Natural Language Processing

NLP | Lecture 4

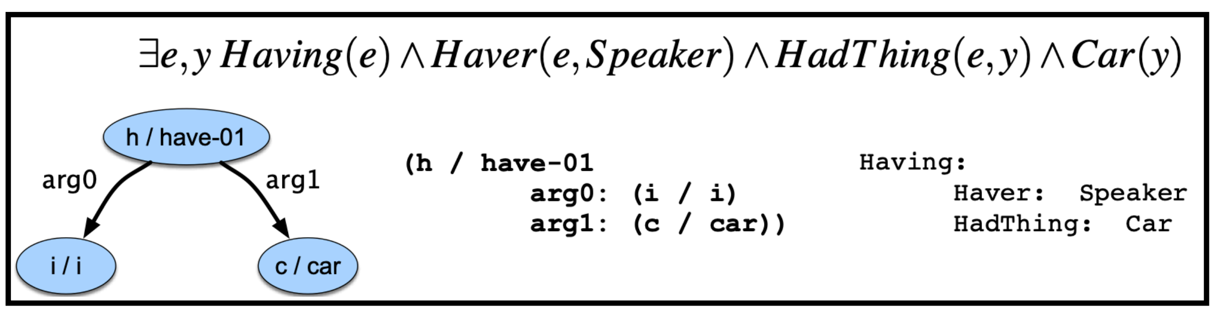
Word Embeddings and Language Models

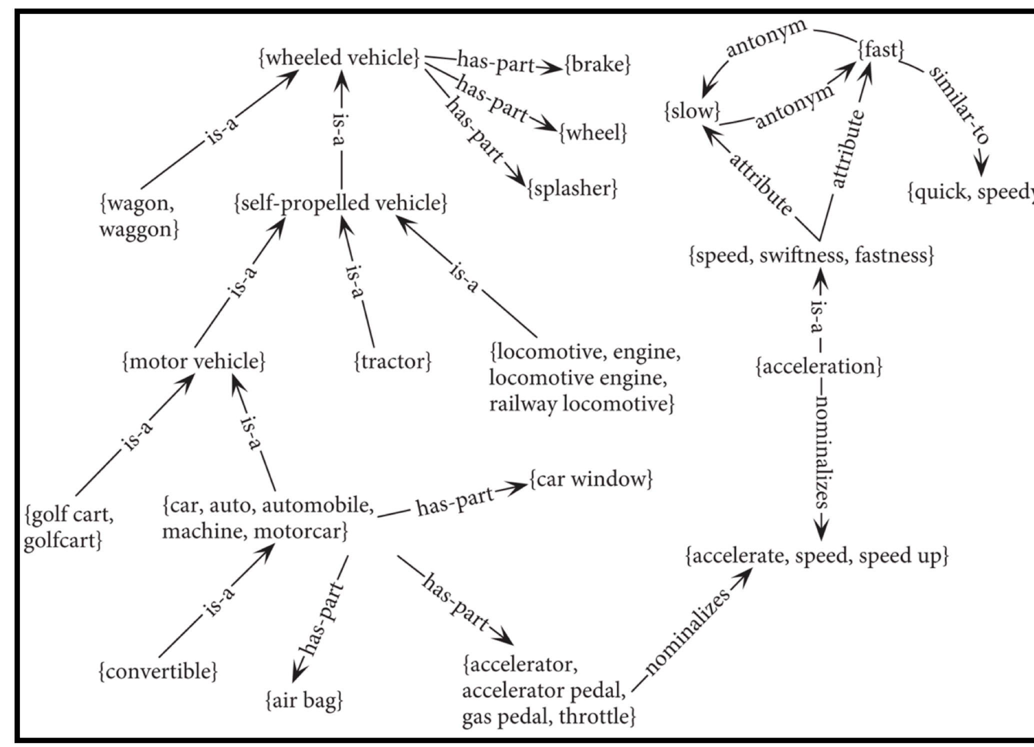
Aron Henriksson

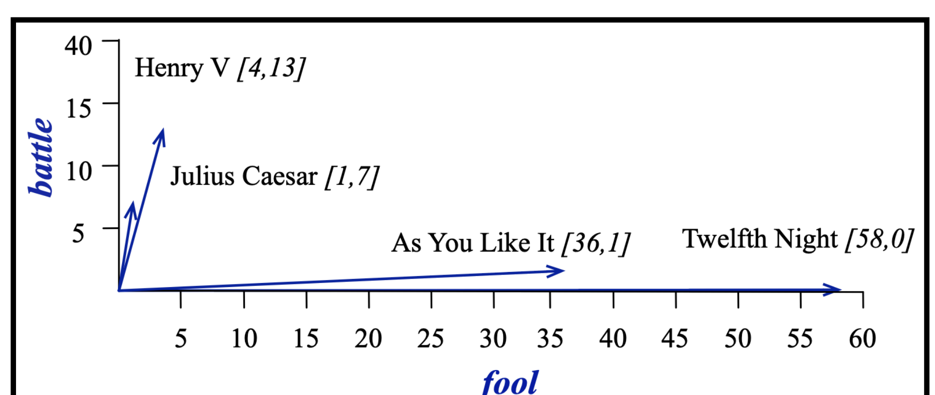
Semantics in Natural Language Processing

NLP systems must account for semantics, i.e. linguistic meaning

- Many different approaches to semantics in NLP
  - Computational semantics
  - Frame semantics
  - Distributional semantics







Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

Lexical Semantics

The linguistic study of word meaning

- The meaning of a text can be derived from the meaning of words: **semantic composition**
- How should we **represent** the **meaning** of a word?
- Lexical semantics can help us to formulate **desiderata** of **word representations**
  - Lemmas and wordforms
  - Senses and polysemy
  - Semantic and taxonomic relationships
  - Word similarity
  - Word relatedness
  - Connotations

Stockholm University

Lemmas and Wordforms

mouse (N)

1. any of numerous small rodents...
2. a hand-operated device that controls a cursor...

'mouse' is the **lemma**

- Also lemma for 'mice'
- 'sing' is the lemma for 'sing', 'sang', 'sung'
- Lemma for verbs: infinitive form

The specific forms are called **wordforms**

Stockholm University

Senses and Polysemy

mouse (N)

1. any of numerous small rodents...
2. a hand-operated device that controls a cursor...

Each lemma can have multiple meanings

- Each of these aspects of the meaning of mouse is a **word sense**
- Lemmas with multiple meanings are **polysemous**
- Can make interpretation difficult — language is ambiguous

Word sense disambiguation: which sense is used in a particular context?

Semantic Relations

Synonyms

- A sense of a word whose meaning is (nearly) identical to a sense of another word (couch/sofa — vomit/throw up — car/automobile)
- Substitutable in any sentence without changing its truth condition: same **propositional meaning**
- **Principle of contrast**: a difference in linguistic form is always associated with some difference in meaning

Antonyms

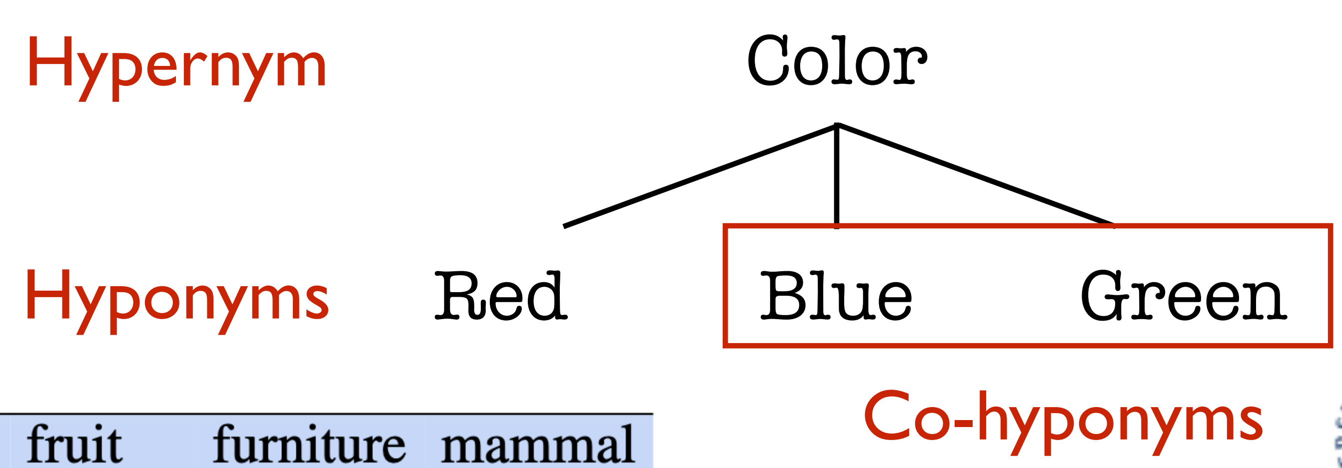
- Words with opposite meanings (bad/good — hard/easy — fast/slow)
- Define binary opposition or at opposite ends of some scale (long/short)
- **Reversives** describe change/movement in opposite directions (up/down)

Stockholm University

Taxonomic Relations

Taxonomic relations

- A word is a **hyponym** of another word if it is more specific, denoting a subclass of the other (car/vehicle — dog/animal)
- A word is a **hypernym** of another word if it is more general, denoting a superclass of the other (vehicle/car — animal/dog)
- Also known as **superordinate/subordinate** or **IS-A hierarchy**
- Hypernymy useful for **textual entailment** and **question answering**
- Other taxonomic relations: meronymy, metonymy etc.



