UE 803 - Data Science for NLP

Lecture 11: Lexical Semantics

Claire Gardent - CNRS / LORIA

Lexical Semantics

What is the meaning of a word ?

How can we identify words with similar meanings ?

Two main approaches

- Relational
- Distributional

Relational Lexical Semantics

- Based on lexical relations
  - synonymy
  - antonymy
  - hyperonymy
  - etc.
- Modelled in databases such as WordNet, BabelNet ...

Distributional Lexical Semantics

- Similar words occur in similar context

John eats an apple
Peter eats a pear
The apple is ripe.
The pear is ripe.

- Words are represented by vectors
- Similar words have vectors that are close in space
- Prevails in today's research and applications

Relational Lexical Semantics

Lexical Databases

- Lexicons, thesauri, ontologies, lexical network
- Explicit representations of lexical relations between words
  E.g., "violin" and "fiddle" are synonyms
- Constructed manually
- E.g., WordNet, BabelNet, Framenet, Verbnet

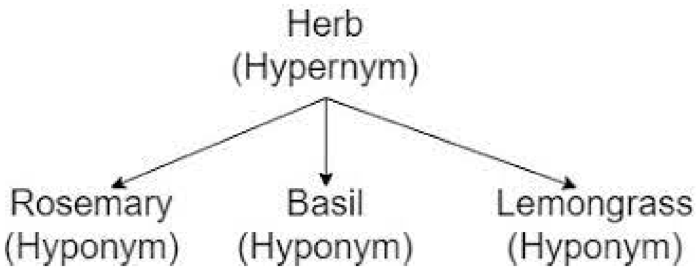
Lexical Relations

Synonymy (similar meaning)

- violin, fiddle

Antonymy (opposite meaning)

- rich, poor



Hypernymy (more generic)

- fiddle is a hypernym of Stradavarisu

Hyponymy (more specific)

- Stradavarisu is a hyponym of fiddle

Meronymy (part-of)

- bow is a meronym of fiddle

WordNet Search - 3.1

- WordNet home page - Glossary - Help

Word to search for: violin [Search WordNet]

Display Options: (Select option to change) [Change]

Key: "S:" = Show Synset (semantic) relations, "W:" = Show Word (lexical) relations

Display options for sense: [lexical file number] (gloss) "an example sentence"

Noun

- [06] S: (n) **violin**, fiddle (bowed stringed instrument that is the highest member of the violin family; this instrument has four strings and a hollow body and an unfretted fingerboard and is played with a bow)
  - *direct hyponym* / *full hyponym*
    - [06] S: (n) Amati (a violin made by Nicolo Amati or a member of his family)
    - [06] S: (n) Guarnerius (a violin made by a member of the Guarneri family)
    - [06] S: (n) Stradavarius, Strad (a violin made by Antonio Stradivari or a member of his family)
  - *part meronym*
    - [06] S: (n) chin rest (a rest on which a violinist can place the chin)
    - [06] S: (n) fiddlestick, violin bow (a bow used in playing the violin)
  - *direct hypernym* / *inherited hypernym* / *sister term*
    - [06] S: (n) bowed stringed instrument, string (stringed instruments that are played with a bow) *"the strings played superlatively well"*
  - ***derivationally related form***
    - W: (n) violinist [Related to: violin] (a musician who plays the violin)
    - W: (v) fiddle [Related to: fiddle] (play the violin or fiddle)

Wordnet

- A database of lexical relations
- English Wordnet
  - 120K nouns, 12K verbs, 21K adjectives, 4K adverbs
  - accessible via NLTK
- WordNets available in many languages
  - globalwordnet
- BabelNet: aka multilingual WordNet (500 languages)

Issues with Lexical Databases

- Manually constructed
    - Costly: requires time and expertise
- Language is dynamic
    - New words, new senses need to be added continuously

Distributional Semantics

Distributional Semantics

Distributional

"You shall know a word by the company it keeps" (Firth, 1957)

- Words which appear in similar contexts have similar/related meanings

Vector Based

- Words are represented by ***vectors***
- These vectors capture the ***contexts*** in which each word frequently occurs
- Words whose vectors are ***close in space*** have similar/related meanings

Information Retrieval vs. Lexical Semantics Vectors

Information Retrieval Document Vectors

IR: Find all documents which satisfies a query
E.g., query = Eiffel tower. Find all documents which are about the Eiffel tower.

