U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

M.EIC

Natural Language Processing

Henrique Lopes Cardoso

FEUP / LIACC

hlc@fe.up.pt

U.PORTO
FEUP FACULDADE DE ENGENHARIA UNIVERSIDADE DO PORTO

LIACC

Vectorized Representations of Words

lexical semantics, vector semantics, word embeddings

Word Meaning

- How can we represent the meaning of a word?

- Unigrams: words as strings of letters (symbolic representation)
  - Meaning of 'cat': CAT
  - Meaning of 'dog': DOG

- In vector space terms, each word would be represented as a 1-hot vector, with length $|V|$ (aka localist representation)
  - [0 0 0 0 … 0 0 0 1 0 … 0 0]

Word Meaning

- But how do we relate the meaning of words?
  - If we search for “Dell notebook battery size”, we also mean “Dell laptop battery capacity”
  - If we search for “Porto hostel”, we probably also want to get “Porto inn”

```
hostel = [ 0 0 0 1 0 0 0 0 … 0 0 ]
   inn = [ 0 0 0 0 0 0 0 1 … 0 0 ]
```

    - Orthogonal vectors

Lexical Semantics

- Lexical semantics is the linguistic study of word meaning
- A single word may have multiple senses: polysemy
  - *mouse* *bank* *slide* *câmara* *cadeira* *sol* *café*
  - NLP task: word sense disambiguation

- Synonymy: *couch/sofa*, *filbert/hazelnut*, *water/H_2O*, *sofá/canapé*, *banco/assento*
  - Relation between senses rather than words
  - Propositional meaning: words that are substitutable in a sentence without changing its truth conditions
  - No two words are absolutely identical in meaning

Lexical Semantics

- Similarity: *cat/dog*, *lunch/dinner*, *livro/caderno*, *rei/presidente*
  - Relation between words

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

SimLex-999

- Relatedness: *coffee/cup*, *scalpel/surgeon*, *cavalo/estribo*, *lápis/borracha*
  - Words that belong to the same semantic field
- Relations
  - Hyponymy/hypernymy (is-a): eagle-bird, bird-animal
  - Meronymy/holonymy (part-whole): finger-hand, engine-car
  - Antonymy (opposition): hot-cold, day-night

Lexical Semantics

- Semantic frames and roles: *commercial transaction* = {*buy, sell, pay, buyer, seller*}
  - Set of words that relate to the same type of event
  - *Sam bought the book from Ling* ≡ *Ling sold the book to Sam*

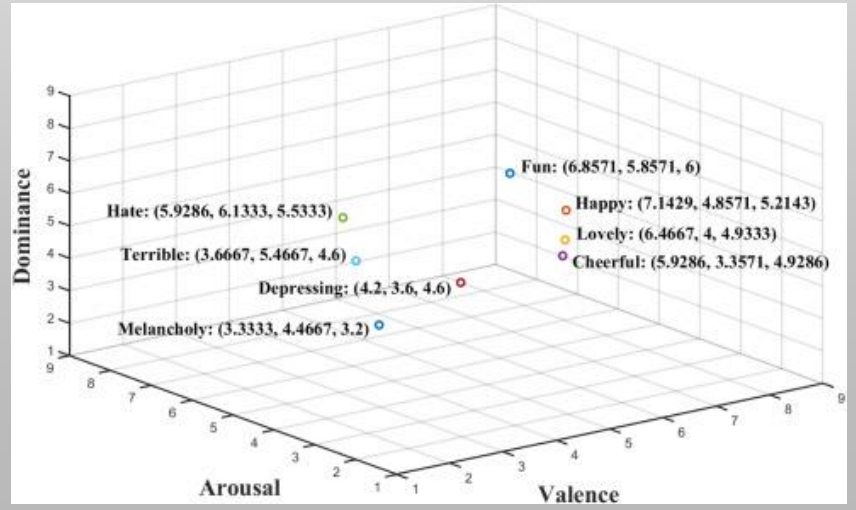
- Connotation: *happy, sad, great, love, terrible, hate*
  - Sentiment, affective meaning:
    - Valence (polarity): *happy, annoyed*
    - Arousal (intensity): *excited, calm*
    - Dominance (degree of control): *important, influenced*

