Lexical Semantics / Word Vectors

CS 5525: Foundations of Speech and Language Processing

https://shocheen.github.io/cse-5525-spring-2025/

The Ohio State University

Sachin Kumar (kumar.1145@osu.edu)

Slide Credits: Yoav Artzi, Greg Durrett, Yulia Tsvetkov, Ana Marasović

Logistics

- Gradescope for Hw1 will be up by tonight. We will announce in teams and also update the homework instructions.
  - Any questions about the homework? (due Jan 22)

Neural Networks Basics Recap

- Why:
  - Learning the features along with model weights (representation learning)
  - learning to model more complex relationships between features than a linear model can – by stacking layers (deep learning)
- What: Neurons, hidden layers, activation functions.

$$\boldsymbol{y} = g(\mathbf{W}\boldsymbol{x} + \boldsymbol{b})$$

$$\mathbf{z} = g(\mathbf{V}\mathbf{y} + \mathbf{c})$$

$$\mathbf{z} = g(\mathbf{V}\underbrace{g(\mathbf{W}\mathbf{x} + \mathbf{b})}_{\text{output of first layer}} + \mathbf{c})$$

Building Blocks of Neural NLP

One-hot Word Representations

- Create a vocabulary of all unique tokens in your dataset (for now tokens = words, we will make it clearer next week) --- size of vocabulary: V
    - Each unique token is represented by an index in this vocabulary. For example, "hotel" could be at index 100 (the indices are arbitrary)

- Given a document with L tokens. We will represent it as a matrix of size L x V.
    - Each token is a "one hot" vector.

$$\begin{aligned} hotel &= [0 \quad 0 \quad 0 \quad \cdots 0 \quad 0 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0] \\ conference &= [0 \quad 0 \quad 0 \quad \cdots 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0 \quad 1 \quad 0 \quad 0] \end{aligned}$$

Building Blocks

Word Embeddings

- Embedding layer is the first layer in any NLP model.
- Converts one-hot representations of any word into a low dimensional "dense representations".
  - Embedding layer is linear layer represented by a simple matrix: V x D (the dimension of the representations)
- Given a document with one hot representation L x D, we multiply with the embedding matrix to get a dense representation of the document L x D (in practice implemented as a look up):
  - This matrix serves as an input to a neural network model.

Training Neural Networks

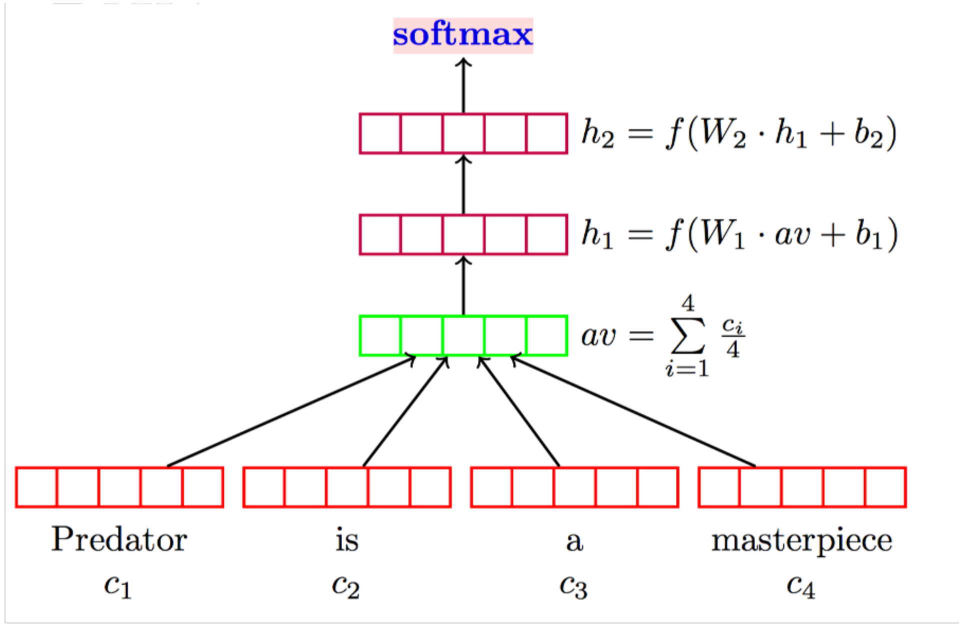
- No hidden layer → same as logistic regression (convex, guaranteed to converge)
- With hidden layers:
  - Latent units → not convex
  - What do we do? Compute gradients, apply gradient descent – but no convergence guarantees.
  - How to compute gradients: Back-propagation (aka chain rule)

Neural Bag of Words

- One of the most basic neural models
- Example: sentiment classification
  - Input: text document
  - Classes: very positive, positive, neutral, negative, very negative
- We discussed doing this with a bag-of-words feature-based model
- What would be the neural equivalent?
  - Concatenate all vectors, i.e. use the matrix L x D as the input
    - Problem: different documents → different input length L, we want a model that takes as fixed size input.
  - A Solution: Take the average of all vectors in the L X D → get a vector of size D.

Neural Bag of Words

Deep Averaging Networks (Iyyer et al. 2015)



IMDB Sentiment Analysis

BOW + linear model	88.23
NBOW DAN	89.4

Computation Graphs

- The descriptive language of deep learning models
- Functional description of the required computation
- Can be instantiated to do two types of computation:
    - Forward computation
    - Backward computation

expression:

$$\mathbf{x}$$

graph:

A **node** is a {tensor, matrix, vector, scalar} value

$$\mathbf{x}$$

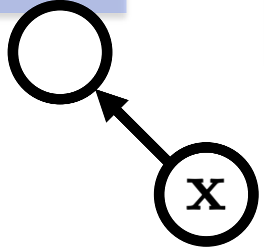
An **edge** represents a function argument (and also data dependency). They are just pointers to nodes.

A **node** with an incoming **edge** is a **function** of that edge's tail node.

A **node** knows how to compute its value and the *value of its derivative w.r.t each argument (edge) times a derivative of an arbitrary input* $\frac{\partial \mathcal{F}}{\partial f(\mathbf{u})}$.

$f(\mathbf{u}) = \mathbf{u}^\top$

$$\frac{\partial f(\mathbf{u})}{\partial \mathbf{u}} \frac{\partial \mathcal{F}}{\partial f(\mathbf{u})} = \left(\frac{\partial \mathcal{F}}{\partial f(\mathbf{u})} \right)^\top$$

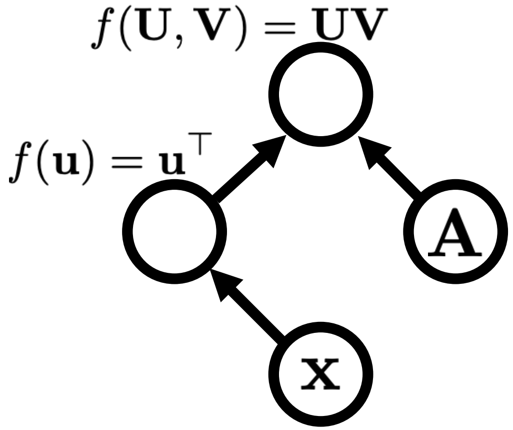


expression:

$$\mathbf{x}^\top \mathbf{A}$$

graph:

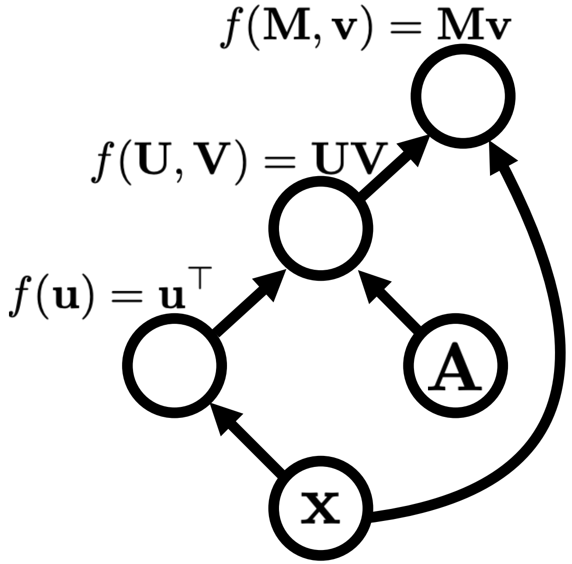
Functions can be nullary, unary,
binary, … *n*-ary. Often they are unary or binary.



expression:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x}$$

graph:

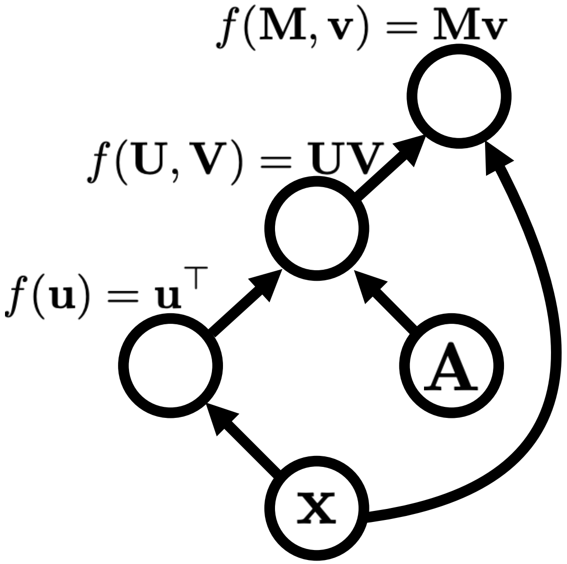


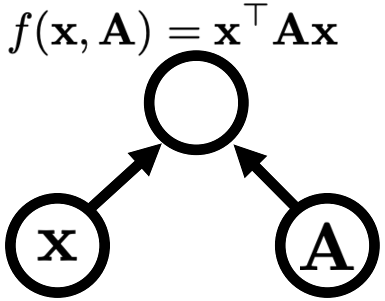
Computation graphs are directed and acyclic (usually)

expression:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x}$$

graph:





