,445				-101,030.25	
Charges for Services	99,811.84				90,317.95				96,596				-6,278.05	
Use of Money and Property														
Interest Income	20,461.28				21,263.49				12,000				9,263.49	
Rents	67,423				36,492.5				67,400				-30,907.5	
Sale of Equipment and Scrap	2,228.4				3,294.94				5,000				-1,705.06	
Other Receipts														
Donations		-			1,180					-			1,180	
Reimbursed Expense	46,151.89				40,804.82					-			40,804.82	
Insurance Proceeds		-			38,750					-			38,750	
Miscellaneous	8,239.7				4,526.25				6,000				-1,473.75	
Operating Transfers from:														
Electric Utility Fund	2,327 ,045.74				2,408,082.7				2,583,695				-175,612.3	
Water and Sewer Utility Fund	694,334.64				777,054.1				723,775				53,279.1	
Community Development Fund	218,655.33					-				-				-
Total Receipts	12,060,562.9				11,913,775.08			§§	12,348,772		§§		-434,996.92	
Expenditures							TT							
General Government														
Personal Services	838,080.11				782,088.99				$ 987,382		§§		-205,293.01	
Contractual Services	259,202.84				269,145.94				313,797				-44,651.06	
Commodities	10,788.03				12,130.62				14,240				-2,109.38	
Capital Outlay	1,442.98				1,832.92				18,450				-16,617.08	
			-92-											

Problem Statement: Column Segmentation

Year Ended December 31	Totals
2018	$ 155,458.46
2019	155,458.46
2020	155,458.46
2021	155,458.46
2022	155,458.46
2023-2026	505,870.61
	1,283,162.91
Less imputed interest	(174,862.64)
Net Present Value of Minimum Lease Payments	1,108,300.27
Less: Current Maturities	(118,236.87)
Long-Term Capital Lease Obligations	$ 990,063.40



Year Ended December 31		Totals
	2018	$155,458.46
	2019	155,458.46
	2020	155,458.46
	2021	155,458.46
	2022	155,458.46
2023-2026		505,870.61
		1,283,162.91
Less imputed interest		-174,862.64
Net Present Value of Minimum		
Lease Payments		1,108,300.27
Less: Current Maturities		-118,236.87
Long-Term Capital Lease Obligations		$990,063.40

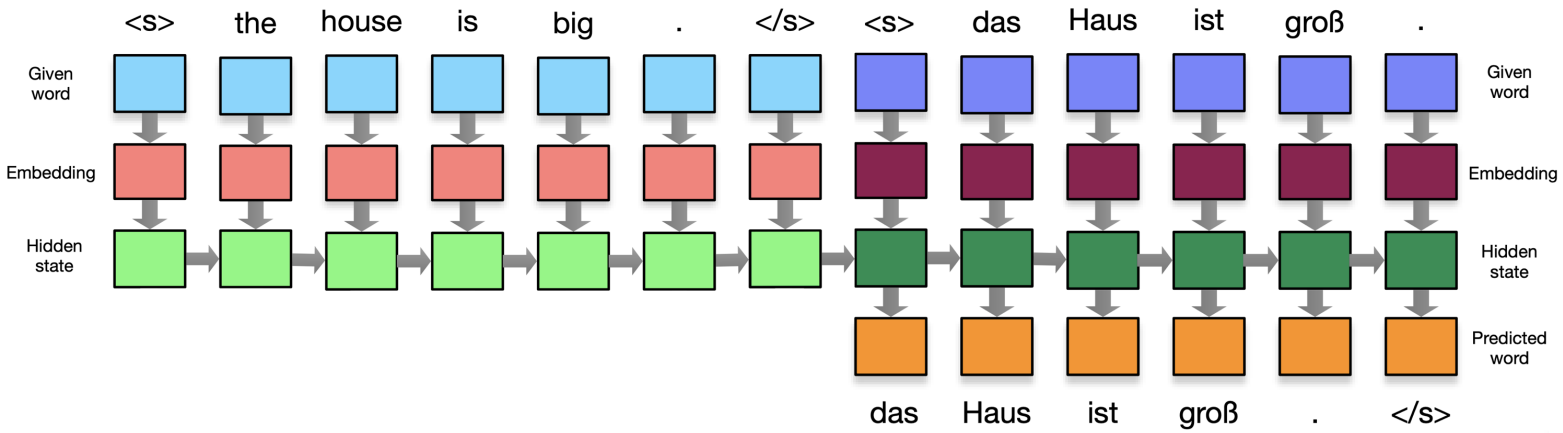
- Given a sequence of text, can we predict where to insert columns?

Revisiting Column Segmentation as a Seq2Seq Problem

- Translation Task
- Given a sequence of tokens
- Translate into the same set of tokens, but with a special ⟨SEG⟩ token

```
Less  imputed  interest        (  174,862.64  )
                          ⇓
Less  imputed  interest  ⟨SEG⟩  (  174,862.64  )
```

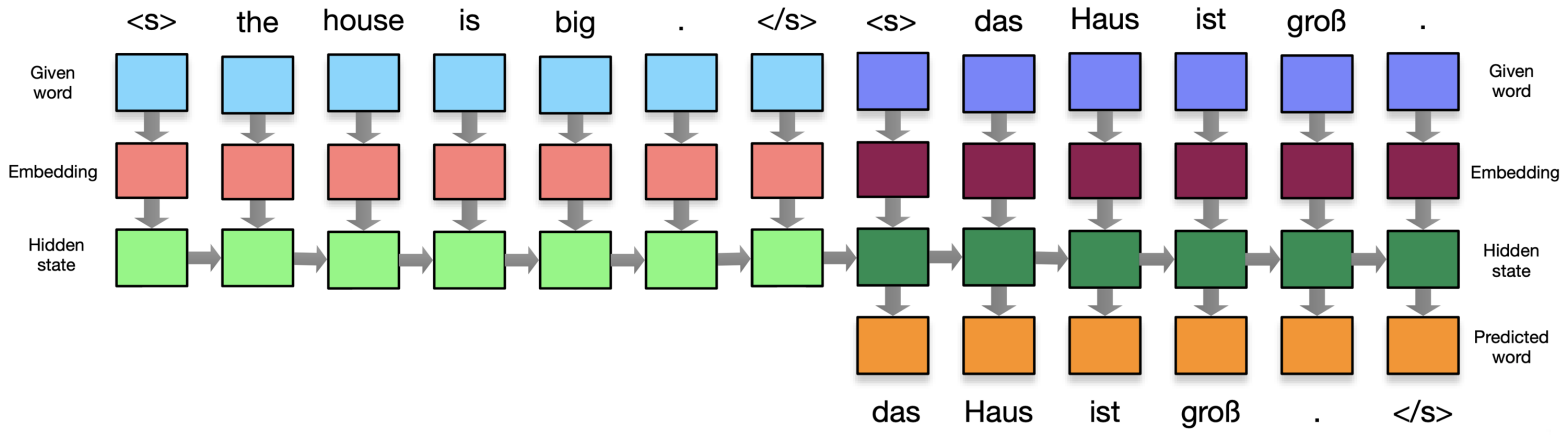
Seq2Seq Architecture



- We can use a **Encoder-Decoder** style model!

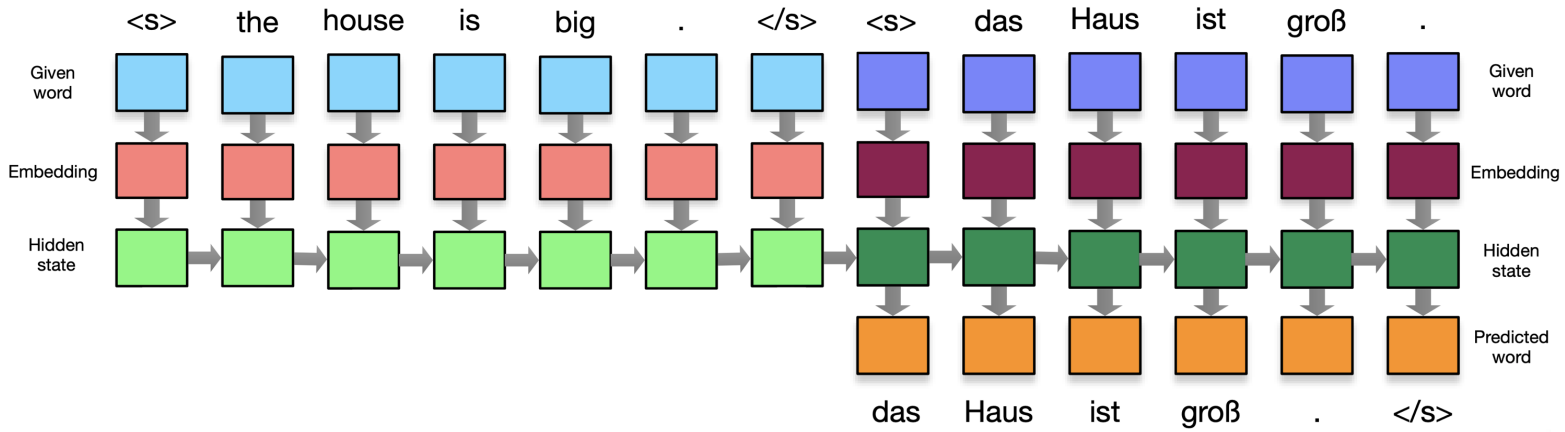
Encoder-Decoder Architecture

Overview: Encoders and Decoders



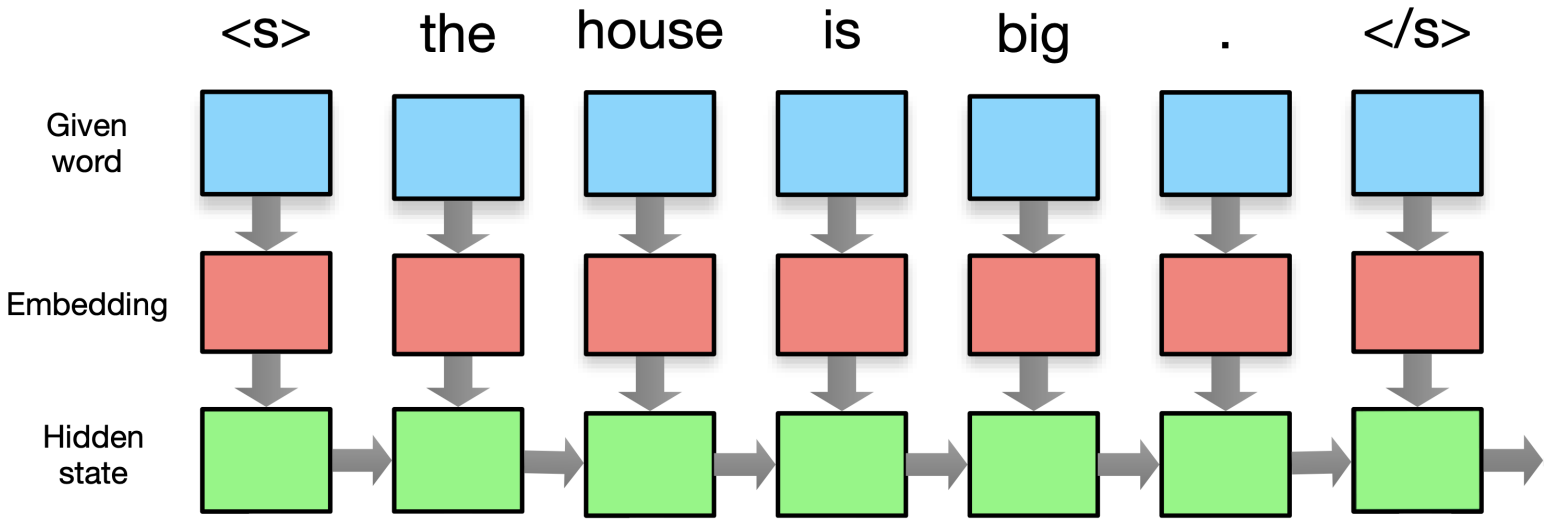
- We have 2 sub-networks: an **Encoder** and a **Decoder**
- Encoders
  - Give the source sentence meaning
- Decoders
  - Emit a variable-length sequence
- We will discuss how to connect the two for joint training

Overview: Encoders and Decoders



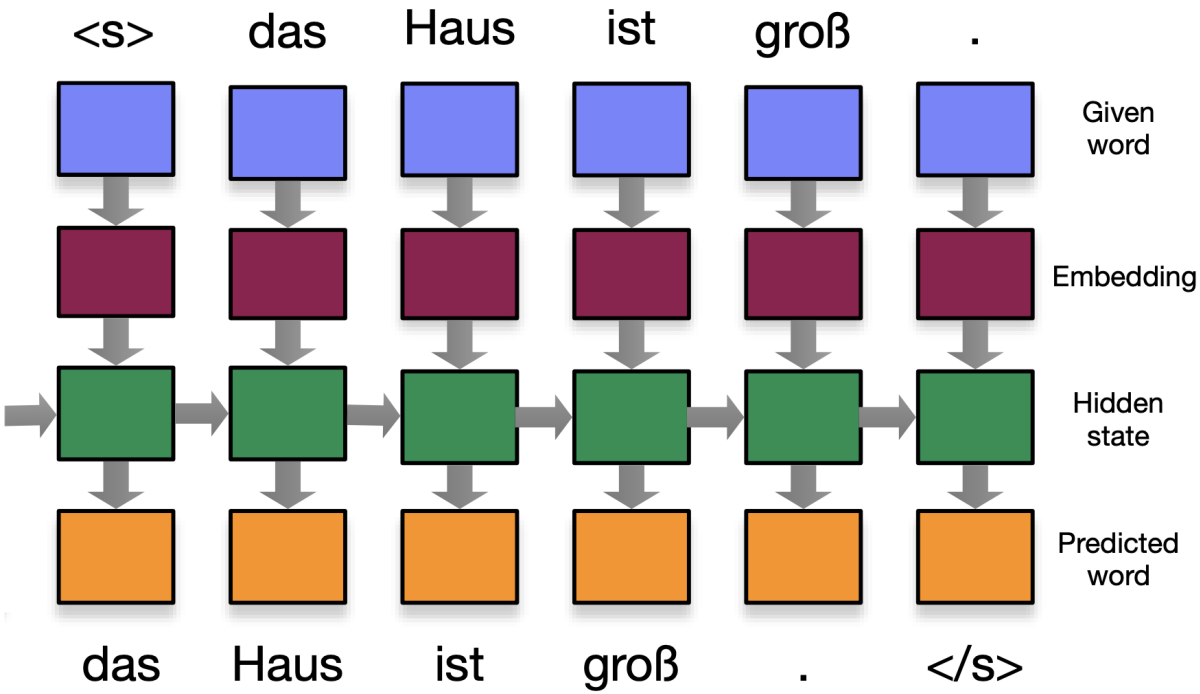
- Encapsulation allows for flexible design choices
- **Embeddings**
  - Pre-trained
  - DIY
- **Recurrent Layer**
  - Type
  - Depth
  - Directionality

What makes an Encoder?



- Recall: **Encoders** give the source sentence meaning
- Effectively a language model, without a layer to predict the next word
- Idea is to pass on the hidden state, and possibly use the encodings directly

What makes a Decoder?



- Recall: **Decoders** provide a new sequence conditioned on the Encoder's hidden state
- Starts with the Encoder's hidden state, and predicts one token at a time
- Re-feed the predicted token back into the decoder

Word Embeddings: Practical Considerations

Column Segmentation Issue: Vocabulary Size

- Is the text actually important?
- Over **10K unique tokens**, mostly numbers

Column Segmentation Issue: Vocabulary Size

- Is the text actually important?
- Over **10K unique tokens**, mostly numbers
- **Solutions:**
  - Token $\rightarrow$ Part-of-Speech
  - Numeric Value $\rightarrow \langle \text{NUM} \rangle$
- Reduces vocabulary size to **21 tokens**
- But lets take a look at alternative solutions to managing vocab size

Recap: Word Embeddings

$$
\begin{aligned}
\texttt{the} &\rightarrow \begin{bmatrix} 1.23 & -0.58 & 0.22 & -0.80 & 0.61 \end{bmatrix} \\
\texttt{cat} &\rightarrow \begin{bmatrix} -1.10 & 1.23 & 0.17 & -0.21 & 1.43 \end{bmatrix} \\
\texttt{is} &\rightarrow \begin{bmatrix} -0.26 & 0.70 & 0.27 & 0.59 & -1.04 \end{bmatrix} \\
\texttt{black} &\rightarrow \begin{bmatrix} -1.13 & -0.81 & -0.53 & -0.59 & 0.26 \end{bmatrix}
\end{aligned}
$$

- A trick to map tokens to vector representations

Recap: Pretrained Word Embeddings

- Co-occurence Matrix
- Pointwise Mutual Information
- SVD Co-occurence
- Ngram
- CBOW, Skip Gram
- GloVe
- ELMo
- BERT

DIY Embeddings

- We can always train our own!

Special Tokens for Sequence Modeling

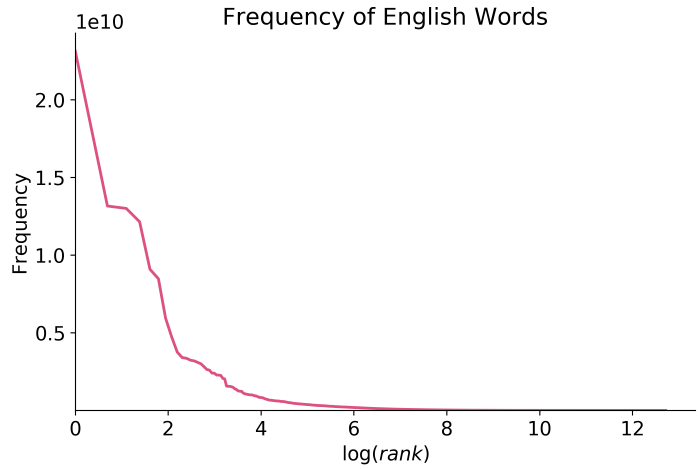
Special Tokens for Sequence Modeling

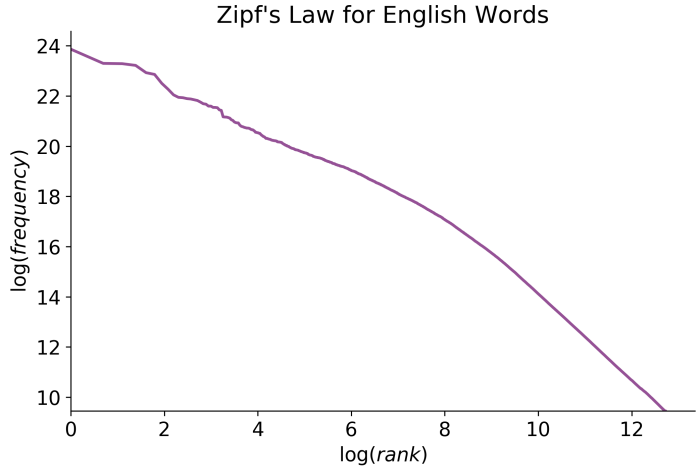
- ⟨PAD⟩ → Padding / Masking
- ⟨UNK⟩ → Unknown words
- ⟨SOS⟩ → Start of Sequence
- ⟨EOS⟩ → End of Sequence

⟨SOS⟩	Data	Science	is	the	⟨UNK⟩	!	⟨EOS⟩
⟨SOS⟩	I	⟨UNK⟩	for	S&P	.	⟨EOS⟩	⟨PAD⟩
⟨SOS⟩	Hello	World	!	⟨EOS⟩	⟨PAD⟩	⟨PAD⟩	⟨PAD⟩

Managing the Vocab Size

- Languages are unevenly distributed
- Many rare words, names $\rightarrow$ inflates the size of the vocabulary
- **Problem:**
  - Large embedding matrices for source, target language
  - Large output layers for prediction and softmax





Morphology, Compounding, and Transliteration

- Morphological Analysis

  tweet, *tweets*, *tweeted*, *tweeting*, *retweet*, ...

