

Applied Machine Learning!!!

W207 Section 9

Rasika Bhalerao

rasikabh@berkeley.edu

Aug 23: Welcome!
Nov 8 and 22: No classes

Schedule

Supervised learning methods

	Sync	Topic
2	Aug 30	Linear Regression / Gradient Descent
3	Sep 6	Feature Engineering Bonus: Naive Bayes
4	Sep 13	Logistic Regression
5	Sep 20	Multiclass classification / Eval Metrics Bonus: Reinforcement learning
6	Sep 27	Neural Networks
7	Oct 4	KNN, Decision Trees, Ensembles

Unsupervised learning methods

	Sync	Topic
8	Oct 11	KMeans and PCA Bonus: LDA
9	Oct 18	Text Embeddings Bonus: Language models
10	Oct 25	CNNs Bonus: GANs
11	Nov 1	EDA, Real data, Baselines
12	Nov 15	Fairness / Ethics
13	Nov 29	Fancy Neural Networks
14	Dec 6	Final Presentations

Assignment Schedule

Due Date	Assignment
Aug 28	HW1
Sep 4	HW2
Sep 11	HW3
Sep 18	HW4
Sep 25	HW5
Oct 2	HW6
Oct 16	Group project baseline
Oct 23	HW8
Nov 6	HW9
Nov 20	HW10
Dec 4	Final project notebook + presentation

Behavior expectations

- Healthy disagreement is expected
- Be mindful of one another's schedules
- Be a good listener
- Have fun in a professional manner
- Share related real-world experience
- Ask questions when something is confusing
- Keep it 100 but be respectful
- Be open-minded to new ideas in the real world and when coding
- On time for group meetings

How are final projects going?

Guidelines:

https://docs.google.com/document/d/1R7mIH0tYXKU8vEQzw10uofb_iK3sgimw8iZLWSTzdgg/edit?usp=sharing

Word Representations

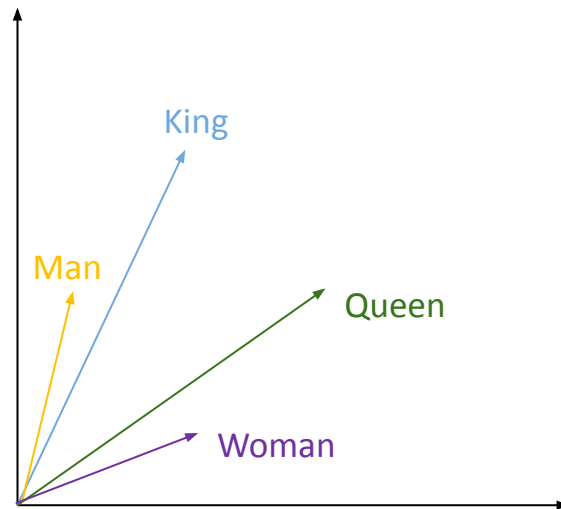
Textbook if you want: Jurafsky & Martin, chapter 6

Word Vector Intuition

Royalty
Masculinity
Femininity
Age

.
. .
.

King	Man	Woman	Queen
0.95	0.07	0.06	0.94
0.85	0.82	0.08	0.11
0.09	0.09	0.85	0.85
0.7	0.7	0.7	0.7



$$\text{King} - \text{Man} + \text{Woman} = \text{Queen}$$

We will calculate these vectors in a self-supervised way.

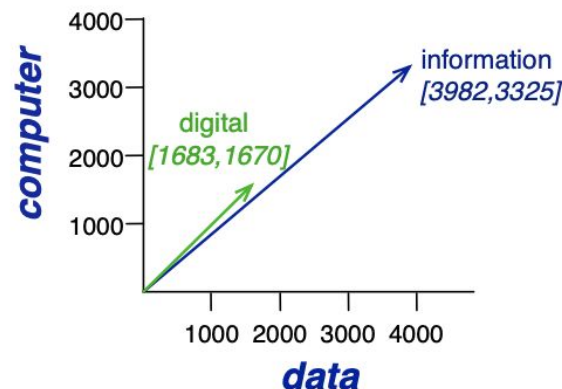
Word Vector Intuition

- Distributional Hypothesis: words with similar meanings appear in similar contexts in the text
- First attempt at word vectors: co-occurrence matrix (aka term-context matrix)

is traditionally followed by **cherry** pie, a traditional dessert
often mixed, such as **strawberry** rhubarb pie. Apple pie
computer peripherals and personal **digital** assistants. These devices usually
a computer. This includes **information** available on the internet

	aardvark	...	computer	data	result	pie	sugar	...
cherry	0	...	2	8	9	442	25	
strawberry	0	...	0	0	1	60	19	
digital	0	...	1670	1683	85	5	4	
information	0	...	3325	3982	378	5	13	

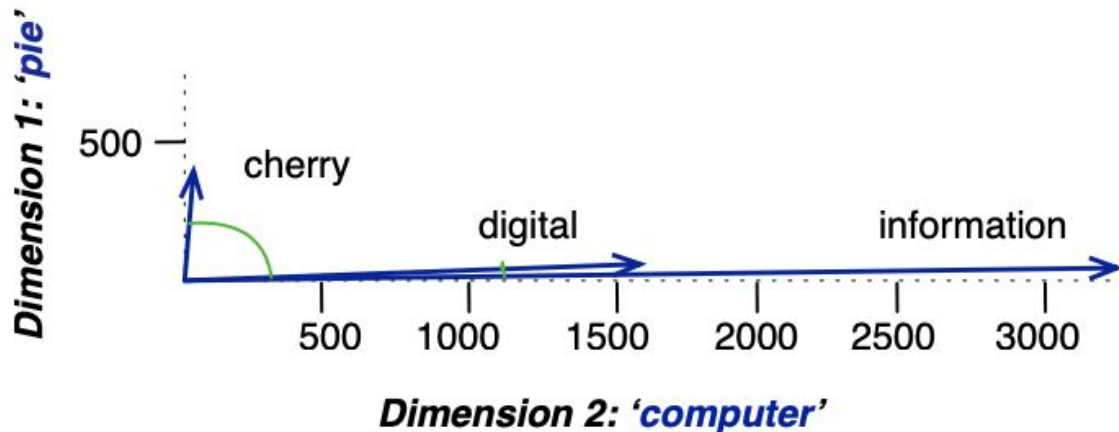
Figure 6.5 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.



Word Vector Evaluation

- SimLex-999 is one of several datasets of human judgements of word similarity (averaged over multiple annotators)
- Evaluation: correlation between cosine similarity of vectors and human similarity judgements

vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3



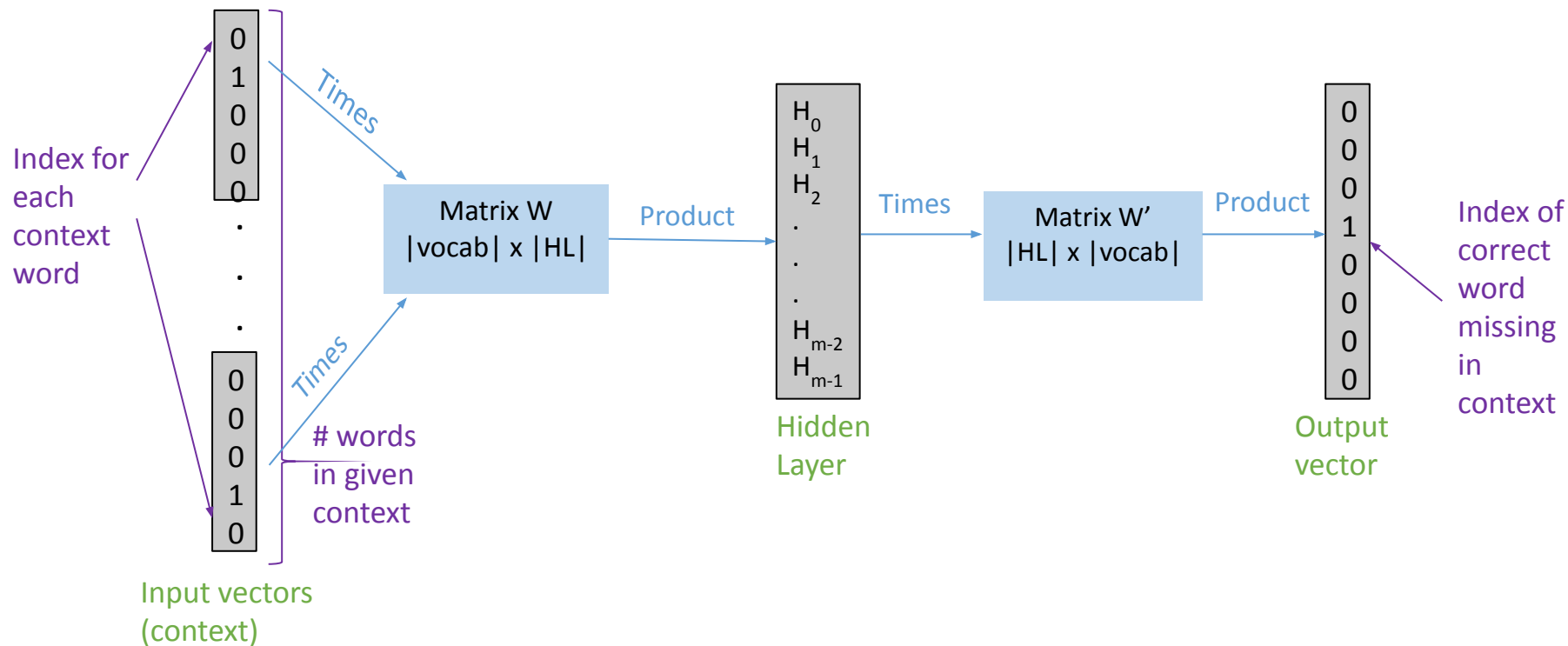
Drawbacks of that “first attempt”

- Huge sparse vectors
- Some words show up overall less often, but they should still matter more because they are more important
 - For example, “extraordinary” shows up less often than “the” but is probably more important in determining word meanings
 - Possible solution: take into account how often you would expect two words to co-occur given their proportion in the text (PMI)
- Better solution: Word2Vec (and GloVe and FastText)