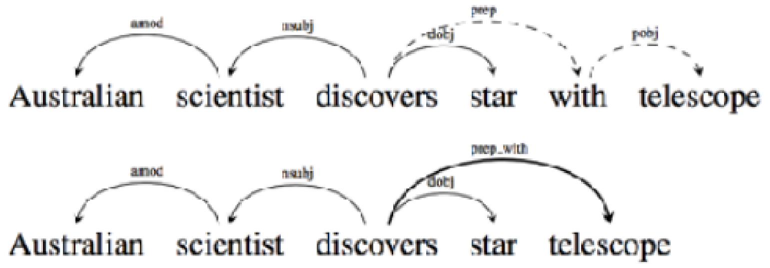
- Context = document
- A document vector represents the words occurring in that document
- Two documents are similar if they contain similar words
- A document is similar/relevant to a query if its vector is similar to the query vector

Lexical Semantics Word Vectors

Lexical Semantics: models the meaning of words
E.g., violin is similar to fiddle

- Context = sentence or word window
- A word vector represents the typical contexts in which a word occurs.
- The context used varies.
  It can be the sentence, a window of words (e.g., 2 words to the left and two words to the right).
  It could also be the syntactic context (the dependents).
- Two words are similar if they occur in similar context (they frequently co-occur with the same words, they have similar neighbours)

Syntactic Context



WORD	CONTEXTS
australian	scientist/amod^{-1}
scientist	australian/amod, discovers/nsubj^{-1}
discovers	scientist/nsubj, star/dobj, telescope/prep_with
star	discovers/dobj^{-1}
telescope	discovers/prep_with^{-1}

Lin 1998; Levy and Goldberg 2014

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

IR Example

Term-Document Matrix for Shakespeare plays (IR)

	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

- Each row represents a word
- Each column represents a document
- Each cell indicates how many times a word occur in a document
- The document vectors can be used to identify similar documents

Similar Documents

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

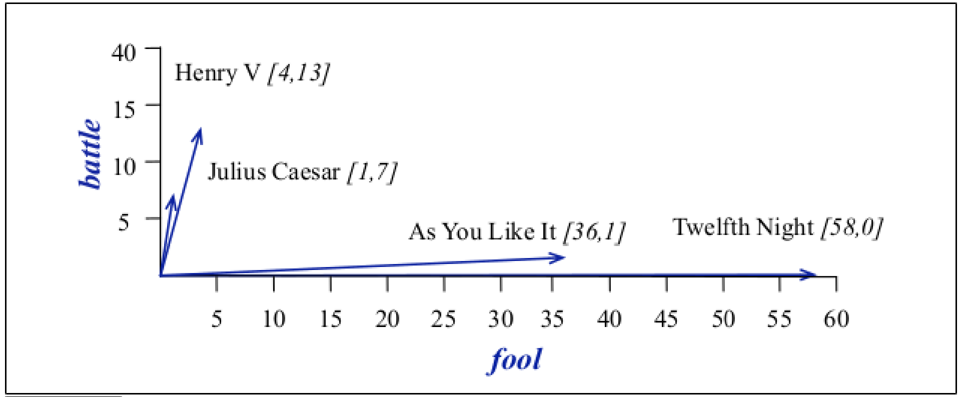


Figure 6.4 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

- JC and HV vectors are similar (similar values for battle and fool) and close in space. The two documents are similar
- AYLI and TN vectors are similar and close in space. The two documents are similar

Information Retrieval

Given a query and a collection of document, return documents that are relevant to query

- Each document is a vector of terms
- The query is also a vector of terms
- Retrieves the document whose vector is closest (has highest cosine value) with the query vector

Term-Document Matrix for Lexical Semantics

	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

Figure 6.5 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each word is represented as a row vector of length four.

- Similar words which occur in similar contexts (documents) have similar vectors
- Their vectors are close in space (high cosine, low angle)

Term-Term Matrix (NLP)

- In NLP, we use a ***term-term*** rather than a term-document matrix.
- the term-term matrix can be derived from the document matrix (see next slides)
- Each cell then indicates ***the number of times two terms co-occur*** in a document/sentence/window of words.

	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	...

Figure 6.6 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

Similar Words

	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	...

Figure 6.6 Co-occurrence vectors for four words in the Wikipedia corpus, showing six of the dimensions (hand-picked for pedagogical purposes). The vector for *digital* is outlined in red. Note that a real vector would have vastly more dimensions and thus be much sparser.