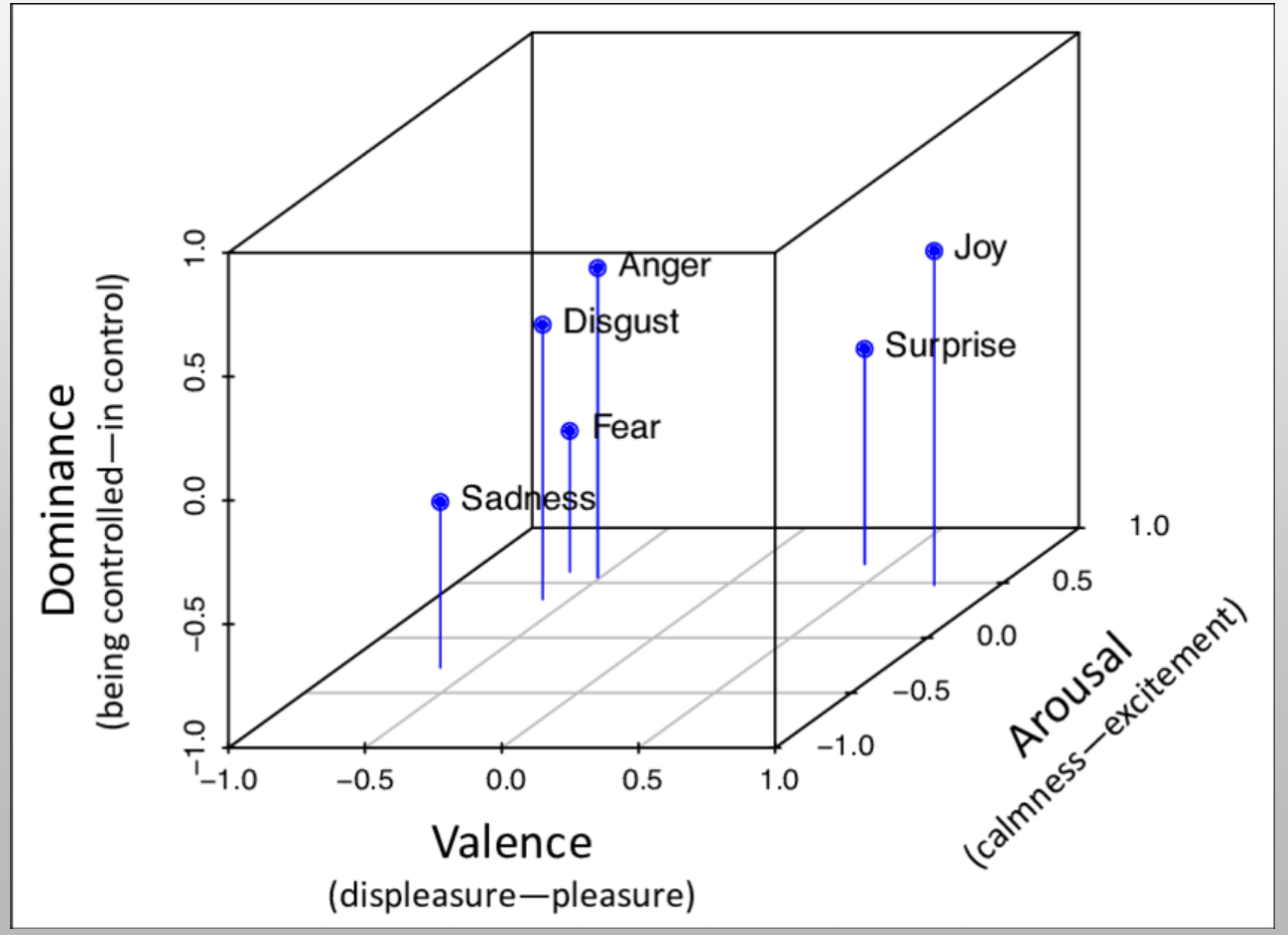
Entries with Highest and Lowest Scores in the VAD Lexicon