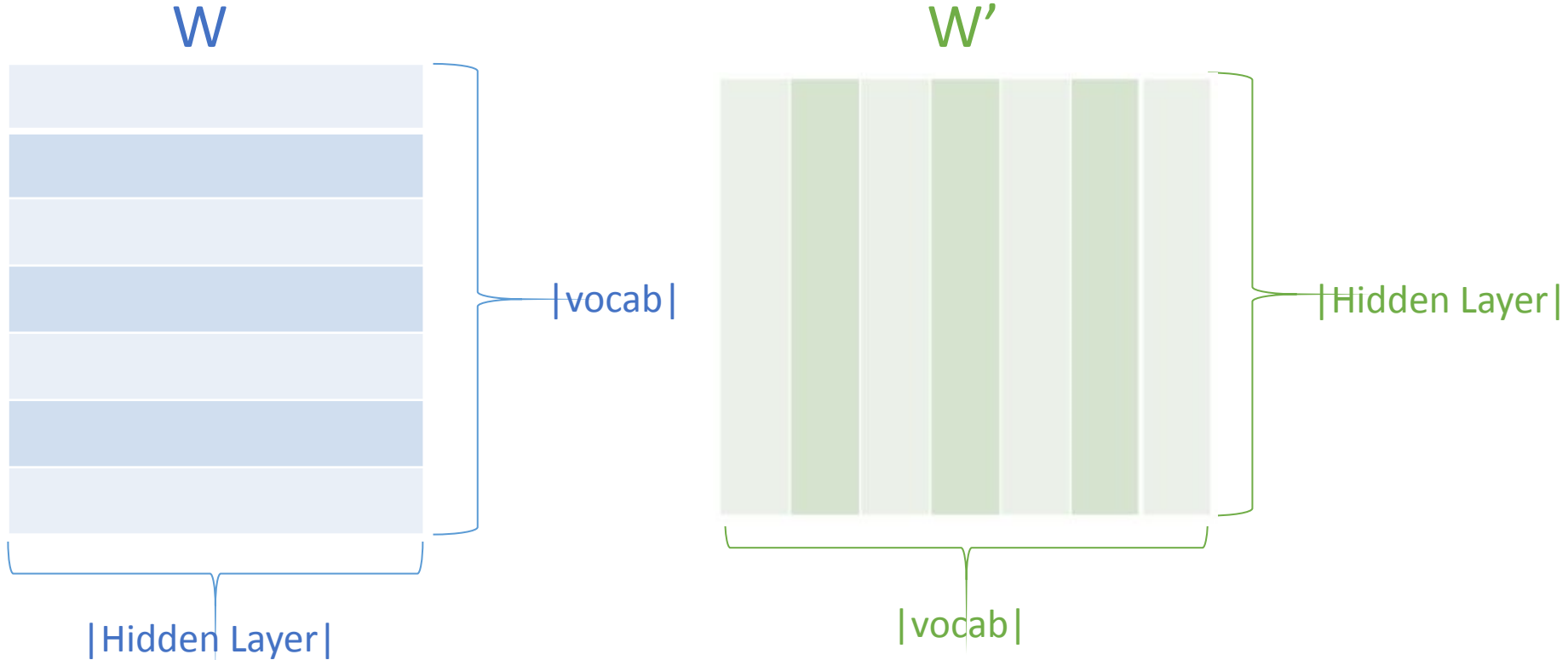
Word2Vec: The Fake Task

- Make vectors where each word has an index
 - (like Bag of Words/Tfidf vectors)
 - Represent each word as a “one-hot encoded” vector (1 for that word, 0 everywhere else)
 - These vectors are the input and output for the fake task
- There are two options for the fake task:
 - Continuous Bag of Words (CBOW)
 - Input: context words that surround v in the text, knowing word v is missing
 - Output: word v
 - Skip-Gram
 - Input: word v
 - Output: words that surround v in the text

Training CBOW



W and W' Matrices



How to train the network

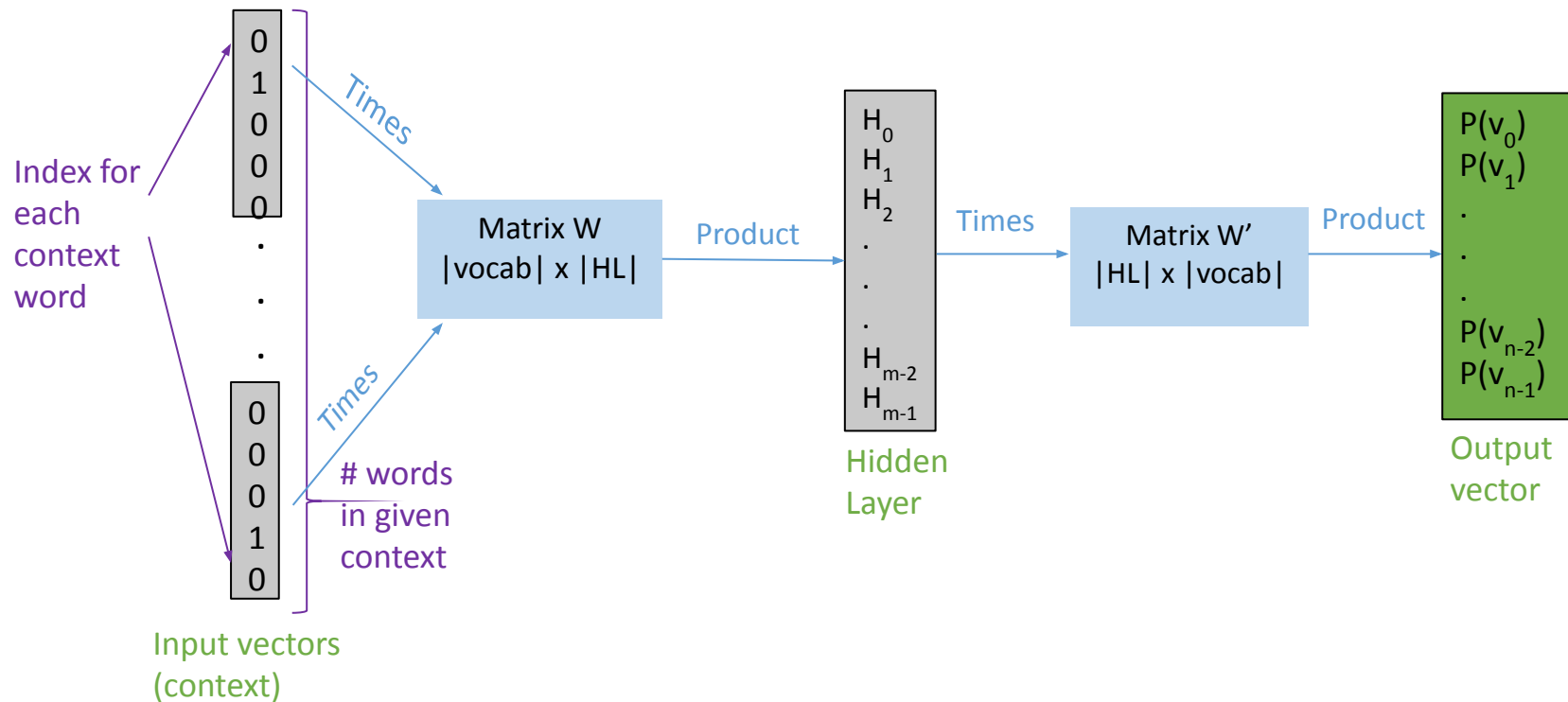
- Data

- Positive examples: pairs of words that are in context of each other
- Negative examples: pairs of words that are not in context of each other
 - Negative sampling: sample these, because if you use all such pairs, that's way too many

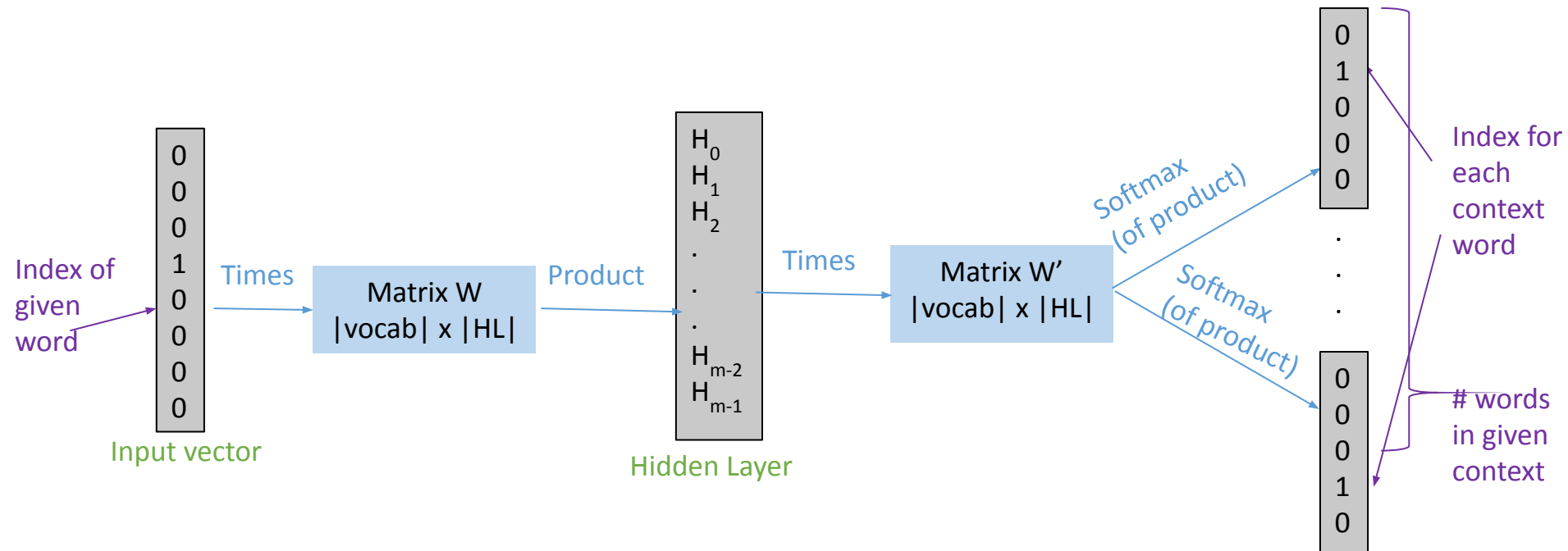
- Training

- Goal: Find values in matrices W and W'
- How: stochastic gradient descent

Revisit CBOW

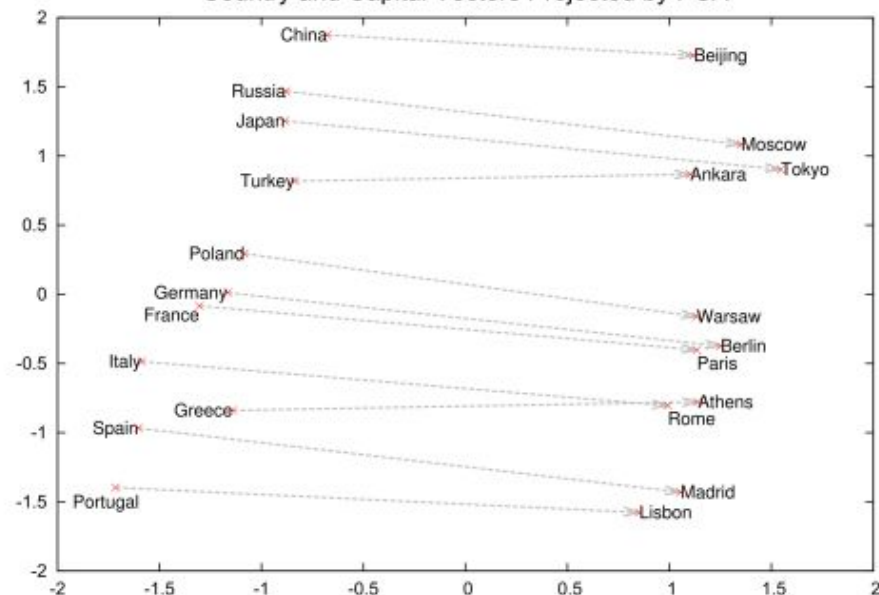


Skip-Gram



Some results

Country and Capital Vectors Projected by PCA



Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

These vectors from [Distributed Representations of Words and Phrases and their Compositionality](#) (Mikilov et al.) were trained on a day of Google News.