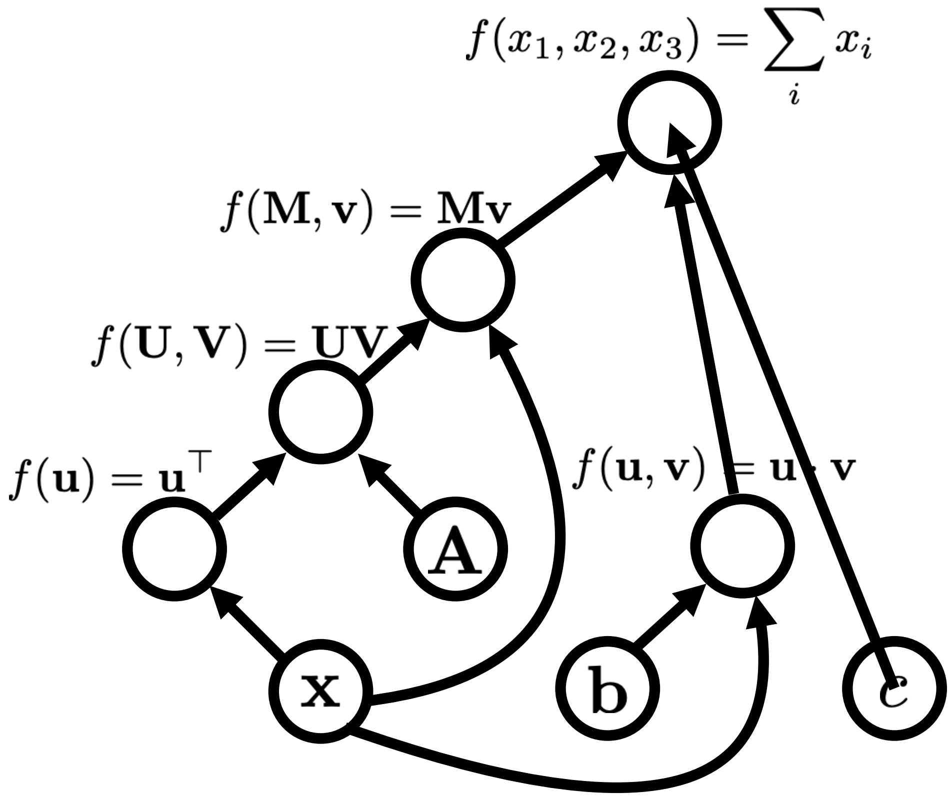
$$\frac{\partial f(\mathbf{x}, \mathbf{A})}{\partial \mathbf{x}} = (\mathbf{A}^\top + \mathbf{A})\mathbf{x}$$

$$\frac{\partial f(\mathbf{x}, \mathbf{A})}{\partial \mathbf{A}} = \mathbf{x}\mathbf{x}^\top$$

expression:

$$\mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b} \cdot \mathbf{x} + c$$

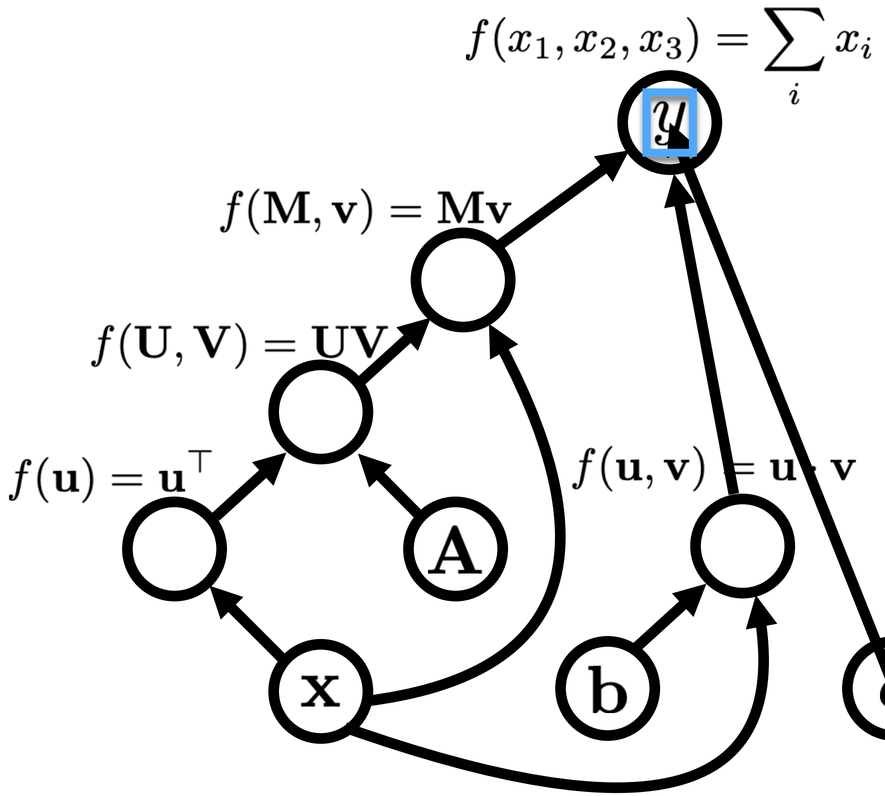
graph:



expression:

$$\boxed{y =} \mathbf{x}^\top \mathbf{A}\mathbf{x} + \mathbf{b} \cdot \mathbf{x} + c$$

graph:



variable names are just labelings of nodes.

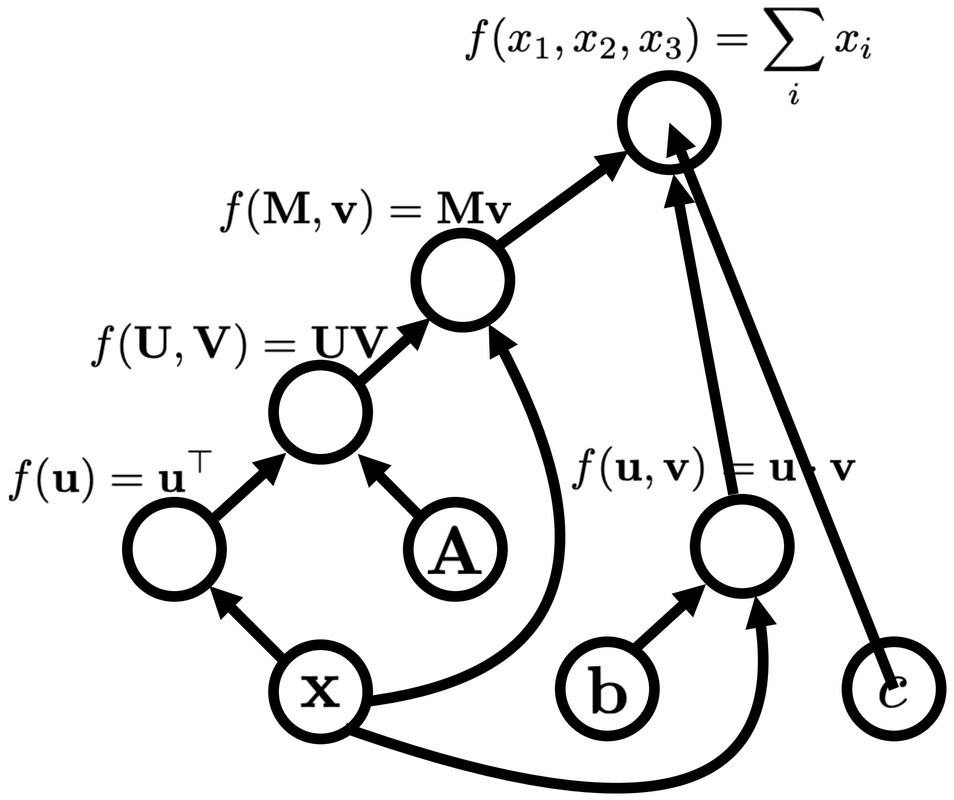
Computation Graphs

Algorithms

- **Graph construction**
- **Forward propagation**
  - Loop over nodes in topological order
    - Compute the value of the node given its inputs
  - *Given my inputs, make a prediction (or compute an "error" with respect to a "target output")*
- **Backward propagation**
  - Loop over the nodes in reverse topological order starting with a final goal node
    - Compute derivatives of final goal node value with respect to each edge's tail node
  - *How does the output change if I make a small change to the inputs?*

