Dimensionality Reduction and Representations

Web Science Lecture

8 March 2020

Sagnik Ray Choudhury
src@di.ku.dk
http://sagnik.in

UNIVERSITY OF COPENHAGEN

Organization

- Part 1: unsupervised dimensionality reduction: PCA.
- Part2: static word representation.
- Part3: contextualized word representation.
- Acknowledgements:
  - Jure Leskovec, Anand Rajaraman, Jeff Ullman(Mining of Massive Datasets)
  - Christopher Manning (Stanford CS224N)
  - Jonathon Shlens (A Tutorial on Principal Component Analysis)
  - Cosma Shalizi (Advanced Data Analysis from an Elementary Point of View)
  - Internet.

Announcement

- On **March 15** you will receive a link for the course evaluation in your KU e-mailbox.
- The deadline to complete the course evaluation is **March 28**.
- Please remember to fill out the course evaluation, even if you do not have anything wrong to report!
- Your feedback is very important to us, so we can understand how to improve this course for the next years.

What is data

- Data: MxN matrix: N is the number of features and M is the number of observations.

Gender	Height	weight
male	5’11”	180 lbs
female	5’4”	140 lbs
male	6’2”	240 lbs
female	5’10”	110 lbs
male	5’6”	170 lbs
female	5’8”	140 lbs

We want to predict the gender from height and weights (features)

There is some sort of a correlation between height and weight, but the data won’t directly tell you that.

Doc1: I go to school
Doc2: He goes to college
Doc3: To be or not to be
Doc4: College is exciting
Doc5: College is boring

Vocab: {i, go, to, school, he, college, be, or, not, is, exciting, boring}

We have lemmatized “goes” to go (look at the last lecture)

tf: frequency of a term in a document (document level statistic)
idf: inverse document frequency or *a function of* 1/(number of documents the term appears in the corpus (corpus level statistic).

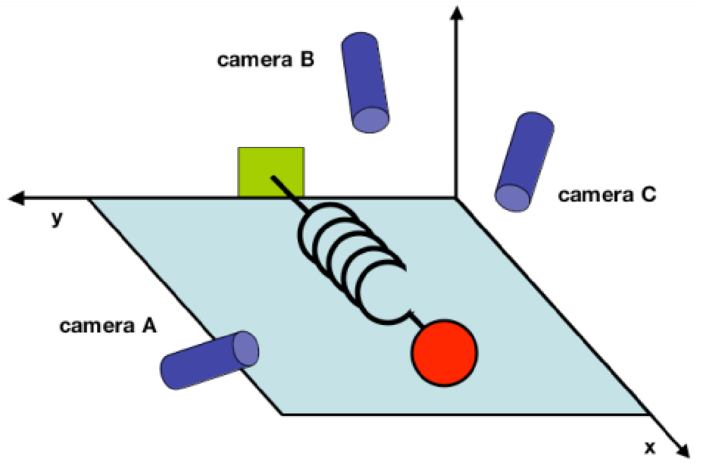
There are many variations of this concept, see Wikipedia for a comprehensive review.

The example below is computed using scikit-learn

	be	boring	college	exciting	go	he	i	is	not	or	school	to
Doc1	0	0	0	0	0.63907	0	0	0	0	0	0.63907	0.427993
Doc2	0	0	0.48624	0	0	0.726044	0	0	0	0	0	0.48624
Doc3	0.716388	0	0	0	0	0	0	0	0.358194	0.358194	0	0.479773
Doc4	0	0	0.462208	0.690159	0	0	0	0.556816	0	0	0	0
Doc5	0	0.690159	0.462208	0	0	0	0	0.556816	0	0	0	0

Do We Need All Dimensions?

- Data collection is inherently a noisy process.



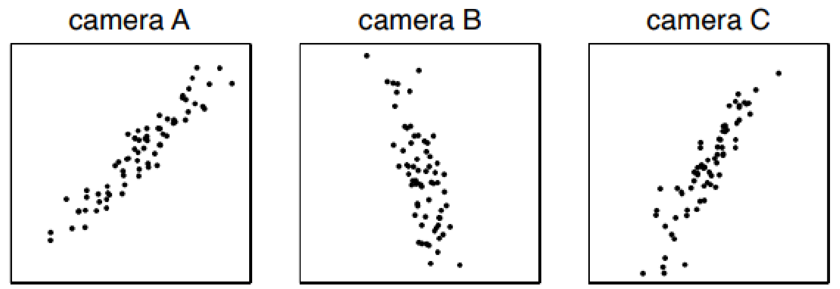


FIG. 1 A toy example. The position of a ball attached to an oscillating spring is recorded using three cameras A, B and C. The position of the ball tracked by each camera is depicted in each panel below.

We are trying to estimate a function for the motion of the spring.

The spring only moves in one direction (along the X axis), but the we (the observers) don't know that.

Therefore, we have placed three cameras in three arbitrary positions.

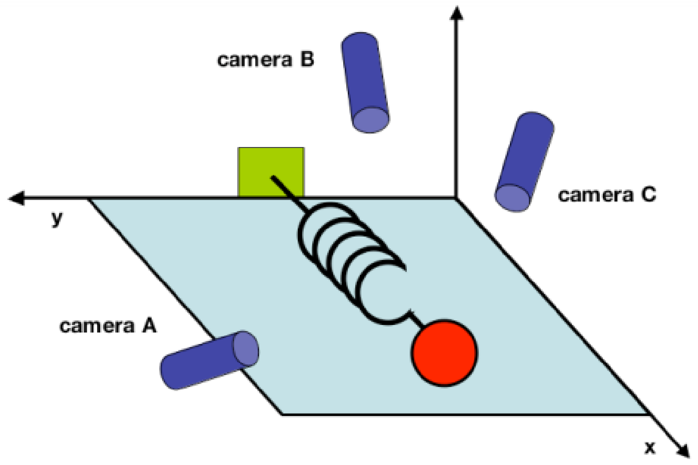
Now the data looks a bit like this:

	Camera A-X	Camera A-Y	Camera B-X	Camera C-X	Camera C-Y
data point 1	10	12	10	15	14
…	…	..	..	..	..
data point n	..	..	..	..	..

But the real data obviously lies at a *subspace* of these observations

Do We Need All Dimensions?

- Data collection is inherently a noisy process.



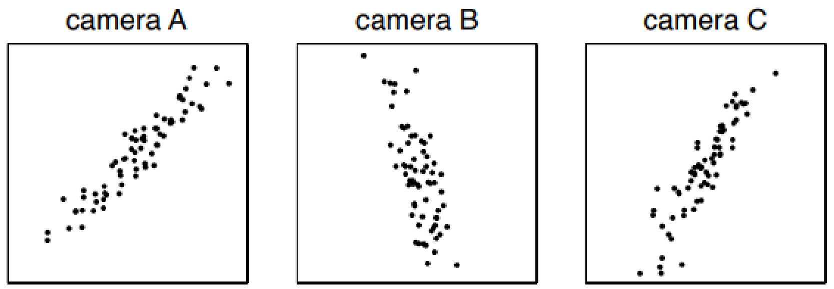


FIG. 1 A toy example. The position of a ball attached to an oscillating spring is recorded using three cameras A, B and C. The position of the ball tracked by each camera is depicted in each panel below.

https://arxiv.org/pdf/1404.1100.pdf

We are trying to estimate a function for the motion of the spring.

The spring only moves in one direction (along the X axis), but the we (the observers) don't know that.

Therefore, we have placed three cameras in three arbitrary positions.

Now the data looks a bit like this:

	Camera A-X	Camera A-Y	Camera B-X	Camera C-X	Camera C-Y
data point 1	10	12	10	15	14
…	…	..	..	..	..
data point n	..	..	..	..	..

We want to find out that ""the dynamics are along the x-axis."

Some Linear Algebra Reviews

- *Span* of a set of vectors is a set of all vectors you can write with linear combinations of them. Does the unit vectors ([1, 0], [0, 1]) span all 2-dimensional vectors?
- Suppose you have two vectors: {[1, 1], [5, 5]}. Can the span of all vectors be written with omitting one of them?
- A set of vectors $\vec{v_1}, \vec{v_2} ... \vec{v_k}$ are linearly dependent iff one vector in that set is in the span of the other ones (IOW, can be expressed as a linear combination of the others).
- If you add a vector to a set of linearly independent vectors and keep the set linearly independent, you must increase the span of the set.
- You can see how the linear independence is giving a measure of expressability and redundance.

This is not the definition of linear independence, just a property useful for out purpose.