Superordinate	vehicle	fruit	furniture	mammal
Subordinate	car	mango	chair	dog

Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

Word Similarity

Synonymy is rare, but words often have many similar words

- 'cat' and 'dog' are not synonymous, but they are similar words
- Notion of **word similarity** is very useful in larger semantic tasks
- Word similarity helps compute similarity between phrases or sentences
- Important for **question answering**, **paraphrasing** and **summarization**

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

Examples from SimLex-999: human judgment of word similarity on a scale from 0 to 10

Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

Word Relatedness

Words can be related in ways other than similarity

- One such class of connections is called **word relatedness**
- ‘coffee’ and ‘cup' are not similar but clearly related — associated by co-participating in an the event of drinking coffee out of a cup
- Some related words belong to the same **semantic field**, i.e. a set of words encompassing a particular semantic domain
  - **Hospitals**: ‘surgeon’, ‘scalpel’, ‘nurse’, ‘anesthetic’, ‘hospital’
  - **Restaurants**: ‘waiter’, ‘menu’, ‘plate’, ‘food’, ‘chef’
  - **Houses**: ‘door’, ‘roof’, ‘kitchen’, ‘family’, ‘bed’
- Semantic fields related to **topic models**

Connotation

Words have affective meanings or connotations

- Aspects of a word's meaning related to emotions, sentiment, opinions, or evaluations
- Some words have positive ('happy') or negative ('sad) connotations
- Similar words can have different connotations: 'innocent' vs. 'naive'
- Positive or negative evaluation language is called **sentiment** (positive: 'great', 'love' — negative: 'terrible', 'hate')
- Word sentiment important in **sentiment analysis** and **stance detection**

Representing the Meaning of Words

Representing words as a string of letters

- Or an index in a **vocabulary**
- Convert symbols to numbers

V = {'files', 'find', 'my'}

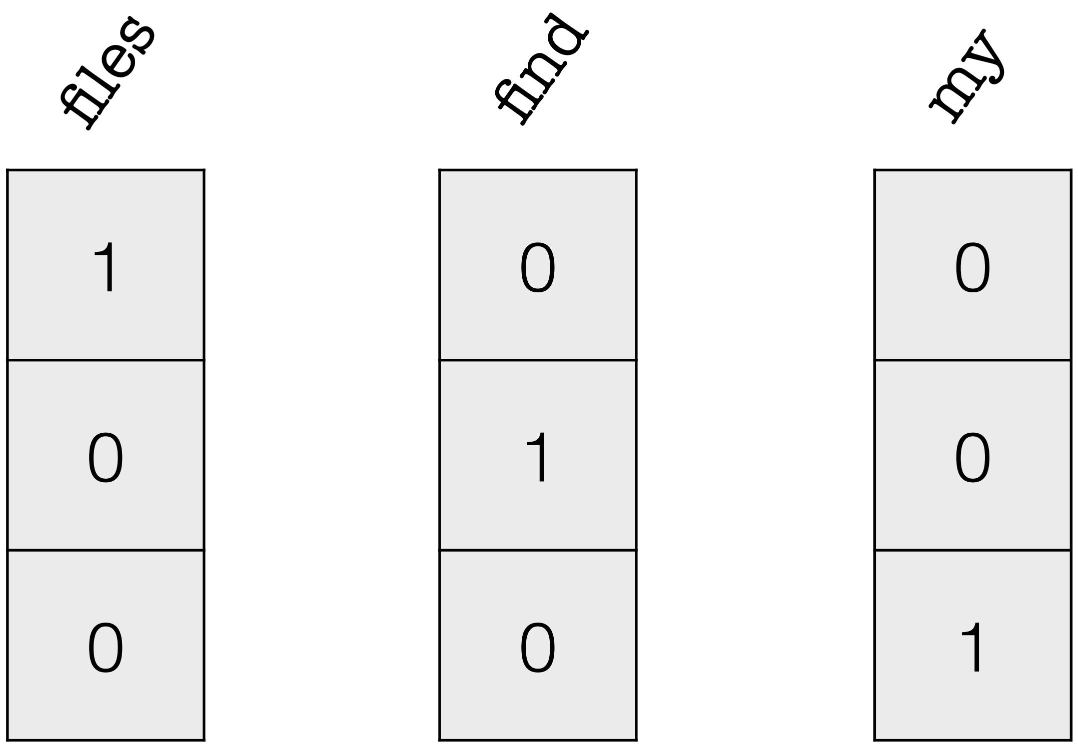
'files' = 1
'find' = 2
'my' = 3

'find my file' —> ['find', 'my', ['file'] —> [2, 3, 1]

One Hot Encoding

Representing words as vectors

- Symbol represented by an array of the same length as the vocabulary size
- All zeros except a single element with a value of one
- Each element corresponds to a separate symbol



Stockholm University

Dot Product

How can we calculate the similarity between words?

$$
\begin{array}{ccccc}
\boxed{0} & \times & \boxed{1} & = & 0 \\
\boxed{1} & \times & \boxed{0} & = & 0 \\
\boxed{1} & \times & \boxed{1} & = & 1 \\
\boxed{2} & \times & \boxed{2} & = & 4 \\
 & & & + & \overline{} \\
 & & & & 5
\end{array}
$$

Dot Product

The dot product of any one-hot vector with itself is one

$$
\begin{array}{|c|}\hline 0 \\ \hline 1 \\ \hline 0 \\ \hline 0 \\ \hline \end{array}
\begin{array}{c} \times \\ \times \\ \times \\ \times \end{array}
\begin{array}{|c|}\hline 0 \\ \hline 1 \\ \hline 0 \\ \hline 0 \\ \hline \end{array}
\begin{array}{l} = 0 \\ = 1 \\ = 0 \\ = 0 \end{array}
$$

$$
\begin{array}{r} + \\ \hline 1 \end{array}
$$

Dot Product

The dot product of any one-hot vector with any other one-hot vector is zero

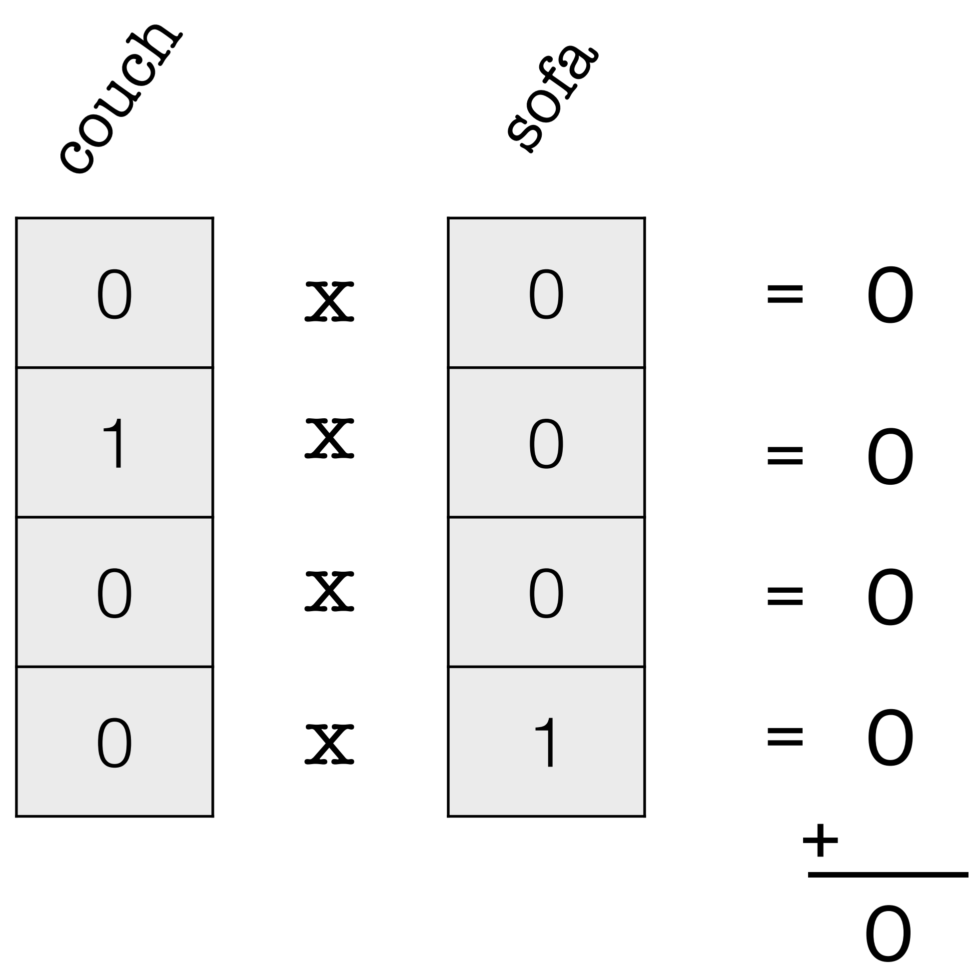
$$
\begin{array}{ccccc}
\boxed{0} & \times & \boxed{0} & = & 0 \\
\boxed{1} & \times & \boxed{0} & = & 0 \\
\boxed{0} & \times & \boxed{0} & = & 0 \\
\boxed{0} & \times & \boxed{1} & = & 0 \\
 & & & + & \overline{} \\
 & & & & 0
\end{array}
$$

Stockholm University

Dot Product

No notion of semantic similarity between words represented

- Synonyms have orthogonal representations with one-hot encoding!



Vector Semantics

Representing words as vectors

- The standard way to represent word meaning in NLP
- Representing word meaning as a vector goes back to 1950s and Osgood's idea to use a point in 3-dimensional space to represent word connotation
- Ideas in 1950 to define word meaning by its **distribution in language use**

	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

Words represented in a 3-dimensional space

Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

Desiderata of Word Representations

We want a model of word meaning to account for many aspects of semantics

- ❖ Some words have similar meanings, others have unrelated meanings
- ❖ Some words are synonyms, others are antonyms
- ❖ Some word pairs have taxonomic relations, e.g hypernym-hyponym
- ❖ Words are polysemous and have ambiguous, context-dependent meanings
- ❖ Some words have positive connotations, others have negative connections

Allow us to make inferences to address meaning-related tasks

Distributional Hypothesis

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

- The meaning of a word can be derived from the contexts in which it appears — **Z. Harris** (1954)

Meaning is use

L. Wittgenstein

You shall know a word by the company it keeps

J.R. Firth

What is there to watch on the television tonight?

I want to watch TV!

We must watch the news on the telly.

Stockholm University

Distributional Hypothesis

What is ‘ongchoi’?

Ongchoi is delicious sautéed with garlic.

Ongchoi is superb over rice.

...**ongchoi** leaves with salty sauces...

...**spinach** sautéed with garlic over rice...

...**chard stems** and leaves are delicious...

...**collard greens** and other salty leafy greens

A leafy green similar to these other leafy greens?

Stockholm University

Distributional Semantics

Exploits **distributional hypothesis** and **large corpora** to model word meaning

- A word is represented as a point (vector) in a multidimensional **semantic space**
- Derived from the distributions of contexts (word neighbors)
- Offers enormous power to NLP applications

A two-dimensional (t-SNE) projection of 60-dimensional semantic representations for words and phrases

Jurafsky &. Martin, 2022. Speech and Language Processing.

Models of Distributional Semantics

An evolution of models of distributional semantics

- Counting-based vs. prediction-based models
- Sparse vs. dense word vectors (or embeddings)
- Static vs. dynamic (context-specific) word vectors

All learned automatically from large corpora without supervision

- No need for manually labeled data!

