UNIT 5
Semantics

What do words mean?

N-gram or text classification methods we've seen so far

- Words are just strings (or indices w_i in a vocabulary list)
- That's not very satisfactory!

Introductory logic classes:

- The meaning of "dog" is DOG; cat is CAT

$$\forall x\ \mathrm{DOG}(x) \longrightarrow \mathrm{MAMMAL}(x)$$

Old linguistics joke by Barbara Partee in 1967:

- Q: What's the meaning of life?
- A: LIFE

That seems hardly better!

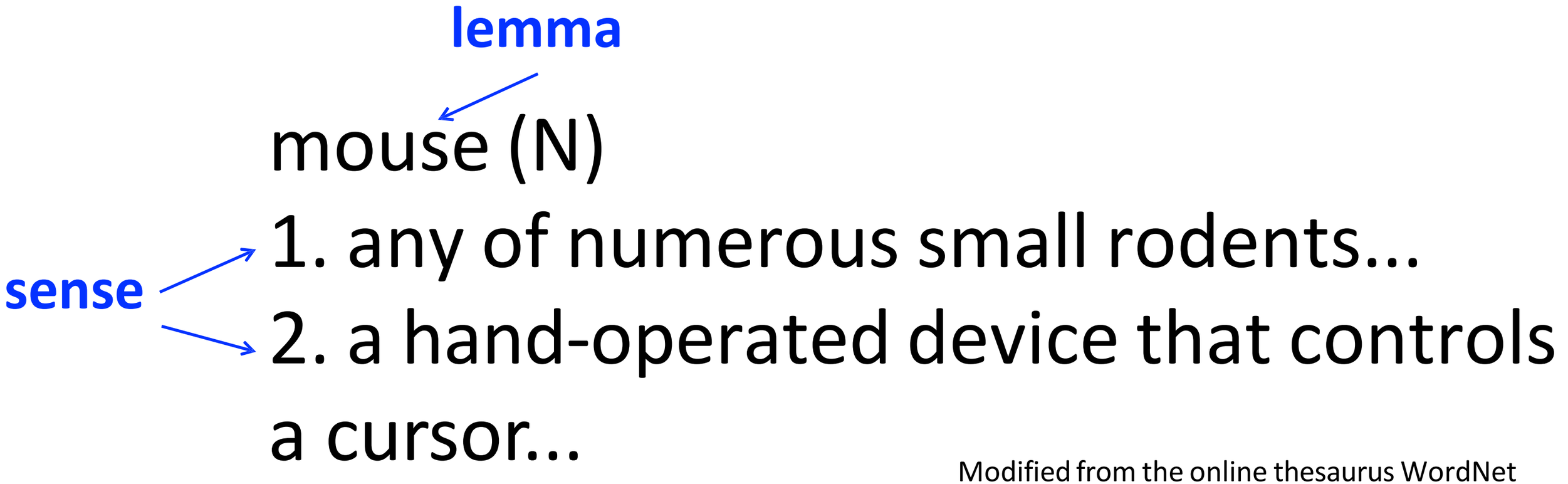
Desiderata

What should a theory of word meaning do for us?

Let's look at some desiderata

From lexical semantics, the linguistic study of word meaning

Lemmas and senses



Modified from the online thesaurus WordNet

A sense or "concept" is the meaning component of a word
Lemmas can be polysemous (have multiple senses)

Relations between senses: Synonymy

Synonyms have the same meaning in some or all contexts.

- filbert / hazelnut
- couch / sofa
- big / large
- automobile / car
- vomit / throw up
- water / H_20

Relations between senses: Synonymy

Note that there are probably no examples of perfect synonymy.

- Even if many aspects of meaning are identical
- Still may differ based on politeness, slang, register, genre, etc.

Relation: Synonymy?

water/H_20

"H_20" in a surfing guide?

big/large

my big sister != my large sister

The Linguistic Principle of Contrast

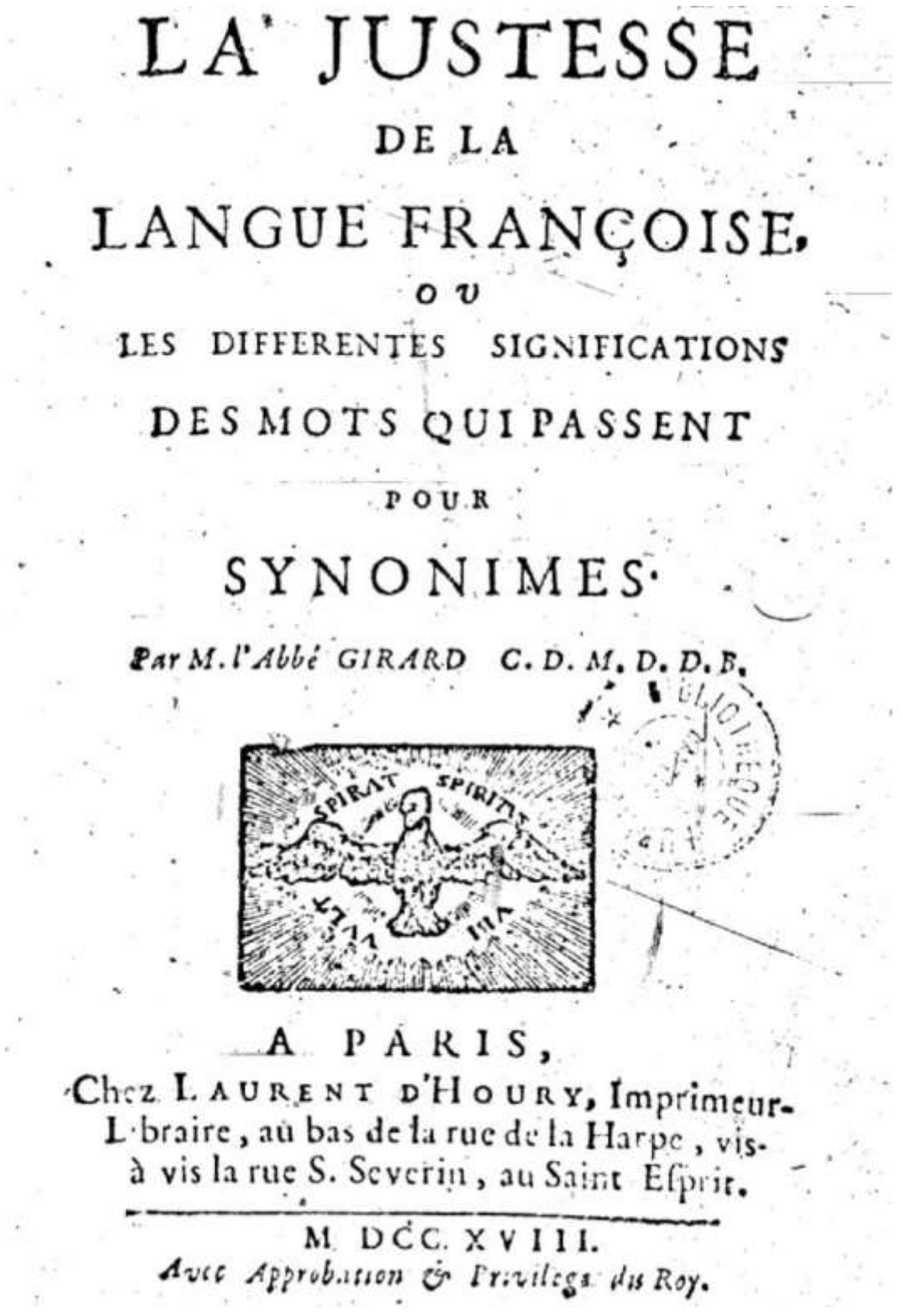
Difference in form → difference in meaning

Abbé Gabriel Girard 1718

Re: "exact" synonyms

"je ne crois pas qu'il y ait de mot ſynonime dans aucune Langue."

[I do not believe that there is a synonymous word in any language]


LA JUSTESSE
DE LA
LANGUE FRANÇOISE,
OU
LES DIFFERENTES SIGNIFICATIONS
DES MOTS QUI PASSENT
POUR
SYNONIMES.
Par M. l'Abbé GIRARD C. D. M. D. D. B.

A PARIS,
Chez LAURENT D'HOURY, Imprimeur-Libraire, au bas de la rue de la Harpe, vis-à-vis la rue S. Severin, au Saint Eſprit.
M. DCC. XVIII.
Avec Approbation & Privilege du Roy.

Thanks to Mark Aronoff!

Relation: Similarity

Words with similar meanings. Not synonyms, but sharing some element of meaning

```
car, bicycle
cow, horse
```

Ask humans how similar 2 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 can be related in any way, perhaps via a semantic frame or field

- `coffee, tea:` **similar**
- `coffee, cup:` **related**, not similar

Semantic field

Words that

- cover a particular semantic domain
- bear structured relations with each other.

hospitals

surgeon, scalpel, nurse, anaesthetic, hospital

restaurants

waiter, menu, plate, food, menu, chef

houses

door, roof, kitchen, family, bed

Relation: Antonymy

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

Otherwise, they are very similar!