Rank of a Matrix

- *So far*
  - Data can be represented as a matrix (row = observations, columns = features)
  - Linear independence: 1. One vector can be expressed as a linear combination of the others. 2. If you increase a set of vectors by destroying the linear independence property, redundancy is added.
- (Row) Column rank of a matrix: Number of linearly independent (row) columns in the matrix.
  - They are the same (why?)

$$\begin{bmatrix} 1 & 2 & 1 \\ -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$$

M1 (data matrix)

R1 = R2 + R3, let's consider them as "basis vectors"

$$\begin{bmatrix} -2 & -3 & 1 \\ 3 & 5 & 0 \end{bmatrix}$$

P (Projection matrix)

New representation of the data with these basis vectors

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}$$

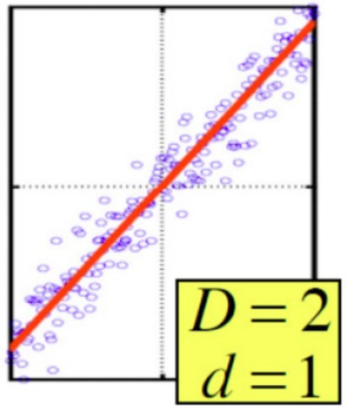
And we have reduced the dimensions!

M2: New representation. M2 x P = M1

Why Reduce Dimensions

- *So far*
  - Sometimes there's redundancy in the data.
  - We can reduce the dimensionality of the data by exploiting those redundancies.
  - This is a lossless reduction.
- Sometimes a lossy reduction can be helpful!
- If we find can project the data to the true representation, i.e., the properly aligned X axis, we will lose some info, but that's not bad!
- Also, more we reduce the data dimensions, the subsequent computations becomes easier.

	Camera A-X	Camera A-Y	Camera B-X	Camera C-X	Camera C-Y
data point 1	10	12	10	15	14
…	…	..	..	..	..



While the points are on 2 dimensions, reducing them to a single dimension (red line) won't be bad.

Principal Components Analysis

- *The general idea*
  - Start with p-dimensional vectors and *project them down* to q-dimensional subspaces.

Linear operation

- Can we just discard dimensions?

Gender	Height	weight
male	5’11”	180 lbs
female	5’4”	140 lbs
male	6’2”	240 lbs
female	5’10”	110 lbs
male	5’6”	170 lbs
female	5’8”	140 lbs

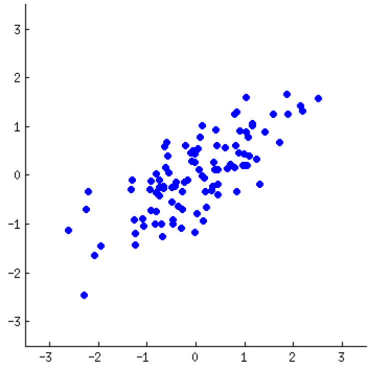
- If there’s a perfect correlation, i.e., you can express weight as c x height where c is a constant, yes you can.
- Why? Height and weight becomes linearly dependent vectors, you just need one of them to express the data.
- Most often it is not, therefore, we need to project them to another subspace that will somehow capture most of the information

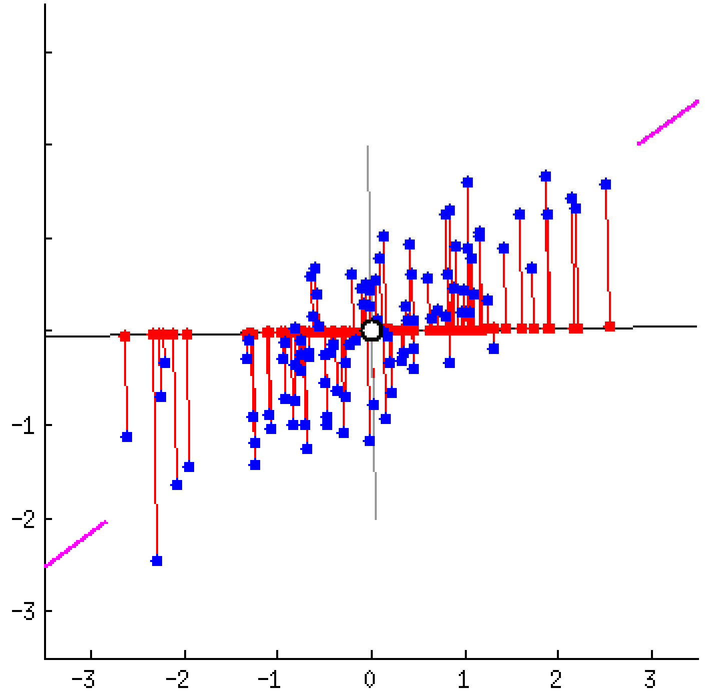
Turns out these things are the same.

What would we want in such projections?
- You want to find out a representation that varies greatly across different classes. For example, if I add a dimension of college and that is KU for all the data points, is that going to be very helpful?
- You want to minimize the reconstruction error.

A Visual Explanation

What would we want in such projections?
- You want to find out a representation that varies greatly across different classes. For example, if I add a dimension of college and that is KU for all the data points, is that going to be very helpful?
- You want to minimize the reconstruction error.

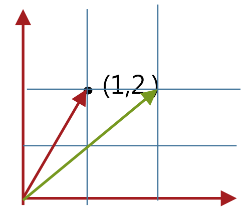




Excellent visual example from here
- Red dots are the projections of the blue dots
- See how the variance changes when you project.
- The distance is the red lines.
- What happens at the magenta line?

A Mathematical Proof

- The subspace should simultaneously maximize variance and minimize projection error.



A vector $\vec{x}$ (1,2) can be projected to another unit vector $\vec{w}(\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$. The new vector is $(\vec{x}.\vec{w})\vec{w}$

$$\begin{bmatrix} x_{00} & x_{01} & \cdots & x_{0n} \\ x_{10} & x_{11} & \cdots & x_{1n} \\ .. & & & \\ x_{m0} & x_{11} & \cdots & x_{mn} \end{bmatrix}$$

We will represent each column by $\vec{x_i}$. This has a dimension of m. There are n such vectors.

Assume that the data have been centered, i.e., mean ($\vec{x_i}$) is 0.

Let's project $\vec{x_i}$ along $\vec{w}$. What is the projection error?

$$\|\vec{x_i} - (\vec{x_i}.\vec{w})\vec{w}\|^2 \rightarrow \vec{x_i}.\vec{x_i} - (\vec{w}.\vec{x_i})^2$$

If we take a mean over all variables: $\boxed{\boxed{\frac{1}{n}\sum_{i=1}^{n}(\|\vec{x_i}\|^2} - (\vec{w}.\vec{x_i})^2)}$

Useless because doesn't depend on w

$$\boxed{\frac{1}{n}\sum_{i=1}^{n}(\vec{w}.\vec{x_i})^2}$$

This is what we want to maximize (because we are subtracting)

This is a sample mean of $(\vec{w}.\vec{x_i})^2$, which is (sample mean) square plus sample variance. In other words, $\left(\frac{1}{n}\sum_{i=1}^{n}\vec{x_i}\cdot\vec{w}\right)^2 + \boxed{\hat{\mathbb{V}}[\vec{w}\cdot\vec{x_i}]}$

Remember before projection we centered $\vec{x_i}$? Therefore, the mean of was $\vec{x_i}$ 0, therefore the mean of the projections is also 0.

Summary

- Reducing dimensions is important
- PCA: a process to find underlying the structure in the data by projecting it to a subspace so that variance is maximized, and reconstruction error is minimized.
    - Goal of the process is to find out the basis vectors of that subspace.
    - The first principal component is a vector on which the variance is maximized. The second principal component is the vector among all the vectors orthogonal to the first one on which the variance is maximized. The K-th one is the variance-maximizing vector orthogonal to the previous $k - 1$ components.
    - Why orthogonal? Orthogonal vectors are linearly independent. They together form a ``basis".
    - For MxN data matrix, there are n such dimensions. We will take the top K.
- Input: MxN matrix, output NxK matrix.
- But how do we actually do it? (SVD, to be discussed in the next lecture).

How does PCA Translate to NLP?

	be	boring	college	exciting	go	he	i	is	not	or	school	to
Doc1	0	0	0	0	0.63907	0	0	0	0	0	0.63907	0.427993
Doc2	0	0	0.48624	0	0	0.726044	0	0	0	0	0	0.48624
Doc3	0.716388	0	0	0	0	0	0	0	0.358194	0.358194	0	0.479773
Doc4	0	0	0.462208	0.690159	0	0	0	0.556816	0	0	0	0
Doc5	0	0.690159	0.462208	0	0	0	0	0.556816	0	0	0	0

So we have seen how PCA works.

How does that help us in NLP?

We want to find a "representation of a word": something that maps semantically similar words to similar vectors

- You can transpose this doc-term matrix to term-doc matrix.
  - Now the terms are represented in the doc space.
  - "A word is known by the company it keeps" [Firth, 1957]
- Using PCA, you can actually represent the subspace of the documents.
- One of the most common methods of term representation before DNNs did a variation of this: also known as latent semantic indexing.
  - The actual implementation is known as SVD, which we will talk about in the next lecture.

