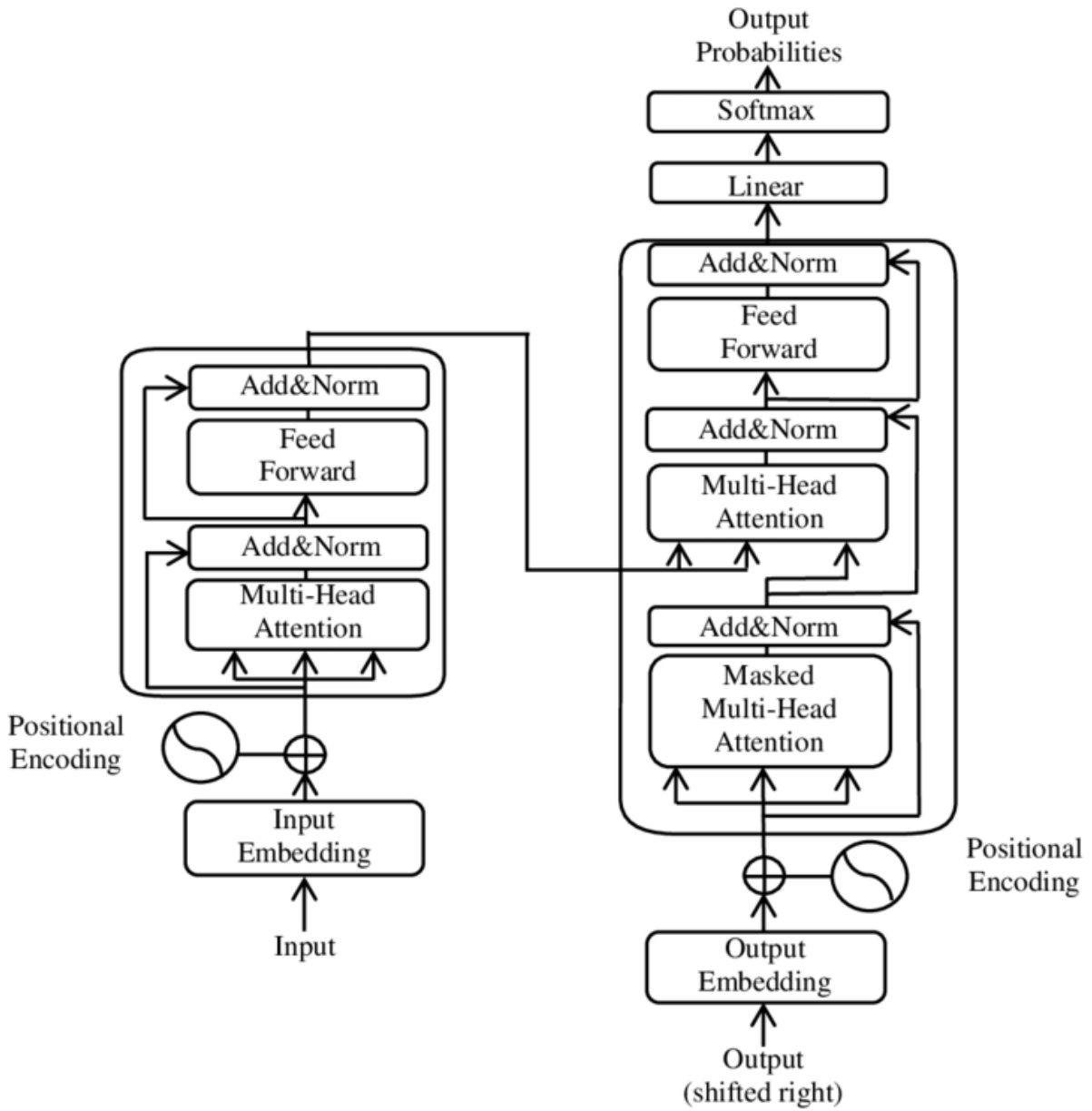
Deep Natural Language Processing

Mark Hasegawa-Johnson

5/2023

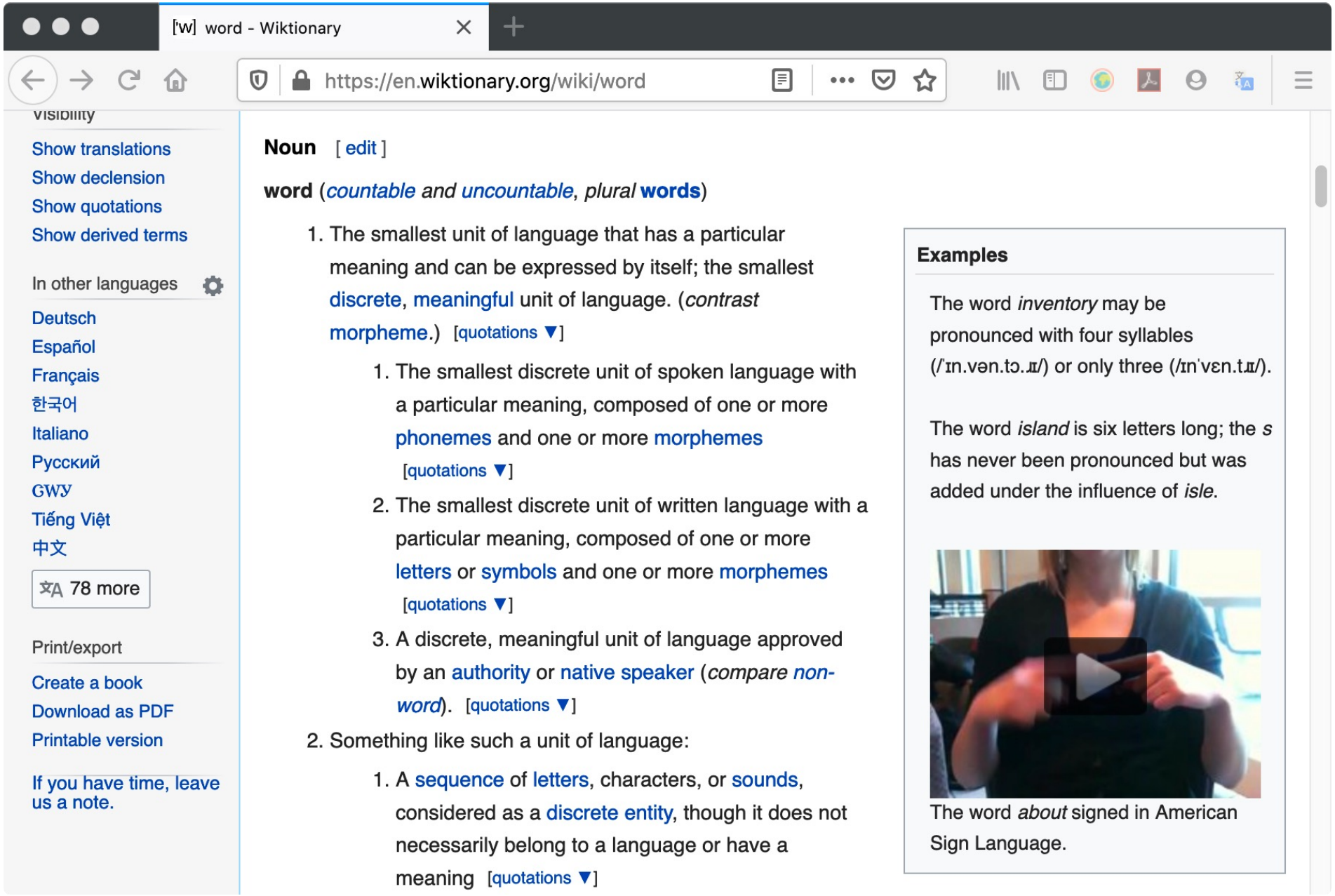


By Yuening Jia - DOI:10.1088/1742-6596/1314/1/012186, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=121340680

Outline

- Lexical semantics
  - Wordforms, lemmas & word senses; Synonymy & similarity
  - Vector semantics: CBOW and skip-gram
  - Generative training vs. Contrastive training
  - Implicit meaning representations in lexical semantic spaces
- Sentence semantics
  - Recurrent neural networks (RNN)
  - Attention
  - Self-attention, Multi-headed attention, Cross-attention, and Masked attention
  - Self-training

What is a word?



Wordform

A wordform is a unique sequence of characters.

- Wordforms are much easier for computers to find than lemmas, therefore most automatic processing deals with wordforms.
- …however, we lose something. “dog” and “dogs” become completely unrelated – as unrelated as “dog” and “exaggerate.”

word (*countable and uncountable*, plural **words**)

1. The smallest unit of language that has a particular meaning and can be expressed by itself; the smallest discrete, meaningful unit of language. (*contrast* morpheme.) [quotations ▼]
    1. The smallest discrete unit of spoken language with a particular meaning, composed of one or more phonemes and one or more morphemes [quotations ▼]
    2. The smallest discrete unit of written language with a particular meaning, composed of one or more letters or symbols and one or more morphemes [quotations ▼]
    3. A discrete, meaningful unit of language approved by an authority or native speaker (*compare non-word*). [quotations ▼]
2. Something like such a unit of language:
    1. A sequence of letters, characters, or sounds, considered as a discrete entity, though it does not necessarily belong to a language or have a meaning [quotations ▼]

Lemma

A lemma is what humans usually think of as a “word.” It is defined to be the form of the word that appears in a dictionary.

- Other wordforms that can be easily predicted from the lemma need not be listed.

word (*uncountable and uncountable*, *plural* **words**)

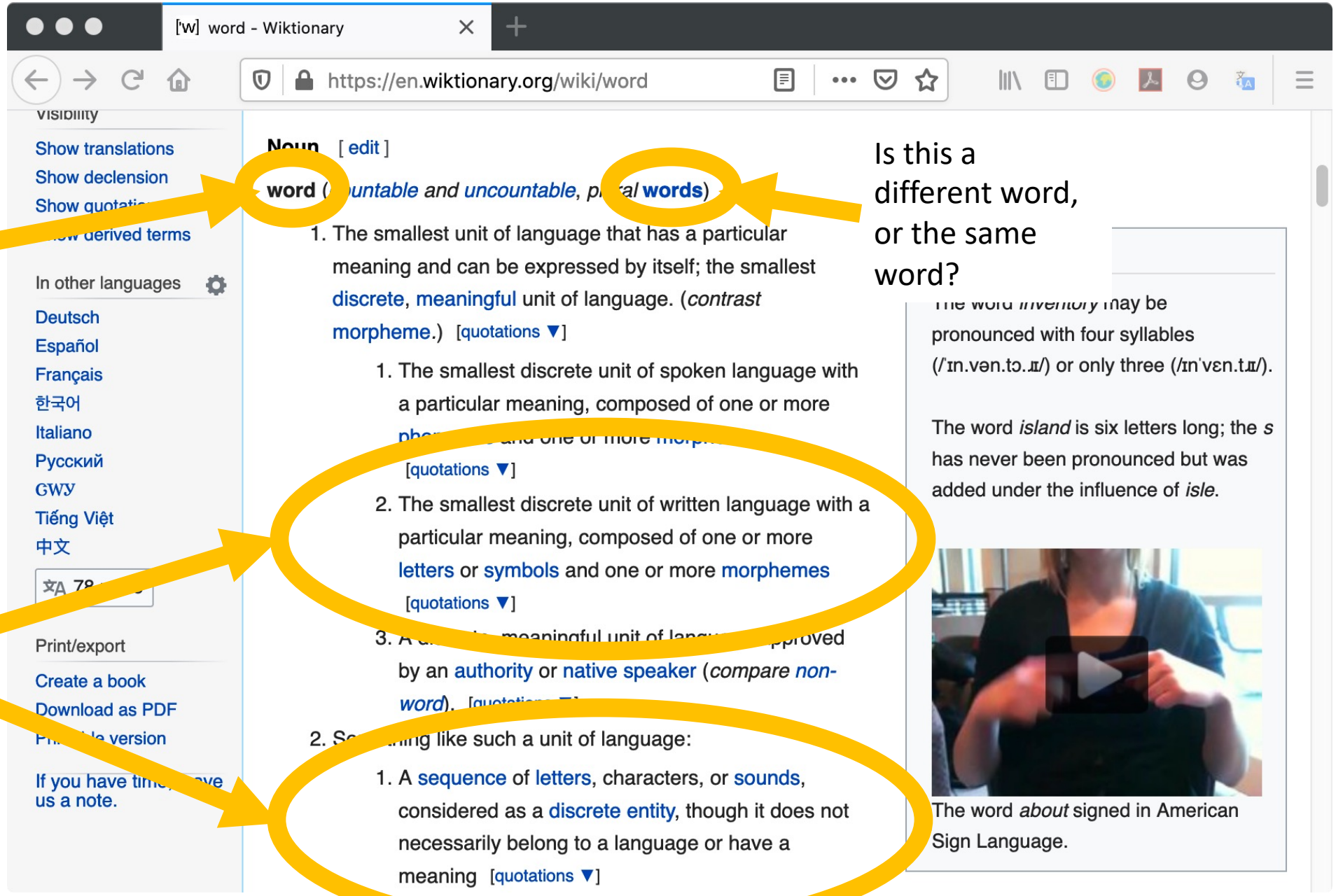
1. The smallest unit of language that has a particular meaning and can be expressed by itself; the smallest discrete, meaningful unit of language. (*contrast* morpheme.) [quotations ▼]
    1. The smallest discrete unit of spoken language with a particular meaning, composed of one or more phonemes and one or more morphemes [quotations ▼]
    2. The smallest discrete unit of written language with a particular meaning, composed of one or more letters or symbols and one or more morphemes [quotations ▼]
    3. A discrete, meaningful unit of language approved by an authority or native speaker (*compare non-word*). [quotations ▼]
2. Something like such a unit of language:
    1. A sequence of letters, characters, or sounds, considered as a discrete entity, though it does not necessarily belong to a language or have a meaning [quotations ▼]

What is a word?

Is this a word?

Are these the same word, or different words?

Is this a different word, or the same word?



Word sense

Often, a word has different meanings that are completely unrelated. We think of them as different words, that just happen to be spelled and pronounced the same way.

We say that these are different “senses” of the same word.