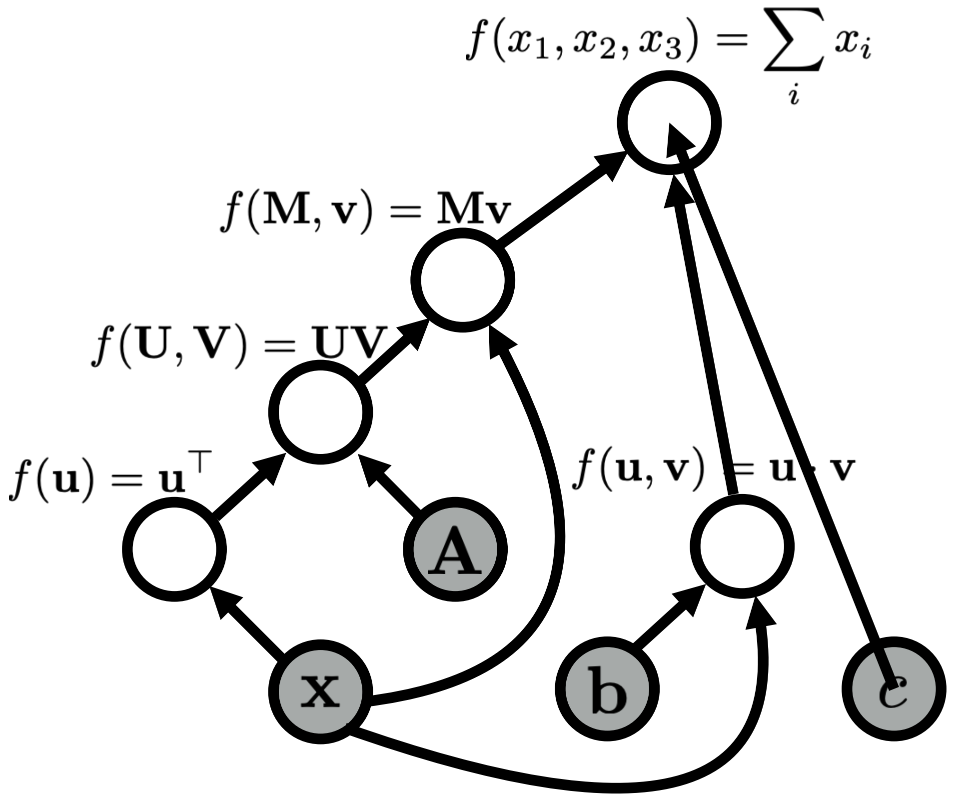
Forward Propagation

graph:



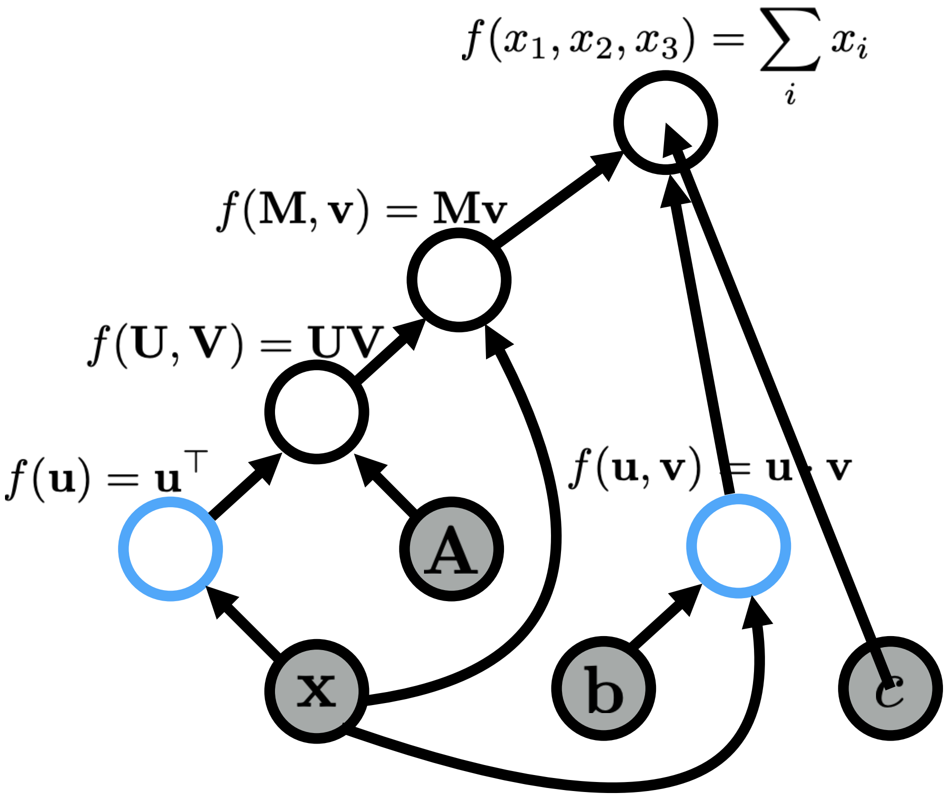
Forward Propagation

graph:



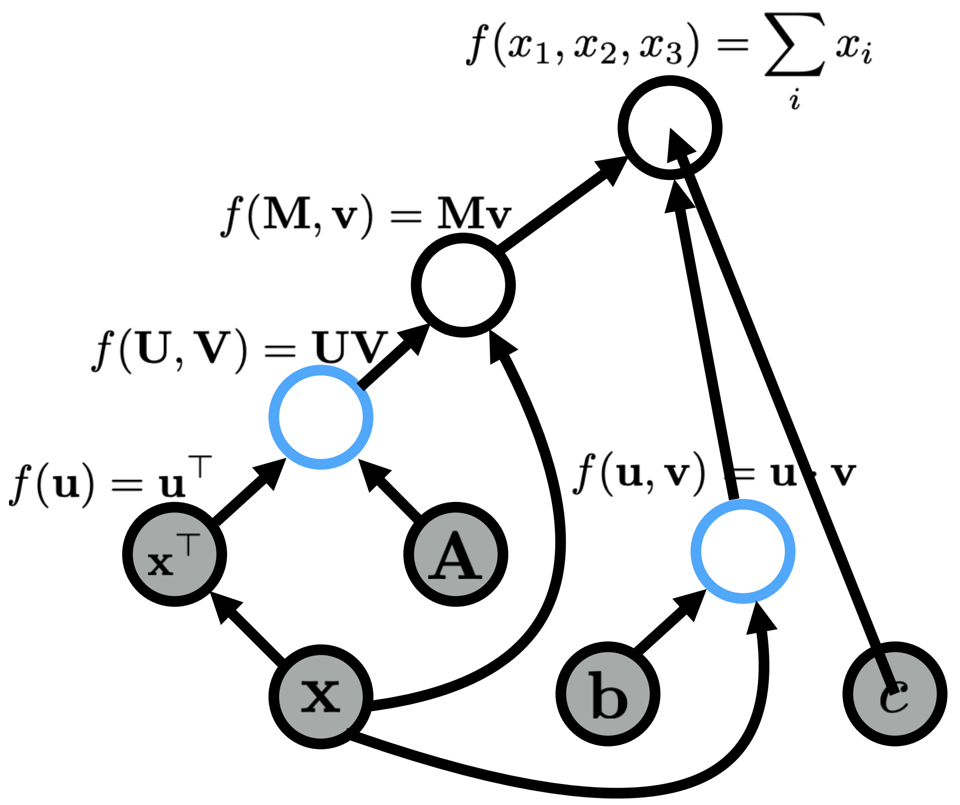
Forward Propagation

graph:



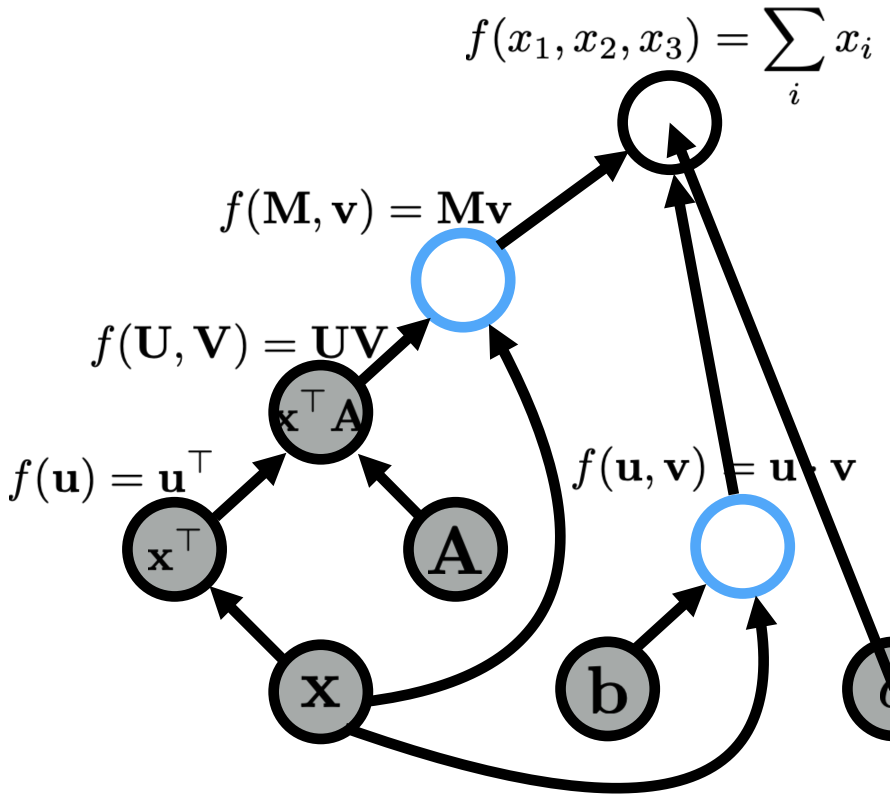
Forward Propagation

graph:



