CSCI 566: Deep Learning and Its Applications

Jesse Thomason

Lecture 6: DL for Natural Language Processing

CSCI 566 Roadmap

Module 1:
Neural Network Basics

Module 2:
Deep Learning Applications

Module 3:
Advanced Topics in Deep Learning

January 2023

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	2	3	4	5	6	7
8	9	10	11	12	13	14
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31				

February 2023

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28				

March 2023

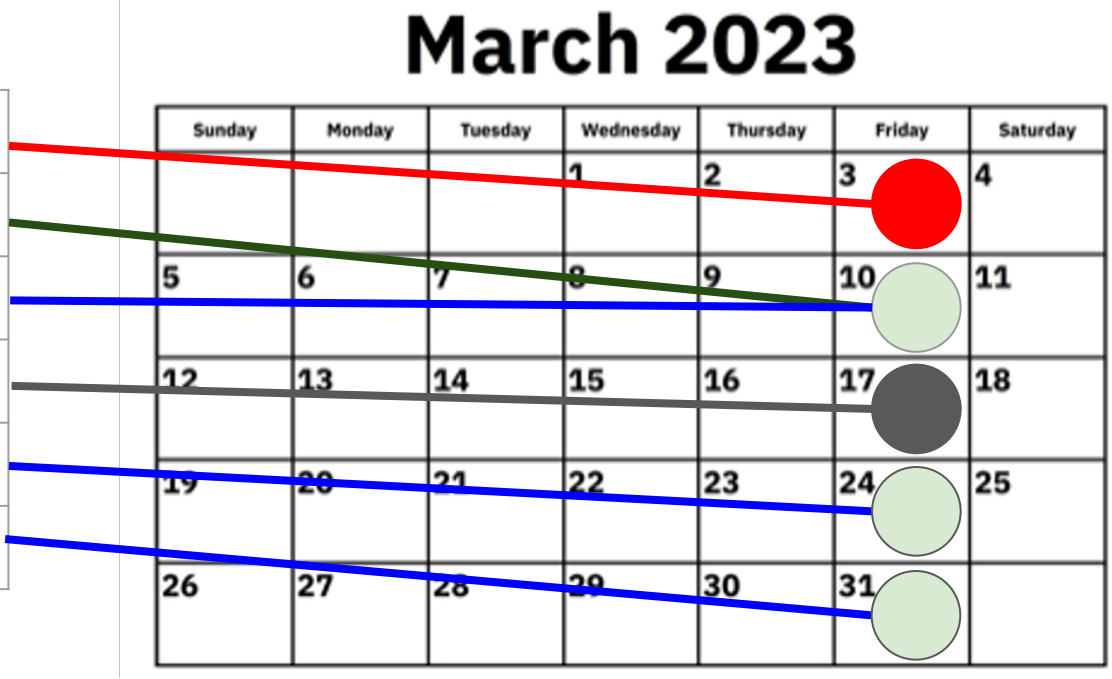
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
			1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	

April 2023

Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
						1
2	3	4	5	6	7	8
9	10	11	12	13	14	15
16	17	18	19	20	21	22
23	24	25	26	27	28	29
30						

[Mar 3] March Deliverables

Project Pitch	Mar 3
Assignment 2 OUT	Mar 10
Project Survey Report	Mar 10
NO CLASS	Mar 17
Assignment 2 Due	Mar 24
Project Midterm Report	Mar 31



Assignment 1 Survey

- “[Poll] How Long You Spent on Assignment 1”
  - https://piazza.com/class/lcpa44ep1pk5aj/post/196
- Please fill this poll on Piazza; it’s in the pinned posts section
- Since this is our first time running the class, we’re calibrating
  - For assignment 2, for weighting of coding assignment points versus exams/project/etc.

Survey Reports Due March 10 [Friday, 11:59pm]

- A literature survey (e.g., "Related Work" section in a conference paper)
- (1) What has been done related to your proposal?
  - This may comprise multiple paragraphs/sections of different related areas and efforts
- (2) What are the limitations or challenges remaining to be solved?
  - What are the open questions? At least one of these should be what your project aims to address, though you should highlight others too.
- Format: a 1.5-2 page (double-column) literature review write-up.
- Additionally, a single page summary of contributions from each team member on your project.
  - Spell out your individual contributions clearly on that final page.

Midterm

- The exam (primary and make-up versions) is still being graded
- We will have an exam *debrief* session next class [Mar 10]
  - Will cover grade distribution, common mistakes, any changes to grading made post-release, etc.
  - Opportunity for Q&A for misunderstandings etc., but grades will be considered final at that time
- Grades for the midterm (and assignment 1, etc.) will be posted to Blackboard

Midterm [Slide from Midterm Briefing; Feb 24; **emph** new]

- The exam will be in the form of a link to a PDF (exam questions) and a link to a Google Form (to submit answers)
- The exam is entirely multiple choice
- The exam is open book / open note, but **not open collaborator; please complete it on your own without assistance from other students or other agents**
- The exam PDF and Google Form will be available *only* during the 1:20pm-3:20pm window; links will go up on Piazza
- I will hang out up here in case anything goes egregiously wrong or you find super confusing typos

Midterm

- **“please complete it on your own without assistance from other students or other agents”**
- If you are already aware that you violated this policy, please contact me so that we can work through the reporting procedures to the Office of Academic Integrity and the associated grade penalty
- Else, I will be contacting you.

Project Pitches

- If you haven’t added your slides to the main project pitch deck, please do so ASAP!
- Pitches will take place in the last 1.25 hours of so of class
- Main pitch deck link [here]
  - Do NOT overwrite other teams’ slides!
  - Your team name should be *extremely visible* on first slide

Overview of Today's Plan

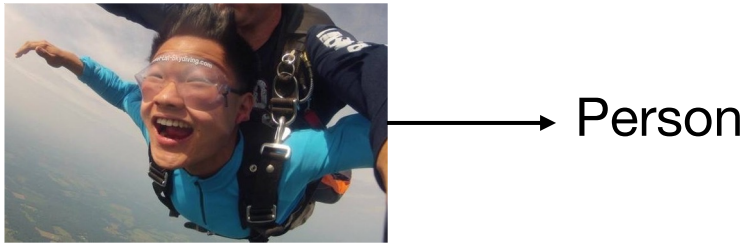
- ~~Course organization and deliverables~~
  - Any questions before we move on?
- Recap from Lecture 5
- Language Models and Word Embeddings
- Project Pitches

Natural Language Processing in the Wild

- Hype?
  - ChatGPT
  - Bing chat
- Mundane but actually useful*?
  - Autocomplete
  - Grammar Corrections
  - Assistants: Google, Amazon Alexa, Apple Siri (well, *)

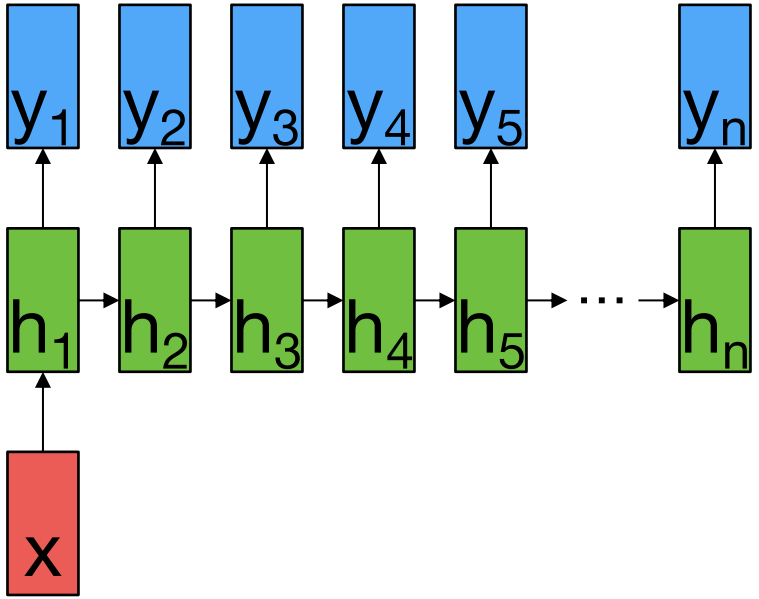
Classes of Sequence-to-Sequence Problems

- One-to-one
  - One input produces one output
    - Image classification



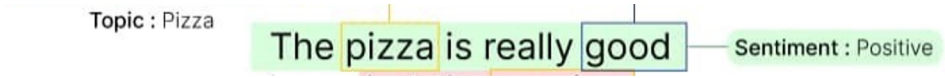
Classes of Sequence-to-Sequence Problems

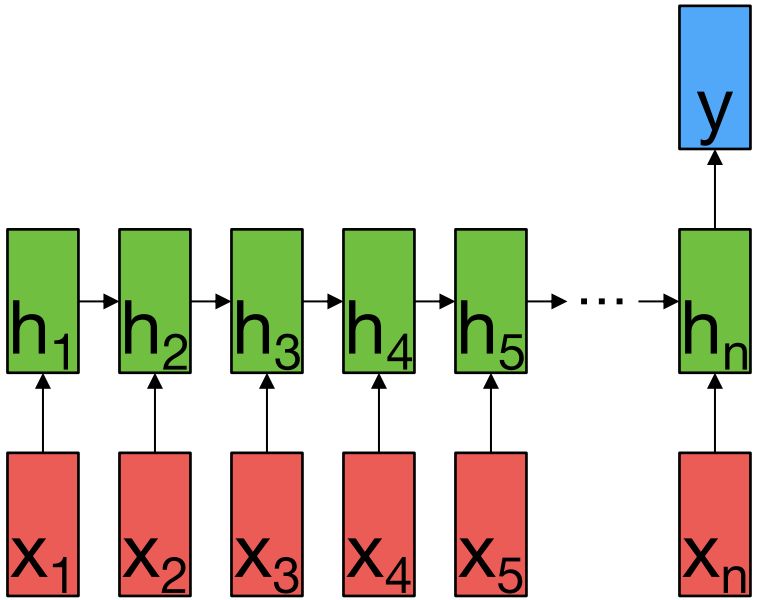
- One-to-many
  - One input produces a sequence of outputs
    - Image captioning



Classes of Sequence-to-Sequence Problems

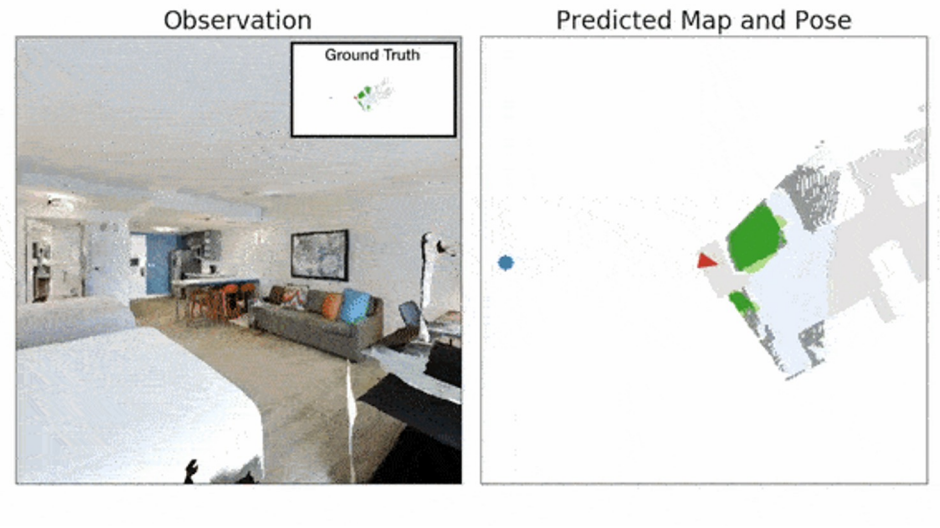
- Many-to-one
  - Sequence of inputs produces a single output
    - Text classification

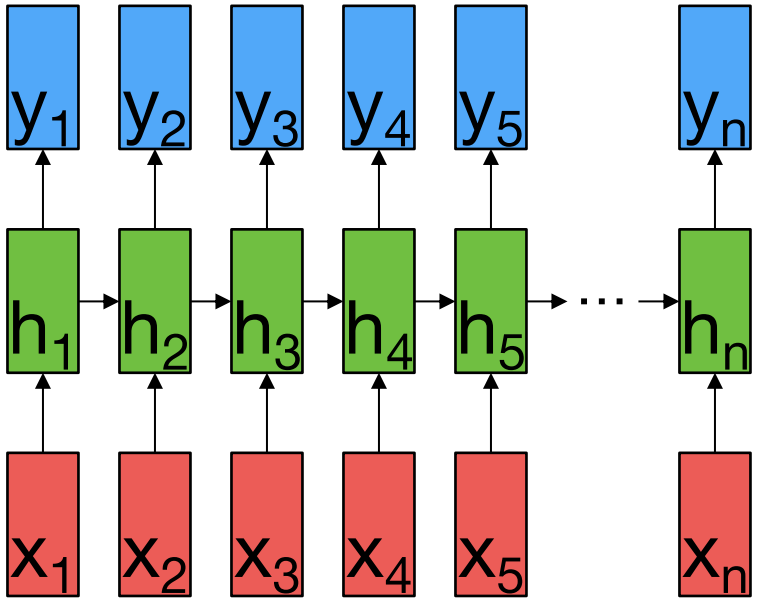




Classes of Sequence-to-Sequence Problems

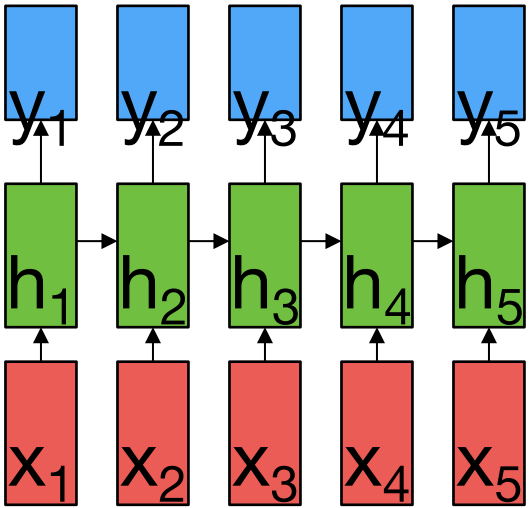
- Many-to-many
  - A sequence of inputs produces a sequence of outputs
    - Robot Actions



