Vector Semantics

- **Lexical Semantics**
- **Vector Semantics**

Lexical Semantics

Lexical Semantics

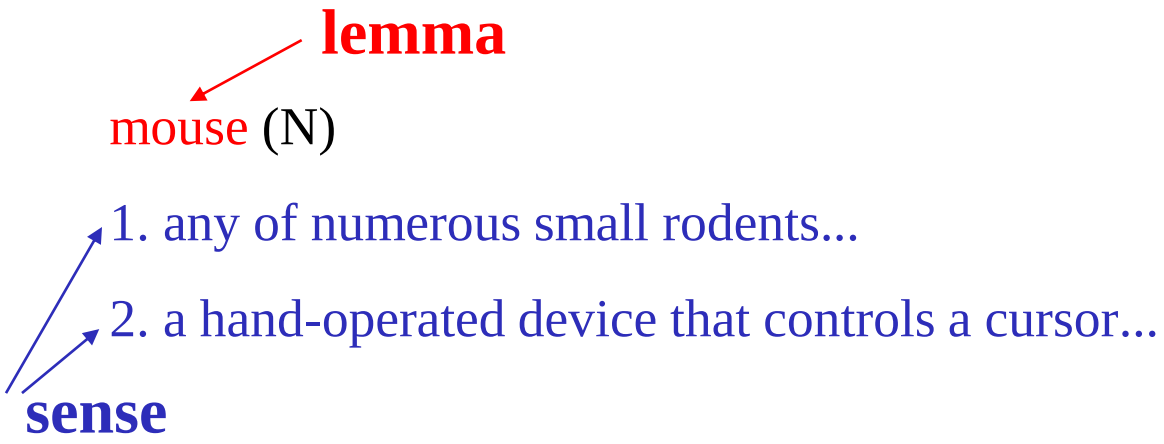
- **How should we represent the meaning of a word?**
  - A model of word meaning should allow us to draw useful inferences that will help us solve meaning-related tasks like question-answering, summarization, paraphrase or plagiarism detection.

- **The linguistic study of systematic meaning related structure of words is called Lexical Semantics.**

- **What's a word?**
  - Types, tokens, stems, roots, inflected forms?

Words, Lemmas and Senses

- A word ***mouse*** might be defined in a dictionary as follows:



- The form mouse is the **lemma**.
  - The form mouse would also be the **lemma** for the word mice;
    - dictionaries don't have separate definitions for **inflected forms** like mice.
  - Similarly sing is the lemma for sing, sang, sung.
    - In many languages the **infinitive form** is used as the **lemma for the verb.**
- The ***specific forms*** *sung*, *carpets* or *sing* are called **word forms**.

Words, Lemmas and Senses

- Each lemma can have multiple meanings;
  - the lemma mouse can refer to *rodent* or *cursor control device*.
- We call each of these aspects of the meaning of *mouse* a **word sense**.

- The fact that lemmas can be **homonymous** (have multiple senses) can make interpretation difficult.
  - mouse ➔ *rodent* or *cursor control device*.
  - **Word sense disambiguation** is the task of determining which sense of a word is being used in a particular context.

- There can be many different **relationships between word senses.**
  - synonymy, antonym, similarity, superordinate/subordinate, …

Relation: Synonym

- **When one word has a sense whose meaning is identical to a sense of another word, or nearly identical, we say the two senses of those two words are synonyms.**
  - couch/sofa, big/large, car/automobile, vomit/throw up, …

- Note that there are probably no examples of *perfect synonymy*.
  - Even if many aspects of meaning are identical
  - Still may not preserve the acceptability based on notions of politeness, slang, register, genre, etc.
    - That's my ***big*** sister a ***big*** plane
    - That's my ***large*** sister a ***large*** plane

- **The Linguistic Principle of Contrast:**
  - Difference in form → difference in meaning

Relation: Antonym

- **Two word senses can be antonyms** if they are opposites with respect to one feature of meaning.