- Compound Splitting
  - homework $\rightarrow$ home·work
  - website $\rightarrow$ web·site
- Names, Places, Proper Nouns
  - Hoboken, Baltimore, Obama, Michelle
  - Can do Transliteration

Handling Numbers

- Do we really need to encode every number? **NO!**

`I pay 950.00 in May 2007` $>$ `I pay 2007 in May 950.00`

Handling Numbers

- Do we really need to encode every number? **NO!**

I pay 950.00 in May 2007 $>$ I pay 2007 in May 950.00

- **Solution 1:** Replace with a ⟨NUM⟩ token, but

I pay ⟨NUM⟩ in May ⟨NUM⟩ $=$ I pay ⟨NUM⟩ in May ⟨NUM⟩

Handling Numbers

- Do we really need to encode every number? **NO!**

I pay 950.00 in May 2007 > I pay 2007 in May 950.00

- **Solution 1:** Replace with a ⟨NUM⟩ token, but

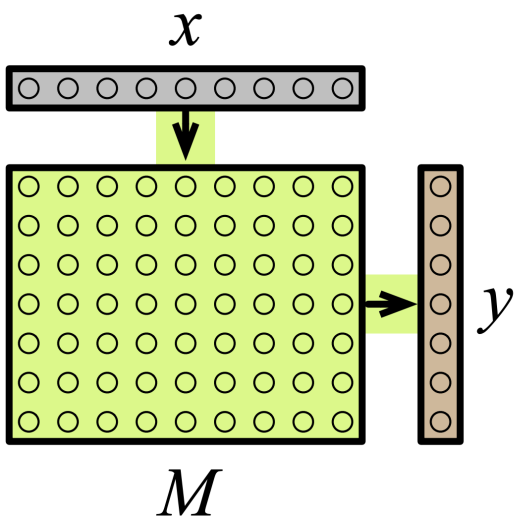
I pay ⟨NUM⟩ in May ⟨NUM⟩ = I pay ⟨NUM⟩ in May ⟨NUM⟩

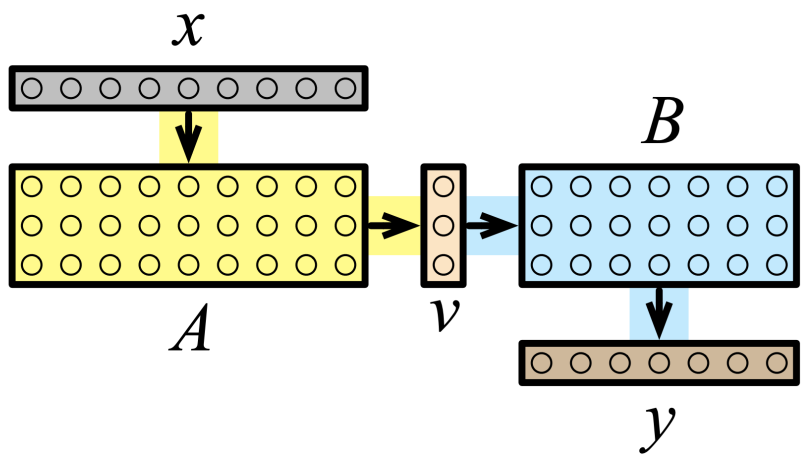
- **Solution 2:** Replace each digit with a unique symbol, e.g. 5

I pay 555.55 in May 5555 > I pay 5555 in May 555.55

- This reduces the need for embeddings, when we can simply do transliteration

Factored Decomposition





- **Problem:** Large input and output vectors
  - $|x| = 20,000$, $|y| = 50,000 \rightarrow |M| = 1,000,000,000$
- **Solution:** Use a bottleneck with smaller matrices A, B
  - $|v| = 100 \rightarrow |A| = 2,000,000$, $|B| = 5,000,000$
  - Total Parameters: $7,000,000$

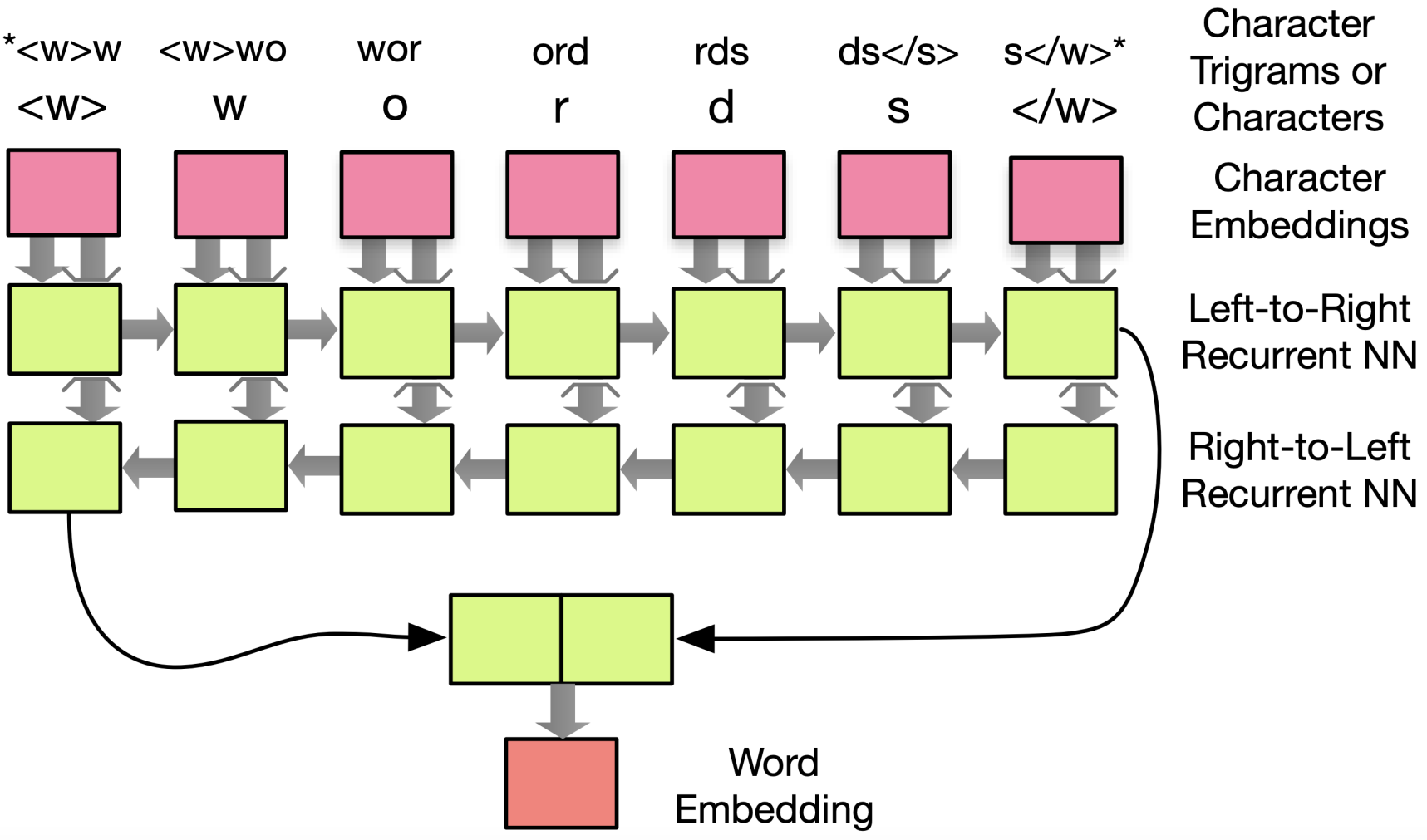
Character-Based Models

- Instead use embeddings for character string
  `b e a u t i f u l`
- Idea is to induce embeddings for unseen morphological variants:
  `beautiful`
- Tokens are single characters, symbols, whitespace

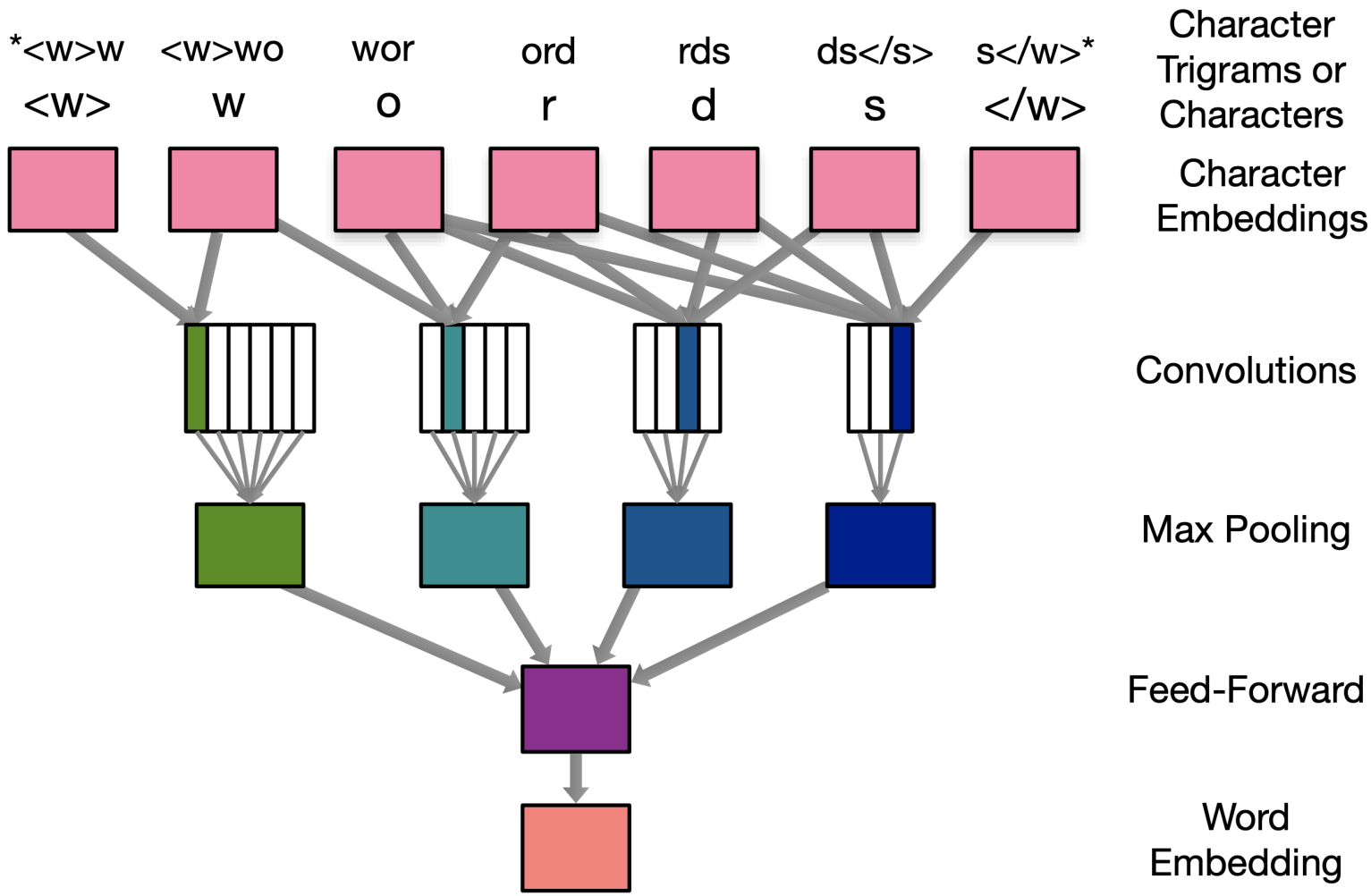
Character-Based Models

- Instead use embeddings for character string
  `b e a u t i f u l`
- Idea is to induce embeddings for unseen morphological variants:
  `beautiful`
- Tokens are single characters, symbols, whitespace
- Generally poor performance

Character-Based Models



Character-Based Models



BPE Subwords

t h e · f a t · c a t · i s · i n · t h e · t h i n · b a g

t h → th	th e · f a t · c a t · i s · i n · th e · th i n · b a g
a t → at	th e · f at · c at · i s · i n · th e · th i n · b a g
i n → in	th e · f at · c at · i s · in · th e · th in · b a g
th e → the	the · f at · c at · i s · in · the · th in · b a g

- Breaks words into subwords
  - Starts with the character set
  - Merges the most frequent pairs, one per iteration
- Unsupervised (accidental) morphology (frequency suffixes)

BPE Tokenization, The Great Gatsby

- Unique Characters: 39
- Unique BPE Tokens (1K iterations): 801
- Unique BPE Tokens (5K iterations): 3,371
- Unique Tokens: 5,856

100

```
s@@ o w@@ e be@@ at on , b@@ o@@ at@@ s a@@ g@@ a@@ in@@ s@@ t
the c@@ ur@@ r@@ en@@ t@@ ,
b@@ or@@ n@@ e b@@ ac@@ k c@@ e@@ as@@ el@@ es@@ s@@ ly
in@@ to the p@@ a@@ st@@ .
```

1K

```
so we be@@ at on, bo@@ at@@ s against the c@@ ur@@ r@@ ent@@ ,
bor@@ n@@ e back c@@ e@@ as@@ el@@ ess@@ ly into the pa@@ st@@ .
```

10K

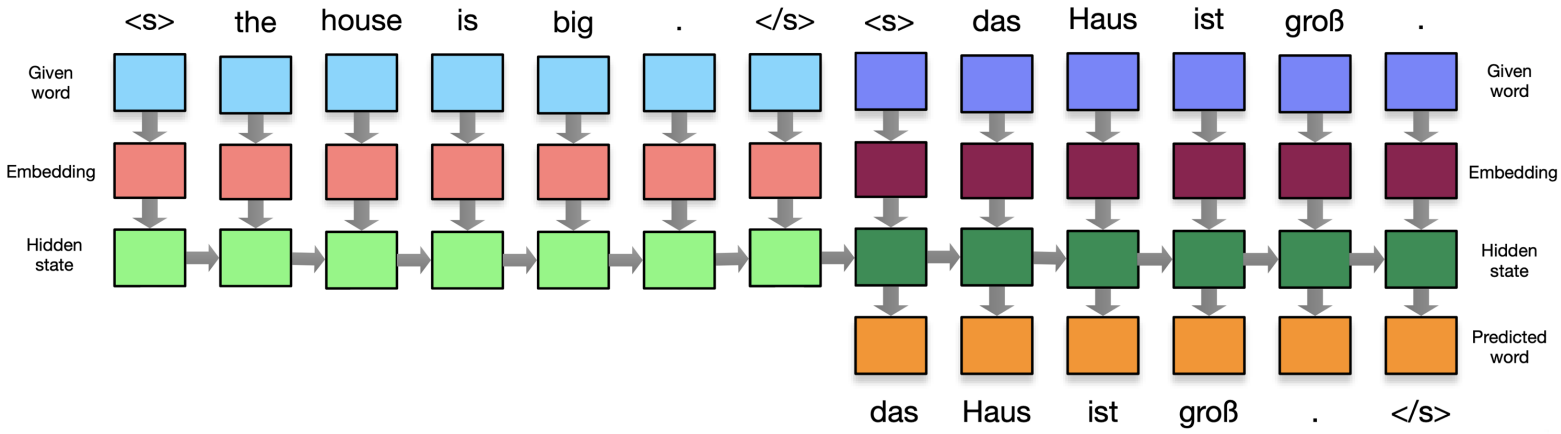
```
so we beat on , bo@@ ats against the curr@@ ent ,
bor@@ ne back ceaselessly into the past .
```

50K

```
so we beat on , boats against the current ,
borne back ceaselessly into the past .
```

Column Segmentation Architecture

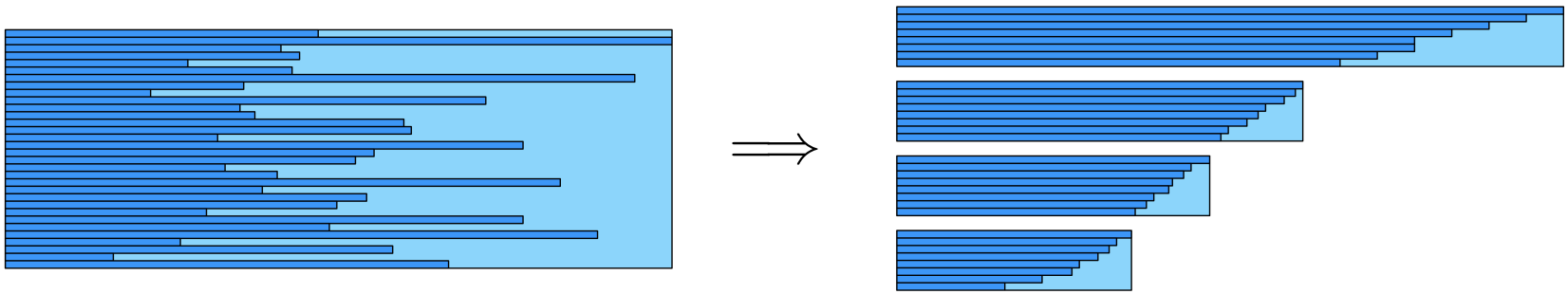
Seq2Seq Architecture




```
Seq2Seq(
  (encoder): Encoder(
    (embedding): Embedding(21, 256)
    (gru): GRU(256, 128, num_layers=2, batch_first=True)
  )
  (decoder): Decoder(
    (embedding): Embedding(22, 256)
    (gru): GRU(256, 128, num_layers=2, batch_first=True)
    (fc): Linear(in_features=128, out_features=22, bias=True)
  )
)
The model has 508,438 trainable parameters
```