Term-Document Matrix

The vector space model of information retrieval

- Each **row** represents a word in the vocabulary
- Each **column** represents a document from a document collection
- Each **cell** represents the number of times a particular word occurs in a particular document
- A document is represented as a **count vector**
- Document vectors of **dimension** |V| = vocabulary size (= 4 in example)

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

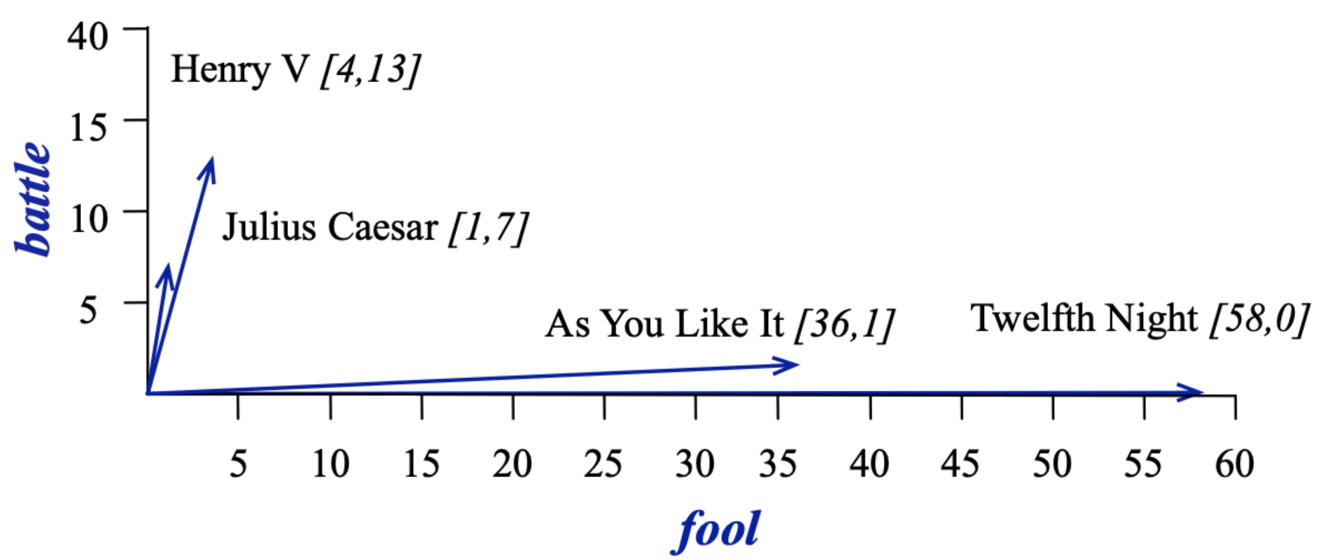
Jurafsky &. Martin, 2022. Speech and Language Processing.

As You Like It —> [1,114,36,20]

Term-Document Matrix

The vector space model of information retrieval

- Originally defined as means of finding similar documents
- Two documents that are similar will have similar words
- Documents with similar words will have similar column vectors
- As You Like It [1,114,36,20] and Twelfth Night [0,80,58,15] more similar to each than to Julius Caesar [7,62,1,2] or Henry V [13,89,4,3]
- The term-document matrix has |V| rows and D columns



Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

Words as Vectors: Document Dimensions

Vector semantics to represent the meaning of words

- Each word is associated with a word vector (a row vector)
- Dimensions correspond to documents (here: Shakespeare plays)
- Similar words have similar vectors because they tend to occur in similar context
- The **term-document matrix** lets us represent word meaning by the documents it occurs in

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	13
good	114	80	62	89
fool	36	58	1	4
wit	20	15	2	3

Jurafsky &. Martin, 2022. Speech and Language Processing.

'battle' —> [1,0,7,13]

Words as Vectors: Word Dimensions

Vector semantics to represent the meaning of words

- Represent words as vectors of document counts
- Columns labeled by words rather than documents
- This is a **term-term matrix** of dimensionality |V| × |V|
- Each cell records the number of times the row (target) word and the column (context) word **co-occur** in some **context** in a given corpus

Words as Vectors: Word Dimensions

Vector semantics to represent the meaning of words

- Different **context definitions** exist:
  - A document
  - Words in a window surrounding the target word (most common)
  - Syntactic dependencies

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

A context window of 4 words to the left and 4 words to the right of the target word

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

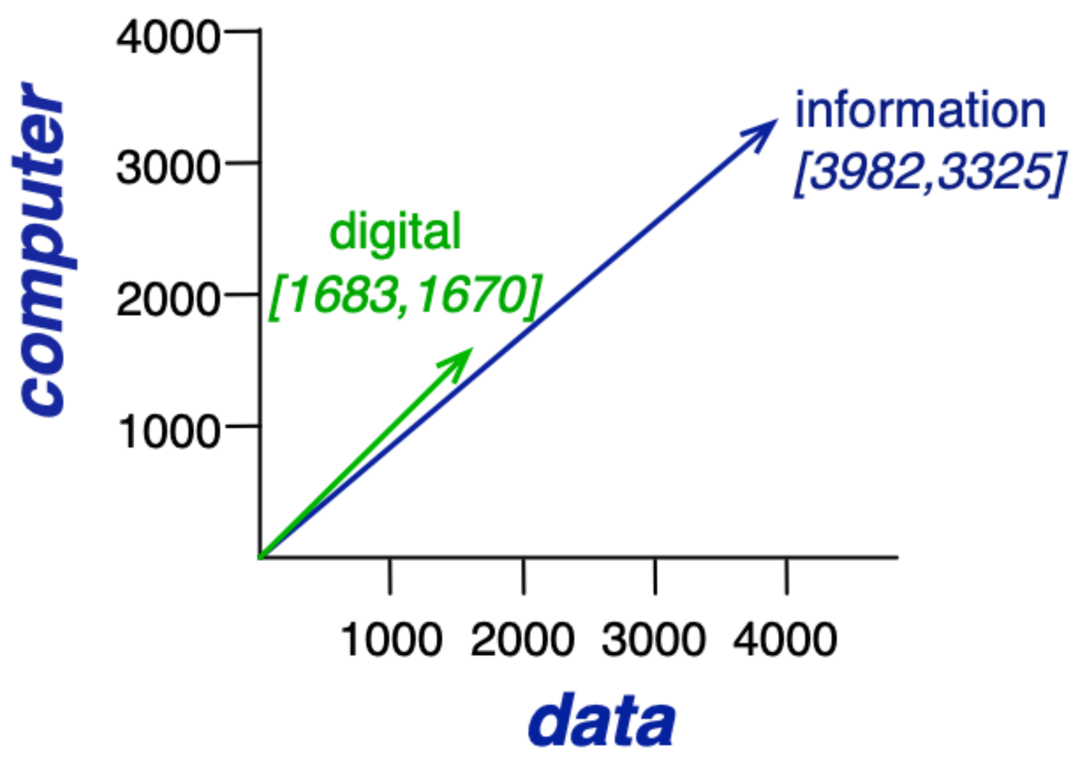
Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

Words as Vectors: Word Dimensions

Vector semantics to represent the meaning of words

- A spatial visualization of word vectors for ‘digital’ and ‘information’, showing just two of the dimensions: ‘computer’ and ‘data’



Jurafsky &. Martin, 2022. Speech and Language Processing.

Measuring Semantic Similarity

How can we calculate the similarity between words?

- Need metric that takes two vectors (of same dimensionality) and returns a measure of their **similarity**
- How about using the **dot product**?
  - High when large values in same dimensions
  - Orthogonal vectors (zeros in different dimensions) will return 0

$$\boxed{\text{dot product}(\mathbf{v},\mathbf{w}) = \mathbf{v}\cdot\mathbf{w} = \sum_{i=1}^{N} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_N w_N}$$

Jurafsky &. Martin, 2022. Speech and Language Processing.

Problem: favors long vectors!

- More frequent words have longer vectors
- We want a similarity metric that is not affected by term frequency!

$$\boxed{|\mathbf{v}| = \sqrt{\sum_{i=1}^{N} v_i^2}}$$

Measuring Semantic Similarity

How can we calculate the similarity between words?

- Solution: modify dot product to normalize for vector length
- This **normalized dot product** is the same as the **cosine** of the angle between the two vectors

$$\boxed{\begin{aligned} \mathbf{a} \cdot \mathbf{b} &= |\mathbf{a}||\mathbf{b}| \cos\theta \\ \frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}||\mathbf{b}|} &= \cos\theta \end{aligned}}$$

Cosine similarity — the most common similarity metric

- 1 for vectors in the same direction, 0 for orthogonal vectors, -1 for vectors in opposite directions
- Since raw frequency values are non-negative: cosine ranges from 0-1

$$\boxed{\text{cosine}(\mathbf{v}, \mathbf{w}) = \frac{\mathbf{v} \cdot \mathbf{w}}{|\mathbf{v}||\mathbf{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2} \sqrt{\sum_{i=1}^{N} w_i^2}}}$$

Jurafsky &. Martin, 2022. Speech and Language Processing.

Measuring Semantic Similarity

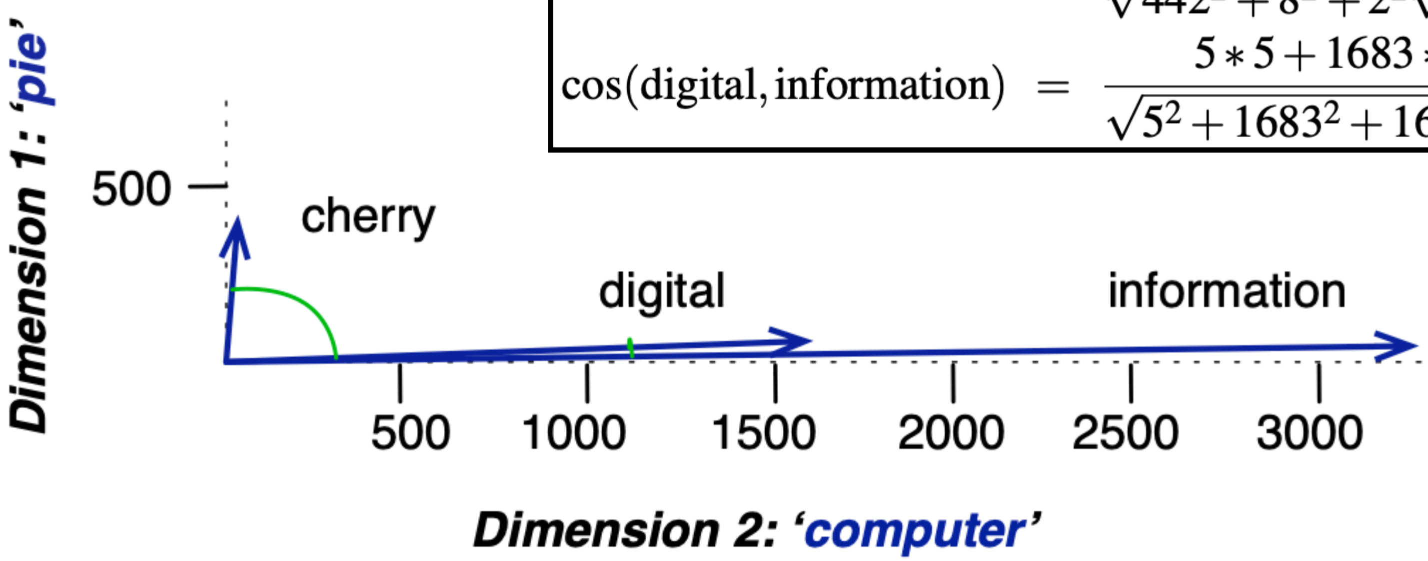
How can we calculate the similarity between words?