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

- **Antonyms** can define a binary opposition with respect to one feature of meaning or are at opposite ends of some scale.
  - `long/short` or `big/little`, which are at opposite ends of the length or size scale.
  - **Reversives** are another group of antonyms which describe change or movement in opposite directions, such as `rise/fall` or `up/down`.
- Antonyms thus differ completely with respect to one aspect of their meaning but they are otherwise very similar, sharing almost all other aspects of meaning.
  - Thus, automatically distinguishing *synonyms* from *antonyms* can be difficult because they appear in similar contexts.

Relation: Word Similarity

- While words don't have many synonyms, most words do have lots of **similar words**.
  - *Cat* is not a synonym of *dog*, but *cats* and *dogs* are certainly **similar words**.

- **Similar words (words with similar meanings) share some element of meaning.**
  - Although similar words are not synonyms, but they share some element of meaning.
    - *Cat* and *dog* are both animals (house pet).

- Which words are similar?

```
car   suv   bicycle   eggplant
```

- The notion of **word similarity** is very useful in larger semantic tasks.
  - *Word similarity* help to determine *phrase or sentence similarity* .
  - *Phrase or sentence similarity* is useful in NLP tasks as question answering, paraphrasing, and summarization.

Relation: Word Similarity

- One way of getting values for word similarity is to ask humans to judge how similar one word is to another.
  - SimLex-999 dataset (Hill et al., 2015) gives values on a scale from 0 to 10 which range from near-synonyms (*vanish*, *disappear*) to pairs that scarcely seem to have anything in common (*hole*, *agreement*):

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

Relation: Word Relatedness

- **Word Relatedness: The meaning of two words can be related in ways others than similarity.**

- The word relatedness is also called as **word association**.

- *Coffee* is **not similar** to *cup* because they share practically no features.
- But *coffee* and *cup* **are related** since they are associated in a shared event (the event of drinking coffee out of a cup).

- *Scalpel* and *surgeon* are **not similar** but **are related** eventively (a *surgeon* tends to make use of a *scalpel*).

Relation: Word Relatedness
Semantic Field

- One common kind of relatedness between words is **if they belong to the same semantic field**.
- A **semantic field** is **a set of words which cover a particular semantic domain and bear structured relations with each other**.

Semantic Field Examples:

- Hospitals: surgeon, scalpel, nurse, anesthetic, hospital
- Restaurants: waiter, menu, plate, food, menu, chef,
- Houses: door, roof, kitchen, family, bed

- Semantic fields are also related to **topic models**.
- **Semantic fields** and **topic models** are a very useful tool for *discovering topical structure in documents*.

Relation: Word Relatedness
Semantic Frame

- A **semantic frame** is a set of words that denote perspectives or participants in a particular type of event.
  - A *commercial transaction* is a kind of event in which one entity trades money to another entity in return for some good, after which the good changes hands.
  - The *commercial transaction* event can be encoded lexically by using verbs like **buy** or **sell**.

- **Semantic frames** have **semantic roles** (like *buyer*, *seller*, *goods*), and words in a sentence can take on these roles.

- Sam bought the book from John could be paraphrased as John sold the book to Sam by knowing
  - a commercial transaction can be encoded by **buy** or **sell** .
  - Semantic roles:
    - Buyer: **Sam** Seller: **John** Goods: **book**

Relation: Taxonomic Relations

- A word (or word sense) is a **hyponym (subordinate)** of another word (or word sense) if the first is more *specific*, denoting a *subclass* of the other.
- A word (or word sense) is a **hypernym (superordinate)** of another word (or word sense) if the first is more *general*, denoting a *superclass* of the other.

– **car** is a **subordinate** of **vehicle**	**vehicle** is a **superordinate** of **car**
– **mango** is a **subordinate** of **fruit**	**fruit** is a **superordinate** of **mango**

Relation: Taxonomic Relations

- **Hypernymy** can also be defined in terms of **entailment**.
  - A word sense A is a hyponym of a word sense B if everything that is A is also B, and hence being an A entails being a B.

- Another name for the **hypernym/hyponym** structure is the **IS-A hierarchy**, in which we say **A IS-A B**, or **B subsumes A**.
  - **IS-A:** **car IS-A vehicle**
  - **Entailment:** every **car** is a **vehicle**

- **Hypernymy** is useful for tasks like *textual entailment* or *question answering*.
  - knowing that *leukemia* is a type of *cancer* would certainly be useful in answering questions about *leukemia*.