Other results from J&M

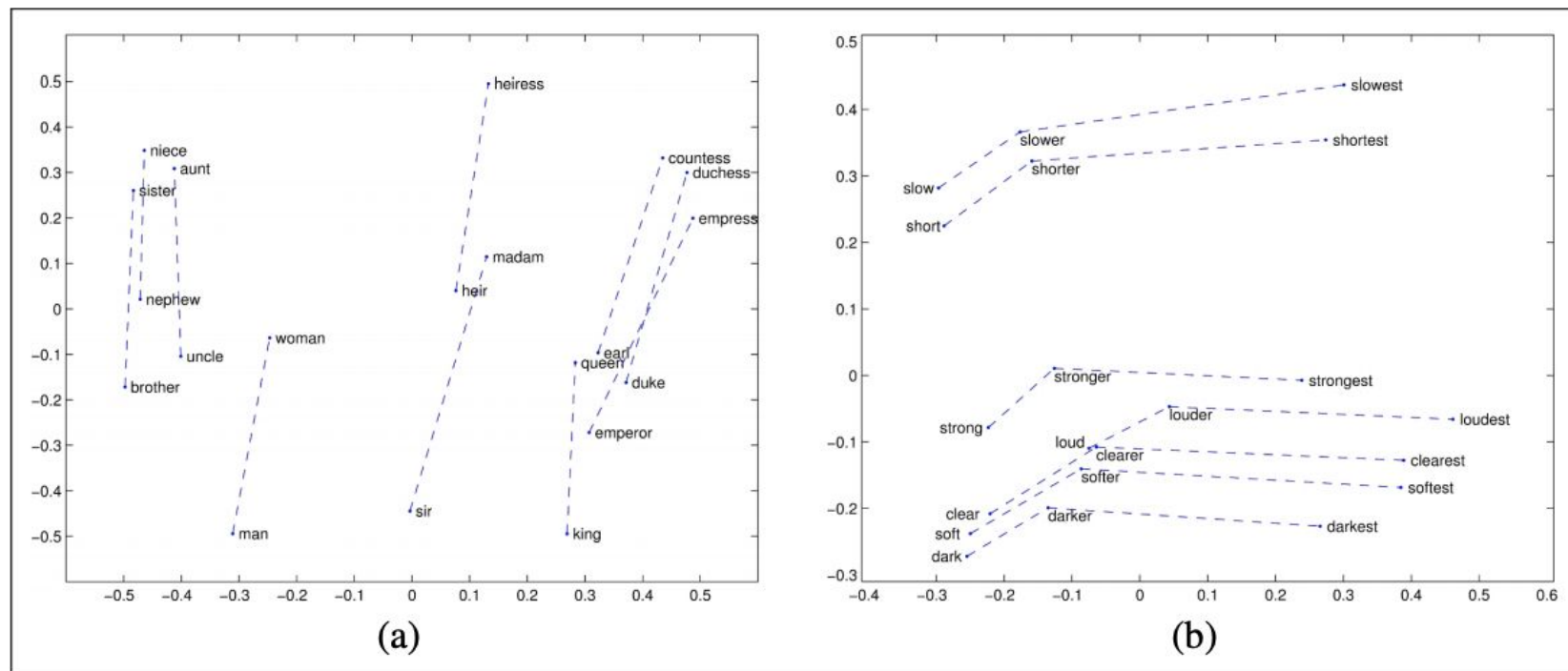


Figure 6.13 Relational properties of the vector space, shown by projecting vectors onto two dimensions. (a) 'king' - 'man' + 'woman' is close to 'queen' (b) offsets seem to capture comparative and superlative morphology (Pennington et al., 2014).

GloVe and FastText

- So far, we covered Word2Vec with CBOW and Skip Gram
- GloVe
 - Like Word2Vec, but it is trained by doing matrix factorization on the co-occurrence matrix (so it reduces the dimension of the co-occurrence matrix)
- FastText
 - Made by Facebook AI
 - Like Word2Vec, but using character ngrams instead of words
 - Each word's vector is the average of the vectors of its character ngrams
 - This has the advantage that if there is a word that wasn't in your training set, you can still assign it a vector from its ngrams

Bias

- Our motivating example was king:man::queen:woman. The same dataset yields computer programmer:man::homemaker:woman ([Bolukbasi 2016](#))
- Vectors for names with African-American names are closer to unpleasant words while vectors for European-American names are closer to pleasant words ([Caliskan 2017](#))
- Word embeddings pick up biases implicitly in the text. “Debiasing” word embeddings is currently an active area of research

Async Practice Quiz Questions (vote!)

The vocabulary always includes all words appearing in the training set.	True	False
The bag-of-words assumption is that word order does not matter.	True	False
One-hot encoding of words does not capture any information about relationships between words.	True	False
If we have a vocabulary with 1000 tokens and our embedding dimension is 32, how many embedding parameters are in our model?		
Learned embeddings encode many complex relationships in a shared vector space.	True	False
Transfer learning assumes that the new task is the same as the original task.	True	False

Notebook!

To access later:

https://github.com/MIDS-W207/rasikabh/blob/main/live_sessions/Week9.ipynb

Also, if you want last semester's assignments:

<https://github.com/MIDS-W207/coursework>

Language Models

Textbook if you want: Jurafsky & Martin, chapter 3

What it means to model a language

- **A language model assigns probabilities to sequences of words**
- For example, the next word in the sequence starting with “Please turn your homework” is probably “in” or “over” but not “the”
- For example, “all of a sudden I notice three guys standing on the sidewalk” is more likely than “on guys all I of notice sidewalk three a sudden standing the”

Uses of language modeling

- **Speech recognition:** decide if it is more likely they said “I will be back soonish” or “I will be bassoon dish”
- **Spelling / grammar correction:** decide how likely it is they meant what they typed, and decide what is the most likely thing they meant to say
- **Text prediction:** what is the most likely next word they are about to type?

- **Machine translation:**

他 向 记者 介绍了 主要 内容
He to reporters introduced main content

he introduced reporters to the main contents of the statement

he briefed to reporters the main contents of the statement

he briefed reporters on the main contents of the statement

Ngram Language Models

Naive first attempt:

$$\begin{aligned} P(w_1^n) &= P(w_1)P(w_2|w_1)P(w_3|w_1^2)\dots P(w_n|w_1^{n-1}) \\ &= \prod_{k=1}^n P(w_k|w_1^{k-1}) \end{aligned}$$

w_a^b = sequence $w_a, w_{a+1}, w_{a+2}, \dots, w_{b-1}, w_b$

We measure $P(w_i | w_1^{i-1})$ by **counting the number of times the sequence w_1^{i-1} happens in the training set**, and measuring the **percent of times it was followed by w_i**

What is wrong with this?

Ngram Language Models

Rather than using the **entire previous sequence**, we use only the **last n terms**

$$P(w_n | w_1^{n-1}) \approx P(w_n | w_{n-N+1}^{n-1})$$

The probability of the sequence is:

$$P(w_1^n) = \prod_{k=1}^n P(w_k | w_1^{k-1}) \approx \prod_{k=1}^n P(w_k | w_{k-N+1}^{k-1})$$

We calculate these conditional probabilities from the training set:

$$P(w_n | w_{n-N+1}^{n-1}) = \frac{C(w_{n-N+1}^{n-1} w_n)}{C(w_{n-N+1}^{n-1})}$$

Ngram Language Model Example

In this example, $N=2$

- If N is bigger, the leftmost column will be sequences of words

We use the Berkeley Restaurant Project dataset, which looks like this:

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

can you tell me about any good cantonese restaurants close by
mid priced thai food is what i'm looking for
tell me about chez panisse
can you give me a listing of the kinds of food that are available
i'm looking for a good place to eat breakfast
when is caffe venezia open during the day

Figure 3.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

Ngram Language Model Example

$P(< s> \text{ i want } \mathbf{english} \text{ food } < / s>)$

$= P(\text{i} | < s>) P(\text{want} | \text{i}) P(\text{english} | \text{want}) P(\text{food} | \text{english}) P(< / s> | \text{food})$

$= .25 \times .33 \times .0011 \times 0.5 \times 0.68$

$= .000031$

$P(< s> \text{ i want } \mathbf{chinese} \text{ food } < / s>)$

$= P(\text{i} | < s>) P(\text{want} | \text{i}) P(\text{chinese} | \text{want}) P(\text{food} | \text{chinese}) P(< / s> | \text{food})$

$= .25 \times .33 \times .0065 \times 0.5 \times 0.52$

$= 0.000139$

	i	want	to	eat	chinese	food	lunch	spend
i	0.002	0.33	0	0.0036	0	0	0	0.00079
want	0.0022	0	0.66	0.0011	0.0065	0.0065	0.0054	0.0011
to	0.00083	0	0.0017	0.28	0.00083	0	0.0025	0.087
eat	0	0	0.0027	0	0.021	0.0027	0.056	0
chinese	0.0063	0	0	0	0	0.52	0.0063	0
food	0.014	0	0.014	0	0.00092	0.0037	0	0
lunch	0.0059	0	0	0	0	0.0029	0	0
spend	0.0036	0	0.0036	0	0	0	0	0

Figure 3.2 Bigram probabilities for eight words in the Berkeley Restaurant Project corpus of 9332 sentences. Zero probabilities are in gray.

Ngram Language Models: Practical Considerations

- When $N > 2$, the first few probabilities have “dummy” words
 - E.g. For $N = 3$ and a sentence starting with “I” we use $P(I \mid \langle s \rangle \langle s \rangle)$
- We often combine $N = 2, 3, \dots$ up to the real N
 - Use a weighted sum of the probabilities
- In the code, we usually use log-probabilities
- We often use Laplace add-k smoothing (like with Naive Bayes) to deal with unknown words (out-of-vocabulary or OOV) and ngram sequences
- Another way to deal with OOV is in the training set, label all words that appear less than n times as $\langle \text{UNK} \rangle$. Test set OOV words are also $\langle \text{UNK} \rangle$

Ngram Language Models: Generating Sentences

- Start with <s>.
- Since we have a probability distribution for $P(w \mid \text{<s>})$, we randomly pick the next word using those probabilities.
- Continue randomly picking the next word according to the probability distributions until you reach </s>.

1 gram	–To him swallowed confess hear both. Which. Of save on trail for are ay device and rote life have –Hill he late speaks; or! a more to leg less first you enter
2 gram	–Why dost stand forth thy canopy, forsooth; he is this palpable hit the King Henry. Live king. Follow. –What means, sir. I confess she? then all sorts, he is trim, captain.
3 gram	–Fly, and will rid me these news of price. Therefore the sadness of parting, as they say, 'tis done. –This shall forbid it should be branded, if renown made it empty.
4 gram	–King Henry. What! I will go seek the traitor Gloucester. Exeunt some of the watch. A great banquet serv'd in; –It cannot be but so.

Figure 3.3 Eight sentences randomly generated from four n-grams computed from Shakespeare's works. All characters were mapped to lower-case and punctuation marks were treated as words. Output is hand-corrected for capitalization to improve readability.