Training Considerations

Increasing Throughput through Batching



- We can pad sentences of different lengths to increase batch size
- While also minimizing the use of padding
- Matrix Operations are faster
- **Warning!** Make sure padding is not influencing the hidden state

Teacher Forcing

- Instead of refeeding the predicted token, replace it with the true token randomly
- This is only done during training, not decoding

$$y_{i+1} = \begin{cases} \text{argmax}_j \, \theta_i & \mathcal{U}(0,1) > \text{TF} \\ t_{i+1} & \textit{else} \end{cases}$$

- t_{i+1} is the true token
- TF is the teacher forcing ratio

Cross Entropy and Label Smoothing

$$\ell(\mathbf{x}, y_i) = -\log\left(\frac{\exp x_{y_i}}{\sum_j \exp x_j}\right)$$
$$= -\underbrace{x_{y_i}}_{max} + \underbrace{\log \sum_j \exp x_j}_{min}$$

- Softmax and Cross-Entropy loss assign all the probability mass to a single word
  - LogSumExp is minimized on confident predictions
- Solution: **smooth** the distribution

Quick Maths: LogSumExp Bounds

$$\begin{aligned}
\max\{x_1, \ldots, x_n\} &= \log\left(\exp(\max x_i)\right) \\
&\leq \log\left(\exp(x_1) + \cdots + \exp(x_n)\right) \\
&\leq \log\left(n \cdot \exp(\max x_i)\right) \\
&= \max\{x_1, \ldots, x_n\} + \log(n).
\end{aligned}$$

- The lower bound is *approached* when **all but one** of the arguments approach $-\infty$
- The upper bound is met when **all** the arguments are equal (uniform)
- Hence LogSumExp is minimized on a confident prediction

Cross Entropy and Label Smoothing

- Softmax

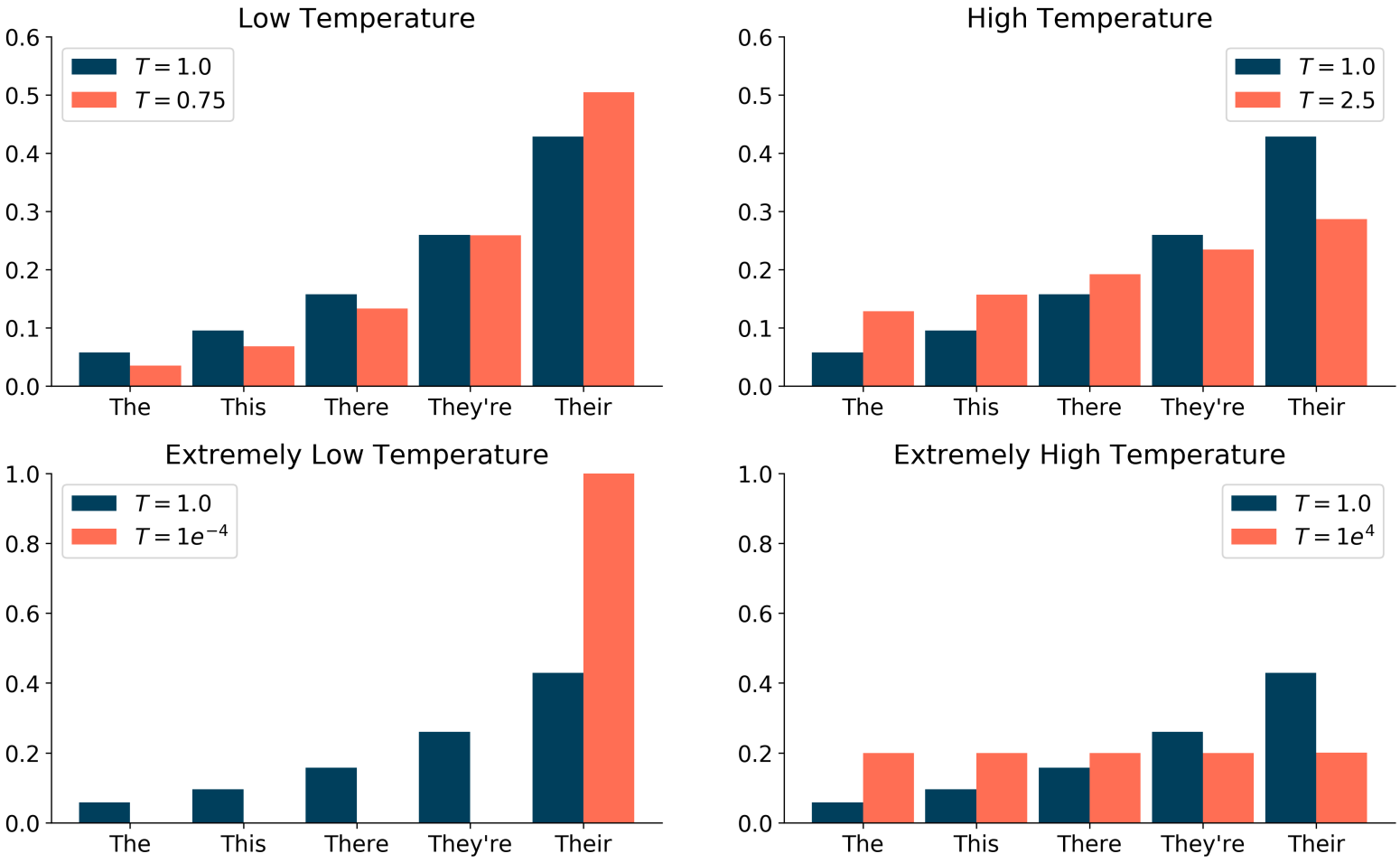
$$p(y_i) = \frac{\exp x_{y_i}}{\sum_j \exp x_j}$$

- Smoothed Softmax with Temperature T

$$p(y_i) = \frac{\exp\left(x_{y_i}/T\right)}{\sum_j \exp\left(x_j/T\right)}$$

- As $T \to \infty$, the distribution is smoother, uniform
- As $T \to 0$, the distribution approaches a kronecker delta centered on the class with the most mass
- **Question:** Why do we divide instead of adding/subtracting?

Visualizing Temperature



Monte Carlo Decoding

- Recall how we select the next token:
  - **Greedy:** Top token weight
  - **Teacher Forcing:** Randomly select the true token
- Note that the outputs are a distribution over the target vocabulary
- **Use these weights in a multinomial to randomly select a continuation**

$$y_{i+1} \sim \text{Multinomial}\left(\theta_i\right)$$

Different Token Decoding Schemes

- **Greedy:**

$$y_{i+1} = \underset{j}{\text{argmax}}\, \theta_i$$

- **Teacher Forcing:**

$$y_{i+1} = \begin{cases} \text{argmax}_j\, \theta_i & \mathcal{U}(0,1) > \text{TF} \\ t_{i+1} & \textit{else} \end{cases}$$

- **Monte Carlo:**

$$y_{i+1} \sim \text{Multinomial}\,(\theta_i)$$

- θ_i are the output weights from the Decoder
- t_{i+1} is the true token at position $i+1$
- TF is the teacher forcing ratio

Masked Loss

- Remember, we don't care what gets predicted after seeing a ⟨EOS⟩
- Hence, we need to mask out the loss for predicted tokens associated with ⟨PAD⟩

pred	→	*le*	*le*	*chat*	*chat*	*chat*	*chat*
		↓	↓	↓	↓	↓	↓
		ℓ	ℓ	ℓ	ℓ	ℓ	0
		↑	↑	↑	↑	↑	↑
true	→	*le*	*chat*	*est*	*noir*	⟨EOS⟩	⟨PAD⟩

- **Solution:** Zero out elements by either:
  - Multiply pad outputs by 0
  - Specify the label to ignore in Cross Entropy call

$$\ell(\mathbf{x}, y_i) = \mathbb{1}_{\{y_i \neq \langle \text{PAD} \rangle\}} \cdot \left(-x_{y_i} + \log \sum_j \exp x_j \right)$$

Automatic Evaluation: BLEU

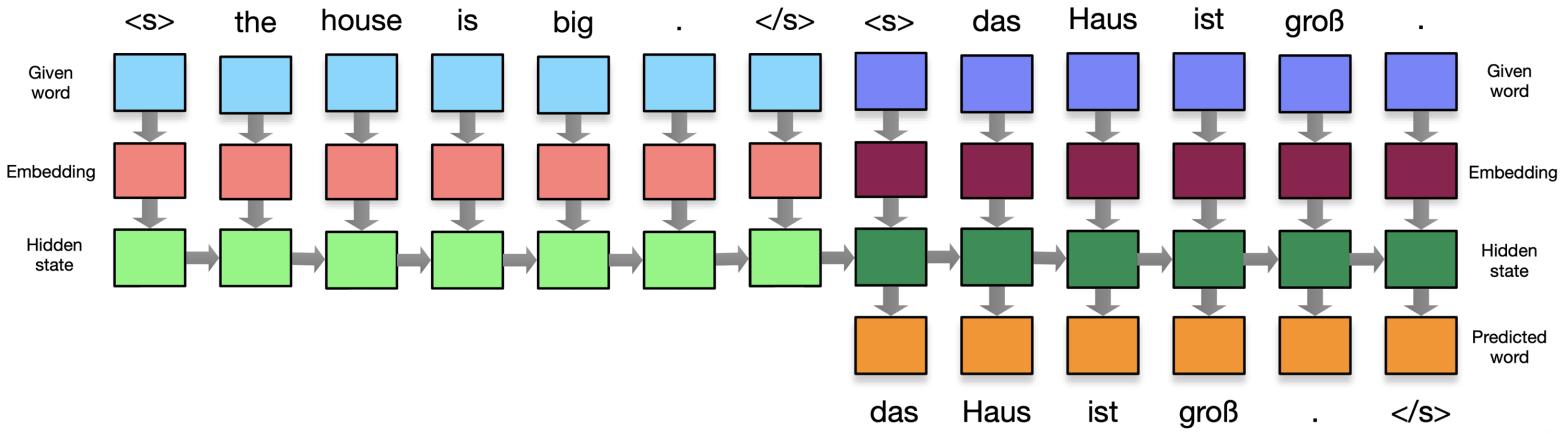
- Bilingual Evaluation Understudy
- For an candidate and reference translation:
  - Compare *n*-gram precision of sizes 1 to 4
  - Brevity penalty for short translations

Source	У меня есть навыки управления временем моркови	
Reference	I have the time management skills of a carrot	1.00
A	You got the scheduling skills of a banana	0.28
B	I have the time to plant lettuce	**0.39**
C	I have time management skills for a carrot	0.35

- **Key: 4-gram, Trigram, Bigram, Unigram, No Match**

Initial Training Results

Recall: Seq2Seq Architecture




```
Seq2Seq(
  (encoder): Encoder(
    (embedding): Embedding(21, 256)
    (gru): GRU(256, 128, num_layers=2, batch_first=True)
  )
  (decoder): Decoder(
    (embedding): Embedding(22, 256)
    (gru): GRU(256, 128, num_layers=2, batch_first=True)
    (fc): Linear(in_features=128, out_features=22, bias=True)
  )
)
The model has 508,438 trainable parameters
```

Seq2Seq: Sample Results

```
>  num num num punct
=  num ⟨SEG⟩ num ⟨SEG⟩ num ⟨SEG⟩ punct
<  num ⟨SEG⟩ num ⟨SEG⟩ num ⟨SEG⟩ punct
```

Seq2Seq: Sample Results

```
>  num num num punct
=  num ⟨SEG⟩ num ⟨SEG⟩ num ⟨SEG⟩ punct
<  num ⟨SEG⟩ num ⟨SEG⟩ num ⟨SEG⟩ punct

>  adj num num num
=  adj ⟨SEG⟩ num ⟨SEG⟩ num ⟨SEG⟩ num
<  adj ⟨SEG⟩ num ⟨SEG⟩ num ⟨SEG⟩ num num num num num
```

Seq2Seq: Sample Results

```
>  num num num punct
=  num ⟨SEG⟩ num ⟨SEG⟩ num ⟨SEG⟩ punct
<  num ⟨SEG⟩ num ⟨SEG⟩ num ⟨SEG⟩ punct

>  adj num num num
=  adj ⟨SEG⟩ num ⟨SEG⟩ num ⟨SEG⟩ num
<  adj ⟨SEG⟩ num ⟨SEG⟩ num ⟨SEG⟩ num num num num num

>  verb noun noun noun noun noun verb
=  verb ⟨SEG⟩ noun noun ⟨SEG⟩ noun noun noun ⟨SEG⟩ verb
<  verb noun ⟨SEG⟩ ⟨SEG⟩ noun noun noun noun ⟨SEG⟩ noun ···
```

Translation Results: TL;DR

- Reasonably for small sequences
- Struggles to terminate properly
  - Maybe more training?
- Long Sequences are a mess

Translation Results: TL;DR

- Reasonably for small sequences
- Struggles to terminate properly
  - Maybe more training?
- Long Sequences are a mess
- **However** this indicates that enough signal can be transfered via the hidden state

Review

- We have covered:
  - What are Seq2Seq Models?
  - How can I reduce my vocabulary size?
  - How do I train a Seq2Seq model?
- Part 2 will explore more advanced concepts such as:
  - Decoding with Beam Search
  - Modeling Recurrent Relations
  - Other applications of Sequence Modeling

Thank You!

Seq2Seq in Action: Column Segmentation

Part 2

Bill Watson

S&P Global

January 30, 2020

Outline Review

- We have covered:
  - Seq2Seq Models, Encoders, Decoders
  - Handling Vocabulary and Word Embeddings
  - Training Tips and Techniques
- Part 2 mixes concepts and practical applications:
  - Table Pipeline Review
  - Decoding with Beam Search
  - Modeling Recurrent Relations (RNNs)
  - Updating the Column Segmentation Model
  - Other applications of Sequence Modeling

Types of Recurrent Models

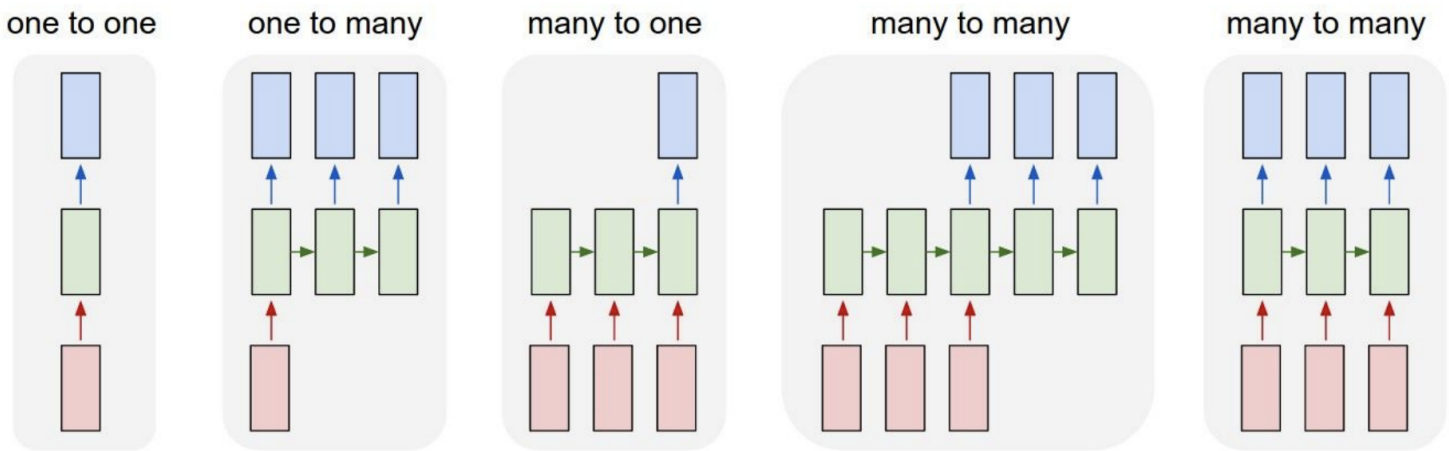


Figure: Source: Fei-Fei Li & Justin Johnson & Serena Yeung, Standford CS231, Recurrent Neural Networks

Type	Input T_x	Output T_y	Example
One-to-One	$T_x = 1$	$T_y = 1$	Vanilla NN
One-to-Many	$T_x = 1$	$T_y > 1$	Image Captioning
Many-to-One	$T_x > 1$	$T_y = 1$	Sentiment Classification
Many-to-Many	$T_x \neq T_y$	$T_x \neq T_y$	Machine Translation
Many-to-Many	$T_x = T_y$	$T_x = T_y$	Name Entity Recognition

Quick Review: Table Extraction

WHEELER INDEPENDENT SCHOOL DISTRICT

NOTES TO THE FINANCIAL STATEMENTS

YEAR ENDED JUNE 30, 2018

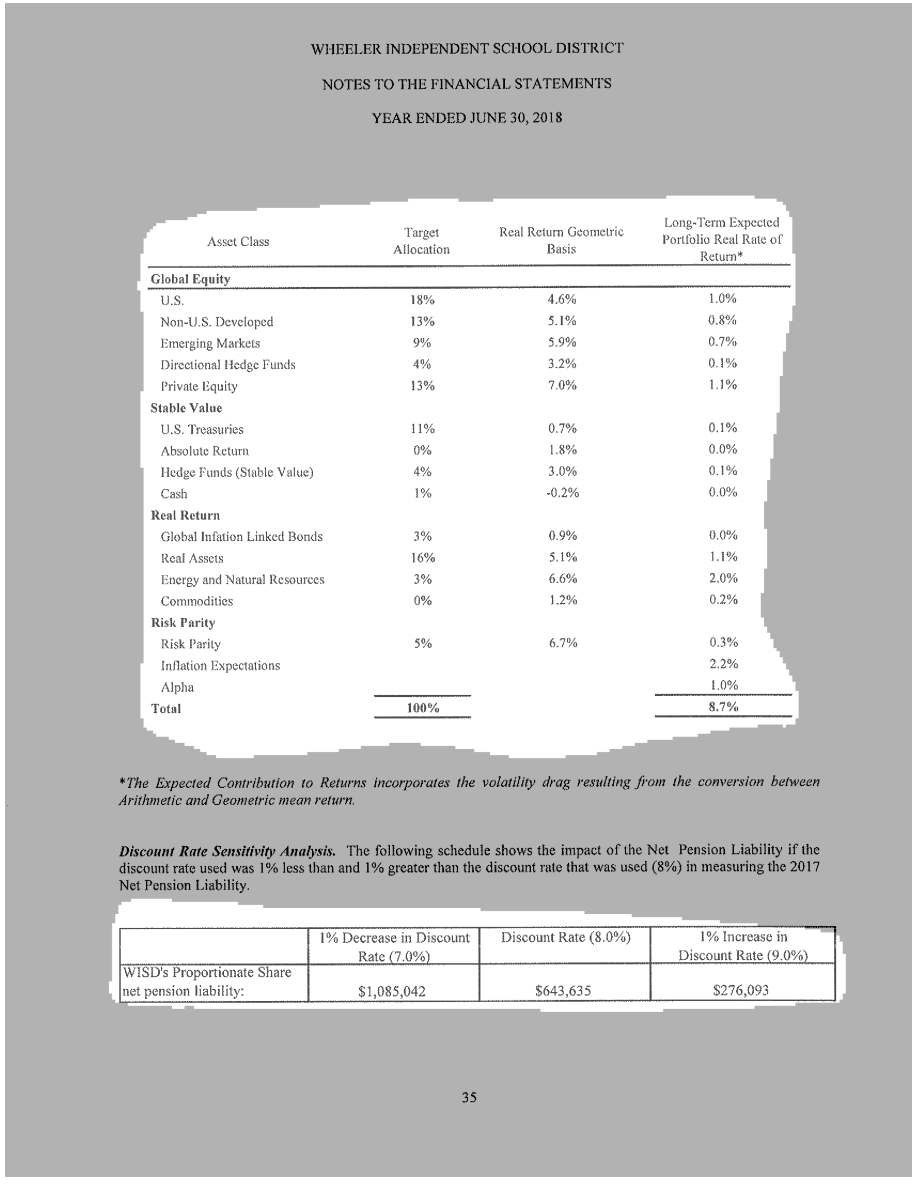
Asset Class	Target Allocation	Real Return Geometric Basis	Long-Term Expected Portfolio Real Rate of Return*
Global Equity			
U.S.	18%	4.6%	1.0%
Non-U.S. Developed	13%	5.1%	0.8%
Emerging Markets	9%	5.9%	0.7%
Directional Hedge Funds	4%	3.2%	0.1%
Private Equity	13%	7.0%	1.1%
Stable Value			
U.S. Treasuries	11%	0.7%	0.1%
Absolute Return	0%	1.8%	0.0%
Hedge Funds (Stable Value)	4%	3.0%	0.1%
Cash	1%	-0.2%	0.0%
Real Return			
Global Inflation Linked Bonds	3%	0.9%	0.0%
Real Assets	16%	5.1%	1.1%
Energy and Natural Resources	3%	6.6%	2.0%
Commodities	0%	1.2%	0.2%
Risk Parity			
Risk Parity	5%	6.7%	0.3%
Inflation Expectations			2.2%
Alpha			1.0%
Total	**100%**		**8.7%**

**The Expected Contribution to Returns incorporates the volatility drag resulting from the conversion between Arithmetic and Geometric mean return.*

Discount Rate Sensitivity Analysis. The following schedule shows the impact of the Net Pension Liability if the discount rate used was 1% less than and 1% greater than the discount rate that was used (8%) in measuring the 2017 Net Pension Liability.