- Calculating cosine similarity between ‘cherry’ and ‘information’ vs. ‘digital’ and ‘information’
- 2D visualization of three vectors

	pie	data	computer
cherry	442	8	2
digital	5	1683	1670
information	5	3982	3325

$$\cos(\text{cherry}, \text{information}) = \frac{442*5+8*3982+2*3325}{\sqrt{442^2+8^2+2^2}\sqrt{5^2+3982^2+3325^2}} = .018$$

$$\cos(\text{digital}, \text{information}) = \frac{5*5+1683*3982+1670*3325}{\sqrt{5^2+1683^2+1670^2}\sqrt{5^2+3982^2+3325^2}} = .996$$



Jurafsky &. Martin, 2022. Speech and Language Processing.

Weighting Co-Occurrences

Not all co-occurrence events are equally significant!

- Raw frequency not the best measure of word association
- Raw frequency is skewed and not very discriminative
- To discriminate 'cherry' and 'strawberry' from 'digital' and 'information'
  - Words like 'the', 'it' and 'they' will not provide good discrimination
  - Occur frequently will all words and therefore not informative
- **Paradox**: words that appear nearby frequently are important, but words that are too frequent are unimportant

How can we balance these two conflicting constraints?

- Two common solutions: **tf-idf** and **PPMI** weighting

Weighting Co-Occurrences

tf-idf

- Used when the dimensions are documents
- **Term frequency × inverse document frequency**
  - tf: the frequency of word t in document d
  - idf: give a higher weight to words that occur only in a few documents (N/df_t — N = total # of docs, df_t = # of docs in which term t occurs)
- **Intuition**: words appearing in few documents have a high discriminative power!

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	0.074	0	0.22	0.28
good	0	0	0	0
fool	0.019	0.021	0.0036	0.0083
wit	0.049	0.044	0.018	0.022

A tf-idf weighted term-document matrix

Jurafsky &. Martin, 2022. Speech and Language Processing.

Weighting Co-Occurrences

PPMI

- Used when dimensions are words (term-term matrices)
- Positive Pointwise Mutual Information
- **Intuition**: weigh association between two words according to how much more the words co-occur than expected
- PMI values range from negative to positive infinity
  - Negative PMI values tend to be unreliable — PPMI often used instead (replaces negative PMI values with 0)

$$\boxed{\text{PMI}(w,c) = \log_2 \frac{P(w,c)}{P(w)P(c)}}$$

$$\boxed{\text{PPMI}(w,c) = \max(\log_2 \frac{P(w,c)}{P(w)P(c)},0)}$$

	computer	data	result	pie	sugar
cherry	0	0	0	4.38	3.30
strawberry	0	0	0	4.10	5.51
digital	0.18	0.01	0	0	0
information	0.02	0.09	0.28	0	0

PPMI matrix

Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

High-Dimensional and Sparse Vectors

Context-counting vectors are high-dimensional and sparse

- The dimensionality of the vector is the size of the vocabulary: |V|
- The vocabulary of a corpus can be > 1M
  - Common to limit vocabulary to 10,000-50,000 most frequent words
- Co-occurrences between all word pairs are rare events!
  - Most cells are 0
  - Sparse vectors
- Dimensionality reduction

Stockholm University

Sparse vs. Dense Vector Representations

Sparse vectors

- Dimensions corresponding to words in the vocabulary

Dense vectors

- Dimensions ranging from 50–1000, difficult to interpret
- Dense vectors are often referred to as **word embeddings**

Dense vectors work better for most NLP tasks — why?

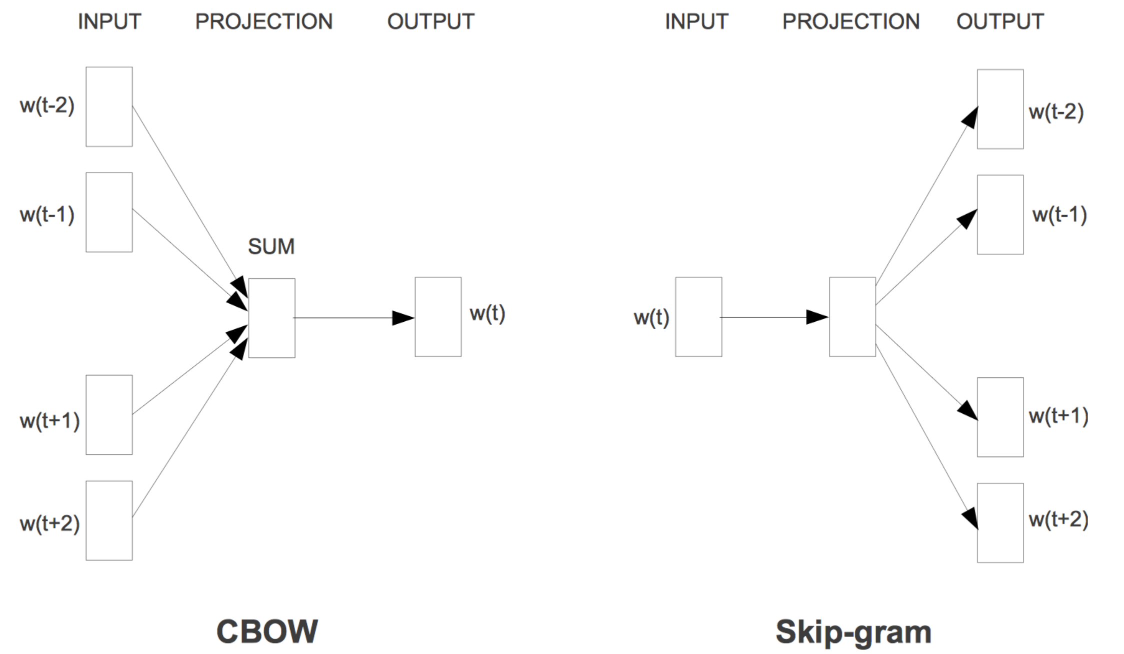
- Smaller parameter space helps with generalization and avoiding overfitting
- Dense vectors better at capturing synonymy? Sparse vectors have distinct dimensions for synonymous context words

Stockholm University

word2Vec

Prediction-based model for creating word embeddings

- Instead of counting how often each word w occurs in different contexts, we train a classifier on one of two binary prediction tasks
  - Predict target word based on an neighboring context words (CBOW)
  - Predict neighboring context words based on target word (Skip-gram)
- **The prediction task itself is uninteresting** — we use the learned classifier weights as the word embeddings



Mikolov et al., 2013. Efficient estimation of word representation in vector space.

Self-Supervision

Use unlabeled data in a supervised learning setting

- Use running text as implicitly supervised training data for the classifier
- Words that occurs near a target word provide positive examples
- Words that do not occur near a target word provide negative examples

Self-supervision was first proposed in neural language modeling task

- A neural network that learned to predict the next word
- Used the next word as its supervision signal
- word2vec is much simpler model
  - Simpler task: binary classification instead of word prediction
  - Simpler architecture: logistic regression instead of multi-layer NN

Skip-gram with Negative Sampling

The intuition of SGNS

1. Treat the target word and a neighboring context word as positive examples
2. Randomly sample other words in the vocabulary to get negative samples
3. Use logistic regression to train a classifier to distinguish those two cases
4. Use the learned weights as the embeddings

Skip-gram with Negative Sampling

The classification task

- Train classifier such that:
  - Given a tuple (w,c) of a target word w paired with a candidate context word c — e.g. (‘apricot’, ’jam’) or (‘apricot', ‘aardvark’)
  - It will return the probability that c is a real context word (true for ‘jam’, false for ‘aardvark’)

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

Jurafsky &. Martin, 2022. Speech and Language Processing.

- Probabilities computed based on **embedding similarity**: a word is likely to occur near target word if its embedding is similar to target embedding
  - Apply sigmoid function to dot product of embeddings of target word with each context word

Stockholm University

Skip-gram with Negative Sampling

Learning skip-gram embeddings

- Learning algorithm takes as input a corpus and vocabulary size N
- First, assigns random embedding to each word
- Then, proceeds to iteratively shift embeddings of each word w to be more likely embeddings of context words and less like non-context words
- Negative examples: randomly sampled ‘noise words’

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

positive examples +

w	c_{pos}
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

Skip-gram with Negative Sampling

Given training data — and an initial set of embeddings — the goal of the learning algorithm is 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 the (w,c_{neg}) pairs from the negative examples

In other words, we want to:

- Maximize the dot product of the word with the actual context words
- Minimize the dot products of the word with the k negative sampled non-neighbor words.