The Bank of England. By Diliff - Own work, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=40912212

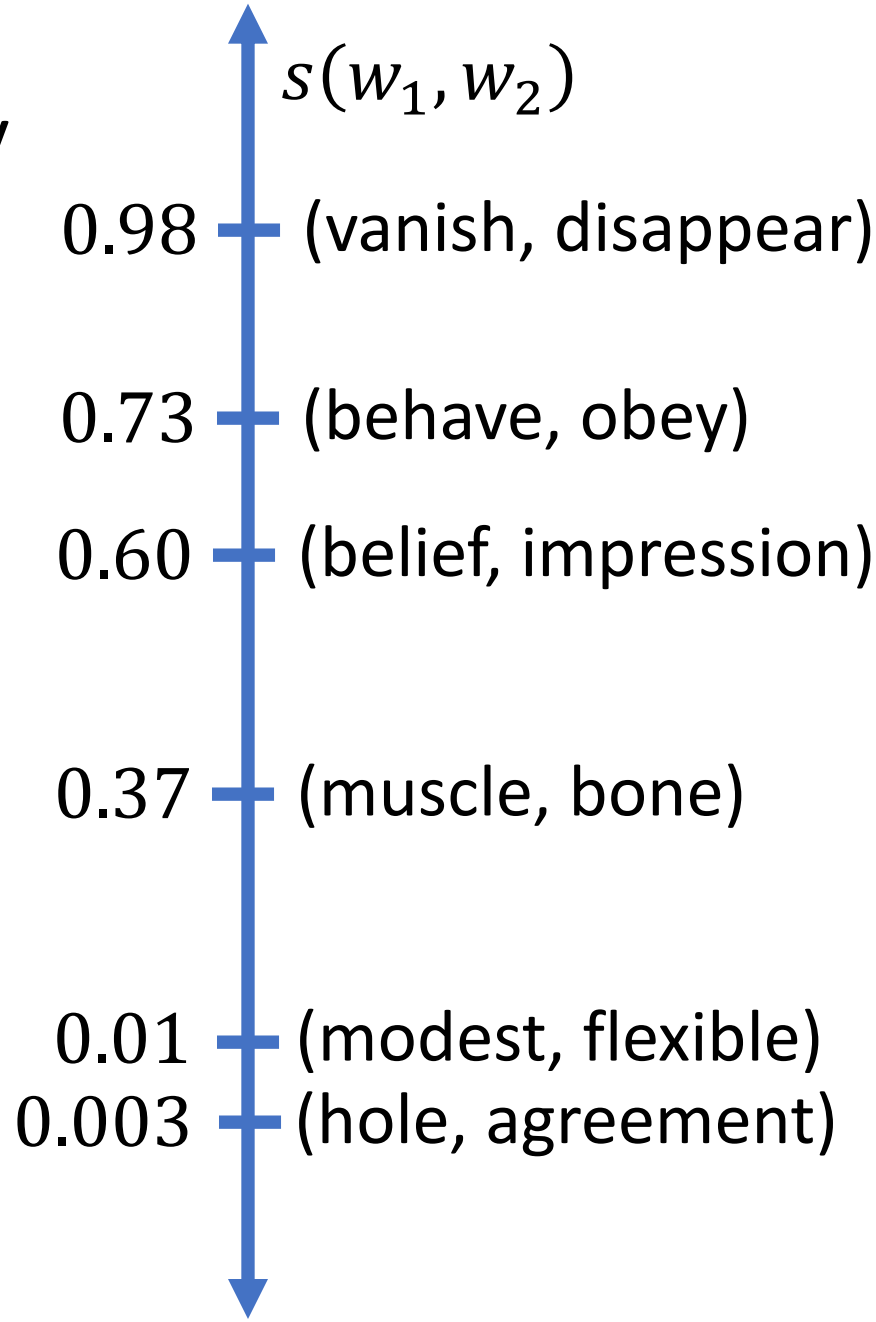
The Bank of the Thames. By Diliff - Own work, CC BY 3.0, https://commons.wikimedia.org/w/index.php?curid=3639626

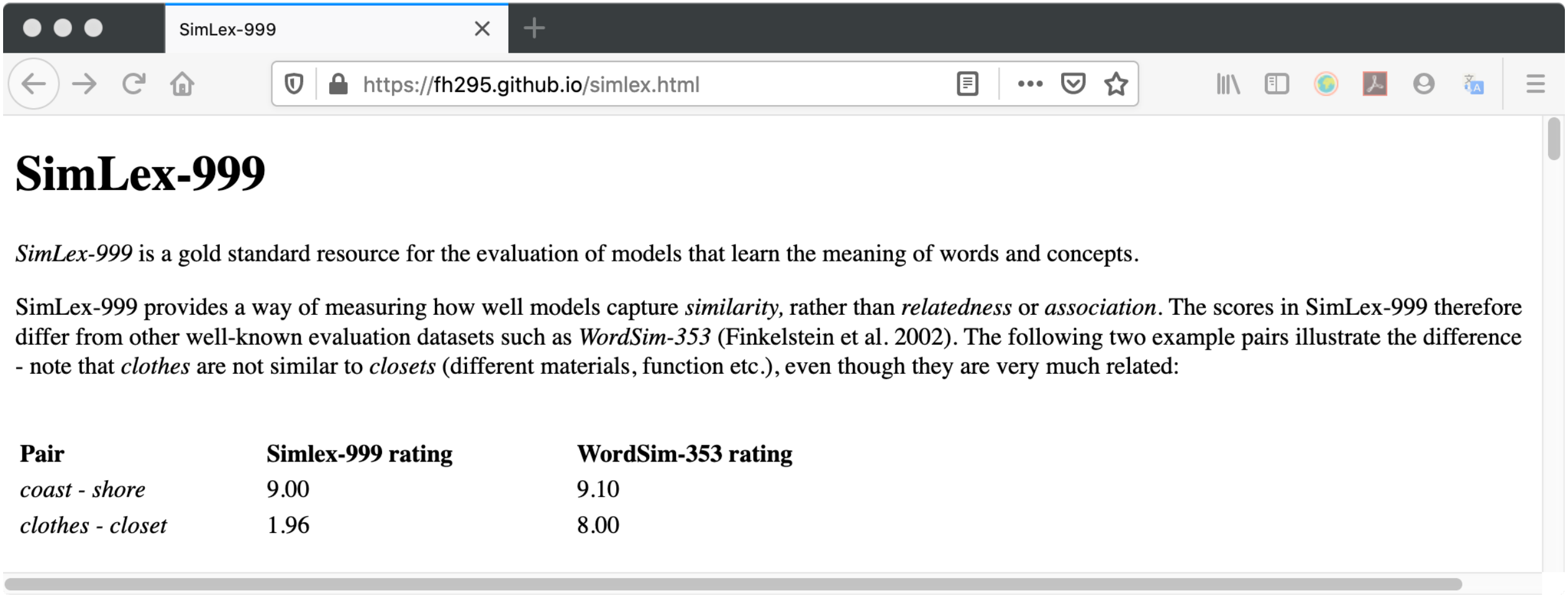
Wordform, lemma, and word sense

- wordform
  - easy for a computer to work with: just look for space-bounded sequences of characters
- lemma
  - This is what humans think of as a word. A cluster of wordforms whose spellings, pronunciations, and meanings can all be derived from one another by applying simple rules.
- word sense
  - A meaning so distinct from the other meanings of the word that it’s hard to consider them the same word.

Synonymy and similarity

- Words are “synonyms” if they have exactly the same meaning.
- No words ever have ***exactly*** the same meaning, so no two words are ever exactly synonyms.
- We prefer to talk about word similarity, $0 \leq s(w_1, w_2) \leq 1$
  - $s(w_1, w_2) = 1$: w_1 and w_2 are perfect synonyms. Never happens in practice, but sometimes close.
  - $s(w_1, w_2) = 0$: w_1 and w_2 are completely different.



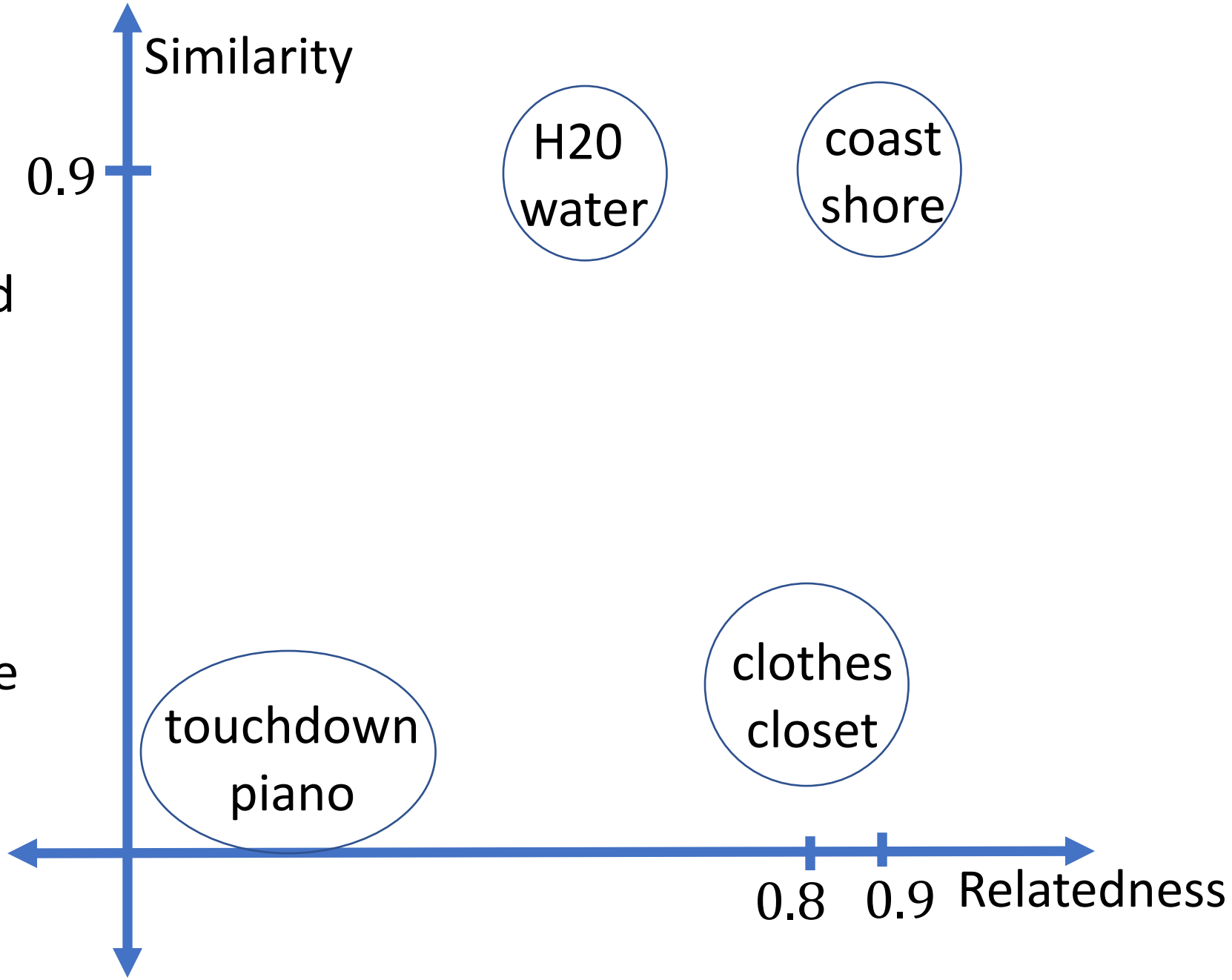


- Algorithms that try to estimate the similarity of two wordforms can be tested on databases such as SimLex-999.
- Humans rated the similarity of each word pair on a 10-point scale.

Similarity vs. Relatedness

Similar: words can be used interchangeably in most contexts

Related: there is some connection between the two words, such that they tend to appear in the same documents.



Outline

- Lexical semantics
  - Wordforms, lemmas & word senses; Synonymy & similarity
  - Vector semantics: CBOW and skip-gram
  - Generative training vs. Contrastive training
  - Implicit meaning representations in lexical semantic spaces
- Sentence semantics
  - Recurrent neural networks (RNN)
  - Attention
  - Self-attention, Multi-headed attention, Cross-attention, and Masked attention
  - Self-training

Review: Naïve Bayes: the "Bag-of-words" model

We can estimate the likelihood of an e-mail by pretending that the e-mail is just a bag of words (order doesn't matter).

With only a few thousand spam e-mails, we can get a pretty good estimate of these things:

- $P(W = \text{“hi”}|Y = \text{spam})$, $P(W = \text{“hi”}|Y = \text{ham})$
- $P(W = \text{“vitality”}|Y = \text{spam})$, $P(W = \text{“vitality”}|Y = \text{ham})$
- $P(W = \text{“production”}|Y = \text{spam})$, $P(W = \text{“production”}|Y = \text{ham})$

Then we can approximate $P(X|Y)$ by assuming that the words, W, are ***conditionally independent of one another given the category label***:

$$P(X = x|Y = y) \approx \prod_{i=1}^{n} P(W = w_i|Y = y)$$

Similarity: The Internet is the database

Similarity = words can be used interchangeably in most contexts

How do we measure that in practice?

Answer: extract examples of word w_1, +/- N words (N=2 or 3):

…hot, although iced **coffee** is a popular…

…indicate that moderate **coffee** consumption is benign…

…and of w_2:

…consumed as iced **tea**. Sweet tea is…

…national average of **tea** consumption in Ireland…

The words “iced” and “consumption” appear in both contexts, so we can conclude that $s(\text{coffea}, \text{tea}) > 0$. No other words are shared, so we can conclude $s(\text{coffee}, \text{tea}) < 1$.

skip-gram context probability

Consider the "…hot although iced **coffee** is a popular…".

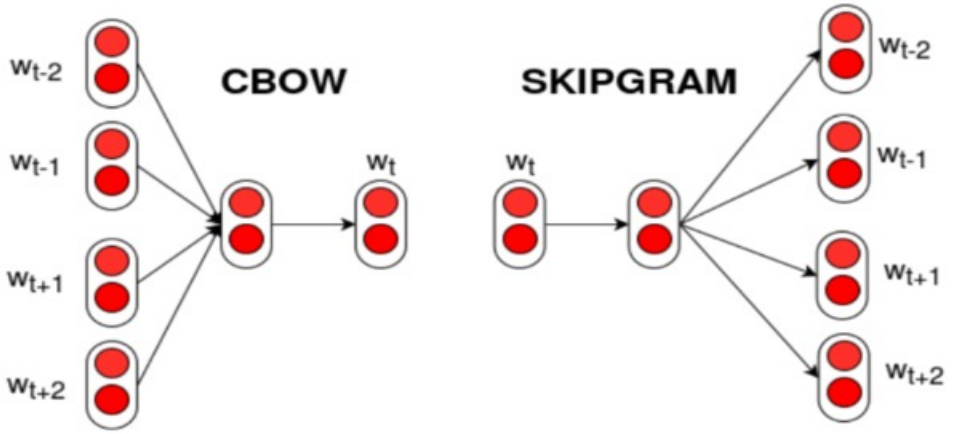
Define the target word to be w_0 =coffee.

Define the context words w_{-3} =hot, w_{-2} =although, …, w_3 =popular.

The skip-gram probability is a naïve Bayes model of the context:

$$p(w_{-3}, \dots, w_3 | w_0) = \prod_{\substack{i \neq 0 \\ i=-3}}^{3} p(w_i | w_0)$$

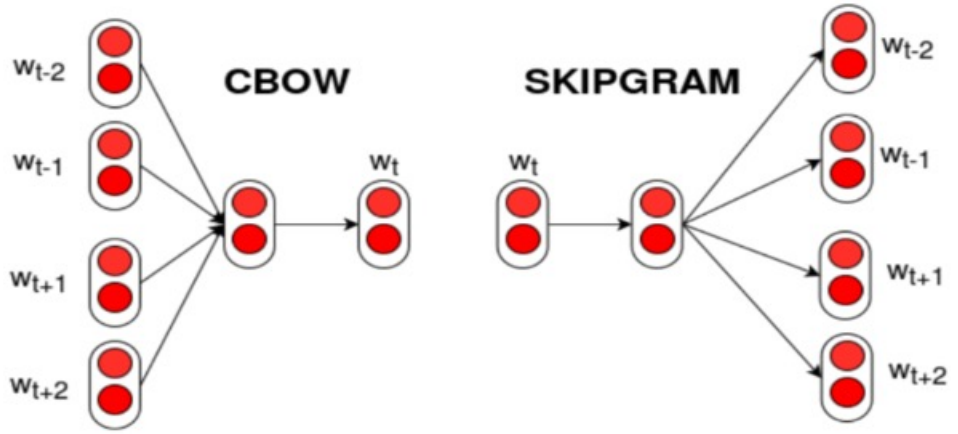
The skip-gram model



- Skip-gram is a model of word meaning:
- The meaning of a word is defined to be the distribution of context words that it can predict.
- We find out which words w_t can predict by learning neural nets that predict its context words w_{t+j}:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c, j\neq 0}^{c} \ln P\left(w_{t+j}|w_t\right)$$

The “continuous bag of words” model (CBOW)



$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} logp(w_t|w_{t+j}) \qquad \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} logp(w_{t+j}|w_t)$$

- CBOW is a similar model of word meaning:
- The meaning of a word is defined to be the distribution of context words that predict it the best.
- We find out which words predict w_t by learning neural nets that predict w_t given its context words, w_{t+j}, for $-c \le j \le c$:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c, j\neq 0}^{c} \ln P\left(w_t|w_{t+j}\right)$$

“Probability,” for a NN, means softmax

- What does it mean that we train a neural net to compute $P(w_t|w_{t+j})$?
- It’s a probability, so it must mean a softmax:

$$P(W_t = m|W_{t+j} = n) = \frac{\exp(e_{m,n})}{\sum_{m'} \exp(e_{m',n})}$$

- But what are the inputs to the neural net? What is $e_{m,n}$?