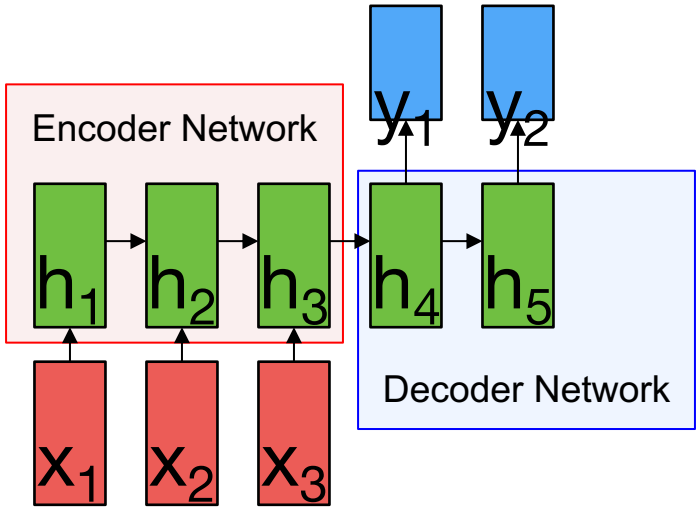


Classes of Sequence-to-Sequence Problems

- Many-to-many variants:
  - Sequential one-to-one
    - Robot actions

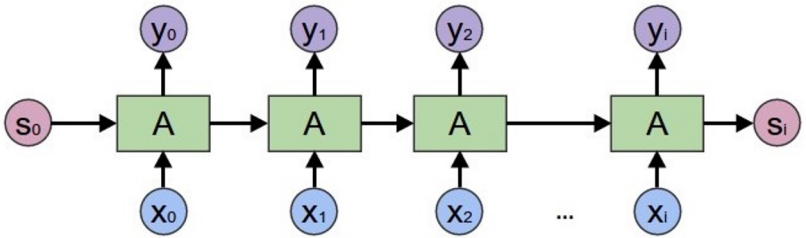


  - Encoder-decoder
    - Machine Translation



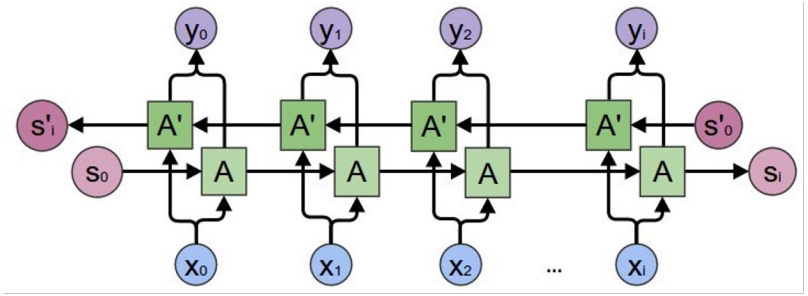
Unidirectional and Bidirectional Seq2Seq Models

Forward RNN



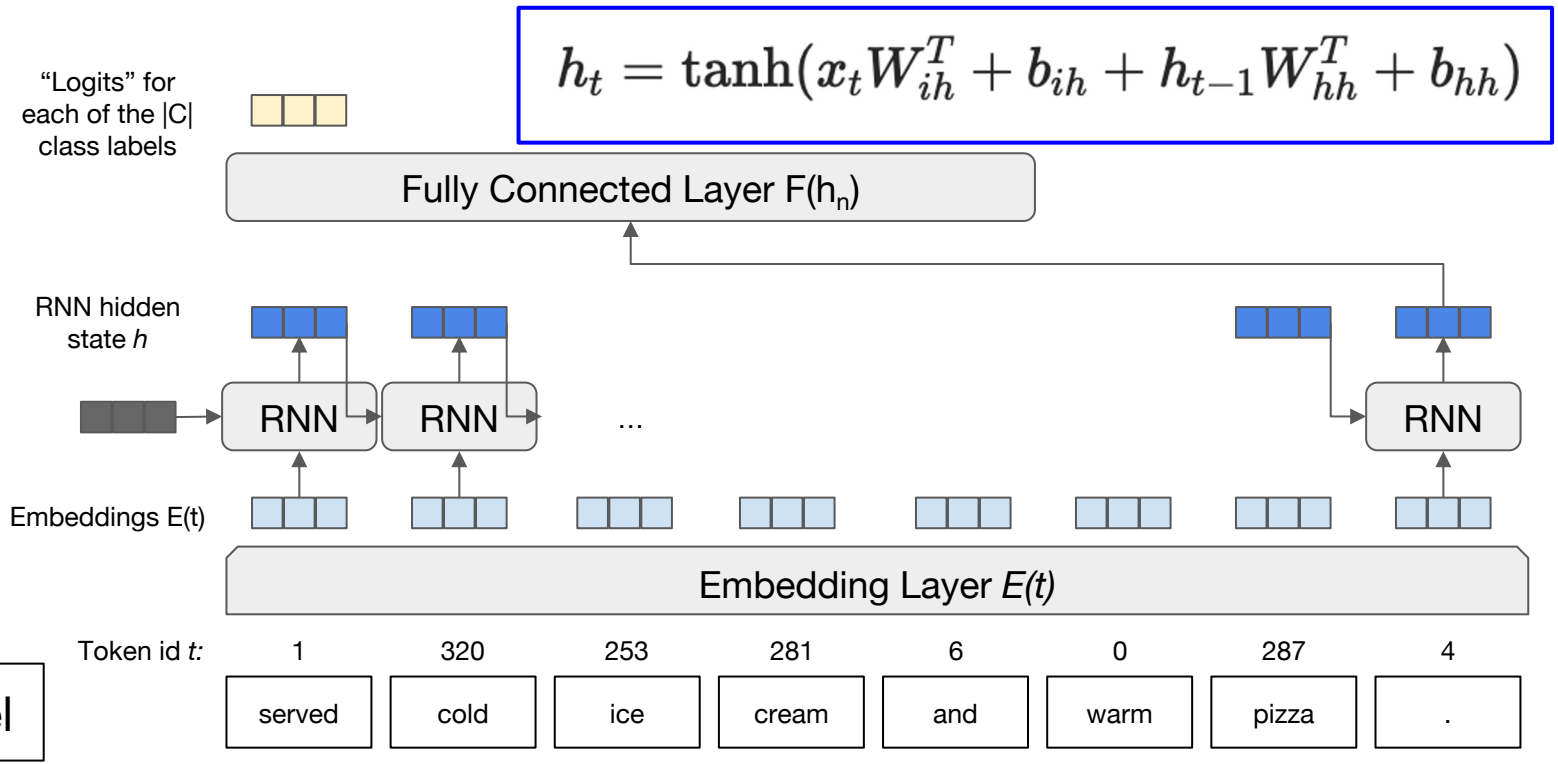
- Output y_t depends only on inputs so far $x_{0:t-1}$
  - E.g., robot actions

Bidirectional RNN



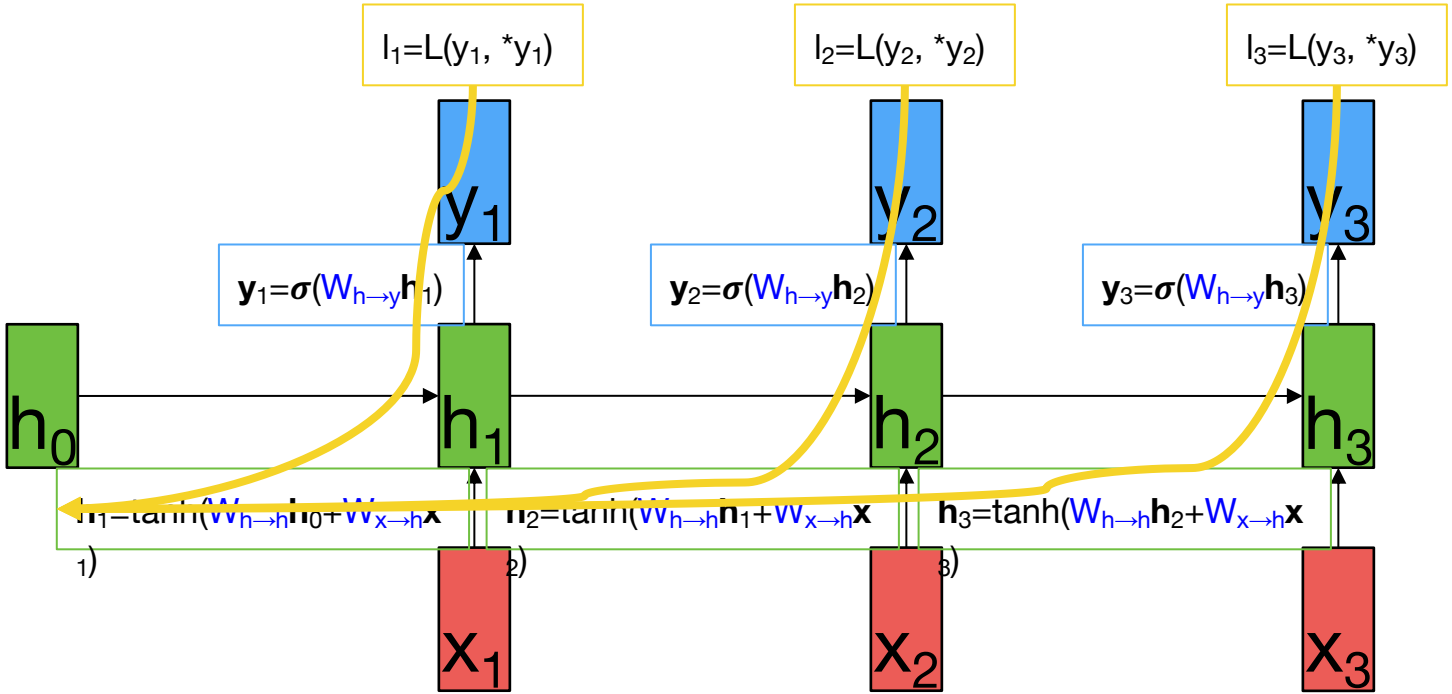
- Output y_t depends on all inputs $x_{0:L}$
  - E.g., machine translation