`dark/light` `short/long` `fast/slow` `rise/fall`
`hot/cold` `up/down` `in/out`

More formally: antonyms can

- define a binary opposition or be at opposite ends of a scale
  - `long/short, fast/slow`
- Be *reversives*:
  - `rise/fall, up/down`

Connotation (sentiment)

- Words have **affective** meanings
  - Positive connotations (*happy*)
  - Negative connotations (*sad*)
- Connotations can be subtle:
  - Positive connotation: *copy, replica, reproduction*
  - Negative connotation: *fake, knockoff, forgery*
- Evaluation (sentiment!)
  - Positive evaluation (*great*, *love*)
  - Negative evaluation (*terrible*, *hate*)

Connotation

Osgood et al. (1957)

Words seem to vary along 3 affective dimensions:

- **valence**: the pleasantness of the stimulus
- **arousal**: the intensity of emotion provoked by the stimulus
- **dominance**: the degree of control exerted by the stimulus

	Word	Score		Word	Score
Valence	love	1.000		toxic	0.008
	happy	1.000		nightmare	0.005
Arousal	elated	0.960		mellow	0.069
	frenzy	0.965		napping	0.046
Dominance	powerful	0.991		weak	0.045
	leadership	0.983		empty	0.081

Values from NRC VAD Lexicon (Mohammad 2018)

Concepts or word senses

- Have a complex many-to-many association with **words** (homonymy, multiple senses)

Have relations with each other

- Synonymy
- Antonymy
- Similarity
- Relatedness
- Connotation

Computational models of word meaning

Can we build a theory of how to represent word meaning, that accounts for at least some of the desiderata?

We'll introduce **vector semantics**

The standard model in language processing!

Handles many of our goals!

Ludwig Wittgenstein

PI #43:

"The meaning of a word is its use in the language"

Let's define words by their usages

One way to define "usage":

words are defined by their environments (the words around them)

Zellig Harris (1954):

If A and B have almost identical environments we say that they are synonyms.

What does recent English borrowing *ongchoi* mean?

Suppose you see these sentences:

- Ong choi is delicious **sautéed with garlic**.
- Ong choi is superb **over rice**
- Ong choi **leaves** with salty sauces

And you've also seen these:

- …spinach **sautéed with garlic over rice**
- Chard stems and **leaves** are **delicious**
- Collard greens and other **salty** leafy greens

Conclusion:

- Ongchoi is a leafy green like spinach, chard, or collard greens
  - We could conclude this based on words like "leaves" and "delicious" and "sauteed"

Ongchoi: *Ipomoea aquatica "Water Spinach"*

空心菜
kangkong
rau muống
…

Yamaguchi, Wikimedia Commons, public domain

Idea 1: Defining meaning by linguistic distribution

Let's define the meaning of a word by its distribution in language use, meaning its neighboring words or grammatical environments.

Idea 2: Meaning as a point in space (Osgood et al. 1957)

3 affective dimensions for a word

- **valence**: pleasantness
- **arousal**: intensity of emotion
- **dominance**: the degree of control exerted

	Word	Score		Word	Score
Valence	love	1.000		toxic	0.008
	happy	1.000		nightmare	0.005
Arousal	elated	0.960		mellow	0.069
	frenzy	0.965		napping	0.046
Dominance	powerful	0.991		weak	0.045
	leadership	0.983		empty	0.081

NRC VAD Lexicon
(Mohammad 2018)

Hence the connotation of a word is a vector in 3-space

Idea 1: Defining meaning by linguistic distribution

Idea 2: Meaning as a point in multidimensional space

Defining meaning as a point in space based on distribution

Each word = a vector (not just "good" or "w_{45}")

Similar words are "**nearby in semantic space**"

We build this space automatically by seeing which words are **nearby in text**

We define meaning of a word as a vector

Called an "embedding" because it's embedded into a space (see textbook)

The standard way to represent meaning in NLP

Every modern NLP algorithm uses embeddings as the representation of word meaning

Fine-grained model of meaning for similarity

Intuition: why vectors?

Consider sentiment analysis:

- With **words**, a feature is a word identity
  - Feature 5: 'The previous word was "terrible"'
  - requires **exact same word** to be in training and test
- With **embeddings**:
  - Feature is a word vector
  - 'The previous word was vector [35,22,17…]
  - Now in the test set we might see a similar vector [34,21,14]
  - We can generalize to **similar but unseen** words!!!

We'll discuss 2 kinds of embeddings

tf-idf

- Information Retrieval workhorse!
- A common baseline model
- **Sparse** vectors
- Words are represented by (a simple function of) the **counts** of nearby words

Word2vec

- **Dense** vectors
- Representation is created by training a classifier to **predict** whether a word is likely to appear nearby
- Later we'll discuss extensions called **contextual embeddings**

From now on:

Computing with meaning representations instead of string representations

C ⇧ @ Â (| , ó | 7 ÿ C	Nets are for fish;
	Once you get the fish, you can forget the net.
p ⇧ @ Â (▬ , ó ▬ 7 ÿ p	Words are for meaning;
	Once you get the meaning, you can forget the words

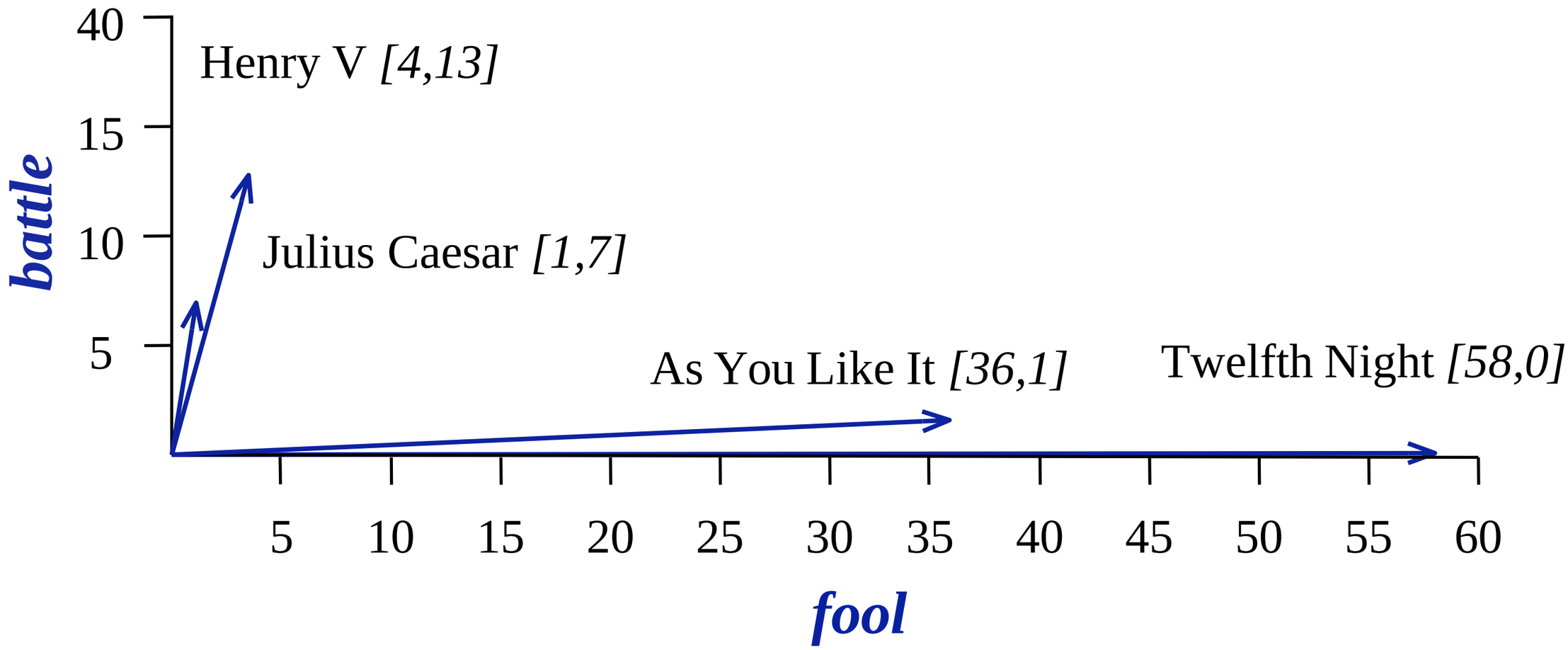
ÑP(Zhuangzi), Chapter 26

Term-document matrix

Each document is represented by a vector of words

	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

Visualizing document vectors



Vectors are the basis of information retrieval

	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

Vectors are similar for the two comedies

But comedies are different than the other two

Comedies have more *fools* and *wit* and fewer *battles*.

Idea for word meaning: Words can be vectors too!!!