Vector Semantics

- The simplest useful assumption is this: the meaning of word $W_t = m$ is represented by the vector v_m:

$$P(W_t = m | W_{t+j} = n) = \frac{\exp(v_m^T v_n)}{\sum_k \exp(v_k^T v_n)}$$

- …where v_m is a d-dimensional vector, $v_m^T = [v_{m,1}, \ldots, v_{m,d}]$
- The only trainable parameters in this model are the word vectors!
- The dictionary, v , is a matrix, with as many rows as there are words in the vocabulary:

$$v = \begin{bmatrix} v_{\text{aardvark}}^T \\ \vdots \\ v_{\text{zebra}}^T \end{bmatrix} = \begin{bmatrix} v_{\text{aardvark},1} & \cdots & v_{\text{aardvark},d} \\ \vdots & \ddots & \vdots \\ v_{\text{zebra},1} & \cdots & v_{\text{zebra},d} \end{bmatrix}$$

cosine similarity

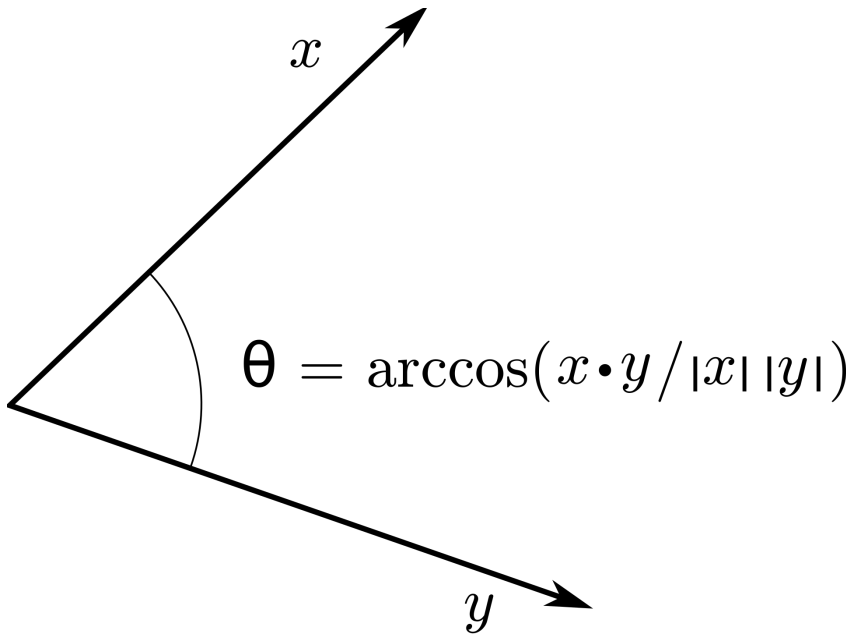
If words w_1 and w_2 are similar, w_1 is represented by vector $\vec{v}_1$, and w_2 by vector $\vec{v}_2$, then the angle between the two vectors should be small.

Angle between two vectors can be measured by their dot product:

$$\cos\theta = \frac{v_1^T v_2}{|v_1||v_2|}$$

where

$$v_1^T v_2 = \sum_{i=1}^{d} v_{1,i} v_{2,i}\,, \qquad |v_1| = \sqrt{\sum_{i=1}^{d} v_{1,i}^2}$$



Vector Semantics

There are many ways to make this model more flexible. For example:

- Every word could have two different vectors: one (v_m) for when it's being predicted, one (c_n) for when it is predicting, thus $e_{m,n} = v_m^T c_n$.
- We could put a delay-weight matrix, W_j, in between the word vectors, thus $e_{m,j,n} = v_m^T W_j v_n$.
- We could even use an MLP to calculate the similarity, for example, $e_{m,n} = W_1 \mathrm{ReLU}\left(W_0 \begin{bmatrix} v_m \\ v_n \end{bmatrix}\right)$.
- …but notice, all these methods are based on the idea of a matrix as a dictionary:

$$v = \begin{bmatrix} v_{\text{aardvark}}^T \\ \vdots \\ v_{\text{zebra}}^T \end{bmatrix} = \begin{bmatrix} v_{\text{aardvark},1} & \cdots & v_{\text{aardvark},d} \\ \vdots & \ddots & \vdots \\ v_{\text{zebra},1} & \cdots & v_{\text{zebra},d} \end{bmatrix}$$

Vector Semantics

The CBOW probability is now:

$$P\left(W_t = m | W_{t+j} = n\right) = \frac{\exp(v_m^T v_n)}{\sum_k \exp(v_k^T v_n)}$$

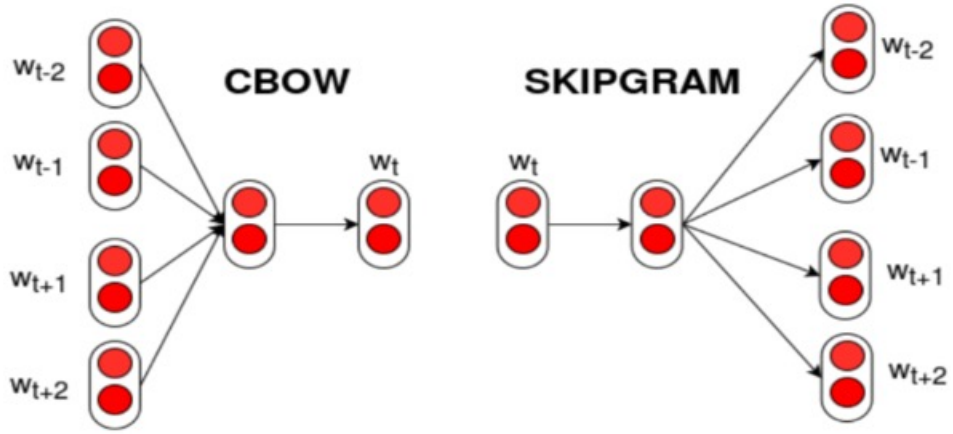
Remember the derivative of a softmax:

$$\frac{\partial \sigma_m(e)}{\partial e_k} = \sigma_m(e)(1_{m=k} - \sigma_k(e))$$

If $\mathcal{L}_t = -\ln \sigma_{W_t}(e)$, then

$$\frac{\partial \mathcal{L}_t}{\partial e_k} = -\left(1_{W_t=k} - \sigma_k(e)\right)$$

Training a CBOW model



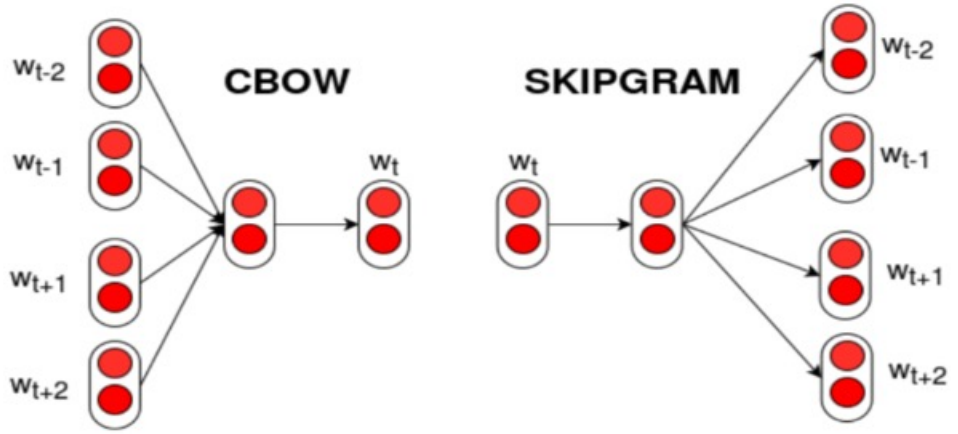
$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} logp(w_t|w_{t+j}) \qquad \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} logp(w_{t+j}|w_t)$$

In order to find the parameters, we use gradient descent:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c, j\neq 0}^{c} \ln P\left(w_t|w_{t+j}\right) = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c, j\neq 0}^{c} \ln \frac{\exp(v_m^T v_n)}{\sum_k \exp(v_k^T v_n)}$$

$$\frac{\partial \mathcal{L}}{\partial v_m} = -\frac{1}{T}\sum_{t:w_t=m}\sum_{j=-c, j\neq 0}^{c}\left(1 - P\left(W_t = m|w_{t+j}\right)\right)\times v_{W_{t+j}}$$

Training a CBOW model



$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} logp(w_t|w_{t+j}) \qquad \frac{1}{T}\sum_{t=1}^{T}\sum_{-c\le j\le c, j\neq 0} logp(w_{t+j}|w_t)$$

The CBOW model is trained by setting every vector equal to a weighted average of the words that occurred near it!

$$v_m \leftarrow v_m - \eta \frac{\partial \mathcal{L}}{\partial v_m}$$

$$v_m \leftarrow v_m + \frac{\eta}{T} \sum_{t:w_t=m} \sum_{j=-c, j\neq 0}^{c} \left(1 - P\left(W_t = m | w_{t+j}\right)\right) v_{w_{t+j}}$$

Outline

- Lexical semantics
  - Wordforms, lemmas & word senses; Synonymy & similarity
  - Vector semantics: CBOW and skip-gram
  - Generative training vs. Contrastive training
  - Implicit meaning representations in lexical semantic spaces
- Sentence semantics
  - Recurrent neural networks (RNN)
  - Attention
  - Self-attention, Multi-headed attention, Cross-attention, and Masked attention
  - Self-training

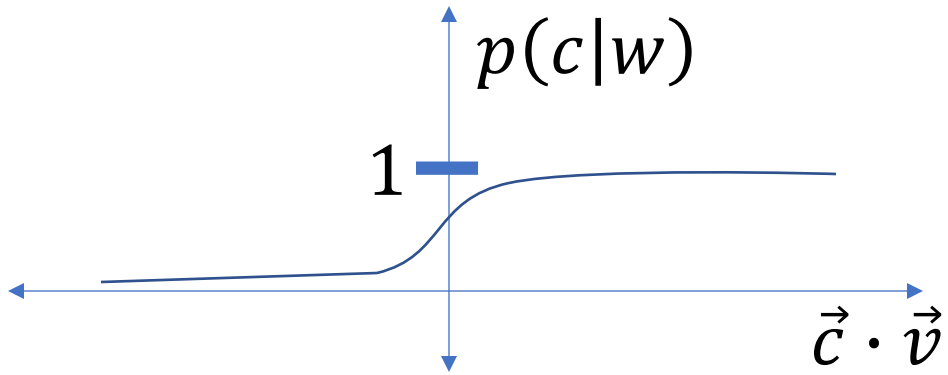
skip-gram: context probability

Now suppose we want to embed w =coffee with a vector v.

…and we want to embed c_3 =hot with a vector c.

Define the probability that “hot” occurs within +/-N words of “coffee” to be just a sigmoid:

$$p(c|w) = \frac{1}{1 + e^{-c^T v}}$$



Contrastive loss vs. Generative loss

- A generative loss is one like this:

$$\mathcal{L} = -\frac{1}{T}\sum_{t=0}^{T-1}\sum_{j=-c, j\neq 0}^{c} \ln \frac{\exp(v_m^T v_n)}{\sum_k \exp(v_k^T v_n)}$$

- Notice that this loss term compares each word, w_t, to every other word in the dictionary.
- Sometimes, generative training can take a very long time to converge.
- Sometimes, we get faster training using contrastive loss.

Contrastive loss

We train the neural network by listing, as positive examples, the words that occur in the context of "w =coffee," e.g.,

$$\mathcal{D}_+(w) = \{\text{hot}, \text{although}, \text{iced}, \text{moderate}, \text{hot}, \text{consumption}\}$$

Create a contrastive database by choosing the same number of words, at random, from among the words that never appeared in the context of "coffee:"

$$\mathcal{D}_-(w) = \{\text{aardvark}, \text{dog}, \text{gazebo}, \text{actor}, \text{precipitates}, \text{iceberg}\}$$

Training with contrastive loss

The coefficients $\vec{v}_i = [v_{i,1}, \ldots, v_{i,d}]$ for each vector are then learned in order to maximize the log probability that:

- Words in $\mathcal{D}_+(w)$ should occur in the context of w,
- Words in $\mathcal{D}_-(w)$ should NOT occur in the context of w:

$$\mathcal{L} = -\ln p(\text{Data}) = -\sum_{w \in \mathcal{V}} \sum_{c \in \mathcal{D}_+(w)} \ln p(c|w) - \sum_{w \in \mathcal{V}} \sum_{c \in \mathcal{D}_-(w)} \ln\big(1 - p(c|w)\big)$$

- … where $p(c|w)$ is the probability that word c occurs in the context of word w.

Training with contrastive loss

Suppose we define $p(c|w) = \frac{1}{1+e^{-c^T v}}$.

Then $1 - p(c|w) = 1 - \frac{1}{1+e^{-c^T v}} = \frac{1}{1+e^{c^T v}}$.

Then $\ln p(c|w) = -\ln\left(1 + e^{-c^T v}\right)$

...and $\ln\left(1 - p(c|w)\right) = -\ln\left(1 + e^{c^T v}\right)$.

Training with contrastive loss

The loss function then gets a form which is a differentiable convex function of the data vectors:

$$\mathcal{L} = -\sum_{w \in \mathcal{V}} \sum_{c \in \mathcal{D}_+(w)} \ln p(c|w) - \sum_{w \in \mathcal{V}} \sum_{c \in \mathcal{D}_-(w)} \ln\left(1 - p(c|w)\right)$$

$$= \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{D}_+(w)} \ln\left(1 + e^{-c^T v}\right) + \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{D}_-(w)} \ln\left(1 + e^{c^T v}\right)$$

Outline

- Lexical semantics
  - Wordforms, lemmas & word senses; Synonymy & similarity
  - Vector semantics: CBOW and skip-gram
  - Generative training vs. Contrastive training
  - Implicit meaning representations in lexical semantic spaces
- Sentence semantics
  - Recurrent neural networks (RNN)
  - Attention
  - Self-attention, Multi-headed attention, Cross-attention, and Masked attention
  - Self-training

Visualizations: Similarity

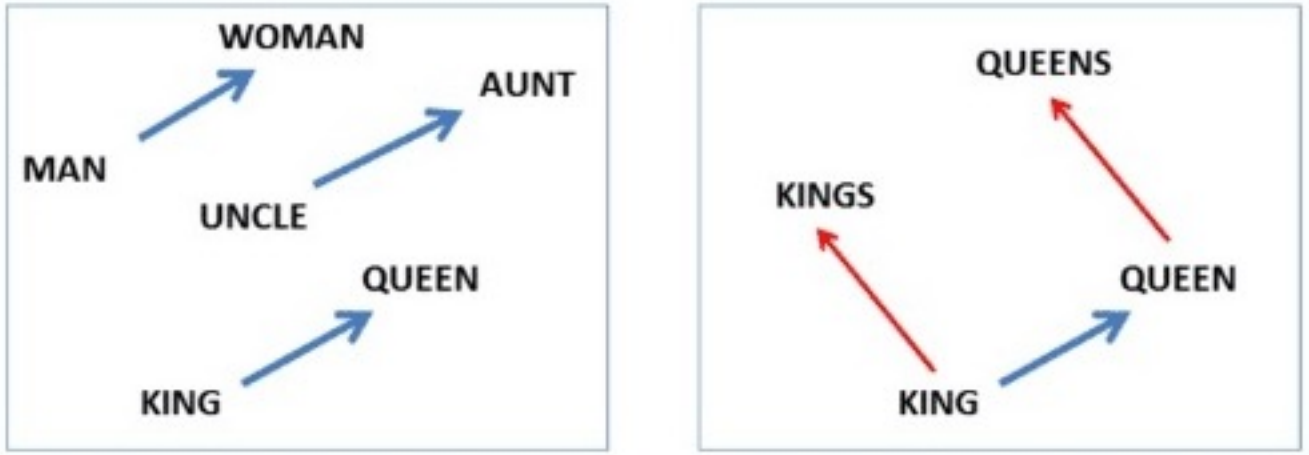
Mikolov et al. (2013) tested word2vec on SimLex-999, and had better results than previously published baselines. Here are some examples from their paper. Notice that not all of their ”similar words” are really similar – some are just related. I’ll talk more about that next time.