Naive Neural Language Model: Sliding Window

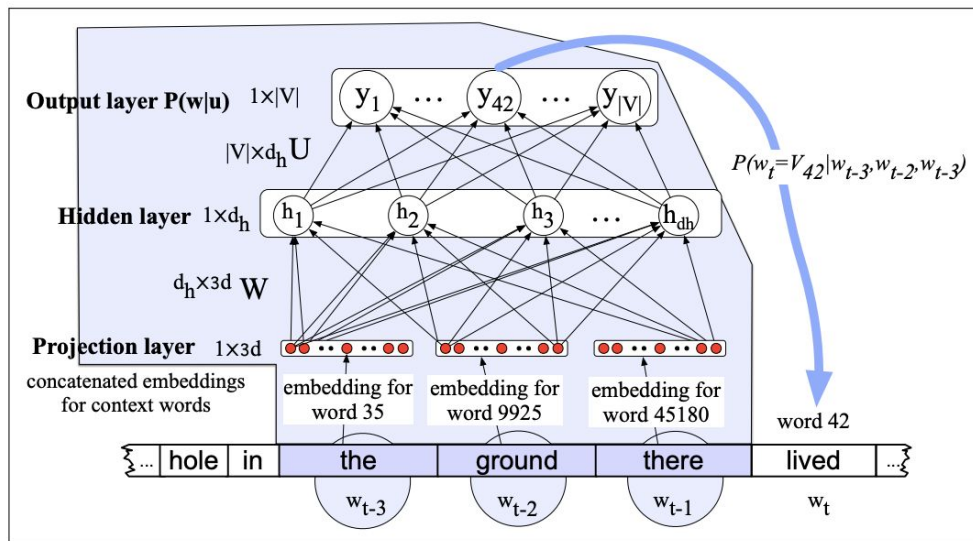


Figure 9.1 A simplified view of a feedforward neural language model moving through a text. At each time step t the network takes the 3 context words, converts each to a d -dimensional embedding, and concatenates the 3 embeddings together to get the $1 \times Nd$ unit input layer x for the network.

- Feedforward neural LMs are no longer SOTA, but the basics still apply
- **Predict each next word**
- Train using cross entropy loss for the correct next word
$$L = -\log p(w_t | w_{t-1}, \dots, w_{t-n+1})$$
- Represent inputs using **word embeddings** (this helps because similar words are similar)

Naive Neural LM: Sliding Window

- **Pretraining** = learning the word embeddings via another method (like Word2Vec) beforehand
- To learn the embeddings while training for the downstream task, represent the inputs as one-hot vectors and add another layer E

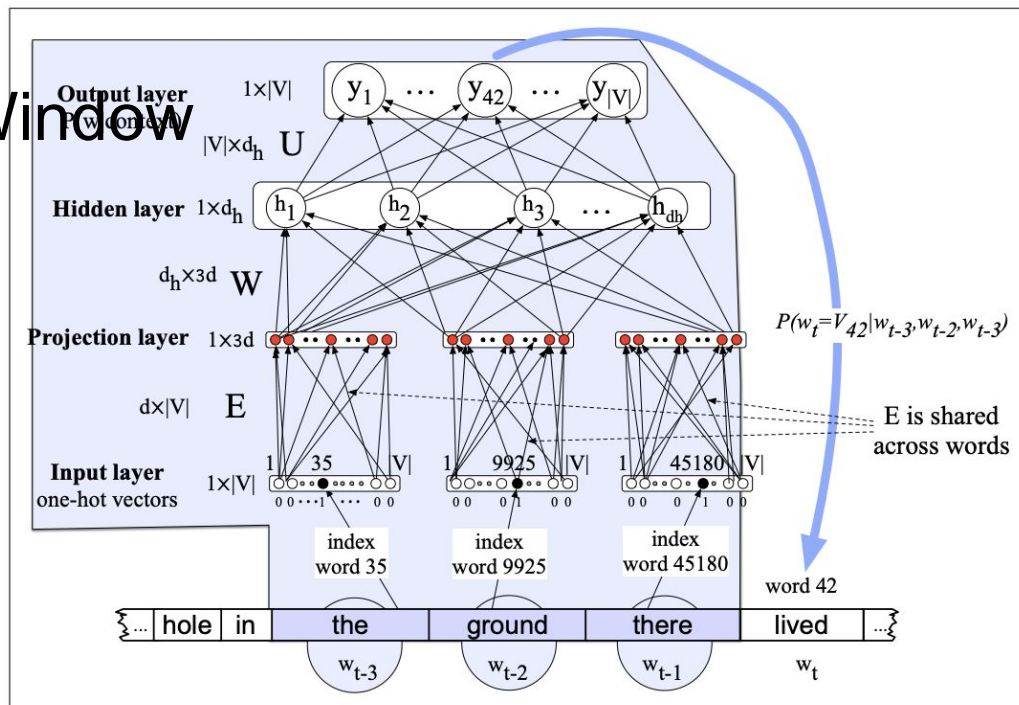


Figure 7.13 Learning all the way back to embeddings. Notice that the embedding matrix E is shared among the 3 context words.

- It is the same matrix E for all steps, but it still works because multiplying by the one-hot makes it select the part that is specific to the given input word
- This helps if your task only cares about certain aspects of words (e.g. sentiment)

Naive Neural Language Model: Sliding Window

- Output of each window has no impact on next window
- Fixed window size \rightarrow cannot use information from parts of sequence outside of the window
- Has to learn the same pattern in multiple “locations” in the features (ex. “the ground” appears as different features in different iterations, so any patterns have to be learned in multiple locations)

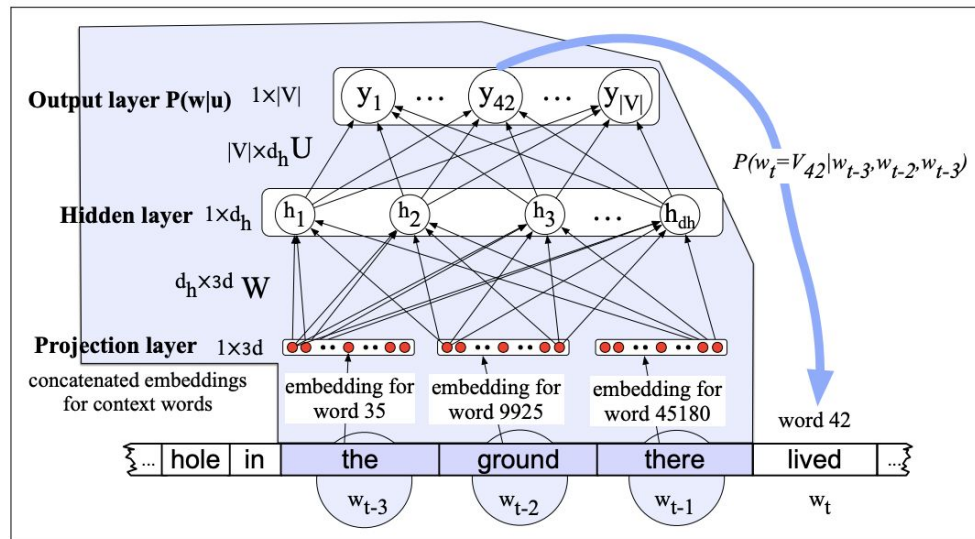


Figure 9.1 A simplified view of a feedforward neural language model moving through a text. At each time step t the network takes the 3 context words, converts each to a d -dimensional embedding, and concatenates the 3 embeddings together to get the $1 \times Nd$ unit input layer x for the network.

Sequences

- Non-sequential machine learning takes input from a single point in time, and outputs a prediction or classification:

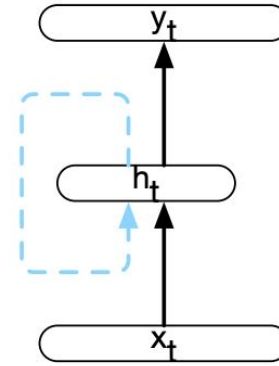
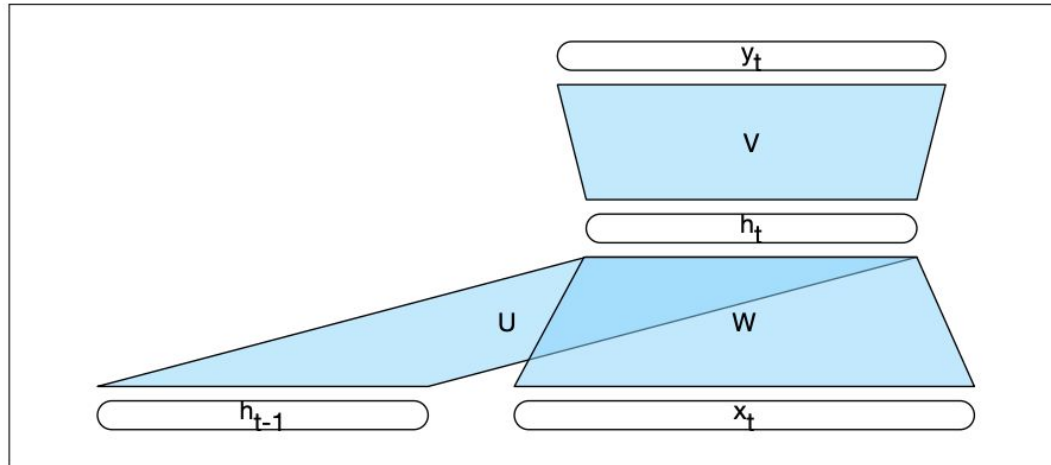
$$y = f(x(t))$$

- Now we look at sequential inputs where the output y can depend on more than just the immediate input:

$$y = f(s(t)) = F(x(t), x(t-1), \dots, x(1))$$

Solution: Recurrent Neural Network

- The hidden state can loop back to itself to use again with the next term in the sequence
- No fixed length



neural network after Elman (Elman, 1990). The hidden layer is fed back into the input layer as part of its input. That is, the activation value of the hidden layer is fed back into the input layer as well as the activation value of the hidden layer from the

Figure 9.3 Simple recurrent neural network illustrated as a feedforward network.

Training a Neural Network

- **Forward pass:** process the sequence
- **Backward pass:** Backpropagation Through Time
- Note: since the input at each step affects multiple outputs (and therefore different parts of loss), we “assign blame” proportionally to weights when updating weights in gradient descent

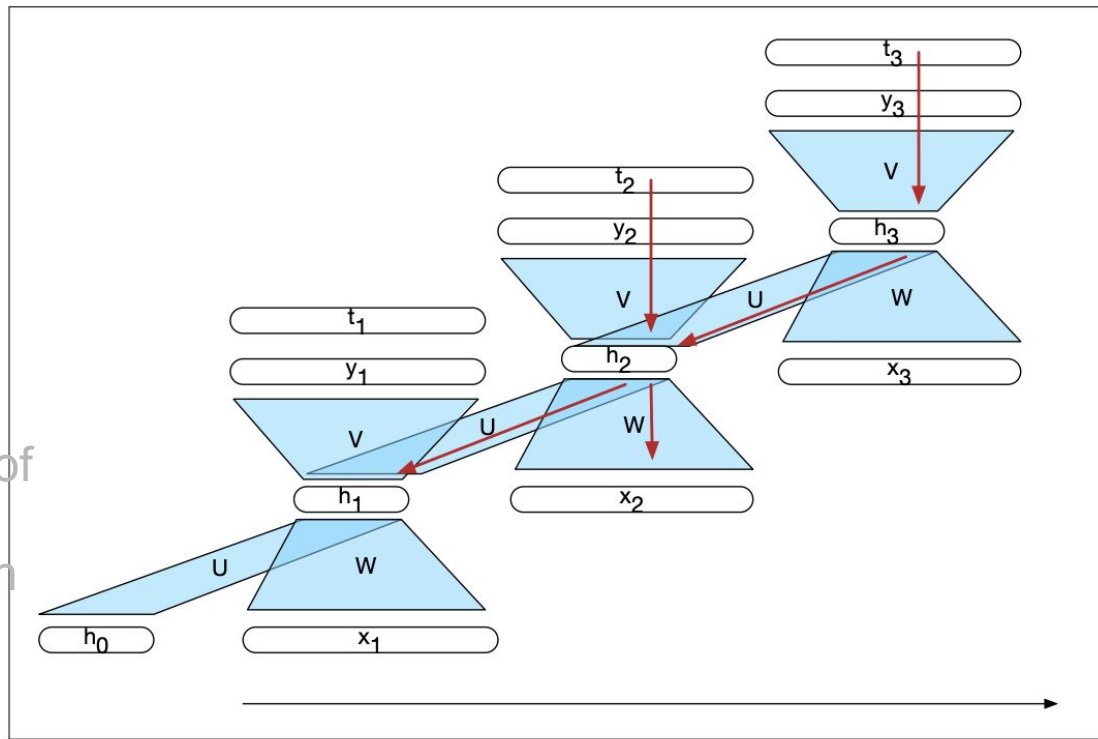


Figure 9.6 The backpropagation of errors in a simple RNN t_i vectors represent the targets for each element of the sequence from the training data. The red arrows illustrate the flow of backpropagated errors required to calculate the gradients for U , V and W at time 2. The two incoming arrows converging on h_2 signal that these errors need to be summed.