	1% Decrease in Discount Rate (7.0%)	Discount Rate (8.0%)	1% Increase in Discount Rate (9.0%)
WISD's Proportionate Share net pension liability:	$1,085,042	$643,635	$276,093

Quick Review: Table Extraction



WHEELER INDEPENDENT SCHOOL DISTRICT

NOTES TO THE FINANCIAL STATEMENTS

YEAR ENDED JUNE 30, 2018

Asset Class	Target Allocation	Real Return Geometric Basis	Long-Term Expected Portfolio Real Rate of Return*
Global Equity			
U.S.	18%	4.6%	1.0%
Non-U.S. Developed	13%	5.1%	0.8%
Emerging Markets	9%	5.9%	0.7%
Directional Hedge Funds	4%	3.2%	0.1%
Private Equity	13%	7.0%	1.1%
Stable Value			
U.S. Treasuries	11%	0.7%	0.1%
Absolute Return	0%	1.8%	0.0%
Hedge Funds (Stable Value)	4%	3.0%	0.1%
Cash	1%	-0.2%	0.0%
Real Return			
Global Inflation Linked Bonds	3%	0.9%	0.0%
Real Assets	16%	5.1%	1.1%
Energy and Natural Resources	3%	6.6%	2.0%
Commodities	0%	1.2%	0.2%
Risk Parity			
Risk Parity	5%	6.7%	0.3%
Inflation Expectations			2.2%
Alpha			1.0%
Total	**100%**		**8.7%**

**The Expected Contribution to Returns incorporates the volatility drag resulting from the conversion between Arithmetic and Geometric mean return.*

Discount Rate Sensitivity Analysis. The following schedule shows the impact of the Net Pension Liability if the discount rate used was 1% less than and 1% greater than the discount rate that was used (8%) in measuring the 2017 Net Pension Liability.

	1% Decrease in Discount Rate (7.0%)	Discount Rate (8.0%)	1% Increase in Discount Rate (9.0%)
WISD's Proportionate Share net pension liability:	$1,085,042	$643,635	$276,093

35

Quick Review: Table Extraction

level	page_num	block_num	par_num	line_num	word_num	left	top	width	height	conf	text
5	1	1	1	1	1	347	266	84	28	96	Asset
5	1	1	1	1	2	442	265	79	28	96	Class
5	1	1	1	1	3	901	237	93	68	50	Varget
5	1	1	1	1	4	1169	238	66	28	93	Real
5	1	1	1	1	5	1247	239	100	27	93	Return
5	1	1	1	1	6	1360	233	157	68	87	Geometric
5	1	1	1	1	9	1853	207	54	92	84	Rete
5	1	1	1	1	10	1914	211	47	76	84	of
5	1	1	1	2	1	871	283	157	45	87	Allocation
5	1	1	1	2	2	1301	286	81	28	91	Basis
5	1	1	1	2	3	1733	307	118	28	83	Return*
5	1	1	2	1	1	190	370	104	29	91	Global
5	1	1	2	1	2	307	370	105	36	96	Equity
5	1	1	3	1	1	217	431	64	28	65	US.

Quick Review: Table Extraction

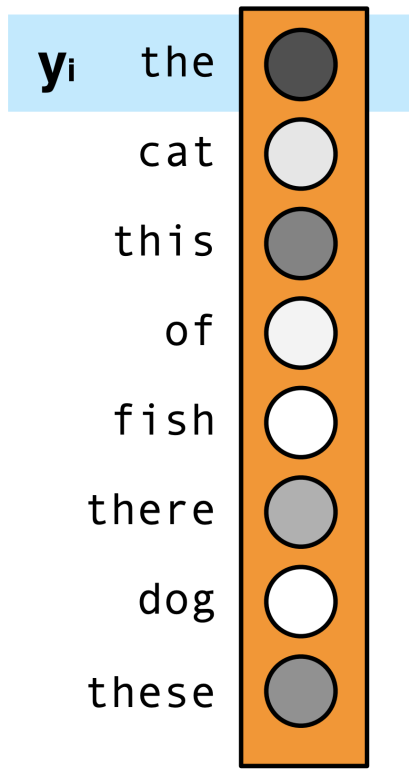
Asset Class	Varget	Real Return Geometric		Rete of
	Allocation	Basis	Return*	
Global Equity				
US.	18%	4.6%	1.0%	
Non-U.S. Developed	13%	5.1%	0.8%	
Emerging Markets	9%	5.9%	0.7%	
Directional Hedge Funds	4%	3.2%	1%	
Private Equity	13%	7.0%	11%	
Stable Value				
U.S. Treasuries	11%	0.7%	0.1%	
Absolute Return	0%	1.8%	0.0%	
Hedge Funds (Stable Value)	4%	3.0%	0.1%	
Cash	1%	-0.2%	0.0%	
Real Return				
Global Infation Linked Bonds	3%	0.9%	0.0%	
Real Assets	16%	5.1%	1.1%	
Energy and Natura] Resources	3%	6.6%	2.0%	
Commodities	0%	1.2%	0.2%	
Risk Parity				
Risk Parity	5%	6.7%	0.3%	
Inflation Expectations			2.2%	
Alpha			1.0%	
	100%		8.7%	

Decoding: Making better Translations

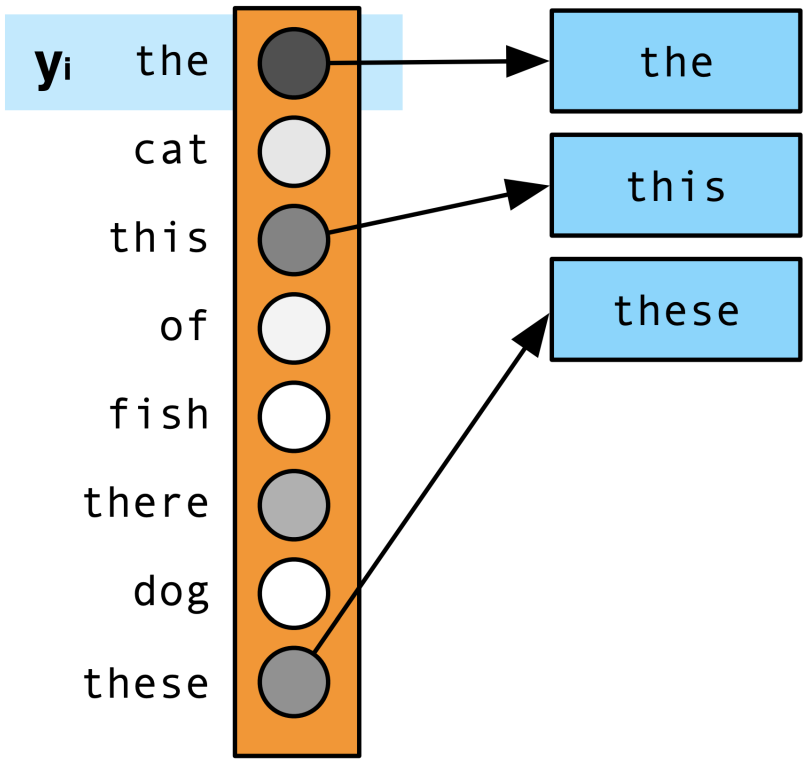
Beam Search

- Beam Search is a technique to explore different translation paths
- Based on the idea that you **shouldn't take the greedy path**
- And that the best path may be sub-optimal during early iterations
- How to determine what's optimal? **N-Gram Lanugage Model**