Relation: Connotation

- Words have **affective meanings (**or **connotations**).
  - **connotation** means the aspects of a word's meaning that are related to a writer or reader's emotions, sentiment, opinions, or evaluations.

  - ***positive connotations:*** happy
  - ***negative connotations:*** sad
  - ***positive evaluation:*** great, love
  - ***negative evaluation:*** terrible, hate

- *Positive or negative evaluation expressed through language* is called **sentiment.**

Vector Semantics

Vector Semantics

- How can we build a computational model that successfully deals with the different aspects of word meaning (word senses, word similarity and relatedness, semantic fields and frames, connotation)?
- There is **NO perfect model** that completely deals with each of the different aspects of word meaning.

- One computational model is the usage of a lexicon such as **WordNet**:
  - WordNet is a database of **lexical relations** for English (and other languages).

- The current best model is the usage of **vector semantics**.

Vector Semantics

Distributionalist Approach

- **Ludwig Wittgenstein** who was skeptical of the possibility of building a completely formal theory of meaning definitions for each word suggested instead that **"the meaning of a word is its use in the language".**
  - That is, instead of using some logical language to define each word, we should define words by some representation of how the word was used by actual people.

- **In distributionalist approach, a word is defined by its environment or its distribution in language use.**
  - **Words are defined by their usages.**

- A **word's distribution** is the set of contexts in which it occurs, the neighboring words or grammatical environments.

- The idea is that two words that occur in very similar distributions (that occur together with very similar words) are likely to have the same meaning.

Vector Semantics

Distributionalist Approach: Example

- We do not know the meaning of the word **ongchoi**, but we see its usage in the following sentences (contexts).
  - **Ongchoi** is delicious sauteed with garlic.
  - **Ongchoi** is superb over rice.
- Some of context words of **ongchoi** (delicious, sauteed, garlic, rice) also occur in the following contexts.
  - ...**spinach** sauteed with garlic over rice...
  - ...**spinach** leaves are delicious...
- The fact that the context words of **ongchoi** also occur around **spinach** can help us discover the similarity between **spinach** and **ongchoi**.
  - **Ongchoi** is **"water spinach"**

Vector Semantics

- Vector semantics combines two intuitions:

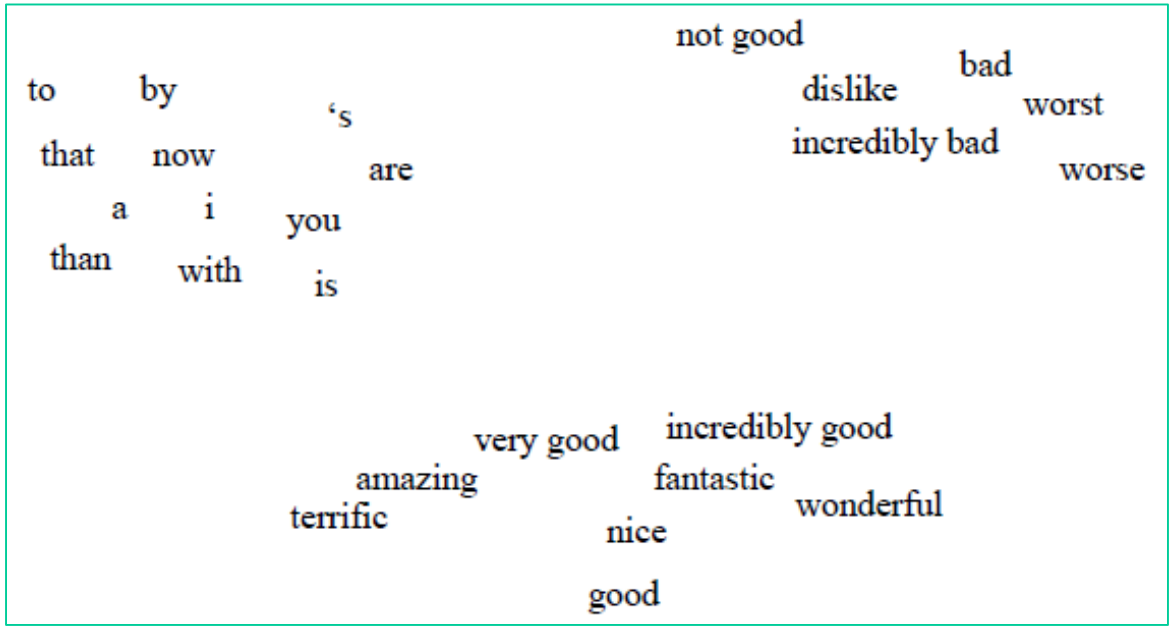
  Distributionalist Intuition: Define a word by counting what other words occur in its environment.