Loss function minimized using stochastic gradient descent

Stockholm University

Other Static Embeddings

fasttext

- An extension of word2vec
- Addresses the inability to deal with **unknown words** — words in a test corpus that were not seen in the training corpus
- Also deals with **word sparsity**, e.g. in languages with rich morphology
- Uses **subword models**, representing each word as itself + bag of constituent **character n-grams**, with special boundary symbols <>
- Example: with n=3, 'where' — <where> + <wh, whe, her, ere, re>
- Skipgram embedding learned for each constituent n-gram and the word is represented by sum of embeddings
- Unknown words represented only by sum of of constituent n-grams

Other Static Embeddings

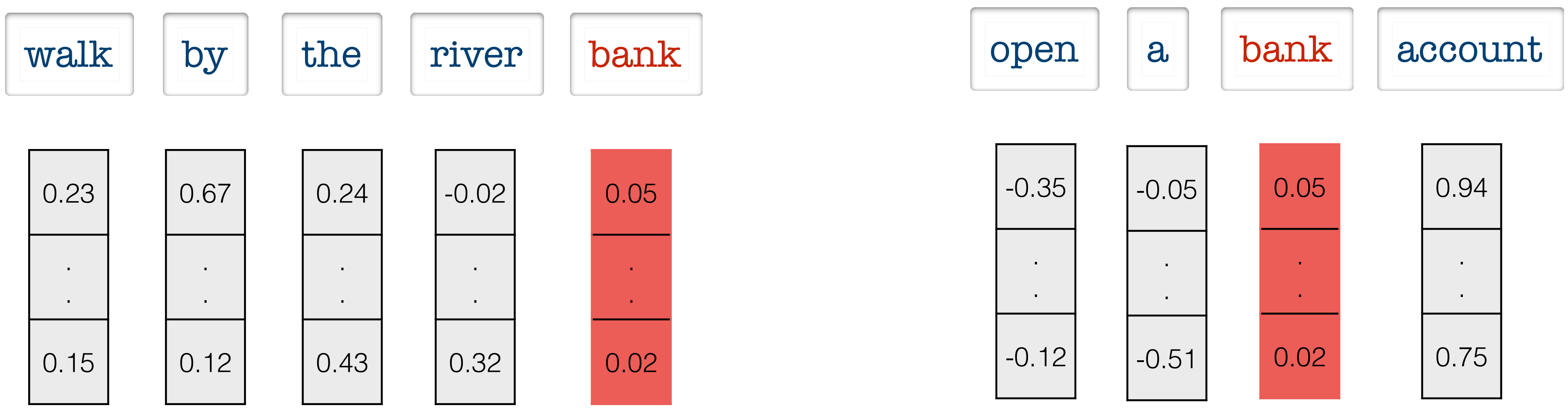
GloVe

- Short for **Global Vectors**
- Captures global **corpus statistics**
- Based on ratios of probabilities from word-word co-occurrence matrix
- Combines the intuitions of count-based sparse vector models while capturing linear structures by prediction-based dense vector models

Static vs. Dynamic Word Embeddings

Static Embeddings

- A single vector for each unique word w in the vocabulary (types)



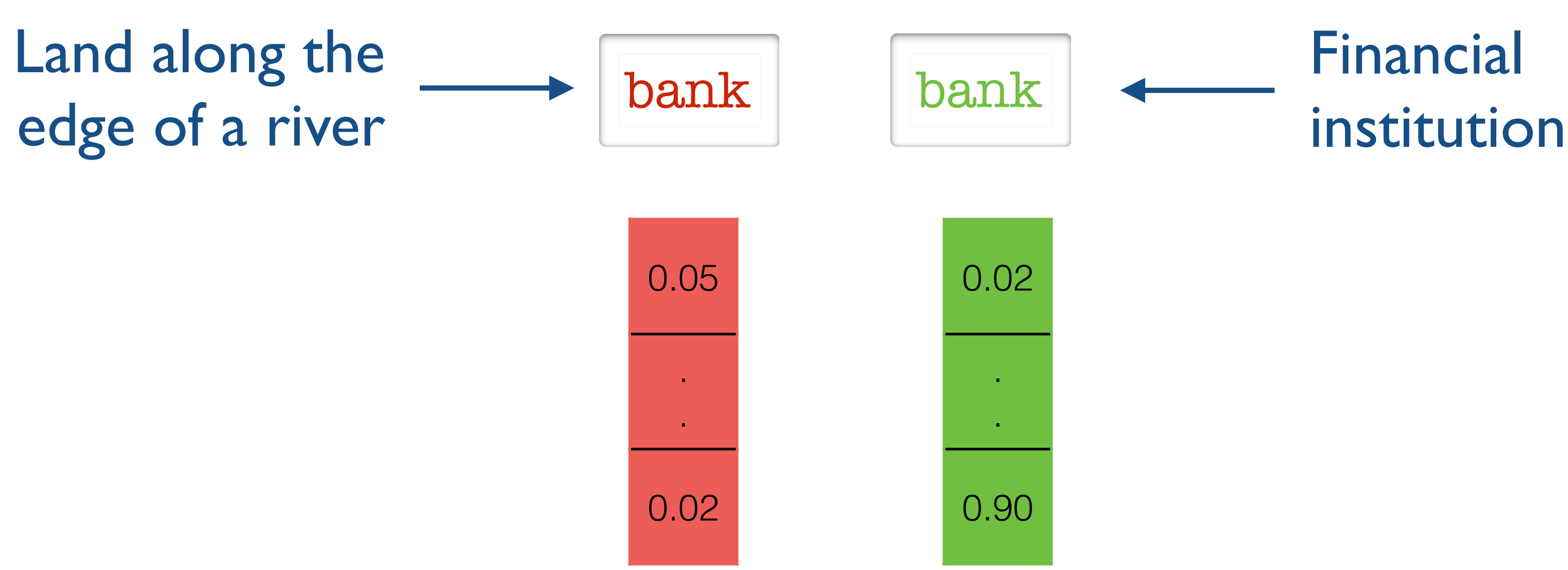
Dynamic Embeddings

- Representations for words in context (tokens)
- Each word w represented by a different vector each time it appears in a different context
- Also known as **contextual word embeddings**

Contextual Word Embeddings

Sense-specific and context-aware word representations

- A function of the entire sentence/sequence containing that word
- Different word vectors under different contexts
- Helps with polysemy!



Language Models

Language modeling — assign probabilities to word sequences and predicting upcoming words

- An important task in itself
- Plays a role in many NLP applications

Can also be used for learning contextual word embeddings

- ELMo
- BERT

Pre-trained language models

- Pre-training and fine-tuning paradigm
- Transfer learning

Stockholm University

BERT Overview

Bidirectional Encoder Representations from Transformers

- One of the biggest leaps in NLP

Applies bidirectional training of **Transformer** — an **attention** model — to language modeling

- In contrast to single-direction language models, where a text sequence is processed left to right or right to left (or combination)
- Bidirectionally trained language model leads to a deeper sense of language context and flow
- Introduced **masked language modeling** for bidirectional training

Transformer Encoder of BERT

BERT is based on the transformer

- An **attention mechanism**
- Learns contextual relations between words — contextual word embeddings

Transformer includes two separate mechanisms

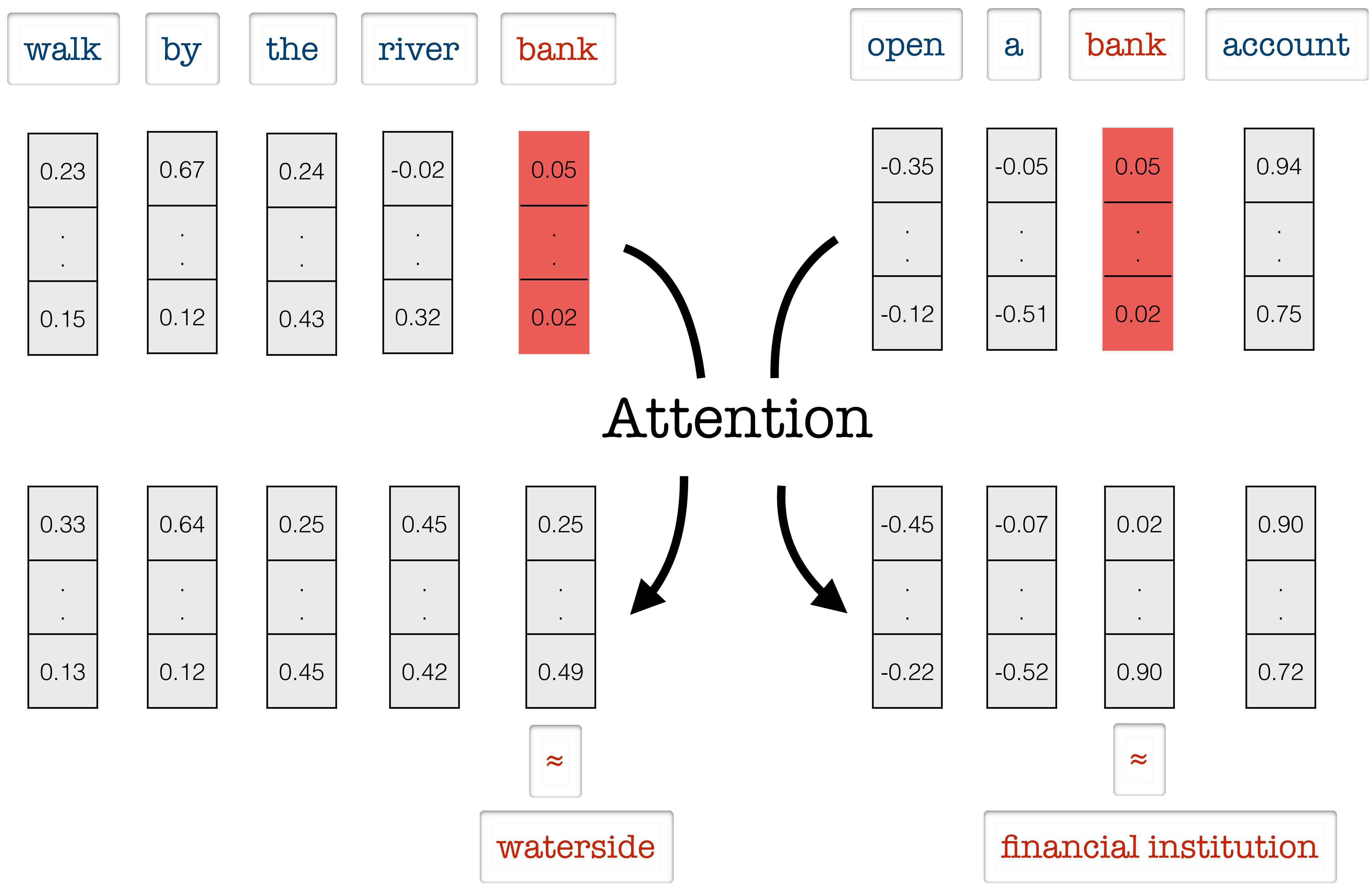
- An **encoder** that reads input text
- A **decoder** that produces the prediction for the task
- BERT goal is to create a language model — only encoder needed