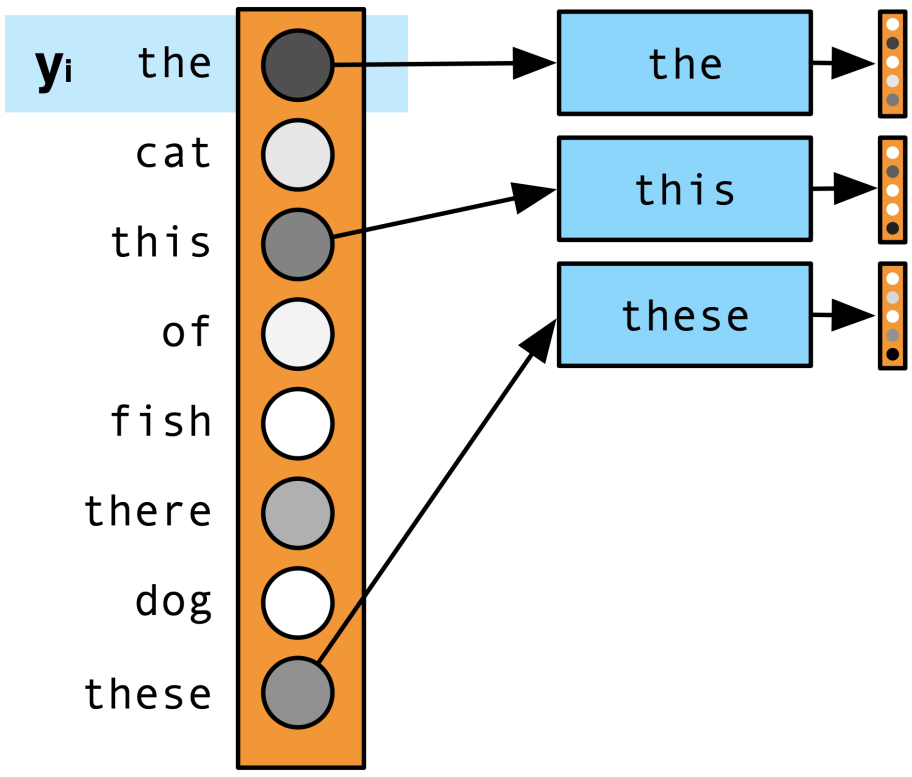
Beam Search



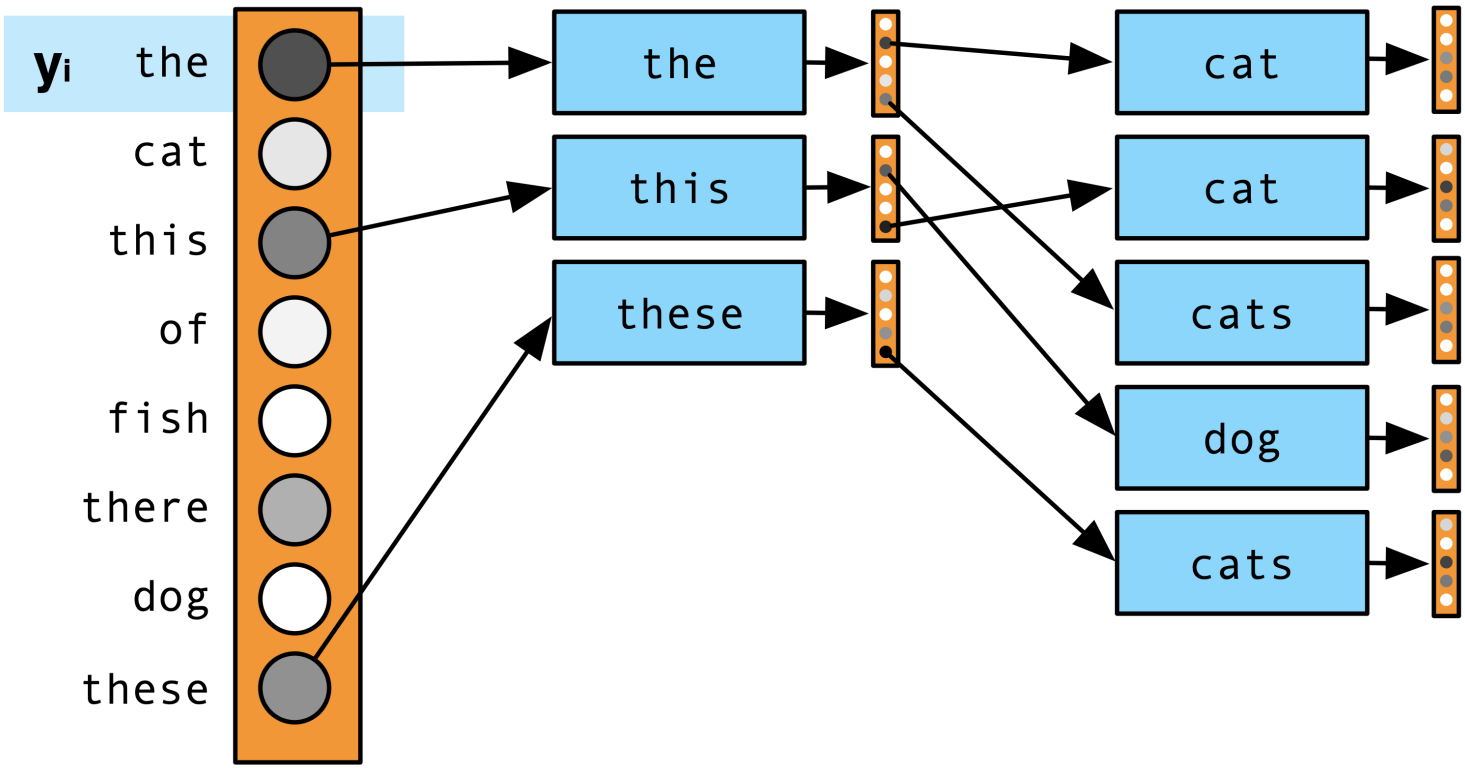
Beam Search



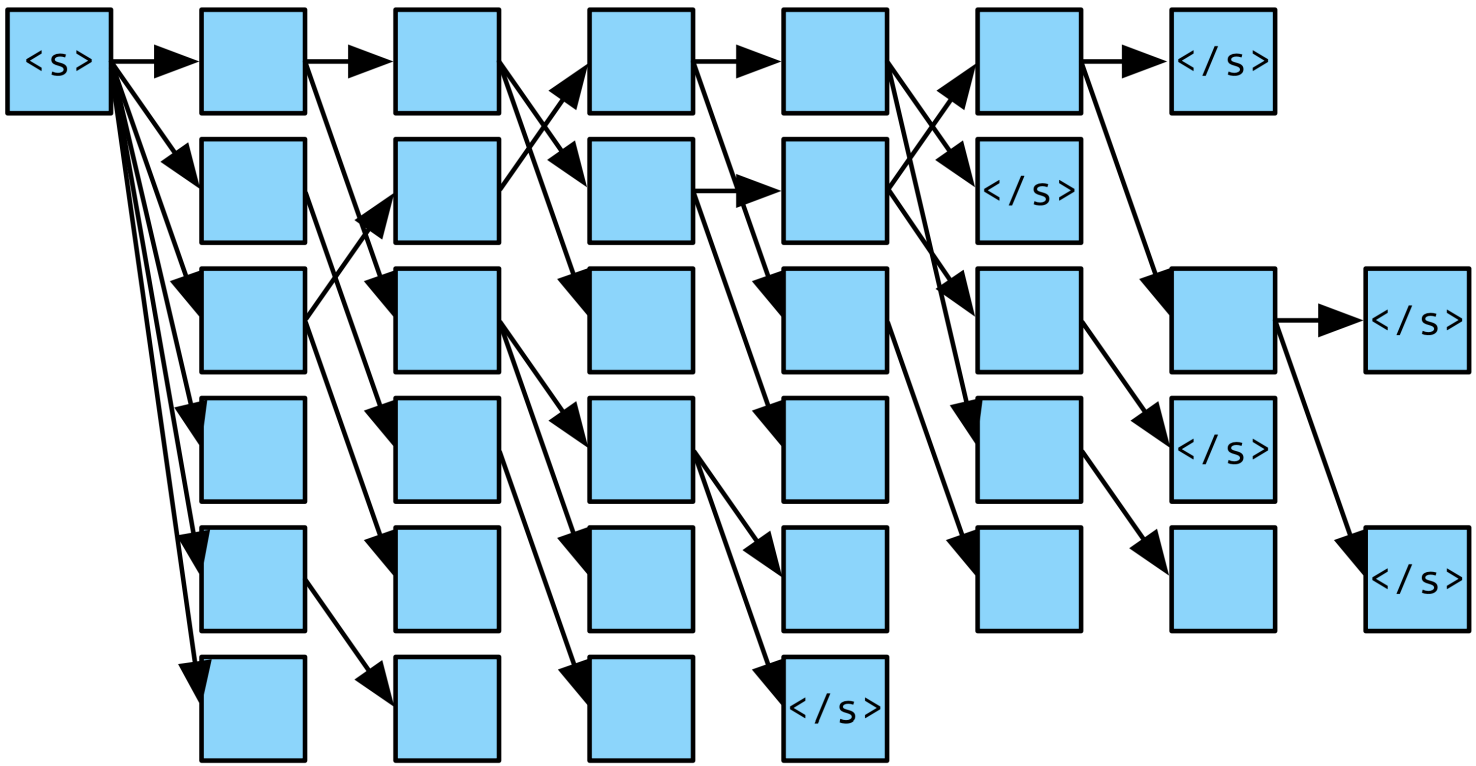
Beam Search



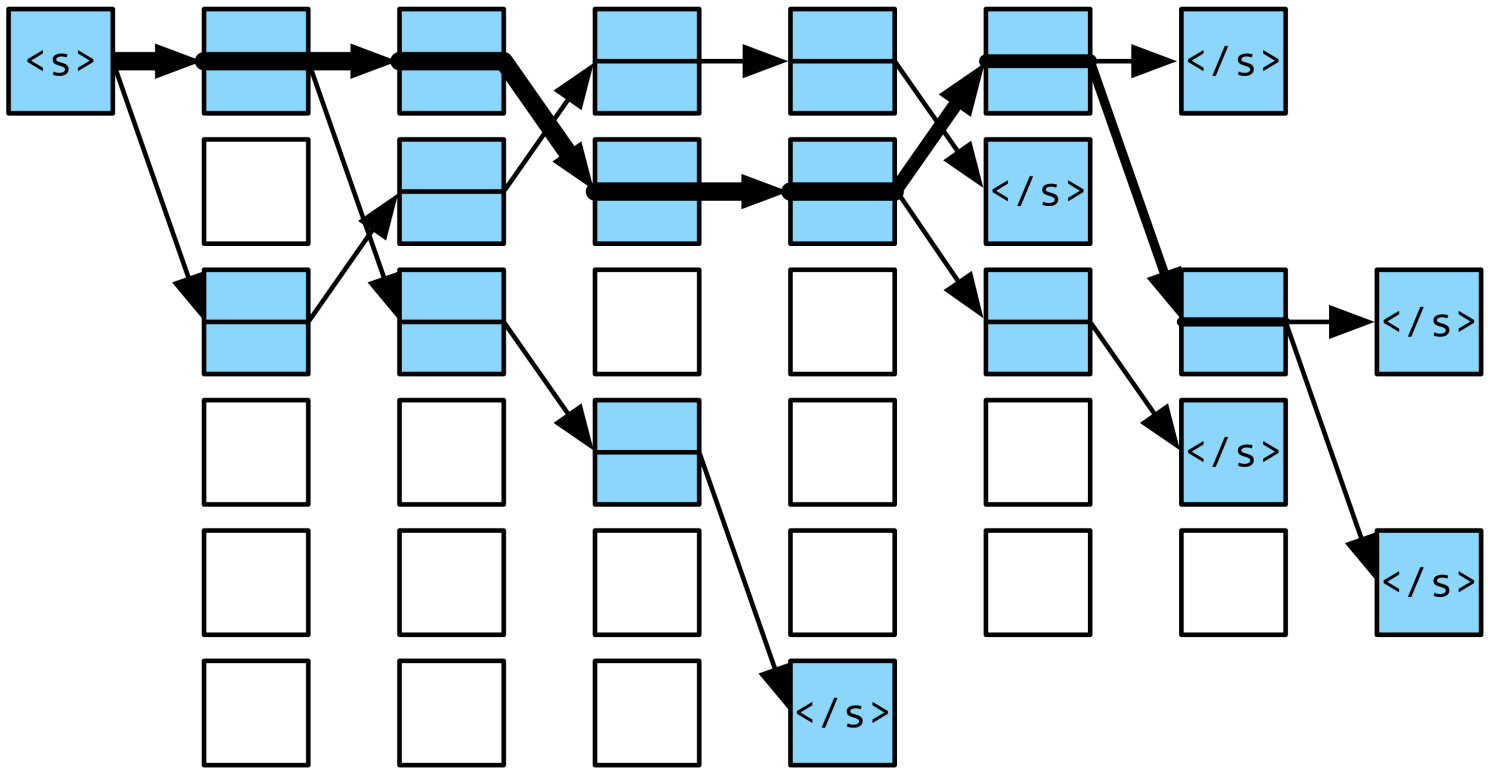
Beam Search


y_i
the
cat
this
of
fish
there
dog
these
the
this
these
cat
cat
cats
dog
cats

Beam Search

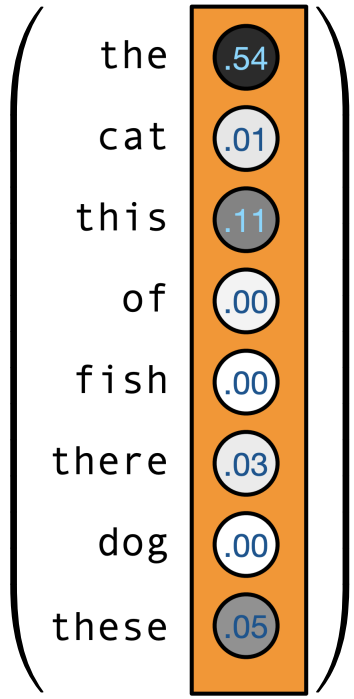


Beam Search



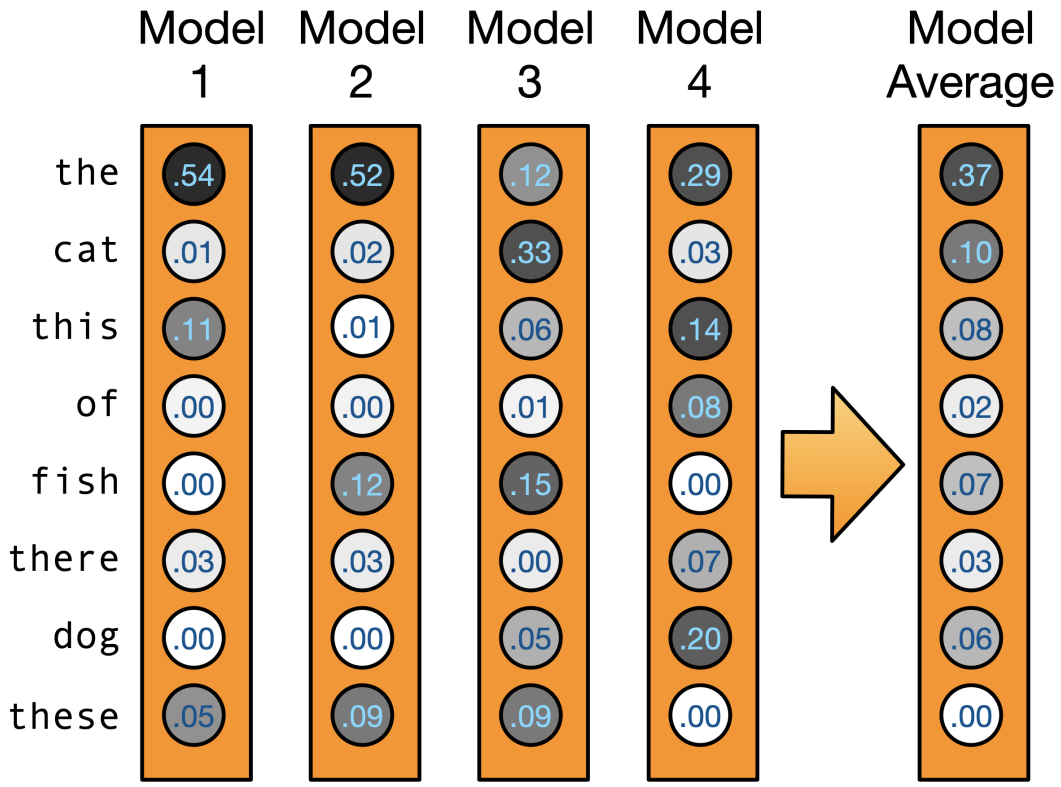
Monte Carlo Beam Search

$$y_{i+1} \sim \text{Multinomial}$$



- Why not sample n words based on their probabilities?
- Adds more diversity to beam search results

Ensembling

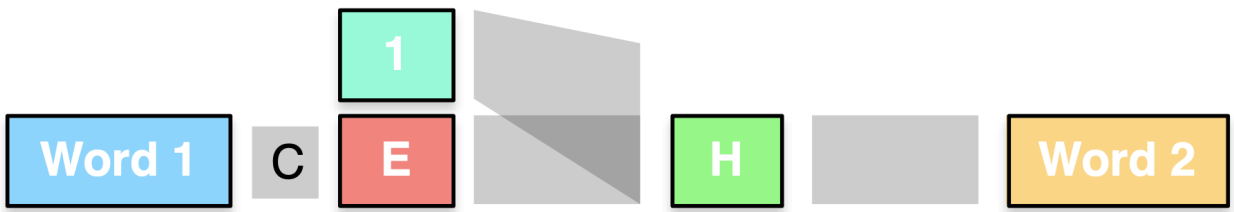


- Why not average different models?
- Random initialization leads to different local solutions
- Could also use model dumps from different iterations

Modeling Recurrent Relations with RNNs

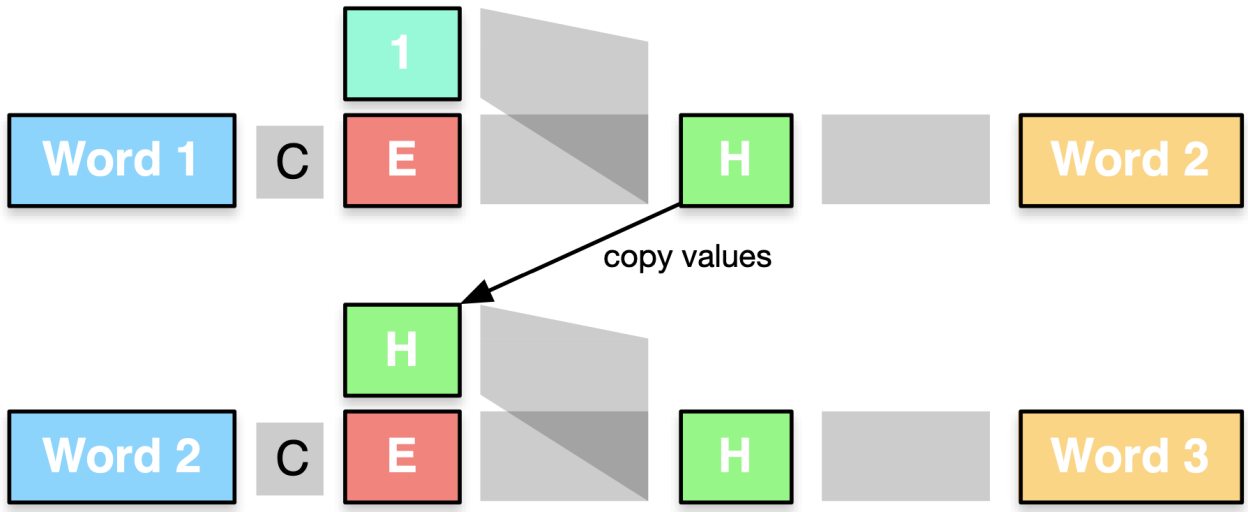
RNN Forward Propagation

- **Concept:** Allow current sample to mix with previous information
- Learn from the pattern's history



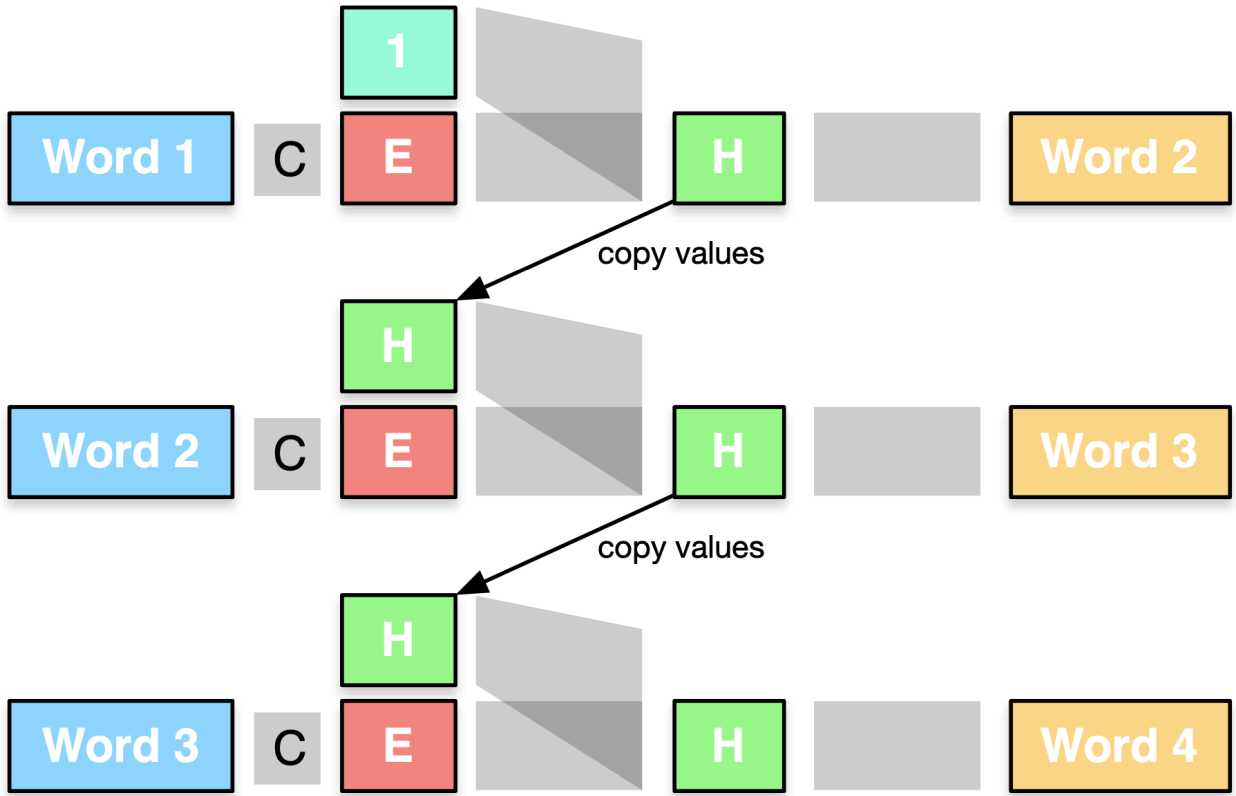
RNN Forward Propagation

- We can unfold the computation graph



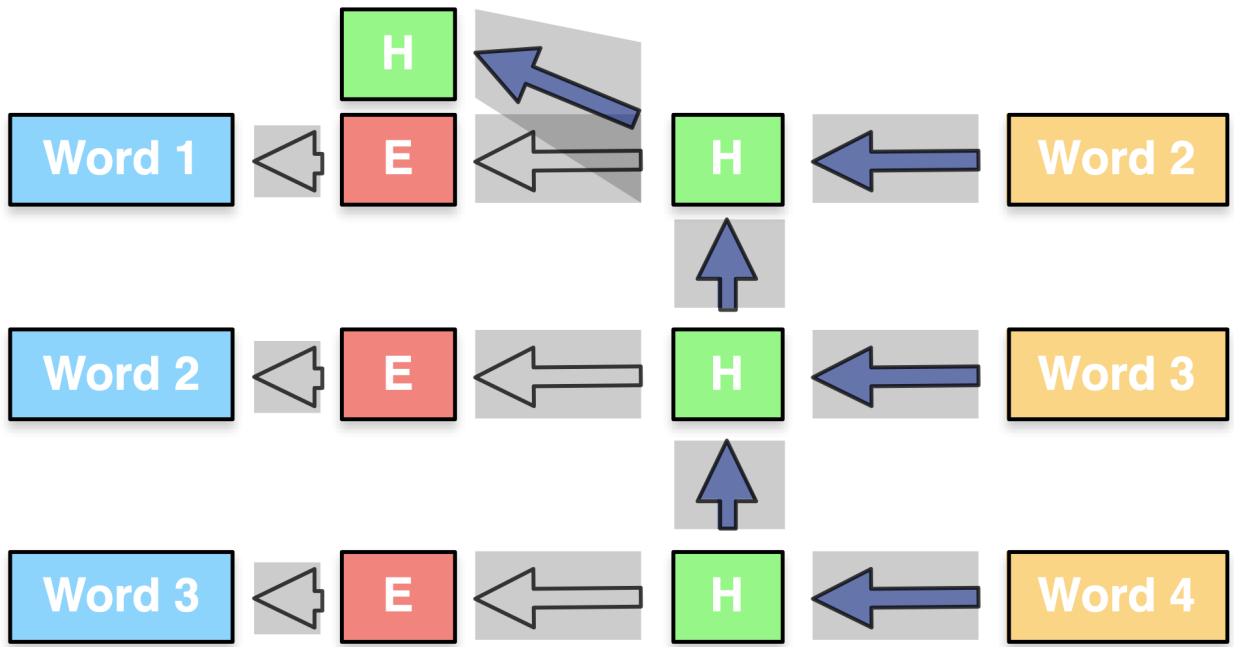
RNN Forward Propagation

- Notice how each cell output is dependent on all prior computations



Back-Propagation Through Time

- Hence gradients flow through the entire unfolded graph
- Note: we reuse the cell, so the parameters are the same at each time step



Vanilla RNNs

$$h_t = \tanh\left(\overbrace{W_{ih}x_t + b_{ih}}^{input} + \underbrace{W_{hh}h_{t-1} + b_{hh}}_{hidden}\right)$$

- h_t is the hidden state at time t
- x_t is the input at time t
- h_{t-1} is the previous hidden state
- h_0 is initialized to $\mathbf{0}$

Long Short Term Memory (LSTM)

$$
\begin{aligned}
\text{Gates} \rightarrow & \begin{cases}
i_t = \sigma(W_{ii} x_t + b_{ii} + W_{hi} h_{t-1} + b_{hi}) \\
f_t = \sigma(W_{if} x_t + b_{if} + W_{hf} h_{t-1} + b_{hf}) \\
o_t = \sigma(W_{io} x_t + b_{io} + W_{ho} h_{t-1} + b_{ho}) \\
g_t = \tanh(W_{ig} x_t + b_{ig} + W_{hg} h_{t-1} + b_{hg})
\end{cases} \\
\text{Outputs} \rightarrow & \begin{cases}
c_t = f_t \odot c_{t-1} + i_t \odot g_t \\
h_t = o_t \odot \tanh(c_t)
\end{cases}
\end{aligned}
$$