Model (training time)	Redmond	Havel	ninjutsu	graffiti	capitulate
Collobert (50d) (2 months)	conyers lubbock keene	plauen dzerzhinsky osterreich	reiki kohona karate	cheesecake gossip dioramas	abdicate accede rearm
Turian (200d) (few weeks)	McCarthy Alston Cousins	Jewell Arzu Ovitz	- - -	gunfire emotion impunity	- - -
Mnih (100d) (7 days)	Podhurst Harlang Agarwal	Pontiff Pinochet Rodionov	- - -	anaesthetics monkeys Jews	Mavericks planning hesitated
Skip-Phrase (1000d, 1 day)	Redmond Wash. Redmond Washington Microsoft	Vaclav Havel president Vaclav Havel Velvet Revolution	ninja martial arts swordsmanship	spray paint grafitti taggers	capitulation capitulated capitulating

Table 6: Examples of the closest tokens given various well known models and the Skip-gram model trained on phrases using over 30 billion training words. An empty cell means that the word was not in the vocabulary.

Visualizations: Relatedness

vec(“woman”) - vec(“man”) + vec(“king”) = vec(“queen”)



Christian S. Perone, "Voynich Manuscript: word vectors and t-SNE visualization of some patterns," in *Terra Incognita*, 16/01/2016, http://blog.christianperone.com/2016/01/voynich-manuscript-word-vectors-and-t-sne-visualization-of-some-patterns/.

Mikolov (2013) showed that word2vec captures similarity relationships among words. For example, the difference between the vectors for “woman” and “man” is roughly the same as the difference between the vectors for “queen” and “king.” Perone (2016) showed that this effect works differently depending on the training corpus: in his blog post, he looks at word relatedness in the 15th century Voynich manuscript.

Learning biased analogies from data

- It's useful that algorithms like word2vec learn appropriate analogies, like "Paris → France as Tokyo → Japan" and "kings → king as queens → queen."
- Unfortunately, it also learns other analogies that were implied in the training corpus, but that are invalid analogies.
- The paper that first demonstrated that problem was named after one of the worst such discovered analogies:

"Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings," Bolukbasi et al., 2016

Biased analogies

Bolukbasi et al. defined a “male-female” continuum by subtracting vec(female)-vec(male), vec(woman)-vec(man), and so on, then averaging these difference vectors.

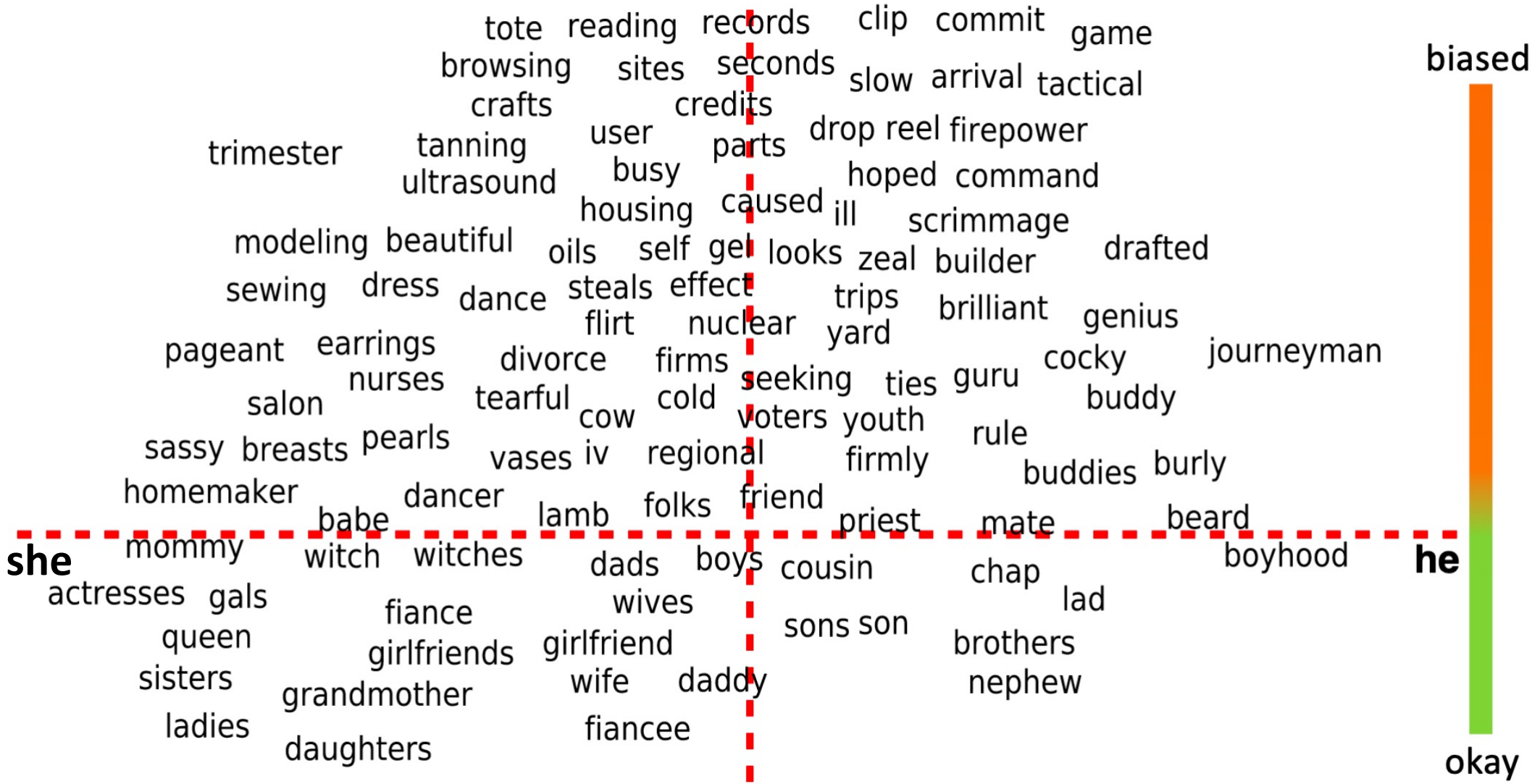
They then took all of the words whose dictionary definitions included gender-specific language (man, woman), and considered those to be the gender-specific words (words for which a gender difference is appropriate).

All other words were considered gender-neutral (any difference on the male-female dimension is inappropriate).

The result is a second dimension: the appropriateness of a gender bias.

The Male-Female vs. Neutral-Specific Space

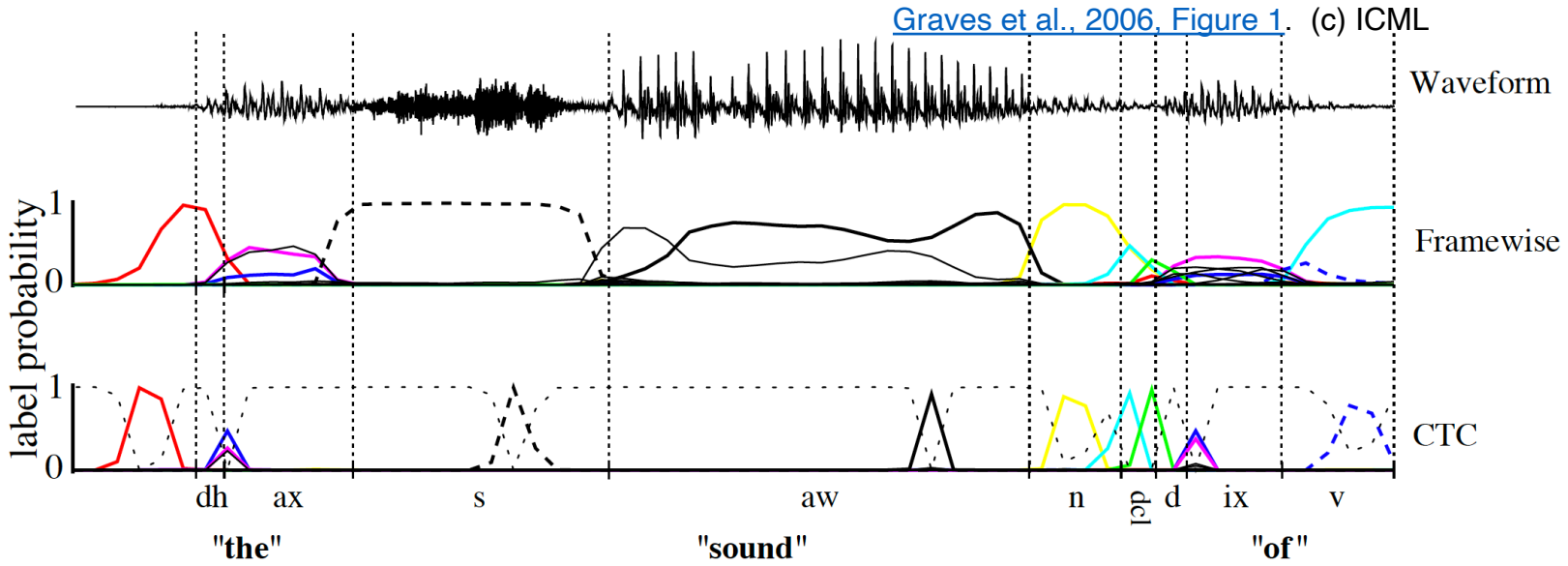
Here's the resulting 2D space, from Bolukbasi et al., 2016:

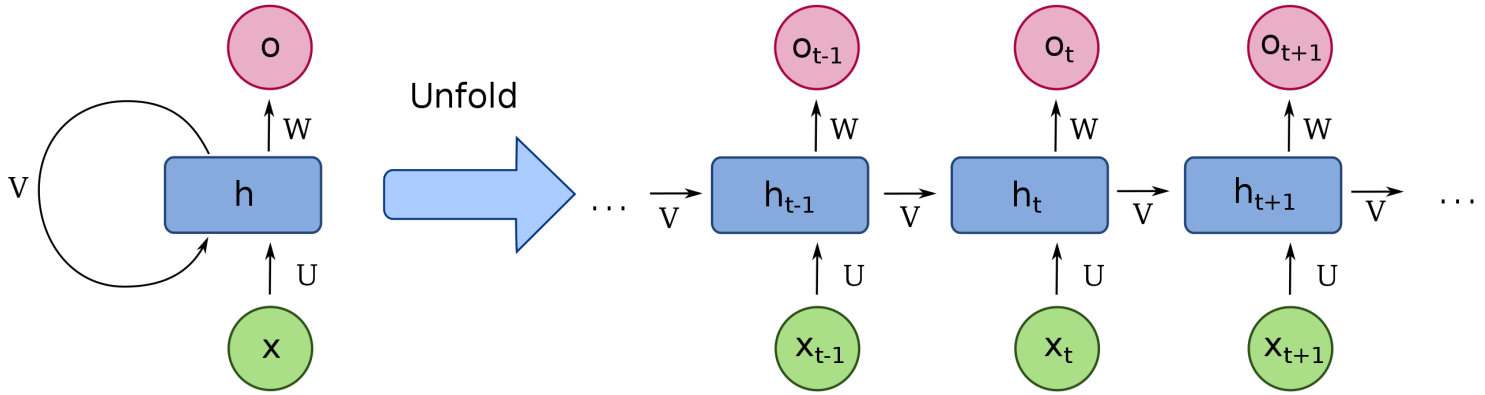


Outline

- Lexical semantics
  - Wordforms, lemmas & word senses; Synonymy & similarity
  - Vector semantics: CBOW and skip-gram
  - Generative training vs. Contrastive training
  - Implicit meaning representations in lexical semantic spaces
- Sentence semantics
  - Recurrent neural networks (RNN)
  - Attention
  - Self-attention, Multi-headed attention, Cross-attention, and Masked attention
  - Self-training

Recurrent neural network

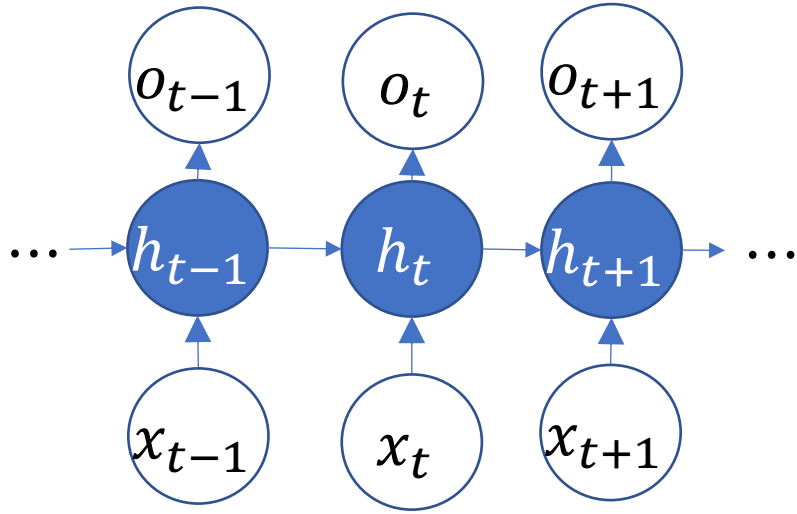




By fdeloche - Own work, CC BY-SA 4.0, https://commons.wikimedia.org/w/index.php?curid=60109157

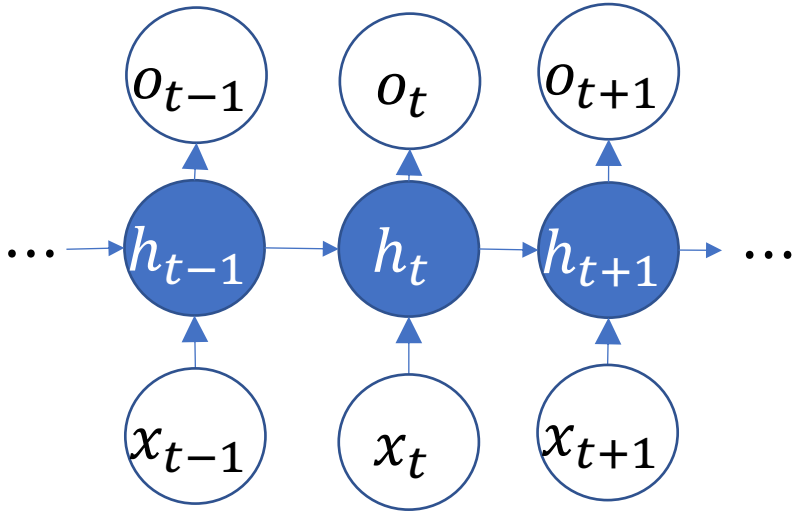
- In a recurrent neural network (RNN), the hidden node activation vector, h_t, depends on the value of the same vector at time $t-1$.
- From 2014-2017, the best speech recognition and machine translation used RNNs.
- The input is x_t=speech or input-language text
- The output is o_t=text in the target language

Example: Part of speech tagging



- x_t =vector representation of the t[th] word, e.g., trained using CBOW
- h_t =hidden state vector
- $o_t = \text{softmax}(h_t^T x_t) = [P(Y_t = \text{Noun}|X_1, \dots, X_t), P(Y_t = \text{Verb}|X_1, \dots, X_t), \dots]$

Training an RNN



An RNN is trained using gradient descent, just like any other neural network!

$$u_{j,i} \leftarrow u_{j,i} - \eta \frac{\partial \mathfrak{L}}{\partial u_{j,i}}$$

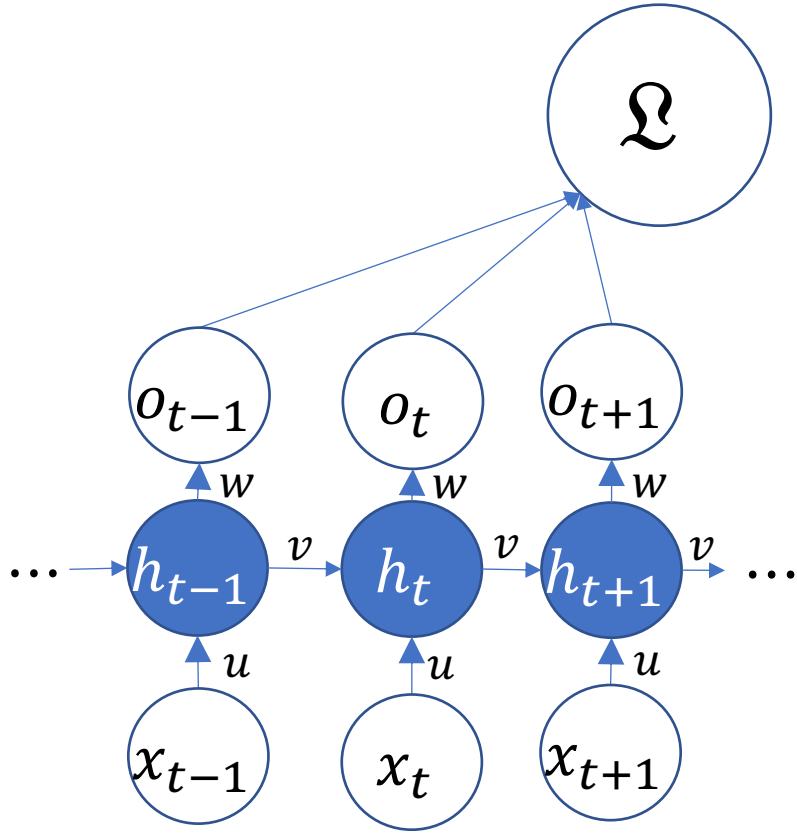
$$w_{j,k} \leftarrow w_{j,k} - \eta \frac{\partial \mathfrak{L}}{\partial w_{j,k}}$$

…where $\mathfrak{L}$ is the loss function, and η is a step size.