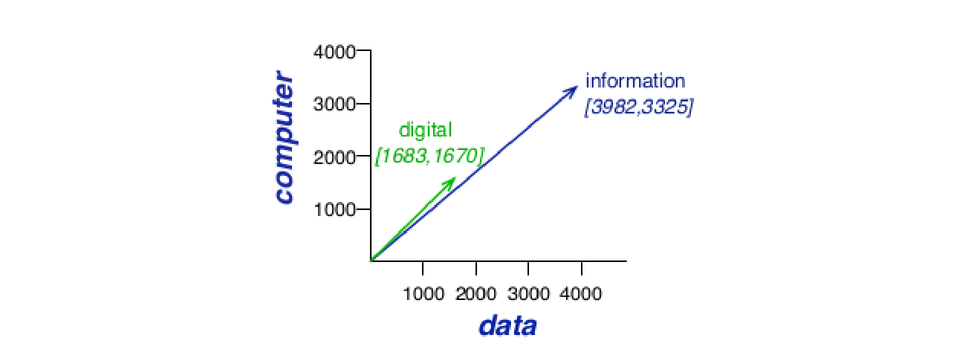


Figure 6.7 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *computer*.

- "cherry/strawberry" co-occur with "pie/sugar"
- "information/digital" co-occur with "computer/data/result"

Matrix Shape

- The shape (m,n) indicates its number of rows (m) and of columns (n)
- The matrix below has shape $(2, 3)$

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}$$

Matrix Transpose

$$\begin{bmatrix} 6 & 4 & 24 \\ 1 & -9 & 8 \end{bmatrix}^{T} = \begin{bmatrix} 6 & 1 \\ 4 & -9 \\ 24 & 8 \end{bmatrix}$$

- The transpose of a matrix transposes rows to columns and columns to rows
- The transpose of a matrix X with shape (m,n) has shape (n,m).
- Here the inital matrix (left) has shape $(2, 3)$ and its transpose (right) has shape $(3, 2)$

Matrix Multiplication

- Two matrix X and Y can only be multiplied if X is of shape (m,**n**) and Y of shape (**n**,k)
- Multiplying a matrix of size (m, n) by a matrix of size (n, m) yields a matrix of size (n, n)
- Here we obtain a matrix of size $(2, 2)$ by multiplying a matrix of size $(2, 3)$ with a matrix of size $(3, 2)$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} \times \begin{bmatrix} 7 & 8 \\ 9 & 10 \\ 11 & 12 \end{bmatrix} = \begin{bmatrix} 58 & 64 \\ 139 & 154 \end{bmatrix}$$

Converting a term-document matrix to a term-term matrix

The term-document matrix X has shape (t, d)

- t: the number of terms/rows
- d: the number of documents/columns

The transpose $X^\top$ of X has shape (d, t)

By multiplying X with $X^\top$, we obtain a ***term-term matrix*** Y of size (t, t)

$$Y_{(t,t)} = X_{(t,d)} \times X^\top_{(d,t)}$$

Calculating Similarity

Similarity

- If words are represented by vectors, then two words are semantically related if their vectors are similar
- Vectors that are similar are close to each other
- There are several possible ways of computing the distance or similarity between two vectors: Manhattan, Euclidean, Dot Product, Cosine
- In NLP, dot product and ***cosine*** are frequently used.

Dot Product

Also called inner product

$$v.w = \sum_{i=1}^{n} v_i w_i = v_1 w_1 + v_2 w_2 + \ldots + v_n w_n$$

- the dot product of two vectors is high when the two vectors have large values in the same dimensions i.e., when they often cooccur with the same words (***similar words***)
- it is low when the two vectors have zeros in different dimensions (***dissimilar words***)

Example

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

dotproduct(cherry,information) = (442 x 5) + (8 x 1683) + (2 x 1670) = 19014

dotproduct(digital,information) = (5 x 5) + (3982 x 1683) + (3325 x 1670) = 12,254,481

Dot Product and Vector Length

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

- the dot product of two vectors is higher for longer vectors i.e., vectors with high values in each dimension
- More frequent words co-occur with more words and thus have longer vectors
- So the dot product will be higher for frequent words
- However ***similarity between two words is independent of their frequency***

Cosine and Dot Product

- To remove the length impact, we can normalise the dot product of two vectors for vector length by dividing the dot product by the lengths of these two vectors.

$$\frac{v.w}{\mid v \mid\mid w \mid}$$

- This normalised dot product is in fact the ***cosine*** of the angle (θ) between the two vectors