- h_t is the hidden state at time t
- c_t is the cell state at time t
- x_t is the input at time t

Gated Recurrent Units (GRU)

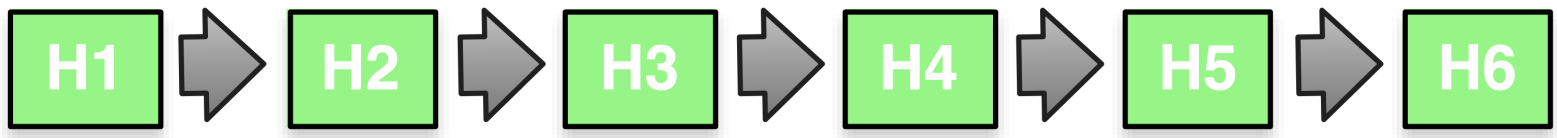
$$\text{Gates} \rightarrow \begin{cases} r_t = \sigma(W_{ir} x_t + b_{ir} + W_{hr} h_{(t-1)} + b_{hr}) \\ z_t = \sigma(W_{iz} x_t + b_{iz} + W_{hz} h_{(t-1)} + b_{hz}) \\ n_t = \tanh(W_{in} x_t + b_{in} + r_t \odot (W_{hn} h_{(t-1)} + b_{hn})) \end{cases}$$

$$\text{Outputs} \rightarrow \begin{cases} h_t = (1 - z_t) \odot n_t + z_t \odot h_{(t-1)} \end{cases}$$

- h_t is the hidden state at time t
- x_t is the input at time t

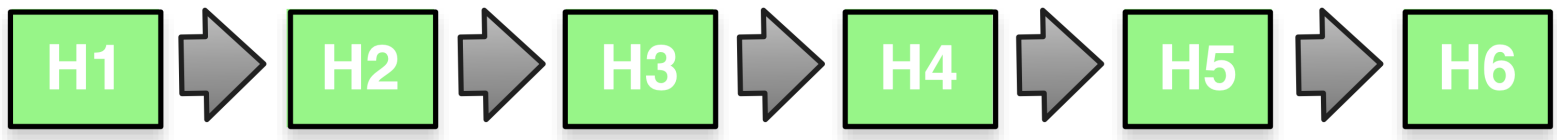
Aside: Different Perspectives on Deep Recurrent Models

- So far we've only seen Left to Right Sequencing

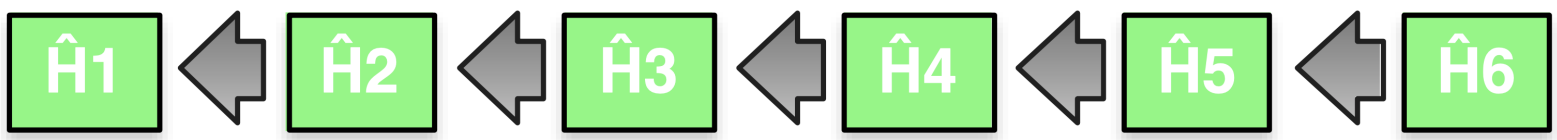


Aside: Different Perspectives on Deep Recurrent Models

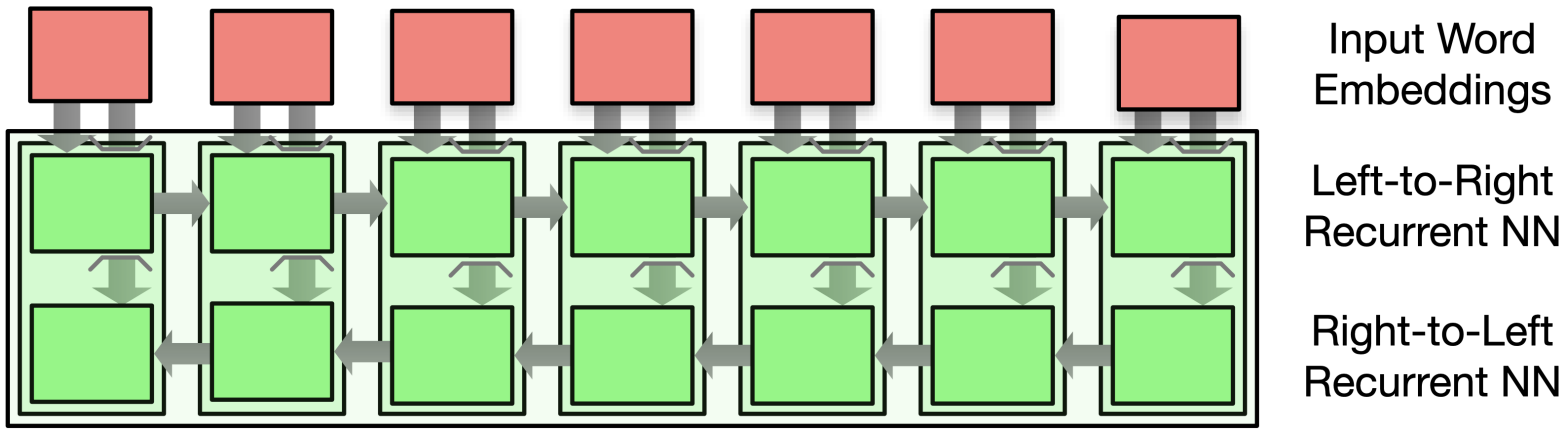
- So far we've only seen Left to Right Sequencing



- Why not Right to Left?

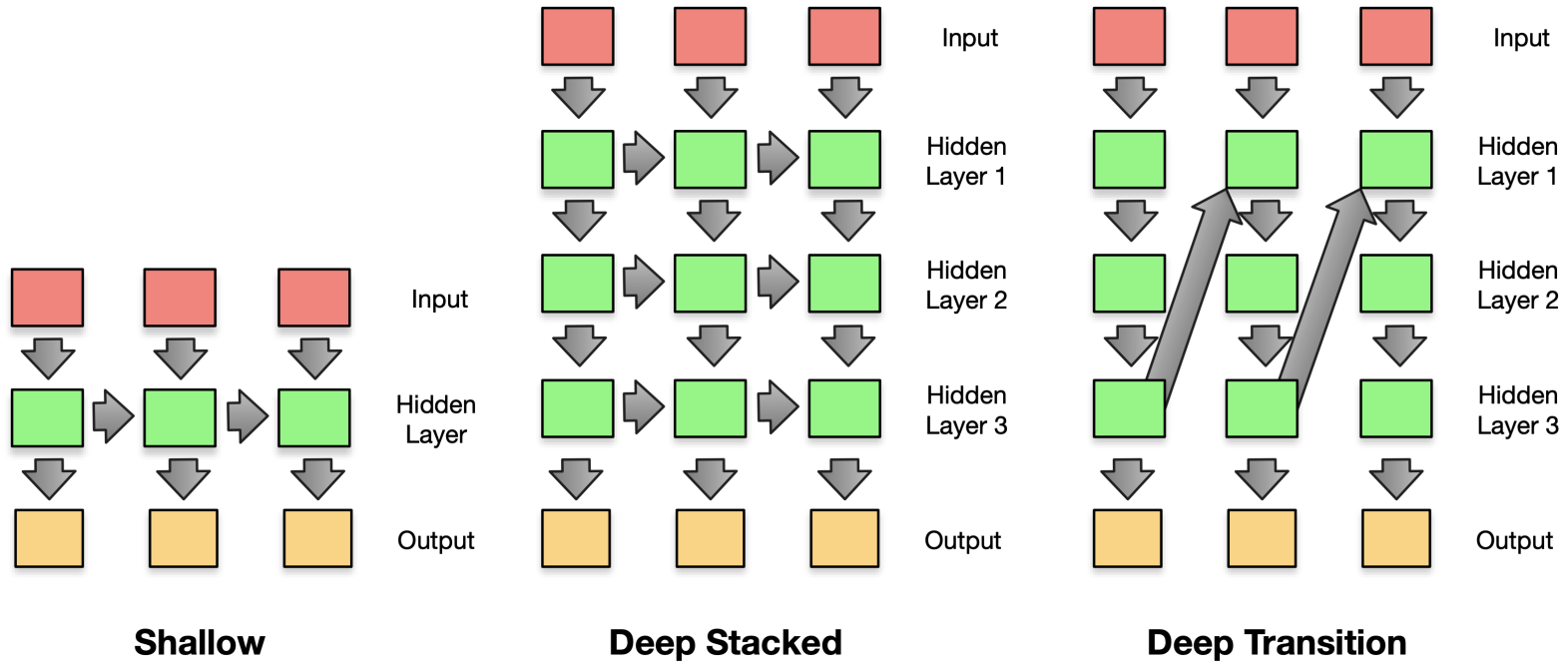


Bidirectional Sequence Modeling



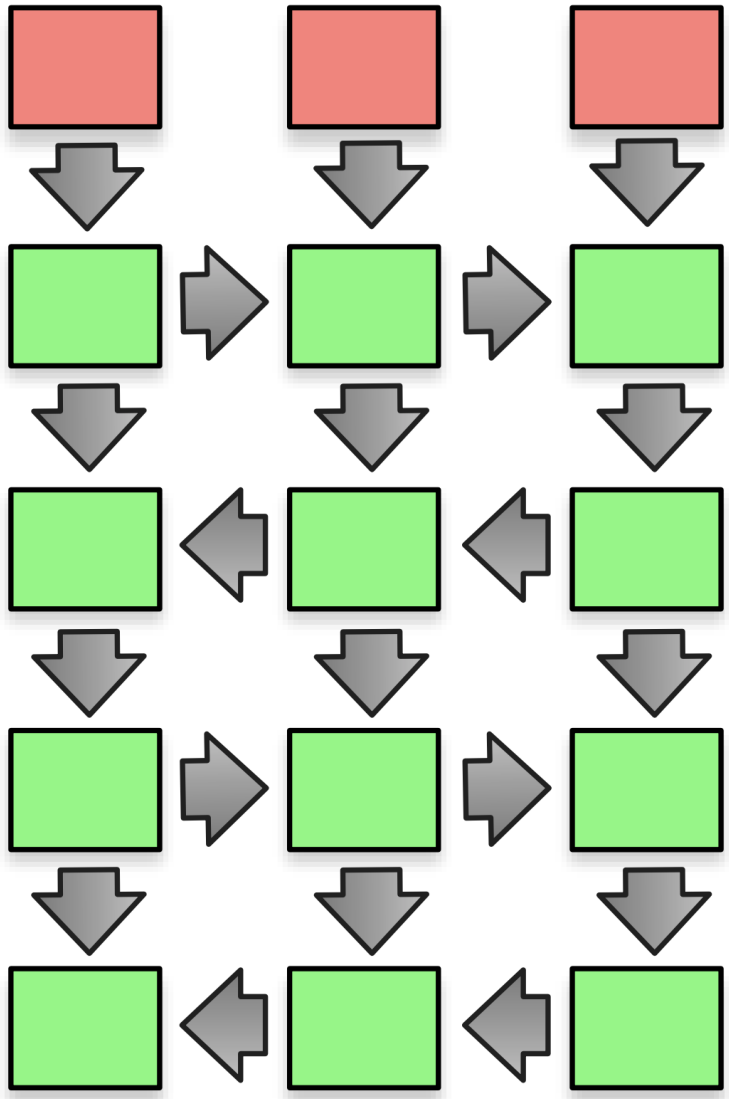
- Can capture both left and right context
- Implementation usually concatenates RNN states

Aside: Different Perspectives on Deep Recurrent Models



- Experiment with different stacking techniques

Alternating Recurrent Directions



Aside: Dimensionality of Inputs and Outputs (Batch First)

Type	RNN	LSTM	GRU
In	B, L, H_{in}	B, L, H_{in}	B, L, H_{in}
h_{t-1}	$N_L \cdot N_D, B, H_{out}$	$N_L \cdot N_D, B, H_{out}$	$N_L \cdot N_D, B, H_{out}$
c_{t-1}	-	$N_L \cdot N_D, B, H_{out}$	-
h_t	$N_L \cdot N_D, B, H_{out}$	$N_L \cdot N_D, B, H_{out}$	$N_L \cdot N_D, B, H_{out}$
c_t	-	$N_L \cdot N_D, B, H_{out}$	-
Out	$B, L, N_D \cdot H_{out}$	$B, L, N_D \cdot H_{out}$	$B, L, N_D \cdot H_{out}$

- B is the batch size
- L is the sequence length
- N_D is the number of directions
- N_L is the number of layers
- H_{in}, H_{out} are the input and hidden size

Aside: The Influence of Padding in RNNs

- Assume the embedding $E[\langle \text{PAD} \rangle] = \vec{0}$
- Are we safe?

Aside: The Influence of Padding in RNNs

- Assume the embedding $E[\langle \text{PAD} \rangle] = \vec{0}$
- Are we safe? **NO!**
- Because of the bias term, zero input does not result in a zero output
  - This alters the hidden state being passed onto the next iteration
  - So be carful if taking the last output of a sequence with padding
  - Make sure that the reverse direction in bidirectionals are not computing padded states
- **Question:** Why does this matter?

Aside: The Influence of Padding in RNNs

- Assume the embedding $E[\langle \text{PAD} \rangle] = \vec{0}$
- Are we safe? **NO!**
- Because of the bias term, zero input does not result in a zero output
  - This alters the hidden state being passed onto the next iteration
  - So be carful if taking the last output of a sequence with padding
  - Make sure that the reverse direction in bidirectionals are not computing padded states
- **Question:** Why does this matter?
- **Answer:** RNNs assume $\vec{0}$ initial hidden state

Aside: The Influence of Padding in RNNs

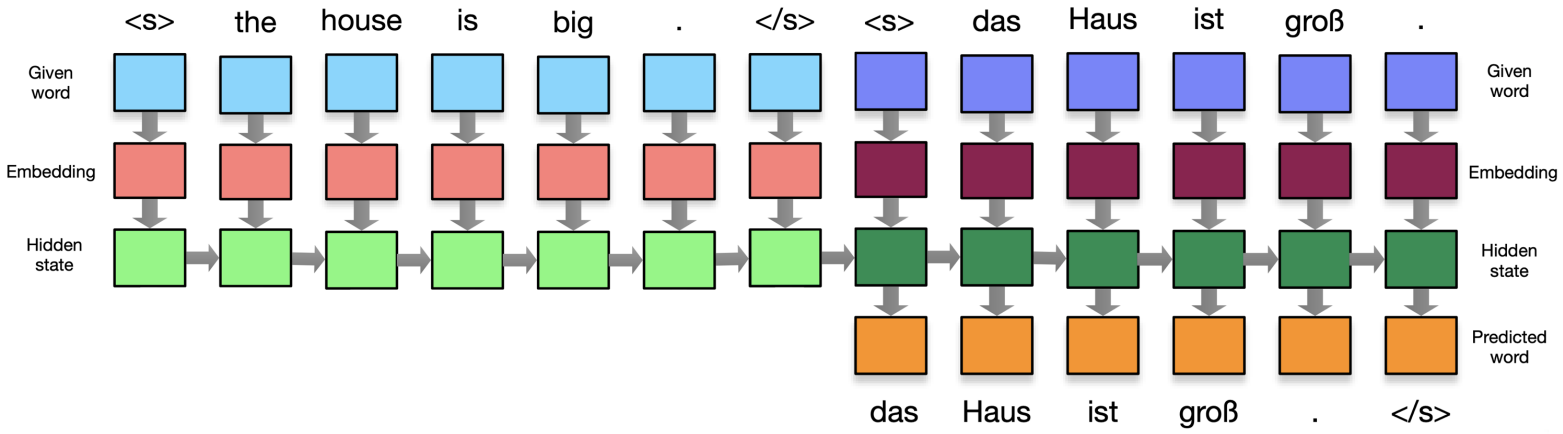
- Assume the embedding $E[\langle \text{PAD} \rangle] = \vec{0}$
- Are we safe? **NO!**
- Because of the bias term, zero input does not result in a zero output
  - This alters the hidden state being passed onto the next iteration
  - So be carful if taking the last output of a sequence with padding
  - Make sure that the reverse direction in bidirectionals are not computing padded states
- **Question:** Why does this matter?
- **Answer:** RNNs assume $\vec{0}$ initial hidden state
- **Question:** Does this mean we learn the amount of padding for a given sequence?

Aside: The Influence of Padding in RNNs

- Assume the embedding $E[\langle \text{PAD} \rangle] = \vec{0}$
- Are we safe? **NO!**
- Because of the bias term, zero input does not result in a zero output
  - This alters the hidden state being passed onto the next iteration
  - So be carful if taking the last output of a sequence with padding
  - Make sure that the reverse direction in bidirectionals are not computing padded states
- **Question:** Why does this matter?
- **Answer:** RNNs assume $\vec{0}$ initial hidden state
- **Question:** Does this mean we learn the amount of padding for a given sequence?
- **Answer:** Most likely, since gradients now incorporate repeated application of activation functions for the bias term.