Word Representation

- Last lecture: words are represented as d dimensional real vectors.
- How do we create these representations?
  - Use a term document matrix and find out the document subspace.
- What could be a better representation?
- But first, a bit of neural network primer.
  - A bit of matrix calculus before

Gradient, Jacobian and Chain Rules

- One input, one output: $f(x) = x^2$ m gradient w.r.t. x is derivative of the function, which is x.
- Gradient is a measurement of how much the output changes when we change the input a bit.

$$f(\mathcal{R}^n \rightarrow \mathcal{R}^1), \vec{x} = [x_1..x_n], \frac{\partial f}{\partial \vec{x}} = [\frac{\partial f}{\partial x_1} \dots \frac{\partial f}{\partial x_n}]$$

$$f(\vec{x}) = \vec{x}.\vec{w}, \frac{\partial f}{\partial \vec{x}} = \vec{w}$$

$$f(\mathcal{R}^n \rightarrow \mathcal{R}^m)$$

$$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$\boxed{\left(\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{x}}\right)_{ij} = \frac{\partial f_i}{\partial x_j}}$$

$$\frac{\partial}{\partial \boldsymbol{x}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{W}$$

$$\frac{\partial}{\partial \boldsymbol{b}}(\boldsymbol{W}\boldsymbol{x} + \boldsymbol{b}) = \boldsymbol{I} \quad \text{(Identity matrix)}$$

$$\frac{\partial}{\partial \boldsymbol{u}}(\boldsymbol{u}^T\boldsymbol{h}) = \boldsymbol{h}^T$$

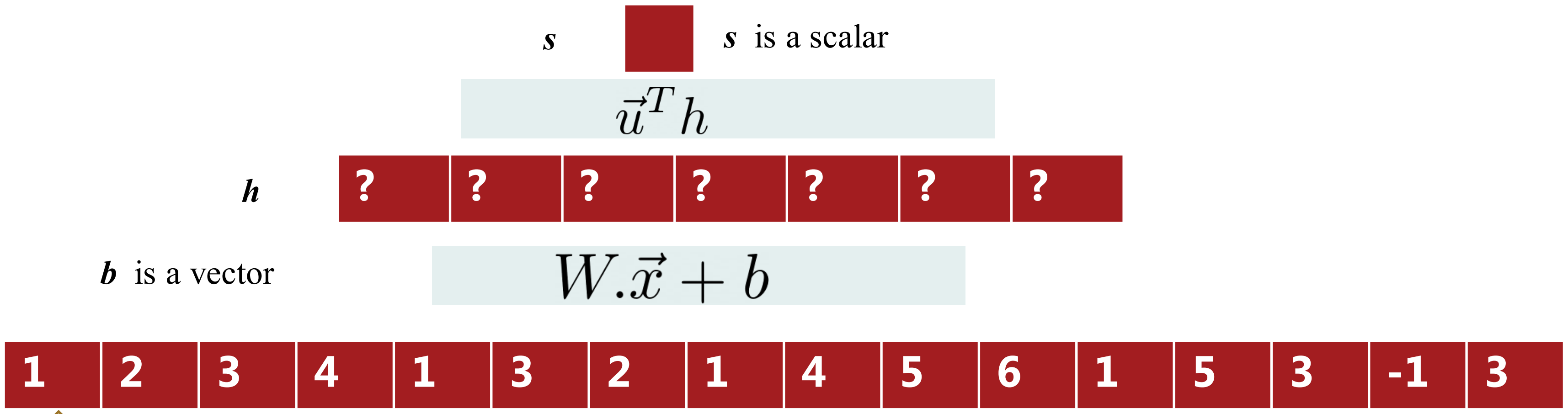
I is n x n where n is the dimension of b

- Chain rule holds: $z = f(\vec{y}), y = f(\vec{x}), \frac{\partial z}{\partial \vec{x}} = \frac{\partial z}{\partial \vec{y}} \frac{\partial \vec{y}}{\partial \vec{x}}$

Let’s Apply This to the Neural Nets

- Sentiment classification

this	1.	2.	3.	4.
movie	1.	3.	2.	1.
is	4.	5.	6.	1.
good	5.	3.	-1.	3.



Very basic word to sentence representation

We are trying to find out:

$$\frac{\partial s}{\partial \vec{b}} \qquad \frac{\partial s}{\partial \vec{u}} \qquad \frac{\partial s}{\partial W}$$

Not really, s is not the loss function, but we will go there later.

- The last step gives you the output. If it is > 0, we will say the sentiment is positive, if < 0, we will say the sentiment is negative.

- Based on this, we will design a loss function.

- Our goal would be to find out the W, b and u (parameters) that minimize the loss on a sample of training data.

- This is a very *imprecise* formulation of the problem, but I am using it to show the gradient calculation.

Let's Apply This to the Neural Nets

We are trying to find out: $\frac{\partial s}{\partial \vec{b}}$

$$\frac{\partial s}{\partial b} = \underset{u^T}{\underset{\downarrow}{\frac{\partial s}{\partial h}}}\ \underset{I}{\underset{\downarrow}{\frac{\partial h}{\partial b}}} \longrightarrow u^T$$

Removing the arrows (it is clear from the context what is a vector and what is a scalar)

$$s = u^T h$$

$$h = Wx + b$$

$$x(input)$$

Let's look at the dimensions:
input = 1 x 16
h = 1 x 7. W must be 16 x 7, b must be 1 x 7 u^T must be 7x1

Let’s Apply This to the Neural Nets

We are trying to find out: $\frac{\partial s}{\partial W}$

$$\frac{\partial s}{\partial W} = \frac{\partial s}{\partial h}\frac{\partial h}{\partial W}$$

Removing the arrows (it is clear from the context what is a vector and what is a scalar)

$$s = u^T h$$

$$h = Wx + b$$

$$x(input)$$

This is where the dimensions get tricky. We have $W = \mathcal{R}^{n \times m}$ We can flat out the dimensions and think $s = f(W), \mathcal{R}^{n \times m} \rightarrow \mathcal{R}^1,$ so should we have a (1 x nm) Jacobian?

This will cause inconvenience later, therefore we will adapt a convention that shape of the gradient is the shape of the parameter, so,

$\frac{\partial s}{\partial \boldsymbol{W}}$ is *n* by *m*: $\begin{bmatrix} \frac{\partial s}{\partial W_{11}} & \cdots & \frac{\partial s}{\partial W_{1m}} \\ \vdots & \ddots & \vdots \\ \frac{\partial s}{\partial W_{n1}} & \cdots & \frac{\partial s}{\partial W_{nm}} \end{bmatrix}$

In general, there is a huge mess in aligning the dimensions, so always use the shape convention, i.e., calculate the gradients and then transpose to make sure the dimensions match.