	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

battle is "the kind of word that occurs in Julius Caesar and Henry V"

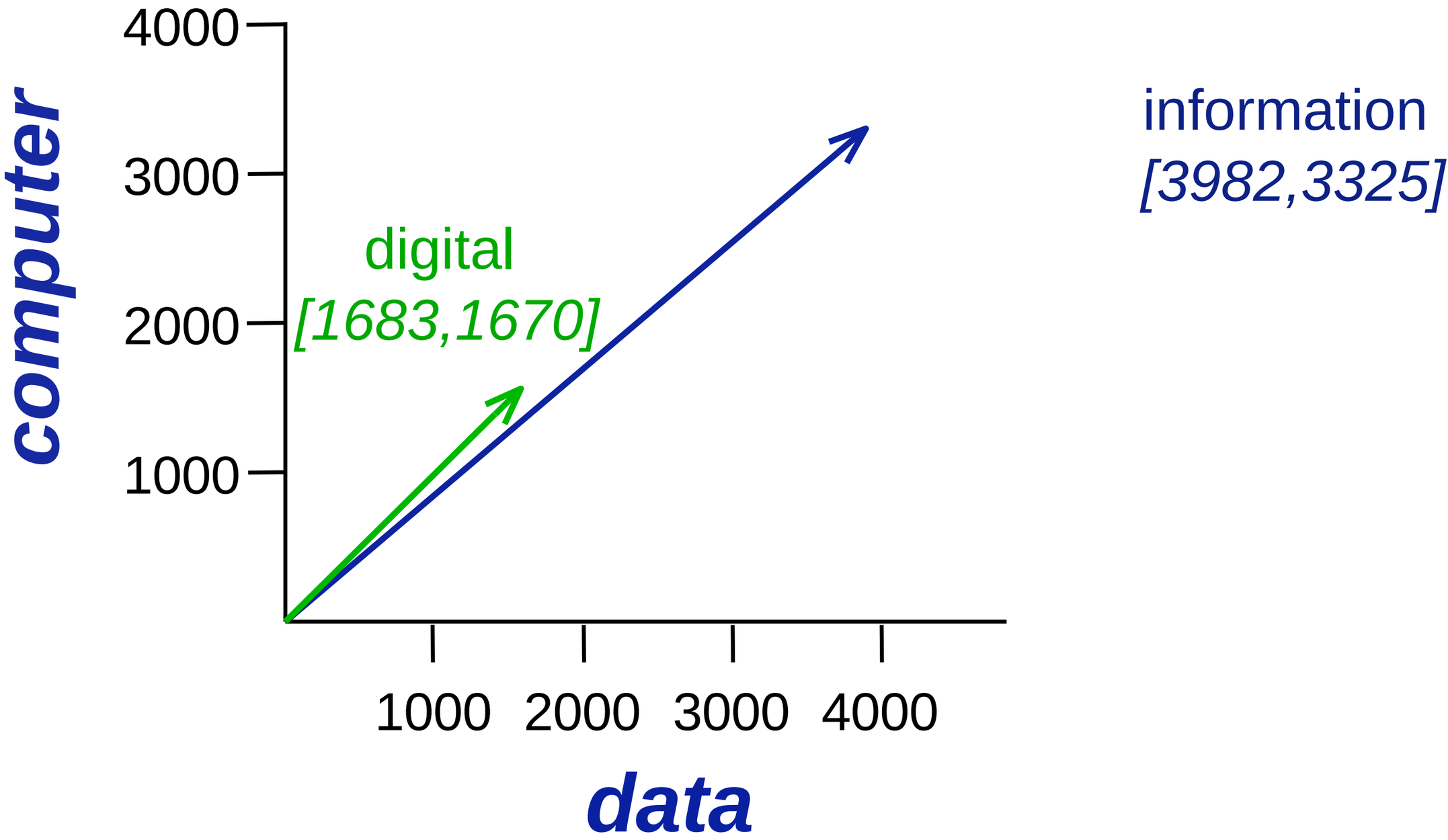
fool is "the kind of word that occurs in comedies, especially Twelfth Night"

More common: word-word matrix (or "term-context matrix")

Two **words** are similar in meaning if their context vectors are similar

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

	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	...


4000
3000
2000
1000
computer
digital
[1683,1670]
information
[3982,3325]
1000
2000
3000
4000
data

Computing word similarity: Dot product and cosine

The dot product between two vectors is a scalar:

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

The dot product tends to be high when the two vectors have large values in the same dimensions

Dot product can thus be a useful similarity metric between vectors

Problem with raw dot-product

Dot product favors long vectors

Dot product is higher if a vector is longer (has higher values in many dimension)

Vector length:

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

Frequent words (of, the, you) have long vectors (since they occur many times with other words).

So dot product overly favors frequent words

Alternative: cosine for computing word similarity

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

Based on the definition of the dot product between two vectors a and b

$$\mathbf{a} \cdot \mathbf{b} = |\mathbf{a}||\mathbf{b}| \cos q$$

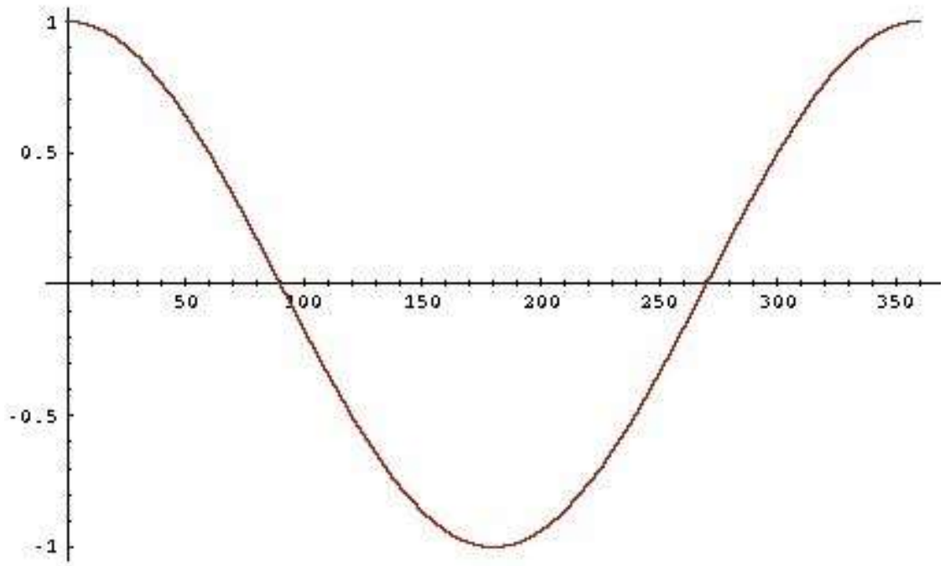
$$\frac{\mathbf{a} \cdot \mathbf{b}}{|\mathbf{a}||\mathbf{b}|} = \cos q$$

Cosine as a similarity metric

-1: vectors point in opposite directions

+1: vectors point in same directions

0: vectors are orthogonal



But since raw frequency values are non-negative, the cosine for term-term matrix vectors ranges from 0–1

Cosine examples

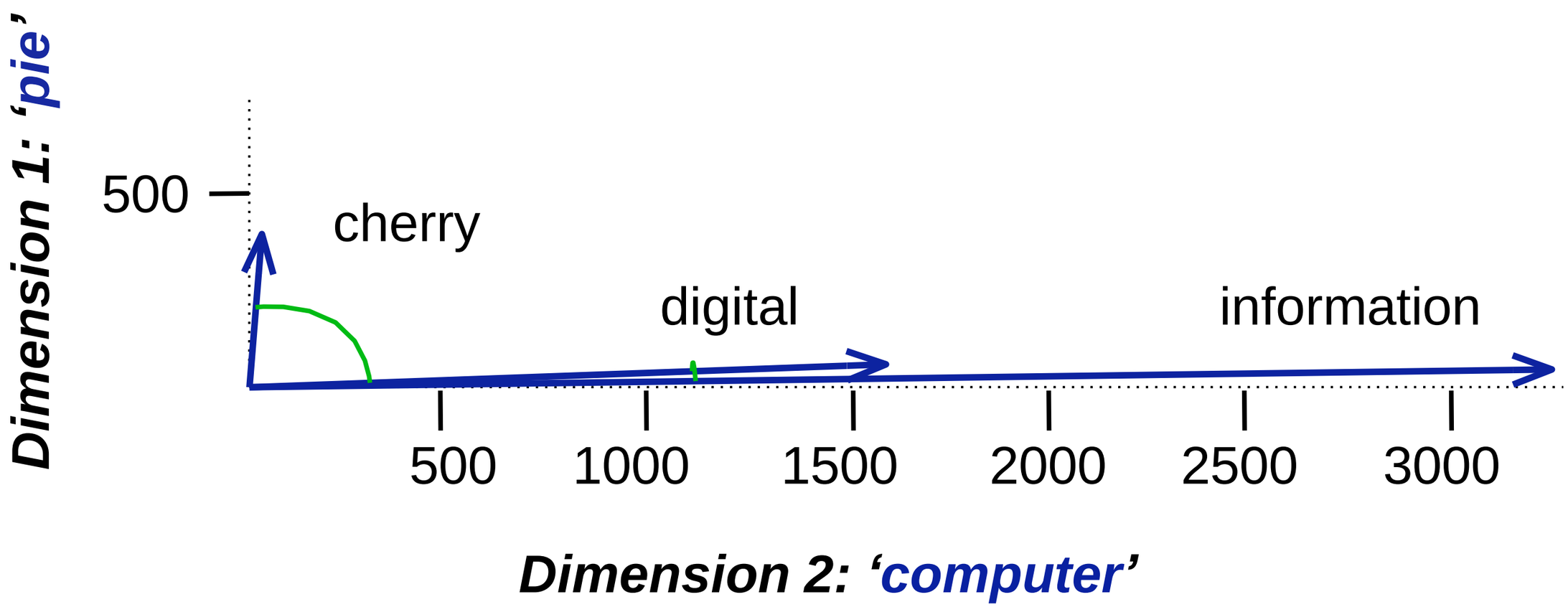
$$\cos(\vec{v},\vec{w}) = \frac{\vec{v}\bullet\vec{w}}{|\vec{v}||\vec{w}|} = \frac{\vec{v}}{|\vec{v}|}\bullet\frac{\vec{w}}{|\vec{w}|} = \frac{\sum_{i=1}^{N} v_i w_i}{\sqrt{\sum_{i=1}^{N} v_i^2}\sqrt{\sum_{i=1}^{N} w_i^2}}$$

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

$$\cos(\text{cherry, information}) = \frac{442*5 + 8*3982 + 2*3325}{\sqrt{442^2 + 8^2 + 2^2}\sqrt{5^2 + 3982^2 + 3325^2}} = .017$$

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

Visualizing cosines (well, angles)



But raw frequency is a bad representation

- The co-occurrence matrices we have seen represent each cell by word frequencies.
- Frequency is clearly useful; if *sugar* appears a lot near *apricot*, that's useful information.
- But overly frequent words like *the*, *it,* or *they* are not very informative about the context
- It's a paradox! How can we balance these two conflicting constraints?