Dimension	Word	Score↑	Word	Score↓
valence	*love*	1.000	*toxic*	0.008
	happy	1.000	*nightmare*	0.005
	happily	1.000	*shit*	0.000
arousal	*abduction*	0.990	*mellow*	0.069
	exorcism	0.980	*siesta*	0.046
	homicide	0.973	*napping*	0.046
dominance	*powerful*	0.991	*empty*	0.081
	leadership	0.983	*frail*	0.069
	success	0.981	*weak*	0.045

Lexical Semantics

- We can represent part of a word's meaning as a point in 3D space! [Osgood et al., 1957]





Distributional Hypothesis

- Words that occur in similar contexts tend to have similar meanings
- Distributional hypothesis: there is a link between similarity in how words are distributed and similarity in what they mean
- Wittgenstein (1953): *"the meaning of a word is its use in the language"*

Distributional Hypothesis

- Firth (1957): *"You shall know a word by the company it keeps"*

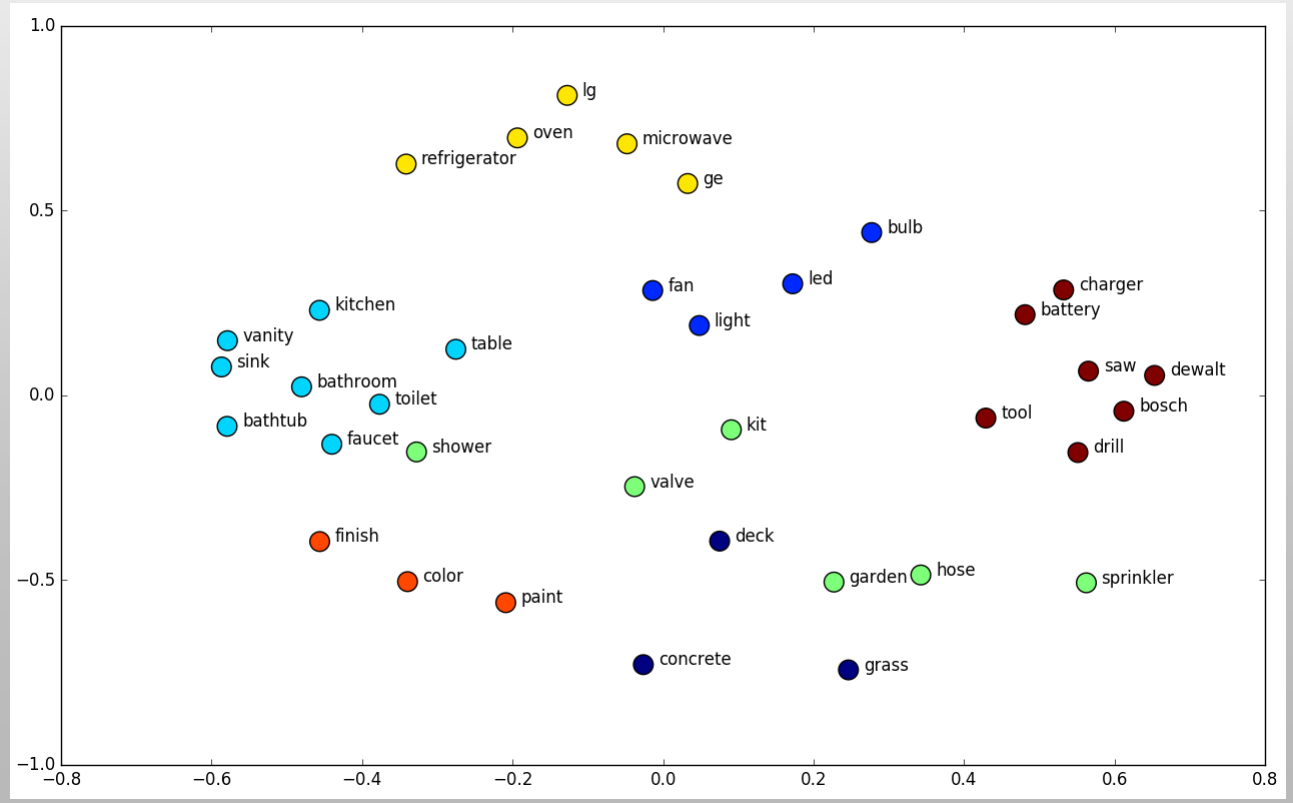
(6.1) Ongchoi is delicious sauteed with garlic.
(6.2) Ongchoi is superb over rice.
(6.3) ...ongchoi leaves with salty sauces...