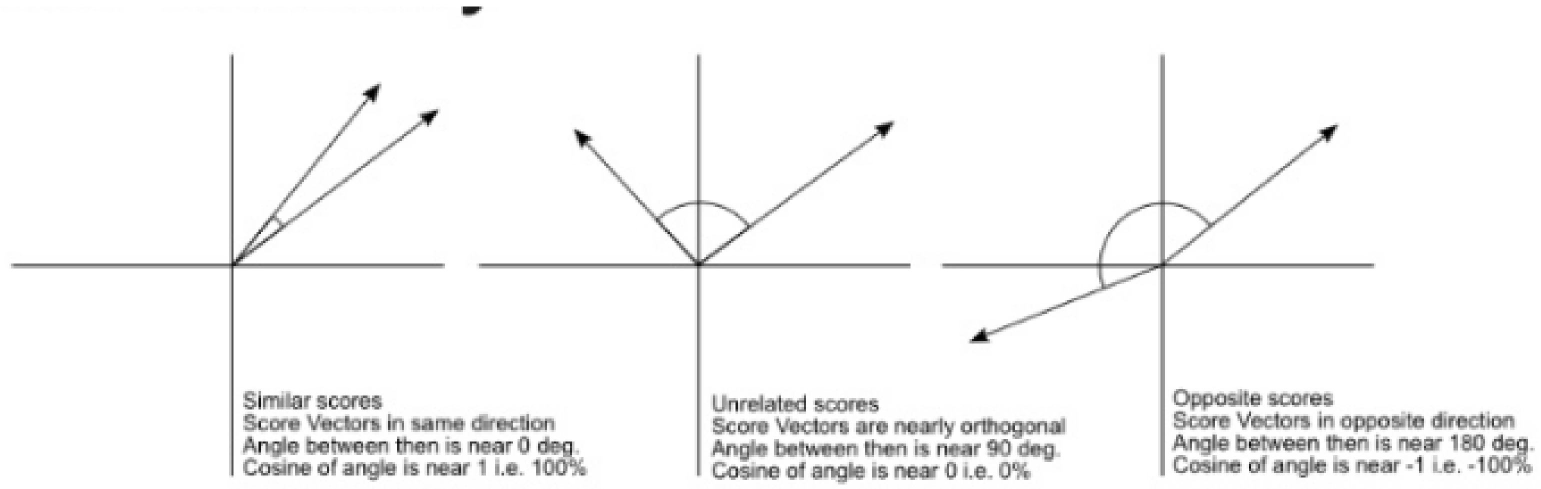
$$\frac{v.w}{\mid v \mid\mid w \mid} = cos(\theta)$$

- So the cosine provides us with a similarity metrics for two (word) vectors and therefore for the similarity between two words (since word meanings are represented by vectors)

Cosine

- $\theta \approx 0, Cosine \approx 1$: the words are similar
- $\theta \approx 90, Cosine \approx 0$: the words are dissimilar
- $\theta \approx 180, Cosine \approx -1$: the words are similar but opposite (antonyms)

$$cos(U,V) = \frac{\sum_{i=1}^{N} U_i V_i}{\sqrt{\sum_{i=1}^{N} U_i^2}\sqrt{\sum_{i=1}^{N} V_i^2}}$$



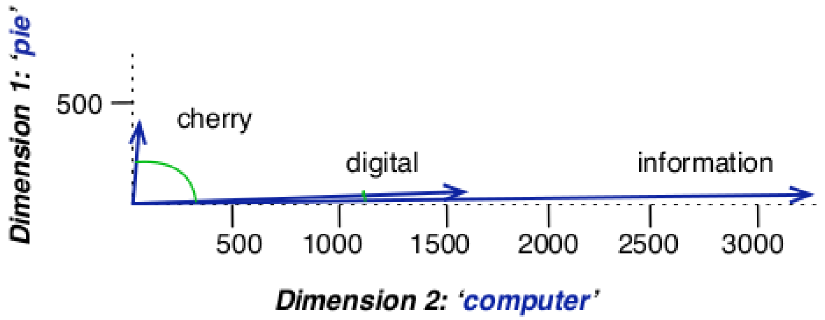
Example

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

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

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

- "cherry" frequently cooccurs with "pie"
- "digital" and "information" frequently cooccur with "computer"



The "cherry" vector is almost orthogonal ($\theta \approx 90, Cosine \approx 0$) to the "digital/information" vectors: "cherry" and "digital/information" have very dissimilar meanings
The "digital" and "information" vectors are very close ($\theta \approx 0, Cosine \approx 1$): "digital" and "information" have related meanings

Vector Components

Vector components

Various metrics can be used as vector components

- Frequency
- Tf-Idf, Term-Frequency x Inverse Document Frequency
- PPMI, Point Wise Mutual Information between co-occuring tokens

Frequency

	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

Figure 6.5 The term-document matrix for four words in four Shakespeare plays. The red boxes show that each word is represented as a row vector of length four.