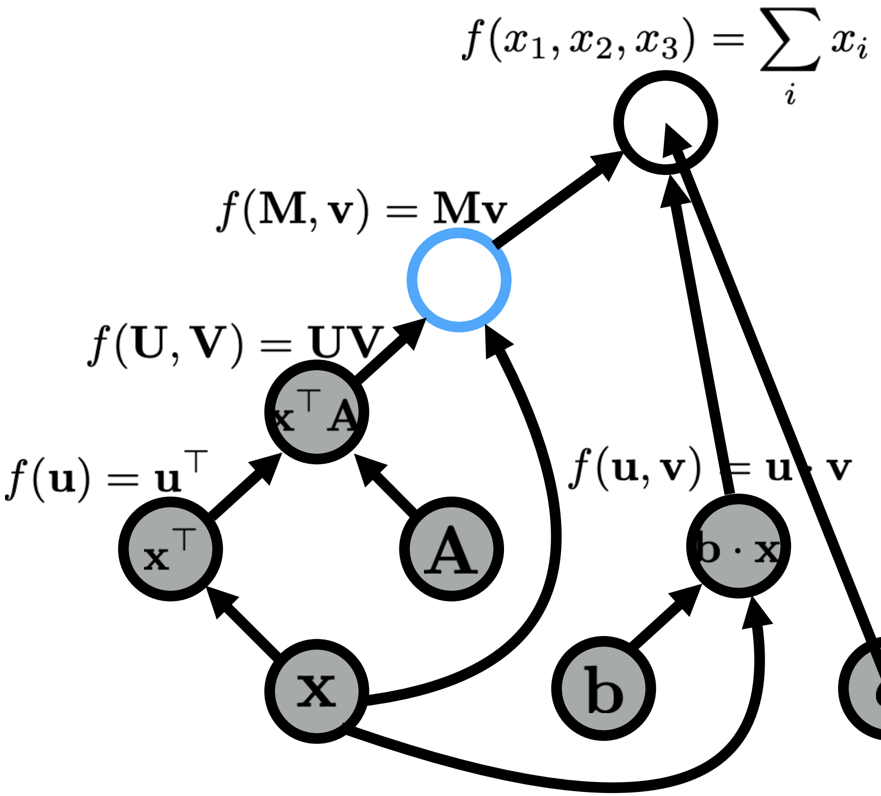
Forward Propagation

graph:



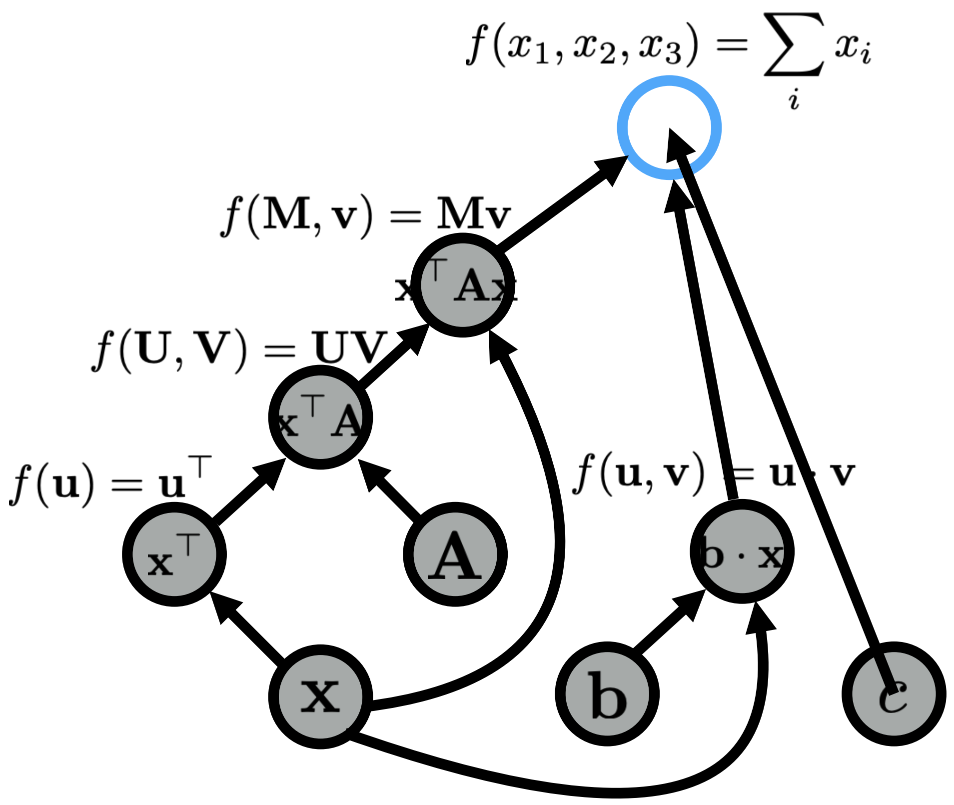
Forward Propagation

graph:



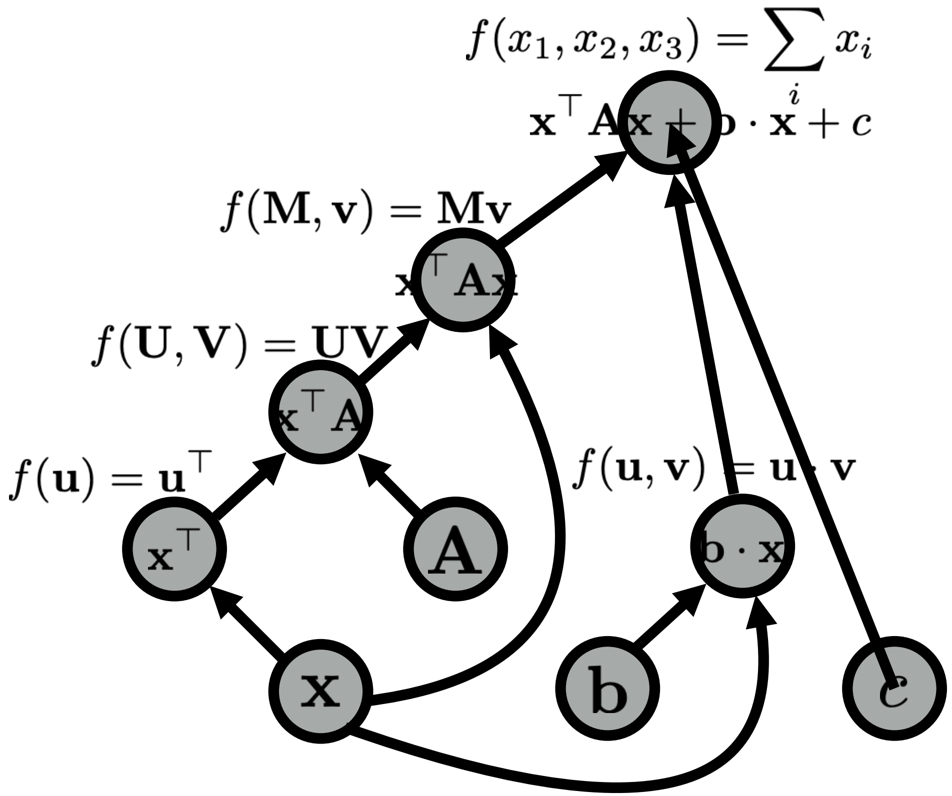
Forward Propagation

graph:



Forward Propagation

graph:



Constructing Graphs

Two Software Models

- Static declaration
  - Phase 1: define an architecture
    (maybe with some primitive flow control like loops and conditionals)
  - Phase 2: run a bunch of data through it to train the model and/or make predictions
- Dynamic declaration (a.k.a define-by-run)
  - Graph is defined implicitly (e.g., using operator overloading) as the forward computation is executed
  - Graph is constructed dynamically
  - This allows incorporating conditionals and loops into the network definitions easily

Batching

- Two senses to processing your data in batch
  - Computing gradients for more than one example at a time to update parameters during learning
  - Processing examples together to utilize all available resources
- CPU: made of a small number of cores, so can handle some amount of work in parallel
- GPU: made of thousands of small cores, so can handle a lot of work in parallel
- Process multiple examples together to use all available cores