Applications to Column Segmentation

Applications to Column Segmentation




```
Seq2Seq(
  (encoder): Encoder(
    (embedding): Embedding(21, 256)
    (gru): GRU(256, 128, num_layers=2, batch_first=True)
  )
  (decoder): Decoder(
    (embedding): Embedding(22, 256)
    (gru): GRU(256, 128, num_layers=2, batch_first=True)
    (fc): Linear(in_features=128, out_features=22, bias=True)
  )
)
The model has 508,438 trainable parameters
```

Column Segmentation: Merge the Encoder and Decoder

- Just use the encoder's embeddings and recurrent layers, with an decoding output layer
- For every token, predict if we should insert a ⟨SEG⟩
- No longer have to learn the actual sequence

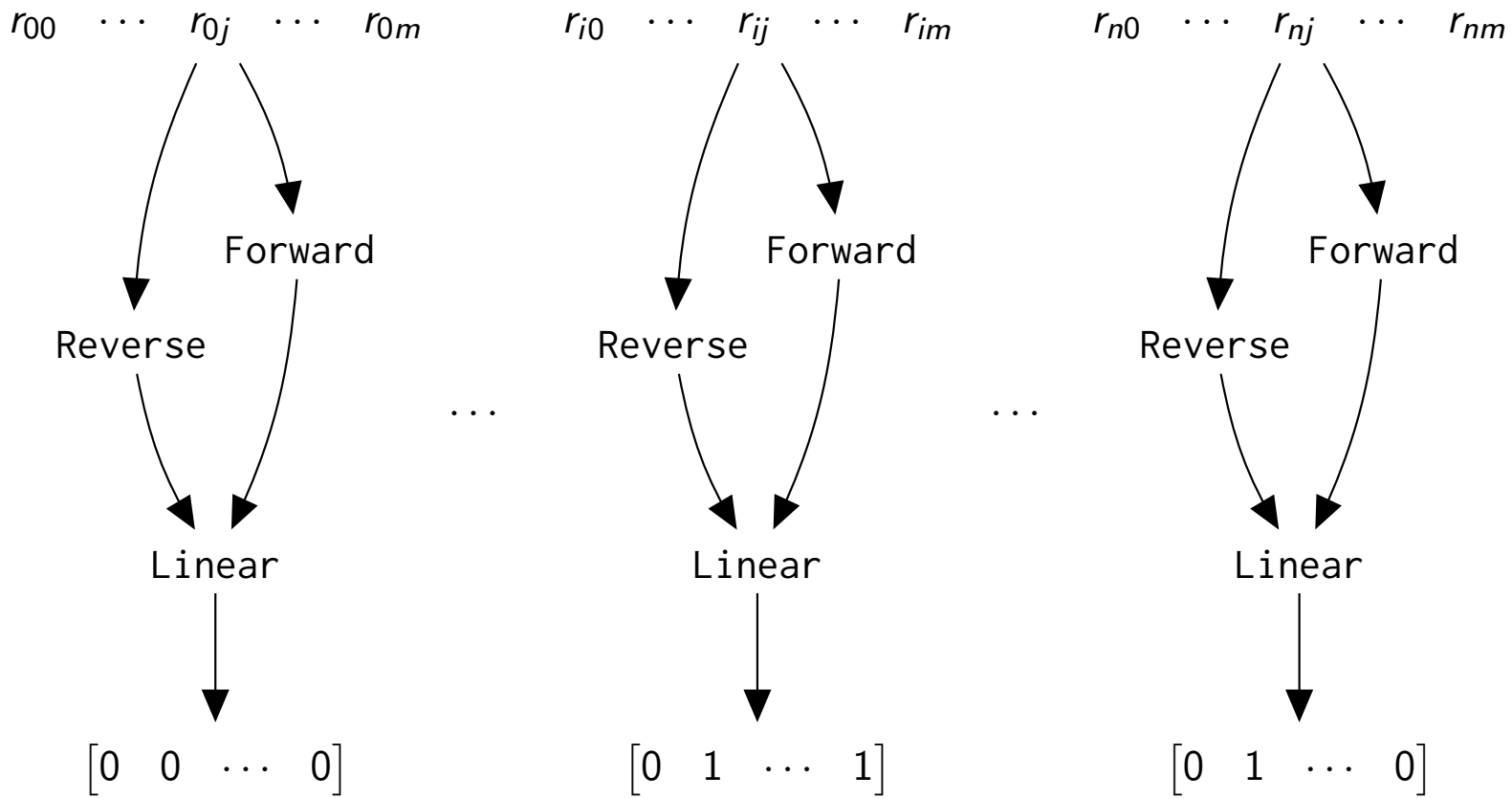
```
Classifier(
  (encoder): Encoder(
    (embedding): Embedding(21, 128)
    (gru): GRU(128, 128, batch_first=True, bidirectional=True)
  )
  (linear): Linear(in_features=256, out_features=1, bias=True)
)
The model has 201,089 trainable parameters
```

Independent Row: Training Labels

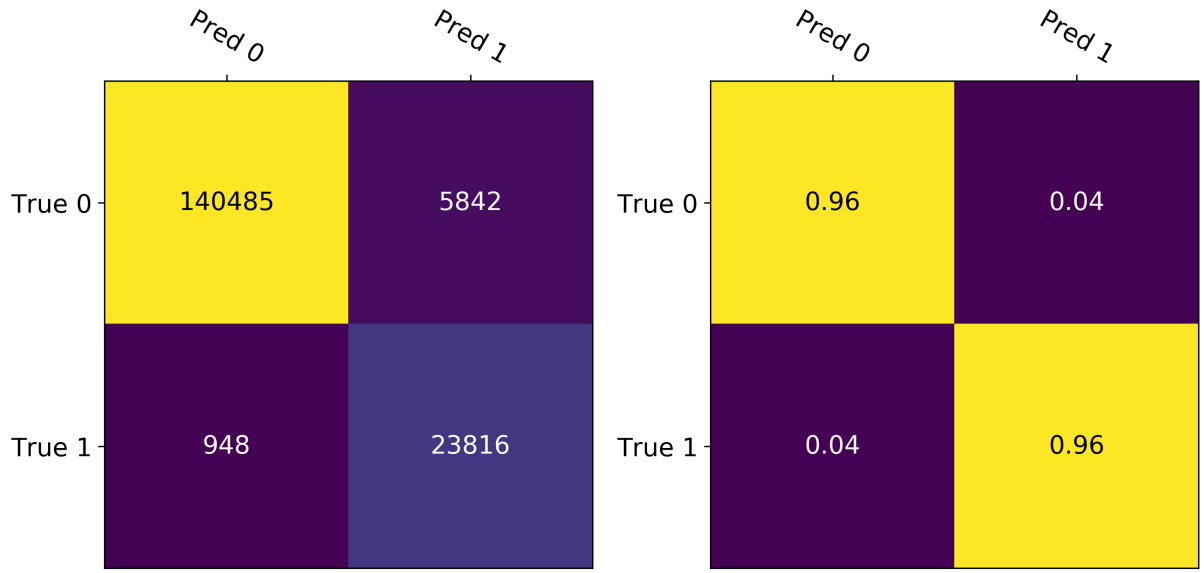
- Sequences are table rows
- Labels are now binary vectors, where $x_i = 1$ implies that there is a ⟨SEG⟩ between x_i and x_{i+1}

Less	imputed	interest		(	174,862.64	)
0	0	1		0	0	0
Less	imputed	interest	⟨SEG⟩	(	174,862.64	)

Independent Row Training Visual

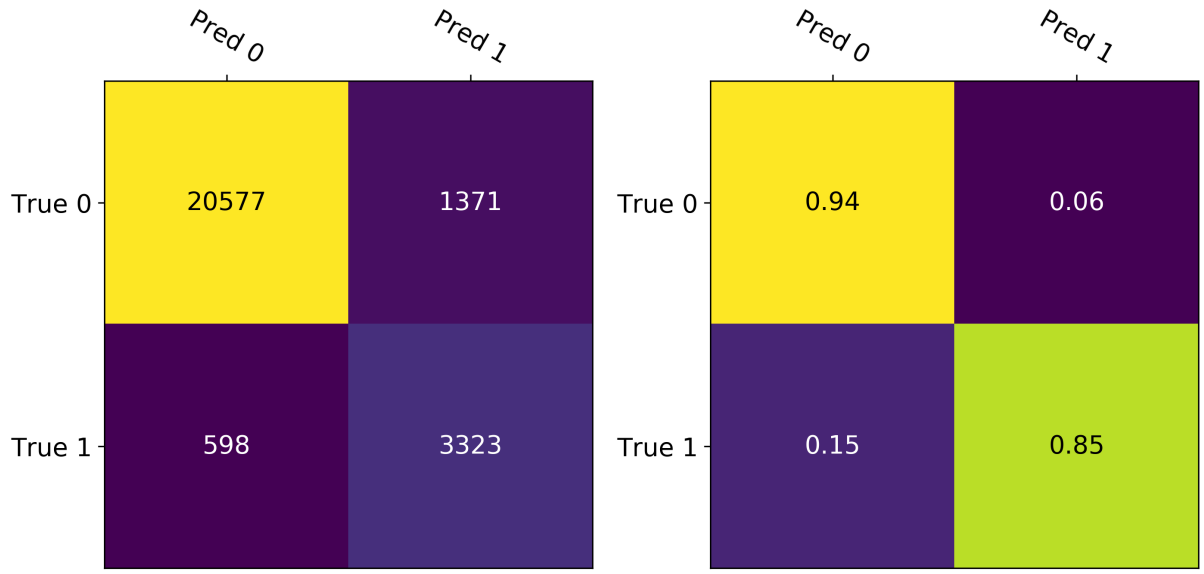


Independent Row: Sample Results (Training)



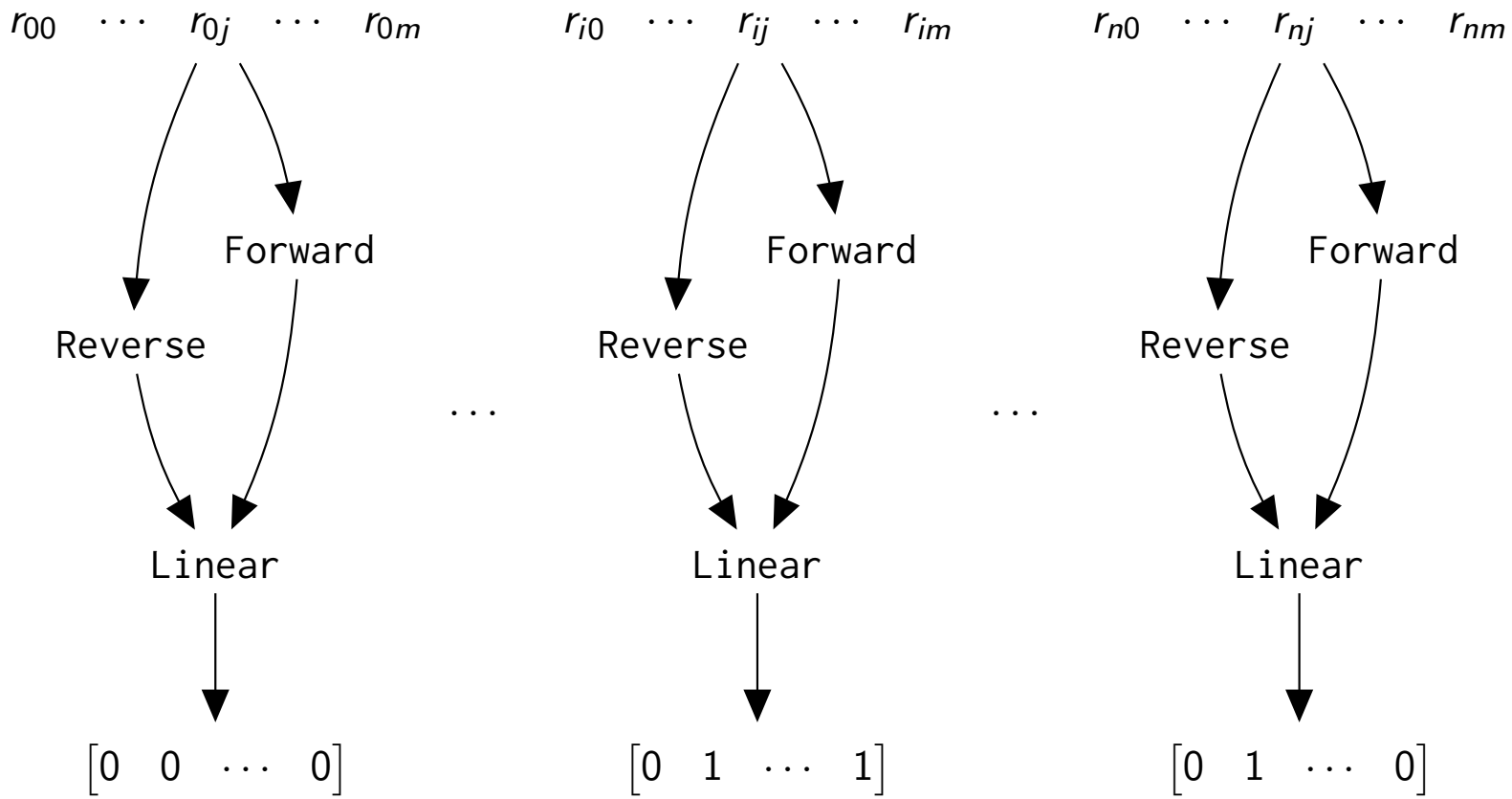
	Precision	Recall	F1-Score	Support
Label: 0	0.99	0.96	0.98	146,327
Label: 1	0.80	0.96	0.88	24,764
accuracy			0.96	171,091
macro avg	0.90	0.96	0.93	171,091
weighted avg	0.97	0.96	0.96	171,091

Independent Row: Sample Results (Validation)



	Precision	Recall	F1-Score	Support
Label: 0	0.97	0.94	0.95	21,948
Label: 1	0.71	0.85	0.77	3,921
accuracy			0.92	25,869
macro avg	0.84	0.89	0.86	25,869
weighted avg	0.93	0.92	0.93	25,869

How can we improve on this?

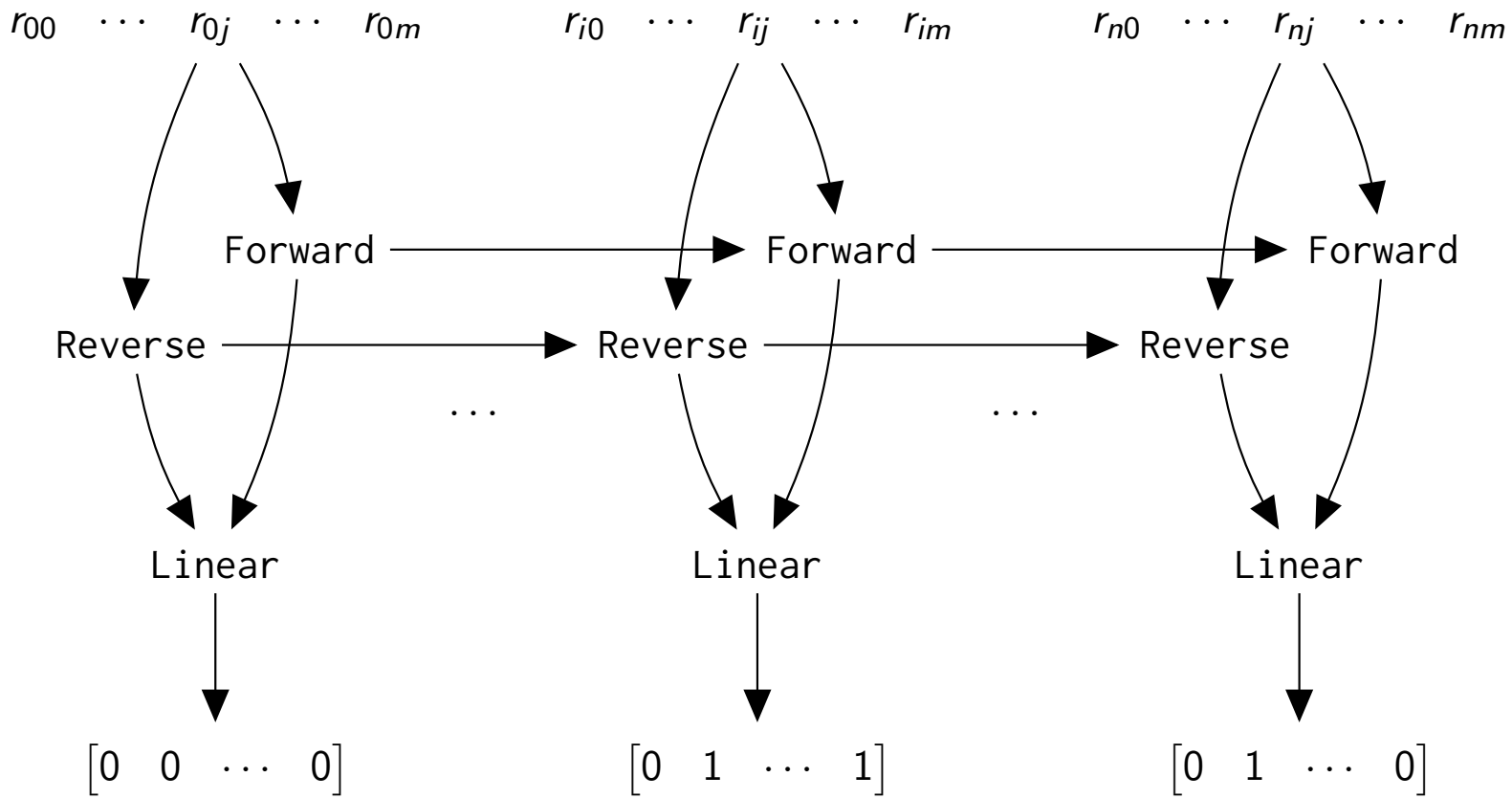


Joint Training on a Table

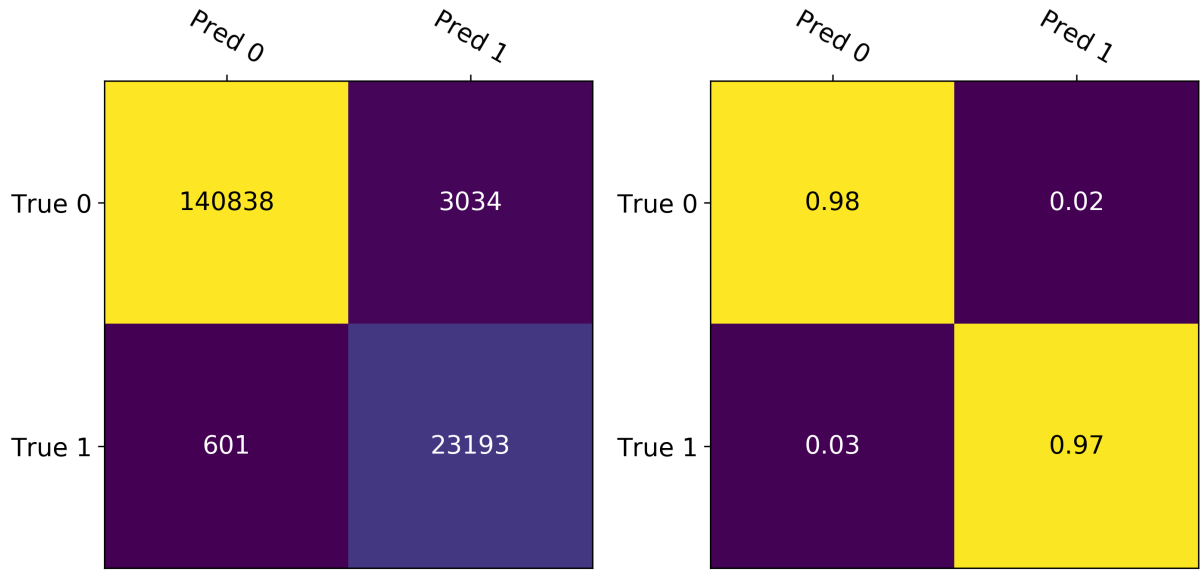
- We saw that we could regenerate a sequence from its hidden state alone.
- Why not use this information to link rows within the same table?
  - Bidirectional for each row
  - Propagate hidden state to next row
  - Jointly train over an entire table

```
TableClassifier(
  (embedding): Embedding(21, 128)
  (rnn): GRU(128, 128, batch_first=True, bidirectional=True)
  (ff): Linear(in_features=256, out_features=1, bias=True)
)
The model has 201,089 trainable parameters
```