Text Classification with a Recurrent Neural Network

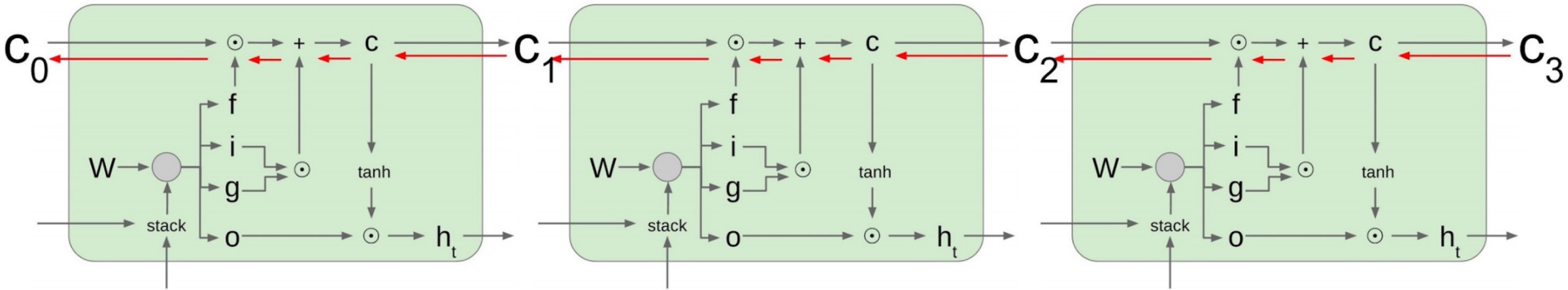


Word-level

Backpropagation for RNN



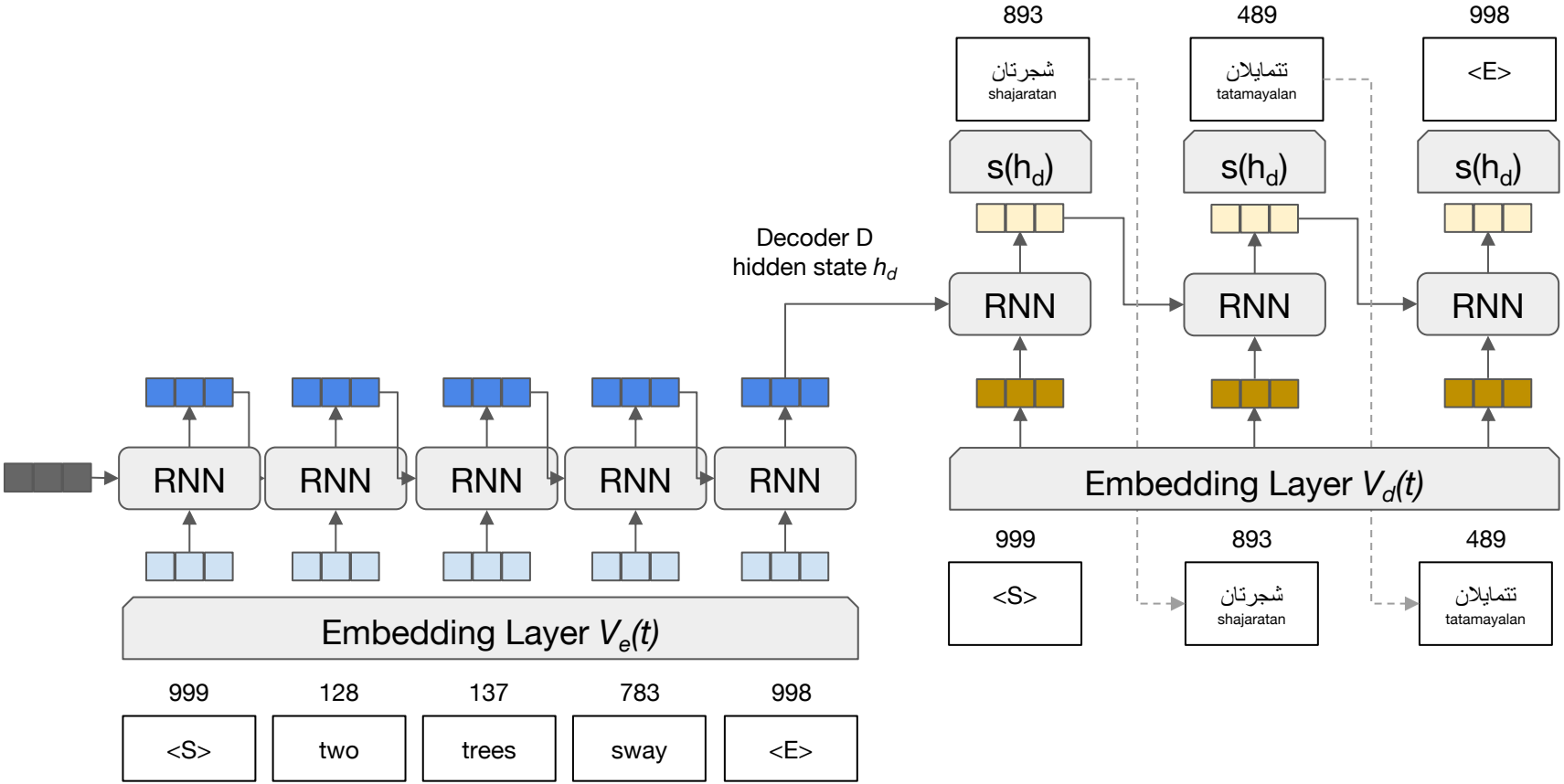
Long short-term memory



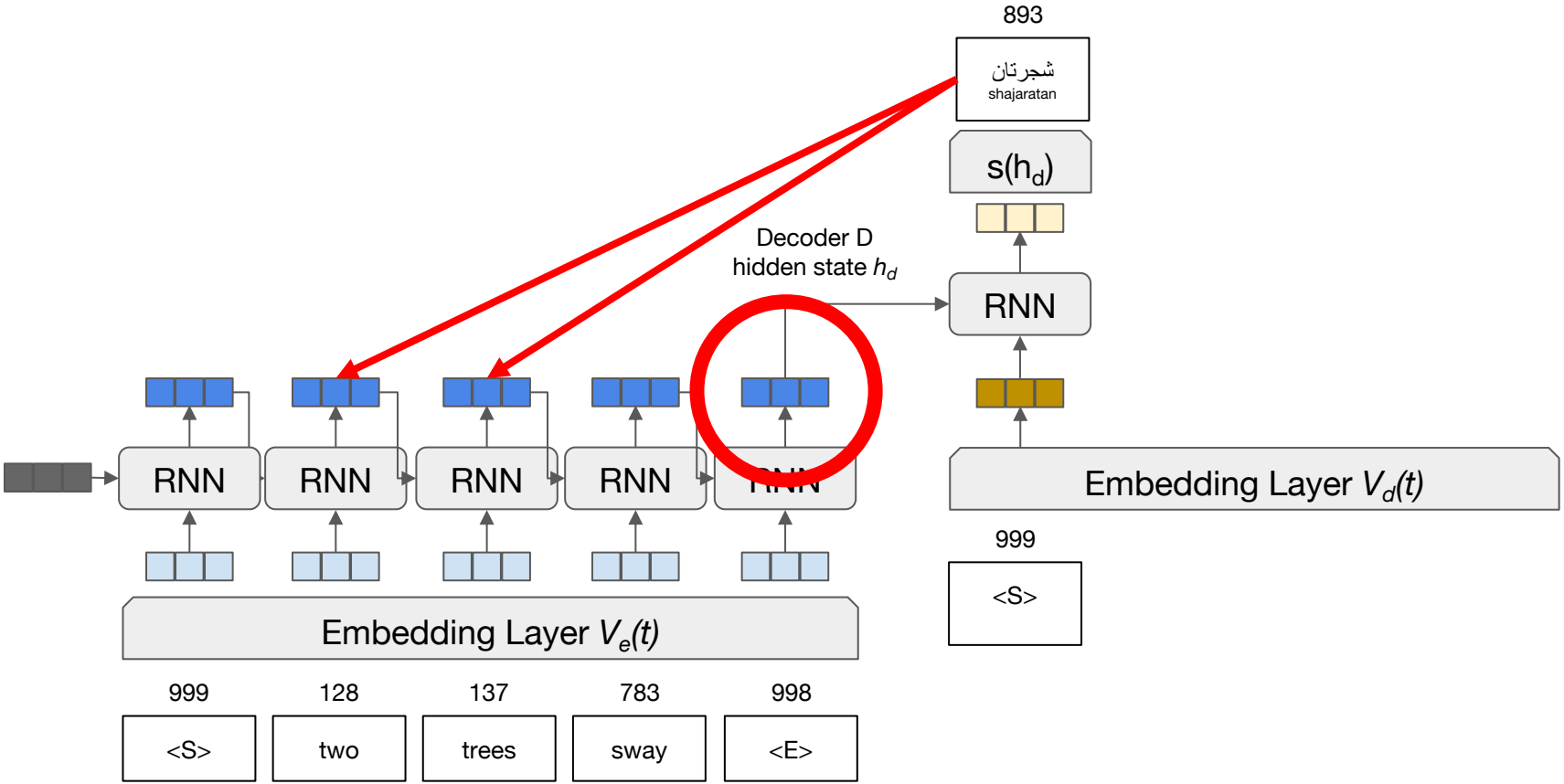
$$c_t = f \odot c_{t-1} + i \odot g$$

$$h_t = o \odot \tanh(c_t)$$

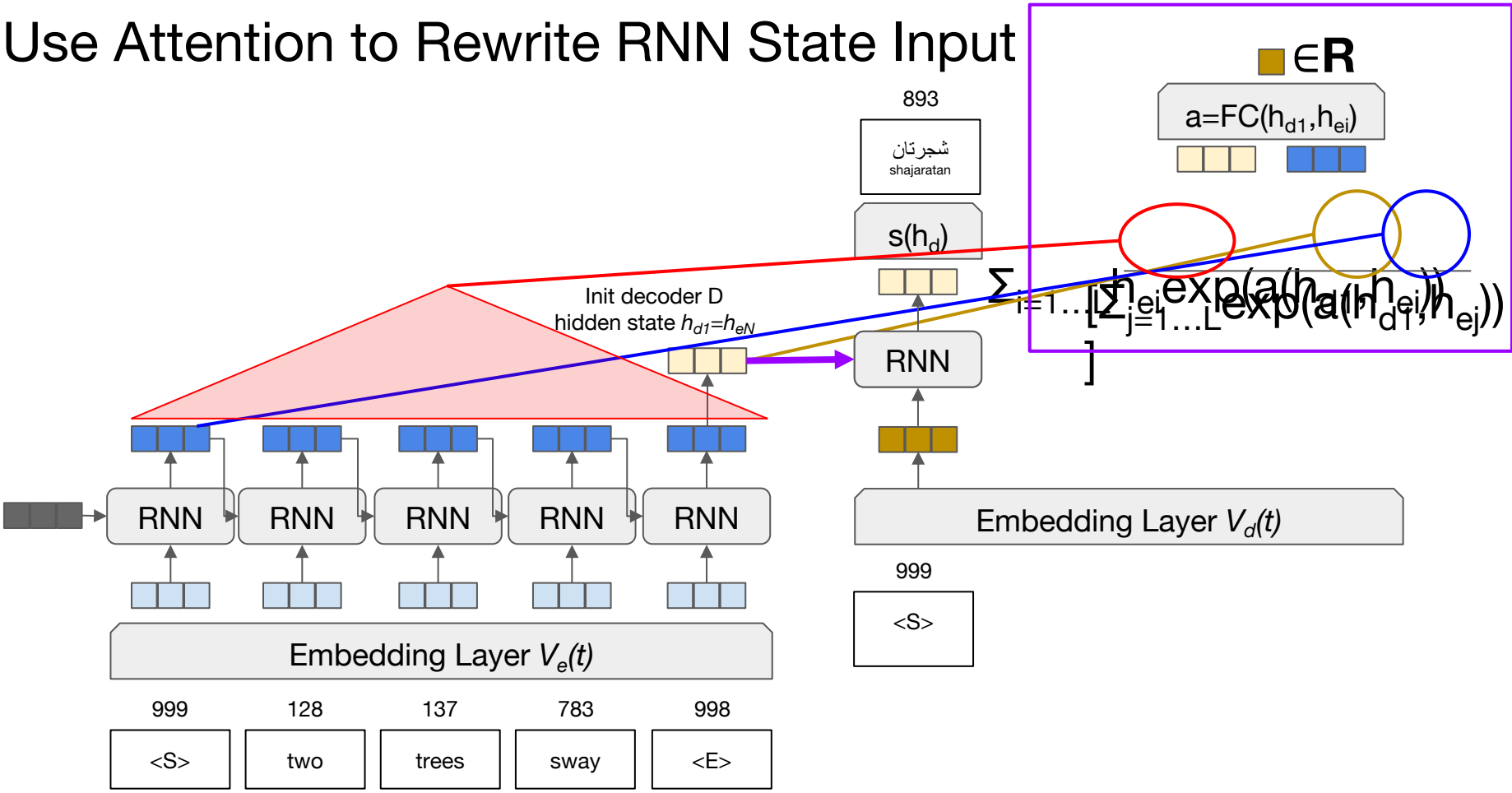
Machine Translation and The Case for Contextual Info



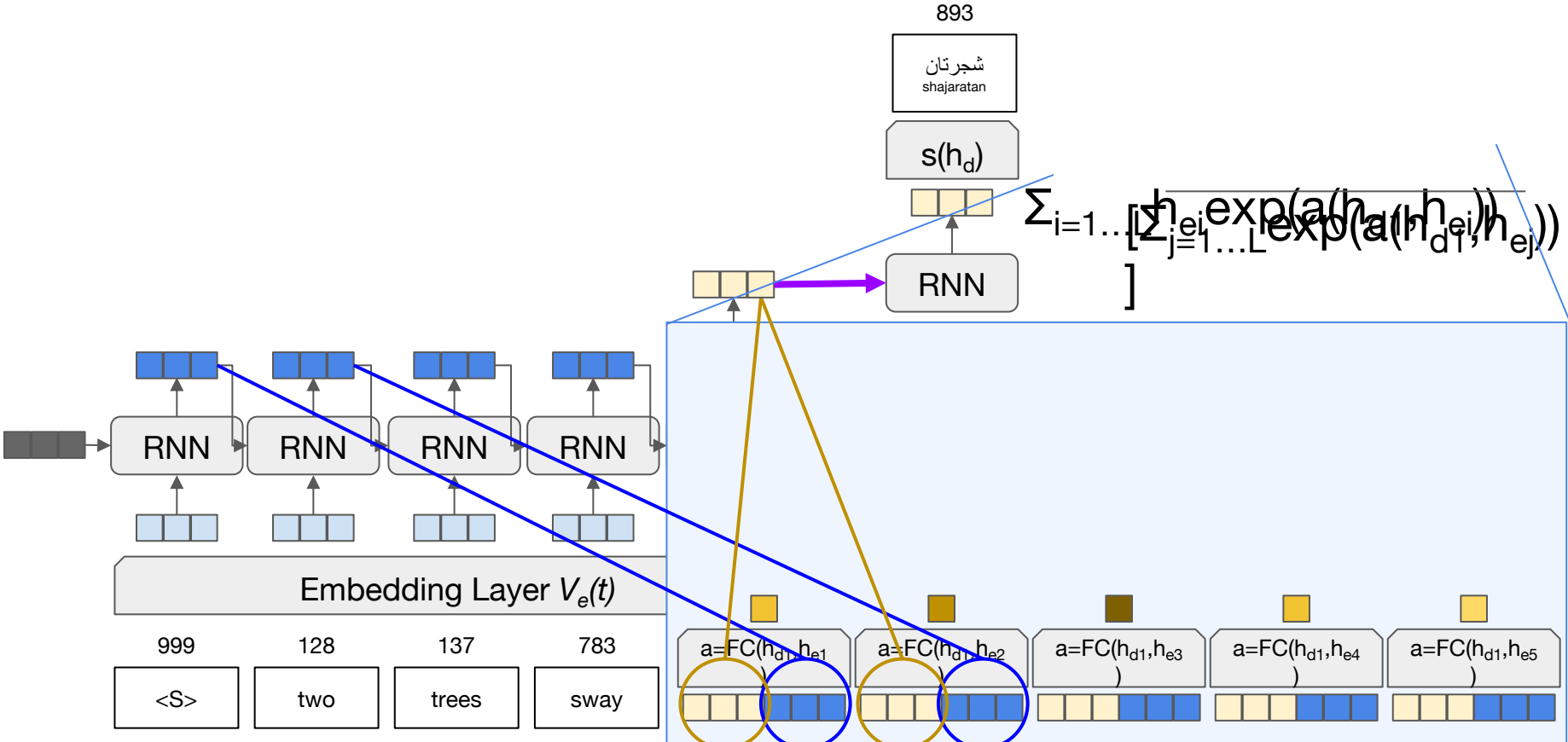
Machine Translation and The Case for Contextual Info