Two common solutions for word weighting

tf-idf: tf-idf value for word t in document d:

$$w_{t,d} = \text{tf}_{t,d} \rightarrow \text{idf}_t$$

Words like "the" or "it" have very low idf

PMI: (Pointwise mutual information)

- $\text{PMI}(w_1, w_2) = log \frac{p(w_1,w_2)}{p(w_1)p(w_2)}$

See if words like "good" appear more often with "great" than we would expect by chance

Term frequency (tf)

$\text{tf}_{t,d} = \text{count}(t,d)$

Instead of using raw count, we squash a bit:

$\text{tf}_{t,d} = \log_{10}(\text{count}(t,d)+1)$

Document frequency (df)

df_t *is* the number of documents *t* occurs in.

(note this is not collection frequency: total count across all documents)

"*Romeo*" is very distinctive for one Shakespeare play:

	Collection Frequency	Document Frequency
Romeo	113	1
action	113	31

Inverse document frequency (idf)

$$\text{idf}_t = \log_{10} \left(\frac{N}{\text{df}_t} \right)$$

N is the total number of documents in the collection

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.246
wit	34	0.037
fool	36	0.012
good	37	0
sweet	37	0

What is a document?

Could be a play or a Wikipedia article

But for the purposes of tf-idf, documents can be **anything**; we often call each paragraph a document!

Final tf-idf weighted value for a word

$$w_{t,d} = \text{tf}_{t,d} \rightarrow \text{idf}_t$$

Raw counts:

	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

tf-idf:

	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

Pointwise Mutual Information

Pointwise mutual information:

Do events x and y co-occur more than if they were independent?

$$\text{PMI}(X,Y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

PMI between two words: (Church & Hanks 1989)

Do words x and y co-occur more than if they were independent?

$$\text{PMI}(word_1, word_2) = \log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}$$

Positive Pointwise Mutual Information

- PMI ranges from $-\infty$ to $+\infty$
- But the negative values are problematic
  - Things are co-occurring **less than** we expect by chance
  - Unreliable without enormous corpora
    - Imagine w1 and w2 whose probability is each 10^{-6}
    - Hard to be sure p(w1,w2) is significantly different than 10^{-12}
    - Plus it's not clear people are good at "unrelatedness"
- So we just replace negative PMI values by 0
- Positive PMI (**PPMI**) between word1 and word2:

$$\text{PPMI}(word_1, word_2) = \max\left(\log_2 \frac{P(word_1, word_2)}{P(word_1)P(word_2)}, 0\right)$$

Computing PPMI on a term-context matrix

Matrix F with W rows (words) and C columns (contexts)

f_{ij} is # of times w_i occurs in context c_j

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{i*} = \frac{\sum_{j=1}^{C} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}} \qquad p_{*j} = \frac{\sum_{i=1}^{W} f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}} \qquad ppmi_{ij} = \begin{cases} pmi_{ij} & \text{if } pmi_{ij} > 0 \\ 0 & \text{otherwise} \end{cases}$$

$$p_{ij} = \frac{f_{ij}}{\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij}}$$

	computer	data	result	pie	sugar	count(w)
cherry	2	8	9	442	25	486
strawberry	0	0	1	60	19	80
digital	1670	1683	85	5	4	3447
information	3325	3982	378	5	13	7703
count(context)	4997	5673	473	512	61	11716

p(w=information,c=data) = 3982/111716 = .3399

p(w=information) = 7703/11716 = .6575

p(c=data) = 5673/11716 = .4842

$$p(w_i) = \frac{\sum_{j=1}^{C} f_{ij}}{N} \qquad p(c_j) = \frac{\sum_{i=1}^{W} f_{ij}}{N}$$

	p(w,context)					p(w)
	computer	**data**	**result**	**pie**	**sugar**	**p(w)**
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

$$pmi_{ij} = \log_2 \frac{p_{ij}}{p_{i*}p_{*j}}$$

	p(w,context)					p(w)
	computer	**data**	**result**	**pie**	**sugar**	**p(w)**
cherry	0.0002	0.0007	0.0008	0.0377	0.0021	0.0415
strawberry	0.0000	0.0000	0.0001	0.0051	0.0016	0.0068
digital	0.1425	0.1436	0.0073	0.0004	0.0003	0.2942
information	0.2838	0.3399	0.0323	0.0004	0.0011	0.6575
p(context)	0.4265	0.4842	0.0404	0.0437	0.0052	

pmi(information,data) = $\log_2$ (.3399 / (.6575*.4842)) = .0944

Resulting PPMI matrix (negatives replaced by 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

Weighting PMI

PMI is biased toward infrequent events

- Very rare words have very high PMI values

Two solutions:

- Give rare words slightly higher probabilities
- Use add-one smoothing (which has a similar effect)

Weighting PMI: Giving rare context words slightly higher probability

Raise the context probabilities to $\alpha = 0.75$:

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

$$P_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha}$$

This helps because $P_\alpha(c) > P(c)$ for rare *c*

Consider two events, P(a) = .99 and P(b)=.01

$$P_\alpha(a) = \frac{.99^{.75}}{.99^{.75}+.01^{.75}} = .97 \quad P_\alpha(b) = \frac{.01^{.75}}{.01^{.75}+.01^{.75}} = .03$$

Sparse versus dense vectors

tf-idf (or PMI) vectors are

- **long** (length |V|= 20,000 to 50,000)
- **sparse** (most elements are zero)

Alternative: learn vectors which are

- **short** (length 50-1000)
- **dense** (most elements are non-zero)

Sparse versus dense vectors

Why dense vectors?

- Short vectors may be easier to use as **features** in machine learning (fewer weights to tune)
- Dense vectors may **generalize** better than explicit counts
- Dense vectors may do better at capturing synonymy:
  - *car* and *automobile* are synonyms; but are distinct dimensions
    - a word with *car* as a neighbor and a word with *automobile* as a neighbor should be similar, but aren't
- **In practice, they work better**

Common methods for getting short dense vectors

"Neural Language Model"-inspired models

- Word2vec (skipgram, CBOW), GloVe

Singular Value Decomposition (SVD)

- A special case of this is called LSA – Latent Semantic Analysis

Alternative to these "static embeddings":

- Contextual Embeddings (ELMo, BERT)
- Compute distinct embeddings for a word in its context
- Separate embeddings for each token of a word

The kinds of neighbors depend on window size

Small windows (C= +/- 2) : nearest words are syntactically similar words in same taxonomy

- *Hogwarts* nearest neighbors are other fictional schools
- *Sunnydale, Evernight, Blandings*

Large windows (C= +/- 5) : nearest words are related words in same semantic field

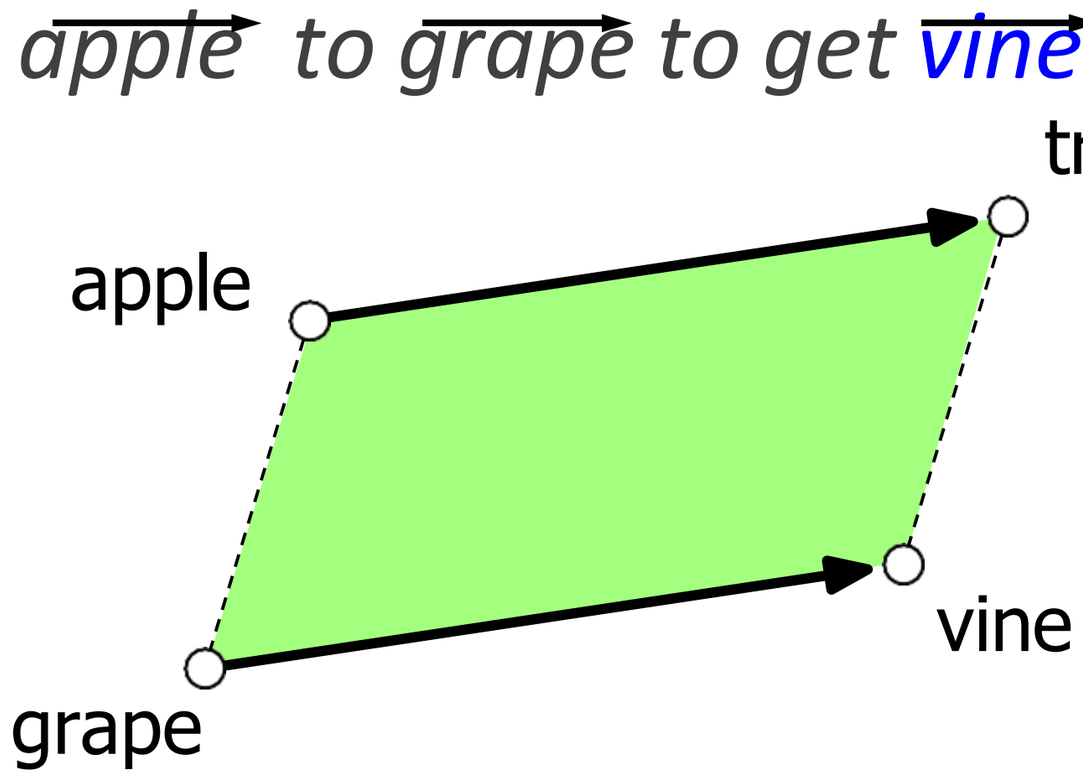
- *Hogwarts* nearest neighbors are Harry Potter world:
- *Dumbledore, half-blood, Malfoy*