Reads entire sequence of words at once

- Bidirectional (or non-directional)
- Learns contextual word embeddings

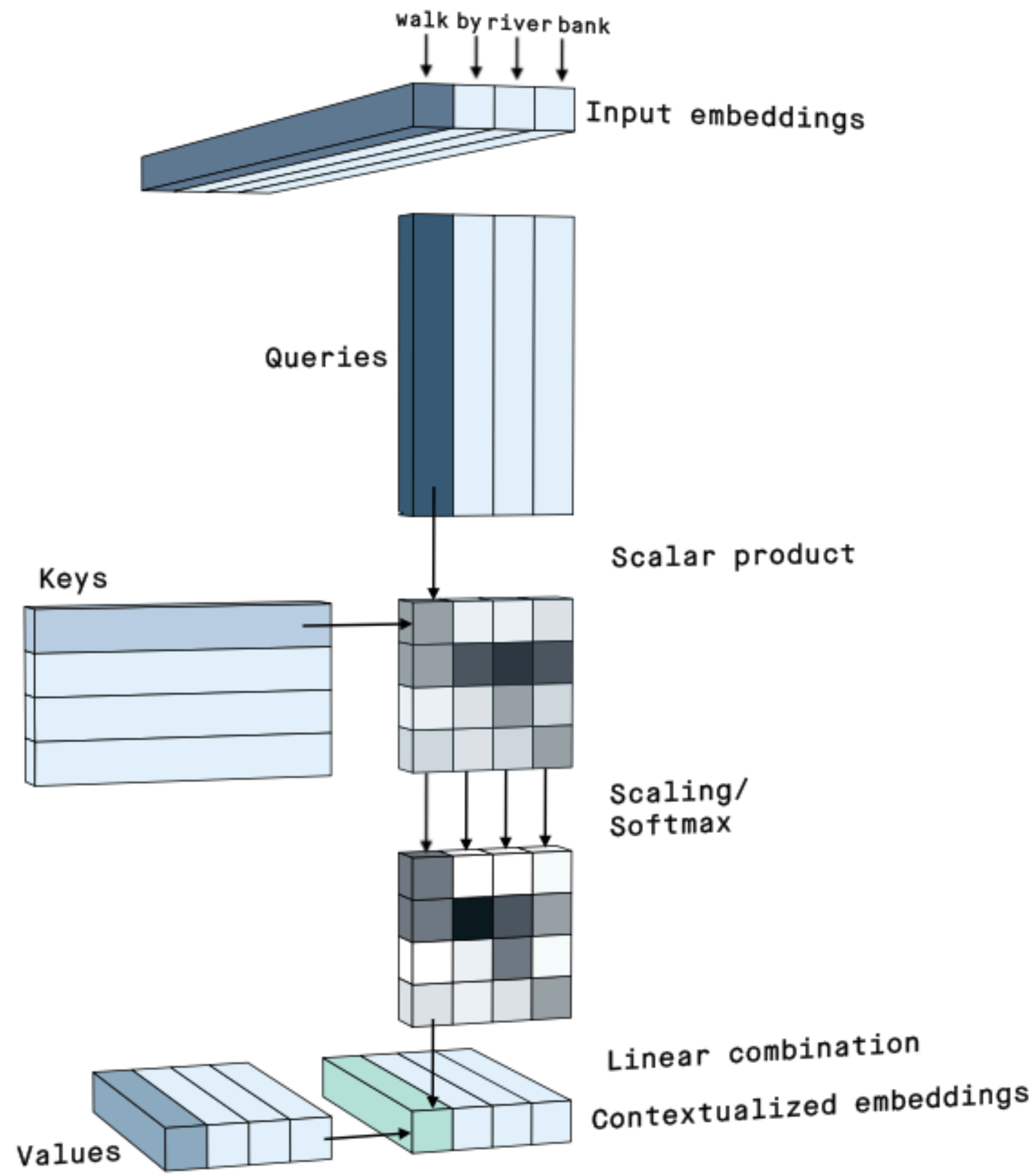
Stockholm University

Contextual Word Embeddings in BERT



Contextual Word Embeddings in BERT

1. Each token replaced with default, **context-independent embedding**
2. Calculate **scalar product** between pairs of embeddings in input sequence
   - High when similar
   - Low when dissimilar
3. Scalar values are passed to a **softmax** activation function column by column
   - Amplifies large values
   - Crushes low and negative values
4. **Contextualized embedding** created for each token through a linear combination of input embeddings in proportion to softmax results

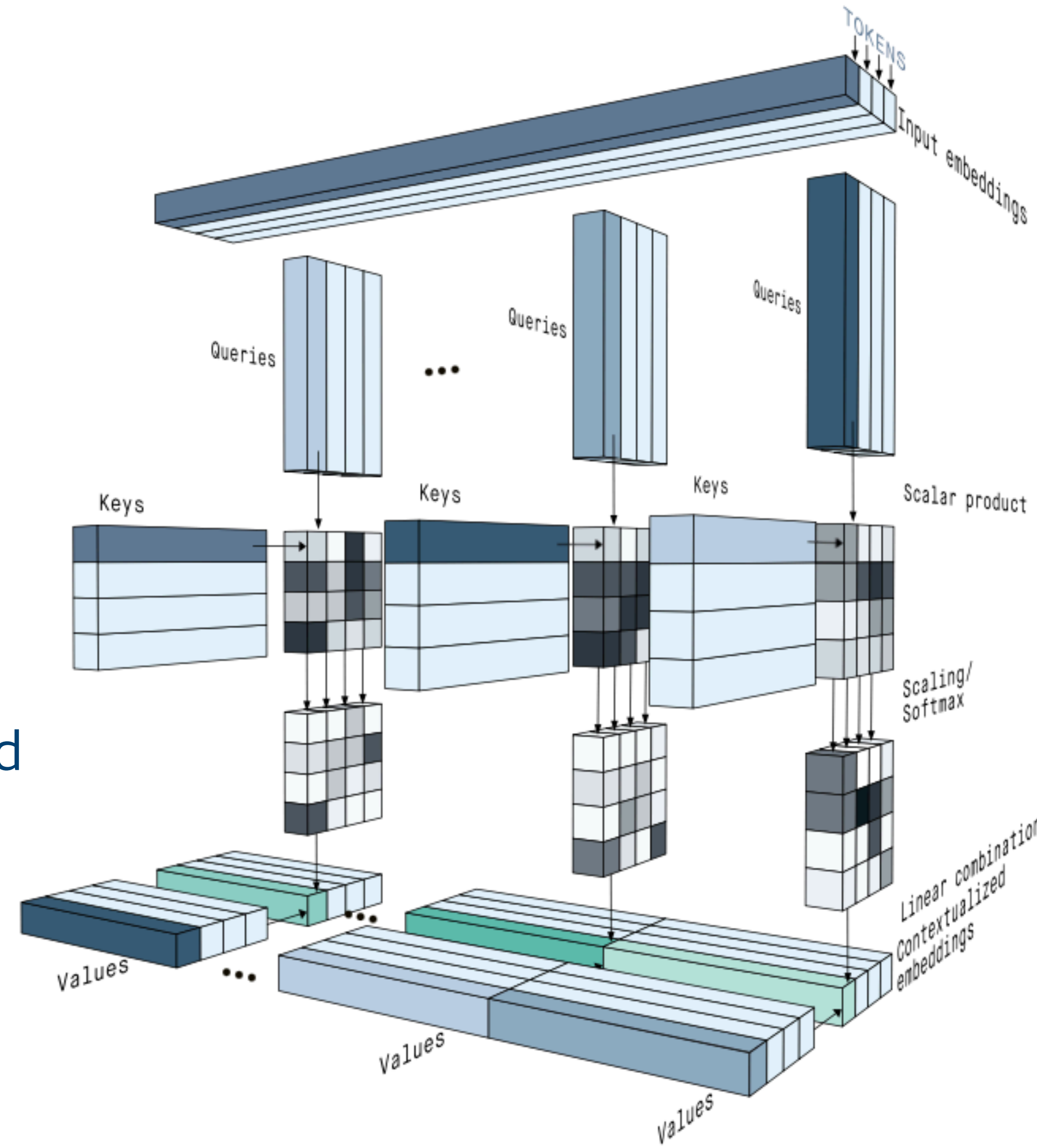


https://peltarion.com/blog/data-science/self-attention-video

Multi-head Attention and BERT

BERT uses multi-head attention

- Sequence of input embeddings projected using many different sets of **key**, **query**, and **value** projections
- Each focus on different types of relationships between tokens
- Specific contextualized embeddings
- Contextualized embeddings from different attention heads concatenated

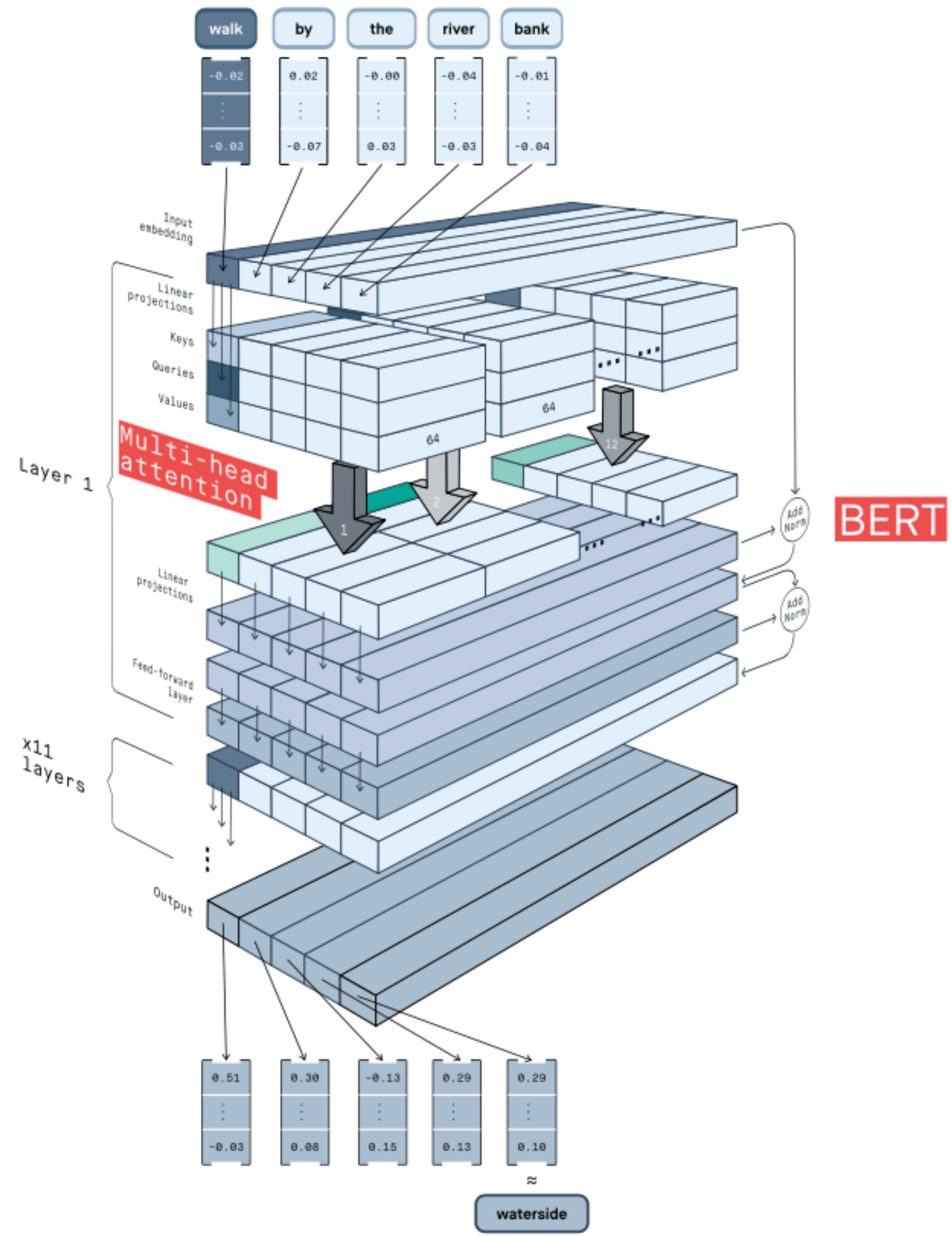


https://peltarion.com/blog/data-science/self-attention-video

BERT Architecture

Many layers of multi-head attention

- BERT encoder uses WordPiece embeddings of tokens
- Begins by adding them to **positional embeddings** — provides information about the order of tokens
- Additional linear projections, normalization and feed-forward layers add flexibility and stability