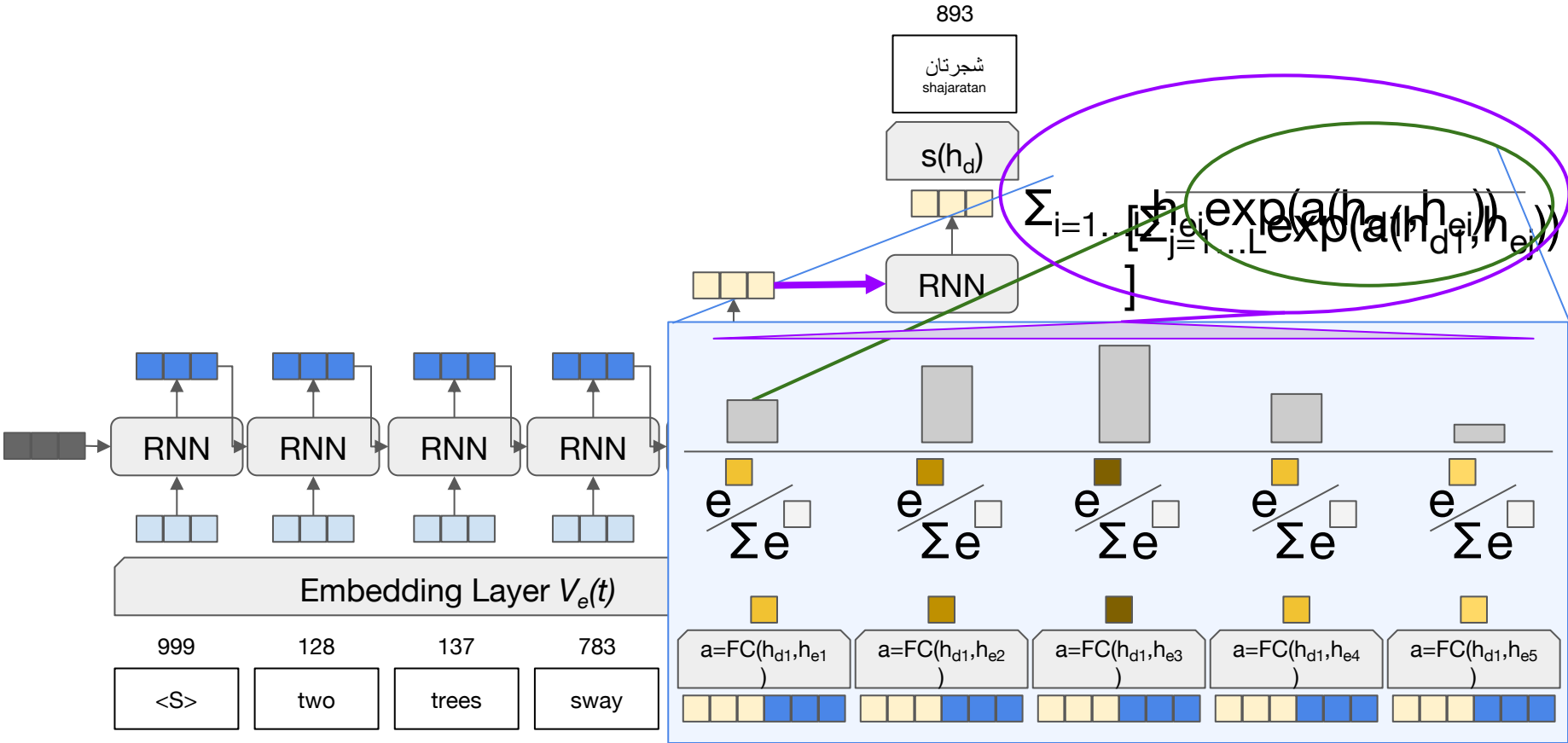
Use Attention to Rewrite RNN State Input



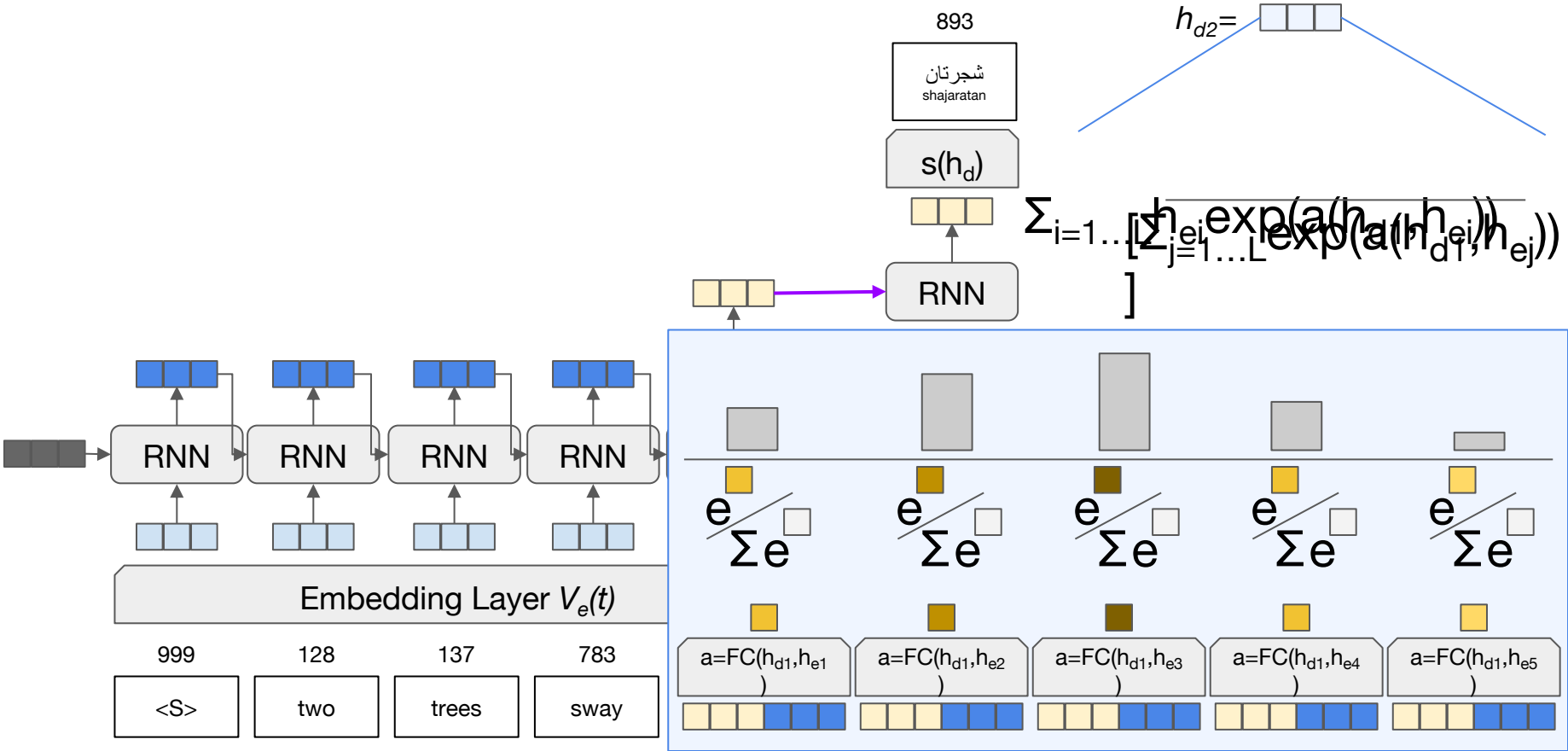
Attention


893
شجرتان
shajaratan
s(h_d)
RNN
Σ_i=1…L [exp(a(h_d1, h_ei)) / Σ_j=1…L exp(a(h_d1, h_ej))]
RNN
RNN
RNN
RNN
Embedding Layer V_e(t)
999
128
137
783
<S>
two
trees
sway
a=FC(h_d1,h_e1)
a=FC(h_d1,h_e2)
a=FC(h_d1,h_e3)
a=FC(h_d1,h_e4)
a=FC(h_d1,h_e5)

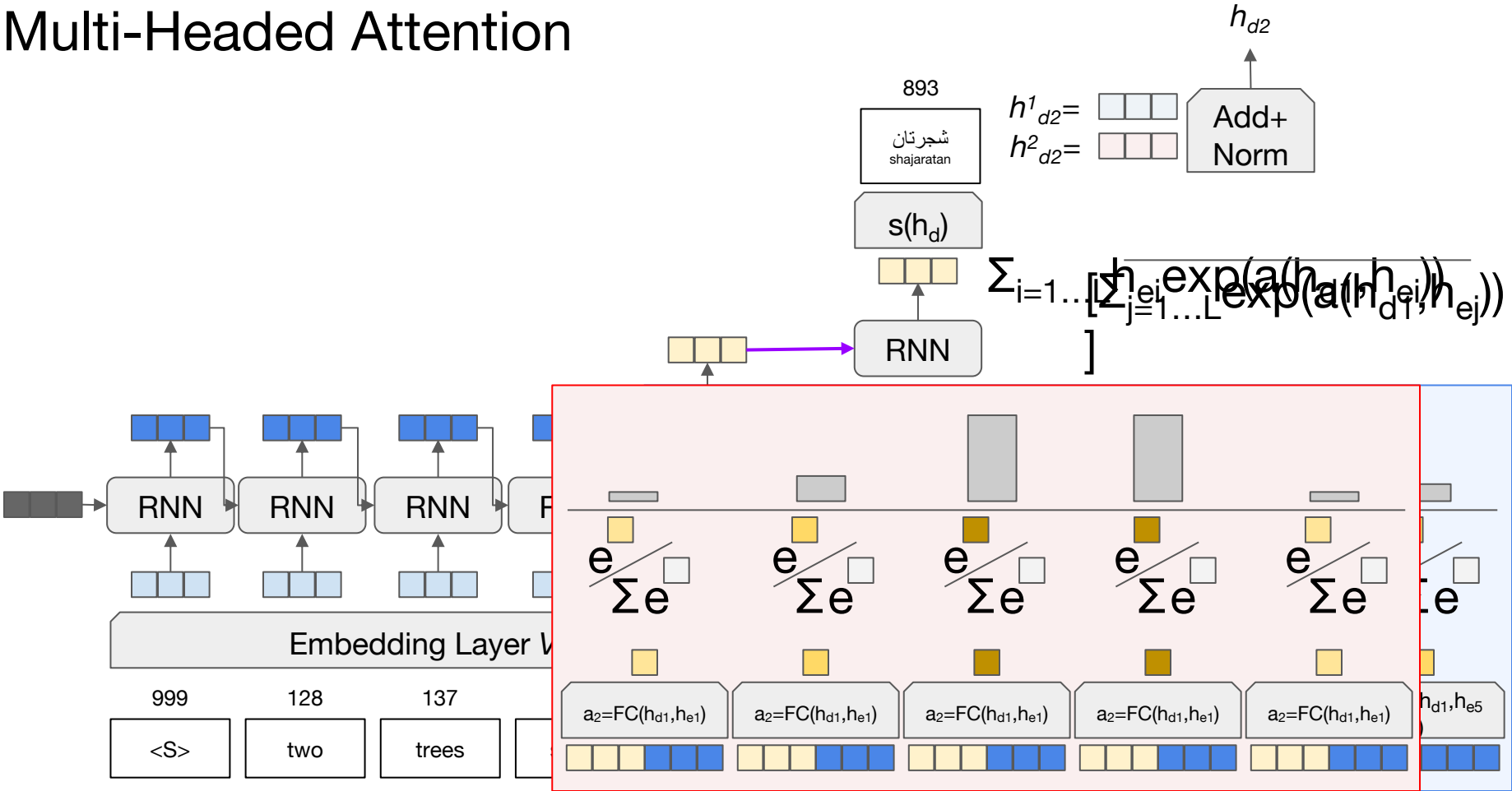
Attention


893
شجرتان
shajaratan
s(h_d)
RNN
RNN
RNN
RNN
RNN
Embedding Layer V_e(t)
999
128
137
783
<S>
two
trees
sway
a=FC(h_d1,h_e1)
a=FC(h_d1,h_e2)
a=FC(h_d1,h_e3)
a=FC(h_d1,h_e4)
a=FC(h_d1,h_e5)