Analogical relations

The classic parallelogram model of analogical reasoning (Rumelhart and Abrahamson 1973)

To solve: *"apple is to tree as grape is to _____"*

Add tree – apple to grape to get vine



Analogical relations via parallelogram

The parallelogram method can solve analogies with both sparse and dense embeddings (Turney and Littman 2005, Mikolov et al. 2013b)

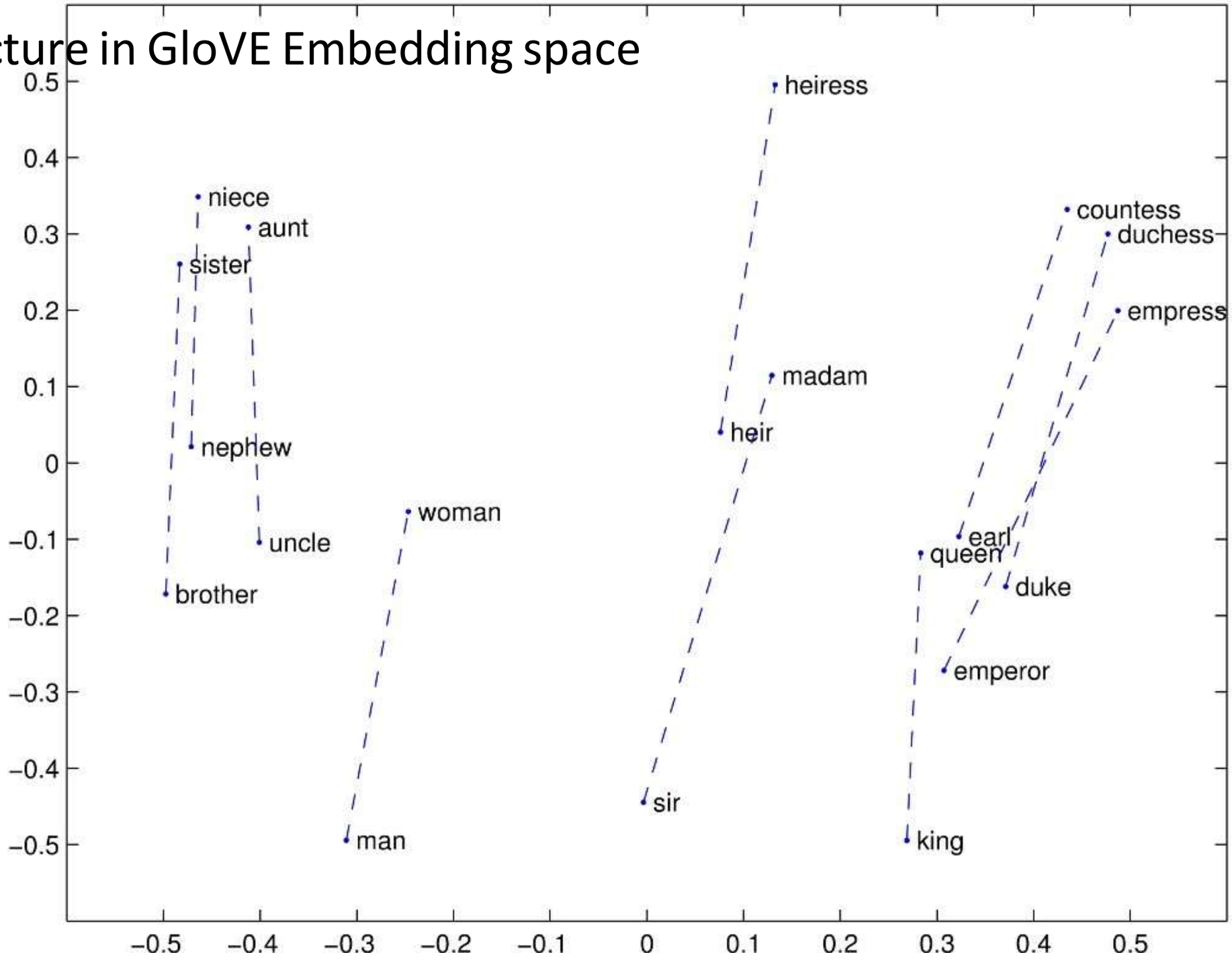
$\overrightarrow{\text{king}} - \overrightarrow{\text{man}} + \overrightarrow{\text{woman}}$ is close to $\overrightarrow{\text{queen}}$

$\overrightarrow{\text{Paris}} - \overrightarrow{\text{France}} + \overrightarrow{\text{Italy}}$ is close to $\overrightarrow{\text{Rome}}$

For a problem a:a*::b:b*, the parallelogram method is:

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

Structure in GloVE Embedding space



Caveats with the parallelogram method

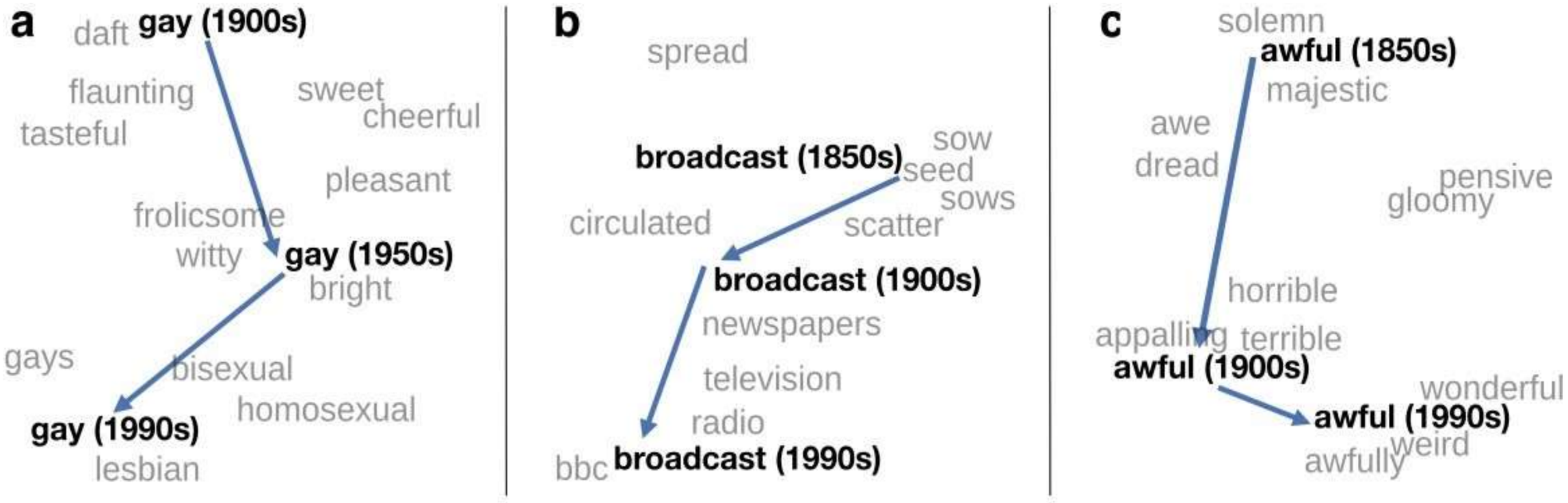
It only seems to work for frequent words, small distances and certain relations (relating countries to capitals, or parts of speech), but not others. (Linzen 2016, Gladkova et al. 2016, Ethayarajh et al. 2019a)

Understanding analogy is an open area of research (Peterson et al. 2020)

Embeddings as a window onto historical semantics

Train embeddings on different decades of historical text to see meanings shift

~30 million books, 1850-1990, Google Books data



William L. Hamilton, Jure Leskovec, and Dan Jurafsky. 2016. Diachronic Word Embeddings Reveal Statistical Laws of Semantic Change. Proceedings of ACL.

Embeddings reflect cultural bias!