Generation with Recurrent Neural Language Models

- Called autoregressive generation
- Shannon's Method: do generation the same way we did with Ngram language models, where we start with <s> and randomly pick the next word using the probabilities until we pick </s> (or reach a fixed length limit)
- Machine translation, summarization, and question answering work by using a variation of this, but with the last hidden state of the “encoder” RNN instead of <s>

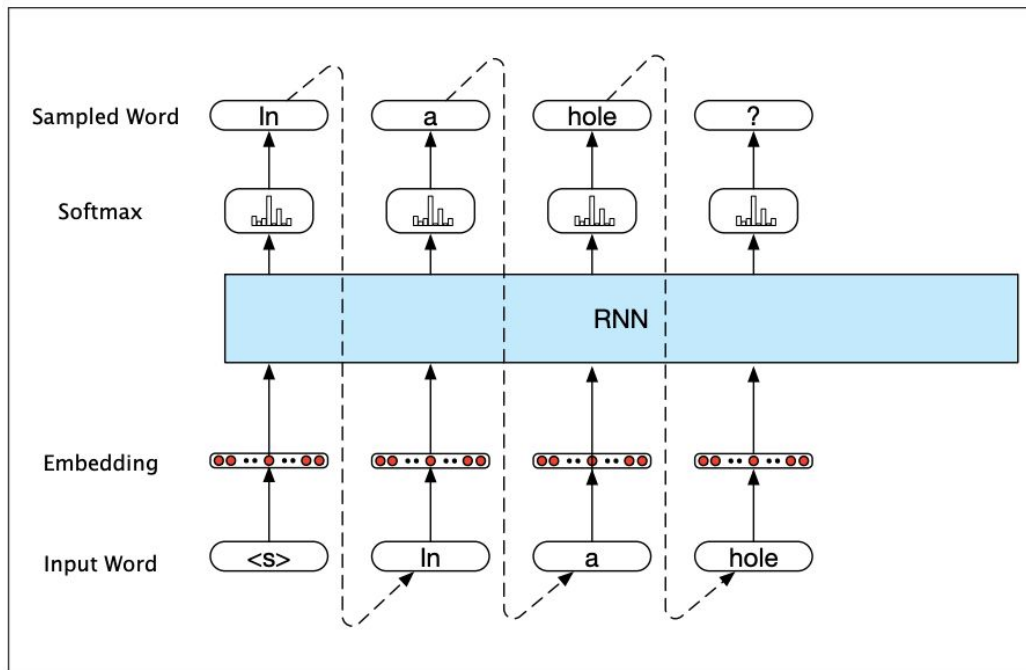


Figure 9.7 Autoregressive generation with an RNN-based neural language model.

Sequence Labeling with RNNs

- Task: label each input word (e.g. POS tags)
- Inputs: word embeddings
- Outputs: tag / label for each word
- Common way to draw RNN →

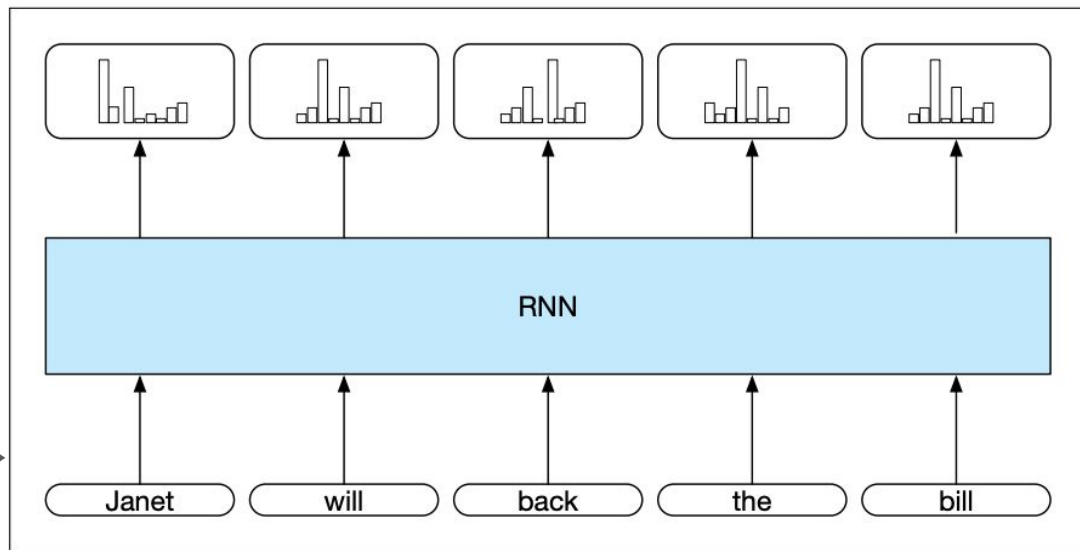
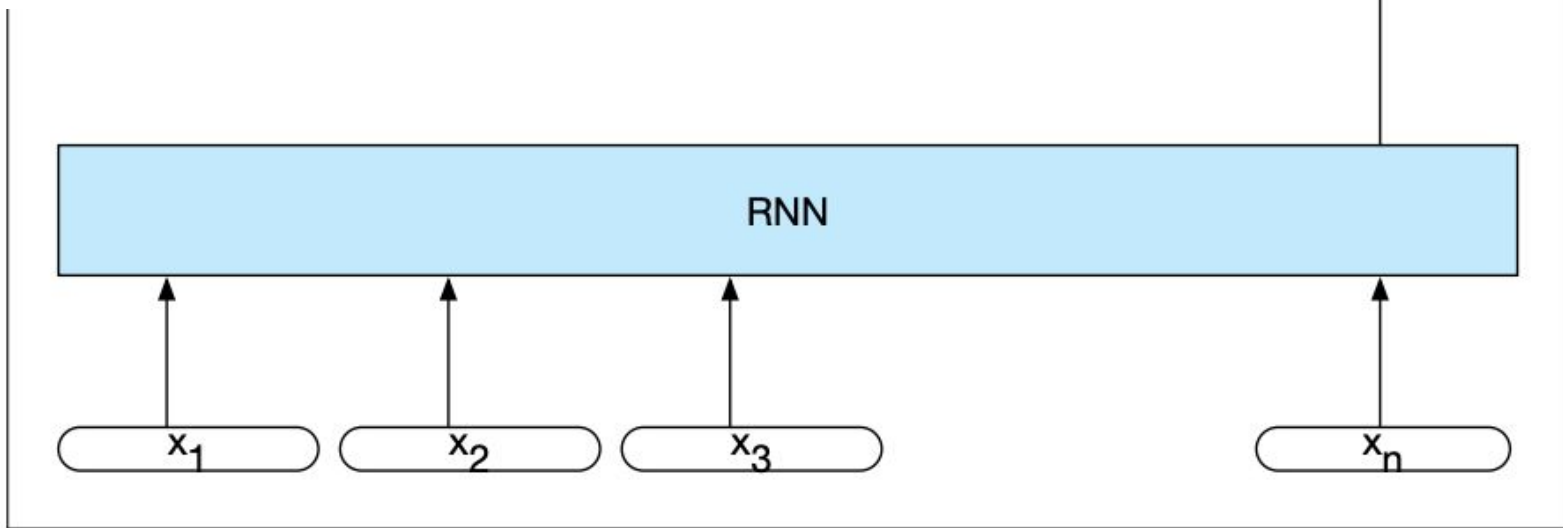


Figure 9.8 Part-of-speech tagging as sequence labeling with a simple RNN. Pre-trained word embeddings serve as inputs and a softmax layer provides a probability distribution over the part-of-speech tags as output at each time step.

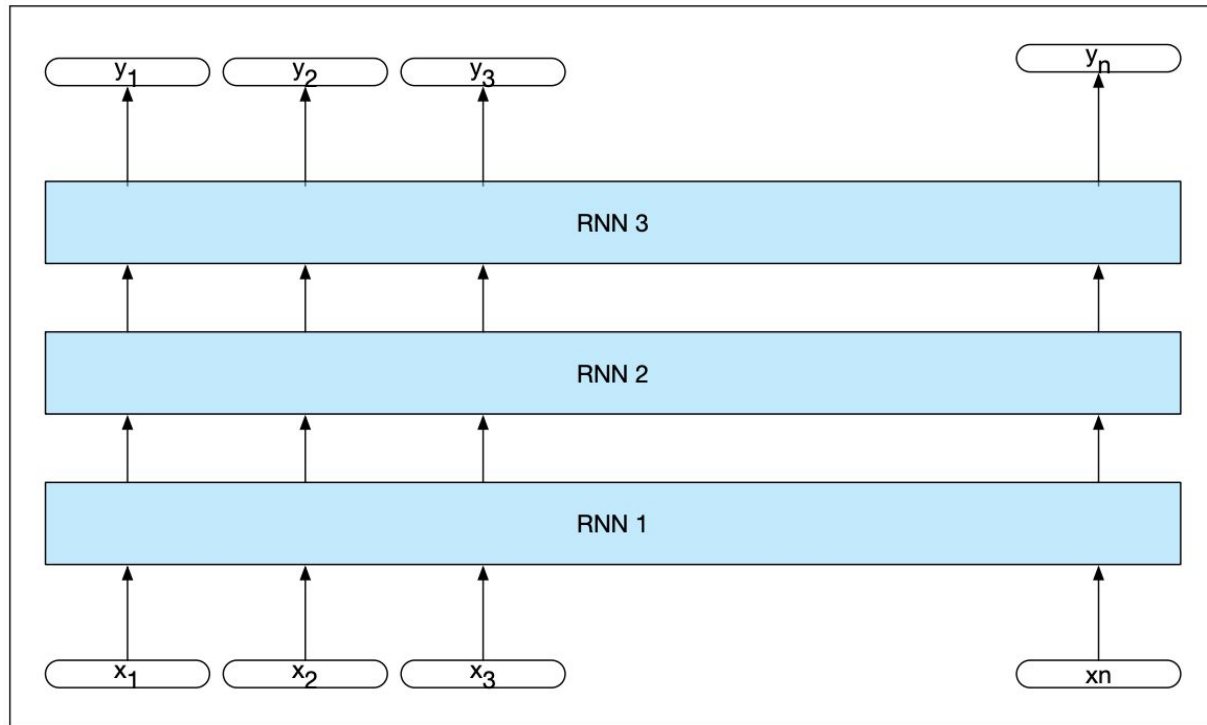
Sequence Classification with RNNs

- Task: classify the entire document
- Inputs: word embeddings (for each word in the document)
- Pass the final hidden layer into a feedforward network
- There is no loss at each step, only at the end