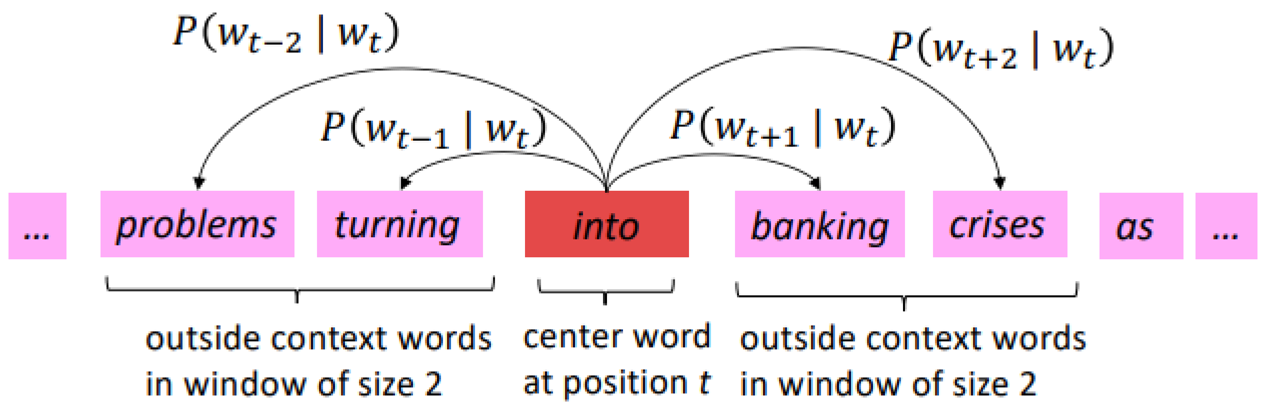
Putting This All Together

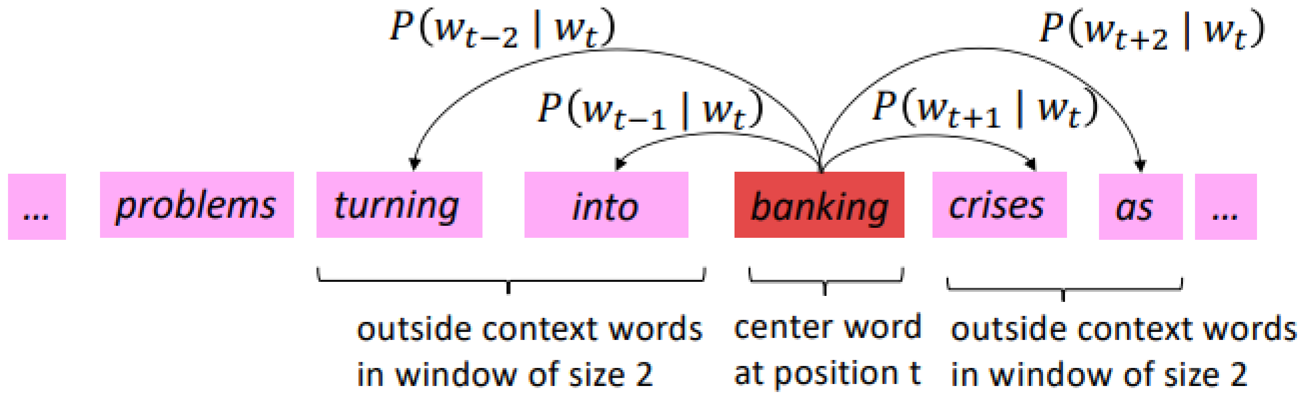
- In the last week we saw a convolutional neural network for sentence representation.
  - How do we find the value of the kernels?
- Any neural net is a combination of some matrices W and b (and some nonlinearity functions, but we won't go there for now).
- How do we find the values for W and b?
- The basic structure: define a loss function and minimize it over the training examples (start with random values for the parameters).
- A process for this is gradient descent: $\theta^{(t+1)} = \theta^{(t)} - \alpha \nabla_{\theta^{(t)}} J$
- What we have seen
  - What is a parameter.
  - How to calculate gradients (w.r.t. these parameters).
- We want to find out the parameters that would minimize the training loss.
- There are many variations of gradient descents, we are not discussing them.
- We are also not discussing back propagation in details (may be on Monday?)

Word Representation: Word2Vec

- Last lecture: words are represented as d dimensional real vectors.
- Words are represented by their contexts.
- Suppose you are given a word; can you predict the words that will appear in the context?
  - Money -> DKK
  - Course -> Web Science
- Let’s turn this idea: how can we represent a word such that it is able to predict the words in the context with reasonable accuracy?
- Iow, we want to define a function that will take two word representations and provide a probability that these words appear close to each other.
  - The word representations are itself the parameters of the model.
  - We want to identify optimal parameters.

Word2Vec Example





$$P(w_{t+j} | w_t)$$

- We will define the probability value as a function of the representations of the words.
- At any point, we will ask the network to predict a context word given a central word.
- Is that prediction correct?
  - Great
  - No: possibly need to get a better representation
- IOW, we will define a loss function on this prediction capability

Word2Vec: What Do We Want to Maximize/Minimize

- From the last slides: parameters are iteratively improved to minimize a loss value.
- What are we trying to do here? Maximizing the data likelihood:

$$\prod_{t=1}^{T} \prod_{\substack{-m \le j \le m \\ j \ne 0}} P(w_{t+j} \mid w_t; \theta)$$

- Equivalent of minimizing the negative log likelihood

$$J(\theta) = -\frac{1}{T} \log L(\theta) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{\substack{-m \le j \le m \\ j \ne 0}} \log P(w_{t+j} \mid w_t; \theta)$$

What is T? all the positions you move the window over

Word2Vec: How to Define the Probabilities?

- How do we approximate $P(w_{t+j}|w_t;\theta)$
- Two vectors per word w.
  - v_w when w is a center word
  - u_w when w is a context word.

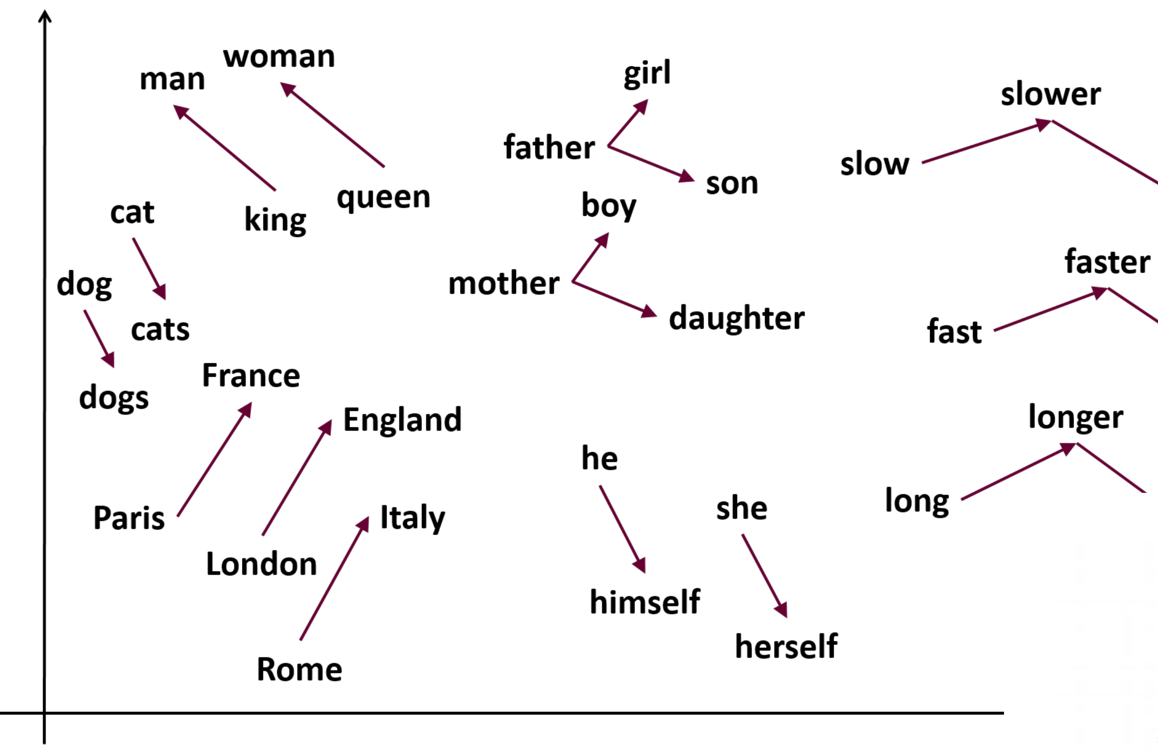
to	be	or	not	to	be
P(to|be)	P(be|or)				
$P(u_{to}\|v_{be})$	$P(u_{be}\|v_{or})$				

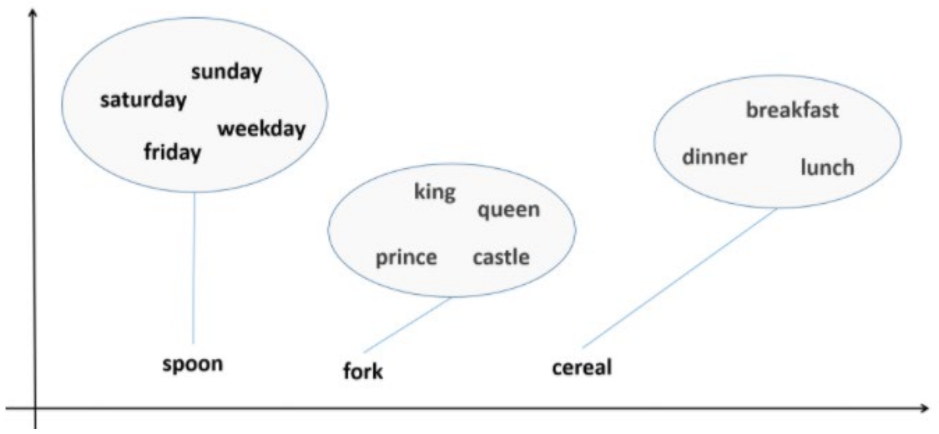
- For a center word c and a context word o, $P(o|c) = \dfrac{\exp(u_o^T v_c)}{\sum_{w \in V} \exp(u_w^T v_c)}$
- Numerator: similarity of o and c.
- Denominator: Normalize over the entire vocabulary.
- This is a very common function in neural nets, also known as softmax.
  - Max because amplifies the probability of largest x_i.
  - Soft because still assigns some probability to smaller x_i.