https://peltarion.com/blog/data-science/self-attention-video

BERT Pre-training

Self-supervision — what prediction task to use?

- Predict the next word in a sequence?
- “The child came home from ____”
- A directional approach limits context learning

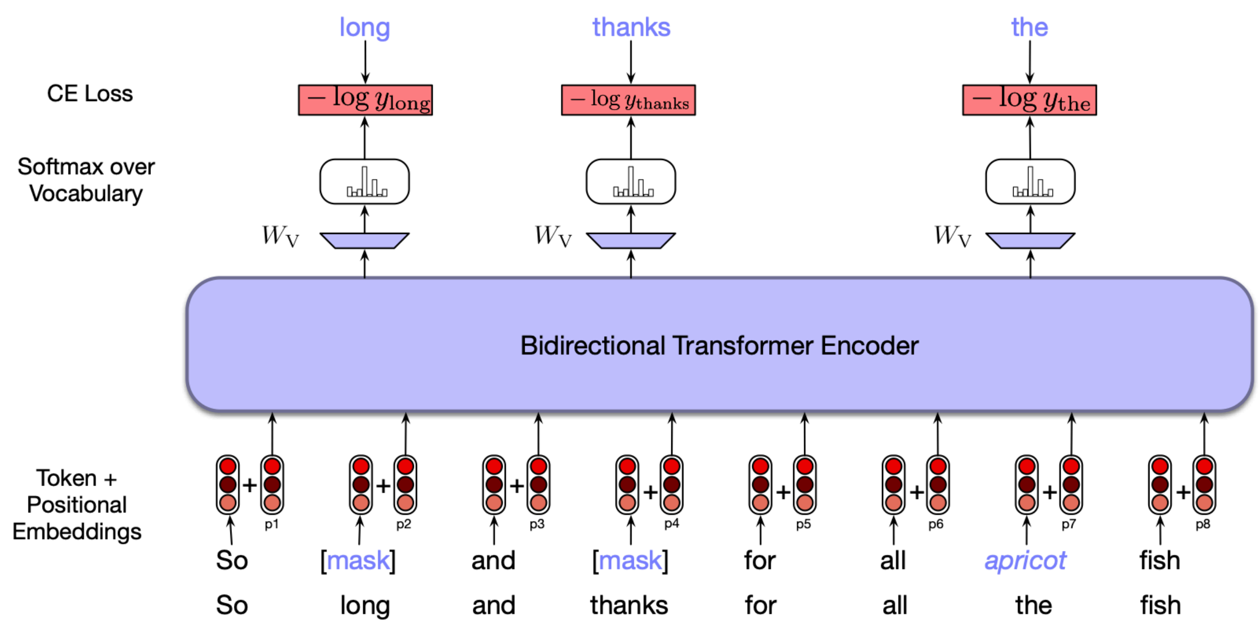
BERT uses two training strategies

- Masked language modeling
- Next sentence prediction

Masked Language Modeling

MLM — a Cloze task

- 15% of words replaced by [MASK] token — actually, of the 15%:
  - 80% replaced by [MASK] token
  - 10% by random word
  - 10% use the original word
- Predict original value of masked words based on the context
- BERT loss function takes into account only prediction of masked values



Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

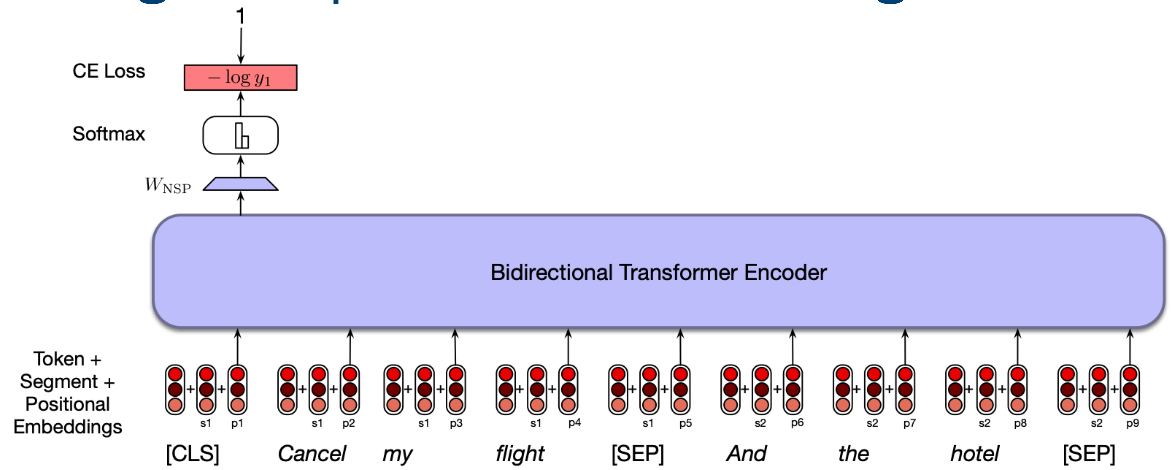
Next Sentence Prediction

NSP

- Model receives pairs of sentences
- Learns to predict if the second sentence is the subsequent sentence
- 50% original sentence sequence, 50% random sequences

Input is processed as follows

- A [CLS] token at beginning of first sentence
- A [SEP] token at the of each sentence
- Sentence embeddings and positional embeddings added to each token



Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

BERT Fine-tuning

Fine-tuning a pre-trained language model is the process of further training the model to perform some **downstream task**

- Relatively cheap compared to pre-training
- Relatively straightforward, only adding a small layer to the core model

BERT can be fine-tuned to perform a variety of NLP tasks

- **Classification**: add classification layer for the [CLS] token
- **QA**: learn two extra vectors marking beginning/end of answer
- **Named Entity Recognition**: feed output vector of each token into a classification layer that predicts the NER label (e.g. Person, Organization)

During fine-tuning, **most hyper-parameters stay the same** — BERT paper gives guidance on the hyper-parameters that require tuning

BERT Performance

Pre-training data

- BooksCorpus — 800M words
- English Wikipedia — 2,500M words

Two architectures

- $\text{BERT}_{\text{BASE}}$ — 12 layers, 110M parameters
- $\text{BERT}_{\text{LARGE}}$ — 24 layers, 340M parameters

Evaluated on 11 downstream tasks — GLUE test results shown here

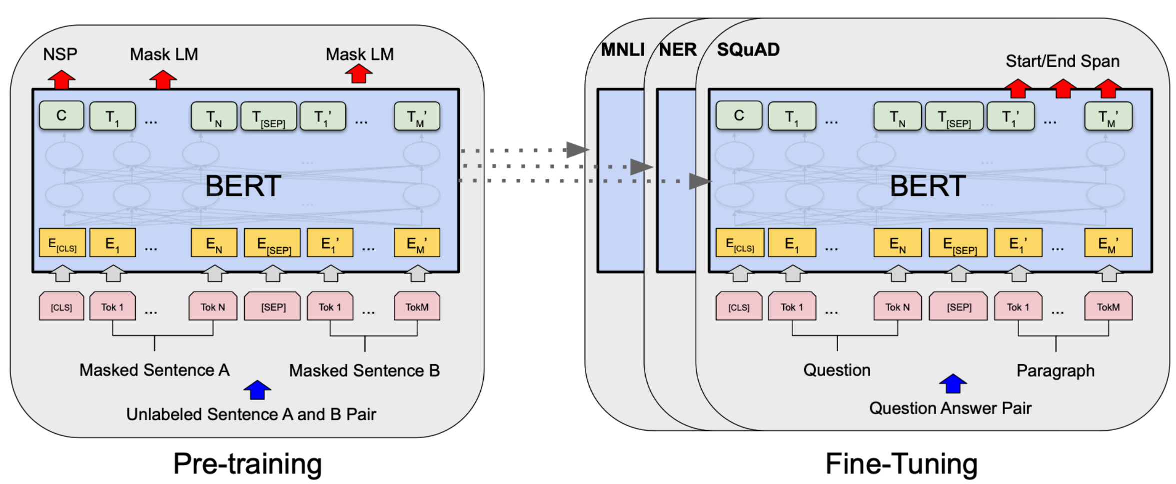
System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

Devlin et al., 2019.BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Paradigm of Pre-training & Fine-tuning

Transfer learning

- Value shown in computer vision
- Has also led to a **paradigm shift in NLP**
- **Pre-training phase**: learns language model with rich representations of word meaning
- **Fine-tuning phase**: enables the model to learn the requirements of a downstream task
- Fine-tuning relatively **cheap** in terms of **computation** and **data**



Devlin et al., 2019.BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Visualizing Embeddings

How can we visualize embeddings in high-dimensional semantic space?

- Important for understanding, applying and improving models of word meaning
- How can we visualize, e.g., a 100-dimensional vector?