(6.4) ...spinach sauteed with garlic over rice...
(6.5) ...chard stems and leaves are delicious...
(6.6) ...collard greens and other salty leafy greens

water spinach

Vector Semantics

- Defining the meaning of a word *w* as a point in N-dimensional space (a vector of numbers)
- Two-dimensional (t-SNE) projections of some word vectors (embeddings):



Vector Semantics

- Fine-grained model of meaning allowing for word similarity (and phrase similarity)

- Unsupervised learning of word embeddings
  - No need for complex labeling or supervision – simply observe words in large collections of text!
  - Learn representations of the meaning of words from their distributions in texts.

- Words that only appear in the test set of a text classification task
  - Only need words with similar meanings (vectors) to occur in the training set!
  - Of course, we still need to have embeddings of every test set word

Word Embeddings

- Two kinds of embeddings:
  - Sparse vectors (TF-IDF): words represented by a function of the counts of nearby words
  - Dense vectors (word2vec, …): trained representations

Document Vectors

- Term-document 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.

vocabulary

4-dimensional document vectors

comedies

- Similar documents tend to have similar words

Document Vectors

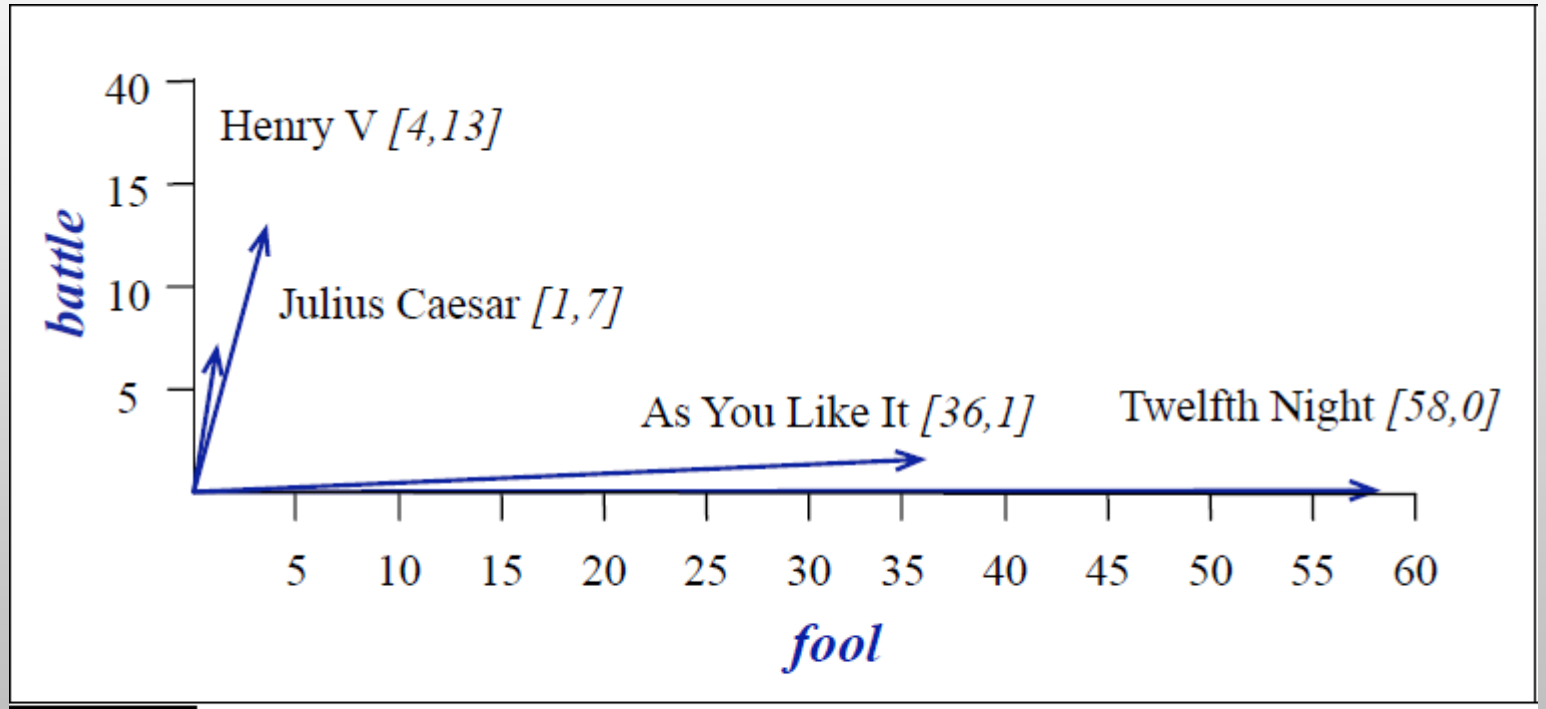


Figure 6.4 A spatial visualization of the document vectors for the four Shakespeare play documents, showing just two of the dimensions, corresponding to the words *battle* and *fool*. The comedies have high values for the *fool* dimension and low values for the *battle* dimension.

Word Vectors

- Word vector

	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.

- Represent the meaning of a word by the documents it tends to occur in
- Similar words tend to occur in similar documents

Word Context Vectors

- Context words: words before/after the target word, within a window
  - Window ±4:

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

  - Count how many times each word appears in the context of the target word

Word Context Vectors

- Term-context matrix: $|V| \times |V|$

	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 6.5 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.