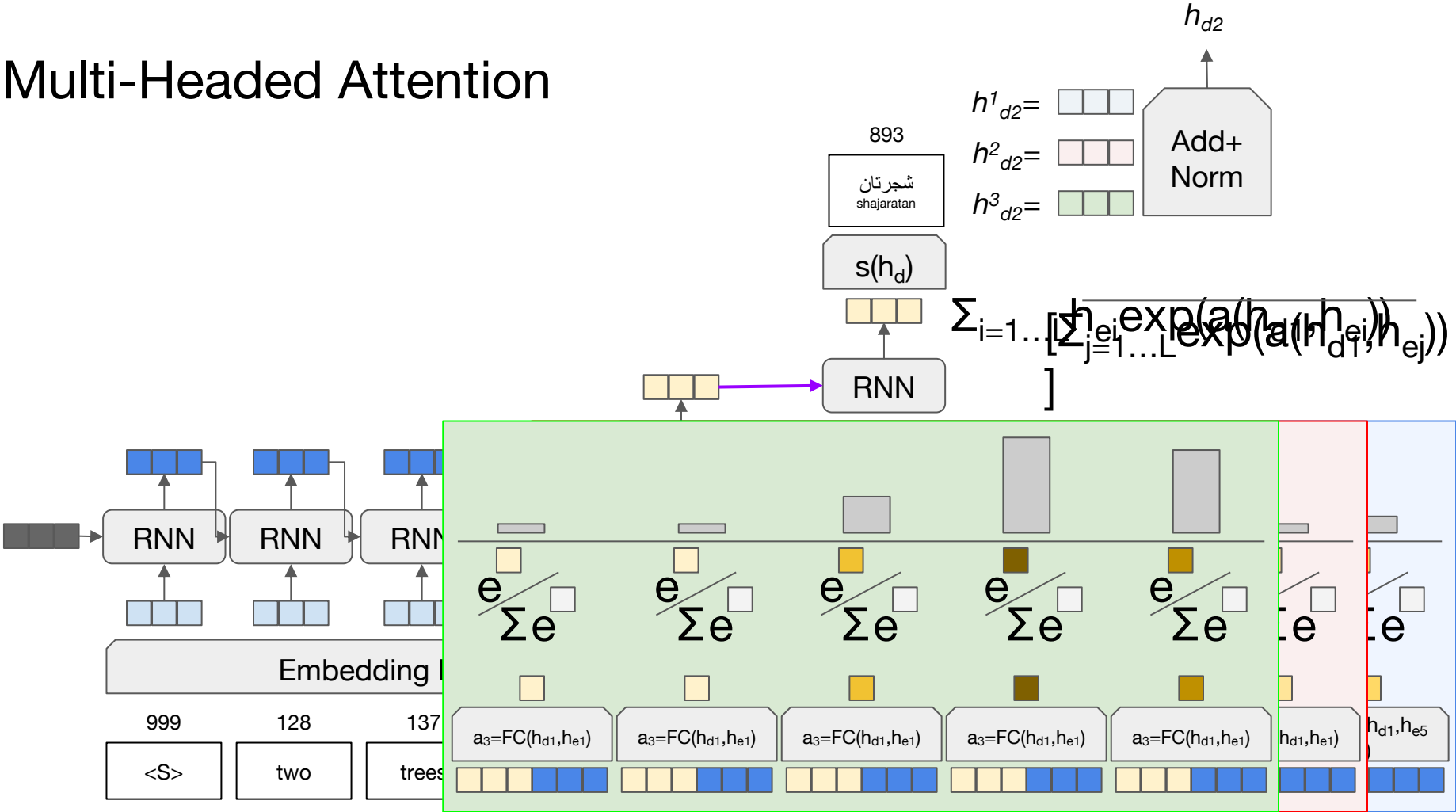
Multi-Headed Attention



Multi-Headed Attention



Multi-Headed Attention



Overview of Today's Plan

- ~~Course organization and deliverables~~
- ~~Recap from Lecture 5~~
  - Any questions before we move on?
- Language Models and Word Embeddings
- Project Pitches

NLP FUNDAMENTALS: Language Models

- A *language model* is an estimate of the likelihood of a string
- Tokenizing the string into component parts is necessary
- So we can consider a string *s* as a sequence of tokens $w_1 \ldots w_n$
- Then a language model estimates:
  - $p(s) = p(w_1 \ldots w_n)$
- Term is overloaded; also used these days to mean:
  - $p(w_k | w_1 \ldots w_{k-1})$ — (e.g., *auto-regressive*)
  - $p(w_k | w_1 \ldots w_{k-1} w_{k+1} \ldots w_n)$ — (e.g., *cloze / MLM*)

N-gram language models

- If we factor a string into words w_i, then
  - $p(s) = p(w_1 \ldots w_n)$
  - $= p(w_n|w_1 \ldots w_{n-1})p(w_1 \ldots w_{n-1})$
    - In other words, the probability of the next word is equal to its chance given the words seen so far compounded by the likelihood of that sequence of words at all
- What would it take to store $p(w_n|w_1 \ldots w_{n-1})$ on disk?

N-gram language models

- How could we *efficiently* estimate $p(w_n|w_1 \ldots w_{n-1})$?
  - Independence assumptions!
- Does a word *k* away in the history really affect what word comes next that much? What if we assume:
  - $p(w_n|w_1 \ldots w_{n-1}) \approx p(w_n|w_{n-k+1} \ldots w_{n-1})$
- Then if we say “a word 4 away in the history does not affect me”, then $p(w_n|w_1 \ldots w_{n-1}) \approx p(w_n|w_{n-3}w_{n-2}w_{n-1})$
- How big on disk?
  - $|V|*|V|*|V|*|V| = |V|^4$

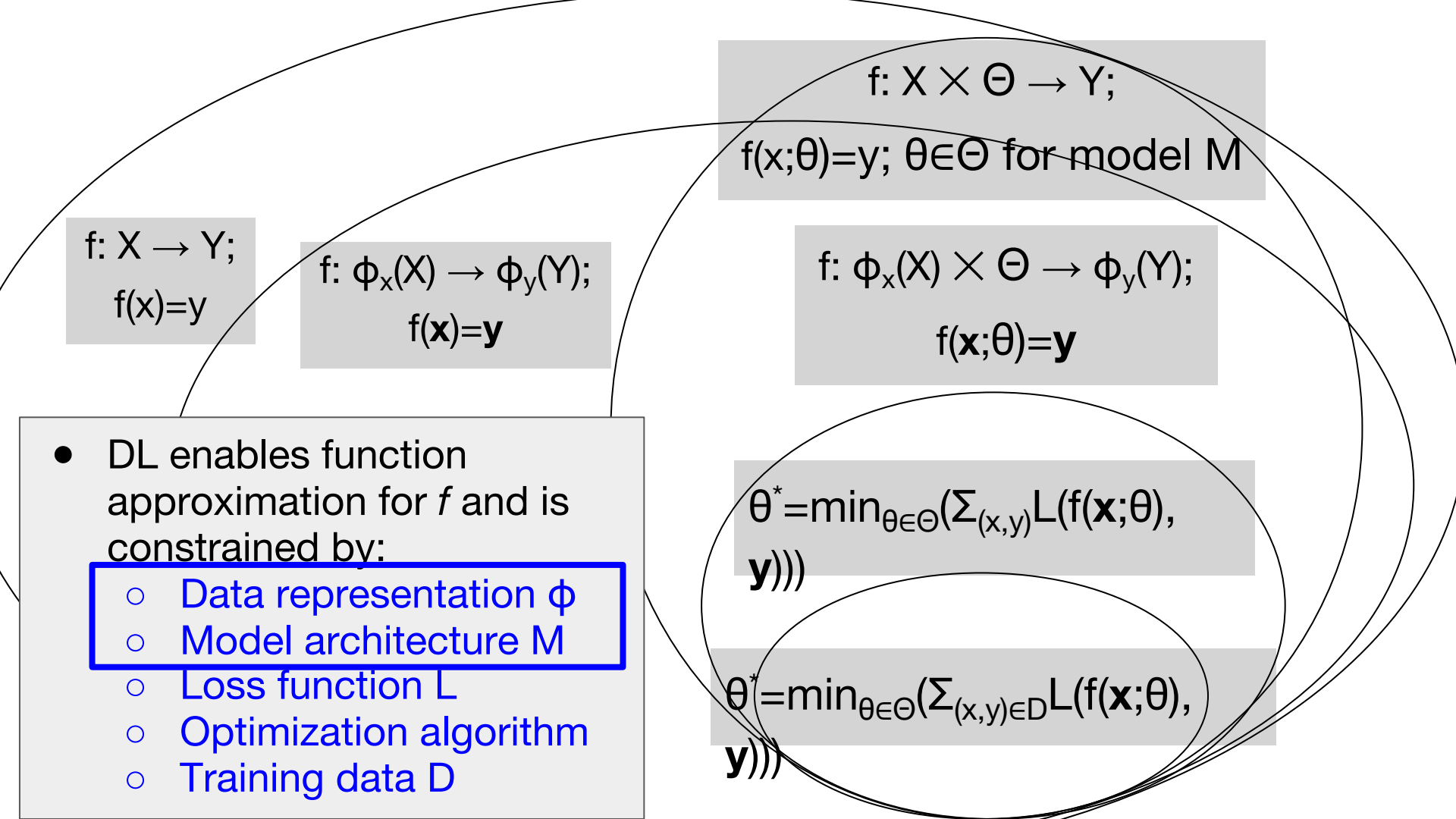
N-gram language models

- *k=3*, trigram LM
  - $p(w_n|w_1 \ldots w_{n-1}) \approx p(w_n|w_{n-2}w_{n-1})$
  - $|V|*|V|*|V| = |V|^3$
- *k=2*, bigram LM
  - $p(w_n|w_1 \ldots w_{n-1}) \approx p(w_n|w_{n-1})$
  - $|V|*|V| = |V|^2$
- *k=1,* unigram LM
  - $p(w_n|w_1 \ldots w_{n-1}) \approx p(w_n)$
  - $|V|$

Bigram Co-occurrence matrix

Count($w_i|w_{i-1}$)

w_i / w_{i-1}	the	students	learned	about	word	vectors	today	in	class
the	0	0	0.05	0.1	0	0	0.05	0.1	0
students	0.1	0	0	0.1	0	0	0.05	0.05	0
learned	0	0.2	0	0	0.05	0.05	0.05	0	0.2
about	0	0.05	0.2	0	0	0.05	0.1	0.2	0.1
word	0.1	0	0.1	0.1	0	0	0.05	0.05	0
vectors	0.1	0	0.2	0.2	0.4	0	0.05	0.05	0
today	0	0.05	0.1	0	0	0	0	0.05	0.05
in	0	0.1	0.1	0.05	0.05	0.05	0.2	0	0.05
class	0.1	0	0.05	0.1	0.05	0	0	0.2	0


f: X → Y;
f(x)=y
f: $\phi_x(X) \rightarrow \phi_y(Y)$;
f(**x**)=**y**
f: X × Θ → Y;
f(x;θ)=y; θ∈Θ for model M
f: $\phi_x(X) \times \Theta \rightarrow \phi_y(Y)$;
f(**x**;θ)=**y**
$\theta^* = \min_{\theta \in \Theta}(\Sigma_{(x,y)} L(f(\mathbf{x};\theta), \mathbf{y}))$
$\theta^* = \min_{\theta \in \Theta}(\Sigma_{(x,y) \in D} L(f(\mathbf{x};\theta), \mathbf{y}))$
• DL enables function approximation for *f* and is constrained by:
○ Data representation ϕ
○ Model architecture M
○ Loss function L
○ Optimization algorithm
○ Training data D

Text Representation

- We need to define the *encoding function*, $\phi(x)=\boldsymbol{x}$ for string x
  - Our encoding function will map raw strings to *input features* for the model *M*
- How can we represent text?
  - *x*="A tropical bird perches in the jungle."
  - *x* is a sequence of characters, if we think about underlying data and the representation it takes in the machine.
  - When *we* think about language, do we think in sequences of characters?
  - What might be helpful for a model to reason about?

Feature Extraction for Language - Tokenization

"A tropical bird perches in the jungle."

D: image captions *x* that describe class label object *l*

Word-level	a	tropical	bird	perches	in	the	jungle	.

- We can break *x* up into the words and punctuation in it.
- We'll consider some alternatives shortly, but let's take notes on what this choice gets us:
  - Can we can learn parameters of *M* that represent *every* word (i.e., **word embeddings**)?
  - Only "yes" if we can fit θ on disk, which requires θ to be at least finite, preferably also not exponential.

Feature Extraction for Language - Tokenization

"The year is 2032. A model was trained on all images, videos and text on the web, using over 100 yottaFLOPs.."

Word-level	...	using	over

- Do we *want* a token representing the "word" *100*?
  - We'll have to learn individual embeddings for 0, 1, 2, …
- Do we *want* a token representing the "word" *yottaFLOPs*?
  - Unlikely we'll ever see this word again!
- What if we find a way to ignore infrequent words?
  - How many words are "infrequent"?

https://twitter.com/gabriel_ilharco/status/1558079922026921985?s=20&t=L9uvSHaczZX1fPc7TDE3ww

Zipf’s Law

- “an empirical law formulated using mathematical statistics that refers to the fact that for many types of data studied in the physical and social sciences, **the rank-frequency distribution is an inverse relation**.”
- The frequency of words in natural language is often assumed to follow a Zipfian distribution

https://en.wikipedia.org/wiki/Zipf%27s_law

Zipf’s Law

- “Zipf's law was originally formulated in terms of quantitative linguistics
- The frequency of any word is inversely proportional to its rank in the frequency table.
- Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc.”

https://en.wikipedia.org/wiki/Zipf%27s_law

Feature Extraction for Language - Tokenization

"The year is 2032. A model was trained on all images, videos and text on the web, using over 100 yottaFLOPs.."