Joint Table Training Visual

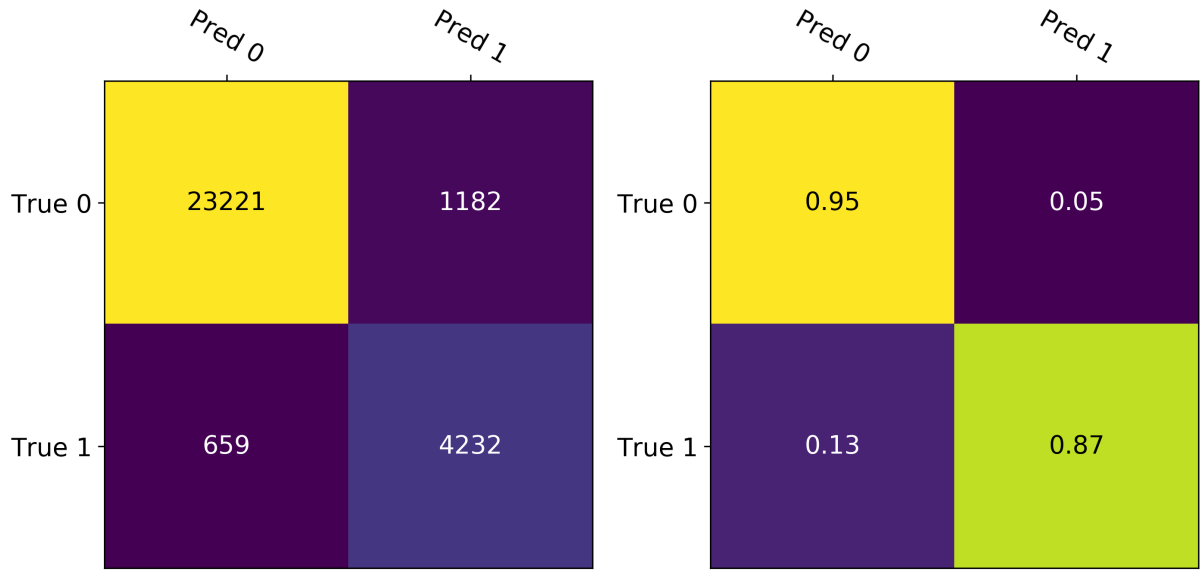


Joint Table Training: Sample Results (Training)



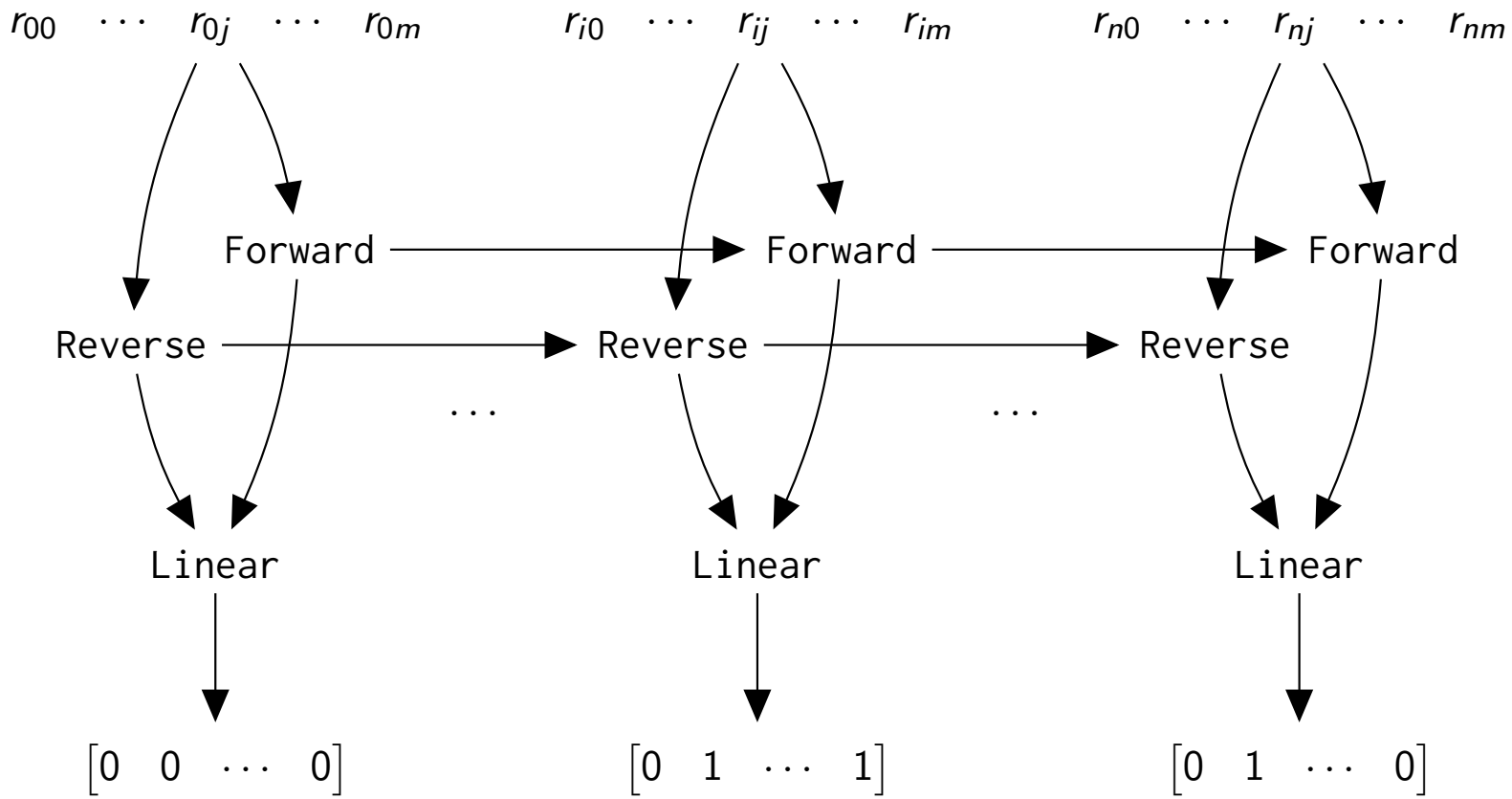
	Precision	Recall	F1-Score	Support
Label: 0	1.00	0.98	0.99	143,872
Label: 1	0.88	0.97	0.93	23,794
accuracy			0.98	167,666
macro avg	0.94	0.98	0.96	167,666
weighted avg	0.98	0.98	0.98	167,666

Joint Table Training: Sample Results (Validation)

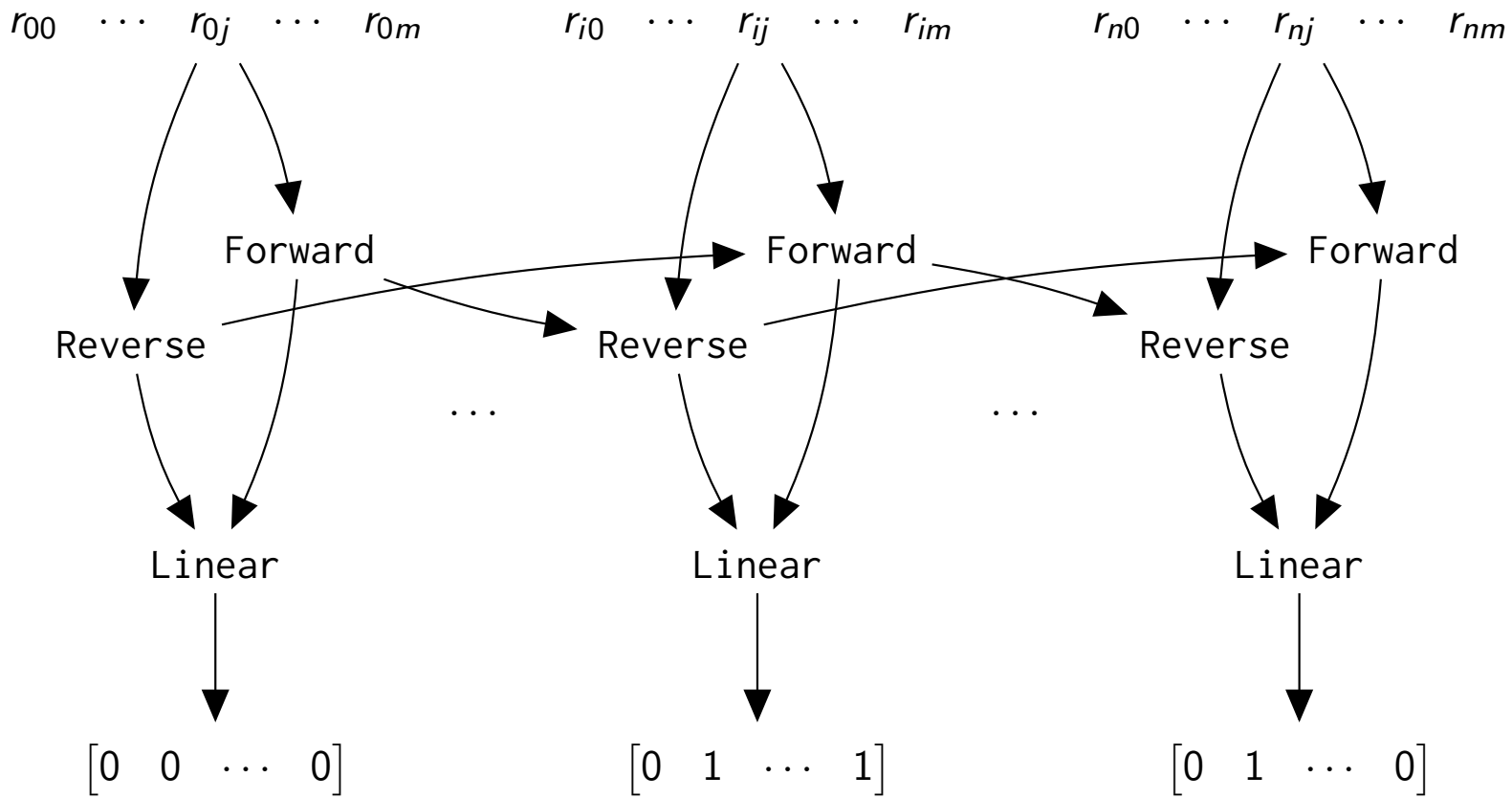


	Precision	Recall	F1-Score	Support
Label: 0	0.97	0.95	0.96	24,403
Label: 1	0.78	0.87	0.82	4,891
accuracy			0.94	29,294
macro avg	0.88	0.91	0.89	29,294
weighted avg	0.94	0.94	0.94	29,294

Is this a correct way to link rows?



Joint Table Training with Swapping Visual

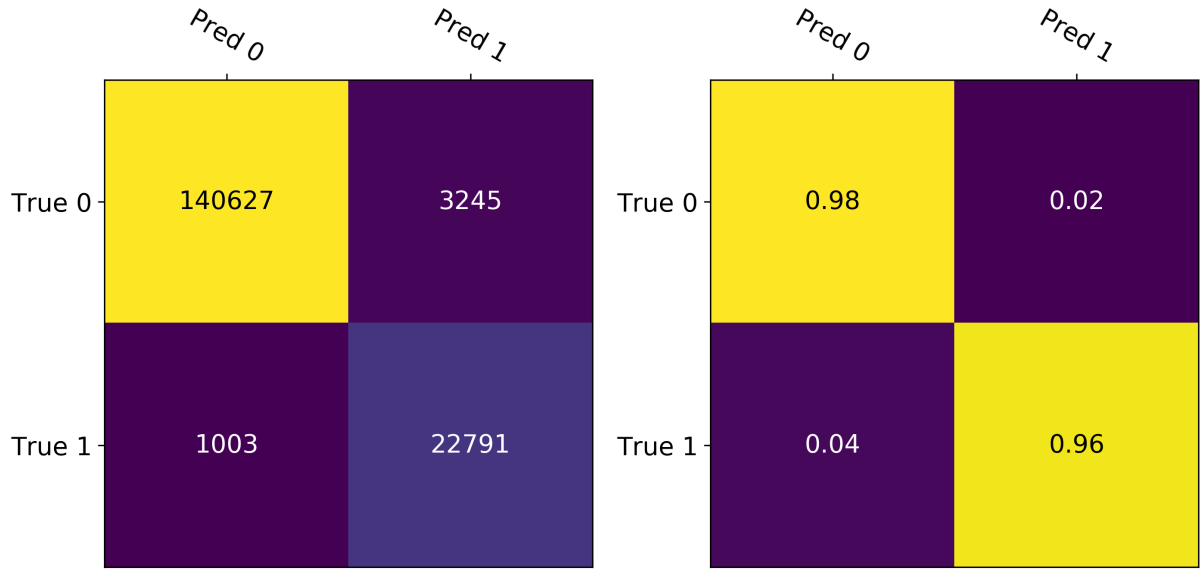


Trick for Swapping in Joint Table Training

- Note that the hidden state is biased towards the items closest to it
- So the beginning of a new row has context from the end of the last row
  - Is this a good choice?
  - What if we flip the bidirectional hidden states every iteration
  - AKA the reverse direction has the immediate context of the end of the row
  - And vice-versa

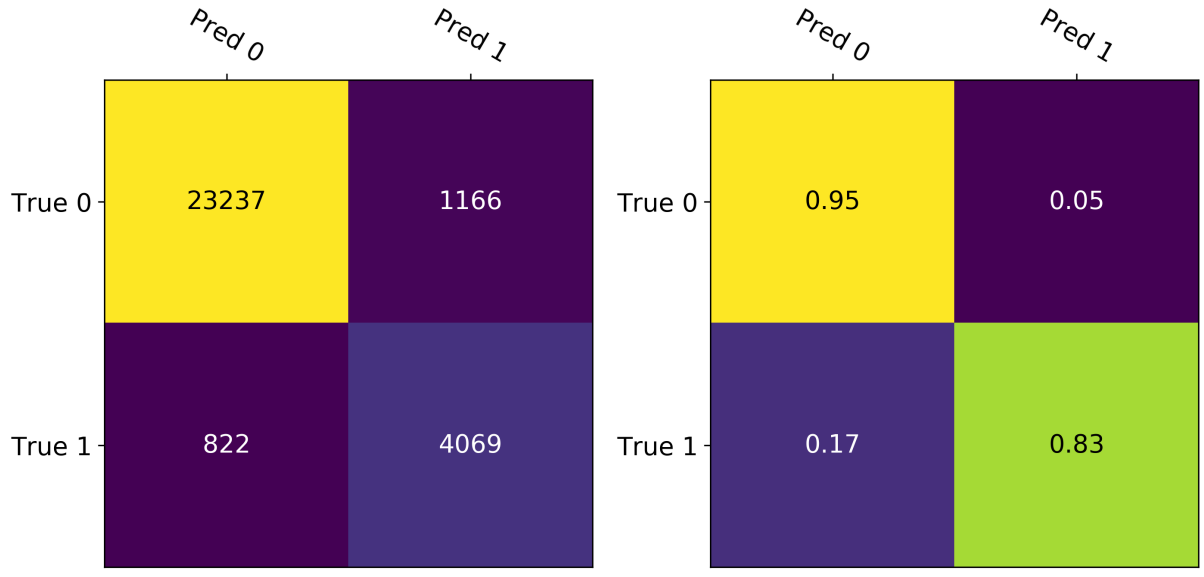
```
TableClassifierSwap(
  (embedding): Embedding(21, 128)
  (rnn): GRU(128, 128, batch_first=True, bidirectional=True)
  (ff): Linear(in_features=256, out_features=1, bias=True)
)
The model has 201,089 trainable parameters
```

Joint Table Training with Swapping: Sample Results (Training)



	Precision	Recall	F1-Score	Support
Label: 0	0.99	0.98	0.99	143,872
Label: 1	0.88	0.96	0.91	23,794
accuracy			0.97	167,666
macro avg	0.93	0.97	0.95	167,666
weighted avg	0.98	0.97	0.98	167,666

Joint Table Training with Swapping: Sample Results (Validation)



	Precision	Recall	F1-Score	Support
Label: 0	0.97	0.95	0.96	24,403
Label: 1	0.78	0.83	0.80	4,891
accuracy			0.93	29,294
macro avg	0.87	0.89	0.88	29,294
weighted avg	0.93	0.93	0.93	29,294

Summary of Methods (Validation)

		Independent	Joint	Joint Swapped
Label: 0	Precision	**0.97**	**0.97**	**0.97**
	Recall	0.94	**0.95**	**0.95**
	F1-Score	0.95	**0.96**	**0.96**
Label: 1	Precision	0.71	**0.78**	**0.78**
	Recall	0.85	**0.87**	0.83
	F1-Score	0.77	**0.82**	0.80
	Accuracy	0.92	**0.94**	0.93

What have we learned?

- We can learn text segments purely from the sequence of Part-of-Speech tags
- It is useful to jointly train/predict on the table instead of assuming independence
- Is there any benefit to swapping hidden states during propagation?
- **Note:**
  - We use the concept of *deep transition* in Seq2Seq for joint training between rows
  - Our training exploits the finding that we can capture enough signal in the hidden state encoding

Possible Experiments

- Features
  - Use word location information
  - Tesseract gives bounding boxes
  - Visual cues
- Encoder Structure
  - Convolution
  - Transformer

Other Applications For Sequence Modeling

Recall: Encoders, Decoders, and Seq2Seq Models

- **Encoders** given a sequence meaning
- **Decoders** generate a new sequence
- **Seq2Seq** generate sequences conditioned on another sequence

What to use for which application?

- **Encoders**
  - POS Tagging
  - Sentence Embeddings
  - Anything where you are given the sentence at test time
- **Decoders**
  - Text Generation
  - Language Modeling
  - Anything where you need to create a sequence at test time
- **Seq2Seq**
  - Translation
  - Speech Recognition
  - Summarization
  - Question/Answering
  - Anything where you convert a sequence into another sequence

Tools, References, and Further Reading

Acknowledgment

- Pieces adapted from my college Machine Translation Course by Philipp Koehn
- Additional acknowledgment:
  - Adam Lopez
  - Matt Post
  - Chris Callison-Burch
  - Philipp Koehn

Papers

- Sutskever et al., Sequence to Sequence Learning with Neural Networks
- Cho et al., Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation
- Sennrich et al., Neural Machine Translation of Rare Words with Subword Units
- Papineni et al., BLEU: a Method for Automatic Evaluation of Machine Translation
- Koehn, Neural Machine Translation
- Koehn, Six Challenges for Neural Machine Translation
- Curated Machine Translation Reading List

Tutorials

- Pytorch
  - Official PyTorch Seq2Seq Tutorial
  - PyTorch Seq2Seq with Torchtext
  - Ben Trevett Seq2Seq Tutorial
- Tensorflow
  - NMT with Attention

Libraries

- Facebook: fairseq (PyTorch)
- Open NMT (PyTorch)
- Open NMT (Tensorflow)

Thank You!