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

Word Context Vectors

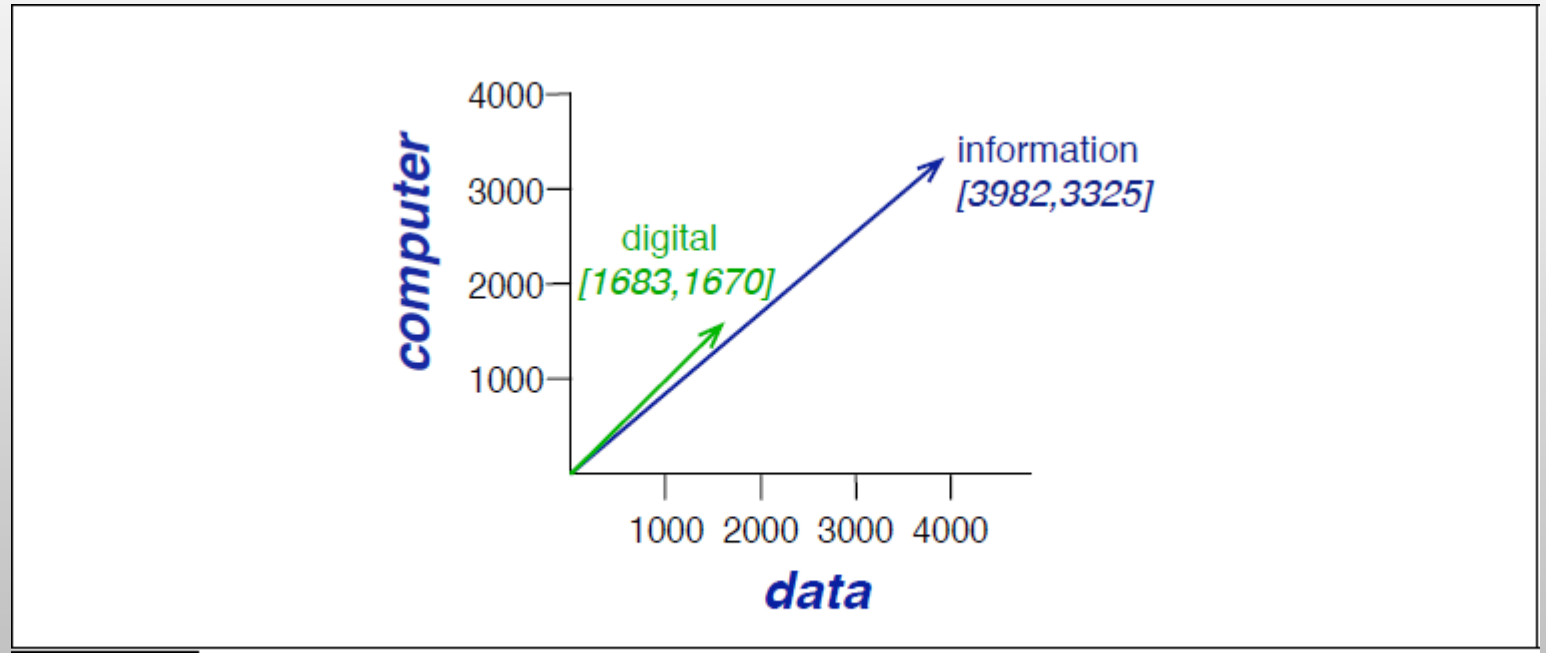


Figure 6.6 A spatial visualization of word vectors for *digital* and *information*, showing just two of the dimensions, corresponding to the words *data* and *computer*.

Similarity through the Dot Product

- Measuring similarity between two words
  - Given two same-dimensional vectors, compute their similarity

- Dot product

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

  - High when both vectors have large values in the same dimensions
  - If vectors have zero's in different dimensions (orthogonal vectors), their dot product will be ≈0
  - Favors long vectors (with higher values): will be higher for frequent words…

Cosine Similarity

- Geometric definition of the dot product

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