  Vector Intuition: Define the meaning of a word w as a vector, a list of numbers, a point in N dimensional space.

- There are various versions of vector semantics, each defining the numbers in the vector somewhat differently, but in each case the numbers are based in some way on counts of neighboring words.
- The **idea of vector semantics** is ***to represent a word as a point in some multidimensional semantic space.***

Vector Semantics: embeddings

- **Vectors for representing words** are generally called **embeddings**, because the word is embedded in a particular vector space.



positive and negative words seem to be located in distinct portions of the space

A two-dimensional projection of embeddings (learned for a sentiment analysis task) for some words and phrases, showing that words with similar meanings are nearby in space.

Vector Semantics: embeddings

- Embeddings are a fine-grained model of meaning for similarity.
- In embeddings, similar words appear "nearby in space".
- In Sentiment analysis:
  - With words, requires **same** word to be in training and test.
  - With embeddings, ok if **similar** words occurred!!!
- **Vector semantic models** are also extremely practical because they can be learned automatically from text without any complex labeling or supervision.
- We will look at two models:

  tf-idf model:
  - often used a baseline, very long vectors that are sparse,
  - the meaning of a word is defined by a simple function of the counts of nearby words.

  word2vec model:
  - Dense vectors
  - Representation is created by training a classifier to distinguish nearby and far-away words.