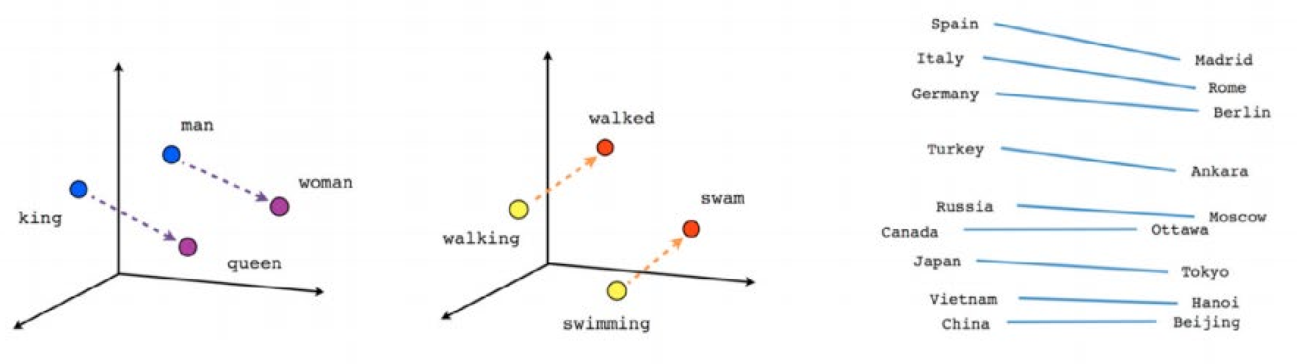
How Do We Train This?

- Minimizing the negative loss using gradient descent (stochastic)
- What is the parameter?
- Two model variants:
  - Predict context words given center word (skip gram)
  - Predict center word from bag of context words (CBOW)
- Negative sampling for better efficiency
- Finally, we have a continuous representation of a word utilizing the context: words that appear in the similar context have similar representation.

Are These Representations Good?







Male-Female Verb tense Country-Capital

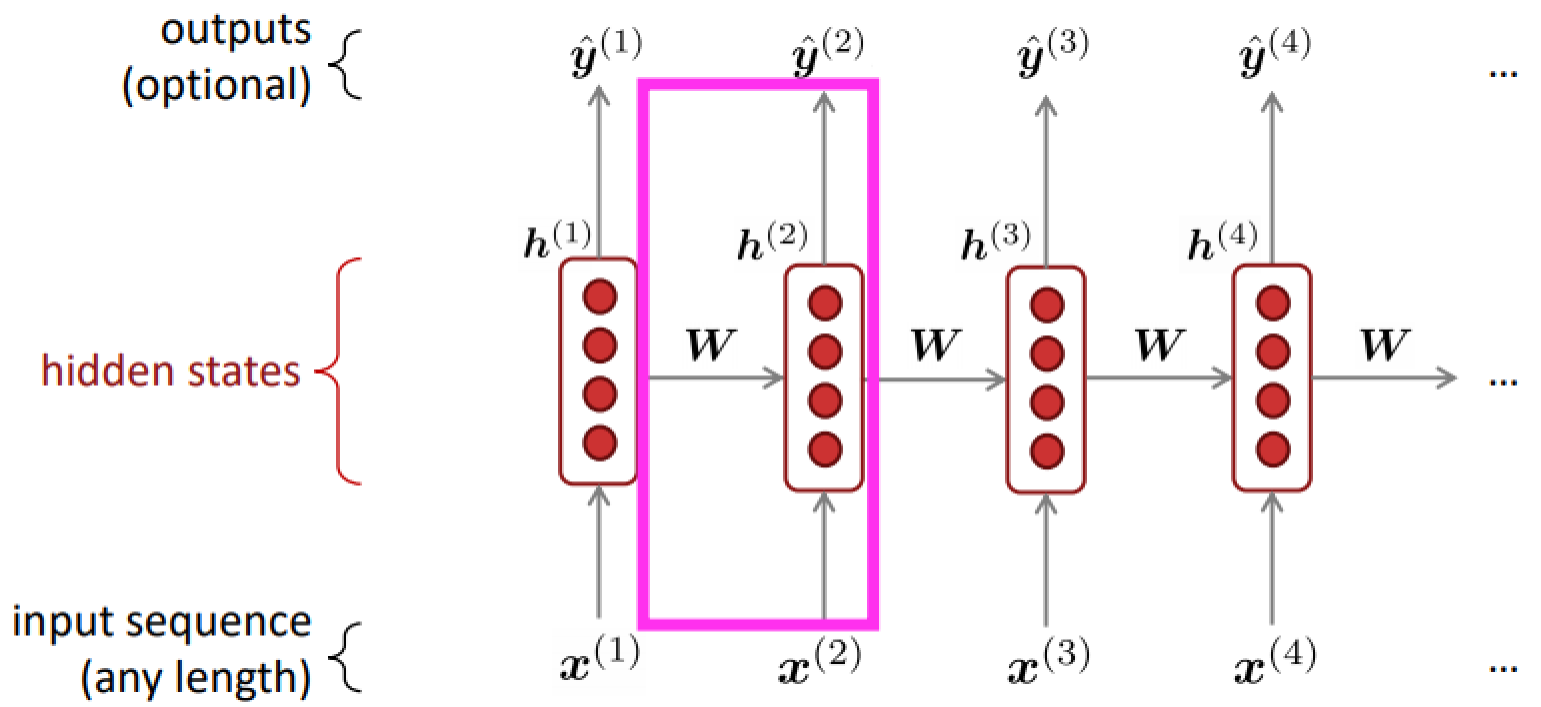
Contextual Word Embeddings

- We will call these representations “embeddings”.
- However, these representations are static, i.e., they don’t change depending on the context.
- But that’s not true for language!
  - We went to the river **bank** to enjoy the nature.
  - We went to the **bank** to borrow some money!
- So what do we do?
  - Language modeling and RNNs.

Language Modeling

- Simply, the task of predicting what word comes next in a sequence.
  - He put on the headphone to __ (listen/ walk/ cook/)
  - He put on the headphone to listen __ (to/ music)
  - He put on the headphone to listen to __ (music/ himself)
- Formally: what is the x for which $P(x_{t+1}|x_t...x1)$ is maximized, where x is drawn from a vocabulary. $x_t \in V = \{w_1..w_v\}$
- The language model should provide $P(x_{t+1}|x_t..x_0)$
- As you can imagine, a representation for words that helps us to predict this should be a good representation.
- There are many ways of doing this, we will look at a particular architecture.

Recurrent Neural Nets



$$\boldsymbol{h}^{(t)} = f(\boldsymbol{h}^{(t-1)}, \boldsymbol{x}^{(t)}; \boldsymbol{\theta})$$

Whatever information was encoded by seeing something till x1 is now used along with the new information that's coming from x2.

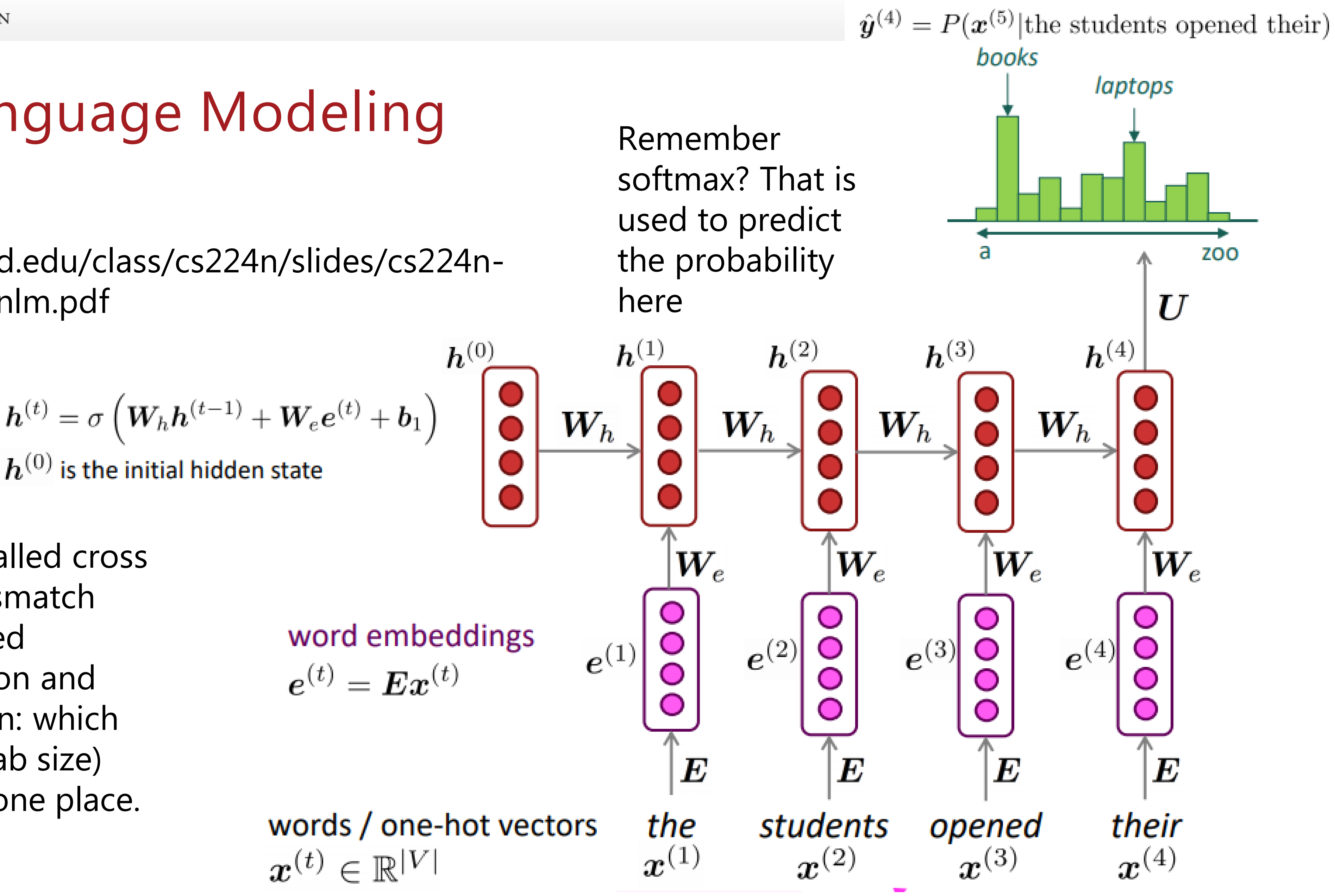
We are using the same weight matrix W, otherwise the number of parameters would explode.