Training an RNN: Infinite recursion?

The big difference is that now the loss function depends on u, v and w in many different ways:

- The loss function depends on each of the state vectors h_t, which depends directly on u and v.
- But h_t also depends on h_{t-1}, which, in turn, depends on u and v.
- … and so on.

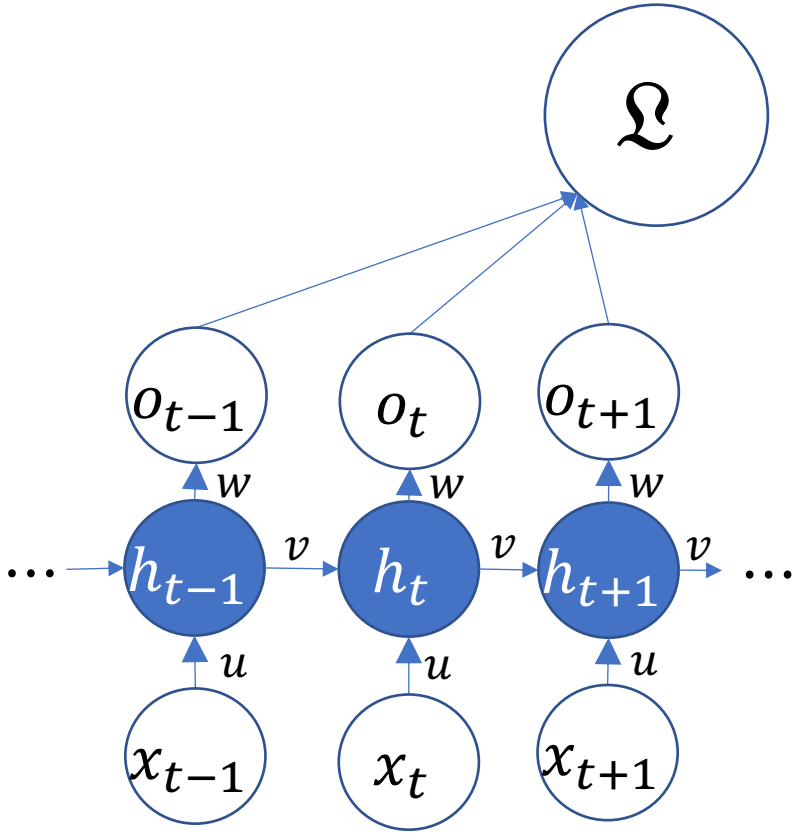


Back-propagation through time

The solution is something called back-propagation through time:

$$\frac{d\mathfrak{L}}{dh_{i,t}} = \frac{\partial \mathfrak{L}}{\partial h_{i,t}} + \sum_j \frac{d\mathfrak{L}}{dh_{j,t+1}} \frac{\partial h_{j,t+1}}{\partial h_{i,t}}$$

- The first term measures losses caused directly by $h_{i,t}$, for example, if $o_{i,t}$ is wrong.
- The second term measures losses caused indirectly, for example, because $h_{i,t}$ caused $h_{j,t+1}$ to be wrong.

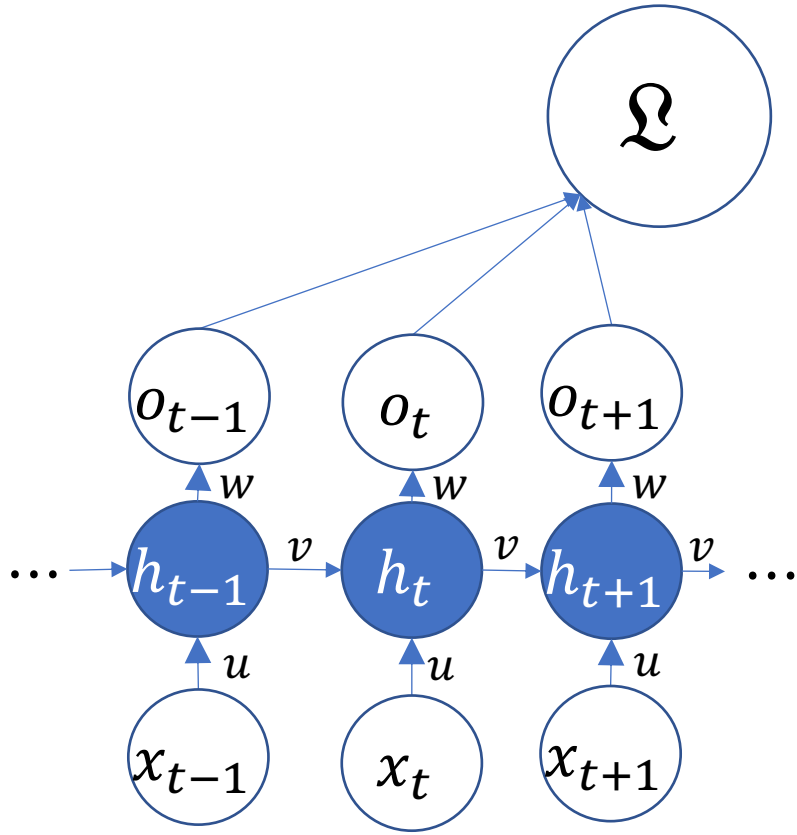


Back-propagation through time

Notice that this is just like training a very deep network!

- Back-propagation through time: back-propagate from time step $t + 1$ to time step t
- Back-propagation in a very deep network: back-propagate from layer $l + 1$ to layer l

Toolkits like PyTorch may use the same code in both cases.



Outline

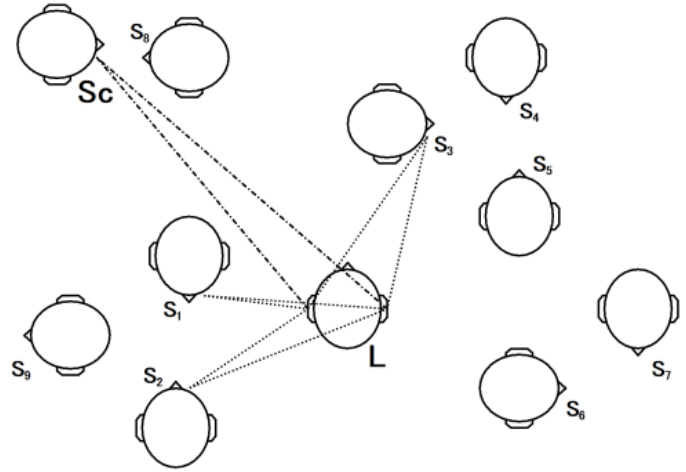
- Lexical semantics
  - Wordforms, lemmas & word senses; Synonymy & similarity
  - Vector semantics: CBOW and skip-gram
  - Generative training vs. Contrastive training
  - Implicit meaning representations in lexical semantic spaces
- Sentence semantics
  - Recurrent neural networks (RNN)
  - Attention
  - Self-attention, Multi-headed attention, Cross-attention, and Masked attention
  - Self-training

The Cocktail-Party Effect

- If you are focusing on one person's voice, but hear your name spoken by another person, your attention immediately shifts to the second voice.
- This "cocktail-party effect" suggests a model of hearing in which all sounds are processed preconsciously. Trigger sounds in an unattended source will cause attention to re-orient to that source.

https://commons.wikimedia.org/wiki/File:Cocktail_party_attendees_at_Fuller_Lodge,_1946.jpg

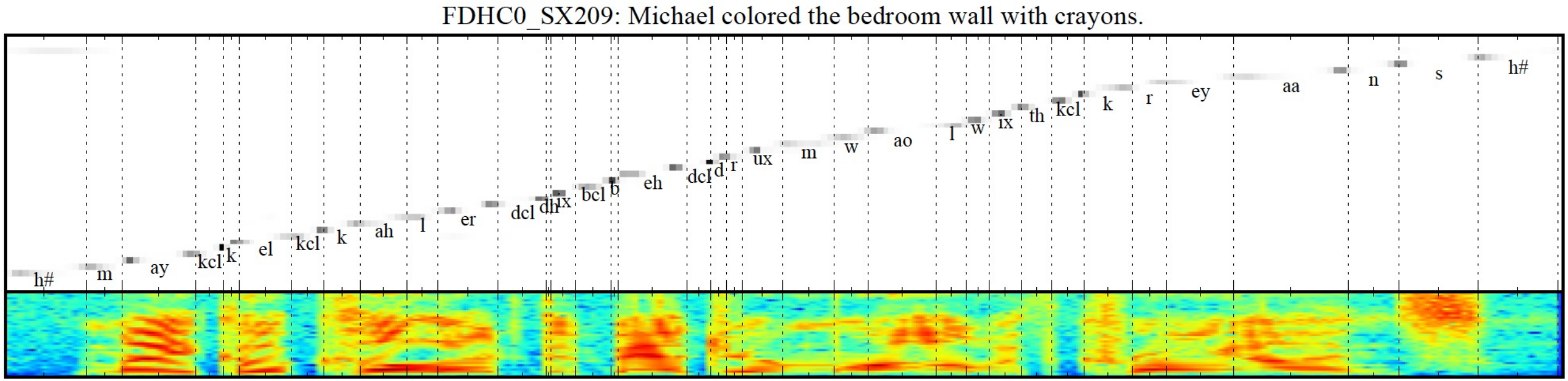
https://commons.wikimedia.org/wiki/File:Cocktail_Party_At_The_Imperial_Hotel_March_13,_1961_(Tokyo,_Japan)_(4966610682).jpg



https://commons.wikimedia.org/wiki/File:Cocktail-party_effect.svg

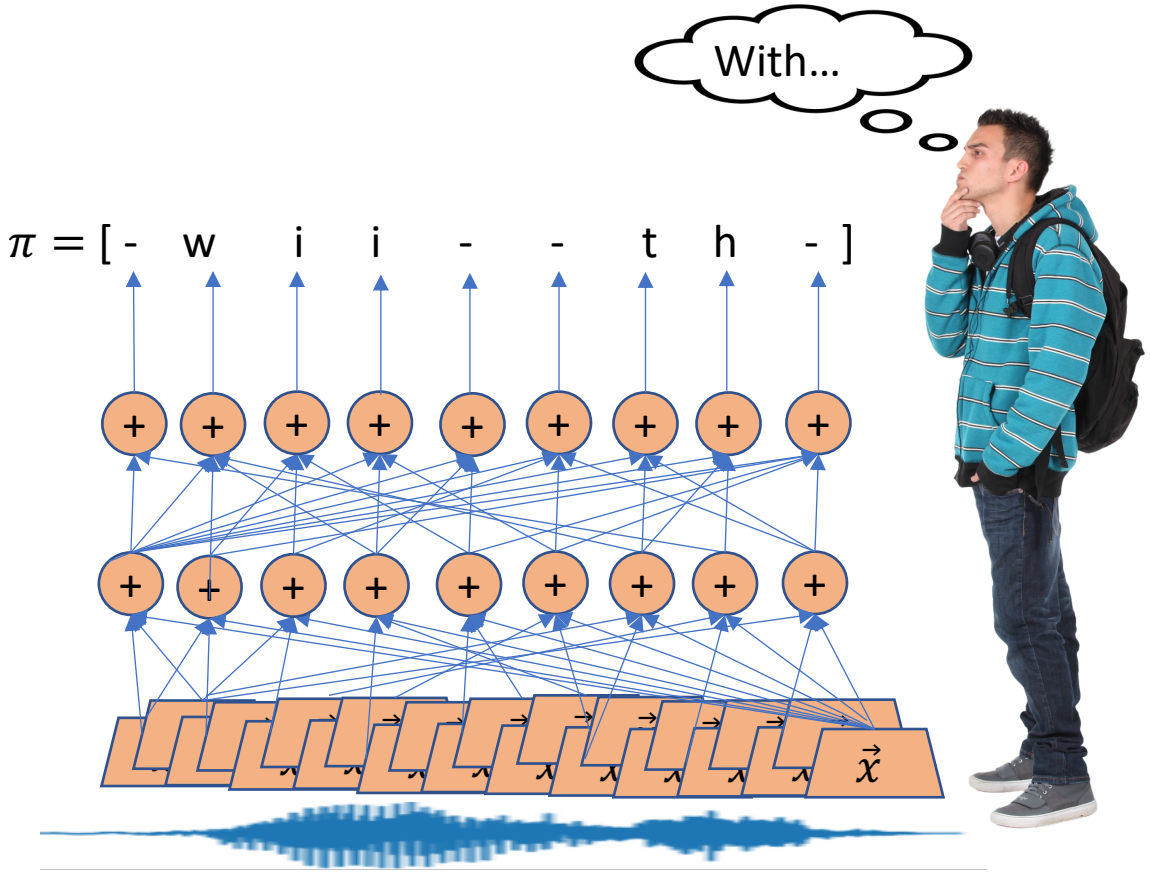
Bottom-up attention as a strategy for machine listening

- In 2014, researchers proposed that the past 200ms of RNN state vectors should be stored in a “short-term memory buffer”
- A speech recognizer can attend to several centiseconds, all at one time, to decide what words it thinks it is hearing



Chorowski, Bahdanau, Serdyk, Cho & Bengio, Attention-Based Models for Speech Recognition, Fig. 1

The Transformer: “Attention is all you need”



- In 2017, researchers proposed that the short-term memory buffer should contain raw signals, not processed signals.
- All processing is done using a model of bottom-up attention.

Attention: Key concepts

- The neural net needs to make a series of decisions, o_i
- Each decision needs to be based on some context, c_i
- Each context vector is a weighted sum of input values, $c_i = \sum_t \alpha_{i,t} v_t$
- $\alpha_{i,t}$ is the amount of attention that the output decision o_i is paying to the input value v_t. It is based on the similarity between a key vector, k_t, that describes the type of information available in v_t, and a query vector, q_i, that describes the type of information necessary in order to make the output decision

Inputs to an attention network

- Neural net inputs: a sequence of row vectors, x_t
- Neural net outputs: a sequence of row vectors, o_i
- Value: What type of information should x_t provide to the output? This may be just a linear transform of x_t, e.g.: $v_t = W_V x_t$
- Query: What type of information does o_i need? This may be just a linear transform of o_{i-1}, e.g.: $q_i = W_Q o_{i-1}$
- Key: The dot product $q_i @ k_t$ should be positive if v_t is useful, and negative if v_t is useless. This may be $k_t = W_K x_t$

Attention = a probability mass over time

- Attention is like probability: You only have a fixed amount of attention, so you need to decide how to distribute it.
- $\alpha_{i,t} = P(v_t|q_i)$ = the probability that v_t is the context that you need in order to make a decision related to the query vector q_i.

$$\sum_t \alpha_{i,t} = 1$$

- Each output context vector (c_i) is based on some input value vectors (h_t). But which ones? Answer: decide which inputs to pay attention to, then pay attention.