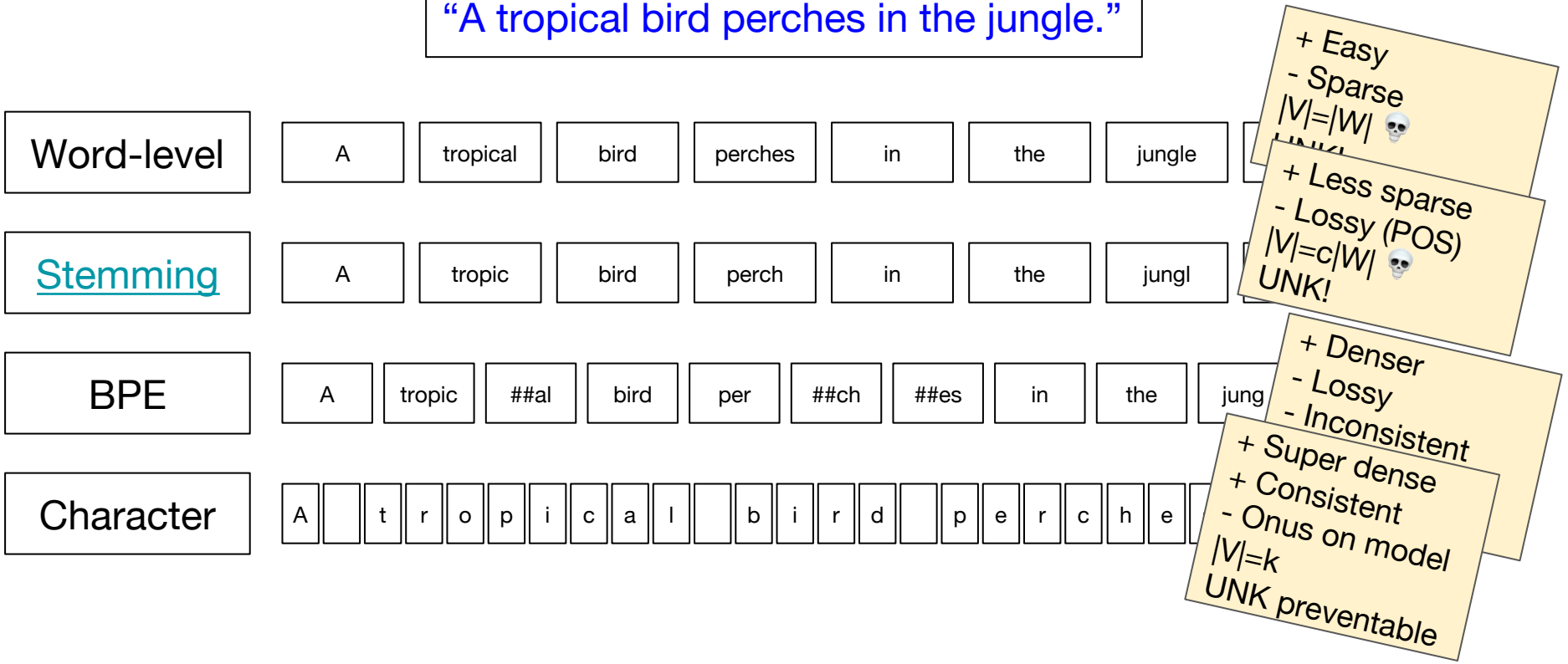
Word-level	...	using	over	100	UNK	.	.

- We *count* all the *tokens* in D_{train} and sort them by frequency
- Keep the top-*k* most common tokens
- What about everything else?
- We can have a catch-all *unknown* (UNK) token with its own learnable representation
  - In other words: we make it the model's problem.

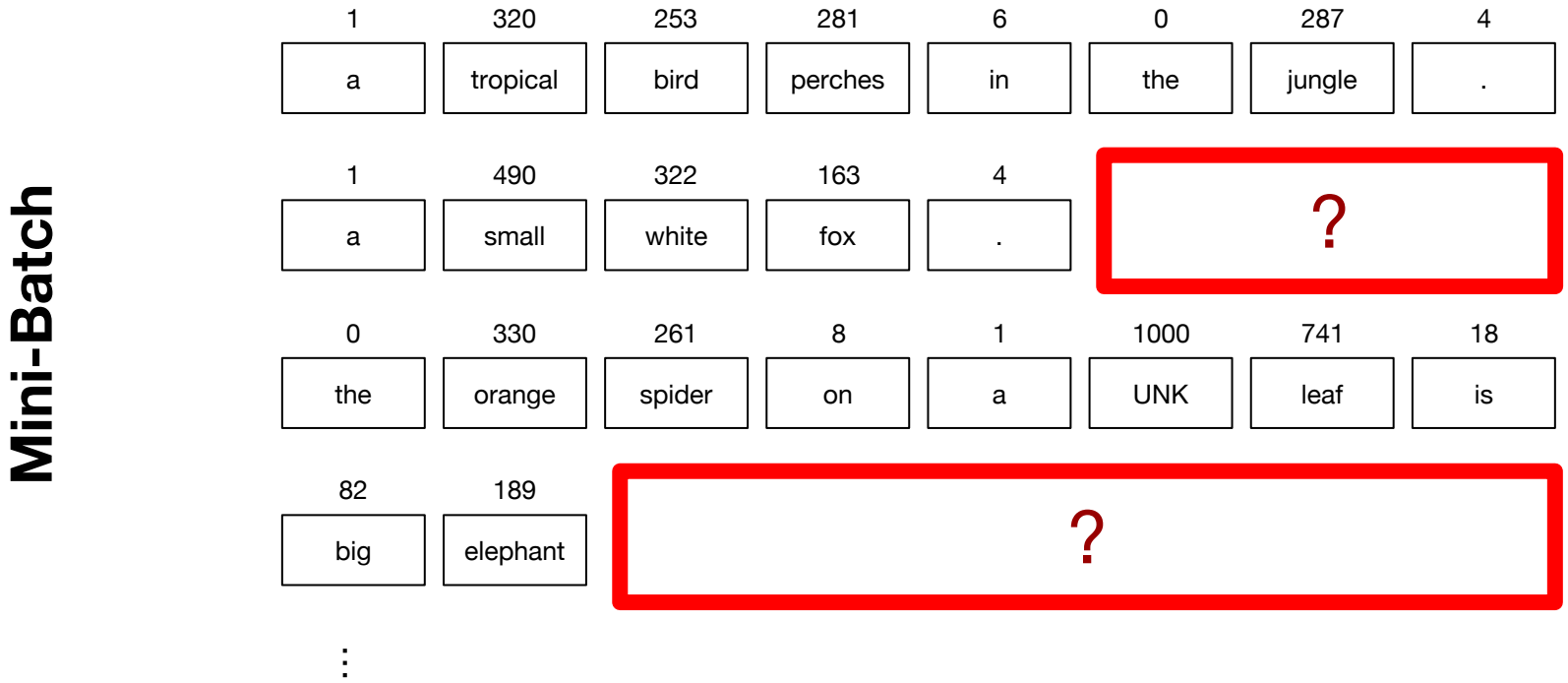
https://twitter.com/gabriel_ilharco/status/1558079922026921985?s=20&t=L9uvSHaczZX1fPc7TDE3ww

Feature Extraction for Language - Tokenization

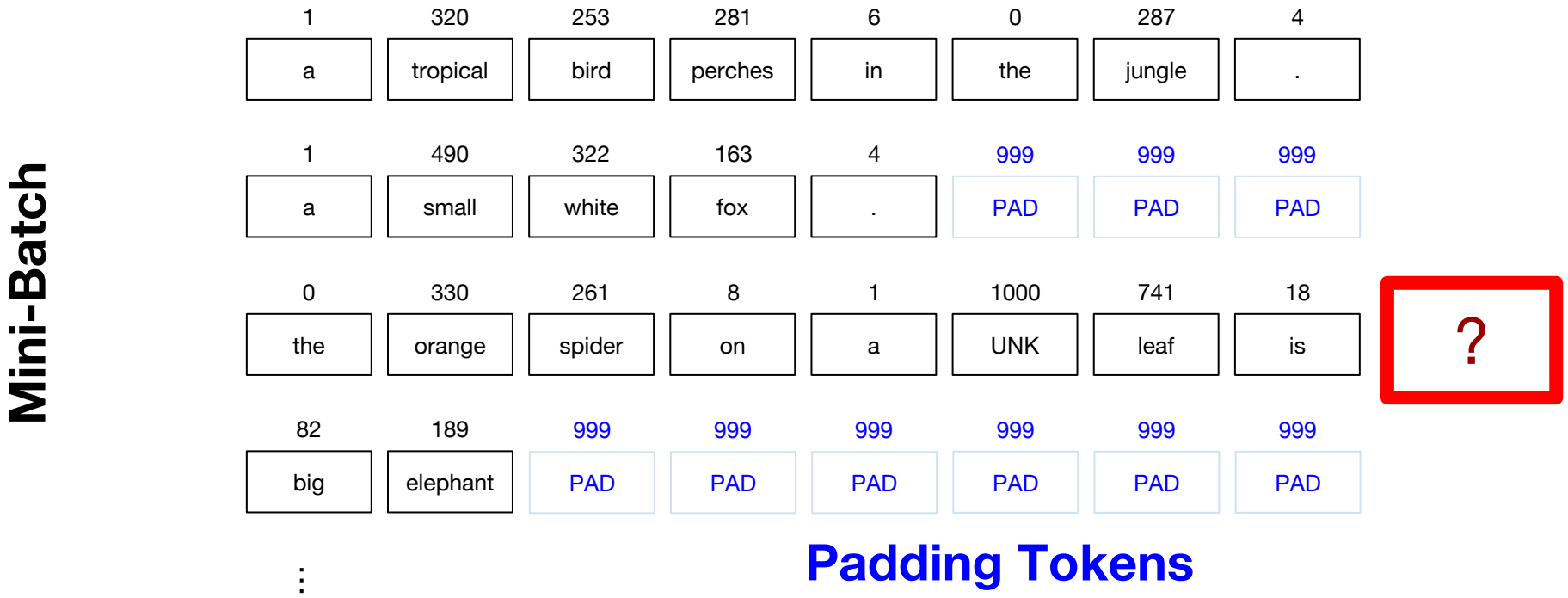
"A tropical bird perches in the jungle."



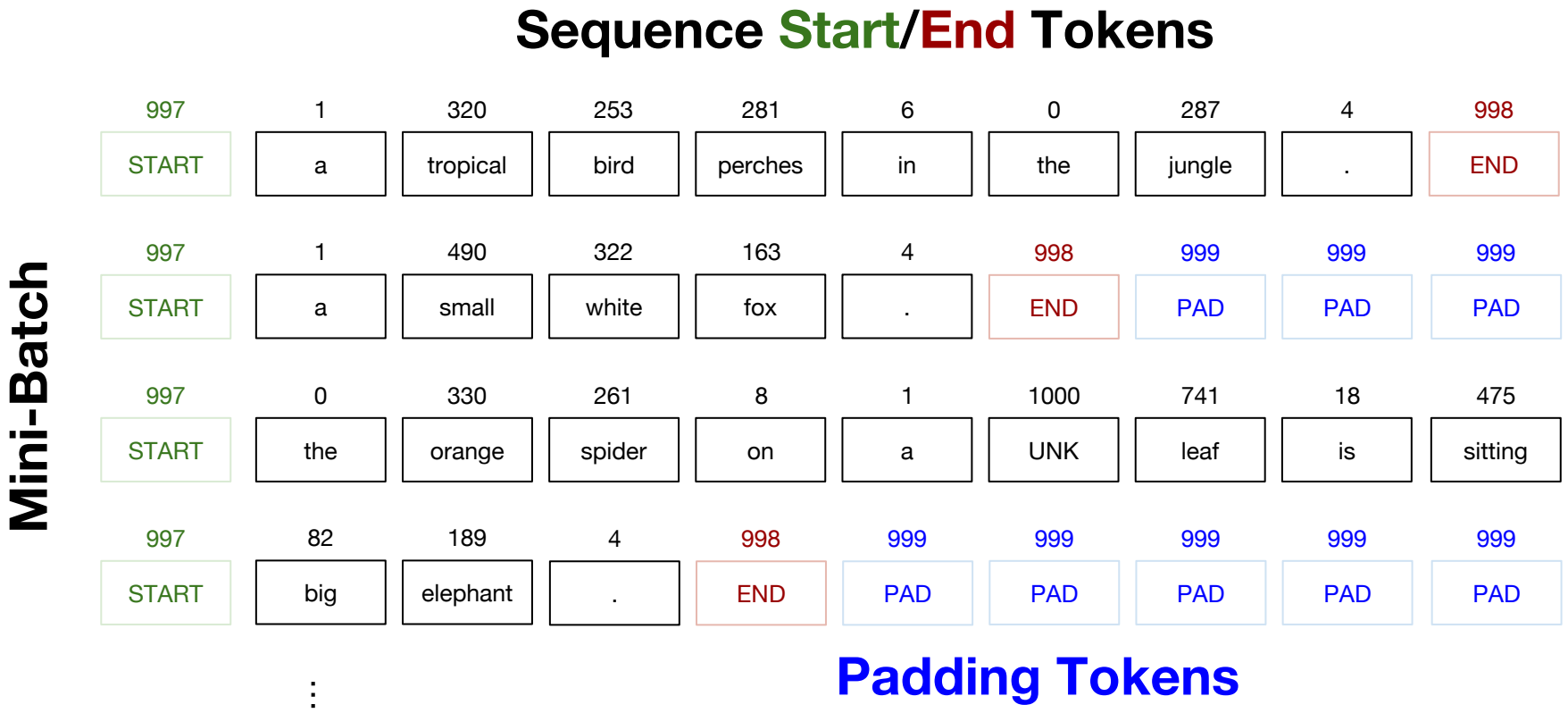
Tokenization and Batching



Tokenization and Batching



Tokenization and Batching



NLP FUNDAMENTALS: Word Embeddings

- A word embedding is a *vector representation* of a word
- Word embeddings can be learned for specific tasks (e.g., our book classification task) or from self-supervised objectives
- **“You shall know a word by the company it keeps” - Firth**
- The guiding principle of many learned word embeddings is that two words who share similar *context* should have embeddings that are close together in vector space