  - θ is the angle between a and b

- Normalized dot product = cosine of the angle between the two vectors

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

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

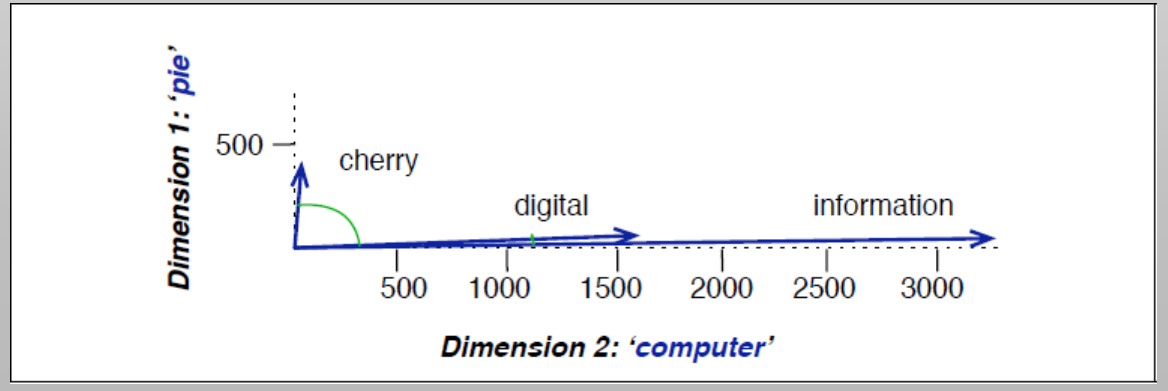
Cosine Similarity

- Which word is closer in meaning to *information*?

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

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

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



TF-IDF

- Raw frequency is not very informative and is usually a bad representation

- Frequency *is* useful:
  - **Words that occur nearby frequently** (*pie* nearby *cherry*) are **more important** than words that only appear once or twice
  - But **words that appear too often are unimportant**: words like *the* or *it* will tend to appear in the context of most words, and are thus uninformative