RNNs for Language Modeling

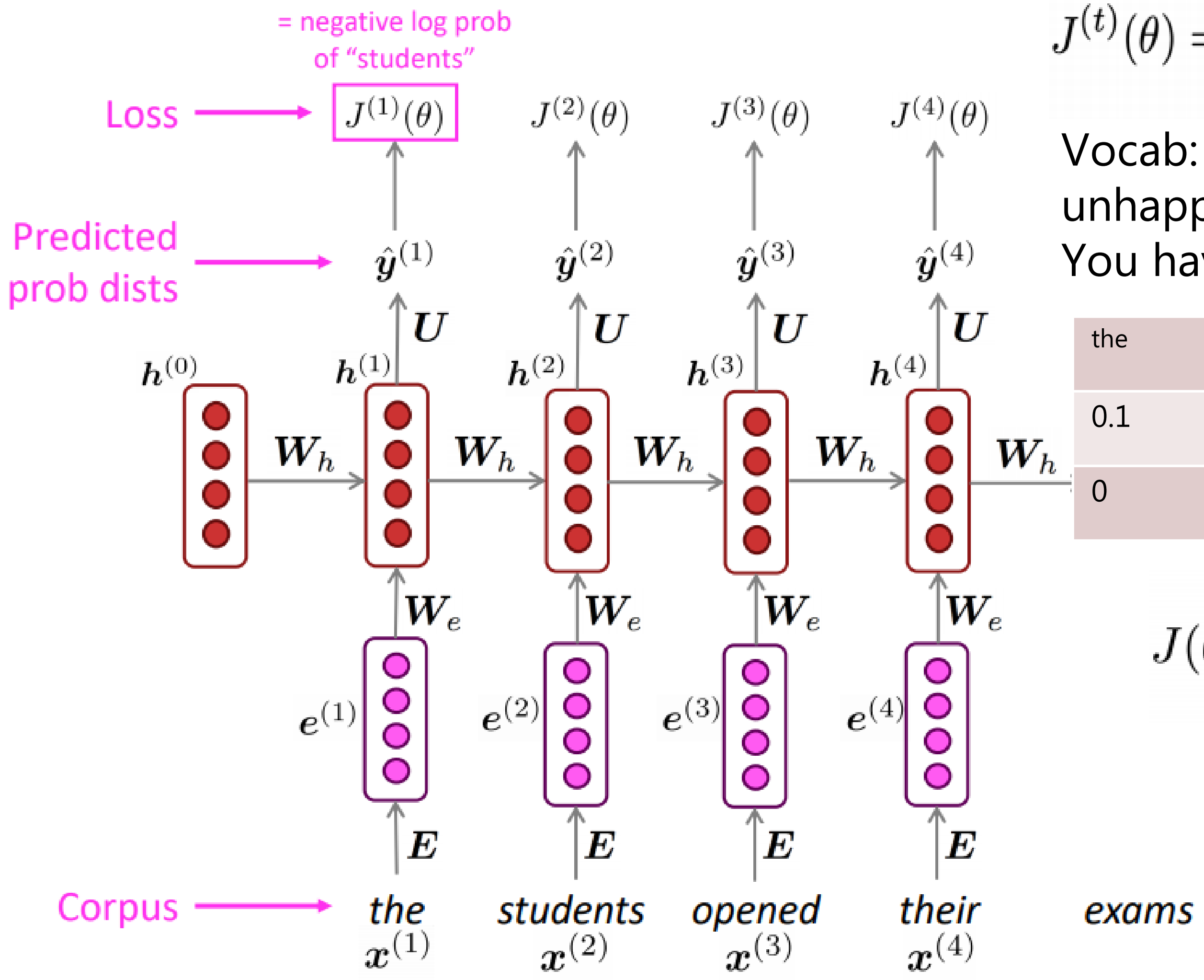
http://web.stanford.edu/class/cs224n/slides/cs224n-2021-lecture05-rnnlm.pdf

Remember softmax? That is used to predict the probability here

We use something called cross entropy loss: The mismatch between the predicted probability distribution and the actual distribution: which has shape 1 x V (vocab size) and all 0 except the one place.



RNN Training Considerations


= negative log prob of "students"
Loss
Predicted prob dists
Corpus
$J^{(1)}(\theta)$ $J^{(2)}(\theta)$ $J^{(3)}(\theta)$ $J^{(4)}(\theta)$
$\hat{y}^{(1)}$ $\hat{y}^{(2)}$ $\hat{y}^{(3)}$ $\hat{y}^{(4)}$
U
$h^{(0)}$ $h^{(1)}$ $h^{(2)}$ $h^{(3)}$ $h^{(4)}$
W_h
W_e
$e^{(1)}$ $e^{(2)}$ $e^{(3)}$ $e^{(4)}$
E
the students opened their exams
$x^{(1)}$ $x^{(2)}$ $x^{(3)}$ $x^{(4)}$

minimize maximize

$$J^{(t)}(\theta) = CE(\boldsymbol{y}^{(t)}, \hat{\boldsymbol{y}}^{(t)}) = -\sum_{w\in V} \boldsymbol{y}_w^{(t)} \log \hat{\boldsymbol{y}}_w^{(t)} = -\log \boxed{\hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}}$$

Vocab: {the, students, opened, their, exams, and, was, unhappy}
You have seen ``the" so you are calculating $J^{(1)}(\theta)$

the	students	opened	their	exams	and	was	unhappy
0.1	0.5	0.1	0.05	0.05	0.04	0.09	0.07
0	1	0	0	0	0	0	0

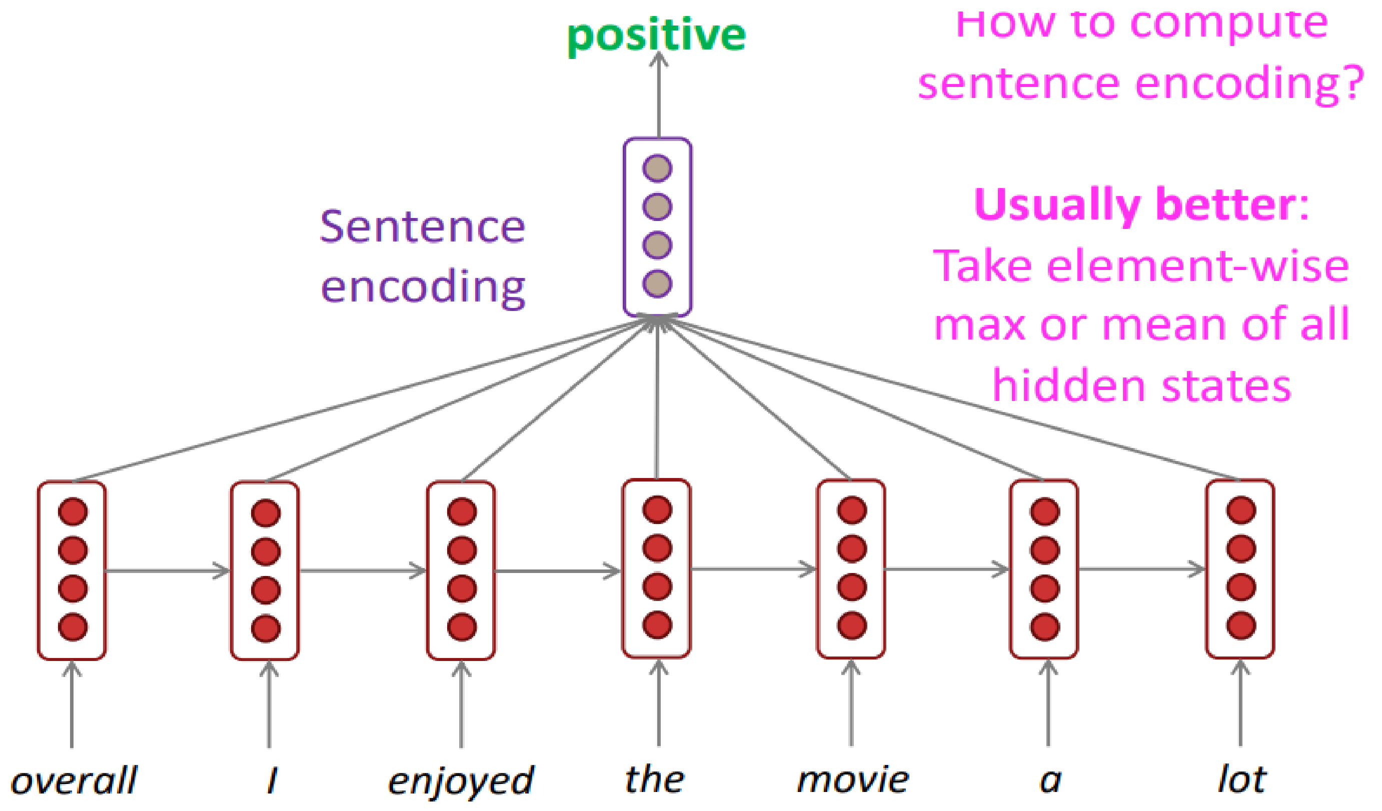
$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T} J^{(t)}(\theta) = \frac{1}{T}\sum_{t=1}^{T} -\log \hat{\boldsymbol{y}}_{\boldsymbol{x}_{t+1}}^{(t)}$$