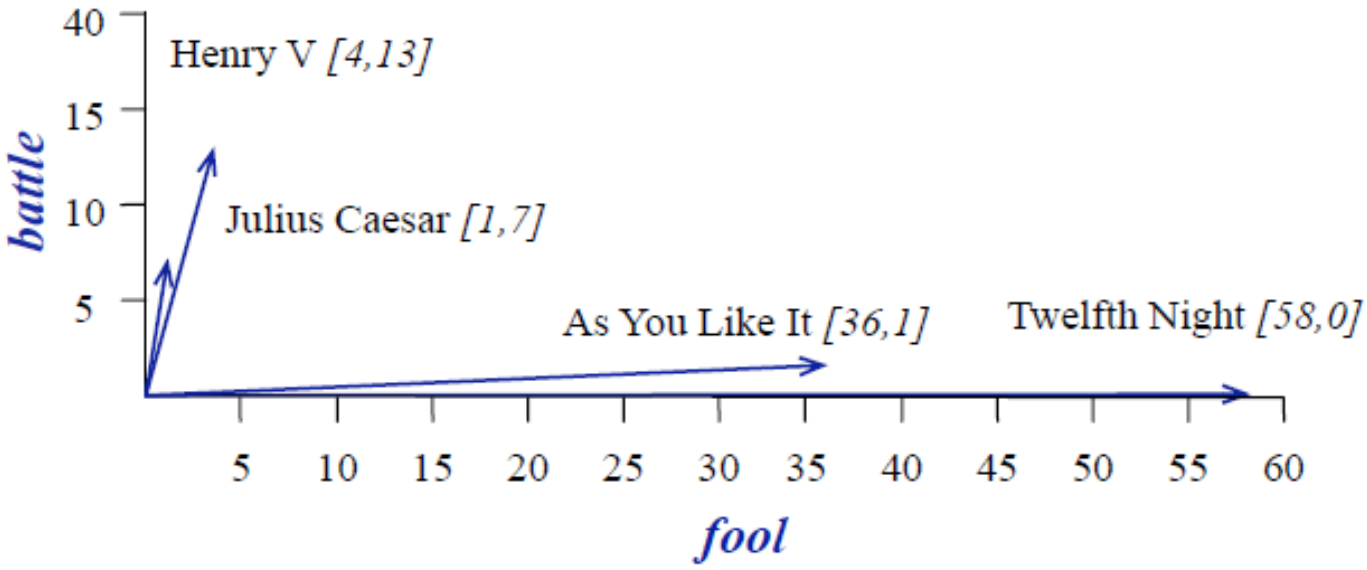
Term-Document Matrix

- Vector models of meaning are generally based on a co-occurrence matrix, a way of representing how often words co-occur.
- In a **term-document matrix**, each row represents a word in the vocabulary and each column represents a document from some collection of documents.
- **Each document is represented by a vector of words.**
  - **The vector for a document is a point in |V|-dimensional space.**

	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

- The term-document matrix for four words in four Shakespeare plays.
- The red boxes show that each document is represented as a column vector of length four.

Visualizing Document Vectors



- A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words **battle** and **fool**.
- The comedies have high values for the **fool** dimension and low values for the **battle** dimension.

Term-Document Matrix

- Term-document matrices are used to find similar documents for the task of ***information retrieval***.
  - Two documents that are similar tend to have similar words.

- The vectors of two comedy documents are similar and they are different than the vectors of two history documents.
  - Comedies have more **fools** and **wit** and fewer **battles**.

	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

Words as Vectors

- The **word vector** is a row vector rather than a column vector.
  - The four dimensions of the vector for **fool**, **wit**, **battle**, **good**.
  - Each entry in the vector thus represents the counts of the word's occurrence in the document corresponding to that dimension.

	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 ***history*** documents (Julius Caesar and Henry V)"
  - **fool** is "the kind of word that occurs in ***comedies***, especially Twelfth Night"

Word-Word Matrix

- Rather than the term-document matrix, we use the **term-term matrix**, more commonly called the **word-word matrix** (or the ***term-context matrix***) in which the columns are labeled by words rather than documents.
  - **The matrix is |V|x|V| matrix and each cell records number of times the row (target) word and column (context) word co-occur in some context in some training corpus.**
    - The context could be the document, in which case the cell represents the number of times the two words appear in the same document.
    - It is most common to use smaller contexts: generally a window around the word, for example of 4 words to left and 4 words to right, in which case the cell represents number of times column word occurs in such a ± 4 word window around row word.
- Each row in word-word matrix is **co-occurrence vector (context vector) of that row (target) word**.
- Two words are **similar in meaning** if **their context vectors are similar**.

Word-Word Matrix

- Co-occurrence vectors for four words, computed from the Brown corpus, showing only six of the dimensions.
  - A real vector would have vastly more dimensions and thus be much sparser.

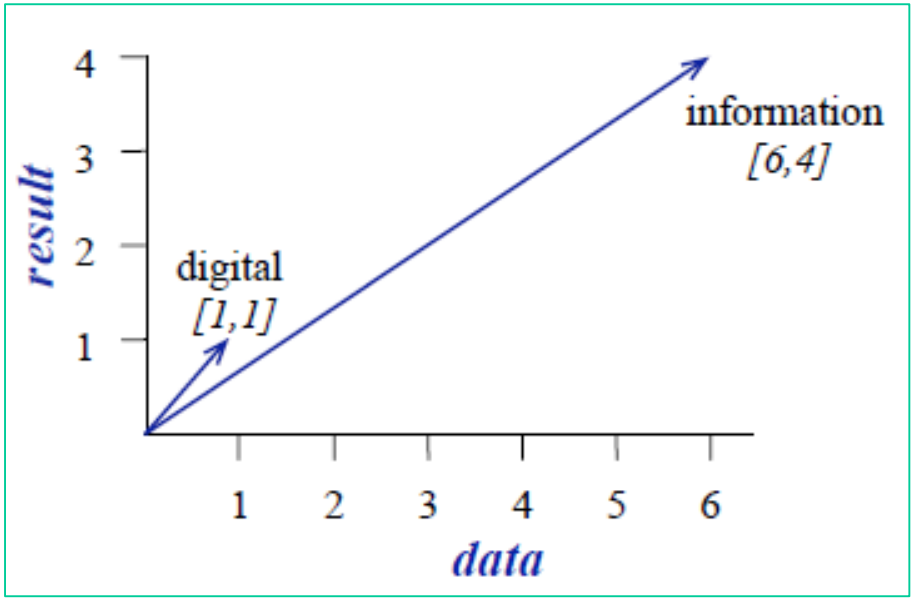
	aardvark	...	computer	data	pinch	result	sugar	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	