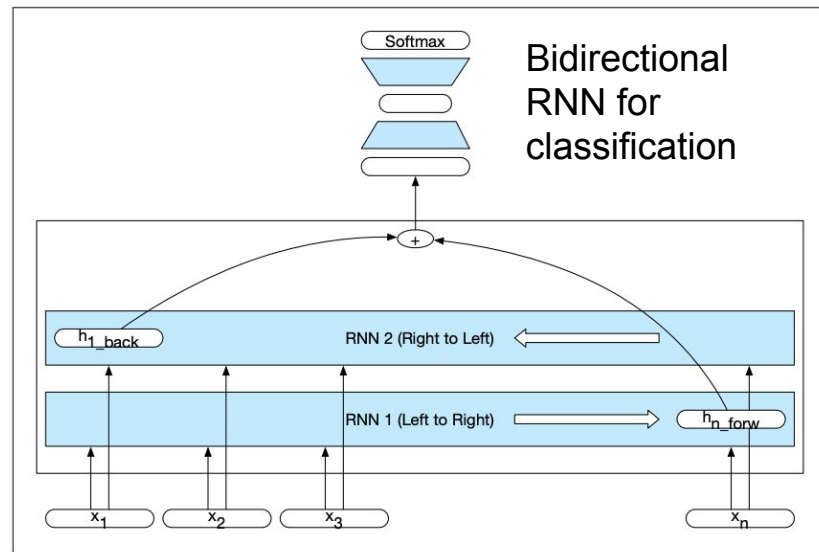
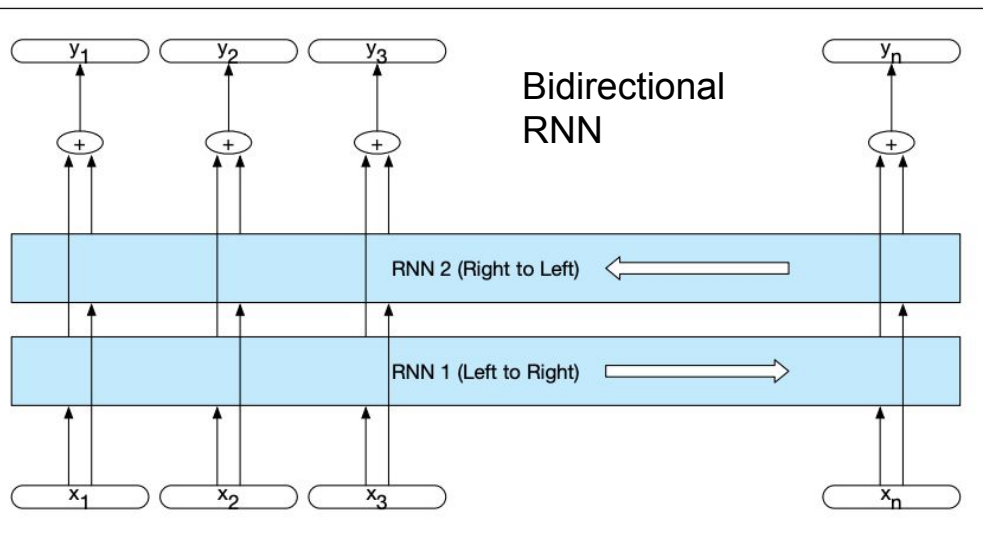
Stacked RNNs

- Take the sequence of hidden layers as input to another RNN
- This has been shown to help detect layers of abstraction (the same way our eyes and conv. neural networks use pixels to detect edges and then use edges to detect shapes)
- This is slow



Bidirectional RNNs

- One RNN goes forward as usual
- Another RNN starts from the last word and goes backwards
- Hidden layer at each step = combination of the hidden state from the forward and backward RNNs
 - Combine using concatenation, addition, multiplication, or averaging



Long Short-Term Memory (LSTM)

- **Vanishing Gradients problem:** with a regular RNN, due to the multiplications in the chain rule, words at the beginning of a sequence have less of an impact on the output than words at the end of the sequence
- This is why **in NLP, we usually use an LSTM**
- We address this by adding a context layer and three “gates”
 - Each gate has a feedforward layer, a **sigmoid activation**, and then **pointwise multiplication** with the layer being gated
 - The **sigmoid almost turns the output of the feedforward layer into a binary mask** (sigmoid pushes numbers close to 0 and 1)
 - This helps **tell the network which elements of the layer being gated are important to remember**
 - The gates are the forget gate, add gate, and output gate

Subwords

Downsides to representing entire words as embeddings:

- Some lexicons have too many words
- Unknown words (rare words and misspellings)
- Morphological information is useful

Approaches to addressing this:

- Character ngrams
- Subwords based on phonetic analysis
- Morphological analysis to get linguistically motivated word pieces (usually overkill)

RNNs with Subwords

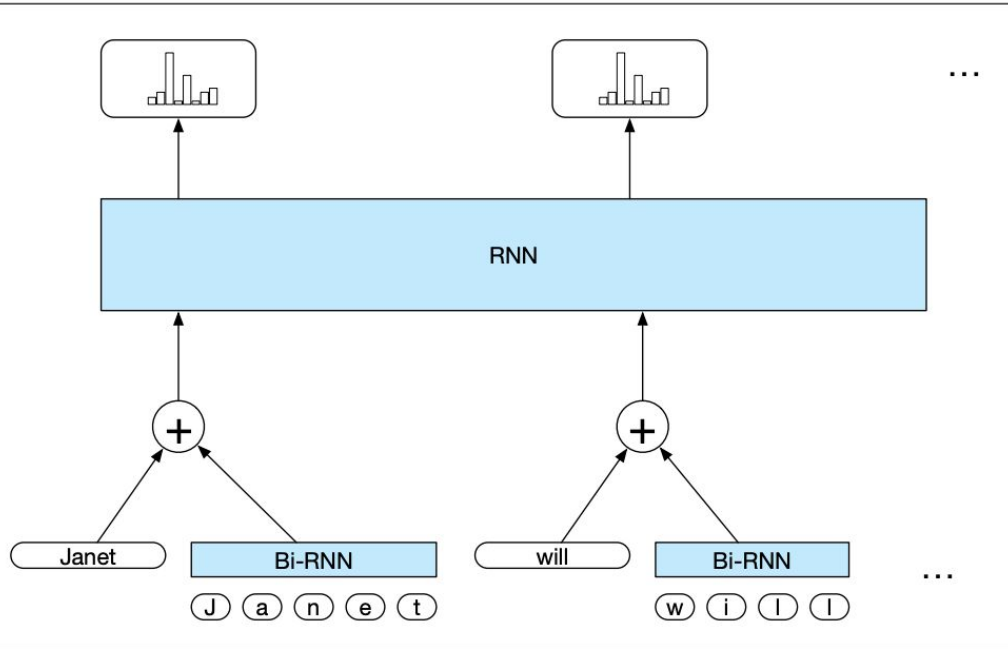


Figure 9.15 Sequence labeling RNN that accepts distributional word embeddings augmented with character-level word embeddings.

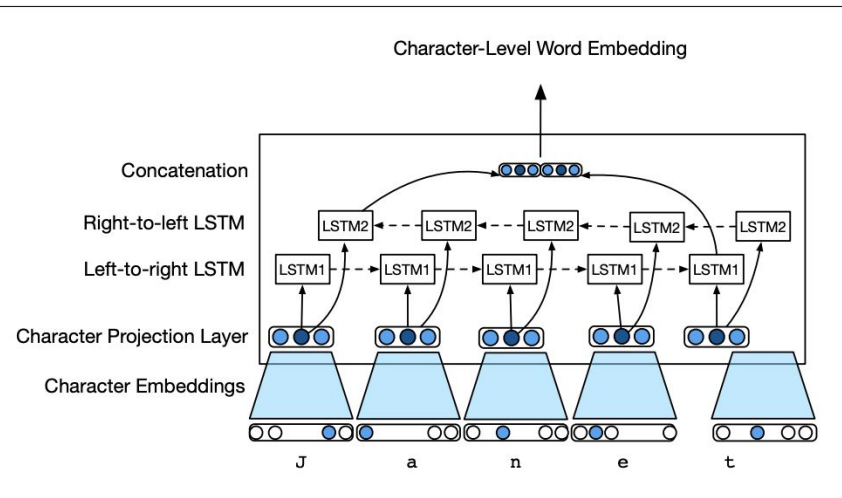


Figure 9.16 Bi-RNN accepts word character sequences and emits embeddings derived from a forward and backward pass over the sequence. The network itself is trained in the context of a larger end-application where the loss is propagated all the way through to the character vector embeddings.

The lower Bi-RNN processes sequences of characters, and the upper RNN processes sequences of words.

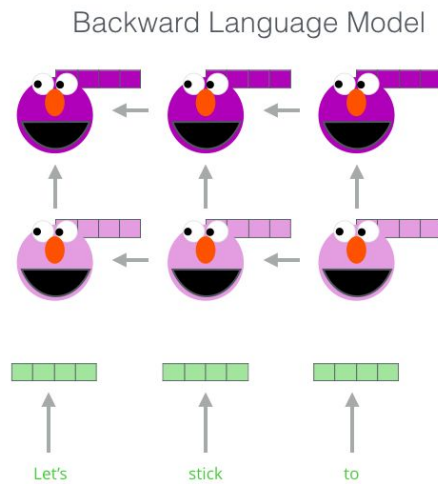
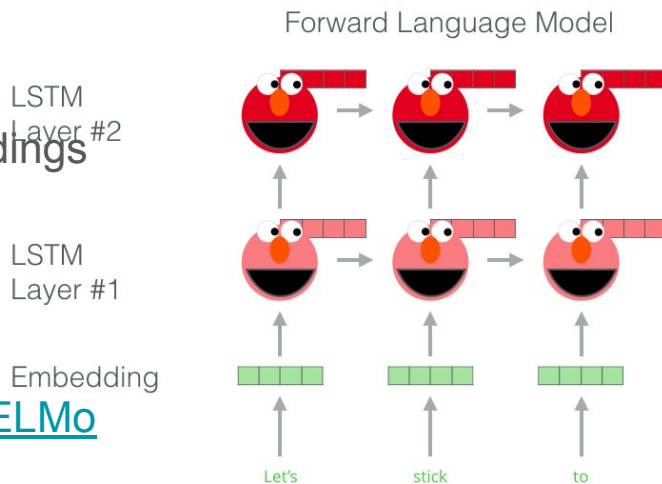
This way you can still get output per word.

The backprop goes from the output task all the way back to the character level.

RNNs can be any variation (LSTM, Bi-RNN, ...)