Theoretically, you can do this over the whole corpus but then would be hard to compute. Limit T to sentence.

Compute loss for a sentence a batch of sentences, compute gradients and update weights. Repeat.

How to Use RNNs for Our Purpose?

- A brief detour: sentence encoding
- Last lecture: sentence encoding through convolutional neural nets.


positive
Sentence encoding
How to compute sentence encoding?
Usually better: Take element-wise max or mean of all hidden states
overall
I
enjoyed
the
movie
a
lot

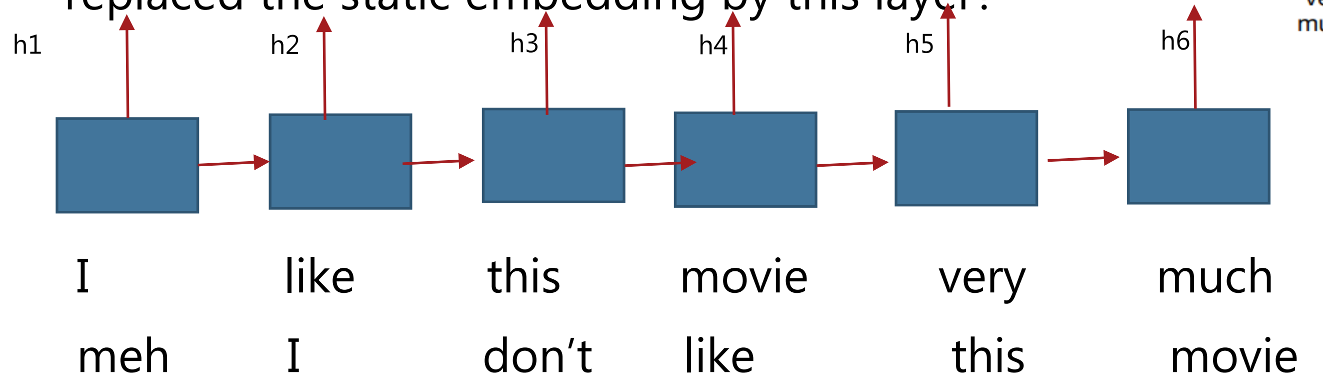
http://web.stanford.edu/class/cs224n/slides/cs224n-2021-lecture05-rnnlm.pdf

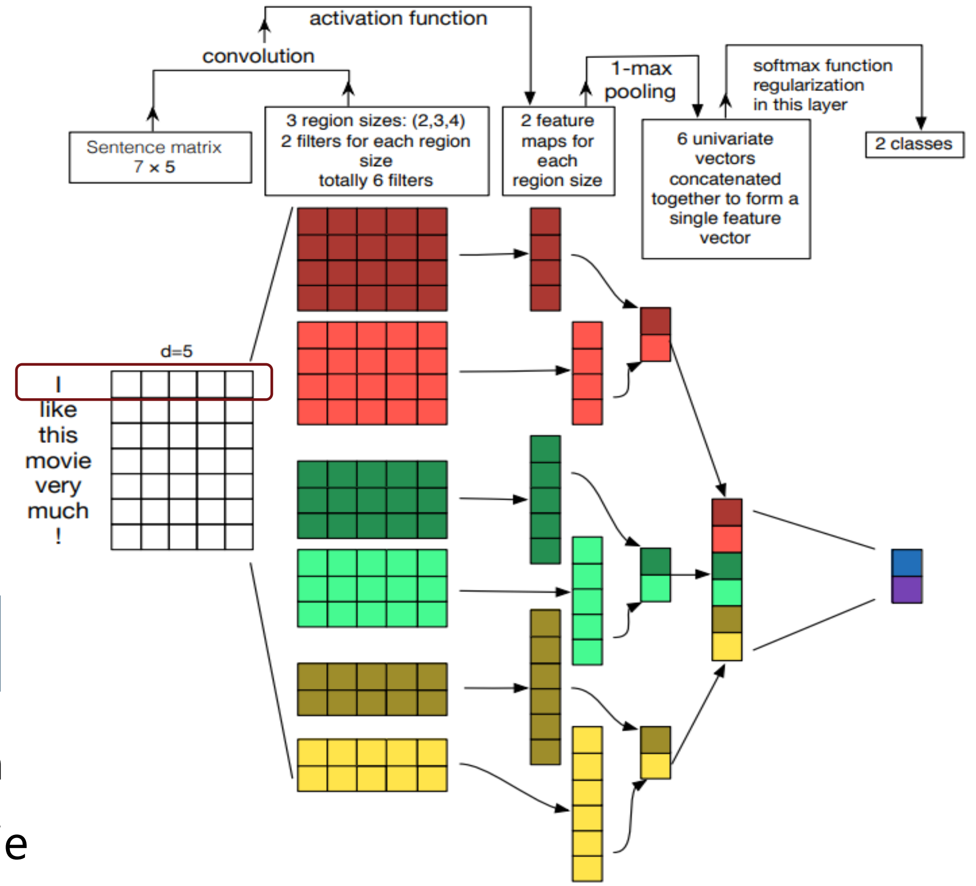
How to Use RNNs for Our Purpose?

- Remember we are trying to get contextual embeddings, i.e., for the same word ``bank", you can have two representation depending on the context (sentence) it was used.
- Remember our classification setup?

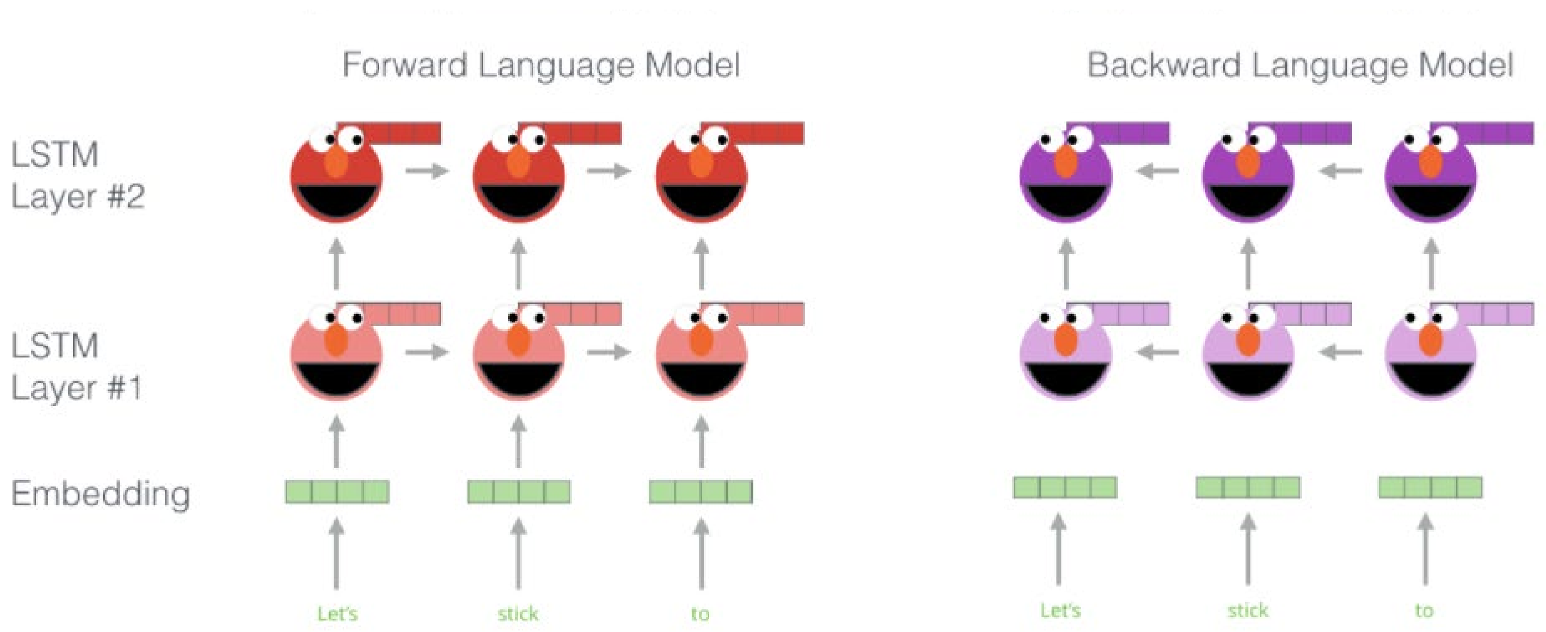
Static embedding, the representation for "like" was fixed to begin with.