Bolukbasi, Tolga, Kai-Wei Chang, James Y. Zou, Venkatesh Saligrama, and Adam T. Kalai. "Man is to computer programmer as woman is to homemaker? debiasing word embeddings." In *NeurIPS*, pp. 4349-4357. 2016.

Ask “Paris : France :: Tokyo : x”

- x = Japan

Ask “father : doctor :: mother : x”

- x = nurse

Ask “man : computer programmer :: woman : x”

- x = homemaker

Algorithms that use embeddings as part of e.g., hiring searches for programmers, might lead to bias in hiring

Historical embedding as a tool to study cultural biases

Garg, N., Schiebinger, L., Jurafsky, D., and Zou, J. (2018). Word embeddings quantify 100 years of gender and ethnic stereotypes. Proceedings of the National Academy of Sciences 115(16), E3635–E3644.

- Compute a **gender or ethnic bias** for each adjective: e.g., how much closer the adjective is to "woman" synonyms than "man" synonyms, or names of particular ethnicities
  - Embeddings for **competence** adjective (*smart, wise, brilliant, resourceful, thoughtful, logical)* are biased toward men, a bias slowly decreasing 1960-1990
  - Embeddings for **dehumanizing** adjectives (barbaric, monstrous, bizarre) were biased toward Asians in the 1930s, bias decreasing over the 20th century.
- These match the results of old surveys done in the 1930s

Semantics with Dense Vectors

To represent a word as a sparse vector with dimensions corresponding to the words in the vocabulary, and whose values were some function of the count of the word co- occurring with each neighboring word. Each word is represented with a vector that is both long (length , with vocabularies of 20,000 to 50,000) and sparse, with most elements of the vector for each word equal to zero.

- Now we turn to an alternative family of methods of representing a word.
- We use the vectors that are short (of length perhaps 50 – 1000) and dense (most values are non-zero).
- Short vectors have a number of potential advantages.
- First, they are easier to include as features in machine learning systems.

For example, if we use 100-dimensional word embeddings as features:

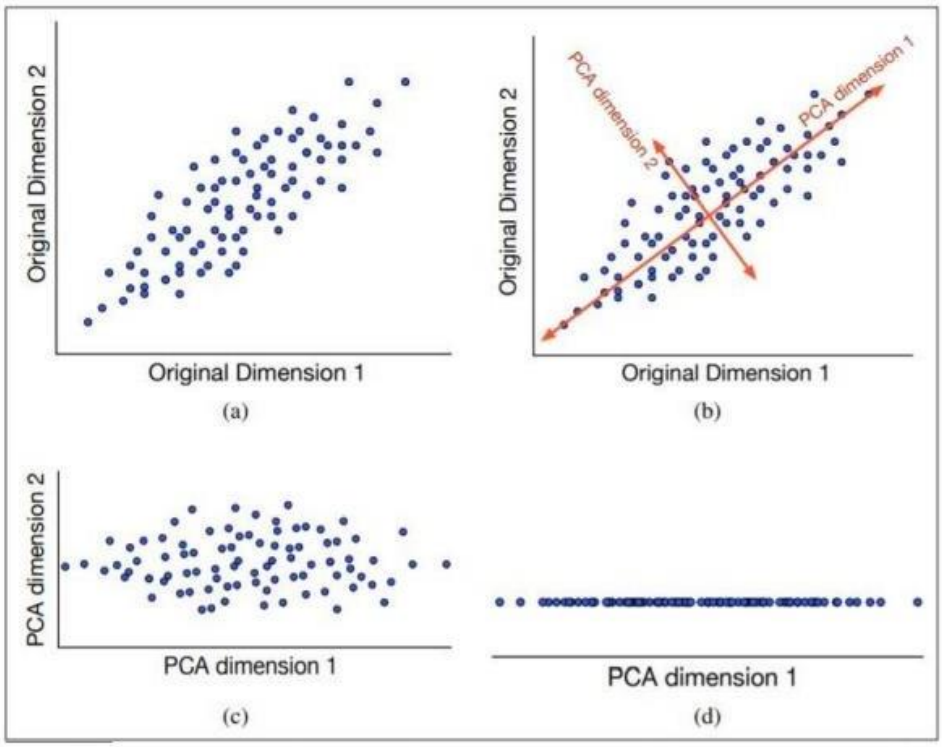
- A classifier can just learn 100 weights to represent a function of word meaning, instead of having to learn tens of thousands of weights for each of the sparse dimensions.
- Because they contain fewer parameters than sparse vectors of explicit counts, dense vectors may generalize better and help avoid overfitting.

Dense vectors may do a better job of capturing synonymy than sparse vectors. For example, car and automobile are synonyms. In a typical sparse vectors representation, the car dimension and the automobile dimension are distinct dimensions. Because the relationship between these two dimensions is not modeled, sparse vectors may fail to capture the similarity between a word with car as a neighbor and a word with automobile as a neighbor.

Singular value decomposition or SVD

SVU is first applied to the task of generating embedding from term-document matrices by Deerwester et al. (1988) in a model called Latent Semantic Indexing or Latent Semantic Analysis (LSA) • Singular Value Decomposition (SVD) is a method for finding the most important dimensions of a data set, those dimensions along which data varies the most.

- In general, dimensionality reduction methods first rotate the axes of the original dataset into a new space.
- The new space is chosen so that the highest order dimension captures the most variance in the dataset.
- The next dimension captures the next most variance, and so on.



- A set of points (vectors) in two dimensions is rotated so that the first new dimension captures the most variation in the data.
- In this new space, we can represent data with a smaller number of dimensions (for example using one dimension instead of two) and still capture much of the variation in the original data.

Latent Semantic Analysis

Latent Semantic Analysis (LSA) is a particular application of SVD to **|V| x c** a term-document matrix X representing **|V|** words and their co-occurrence with c documents or contexts. SVD factorizes any such rectangular **|V| x c** matrix X into the product of three matrices **W, s, and Y^T,** i.e. **W s Y^T**. In the **|V| x M** Matrix W, each of the w rows still represents a word, but columns do not.

- Each column now represents one of m dimensions in a latent space.
- The m column vectors are orthogonal to each other.
- The columns are ordered by the amount of variance in the original dataset each accounts for.
- The number of such dimensions' m is the rank of X (the rank of a matrix is the number of linearly independent rows).
- **s** is a diagonal **M X M** matrix, with singular values along the diagonal, expressing the importance of each dimension.

- The **M x C** matrix $\mathbf{Y^T}$ denoted as **C**, still represents documents or contexts, but each row now represents one of the new latent dimensions and the m row vectors are orthogonal to each other.
- By using only the first k dimensions, of **W, s, and C** instead of all m dimensions, the product of these 3 matrices becomes a least-squares approximation to the original X.
- Since the first dimensions encode the most variance, one way to view the reconstruction is thus as modeling the most important information in the original dataset.

SVD applied to co-occurrence matrix X:

$$\underset{|V| \times c}{\begin{bmatrix} X \end{bmatrix}} = \underset{|V| \times m}{\begin{bmatrix} W \end{bmatrix}} \underset{m \times m}{\begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_m \end{bmatrix}} \underset{m \times c}{\begin{bmatrix} C \end{bmatrix}}$$

Taking only the top **k, K<=m** dimensions after the SVD is applied to the co-occurrence matrix X:

$$\underset{|V| \times c}{\begin{bmatrix} X \end{bmatrix}} = \underset{|V| \times k}{\begin{bmatrix} W_k \end{bmatrix}} \underset{k \times k}{\begin{bmatrix} \sigma_1 & 0 & 0 & \cdots & 0 \\ 0 & \sigma_2 & 0 & \cdots & 0 \\ 0 & 0 & \sigma_3 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & \sigma_k \end{bmatrix}} \underset{k \times c}{\begin{bmatrix} C \end{bmatrix}}$$

SVD factors a matrix into a product of three matrices, **W, Σ, and C**. Taking the first k dimensions gives a **|V| x k** matrix $\mathbf{W_k}$ **that** has one k- dimensioned row per word that can be used as an embedding.

- Using only the top k dimensions (corresponding to the k most important singular values) leads to a reduced |V| x k matrix Wk with one k-dimensioned row per word.
- This row now acts as a dense k-dimensional vector (embedding) representing that word, substituting for the very high-dimensional rows of the original X.
- LSA embedding generally set k=300, so these embedding are relatively short by comparison to other dense embedding.
- Instead of PPMI or tf-idf weighting on the original term-document matrix, LSA implementations generally use a particular weighting of each co-occurrence cell that multiplies two weights called the local and global weights for each cell (i, j) – term i in document j

The local weight of each term i is its log frequency:

$$\log f(i,j) + 1$$

The global weight of term i is a version of its entropy:

$$1 + \frac{\sum_j p(i,j) \log p(i,j)}{\log D}$$

D is the number of documents.