- Words that frequently cooccur with all other words are not very informative
- Here "good" is frequent but does not allow us to distinguish similar documents from dissimilar ones (because it occurs in all documents)

TF-IDF Score

- Frequent words are not all informative (e.g., "the, good")
- Informative words are ***frequent in some documents*** (term frequency) but ***not in all*** (inverse document frequency)

$$tfidf(t, d) = tf(t, d) \times idf(t)$$

- $tf(t, d)$: term frequency, the frequency of a term ***in a document***
- $idf(t)$: inverse document frequency, how often a words occurs in ***the document collection***

Term Frequency (TF)

$$tf(t,d) = log(count(t,d) + 1)$$

We use the log to squash the values.

The intuition is that 100 coocurrences of word w with $w1$ does not make $w1$ more likely to be relevant to the meaning of w.

We add +1 because log(0) is undefined.

Inverse Document Frequency

$$idf(t) = \frac{N}{df(t)}$$

- Gives a higher weight to words which occurs in few documents
- N, the total number of documents
- $df(t)$, the number of documents in which t occurs

The fewer documents in which t occurs the higher its inverse document frequency

Frequency Matrix

	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

Figure 6.2 The term-document matrix for four words in four Shakespeare plays. Each cell contains the number of times the (row) word occurs in the (column) document.

Tf-Idf Matrix

	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

Figure 6.9 A tf-idf weighted term-document matrix for four words in four Shakespeare

"good" is no longer relevant in determining similarity.

PMI - Pointwise Mutual Information

- Measure of how much more two words co-occur than would be if they were independent (unrelated)

$$PMI = log2 \frac{P(w,c)}{P(w)P(c)}$$

- $P(w,c)$: how often two words w,c cooccur
- $P(w)P(c)$: how often two words w,c are expected to cooccur if they were independent

PPMI replaces negative values with zero:

$$PPMI = max(log2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

	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

$$P(info, data) = 3982/11716 = .3399$$

$$P(info) = 7703/11716 = .6575$$

$$P(data) = 5673/11716 = .4842$$

$$PPMI = max(log2 \frac{P(w,c)}{P(w)P(c)}, 0)$$

$PPMI(info, data) =$
$log2(.3399/(.6575 \times .4842)) = .0944$

Frequency Matrix

	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

PPMI Matrix

	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

- "cherry/strawberry" highly related to "pie/sugar"
- "digital/information" mildly related to "computer/data"

Dimensionality Reduction

- In a *term-term matrix* there are typically as many columns as there are words in the corpus being studied (called the ***vocabulary***)
- Hence the word vectors are very large
- They are also very sparse (lots of 0s)
- Dimensionality reduction reduces the size of the vectors by computing from the initial data a smaller matrix where the rows are the words and the columns some latent dimensions (topics).

SVD (Singular Value Decomposition)

From Words to Concepts

SVD can be used to decompose the word/document matrix into 3 matrices

$$A = U \times \Sigma \times V^T$$

These three matrices can be seen as highlighting hidden (latent) semantic dimensions (concepts) associated with each word

- The matrix U connects words/documents to concepts
- Σ represents the strengths of the concepts
- V connects concepts to words/documents

SVD Example

- M: (people $\times$ films)
- U: (people $\times$ latent concepts)
  E.g., scifi and romance
  - Joe (1st row) likes scifi
  - Jane (last row) likes romance
- Σ: weights (in decreasing order)
- V: (latent concepts $\times$ films)
  - First 3 films are about scifi, the last two about romance

	Matrix	Alien	Star Wars	Casablanca	Titanic
Joe	1	1	1	0	0
Jim	3	3	3	0	0
John	4	4	4	0	0
Jack	5	5	5	0	0
Jill	0	0	0	4	4
Jenny	0	0	0	5	5
Jane	0	0	0	2	2

Figure 11.6: Ratings of movies by users

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 0 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 0 & 0 & 2 & 2 \end{bmatrix}}_{M} = \underbrace{\begin{bmatrix} .14 & 0 \\ .42 & 0 \\ .56 & 0 \\ .70 & 0 \\ 0 & .60 \\ 0 & .75 \\ 0 & .30 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} .58 & .58 & .58 & 0 & 0 \\ 0 & 0 & 0 & .71 & .71 \end{bmatrix}}_{V^{\mathrm{T}}}$$