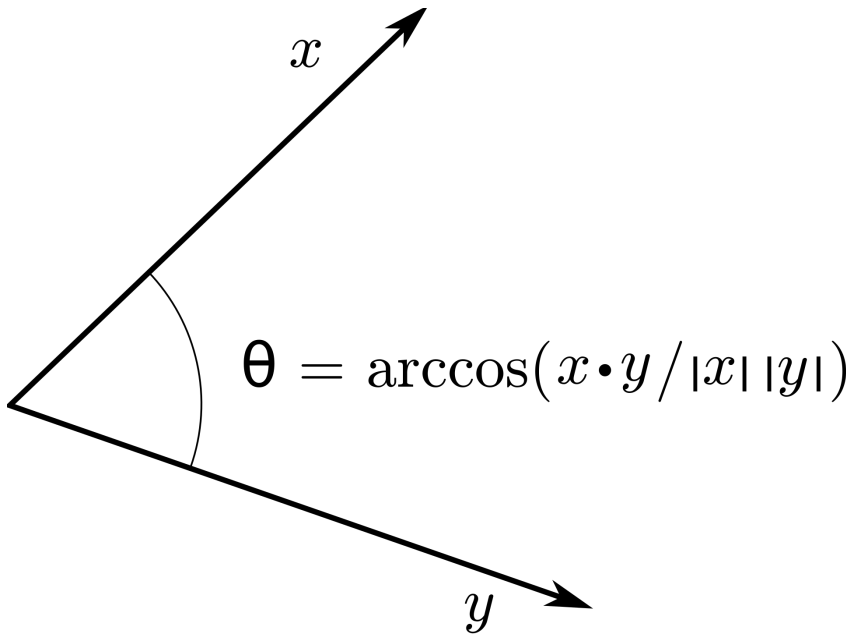
$$c_i = \sum_t \alpha_{i,t} v_t$$

Dot-product attention

How can you decide which value vectors, v_t are most relevant to a particular query? Answer:

1. Create a key vector, k_t, such that $q_i^T k_t > 0$ if v_t is relevant to q_i, otherwise $q_i^T k_t < 0$.
2. Convert the similarity measures into a probability distribution using softmax:

$$\alpha_{i,t} = \frac{\exp\left(q_i^T k_t\right)}{\sum_s \exp\left(q_i^T k_s\right)}$$



Putting it all together

- Stack up v_t, k_t, and q_i into matrices:

$$V = \begin{bmatrix} v_1^T \\ \vdots \\ v_n^T \end{bmatrix}, K = \begin{bmatrix} k_1^T \\ \vdots \\ k_n^T \end{bmatrix}, Q = \begin{bmatrix} q_1^T \\ \vdots \\ q_m^T \end{bmatrix}$$

- $\alpha_t^T = [\alpha_{i,1}, \ldots, \alpha_{i,T}]$ is the softmax whose input vector is Kq_i:

$$\alpha_{i,t} = \sigma_t(k_t^T q_i) = \frac{\exp(k_t^T q_i)}{\sum_s \exp(k_s^T q_i)}$$

- c_i is the product of the vector $\mathrm{softmax}(q_i @ k^T)$ times the v matrix:

$$c_i^T = \alpha_t^T V = \sum_{t=1}^{T} \alpha_{i,t} v_t^T$$

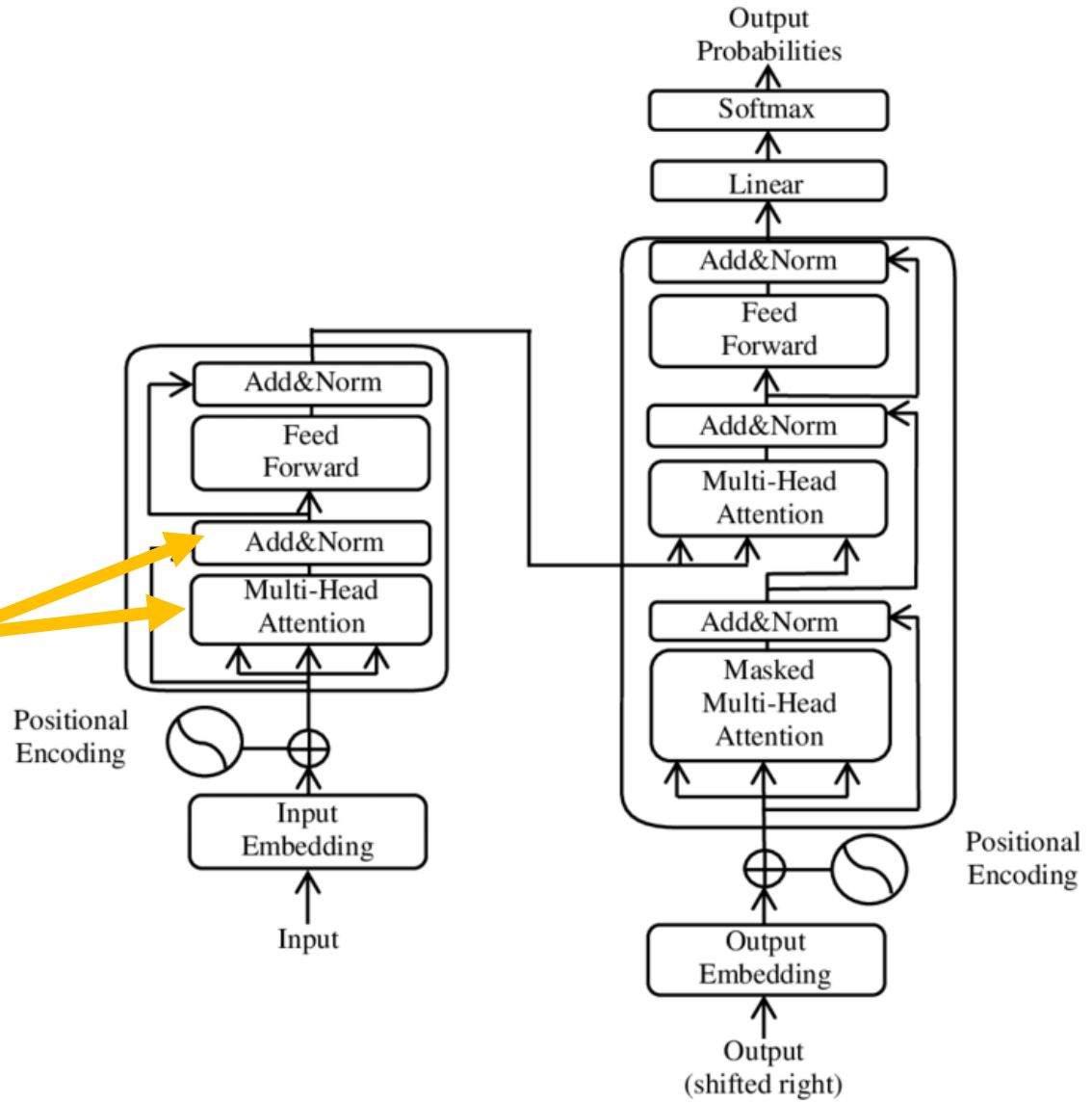
Outline

- Lexical semantics
  - Wordforms, lemmas & word senses; Synonymy & similarity
  - Vector semantics: CBOW and skip-gram
  - Generative training vs. Contrastive training
  - Implicit meaning representations in lexical semantic spaces
- Sentence semantics
  - Recurrent neural networks (RNN)
  - Attention
  - Self-attention, Multi-headed attention, Cross-attention, and Masked attention
  - Self-training

Self-attention

Self-attention (literally!) adds context to each input vector:

$$q_i = W_Q x_t$$
$$k_t = W_K x_t$$
$$v_t = W_V x_t$$
$$c_i = \sigma(K q_i)^T V$$
$$y_i = x_i + c_i$$



By Yuening Jia - DOI:10.1088/1742-6596/1314/1/012186, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=121340680

Multi-headed-attention

Multi-headed-attention uses 8 different W_Q, w_K, and w_V matrices, in order to get 8 different views of the input data:

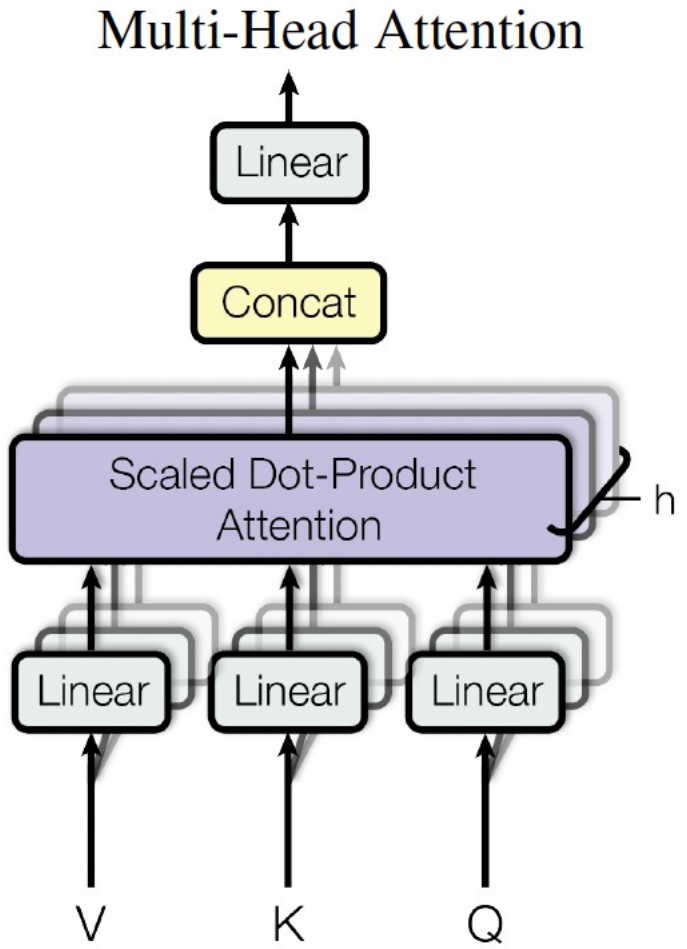
$$q_{j,i} = W_{Q,j} x_i, \qquad 1 \le j \le 8$$

$$k_{j,t} = W_{K,j} x_t, \qquad 1 \le j \le 8$$

$$v_{j,t} = W_{V,j} x_t, \qquad 1 \le j \le 8$$

$$h_{j,i} = \sigma\left(K_j q_{j,i}\right)^T V$$

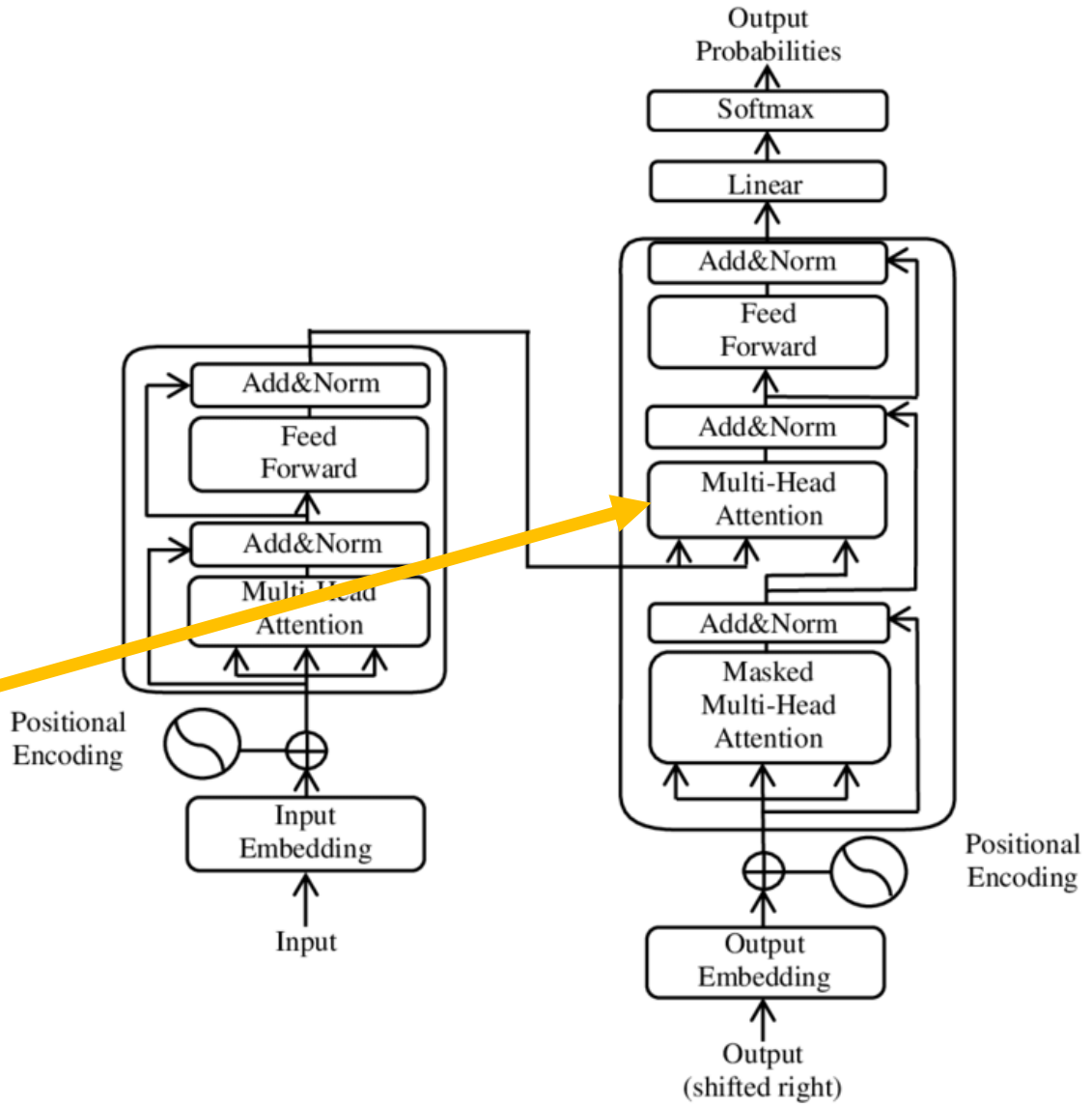
$$c_i = W_O \begin{bmatrix} h_{1,i} \\ \vdots \\ h_{8,i} \end{bmatrix}$$



Cross-attention

Cross-attention: query depends on preceding output, key and value depend on input:

$$q_i = W_Q o_{i-1}$$
$$k_t = W_K x_t$$
$$v_t = W_V x_t$$
$$c_i = \sigma(K q_i)^T V$$