****

Skip-gram and CBOW

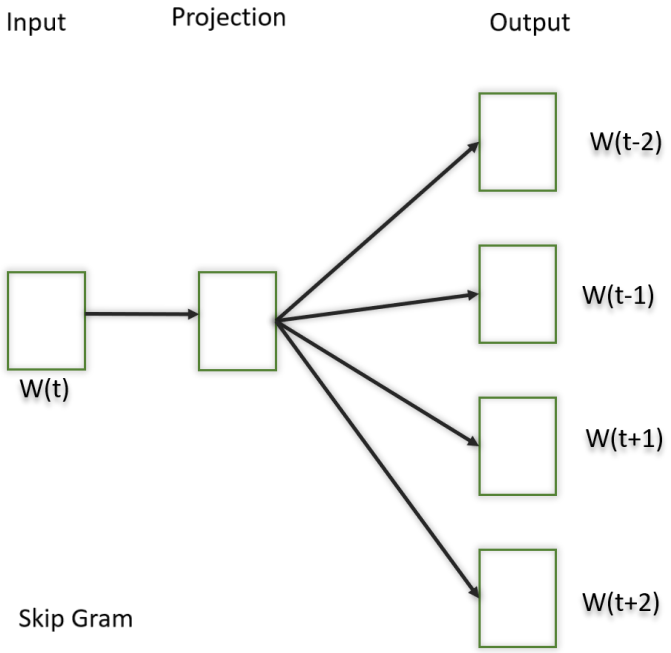
Word2Vec:

In Word2Vec every word is assigned a vector. We start with either a random vector or **one-hot vector**.

One-Hot vector: A representation where only one bit in a vector is 1. If there are 500 words in the corpus, then the vector length will be 500. After assigning vectors to each word we take a window size and iterate through the entire corpus. While we do this there are two **neural embedding methods** which are used:

Skip Gram

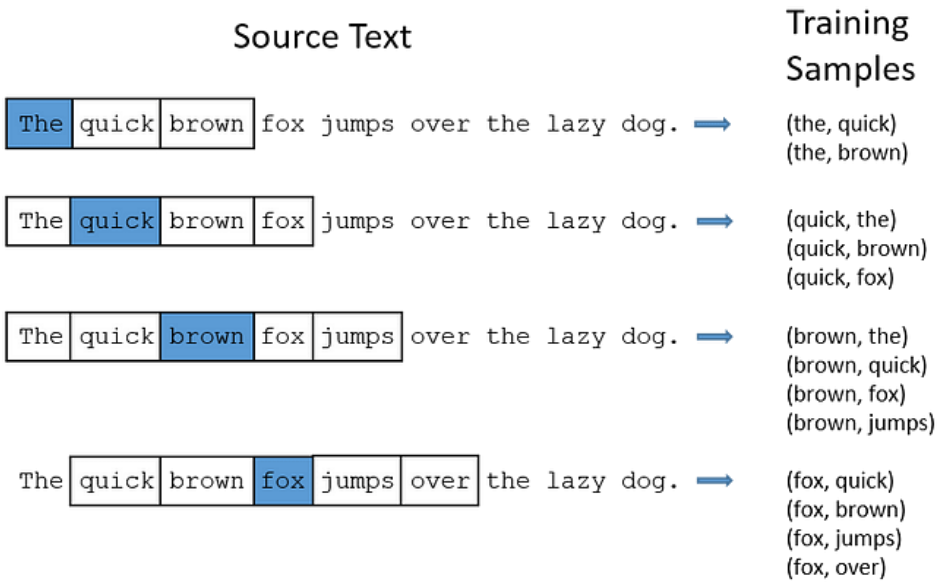
In this model, we try to make the central word closer to the neighboring words. It is the complete opposite of the CBOW model. It is shown that this method produces more meaningful embedding.



After applying the above neural embedding methods, we get trained vectors of each word after many iterations through the corpus. These trained vectors preserve syntactical or semantic information and are converted to lower dimensions. The vectors with similar meaning or semantic information are placed close to each other in space.

Let's say we want to train a skip-gram word2vec model over an input sentence:

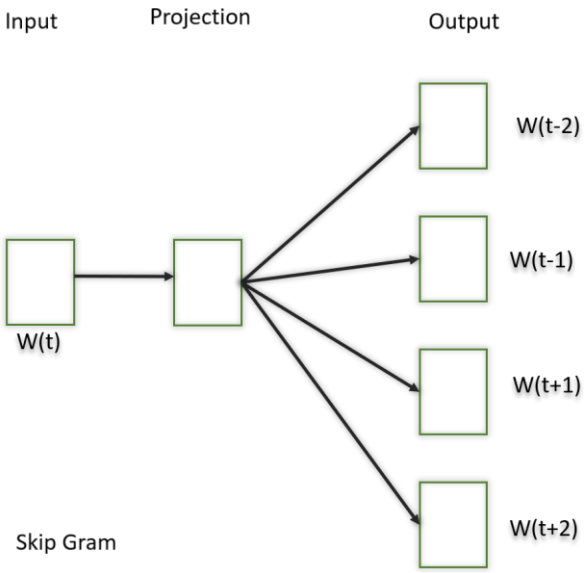
"The quick brown fox jumps over the lazy dog"



- 'The' becomes the first target word and since it's the first word of the sentence, there are no words to its left, so the window of size 2 only extends to its right resulting in the listed training samples.
- As our target shifts to the next word, the window expands by 1 on left because of the presence of a word on the left of the target.
- Finally, when the target word is somewhere in the middle, training samples get generated as intended.

Continuous Bowl of Words(CBOW)

In this model what we do is we try to fit the neighboring words in the window to the central word. After applying the above neural embedding methods, we get trained vectors of each word after many iterations through the corpus. These trained vectors preserve syntactical or semantic information and are converted to lower dimensions. The vectors with similar meaning or semantic information are placed close to each other in space.



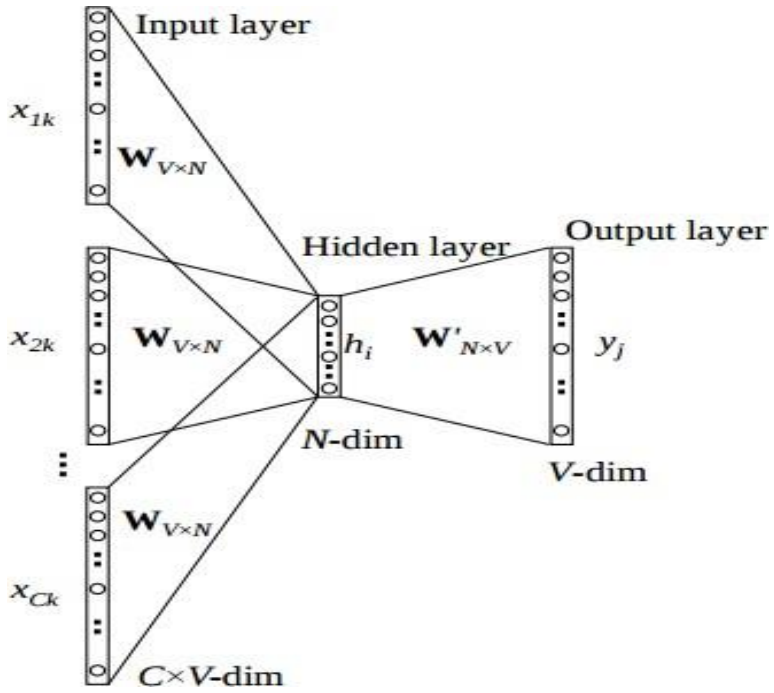
1. I enjoy flying.
2. I like NLP.
3. I like deep learning.

The resulting counts matrix will then be:

$$X = \begin{array}{c} \\ I \\ like \\ enjoy \\ deep \\ learning \\ NLP \\ flying \\ . \end{array} \begin{array}{c} \begin{array}{cccccccc} I & like & enjoy & deep & learning & NLP & flying & . \end{array} \\ \begin{bmatrix} 0 & 2 & 1 & 0 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 & 1 & 0 \end{bmatrix} \end{array}$$

- The dimension of our hidden layer and output layer stays the same as the skip-gram model.
- However, just as we read that a CBOW model takes context words as input, here the input is C context words in the form of a one-hot encoded vector of size 1xV each, where V = size of vocabulary, making the entire input CxV dimensional.
- Now, each of these C vectors will be multiplied with the Weights of our hidden layer which are of the shape VxN, where V = vocab size and N = Number of neurons in the hidden layer.
- If you can imagine, this will result in C, 1xN vectors, and all of these C vectors will be averaged element-wise to obtain our final activation for the hidden layer, which then will be fed into our output softmax layer.
- The learned weight between the hidden and output layer makes up the word embedding representation.

CBOW Model Architecture



****

Concept of Word Sense

Word Sense Disambiguation

Word Sense Disambiguation is an important method of NLP by which the meaning of a word is determined, which is used in a particular context. NLP systems often face the challenge of properly identifying words, and determining the specific usage of a word in a particular sentence has many applications.

Word Sense Disambiguation basically solves the ambiguity that arises in determining the meaning of the same word used in different situations.

Word Sense Disambiguation Applications

WSD has many applications in various text processing and NLP fields.

- WSD can be used alongside Lexicography. Much of the modern Lexicography is corpus-based. WSD, used in Lexicography can provide significant textual indicators.
- WSD can also be used in Text Mining and Information Extraction tasks. As the major purpose of WSD is to accurately understand the meaning of a word in particular usage or sentence, it can be used for the correct labeling of words.
- For example, from a security point of view, a text system should be able to understand the difference between a coal “mine” and a land “mine”.