ELMo

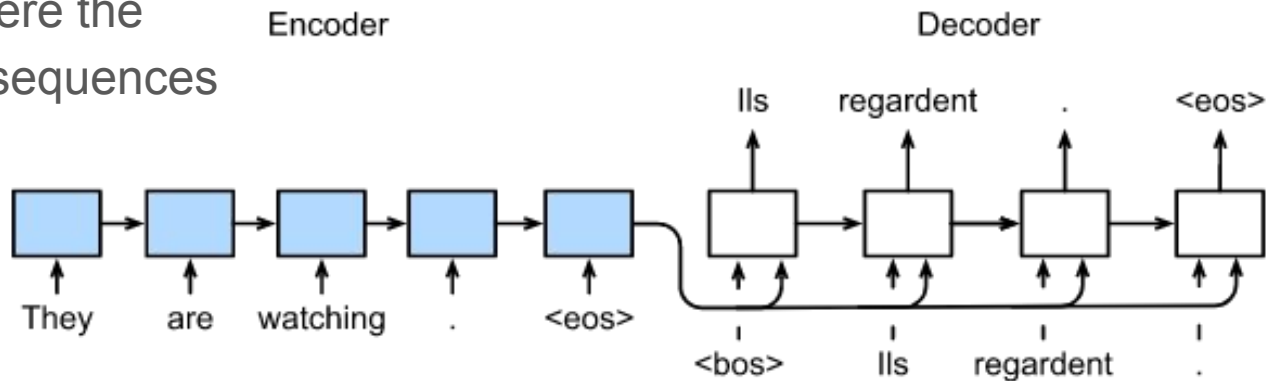
- ELMo is a popular language model published at [NAACL-HTL 2018](#)
- Motivation: contextualized word representations
 - Embeddings for the same word are different when the word shows up in different places
- Represent input words as one-hot vectors, and have stacked, bidirectional LSTM LMs
- The contextualized word embeddings are combinations of the hidden states and the embedding layer
- Contextualized embeddings perform much better than non-contextual
- Layer #1: syntactic
Layer #2: semantic
- [AllenNLP's pretrained ELMo](#)



Encoder-Decoder Models

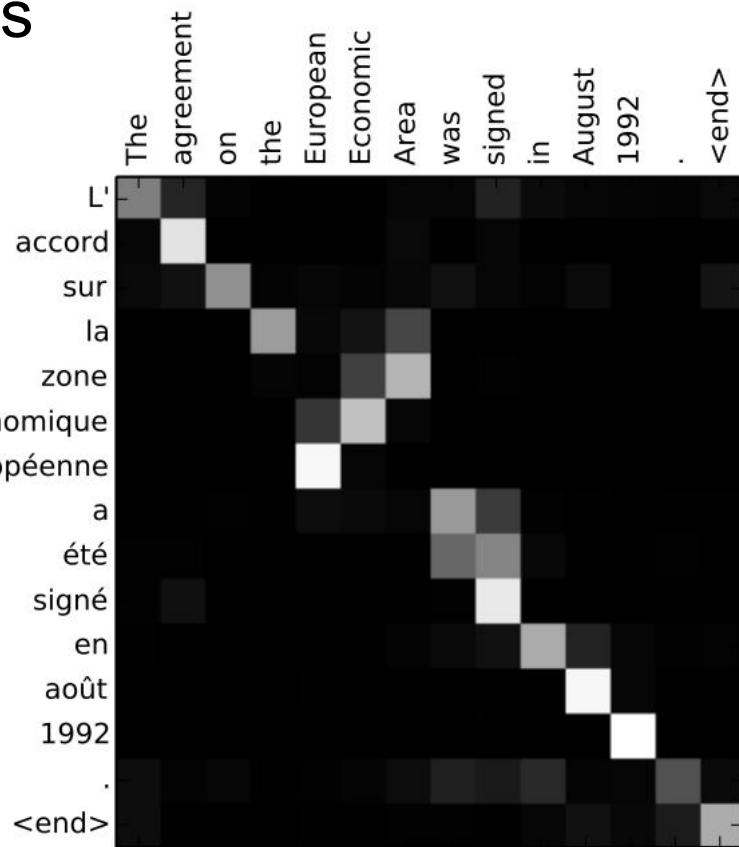
- Also called Sequence to Sequence Models
- Two RNNs: one that “encodes” by taking input from the first sequence, and the other that “decodes” by taking the last hidden state from the encoder as input and generating a sequence
- Used for any task where the input and output are sequences

- Machine translation
- Summarization
- Question answering
- Dialogue

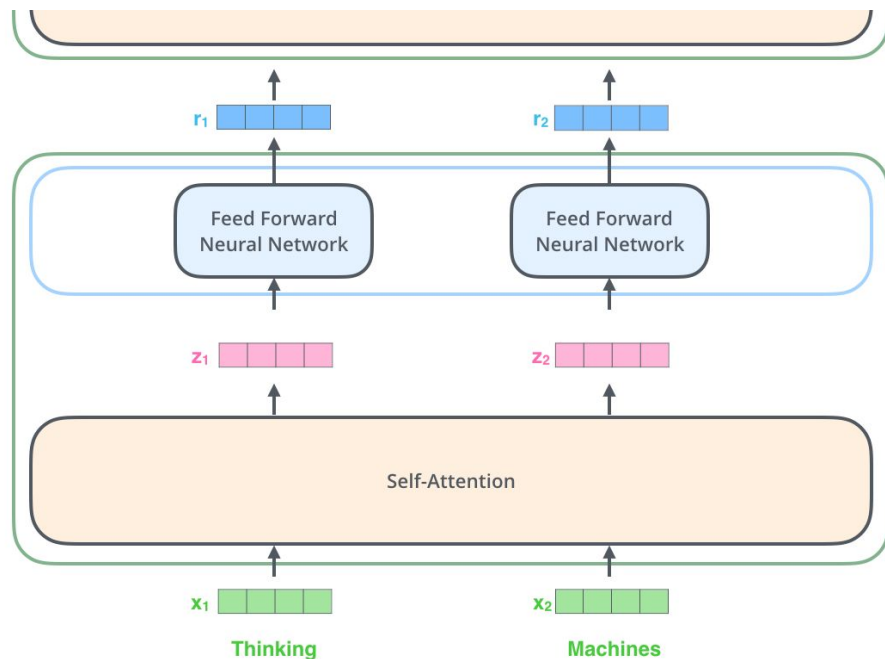
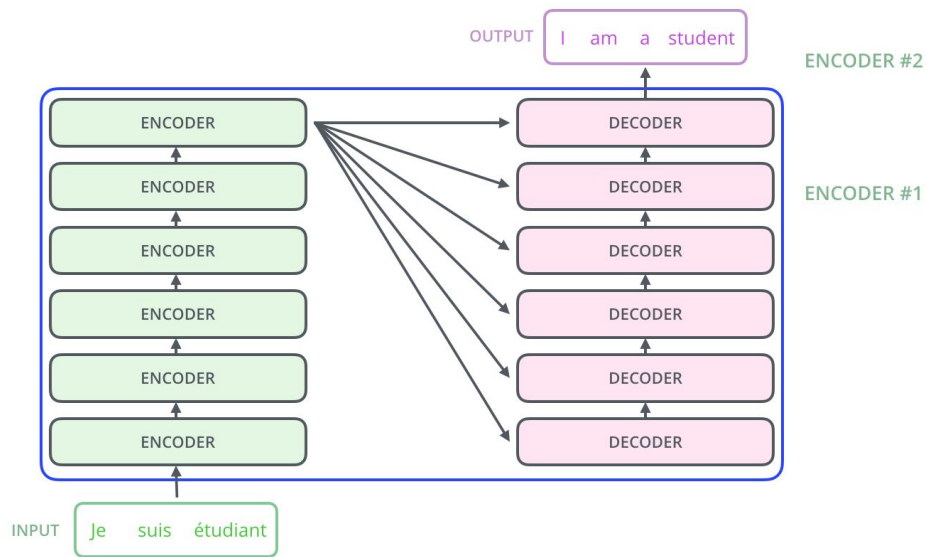


Attention in Encoder-Decoder Models

- Another revelation in NLP in 2018
- Pass all the hidden states from the encoder RNN to the decoder RNN (not just the last hidden state)
- **The decoder uses *attention* to figure out which encoder hidden states are relevant at each step**
- Attention is just another learned parameter, a number for each encoder hidden layer for each decoding step, softmaxed to be between 0 and 1
- At each decoding step, concatenate the current hidden layer with a combination of the attention-multiplied encoder hidden states, and pass this vector into a feedforward layer to get the output word at this step



What if we just use attention (and feedforward)?

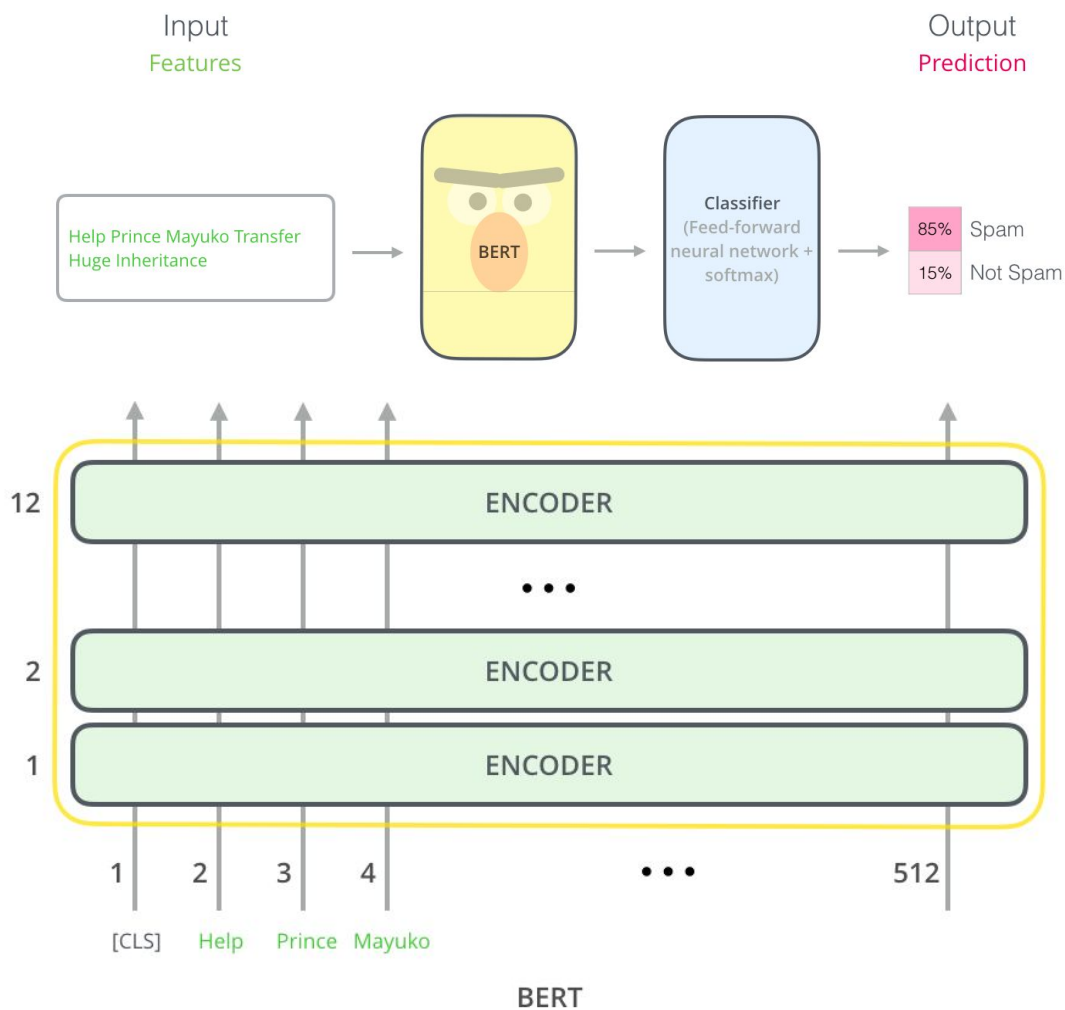


Self-attention is: for each word, how important is each other word?