  - **apricot** and **pineapple** are more similar to each other (both **pinch** and **sugar** tend to occur in their window).
  - **digital** and **information** are more similar to each other.

Word-Word Matrix

	aardvark	...	computer	data	pinch	result	sugar	...
apricot	0	...	0	0	1	0	1	
pineapple	0	...	0	0	1	0	1	
digital	0	...	2	1	0	1	0	
information	0	...	1	6	0	4	0	

- A spatial visualization of word vectors for digital and information, showing just two of the dimensions, corresponding to the words data and result.



Cosine for Measuring Similarity

- To define similarity between two target words **v** and **w**, we need a measure for taking two such vectors and giving a measure of vector similarity.
- The **most common similarity metric** is the **cosine of the angle between the vectors**.

- The **cosine** is based on **dot product** operator, also called **inner product**:

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

- The ***dot product acts as a similarity metric*** because:
  - It will tend to be high just when the two vectors have large values in the same dimensions.
  - Alternatively, vectors that have zeros in different dimensions—orthogonal vectors—will have a dot product of 0, representing their strong **dissimilarity**.

Cosine for Measuring Similarity

- The **raw dot-product** *has a problem as a similarity metric*: it favors long vectors.

- The **vector length** is defined as:

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

  - The dot product is higher if a vector is longer, with higher values in each dimension.
  - More frequent words have longer vectors, since they tend to co-occur with more words and have higher co-occurrence values with each of them.
  - The raw dot product will be higher for frequent words.
  - But this is a problem; we'd like a similarity metric that tells us how similar two words are regardless of their frequency.

Cosine for Measuring Similarity

- The **normalized dot product** is same as **cosine of the angle between two vectors.**
- **Cosine similarity metric between two vectors $\vec{v}$ and $\vec{w}$ can be computed as:**

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

  - The cosine value ranges from **1 for vectors pointing in the same direction**, through **0 for vectors that are orthogonal**, *to -1 for vectors pointing in opposite directions*.
  - But raw frequency values are non-negative, so cosine for these vectors ranges from 0 to 1.

Cosine for Measuring Similarity

- Cosine computes which of the words **apricot** or **digital** is closer in meaning to **information.**

	large	data	computer
apricot	2	0	0
digital	0	1	2
information	1	6	1

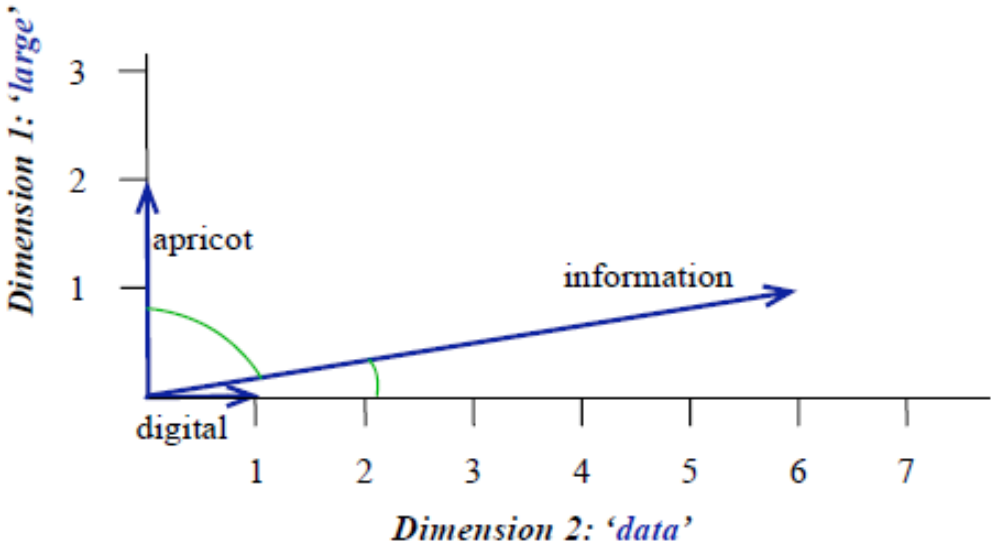
$$\cos(\text{apricot}, \text{information}) = \frac{2+0+0}{\sqrt{4+0+0}\sqrt{1+36+1}} = \frac{2}{2\sqrt{38}} = .16$$