J.R. Firth. *A synopsis of linguistic theory*. Studies in linguistic analysis, Blackwell, Oxford (1957)

Simplest option: Co-occurrence matrix

- Let's given each word *w* a vector of dimension |V| that represents the distribution of words that occur *nearby* to *w*
- Co-occurrence:
  - *Left context*: what words occur before this one
  - *Right context*: what words occur after this one
  - *Bidirectional context*: what words occur on either side
  - *Context length*: how far do we look away from *w*?
    - Shorter context: word vectors favor syntax
    - Longer context: word vectors favor semantics/topic

n-Gram Model

- An *n*-gram model represents the context of each word *w* as its co-occurrence with other words 1-, 2-, …, n-1-steps away
- n-Grams are based on rigid, explicit counting, plus some smoothing to relax that rigidity
- How else could we relax $p(w_i|w_{i-1}w_{i-2}\ldots w_{i-n+1})$?
  - Previously, we kept a $|V|$ count for each distance $i-k$
    - We could just keep a |V| count that merges distances
    - “Bag of Words”-based model

Bag-of-Words Model

- Estimate $p(w_i|w_{i-1}w_{i-2}\ldots w_{i-n+1})$ as distribution of w_i given that $w_{i-1}w_{i-2}\ldots w_{i-n+1}$ occur *in any order* left of w_i
  - Pros:
    - Space to store this model is $|V|^2$
    - Model will be considerably less sparse
  - Cons:
    - Loss of order information will hurt POS guessing / syntax generally
  - Because the *context* from which we predict w_i is now a single vector, what other interesting information is here?

Bag-of-Words Model

- We've been thinking about LM from just one perspective
  - "What *next token* is likely given the *context* so far?"
- We could ask something else…
  - "What *context* will make a *particular token* most likely?"
    - Gives us a sense of which tokens are *similar* based on sharing similar favored *contexts* that produce them!
    - "You shall know a word by the company it keeps."

Vectors Representing Context

- What *context* will make a *particular token w=a* most likely?
  - $\text{argmax}_{w(i-k)\in V}(p(w_i=a|w_{i-1}w_{i-2}\ldots w_{i-n+1}))$
- Let's apply Bayes' Theorem:
  - $p(a|w_{i-1}w_{i-2}\ldots w_{i-n+1})$
  - $= p(w_{i-1}w_{i-2}\ldots w_{i-n+1}|a)p(a) / p(w_{i-1}w_{i-2}\ldots w_{i-n+1})$
- Then we want:
  - $\text{argmax}_{w(i-k)\in V}(p(w_{i-1}w_{i-2}\ldots w_{i-n+1}|a)p(a) / p(w_{i-1}w_{i-2}\ldots w_{i-n+1}))$
  - $\text{argmax}_{w(i-k)\in V}(p(w_{i-1}w_{i-2}\ldots w_{i-n+1}|a) / p(w_{i-1}w_{i-2}\ldots w_{i-n+1}))$
- The *context* that maximizes the chance token *a* comes next is the likelihood of the context given *a* by the marginal probability of the context itself

Vectors Representing Context

- Let's consider a bigram model:
  - $\text{argmax}_{w(i-1) \in V}(p(w_{i-1}|a) / p(w_{i-1}))$
  - Intuitively: the word that frequently occurs before the target word *a*, and seldom occurring without *a*
- If our *context* is a *multi-hot* vector instead of a *one-hot* vector, the intuition is the same, just for the last few words
- Could we have reasoned about likely *context* in our *n*-gram models that considered word order?
  - Yes! Just even less tractable. $|V|^{n-1}$ places to reason about.

Vectors Representing Context

- The core unsatisfying thing about *n*-grams:
  - Sparsity!
- Zipf's law!
  - Zero entries dominate as context (n) grows
- Using a multi-hot context of size |V| for previous n words (our bag-of-words alternative) *does not resolve this fundamental issue* because it's based on co-occurrence counts yet

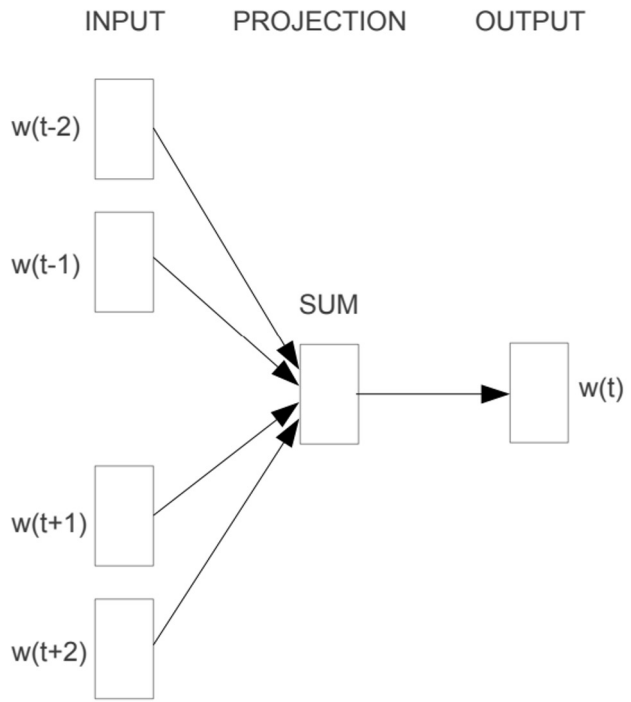
Word Embeddings Compress n-grams!

- What is the standard remedy for sparsity in a data structure?
  - Compression!
- What we would *like to achieve* is to compress all the co-occurrence information in our big table of word/context counts in fewer than $|V|^n$ dimensions
- Luckily, we can
  - The first *neural* approaches to achieving this goal stored $|V|$=30k, n=10 with bidirectional context in 600d vectors
    - Raw co-occurrence: $30{,}000^{10}$
    - Three hundred trillion entries in co-occ matrix

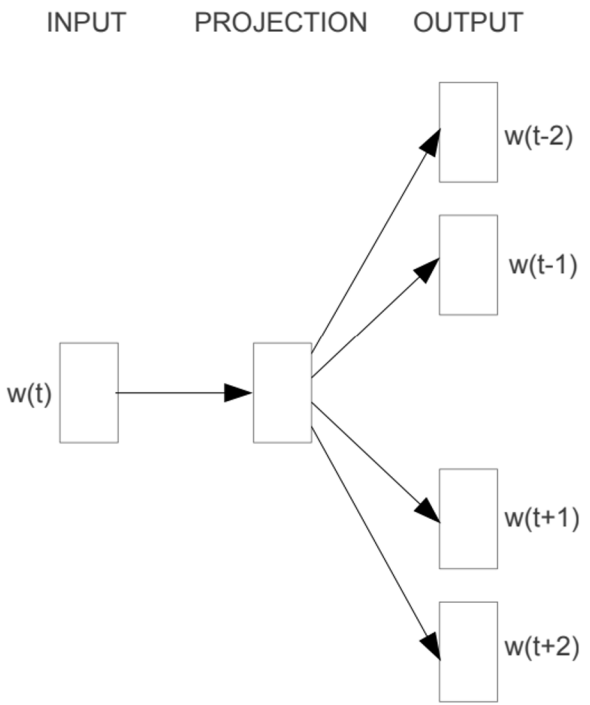
Word Embeddings: A dense alternative to sparse n-grams

- The first *neural* word embeddings learning paper to gain traction:
  - Efficient Estimation of Word Representations in Vector Space. Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. International Conference on Learning Representations (ICLR), 2013.
- Popularly known as?
  - Word2Vec

Word2Vec Proposed Two Models



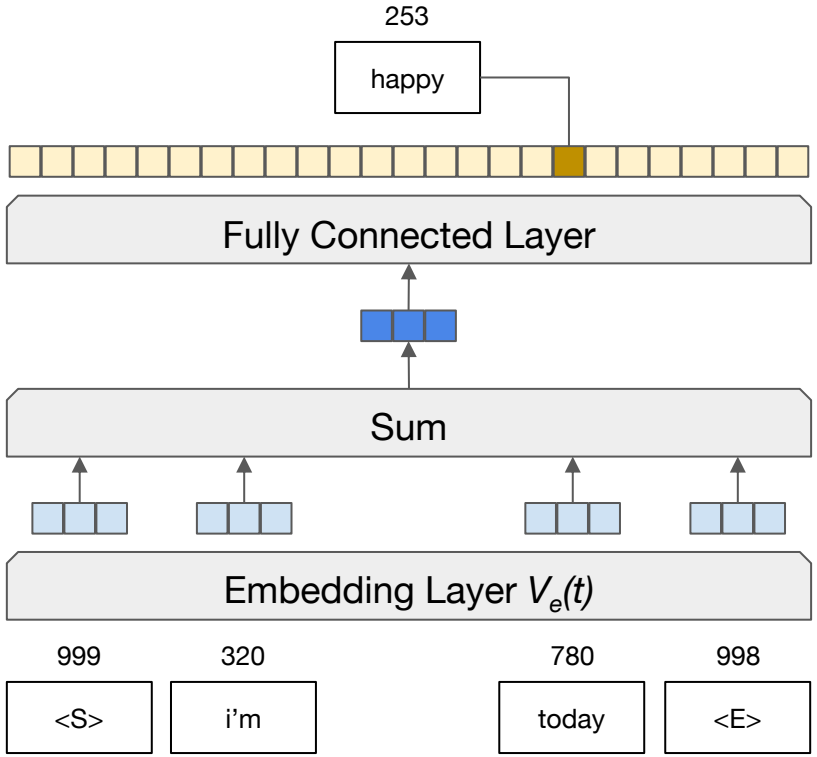
CBOW



Skip-gram

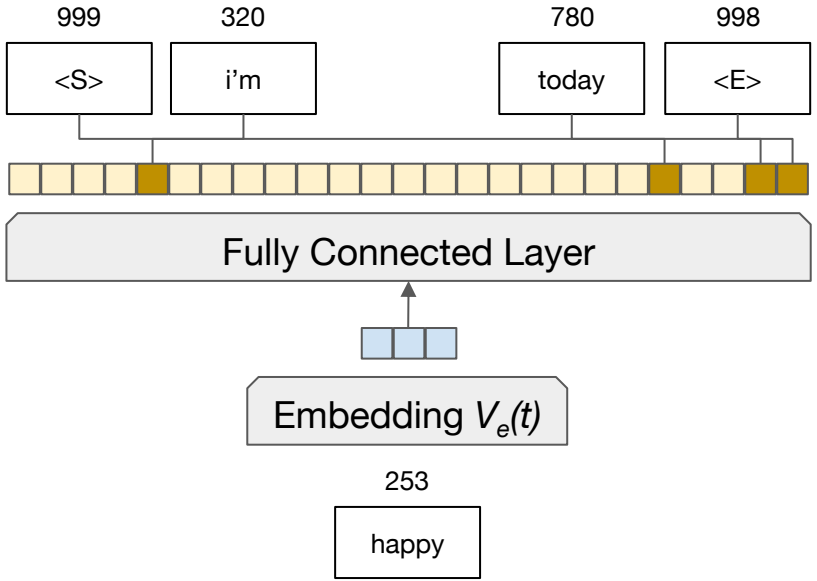
Word2Vec: Continuous Bag-of-Words (CBOW)

- What probability does the FC layer estimate?
  - $p(w_i|w_{i-2}w_{i-1}w_{i+1}w_{i+2})$
- What assumptions?
  - Word order doesn't matter
  - Firth!
- What's being learned?
  - FC + Embeddings



Word2Vec: Continuous Skip-Gram Model

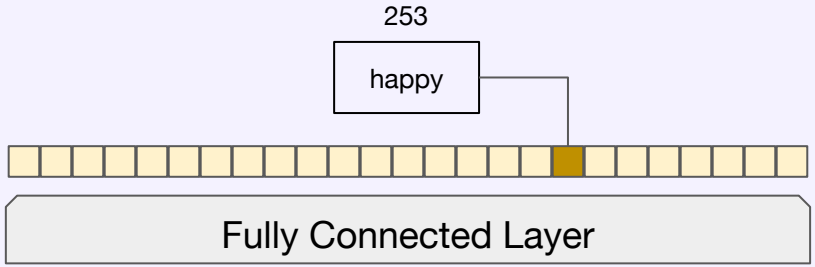
- What probability does the FC layer estimate?
  - $p(w_{i-2}|w_i)p(w_{i-1}|w_i)*$ $p(w_{i+1}|w_i)p(w_{i+2}|w_i)$
- What assumptions?
  - Word order doesn't matter
  - Firth!
- What's being learned?
  - FC + Embeddings