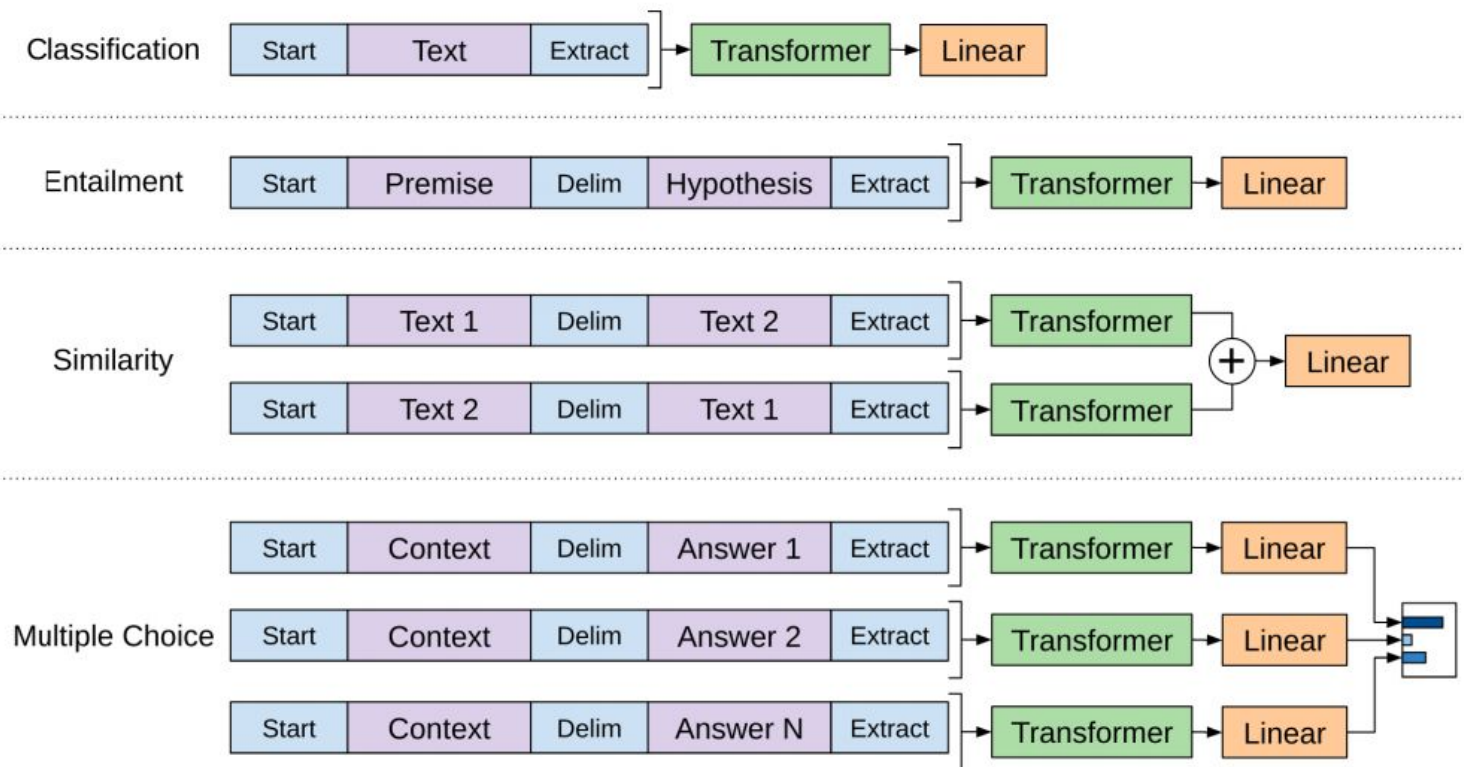
- Details on matrices to calculate that:
<https://jalammar.github.io/illustrated-transformer/>

BERT

- **“Masked” Language Model:** instead of predicting the next word, **mask a random 15% of the words and predict them**
- BERT Base is a stack of 12 encoders (BERT Large is 24)
- Take the output of the encoder and pass it to a layer to do your task (like classification)
- “Attention heads” = the number of times it learns each attention in each encoder (it combines them into one output to pass on)














More Example BERT Tasks



The Glue Benchmark

  SuperGLUE





 Paper  Code  Tasks  **Leaderboard**  FAQ  Diagnostics  Submit  Login

Rank	Name	Model	URL	Score	CoLA	SST-2	MRPC	STS-B	QQP	MNLI-m	MNLI-mm	QNLI	RTE	WNLI	AX
1	HFL iFLYTEK	MacALBERT + DKM		90.7	74.8	97.0	94.5/92.6	92.8/92.6	74.7/90.6	91.3	91.1	97.8	92.0	94.5	52.6
	2	Alibaba DAMO NLP	StructBERT + TAPT 	90.6	75.3	97.3	93.9/91.9	93.2/92.7	74.8/91.0	90.9	90.7	97.4	91.2	94.5	49.1
	3	PING-AN Omni-Sinitic	ALBERT + DAAF + NAS	90.6	73.5	97.2	94.0/92.0	93.0/92.4	76.1/91.0	91.6	91.3	97.5	91.7	94.5	51.2
	4	ERNIE Team - Baidu	ERNIE 	90.4	74.4	97.5	93.5/91.4	93.0/92.6	75.2/90.9	91.4	91.0	96.6	90.9	94.5	51.7
	5	T5 Team - Google	T5 	90.3	71.6	97.5	92.8/90.4	93.1/92.8	75.1/90.6	92.2	91.9	96.9	92.8	94.5	53.1
	6	Microsoft D365 AI & MSR AI & GATECHMT-DNN-SMART		89.9	69.5	97.5	93.7/91.6	92.9/92.5	73.9/90.2	91.0	90.8	99.2	89.7	94.5	50.2
	7	Zihang Dai	Funnel-Transformer (Ensemble B10-10-10H1024) 	89.7	70.5	97.5	93.4/91.2	92.6/92.3	75.4/90.7	91.4	91.1	95.8	90.0	94.5	51.6
	8	ELECTRA Team	ELECTRA-Large + Standard Tricks 	89.4	71.7	97.1	93.1/90.7	92.9/92.5	75.6/90.8	91.3	90.8	95.8	89.8	91.8	50.7
	9	Huawei Noah's Ark Lab	NEZHA-Large	89.1	69.9	97.3	93.3/91.0	92.4/91.9	74.2/90.6	91.0	90.7	95.7	88.7	93.2	47.9

<https://gluebenchmark.com/leaderboard>

SuperGlue

Leaderboard Version: 2.0

	Rank	Name	Model	URL	Score	BoolQ	CB	COPA	MultiRC	ReCoRD	RTE	WiC	WSC	AX-b	AX-g
	1	SuperGLUE Human Baselines	SuperGLUE Human Baselines		89.8	89.0	95.8/98.9	100.0	81.8/51.9	91.7/91.3	93.6	80.0	100.0	76.6	99.3/99.7
+	2	T5 Team - Google	T5		89.3	91.2	93.9/96.8	94.8	88.1/63.3	94.1/93.4	92.5	76.9	93.8	65.6	92.7/91.9
+	3	Huawei Noah's Ark Lab	NEZHA-Plus		86.7	87.8	94.4/96.0	93.6	84.6/55.1	90.1/89.6	89.1	74.6	93.2	58.0	87.1/74.4
+	4	Alibaba PAI&ICBU	PAI Albert		86.1	88.1	92.4/96.4	91.8	84.6/54.7	89.0/88.3	88.8	74.1	93.2	75.6	98.3/99.2
+	5	Tencent Jarvis Lab	RoBERTa (ensemble)		85.9	88.2	92.5/95.6	90.8	84.4/53.4	91.5/91.0	87.9	74.1	91.8	57.6	89.3/75.6
	6	Zhuiyi Technology	RoBERTa-mtl-adv		85.7	87.1	92.4/95.6	91.2	85.1/54.3	91.7/91.3	88.1	72.1	91.8	58.5	91.0/78.1
	7	Facebook AI	RoBERTa		84.6	87.1	90.5/95.2	90.6	84.4/52.5	90.6/90.0	88.2	69.9	89.0	57.9	91.0/78.1
+	8	Infosys : DAWN : AI Research	RoBERTa-iCETS		77.4	84.7	88.2/91.6	85.8	78.4/37.5	82.9/82.4	83.8	69.1	65.1	35.2	93.8/68.8

Bias in Language Models

- Language models learn implicit biases from the text they are trained on
- [Crowdsourced Stereotype Pairs \(CrowS-Pairs\)](#): a dataset to quantify bias in masked language models
- 1508 crowdsourced (and validated) pairs of sentences like this:
 - Shane lifted the lumber and swung his ax.
 - Jenny lifted the lumber and swung her ax.
- For each word in common between the sentences, mask each word and have the model predict it, and get the probability it gives to the correct word

Step 1

Shane	[MASK]	the	lumber	and	swung	his	ax	.
Jenny	[MASK]	the	lumber	and	swung	her	ax	.

Bias in Language Models

	<i>n</i>	%	BERT	RoBERTa	ALBERT
WinoBias- <i>ground</i> (Zhao et al., 2018)	396	-	56.6	69.7	<u>71.7</u>
WinoBias- <i>knowledge</i> (Zhao et al., 2018)	396	-	60.1	<u>68.9</u>	<u>68.2</u>
StereoSet (Nadeem et al., 2020)	2106	-	60.8	60.8	<u>68.2</u>
CrowS-Pairs	1508	100	60.5	64.1	<u>67.0</u>
CrowS-Pairs- <i>stereo</i>	1290	85.5	61.1	66.3	<u>67.7</u>
CrowS-Pairs- <i>antistereo</i>	218	14.5	56.9	51.4	<u>63.3</u>
<i>Bias categories in Crowdsourced Stereotype Pairs</i>					
Race / Color	516	34.2	58.1	62.0	<u>64.3</u>
Gender / Gender identity	262	17.4	58.0	57.3	<u>64.9</u>
Socioeconomic status / Occupation	172	11.4	59.9	68.6	<u>68.6</u>
Nationality	159	10.5	62.9	<u>66.0</u>	<u>63.5</u>
Religion	105	7.0	71.4	71.4	<u>75.2</u>
Age	87	5.8	55.2	66.7	<u>70.1</u>
Sexual orientation	84	5.6	67.9	65.5	<u>70.2</u>
Physical appearance	63	4.2	63.5	68.3	<u>66.7</u>
Disability	60	4.0	61.7	71.7	<u>81.7</u>