$$\cos(\text{digital}, \text{information}) = \frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = \frac{8}{\sqrt{38}\sqrt{5}} = .58$$

- The model decides that **information** is closer to **digital** than it is to **apricot.**
  - cos(0) = 1 cos(90) = 0

Cosine for Measuring Similarity

- A graphical demonstration of **cosine similarity**, showing vectors for three words (**apricot**, **digital**, and **information**) in the ***two dimensional space*** defined by counts of the words **data** and **large** in the neighborhood.
  - The angle between **digital** and **information** is smaller than the angle between **apricot** and **information**.
  - When ***two vectors are more similar, the cosine is larger but the angle is smaller***



	large	data	computer
apricot	2	0	0
digital	0	1	2
information	1	6	1

TF-IDF: Weighing Terms in Vector

- Although frequency is important, but ***simple frequency*** isn't the best measure of association between words.
- **Word that occur nearby frequently** (maybe sugar appears often in our corpus near apricot) **are more important than words that only appear once or twice**.
- **On the other hand, words that are too frequent such as the or a are unimportant.**

- **How can we balance these two conflicting constraints?**
  - **We need a function that resolves this frequency paradox!**

    ➔ **one solution is TF-IDF (term frequency _ inverse document frequency)**
  - **TF-IDF value (the '-' here is a hyphen, not a minus sign) is the product of two terms, each term capturing one of these two intuitions.**

TF-IDF

tf: term frequency: the frequency of the word in the document.

- frequency is down-weighted by using its log value.
  - a word appearing 100 times in a document doesn't make that word 100 times more likely to be relevant to the meaning of the document.

- **term frequency** is defined as (in the book):

$$\mathrm{tf}_{t,d} = \begin{cases} 1 + \log_{10} \mathrm{count}(t,d) & \text{if } \mathrm{count}(t,d) > 0 \\ 0 & \text{otherwise} \end{cases}$$

  - count=0➔tf=0; count=1➔tf=1; count=10➔ tf=2; count=100➔tf=3

Term frequency can also be defined as: $\mathbf{tf_{t,d} = count(t,d) / \#ofwords(d)}$

TF-IDF

idf: inverse document frequency: give a higher weight to words that occur only in a few documents.

- Terms that are limited to a few documents are useful for discriminating those documents; terms that occur frequently across the entire collection aren't as helpful.
- **Document frequency df_t of a term t is the number of documents it occurs in.**
- Inverse document frequency idf can be using the fraction N/df_t where N is the number of documents in the collection.
  - Because of the large number of documents in many collections, this measure is usually squashed with a log function.

Inverse document frequency idf_t is defined as:

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

where N is the number of documents in the collection and df_t is the number of documents that contain the term t.

TF-IDF

TF-IDF Value:

- **tf-idf weighting of the value $w_{t,d}$ for term t in document d combines term frequency with idf:**

$$w_{t,d} = tf_{t,d} * idf_t$$

TF-IDF – Example

- A collection contains 37 documents (Shakespeare's play):
- **idf values** of some words in this collection:

Word	df	idf
Romeo	1	1.57
salad	2	1.27
Falstaff	4	0.967
forest	12	0.489
battle	21	0.074
fool	36	0.012
good	37	0
sweet	37	0

- A **tf-idf** weighted term-document matrix for four words in four Shakespeare plays.