ML FUNDAMENTALS: Cross Entropies

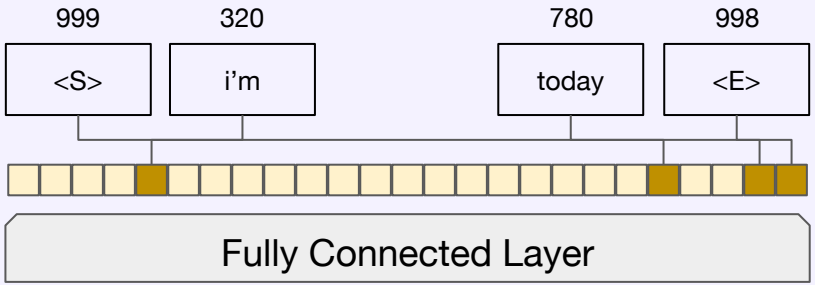
- *Categorical* cross entropy:
  - For F(x)->|C|, [illegible] label $c^* \in C$; *F(x)* estimate[illegible] and the loss encourages $p(c^*|x)=1$

Say which one of |C| labels x exhibits



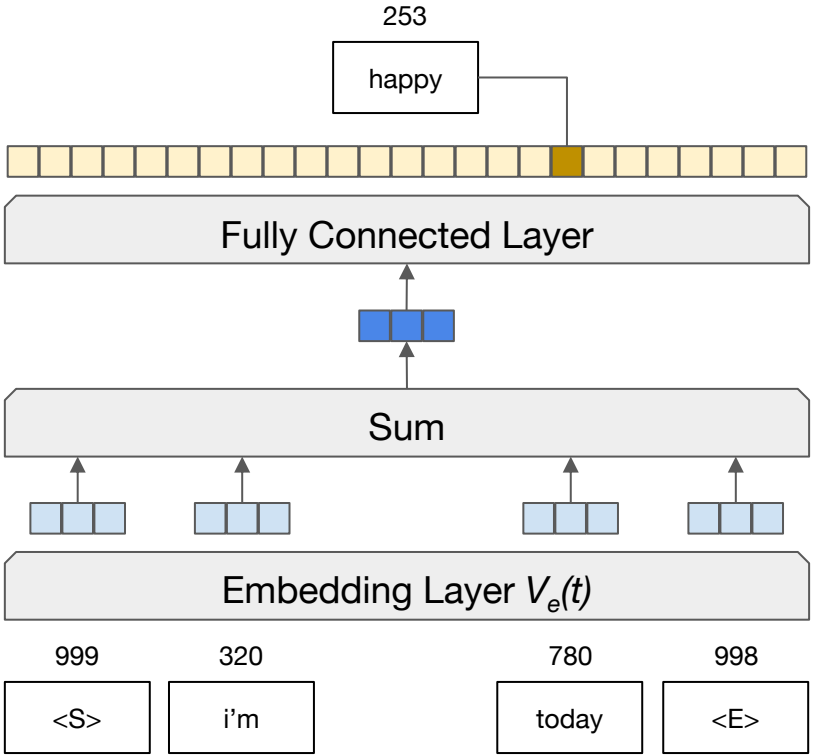
- *Binary* cross ent[illegible]
  - For F(x)->|C|, [illegible] a combination of labels $\mathbf{c}_k \in C$; F(x) estimates p(**c**|x), and the loss encourages $p(\mathbf{c}_k|x)=1$

For each |C| say whether x exhibits it

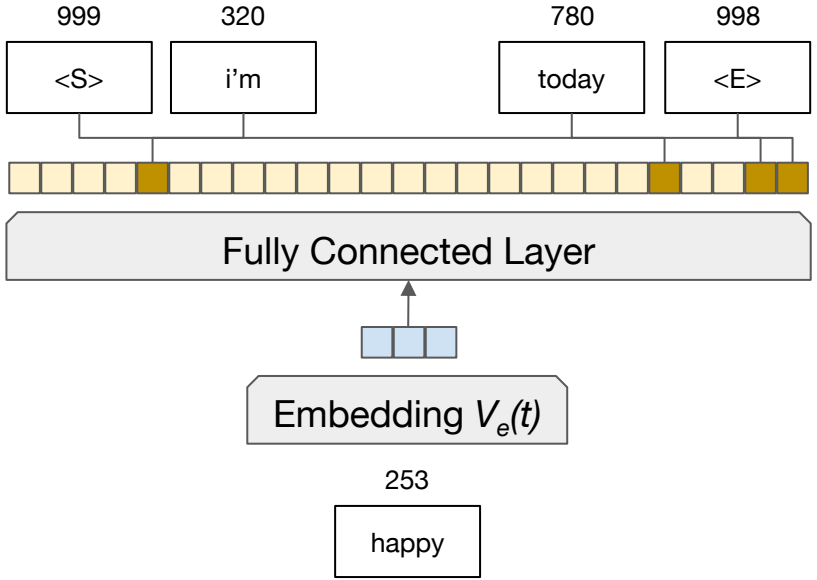


Categorical Cross Entropy; Binary Cross Entropy

Word2Vec: Recap



CBOW; $p(w_i|\{w_{i-2}w_{i-1}w_{i+1}w_{i+2}\})$



Skip-Gram;
$p(w_{i-2}|w_i)p(w_{i-1}|w_i)p(w_{i+1}|w_i)p(w_{i+2}|w_i)$

Why Learn Word Embeddings from Language Modeling?

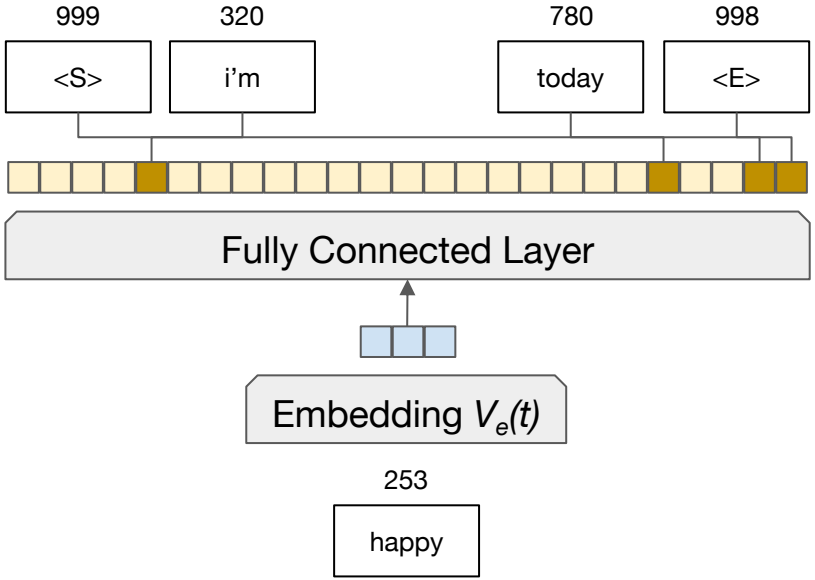
- The techniques we've shown in this lecture learn word embeddings **V** as a side-effect of:
  - estimating $p(w|\mathbf{c})$, which is itself a way to estimate:
    - $p(s)$, the fundamental function of a language model
- How else could we learn word embeddings?
  - Any task! E.g., in text classification or machine translation we can have learnable embedding layers
- So why do we *bother* learning word embeddings via LM?
- **DATA**: LM objectives are unsupervised! Only **x** given; we construct "fake" tasks out of **x** to create pairs

Why Learn Word Embeddings from Language Modeling?

- The key is in the bane of NLP:
  - The curse of dimensionality
- Word embeddings are a form of dimensionality reduction: instead of reasoning about |V| (or worse, all “possible” words), we reason about vectors of fixed dimension
- It’s easier to learn when your data isn’t extremely sparse
- Why would the word embeddings we learn as a side-effect of language modeling be beneficial downstream?
- Word embeddings make co-occurrence less sparse for estimating the *mutual information* between inputs and outputs

Word Embeddings: What are We Actually Learning?

- A language model is ultimately performing *compression* of the co-occurrence matrix
- Consider, what would we expect for each case?



- Embedding vector size:
  - |V| ?
    - *Indicator* features
  - |V|-1 ?
    - Most similar two words (synonyms) share
  - |V|/2 ?
  - $|V|^{½}$?

Word Embeddings: Pointwise Mutual Information

- In estimating latent representations that give us *co-occurrence* information, we tap into *mutual information*
- Pointwise mutual information between two events *x* and *y*:

$$\operatorname{pmi}(x;y) \equiv \log_2 \frac{p(x,y)}{p(x)p(y)}$$

- Ratio of the likelihood that *(x, y)* co-occur to the likelihood that they occur independently
  - PMI(*monte; cristo)*?
  - PMI(*the*; *a*)?

https://en.wikipedia.org/wiki/Pointwise_mutual_information

Word Embeddings: Pointwise Mutual Information

$$\operatorname{pmi}(x;y) \equiv \log_2 \frac{p(x,y)}{p(x)p(y)} = \log_2 \frac{p(x|y)}{p(x)} = \log_2 \frac{p(y|x)}{p(y)}$$

- What do these PMI formulations start to look like?
- Estimate of most likely context given a target word:
  - $\text{argmax}_{w(i-k)\in V}(p(w_{i-1}w_{i-2}\ldots w_{i-n+1}|a) / p(w_{i-1}w_{i-2}\ldots w_{i-n+1}))$
  - $p(w_{i-1}w_{i-2}\ldots w_{i-n+1}|a)$ -> co-occurrence of context and target
  - $p(w_{i-1}w_{i-2}\ldots w_{i-n+1})$ -> likelihood of context independently
- Estimate of the most likely target word given a context:
  - $\text{argmax}_{a\in V}(p(a|w_{i-1}w_{i-2}\ldots w_{i-n+1}) / p(a))$
  - Co-occ of *a* and context over independent likelihood of *a*

https://en.wikipedia.org/wiki/Pointwise_mutual_information

NLP FUNDAMENTALS: Word Embeddings

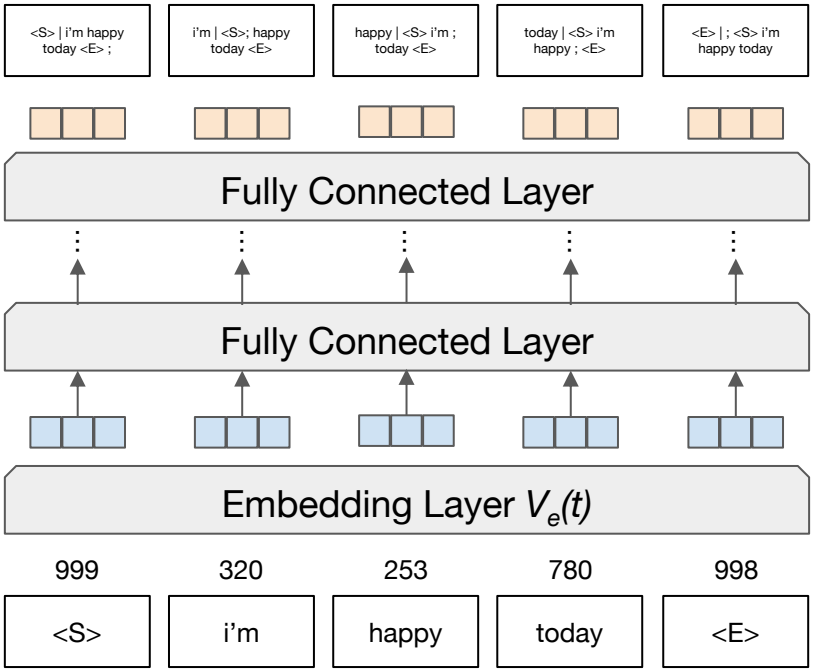
- A word embedding is a *vector representation* of a word
- Word embeddings can be learned for specific tasks (e.g., our book classification task) or from self-supervised objectives
- **“You shall know a word by the company it keeps” - Firth**
- The guiding principle of many learned word embeddings is that two words who share similar *context* should have embeddings that are close together in vector space
- **Word embedding learning algorithms estimate PMI between words and one another (their “company”)**

J.R. Firth. *A synopsis of linguistic theory*. Studies in linguistic analysis, Blackwell, Oxford (1957)

Contextual Word Embeddings

- Consider the word vectors in the following sentences:
  - **X** = {“she launched into a bass solo”,
    “the line went taut as the bass pulled”;
    “the power of the bass broke the string”}
- Many words have multiple *senses*, or meanings; even worse for BPE tokens!
- Polysemy is a key weakness of so-called *lexical* embeddings
- We can instead learn a function that combines *input* lexical embeddings from each word to produce *output* **contextual** embeddings that consider surrounding words

Contextual Word Embeddings



← **Contextual Embeddings**

← **Lexical Embeddings**

Overview of Today's Plan

- ~~Course organization and deliverables~~
- ~~Recap from Lecture 5~~
- ~~Language Models and Word Embeddings~~
    - Any questions before we move on?
- Project Pitches

Action Items for You

- Your project surveys are due **next week** on Friday [Mar 10]
- Next week we will release Coding Assignment 2 [Mar 10]
- Two weeks from today there will be **no class** [March 17]

CSCI 566: Deep Learning and Its Applications

Jesse Thomason

Lecture 6: DL for Natural Language Processing