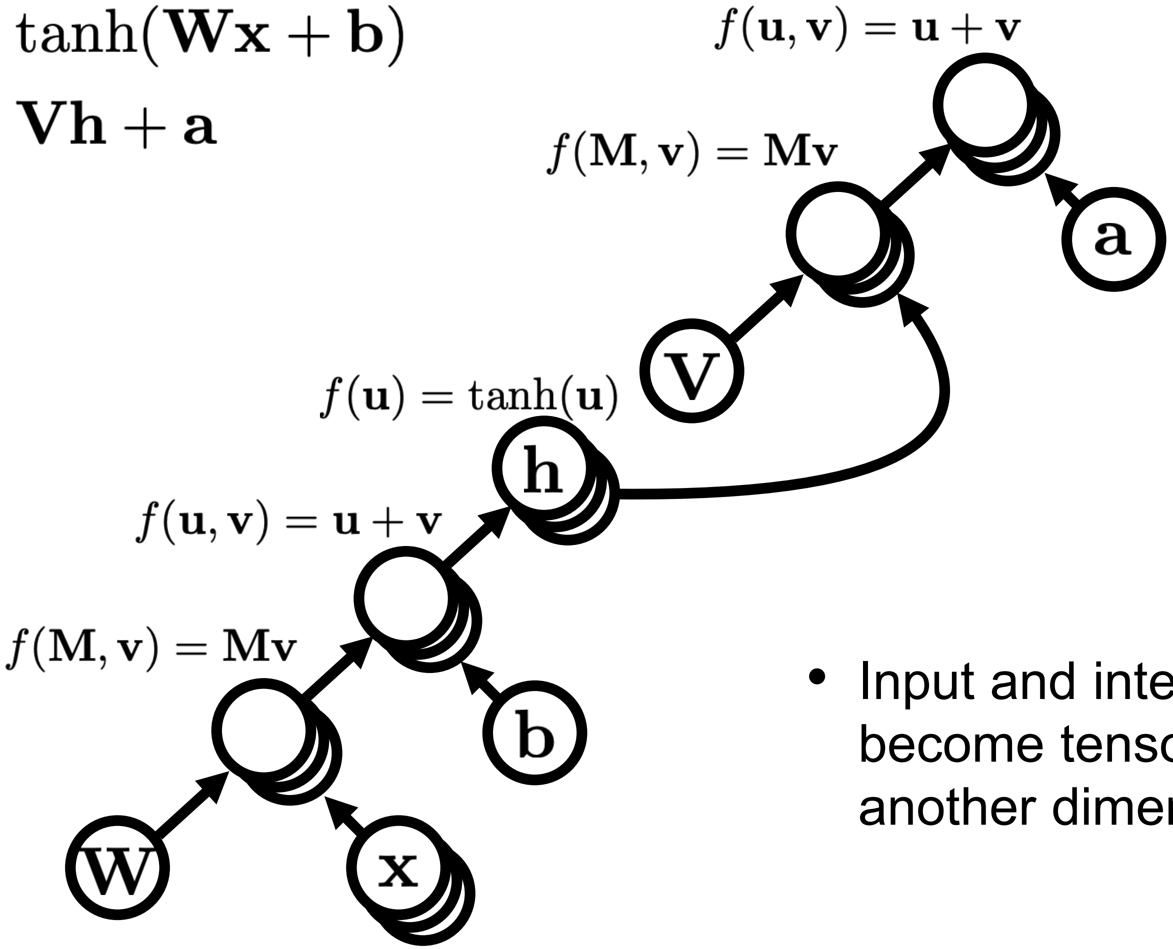
Batching

- Relatively easy when the network looks exactly the same for all examples
- More complex with language data: documents/sentences/words have different lengths
- Frameworks provide different methods to help common cases, but still require work on the developer side
- Key concept is broadcasting: https://pytorch.org/docs/stable/notes/broadcasting.html

Batching

MLP (multi-layer perceptron) Sketch

$$\mathbf{h} = \tanh(\mathbf{W}\mathbf{x} + \mathbf{b})$$
$$\mathbf{y} = \mathbf{V}\mathbf{h} + \mathbf{a}$$



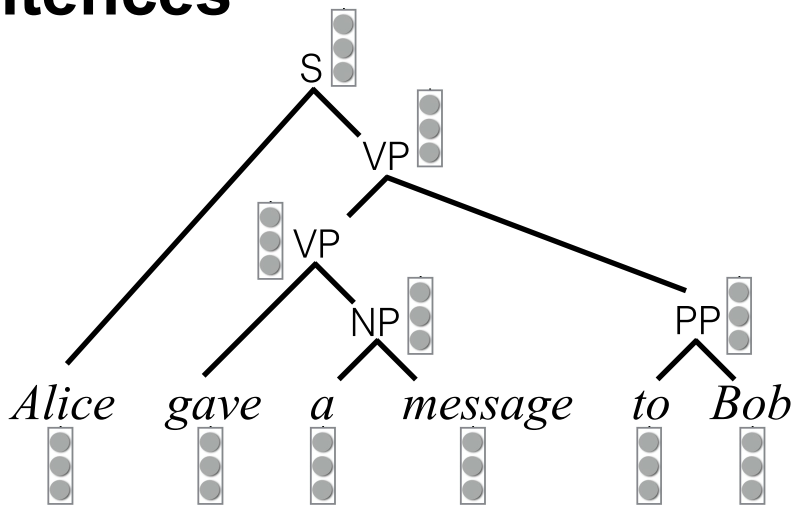
- Input and intermediate results become tensors — batch is another dimension!

Batching

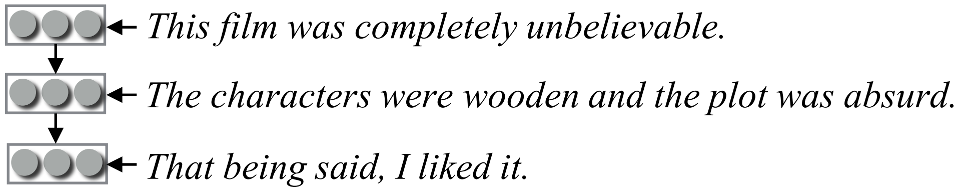
Complex Network Architectures

- Complex networks may include different parts with varying length (more about this later)
- In the extreme, it may be complex to batch complete examples this way
- But: you can still batch sub-parts across examples, so you alternate between batched and non-batched computations

Sentences



Documents



Backpropagation

How to compute the gradient w.r.t. `W_1`?

Apply the chain rule

$$\frac{\partial \mathcal{L}(x, i^*)}{\partial W_{1_{i,j}}} = \frac{\partial \mathcal{L}(x, i^*)}{\partial z} \cdot \frac{\partial z}{\partial W_{1_{i,j}}}$$

$$\frac{\partial z}{\partial W_{1_{i,j}}} = \frac{\partial g(a)}{\partial a}$$

$$a = W_1 f(x)$$

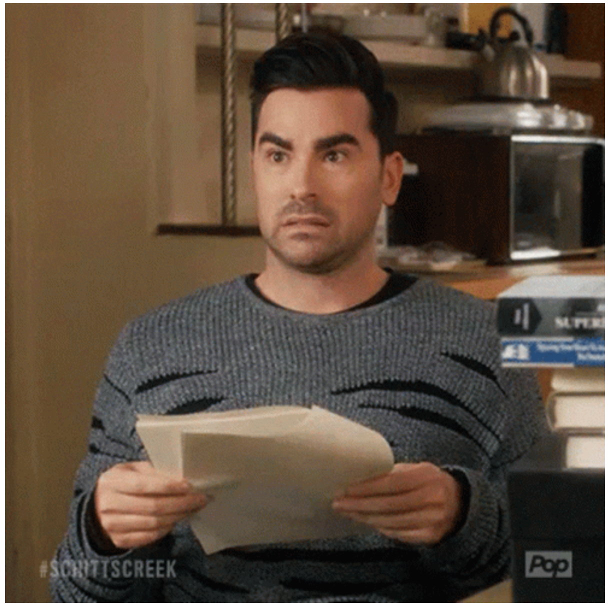
Summary: Neural Network Basics

- Neural networks allow learning complex relationships between input features but come with no learning guarantees
- How to define a feedforward neural network or an MLP
- How to create a deep averaging network (part of hw1)
- Computation graphs
  - Forward pass
  - Backward pass (or backpropagation)

Are we going to compute derivatives ourselves every time?

No, we will use frameworks that we will do them for us!

- [Deep Learning with PyTorch: A 60 Minute Blitz]



Semantics: How to represent the meaning of a word?

Desiderata

Let's look at some desiderata from lexical semantics, the linguistic study of word meaning

Word senses

lemma: the canonical form, dictionary form, or citation form of a set of word forms

basin (*plural* **basins**)

1. A wide bowl for washing, sometimes affixed to a wall. [quotations ▼] [synonym ▲]

   Synonym: sink

2. (*obsolete*) A shallow bowl used for a single serving of a drink or liquidy food. [quotations ▼]
3. A depression, natural or artificial, containing water. [quotations ▼]
4. (*geography*) An area of land from which water drains into a common outlet; drainage basin. [quotations ▼]
5. (*geography*) A shallow depression in a rock formation, such as an area of down-folded rock that has accumulated a thick layer of sediments,

Source: wiktionary

word senses: meanings of the word

Polysemous words: words having multiple senses

Word sense disambiguation

Word Senses

Who Cares?

- Capturing such sense distinctions is important for many NLP problems
- Including very practical ones:
  - Information retrieval / question answering
    - bat care / how do I care for my bat?
  - Machine translation
    - bat: murciélago (animal) or bate (for baseball)
  - Text-to-speech
    - bass (stringed instrument) vs. bass (fish)

Word Senses

Who Cares?

- Can break common semantic expectations
- So an interesting test case for even the latest and largest model
- For example, GPT4V
    - *generate an image of a baseball player caring for his bat in the cave where he lives with all the other bats*

Word Senses

Who Cares?

- Can break common semantic expectations
- So an interesting test case for even the latest and largest model
- For example, GPT4V
    - *generate an image of a baseball player taking care of his bat, who is living in a cave*

Relation: synonymity

Synonyms have the same meaning in some or all contexts.

- filbert / hazelnut
- couch / sofa
- big / large
- automobile / car
- vomit / throw up
- Water / H_2o

Synonyms

Two words are synonymous if they are substitutable for one another in any sentence without changing the truth conditions of the sentence [the situations in which the sentence would be true]

- **Principle of contrast:** A difference in linguistic form is always associated with some difference in meaning [Clark 1987]
    - H2o/water

Word similarity

Not synonyms, but sharing some element of meaning

- belief, impression
- skiing, snowboarding

How similar two words are? ⇒ How similar the meaning of two sentences are?

Antonyms

Senses that are opposites with respect to only one feature of meaning

Antonyms can

- Define a binary opposition or be at opposite ends of a scale
  - hot/cold
- Be reversives:
  - ascend/descend

Ask humans how similar two words are

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999 dataset (Hill et al., 2015)

Relation: word relatedness

Also called "word association"

- Words be related in any way, perhaps via a semantic frame or field
    - car, bicycle: similar
    - car, gasoline: related, not similar

Lexical semantics

- How should we represent the meaning of the word?
  - Dictionary definition
  - Lemma and wordforms
  - Senses
  - Relationships between words or senses
  - Word similarity, word relatedness
  - Semantic frames and roles
  - Connotation and sentiment

Lexical semantics

- How should we represent the meaning of the word?
  - Dictionary definition
  - Lemma and wordforms
  - Senses
  - Relationships between words or senses
  - Word similarity, word relatedness
  - Semantic frames and roles
    - *John hit Bill*
    - *Bill was hit by John*

Lexical Semantics

- How should we represent the meaning of the word?
  - Dictionary definition
  - Lemma and wordforms
  - Senses
  - Relationships between words or senses
  - Word similarity, word relatedness
  - Semantic frames and roles
  - Connotation and sentiment
    - *valence*: the pleasantness of the stimulus
    - *arousal*: the intensity of emotion
    - *dominance*: the degree of control exerted by the stimulus

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24
life	6.68	5.59	5.89

Lexical Semantics are discrete and sparse

- Hard to use in machine learning models which expect continuous inputs

Distributional Semantics

Artemia

A cluster of _________ is floating in the lake.