Nearest neighbors

- List the most similar words according to cosine similarity

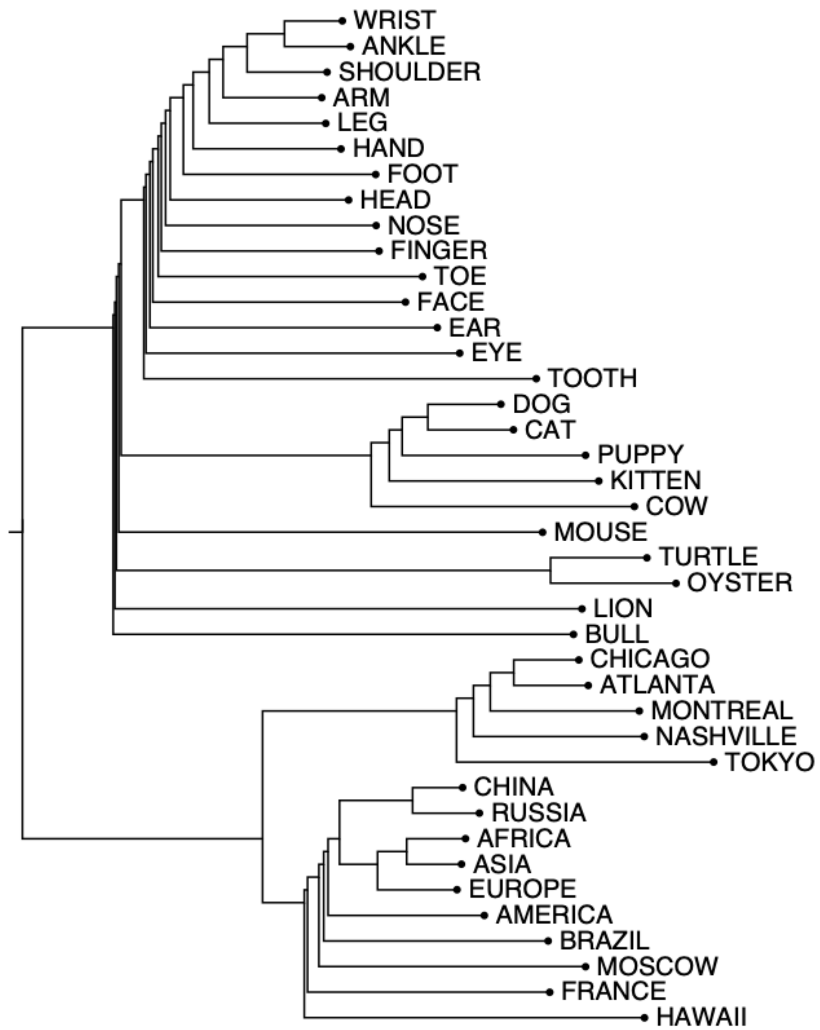
Example **frog** :

- frogs, toad, Vitoria, leptodactylidae, rana, lizard, eleutherodactylus

Visualizing Embeddings

Hierarchical clustering

- Hierarchical representation of which words are similar to others in embedding space
- Example: hierarchical clustering of embedding vectors for nouns



Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

Visualizing Embeddings

Projects high-dimensional space down to two dimensions

- Using a projection method called **t-SNE**

Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

Semantic Properties of Embeddings

Different types of similarity or association

- Affected by the **size of the context window**
  - Choice depends on goals of the representation
- Smaller context windows yields more **syntactic representations**
  - Similar words tend to be semantically similar words with same PoS
- Larger context windows yields more **topical representations**
  - Similar words tend to be topically related but not similar words
- **First-order co-occurrence**
  - Words typically near each other (‘wrote’ and ‘book’)
  - **Syntagmatic association**
- **Second-order co-occurrence**
  - Two words with similar neighbors (‘wrote’ and ‘said’)
  - **Paradigmatic association**

Stockholm University

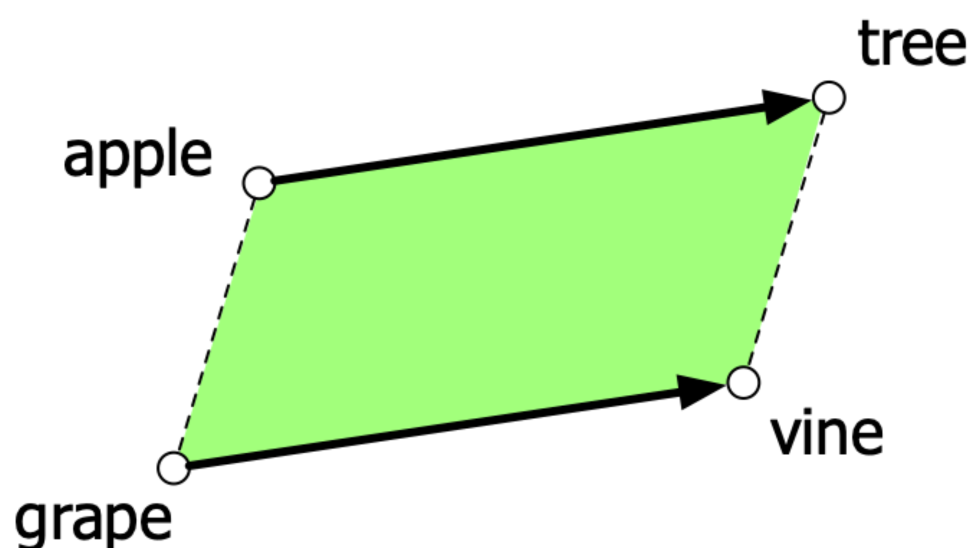
Semantic Properties of Embeddings

Analogy / Relational Similarity

- Embeddings can capture relational meanings

vector(“King”) - vector(“Man”) + vector(“Woman”) ≈ vector(“Queen”)

vector(“Paris”) - vector(“France”) + vector(“Italy”) ≈ vector(“Rome”)



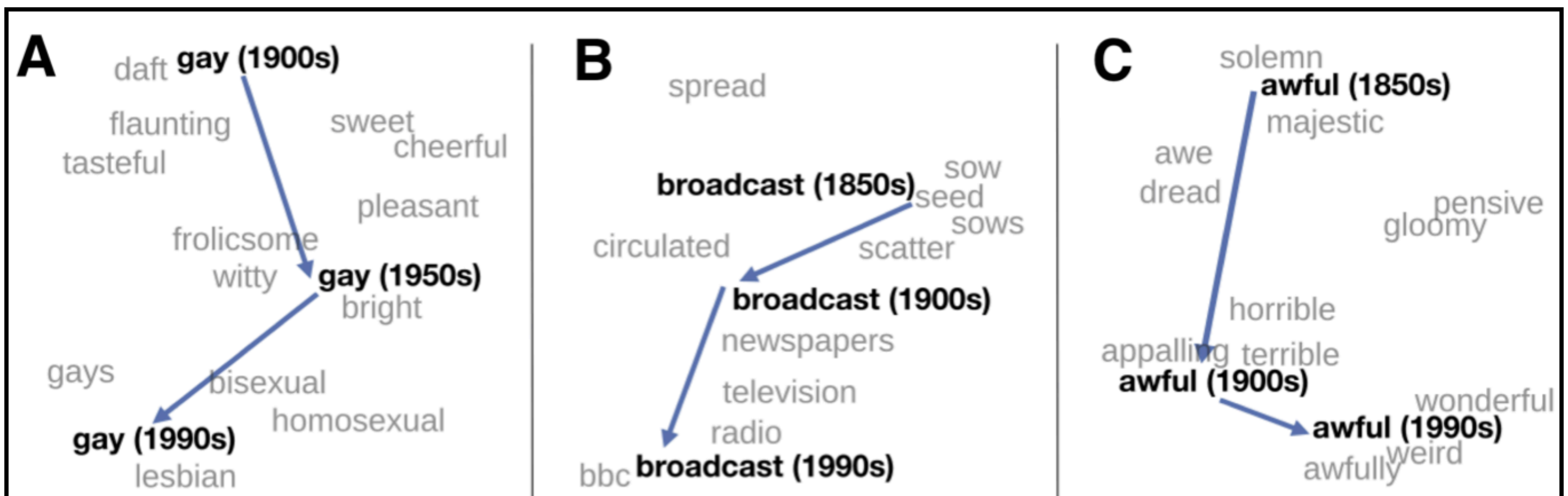
Jurafsky &. Martin, 2022. Speech and Language Processing.

Stockholm University

Semantic Shift

Embeddings and historical semantics

- Visualizing changes in meaning in English words over last two centuries
- Computed by building separate embedding spaces for each decade using historical corpora (here: Google n-grams and the Corpus of Historical American English)



Jurafsky &. Martin, 2022. Speech and Language Processing.

Bias in Embeddings

Allocational harm

Word embeddings reproduce implicit biases and stereotypes

- **Gender stereotypes** in word2vec embeddings trained on news text
  - Closest occupation to ‘man’ – ‘computer programmer’ + ‘woman’ is ‘homemaker’
  - Analogy: ‘father’ is to ‘doctor’ as ‘mother’ is to ‘nurse’
- Could result in **allocational harm**: a system allocates resources unfairly to different groups

Bias in Embeddings

Bias amplification

- Embeddings do not just reflect input statistics, they also **amplify bias**
- Gendered terms more gendered in embedding space than in input text statistics
- Biases more exaggerated than in actual labor employment statistics

Bias in Embeddings

Representational harm

Embeddings also encode implicit associations

- Implicit Association Test: measures people's associations between concepts and attributes by differences in latency when labeling words in various categories
- People in the U.S. shown to associate:
  - African-American and old people's names with unpleasant words
  - Male names more with mathematics; female names with the arts
- Findings replicated using **GloVe vectors** and **cosine similarity** instead of human latency
- **Representational harm** caused by a system demeaning or ignoring social groups

Stockholm University

Bias in Embeddings

Debiasing

- Recent research focuses on ways to remove biases from embeddings
- Developing a transformation of embedding space that removes gender stereotypes but preserves definitional gender
- Changing the training procedure to remove biases
- Shown to reduce but not eliminate bias in embeddings

Evaluating Vector Semantic Models

Extrinsic evaluations

- Most important evaluation metric
- Use word vectors in an NLP task and evaluate model performance compared to baseline model
- Language models fine-tuned and evaluated on downstream tasks

Intrinsic evaluations

- Most common metric is performance on **similarity tasks**, i.e. computing correlation between algorithm's word similarity score and ratings assigned by humans
- Similarity tasks that include context are more realistic
- Analogy tasks

Summary

Lexical Semantics & Distributional Hypothesis

- Deriving word meaning based on context
- Representing word meaning in high-dimensional vector space

Models of Distributional Semantics

- Sparse vs. dense word vectors
- Static vs. dynamic word vectors
- word2vec: skip-gram with negative sampling

Language Models

- BERT for learning contextual word embeddings
- Pre-training & fine-tuning paradigm

Other Aspects of Embeddings

- Visualization, biases and evaluation

Stockholm University