	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

	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 Vector Models

- A **tf-idf vector model of a target word** is a vector with dimensions corresponding to all the words in the vocabulary.
  - The values in each dimension are the frequency with which the target word co-occurs with each neighboring context word, weighted by tf-idf.
  - *The model computes the similarity between two words x and y by taking the cosine of their tf-idf vectors; high cosine, high similarity*.

- To compare two documents:
  - Take the centroid of vectors of all the words in the document
  - Given k word vectors $w_1,\ldots,w_k$, **centroid document vector d** is:

$$d = \frac{w_1 + w_2 + \ldots + w_k}{k}$$

  - Given two documents, we can then compute their centroid document vectors d_1 and d_2, and estimate the similarity between the two documents by $\cos(d_1,d_2)$.

Pointwise Mutual Information (PMI)

- An alternative weighting function to tf-idf is called **PPMI (positive pointwise mutual information**).
  - PPMI draws on the intuition that best way to weigh the association between two words is to ask how much more the two words co-occur in our corpus than we would have a priori expected them to appear by chance.
- **Pointwise mutual information** is ***a measure of how often two events x and y occur, compared with what we would expect if they were independent***:

$$I(x,y) = \log_2 \frac{P(x,y)}{P(x)P(y)}$$

- **The pointwise mutual information between a target word w and a context word c** is then defined as:

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

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
- So, we just replace negative PMI values by 0

- **Positive PMI (PPMI) between a target word w and a context word c:**

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

Computing PPMI on a Term-Context Matrix

- We have a co-occurrence matrix with **W** rows (target words) and **C** columns (contexts), where f_{ij} gives the number of times word w_i occurs in context c_j.

	aardvark	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

- We can compute a PPMI matrix where $PPMI_{ij}$ gives the PPMI value of word w_i with context c_j as follows:

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

$$\text{PPMI}_{ij} = \max\left(\log_2 \frac{p_{ij}}{p_{i*}p_{*j}}, 0\right)$$

Computing PPMI on a Term-Context Matrix – Example

- Compute PPMI(w=information, c=data),

$$\sum_{i=1}^{W}\sum_{j=1}^{C} f_{ij} = 19$$

$$P(\text{w=information,c=data}) = \frac{6}{19} = .316$$

$$P(\text{w=information}) = \frac{11}{19} = .579$$

$$P(\text{c=data}) = \frac{7}{19} = .368$$

$$\text{ppmi(information,data)} = \log 2(.316/(.368 * .579)) = .568$$

	aardvark	computer	data	pinch	result	sugar
apricot	0	0	0	1	0	1
pineapple	0	0	0	1	0	1
digital	0	2	1	0	1	0
information	0	1	6	0	4	0

joint probabilities computed from the counts

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

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

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

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	p(w)
apricot	0	0	0.05	0	0.05	0.11
pineapple	0	0	0.05	0	0.05	0.11
digital	0.11	0.05	0	0.05	0	0.21
information	0.05	.32	0	0.21	0	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

Computing PPMI on a Term-Context Matrix – Example

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	p(w)
apricot	0	0	0.05	0	0.05	0.11
pineapple	0	0	0.05	0	0.05	0.11
digital	0.11	0.05	0	0.05	0	0.21
information	0.05	.32	0	0.21	0	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

$$\text{ppmi}(\text{information},\text{data}) = \log 2(.316/(.368 * .579)) = .568$$

- **PPMI Matrix**

	computer	data	pinch	result	sugar
apricot	0	0	2.25	0	2.25
pineapple	0	0	2.25	0	2.25
digital	1.66	0	0	0	0
information	0	0.57	0	0.47	0

Weighting PMI

Giving rare context words slightly higher probability

- PMI has the problem of being biased toward infrequent events;
  - very rare words tend to have very high PMI values.
- One way to reduce this bias is to slightly change computation for P(c), using a different function $P_\alpha(c)$ that raises contexts to power of α :

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

- Raise the context probabilities to $\alpha = 0.75$:
- This helps because $P_\alpha(c) > P(c)$ for rare c
- Consider two events, P(a) = .99 and P(b) = .01

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