Biologists study the adaptation of _________ in saline environments.

The population of _________ fluctuates with the salinity of the water.

You can observe _________ in the shallows of the Great Salt Lake.

Other words that can appear in this context: *algae*, *microorganisms*, *shrimp*

Other words that can appear in this context: *algae*, *microorganisms*, *shrimp*

We can conclude:

➔ Artemia is a simpler form of life found in aquatic environments like the Great Salt Lake similar to algae, microorganisms, shrimp

Distributional hypothesis

[Joos, 1950; Harris, 1994; Firth, 1957]

Words that occur in **similar contexts** tend to have **similar meanings**

Distributional Semantics

The Distributional Hypothesis

- Words that are used and occur in the same **<u>context</u>** tend to have similar meaning
- Similarity-based generalization: children can figure out how to **<u>use</u>** words by generalizing about their **<u>use</u>** from distributions of similar words
- The more semantically similar words are, the more distributionally similar they are
- **What is context**? Informally: whatever you can get your hands on that makes sense!

Learning from Raw Data

Word Vectors

Raw Data

- Raw text = human-created language without any additional annotation
- A natural by-product of human use of language
- Abundant in text form for many domains and scenarios, but not for all
- How can learn without any annotation? What kind of representations can we get? How can we use them?
- Key idea: self-supervised learning

Raw Data

Self-supervised Learning

- Given: raw data without any annotation
- Formalize a prediction training objective that is using this data only
- Common approach: given one piece of the data, predict another
- The prediction task is often not interesting on its own
- But the learned representations are!
- Big advantage: can use as much data as you can find and have compute for
- In contrast, supervised learning relies on enriching the data with human annotations

Vectors semantics

Lexical semantics is the linguistic study of word meaning

Vector semantics instantiates distributional hypothesis by **learning (vector) representations** of the meaning of words directly from their **distributions** in text

Embeddings

- In mathematics: A mapping from one space or structure to another
- The term grew out the **latent semantic indexing model** recast as **LSA** [Deerwester et al., 1990]
- Each discrete token is embedded in a continuous vector space
- Short, dense

A Sparse Representation
Counting contexts

- Given a vocabulary of V words
- Let $f_i, i = 1 \dots V$ be a binary (or count) indicator for the presence (or count) of the i-th word in the vocabulary
- Represent a word w as:

$$w = (f_1, f_2, f_3, \dots, f_n)$$

where f_i are computed in contexts of all uses of w

- For example:

$$\text{Artemia} = (1,1,0,\dots)$$

lake saline biologists

word2vec

word2vec is a **software** package (https://code.google.com/archive/p/word2vec/) that includes **two algorithms** [Mikolov et al., 2013a; Mikolov et al., 2013b]

1. **Skip-gram** with negative sampling (SGNS) [now]
2. Continuous Bag-Of-Words (**CBOW**) [in the readings]

These algorithms are often loosely referred to as word2vec

The intuition behind word2vec

Instead of counting how often each word `w` occurs near another word, *artemia*, train a classifier on a binary prediction task:

➔ Is word `w` likely to show up near *artemia*?

Specifically, with skip-gram

- Use the target word & a neighboring context word (from a corpus) as positive examples
- Randomly sample other words as negative examples
- Train a classifier to distinguish those two cases
- Use the learned weights as the embeddings

Skip-gram classifier – Intuition

```
... lemon,   a [tablespoon of apricot jam,      a] pinch ...
                c1          c2    w      c3         c4
```

$$p(+|w,c) = 1$$

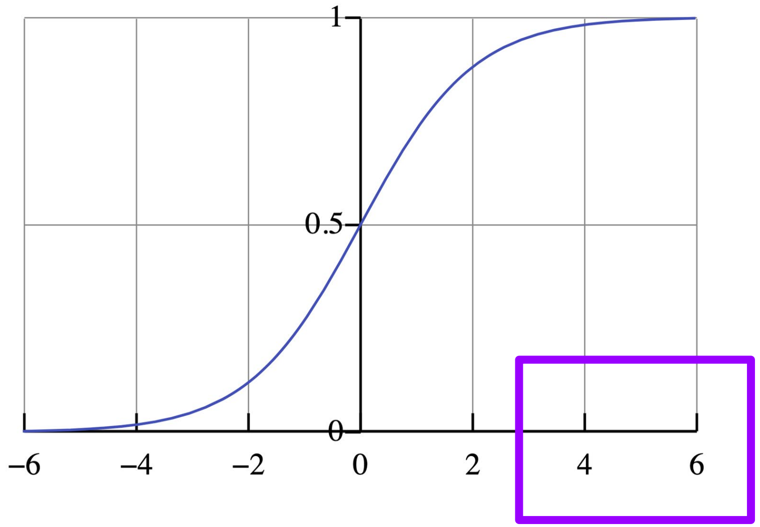
$$f(z) = \frac{e^z}{1+e^z} \ldots \text{logistic function}$$

$$p(+|\text{apricot,tablespoon}) = 1$$

$$p(+|\text{apricot,of}) = 1$$

$$p(+|\text{apricot,jam}) = 1$$

$$p(+|\text{apricot,a}) = 1$$



embedding similarity high
⇒ probability high too

Skip-gram classifier – Intuition

```
... lemon,   a [tablespoon of apricot jam,      a] pinch ...
               c1         c2     w      c3      c4
```

$$p(+|w,c) = 1$$

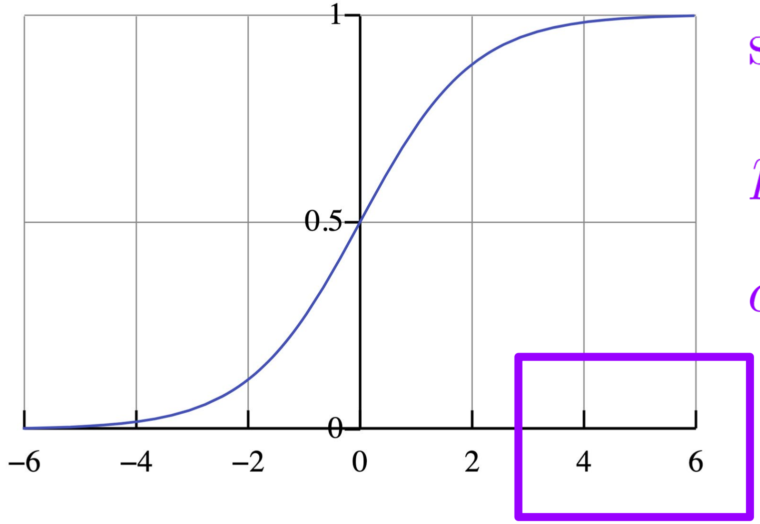
$$f(z) = \frac{e^z}{1+e^z} \ldots \text{logistic function}$$

$p(+|\text{apricot,tablespoon}) = 1$

$p(+|\text{apricot,of}) = 1$

$p(+|\text{apricot,jam}) = 1$

$p(+|\text{apricot,a}) = 1$



$\text{similarity}(w,c) \approx c \cdot w$

$$p(+|w,c) = \frac{e^{c \cdot w}}{1+e^{c \cdot w}}$$

$c \cdot w \to \infty \Rightarrow p(+|w,c) \to 1$

Skip-gram classifier

$$P(+|w, c_{1:L}) = \prod_{i=1}^{L} p(+|w, c_i) = \prod_{i=1}^{L} \frac{e^{c_i \cdot w}}{1 + e^{c_i \cdot w}}$$

$$\log P(+|w, c_{1:L}) = \sum_{i=1}^{L} \log \frac{e^{c_i \cdot w}}{1 + e^{c_i \cdot w}}$$

Skip-gram learning algorithm

Given:

- Set of **positive** and **negative examples**
- An **initial** set of **random embeddings**

The goal of the learning algorithms it to **adjust** those embeddings to:

- Maximize the similarity of the target word, context word pairs `(w,c_pos)` drawn from the positive examples
- Minimize the similarity of `(w,c_neg)` pairs from the negative examples