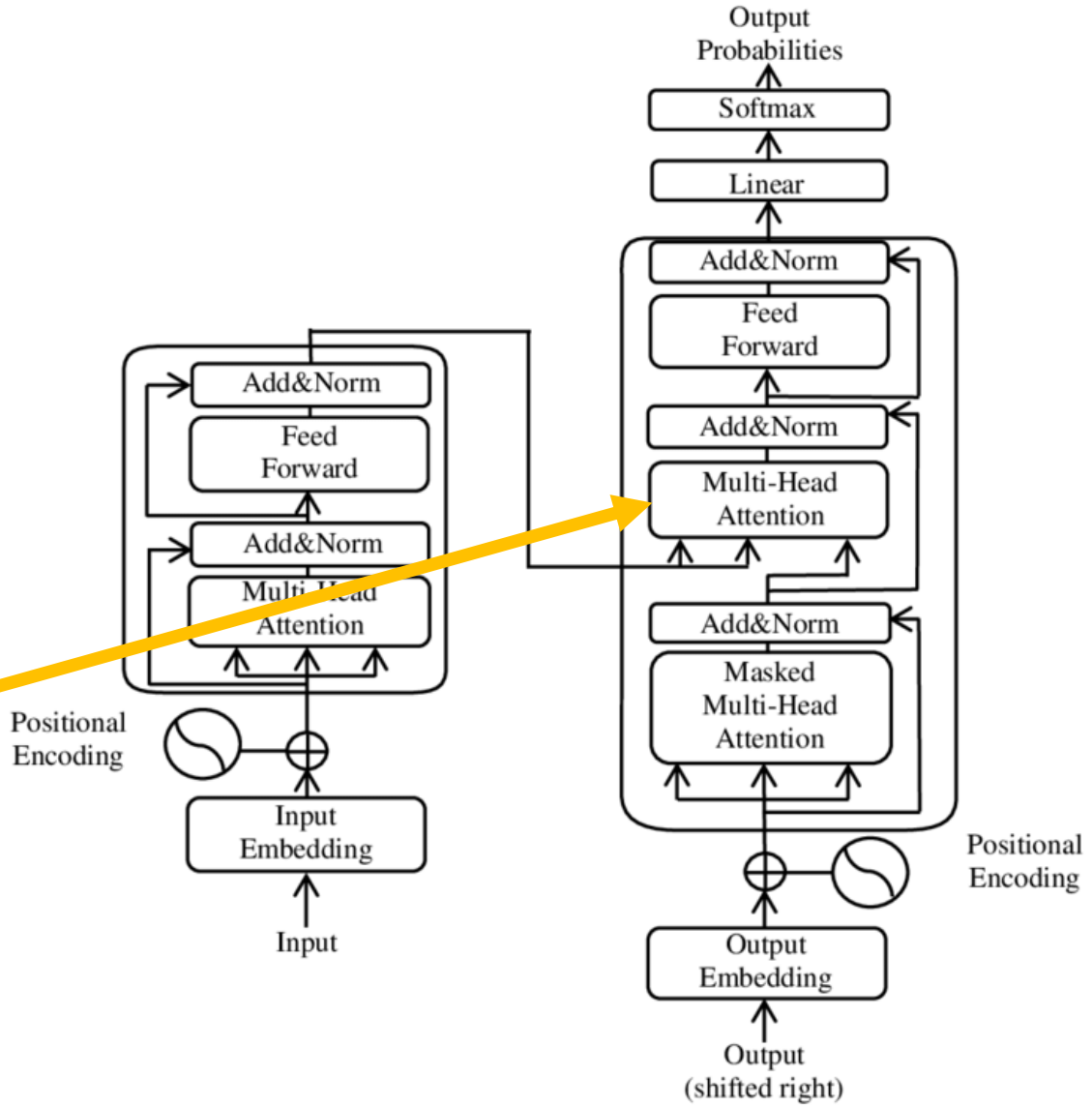
Masked attention

Masked attention forces c_i to pay attention to value vectors v_t only if $t < i$:

$$s(q_i, k_t) = \begin{cases} k_t^T q_i & t < i \\ -\infty & t \geq i \end{cases}$$

$$\alpha_{i,t} = \frac{\exp\big(s(q_i, k_t)\big)}{\sum_\tau \exp\big(s(q_i, k_\tau)\big)}$$

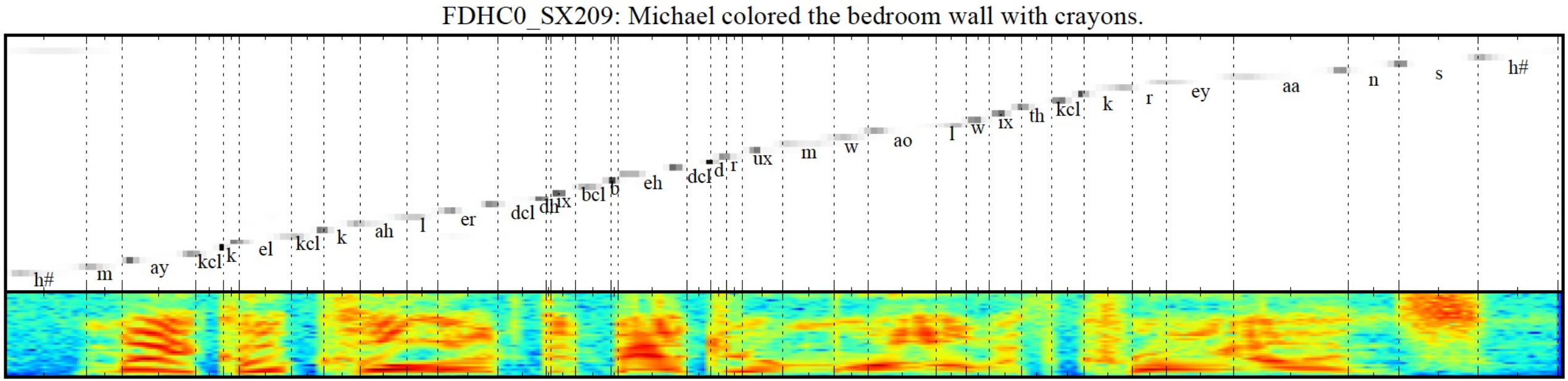
$$= \begin{cases} \sigma_t (K q_i)^T & t < i \\ 0 & t \geq i \end{cases}$$



By Yuening Jia - DOI:10.1088/1742-6596/1314/1/012186, CC BY-SA 3.0, https://commons.wikimedia.org/w/index.php?curid=121340680

Cross-attention visualization

This plot shows $\alpha_{i,t}$ where i = output character, and t = input spectrum



Chorowski, Bahdanau, Serdyk, Cho & Bengio, Attention-Based Models for Speech Recognition, Fig. 1

Word Error Rates using Transformers

By 9/2020, transformers had error rates of:

- 2%: English, quiet recording conditions
- 4%: Chinese or Japanese, quiet recording conditions
- 5-7%: if the reference transcript has errors
- 14%: 2-talker mixtures, synthetic reverberation
- 38%: actual in-home recordings in noisy households

Table 1. CER/WER results on various open source ASR corpora. Both Transformer and Conformer models are implemented based on ESPnet toolkit. ∗ marks ESPnet2 results. † and ‡ indicate only w/ speed or only w/ SpecAugment, respectively. § denotes w/o any data augmentation.

Dataset	Vocab	Metric	Evaluation Sets	Transformer	Conformer
AIDATATANG	Char	CER	dev / test	(†) 5.9 / 6.7	**4.3 / 5.0**
AISHELL-1	Char	CER	dev / test	(†) 6.0 / 6.7	(*) **4.4 / 4.7**
AISHELL-2	Char	CER	android / ios / mic	(†) 8.9 / 7.5 / 8.6	**7.6 / 6.8 / 7.4**
AURORA4	Char	WER	dev_0330 (A / B / C / D)	**3.3 / 6.0 / 4.5** / 10.6	4.3 / 6.0 / 5.4 / **9.3**
CSJ	Char	CER	eval{1, 2, 3}	(∗) 4.7 / 3.7 / 3.9	(∗) **4.5 / 3.3 / 3.6**
CHiME4	Char	WER	{dt05, et05}_{simu, real}	(†) 9.6 / 8.2 / 15.7 / 14.5	**9.1 / 7.9 / 14.2 / 13.4**
Fisher-CallHome	BPE	WER	dev / dev2 / test / devtest / evltest	22.1 / 21.5 / 19.9 / 38.1 / 38.2	**21.5 / 21.1 / 19.4 / 37.4 / 37.5**
HKUST	Char	CER	dev	(†) 23.5	(†) **22.2**
JSUT	Char	CER	our split	(†) 18.7	**14.5**
LibriSpeech	BPE	WER	{dev, test}_{clean, other}	2.1 / 5.3 / 2.5 / 5.5	**1.9 / 4.9 / 2.1 / 4.9**
REVERB	Char	WER	et_{near, far}	(†) 13.1 / 15.4	(†) **10.5 / 13.9**
Switchboard	BPE	WER	eval2000 (callhm / swbd)	17.2 / 8.2	**14.0 / 6.8**
TEDLIUM2	BPE	WER	dev / test	9.3 / 8.1	**8.6 / 7.2**
TEDLIUM3	BPE	WER	dev / test	10.8 / 8.4	**9.6 / 7.6**
VoxForge	Char	CER	our split	(§) 9.4 / 9.1	(§) **8.7 / 8.2**
WSJ	BPE	WER	dev93/ eval92	(‡) **7.4 / 4.9**	(‡) 7.7 / 5.3
WSJ-2mix	Char	WER	tt	(§) 12.6	(§) **11.7**

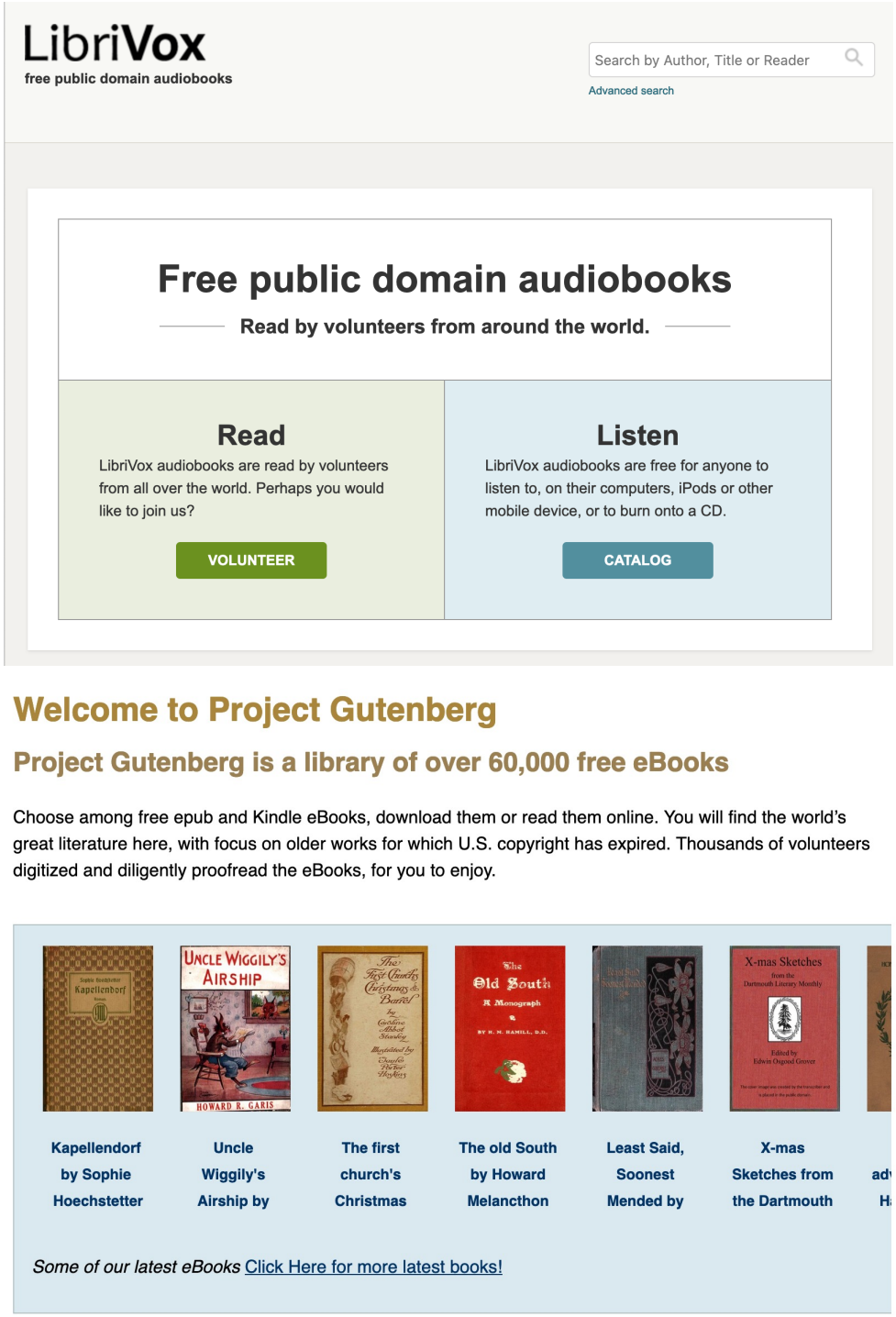
Guo, Boyer, Chang, Hayashi, Higuchi et al., ICASSP 2021, © IEEE

Outline

- Lexical semantics
  - Wordforms, lemmas & word senses; Synonymy & similarity
  - Vector semantics: CBOW and skip-gram
  - Generative training vs. Contrastive training
  - Implicit meaning representations in lexical semantic spaces
- Sentence semantics
  - Recurrent neural networks (RNN)
  - Attention
  - Self-attention, Multi-headed attention, Cross-attention, and Masked attention
  - Self-training

Label quality

- Audiobooks on librivox.org are readings of texts from Gutenberg.org.
- Often, readers make mistakes, or read different versions, or change the title enough to make it hard to know which Gutenberg text they read.
- Until 2020, speech technology was trained using “labeled data:”
  - 400 hours of librivox books with verified transcripts (“Librispeech Clean”)
  - 600 hours of librivox books with acoustic noise or transcript errors (“Librispeech Other”)



What do babies hear?

How much unlabeled speech does a baby hear?

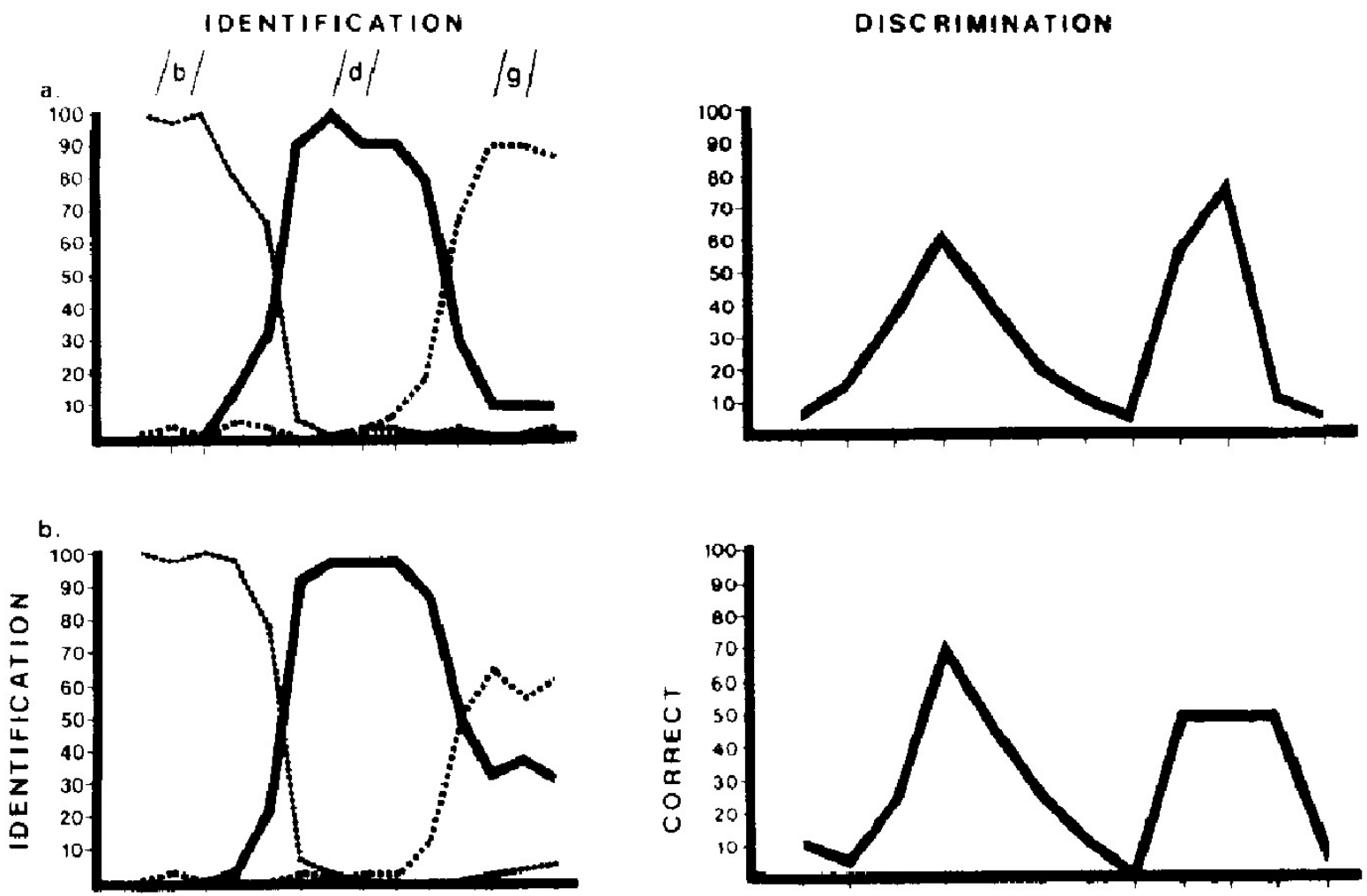
- 2000-15000 words/day = 600-4500 hours of speech by age 6 (Weisleder & Fernald, 2013)

How much labeled speech does a baby hear?