SVD

Using SVD to Create Smaller, Denser Word Vectors

- In a complete SVD for a matrix, U and V are typically as large as the original.
- To use fewer columns for U and V, delete the columns corresponding to the smallest singular values from U, V , and Σ (This choice minimizes the error in reconstructing the original matrix).
- Words are then represented by much denser vectors (fewer 0s)

SVD

Without reduction

$$\underbrace{\begin{bmatrix} 1 & 1 & 1 & 0 & 0 \\ 3 & 3 & 3 & 0 & 0 \\ 4 & 4 & 4 & 0 & 0 \\ 5 & 5 & 5 & 0 & 0 \\ 0 & 2 & 0 & 4 & 4 \\ 0 & 0 & 0 & 5 & 5 \\ 0 & 1 & 0 & 2 & 2 \end{bmatrix}}_{M'} =$$

$$\underbrace{\begin{bmatrix} .13 & .02 & -.01 \\ .41 & .07 & -.03 \\ .55 & .09 & -.04 \\ .68 & .11 & -.05 \\ .15 & -.59 & .65 \\ .07 & -.73 & -.67 \\ .07 & -.29 & .32 \end{bmatrix}}_{U} \underbrace{\begin{bmatrix} 12.4 & 0 & 0 \\ 0 & 9.5 & 0 \\ 0 & 0 & 1.3 \end{bmatrix}}_{\Sigma} \underbrace{\begin{bmatrix} .56 & .59 & .56 & .09 & .09 \\ .12 & -.02 & .12 & -.69 & -.69 \\ .40 & -.80 & .40 & .09 & .09 \end{bmatrix}}_{V^{\mathrm{T}}}$$

With reduction

$$\begin{bmatrix} .13 & .02 \\ .41 & .07 \\ .55 & .09 \\ .68 & .11 \\ .15 & -.59 \\ .07 & -.73 \\ .07 & -.29 \end{bmatrix} \begin{bmatrix} 12.4 & 0 \\ 0 & 9.5 \end{bmatrix} \begin{bmatrix} .56 & .59 & .56 & .09 & .09 \\ .12 & -.02 & .12 & -.69 & -.69 \end{bmatrix}$$

$$= \begin{bmatrix} 0.93 & 0.95 & 0.93 & .014 & .014 \\ 2.93 & 2.99 & 2.93 & .000 & .000 \\ 3.92 & 4.01 & 3.92 & .026 & .026 \\ 4.84 & 4.96 & 4.84 & .040 & .040 \\ 0.37 & 1.21 & 0.37 & 4.04 & 4.04 \\ 0.35 & 0.65 & 0.35 & 4.87 & 4.87 \\ 0.16 & 0.57 & 0.16 & 1.98 & 1.98 \end{bmatrix}$$

Neural Embeddings

Neural Word Embeddings

- Word vectors learned using neural models on very large corpora
- Unsupervised learning

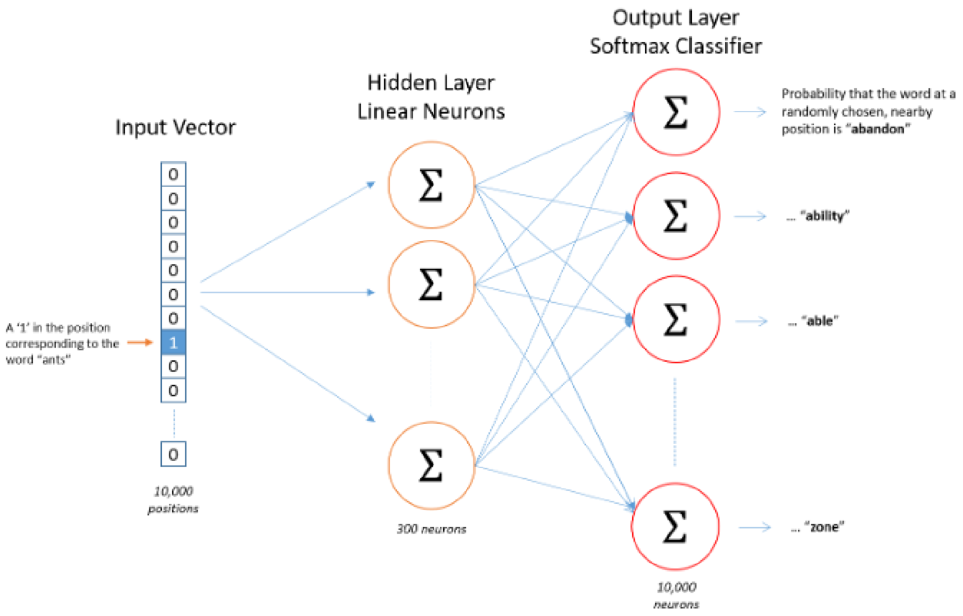
SkipGram Word Vectors (Intuition)

Word Vectors (***embeddings***) are learned as follows:

- ***Train a simple neural network with a single hidden layer*** to perform a certain task
  That neural network is not used for the task we trained it on!