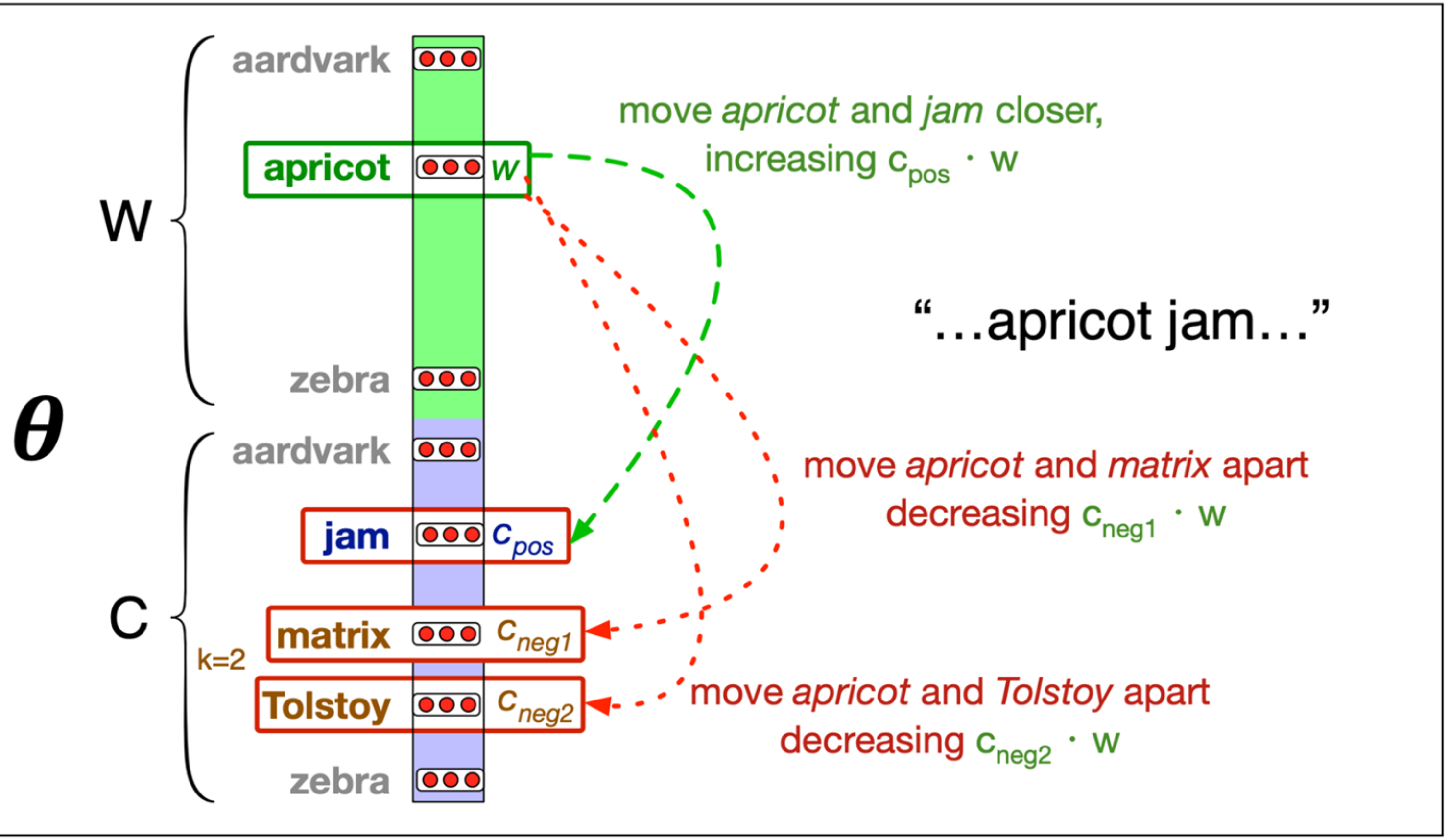
$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos})+\sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos})+\sum_{i=1}^{k}\log\left(1-P(+|w,c_{neg_i})\right)\right] \\
&= -\left[\log\sigma(c_{pos}\cdot w)+\sum_{i=1}^{k}\log\sigma(-c_{neg_i}\cdot w)\right]
\end{aligned}
$$

Skip-gram learning algorithm – Stochastic gradient descent

$$\frac{\partial L_{CE}}{\partial c_{pos}} = [\sigma(\mathbf{c}_{pos} \cdot \mathbf{w}) - 1]\mathbf{w}$$

$$\frac{\partial L_{CE}}{\partial c_{neg}} = [\sigma(\mathbf{c}_{neg} \cdot \mathbf{w})]\mathbf{w}$$

$$\frac{\partial L_{CE}}{\partial w} = [\sigma(\mathbf{c}_{pos} \cdot \mathbf{w}) - 1]\mathbf{c}_{pos} + \sum_{i=1}^{k}[\sigma(\mathbf{c}_{neg_i} \cdot \mathbf{w})]\mathbf{c}_{neg_i}$$



Word Embeddings

How to Use Them?

- Word embeddings are often input to models of various end applications
- They provide lexical information beyond the annotated task datasets, which is often small
- Can be kept fixed or fine tuned (i.e. trained) with the task network
- Can also be input to sentence embedding models

Visualizations

Project embeddings to a 2D space and visualize them

- [How to Use t-SNE Effectively]

Check k-nearest neighbors

[Li et al., 2016]

Measuring Vector Similarity

- Similarity can be measured using vector distance measures
- Two typical examples: Euclidean distance and cosine similarity
- Cosine similarity:

$$\mathrm{similarity}(w,u) = \frac{w \cdot u}{\parallel w \parallel \parallel u \parallel} = \frac{\sum_{i=1}^{n} w_i u_i}{\sqrt{\sum_{i=1}^{n} w_i^2}\sqrt{\sum_{i=1}^{n} u_i^2}}$$

which gives values between -1 (completely different), 0 (orthogonal), and 1 (completely identical)

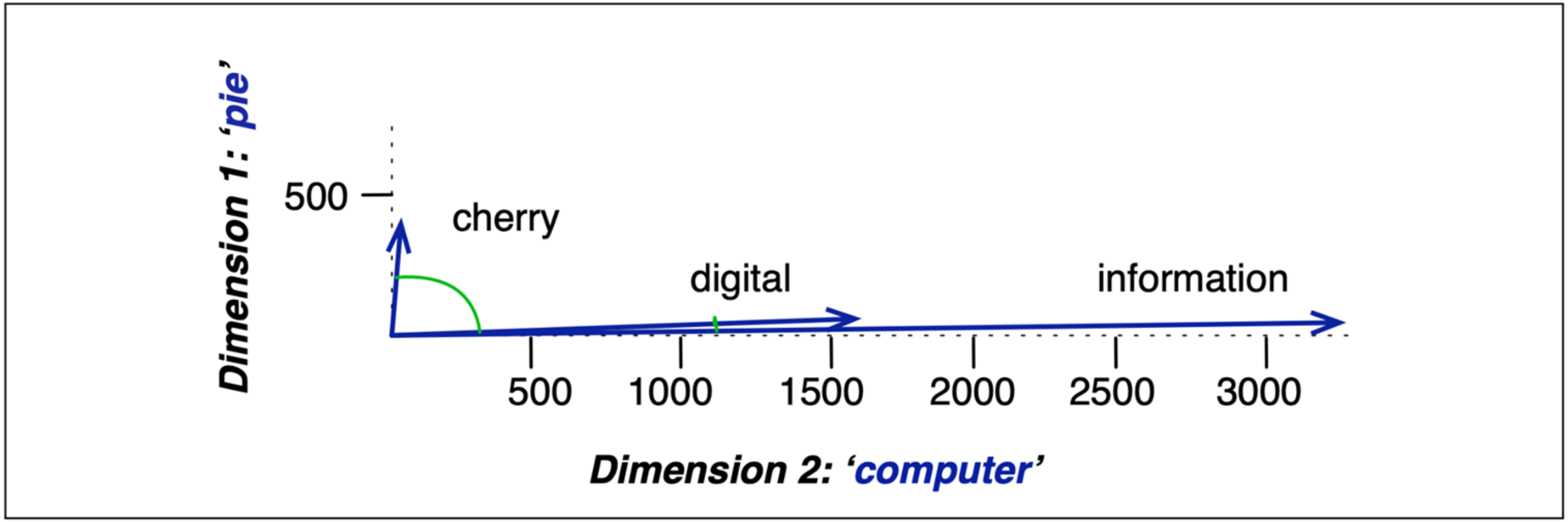
Measuring vector similarity

Cosine similarity: The angle between the vectors

$$\cos(v, u) = \frac{v \cdot u}{||v|| \cdot ||u||}$$

The cosine similarity of unit vectors is the same as their dot product

The cosine similarity determines the similarity based solely on the directions and ignores the magnitudes



Other kinds of static embeddings

Fasttext [Bojanowski et al, 2017]

- Limitation of word2vec: a distinct vector representation for each word, but we learned about subwords and their benefits
- An extension which takes into account subword information
- https://fasttext.cc/

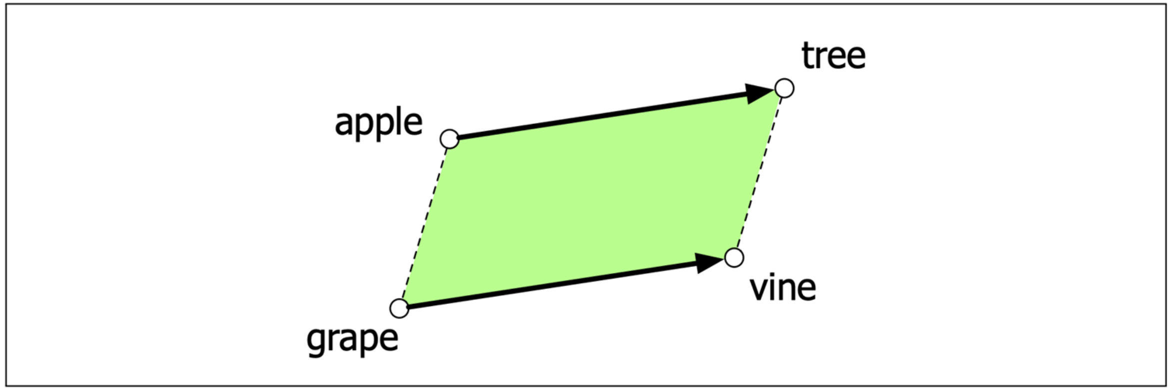
GloVe [Pennington et al., 2014]