- 30 (?) words/day accompanied by referential gestures = 9.1 hours of speech by age 6

By Steve Jurvetson from Los Altos, USA - A Proper Space Book for Babies, CC BY 2.0, https://commons.wikimedia.org/w/index.php?curid=105132804

Is speech learned, or innate? (Hint: it's a trick question)

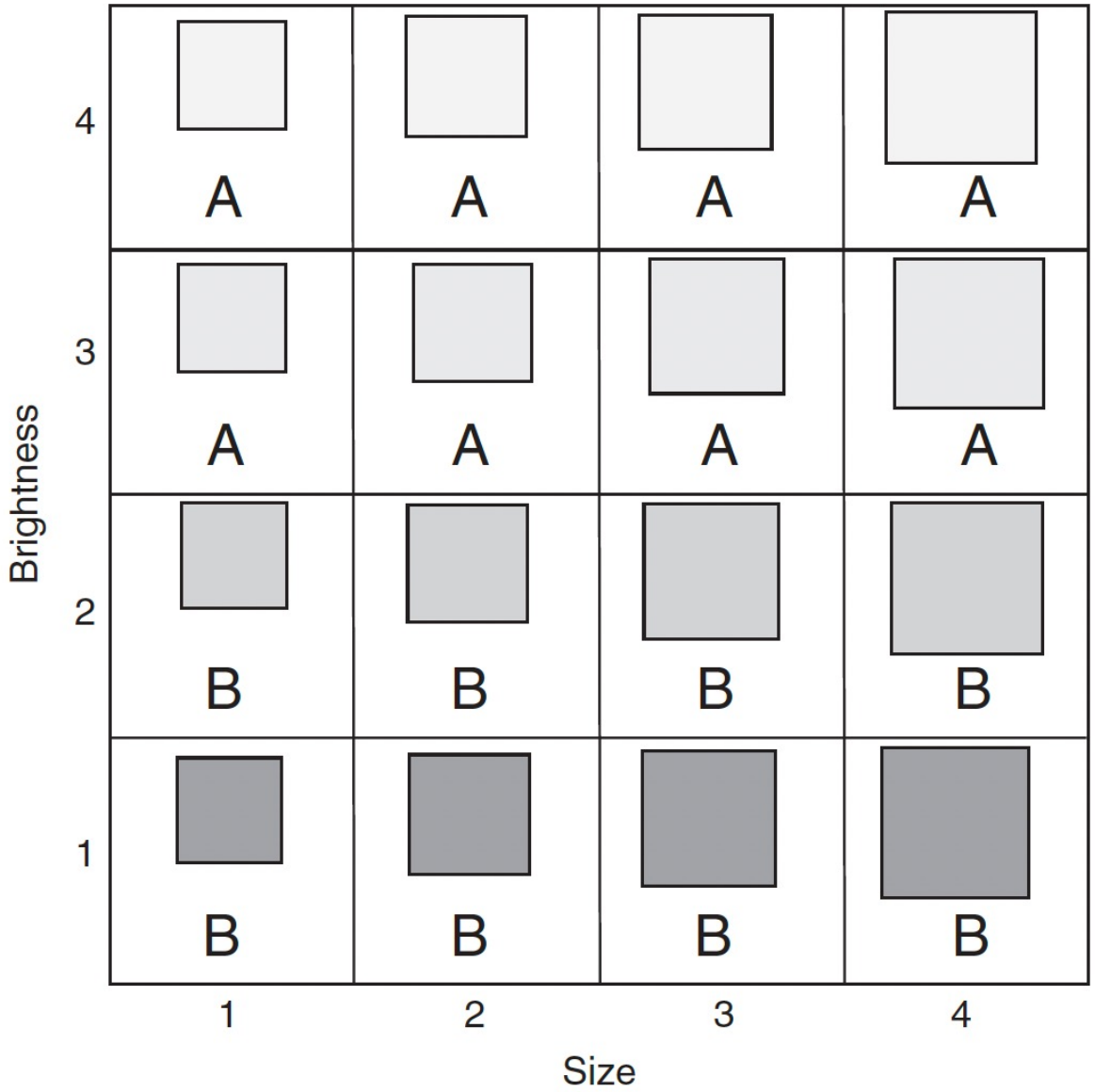


[Brandt & Rosen, 1980](#) © Brain & Language

- 15 synthetic syllables, continuous from /ba/ to /da/ to /ga/
- same label ⇒ hard to tell if they are same sound or different sounds
- different labels ⇒ heard as obviously different

Categorical perception can be learned!

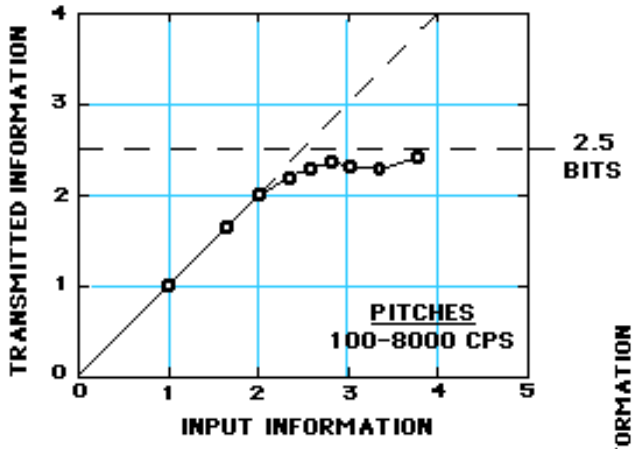
- People trained to categorize based on ***brightness*** show reduced within-category perceptual memory, and greater across-category perceptual memory, for ***brightness***, but ***size*** is perceived on a continuum.
- People trained to categorize based on ***size*** show reduced within-category perceptual memory, and greater across-category perceptual memory, for ***size***, but ***brightness*** is perceived on a continuum.

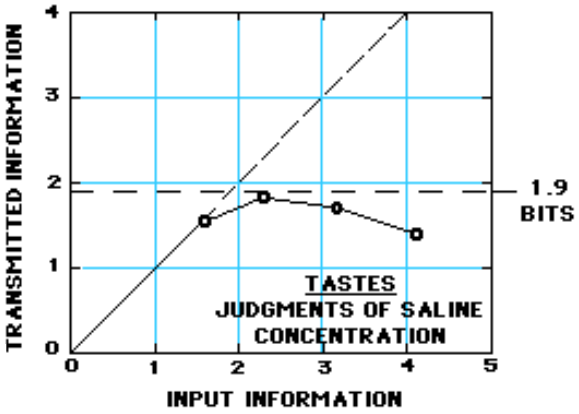


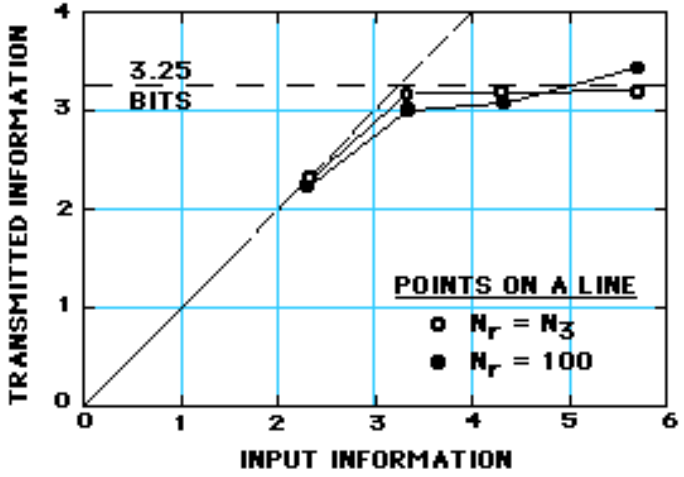
Goldstone & Hendrickson, 2009. © Wiley & Sons

Categorical perception as a cognitive bias

- If we categorize things, maybe we can remember them longer.
- In “The Magic Number Seven,” Miller argued that people can be taught to categorize any continuum (pitch, taste, position, size) into seven categories, but not more.

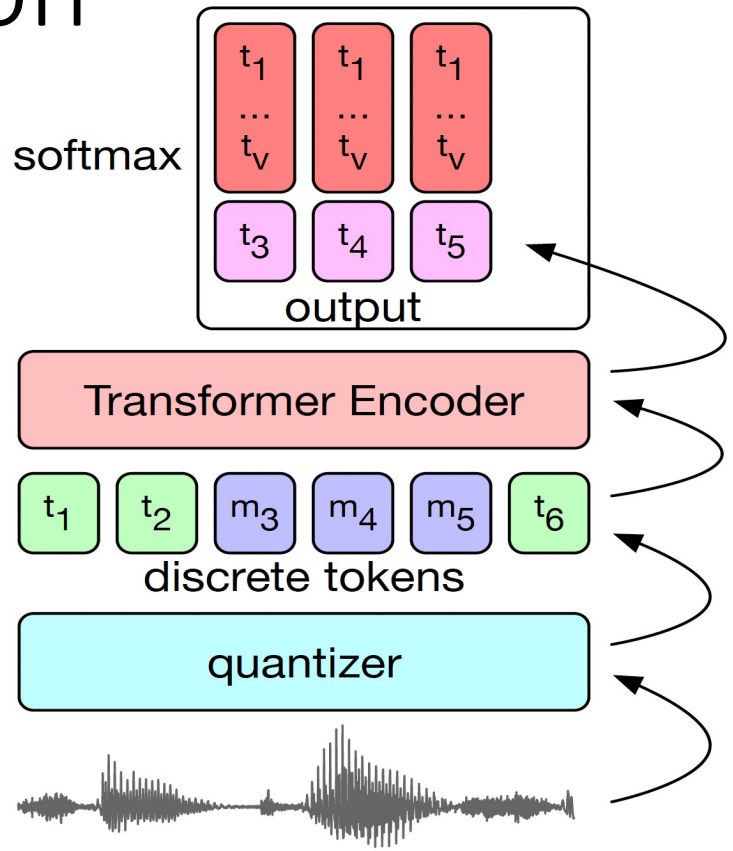






Unsupervised pre-training of transformers based on categorical perception

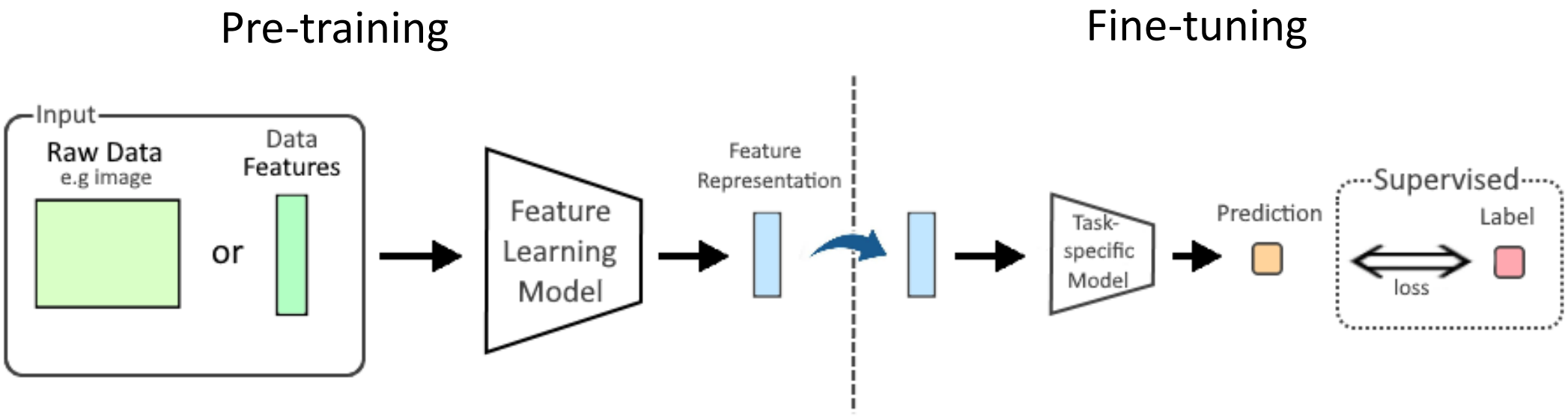
- Given: 60,000 hours of speech, with no associated text.
- Suppose we train the neural network to form its own categories. What would make those categories speech-like?
- **Context-Predictable Speech Categories**: given the context (the quantized units t_1, t_2, and t_6), it should be possible to figure out what phonemes were masked (t_3, t_4, t_5).



Baevski, Auli & Mohamed, 2019

Pre-training and Fine-tuning

- A transformer is pre-trained to create its own context-predictable speech categories using, say, 60,000 hours of speech
- Then it is fine-tuned using a few hours, or a few hundred hours, or labeled speech



Word Error Rates using Pre-Training

Pre-training makes it possible to achieve error rates of

- 4.4% using only 10 minutes of labeled data
- 2.6% using only 1 hour of labeled data

Model	Unlabeled Data	LM	dev-clean	dev-other	test-clean	test-other
		10-min labeled				
DiscreteBERT [52]	LS-960	4-gram	15.7	24.1	16.3	25.2
wav2vec 2.0 BASE [7]	LS-960	4-gram	8.9	15.7	9.1	15.6
wav2vec 2.0 LARGE [7]	LL-60k	4-gram	6.3	9.8	6.6	10.3
wav2vec 2.0 LARGE [7]	LL-60k	Transformer	4.6	7.9	4.8	8.2
HUBERT BASE	LS-960	4-gram	9.1	15.0	9.7	15.3
HUBERT LARGE	LL-60k	4-gram	6.1	9.4	6.6	10.1
HUBERT LARGE	LL-60k	Transformer	**4.3**	7.0	4.7	7.6
HUBERT X-LARGE	LL-60k	Transformer	4.4	**6.1**	**4.6**	**6.8**
		1-hour labeled				
DeCoAR 2.0 [51]	LS-960	4-gram	-	-	13.8	29.1
DiscreteBERT [52]	LS-960	4-gram	8.5	16.4	9.0	17.6
wav2vec 2.0 BASE [7]	LS-960	4-gram	5.0	10.8	5.5	11.3
wav2vec 2.0 LARGE [7]	LL-60k	Transformer	2.9	5.4	2.9	5.8
HUBERT BASE	LS-960	4-gram	5.6	10.9	6.1	11.3
HUBERT LARGE	LL-60k	Transformer	**2.6**	4.9	2.9	5.4
HUBERT X-LARGE	LL-60k	Transformer	**2.6**	**4.2**	**2.8**	**4.8**

Hsu, Boldt, Tsai, Lakhotia, Salakhutdinov & Mohamed,

Summary

Contrastive learning of CBOW or skip-gram:

$$\mathcal{L} = \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{D}_+(w)} \ln\left(1 + e^{-c^T v}\right) + \sum_{v \in \mathcal{V}} \sum_{c \in \mathcal{D}_-(w)} \ln\left(1 + e^{c^T v}\right)$$

Attention:

$$c_i = \sigma(K q_i)^T V$$