- The goal is actually just to ***learn the weights of the hidden layer***

- These weights are the ***word vectors*** that we're trying to learn.



Training Task

Given a word w in a sentence, predict its context words.

For every word w_i in the vocabulary V, predict the probability of w_i being a context word for w

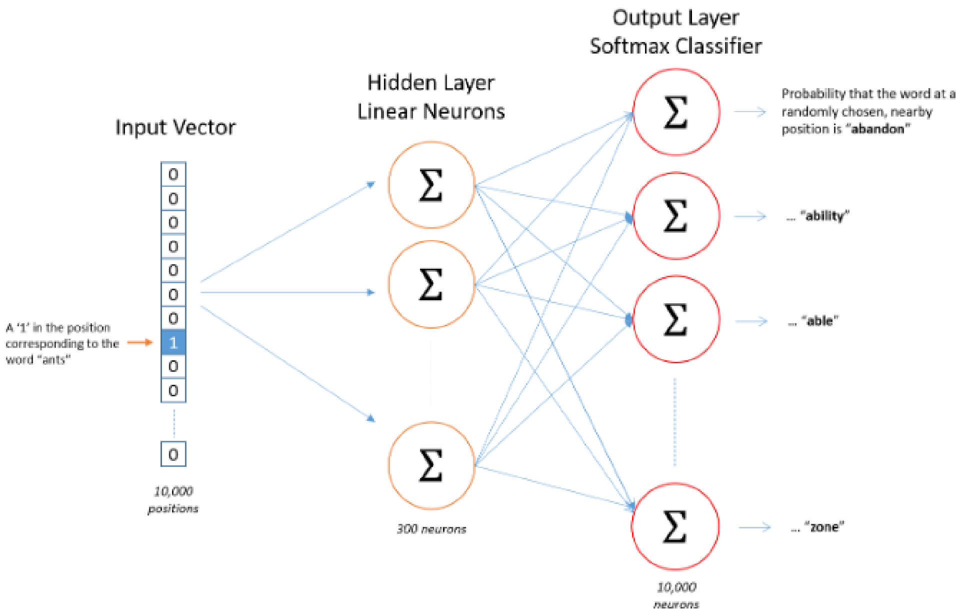
Train on raw text to learn co-occurence statistics

Input and Output

- Input layer = 1-hot vector representing the input word
  All 0 except one 1 which indicates the position of each vocabulary word in the vocabulary index
  Vocabulary size

- Hidden layer = embedding (word vector) size

- Output layer = Probability distribution over vocabulary
  For each word in the vocabulary, indicates the probability of that word being a context word for the input word
  Vocabulary size



Training

- On pairs of words (word, target)
  The correct output neuron corresponding to the target word is set to 1 and all others to 0.

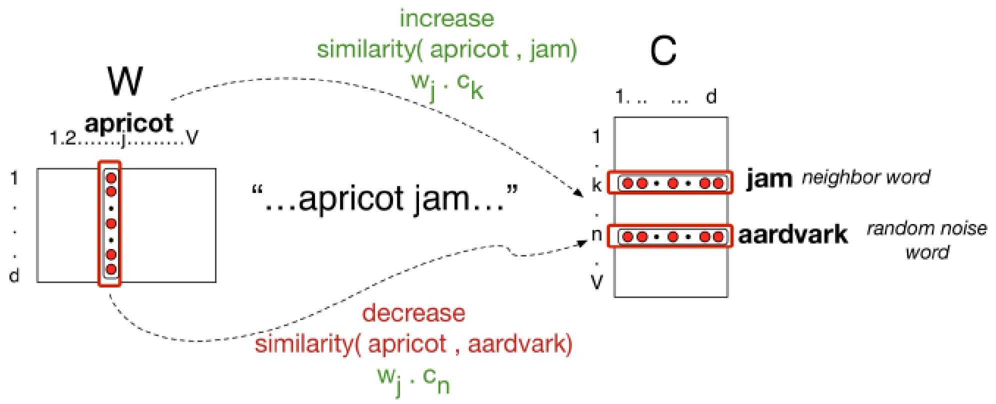
- When training this network on word pairs, the input is a one-hot vector representing the input word and the training output is also a one-hot vector representing the output word.