Analogy/Relational Similarity

Embeddings capture relational meanings

Analogy problems:

- ➔ *a is to b as a* is to what?*
- ➔ *a:b::a*:b**
- ➔ *apple:tree::grape:?*
- ➔ *king:man::woman:?*
- ➔ *Paris:Frace::Italy:?*



Add the vector from the word *apple* to the word *tree*, v(tree)-v(apple), to the vector of the grape, v(grape)

The nearest word to that point is returned

[The (too Many) Problems of Analogical Reasoning with Word Vectors]

$$\hat{b} = \text{argmin}_x \text{distance}(x, b - a + a^*)$$

Societal biases

computer programmer - man + woman = homemaker [Bolukbasi et al., 2016]

doctor - man + woman=nurse

Downstream impact: A tool for hiring doctor or programmers downweights documents with women's names

Allocation harm: a system allocates resources (jobs or credit) unfairly to different groups [Blodgett et al., 2020]

Bias amplification: gendered terms become more gendered in embeddings spaces than they were in the input text statics [Jia et al., 2020]

Representational harm: Harm caused by a system demeaning or even ignoring some social groups

- Names like "Leroy" have a higher cosine similarity with unpleasant words while names like Brad, Greg, Courtney have a higher cosine with pleasant words [Zhou et al., 2022]

Debiasing is very hard [Gonen and Goldberg, 2019]

Dependency Structures

A Linguistic Detour

- A structural formalism of sentence structure
- Will provide a framework to think beyond adjacency contexts
    - More generally: it is model of sentence structure
- Dependency structure shows which words depend on (modify or are arguments of) which other words

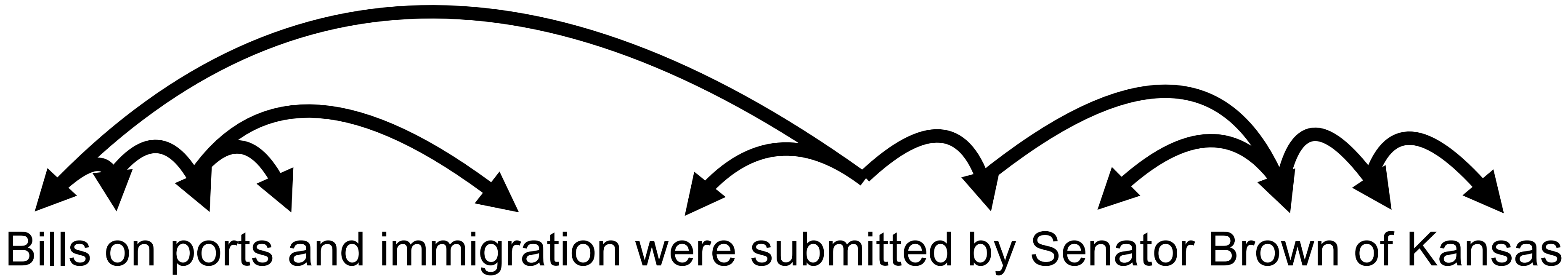
Dependency Structures

- A syntactic structure that consists of:
  - Lexical items (words)

Bills on ports and immigration were submitted by Senator Brown of Kansas

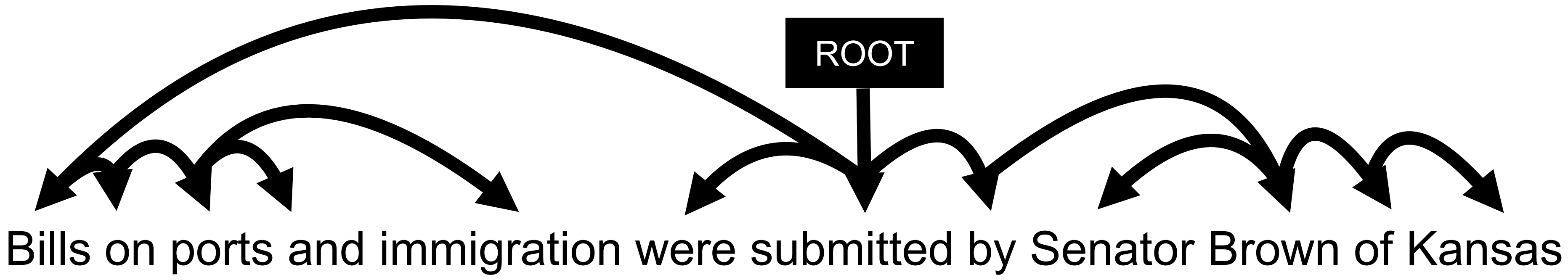
Dependency Structures

- A syntactic structure that consists of:
  - Lexical items (words)
  - Binary asymmetric relations → dependencies
    - Arrow usually from **head** to **modifier**



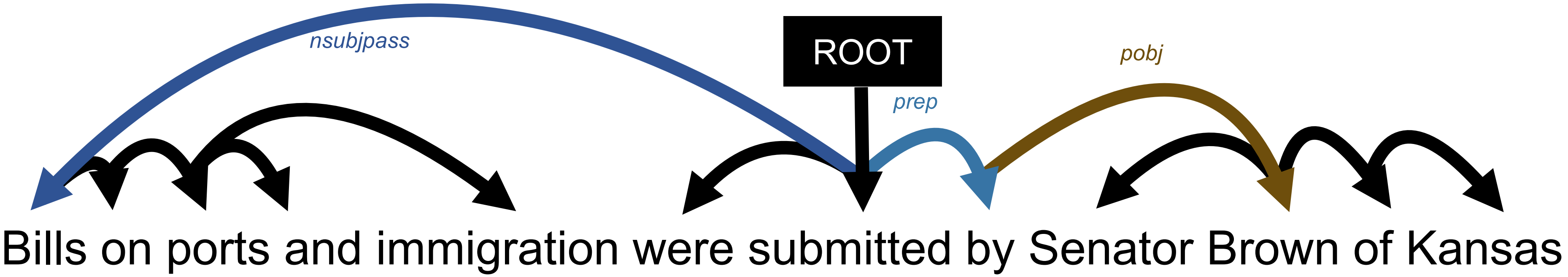
Dependency Structures

- A syntactic structure that consists of:
  - Lexical items (words)
  - Binary asymmetric relations → dependencies
- Dependencies form a tree with a standard root node



Dependency Structures

- A syntactic structure that consists of:
  - Lexical items (words)
  - Binary asymmetric relations → dependencies
- Dependencies form a tree with a standard root node
- Dependencies are typed with names of grammatical relations

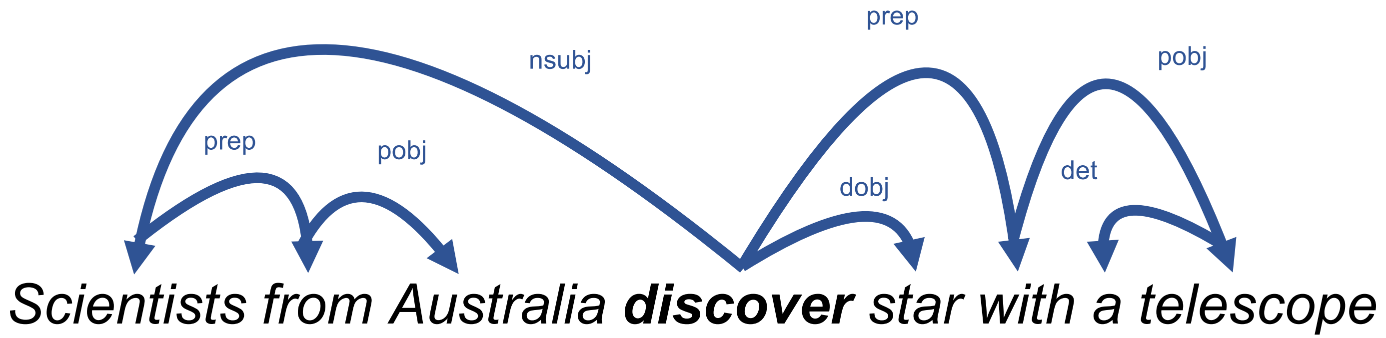


Word2vec

Structured Contexts

- Dependency structures allow us to consider notions of adjacency beyond just neighboring words in the text
- Because we can look at the dependency structure connectivity
- These edges can connect words at arbitrary distances
  - If they have a syntactic relation between them

Word2vec
Dependency Contexts



[Levy and Goldberg 2014]

Word2vec
Dependency Contexts

- What is learned?
- What is the cost?

Target Word	BoW5	BoW2	Deps
batman	nightwing aquaman catwoman superman manhunter	superman superboy aquaman catwoman batgirl	superman superboy supergirl catwoman aquaman
hogwarts	dumbledore hallows half-blood malfoy snape	evernight sunnydale garderobe blandings collinwood	sunnydale collinwood calarts greendale millfield
turing	nondeterministic non-deterministic computability deterministic finite-state	non-deterministic finite-state nondeterministic buchi primality	pauling hotelling heting lessing hamming
florida	gainesville fla jacksonville tampa lauderdale	fla alabama gainesville tallahassee texas	texas louisiana georgia california carolina
object-oriented	aspect-oriented smalltalk event-driven prolog domain-specific	aspect-oriented event-driven objective-c dataflow 4gl	event-driven domain-specific rule-based data-driven human-centered
dancing	singing dance dances dancers tap-dancing	singing dance dances breakdancing clowning	singing rapping breakdancing miming busking

Table 1: Target words and their 5 most similar words, as induced by different embeddings.

[Levy and Goldberg 2014]