- While the former serves industrial purposes, the latter is a security threat. So a text mining application must be able to determine the difference between the two.
- Similarly, WSD can be used for Information Retrieval purposes. Information Retrieval systems work through text data primarily based on textual information. Knowing the relevance of using a word in any sentence will surely help.

Challenges in Word Sense Disambiguation

WSD faces a lot of challenges and problems.

- The most common problem is the difference between various dictionaries or text corpus. Different dictionaries have different meanings for words, which makes the sense of the words to be perceived as different. A lot of text information is out there and often it is not possible to process everything properly.
- Different applications need different algorithms and that is often a challenge for WSD.
- A problem also arises is that words cannot be divided into discrete meanings. Words often have related meanings and this causes a lot of problems.

WSD implement

There are four main ways to implement WSD.

These are:

Dictionary- and knowledge-based methods:

These methods rely on text data like dictionaries, thesaurus, etc. It is based on the fact that words that are related to each other can be found in the definitions. The popularly used Lesk method, which we shall discuss more later is a seminal dictionary-based method.

Supervised methods:

In this type, sense-annotated corpora are used to train machine learning models. But, a problem that may arise is that such corpora are very tough and time-consuming to create.

Semi-supervised Methods:

Due to the lack of such corpus, most word sense disambiguation algorithms use semi-supervised methods. The process starts with a small amount of data, which is often manually created.

This is used to train an initial classifier. This classifier is used on an untagged part of the corpus, to create a larger training set. Basically, this method involves bootstrapping from the initial data, which is referred to as the seed data.

Semi-supervised methods thus, use both labeled and unlabeled data.

Unsupervised Methods:

Unsupervised Methods pose the greatest challenge to researchers and NLP professionals. A key assumption of these models is that similar meanings and senses occur in a similar context. They are not dependent on manual efforts, hence can overcome the knowledge acquisition deadlock.

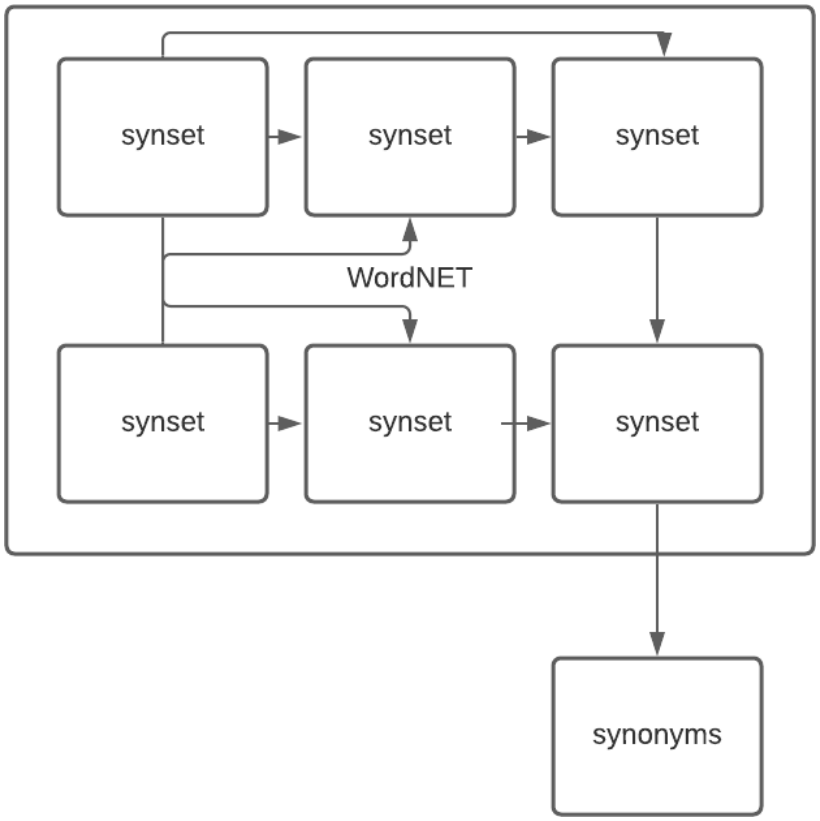
****

Introduction to WordNet

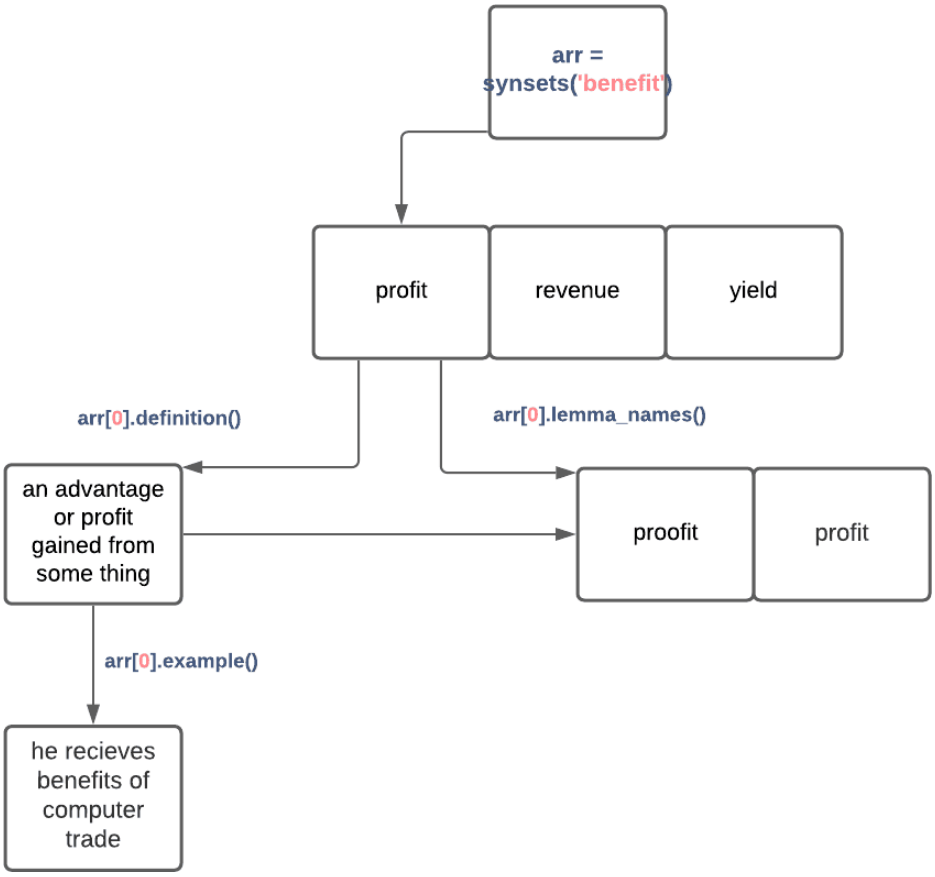
WordNET is a lexical database of words in more than 200 languages in which we have adjectives, adverbs, nouns, and verbs grouped differently into a set of cognitive synonyms, where each word in the database is expressing its distinct concept. The cognitive synonyms which are called synsets are presented in the database with lexical and semantic relations. WordNET is publicly available for download and also we can test its network of related words and concepts using this link. Below are a few test images when accessed this through the browser.

Structure of WordNET

The below image is a basic structure of the WordNET. The main concept of the relationship between the words in the WordNETs network is that the words are synonyms like sad and unhappy, benefit and profit. These words show the same concept of using them in similar contexts by interchanging them. These types of words are grouped into synsets which are unordered sets. Where synsets are linked together if they are having even small conceptual relations. Every synset in the network has its own brief definition and many of them are illustrated with the example of how to use them in a sentence. That definition and example part makes WordNET different from other



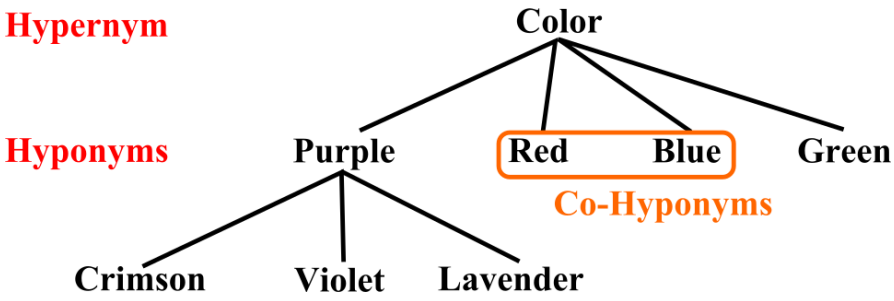
In the below picture we can see the structure of any synset where we are having synonyms of benefit in the array of synsets with the definition and the example of usage of benefit word. This synset is related to another synset word, where the words benefit and profit have exactly the same meaning.



Here we can see the structure of the wordnet and also how the synsets under the networks are interlinked because of the conceptual relation between the words.

Relations in the WordNET

Hyponym: In linguistics, a word with a broad meaning constitutes a category into which words with more specific meanings fall; a superordinate. For example, the colour is a hypernym of red. Where Hyponymy shows the relationship between a hypernym and a specific instance of a hyponym. A hyponym is a word or phrase whose semantic field is more specific than its hypernym. The semantic field of a hypernym, also known as a superordinate.


Hypernym
Color
Hyponyms
Purple
Red
Blue
Green
Co-Hyponyms
Crimson
Violet
Lavender