- Softmax regression (logistic regression on multiple classes)
  Estimates the probability of the target word being a context word for the input word

The number of classes is the size of the vocabulary !

Training (in practice)

- Classifying over the whole vocabulary is too hard
- Two methods to reduce computations
  - subsample frequent words
    - decreases the number of training examples
  - use negative sampling
    - reduce problem to binary classification
    - each step moves embeddings closer for context words and further apart for negative word



Summary

Treat the target word and a neighboring context word as positive examples.

1. Randomly sample other words in the lexicon to get negative samples.
2. Use logistic regression to train a classifier to distinguish those two cases.
3. Use the regression weights as the embeddings.

Neural Word Embeddings

Many neural word embeddings available

- Word2Vec Skipgram and CBOW word embeddings (Mikolov 2013)
- GLOVE
- ELMo (character based, bi-directional)
- BERT (uses Transformers Network, sub-words, bi-directional)
- OpenAI GPT (Transformers, Byte-Pair Encoding)
- LASER (cross-lingual)

  ....

Reading

- A brief history of Word Embeddings
- Another one

Lexical Semantics in Python

Creating the token/tokens matrix (1)

```
from sklearn.feature_extraction.text import CountVectorizer

# Create a frequency vectorizer object
# default unigram model
# Stop words will be removed
# CountVectorizer implements both tokenization and occurrence counting in a single class
count_model = CountVectorizer(ngram_range=(1,1), stop_words = 'english')

# Convert documents to document/token matrix
# docs : list of strings
X = count_model.fit_transform(docs)

# Print out the document / token matrix
# use the todense() attribute to create the matrix view
print(X.todense())
```

To get tf-idf vectors use sklearn TfidfVectorizer instead of CountVectorizer

Two views of the same matrix

```
(0, 10)       1
(0, 11)       1
(0, 5)        1
(1, 11)       1
(1, 1)        1
(1, 3)        1
(2, 11)       1
(2, 8)        1
(2, 6)        1
(3, 2)        1
(3, 7)        1
(3, 9)        1
(4, 7)        1
(4, 4)        1
(4, 0)        1
```

```
[[0 0 0 0 0 1 0 0 0 0 1 1]
 [0 1 0 1 0 0 0 0 0 0 0 1]
 [0 0 0 0 0 0 1 0 1 0 0 1]
 [0 0 1 0 0 0 0 1 0 1 0 0]
 [1 0 0 0 1 0 0 1 0 0 0 0]]
```

Creating the token/tokens matrix (2)

To create a token co-occurence matrix, multiply the transpose of the documents/tokens matrix by the documents/token matrix

- shape of X: (#doc, #tokens)
- shape of X transpose: (#tokens, #doc)
- shape of X transpose * X : (#tokens, #doc) $\times$ (#doc, #tokens) = (#tokens, #tokens)

```
# Create the token co-occurence matrix
Xc = (X.T * X)
# Set the diagonal to 0  (else it will indicate the token count)
Xc.setdiag(0)
# Print out the token co-occurence matrix
print(Xc.todense()) # print out matrix in dense format
```

Display the token/token matrix

```
import pandas as pd
# get the tokens
names = count_model.get_feature_names()
# create a pandas frame whose content is the token co-occurence matrix
# and whose row and column headers are the tokens
# Note that the matrix input to Pandas must be in dense format
df = pd.DataFrame(data = Xc.todense(), columns = names, index = names)
df.head()
```

	john	novels	plays	read	shakespeare	wrote
john	0	1	0	1	0	1
novels	1	0	0	0	0	1
plays	0	0	0	0	1	1
read	1	0	0	0	0	0
shakespeare	0	0	1	0	0	1

Computing similarity between two word vectors

```
import numpy as np
np.dot(vector1, vector2)
```

Applying SVD decomposition to a word co-occurence matrix

```
import numpy as np
import math

# A is a word co-occurence matrix
# On large corpora, make sure to use the full_matrices=False (reduced SVD) option
# else processing will be very slow
U, S, Vt = np.linalg.svd(A,full_matrices=False)

# Keep only the first 50 dimensions of U as word vectors
U = U[:,:50]

# Printing out a word vector
# Define a dictionary mapping tokens to indice
word2index = dict(zip(A.index,range(vocab_size)))
# Print out the vector for "victory"
print(A[word2index['victory']])
```

Useful Links

- Manning and Jurafski's Chapter on Lexical Semantics
- Reader on SVD
- Blog with python code on dimensionality reduction: SVD and LDA