Let's think of the RNN as a layer: You have already trained it, i.e., W(s) and U is calculated. What if we replaced the static embedding by this layer?




activation function
convolution
1-max pooling
softmax function regularization in this layer
Sentence matrix 7 × 5
3 region sizes: (2,3,4) 2 filters for each region size totally 6 filters
2 feature maps for each region size
6 univariate vectors concatenated together to form a single feature vector
2 classes
d=5
I
like
this
movie
very
much
!

Elmo

- An advanced version of this happens in Elmo.
- 

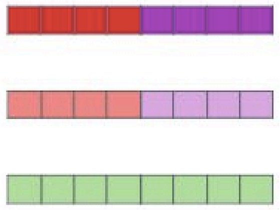
Elmo specifics

1. Character based
2. LSTM
3. Bi-directional
4. Multiple layers

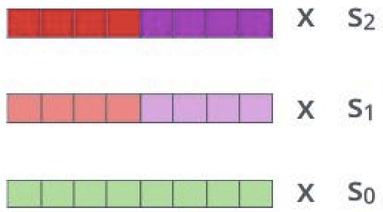
Elmo: Specifics

Embedding of “stick” in “Let’s stick to” - Step #2

1- Concatenate hidden layers

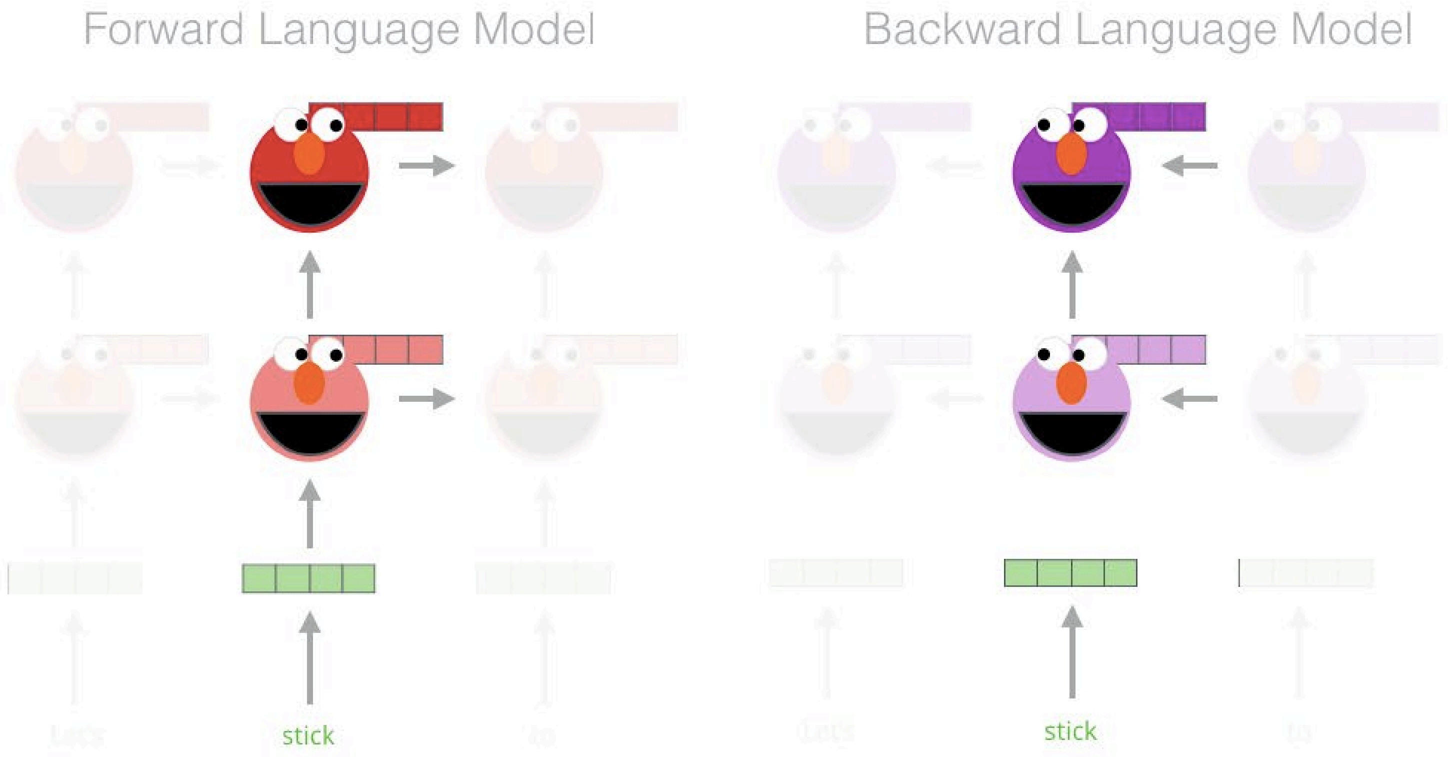


2- Multiply each vector by a weight based on the task



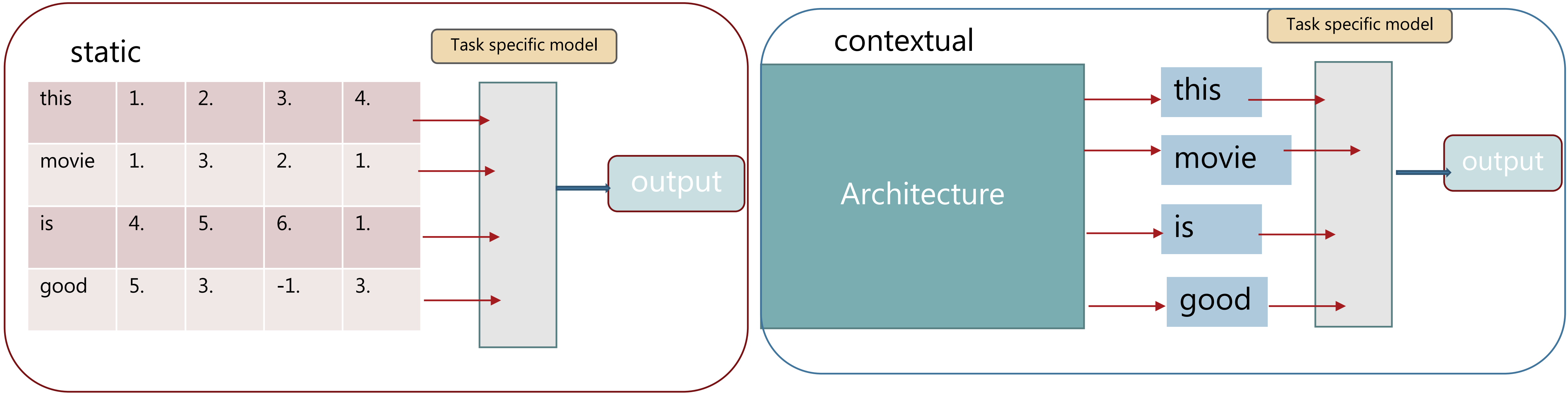
3- Sum the (now weighted) vectors

ELMo embedding of “stick” for this task in this context



Contextual Word Embeddings

- Instead of static vectors for words, load the whole representation.



this	1.	2.	3.	4.
movie	1.	3.	2.	1.
is	4.	5.	6.	1.
good	5.	3.	-1.	3.

- Depending on task, the internal weights in the representation architecture will change.
- This is the new direction in NLP research!

Summary

- Language models and RNNs
- Static embeddings are not enough, you need context!
- We have not discussed transformers: which is one of the most exciting contextual word representations. Hopefully later!

Attend the QA session on Monday for clarifications and interesting discussions!