  → Need to balance these two perspectives

TF-IDF

- Term Frequency (TF): the frequency of a word in a document
  - $\mathrm{tf}_{t,d} = \mathrm{count}(t, d)$
  - Squashing the raw frequency: $\mathbf{tf}_{t,d} = \mathbf{log_{10}}\,(\mathbf{count}(t, d) + 1)$
- Inverse Document Frequency (IDF): inverse of the fraction of documents a term occurs in
  - $\mathrm{idf}_t = N/\mathrm{df}_t$, where N is the number of docs, and df_t is the number of docs in which term t occurs
  - $\mathbf{idf}_t = \mathbf{log_{10}}\,(N/\mathbf{df}_t)$
- TF-IDF
  - $w_{t,d} = \mathbf{tf}_{t,d} \times \mathbf{idf}_t$

TF-IDF

- Term-document 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

- TF-IDF weighted term-document 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

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

Applications of Similarity

- Comparing documents
  - Document representation: centroid of the vectors for all the words in the document

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

  - Similarity between two documents: $\cos(d_1, d_2)$
  - Information retrieval, plagiarism detection, news recommendation, …
- Tracking changes in word meaning
- Meanings of words in different corpora
- Finding similar words

Dense Vectors

- Representing words using short (50-1000) and dense (mostly non-zero values) vectors
- Dense vectors work better than sparse vectors in every NLP task
- Better capturing of synonymy

- Available tools and embeddings:
  - word2vec: continuous BoW and skip-gram architectures for computing vector representations of words
  - GloVe: based on ratios of word co-occurrence probabilities
  - fastText: subword models, summing embeddings of the bag of character n-grams that make up a word
  - …

word2vec

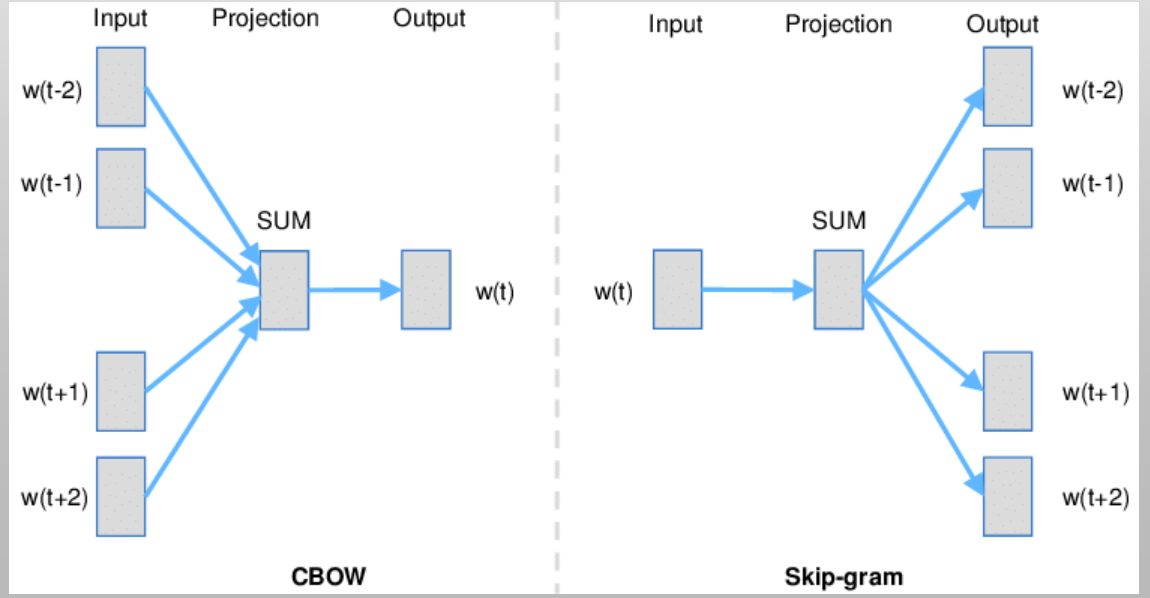
- Instead of counting, train a binary classifier to predict if a word *w* is likely to occur near a target word

- Use running text as "implicitly supervised" training data: words near the target word are positive cases

- We don't really care about this prediction task
  - ➢ Use the learned classifier weights as the word embeddings!

word2vec

- Two main algorithms:
  - Continuous Bag of Words (CBOW): predict target word from bag-of-words context
  - Skip-gram: predict context words given a target word

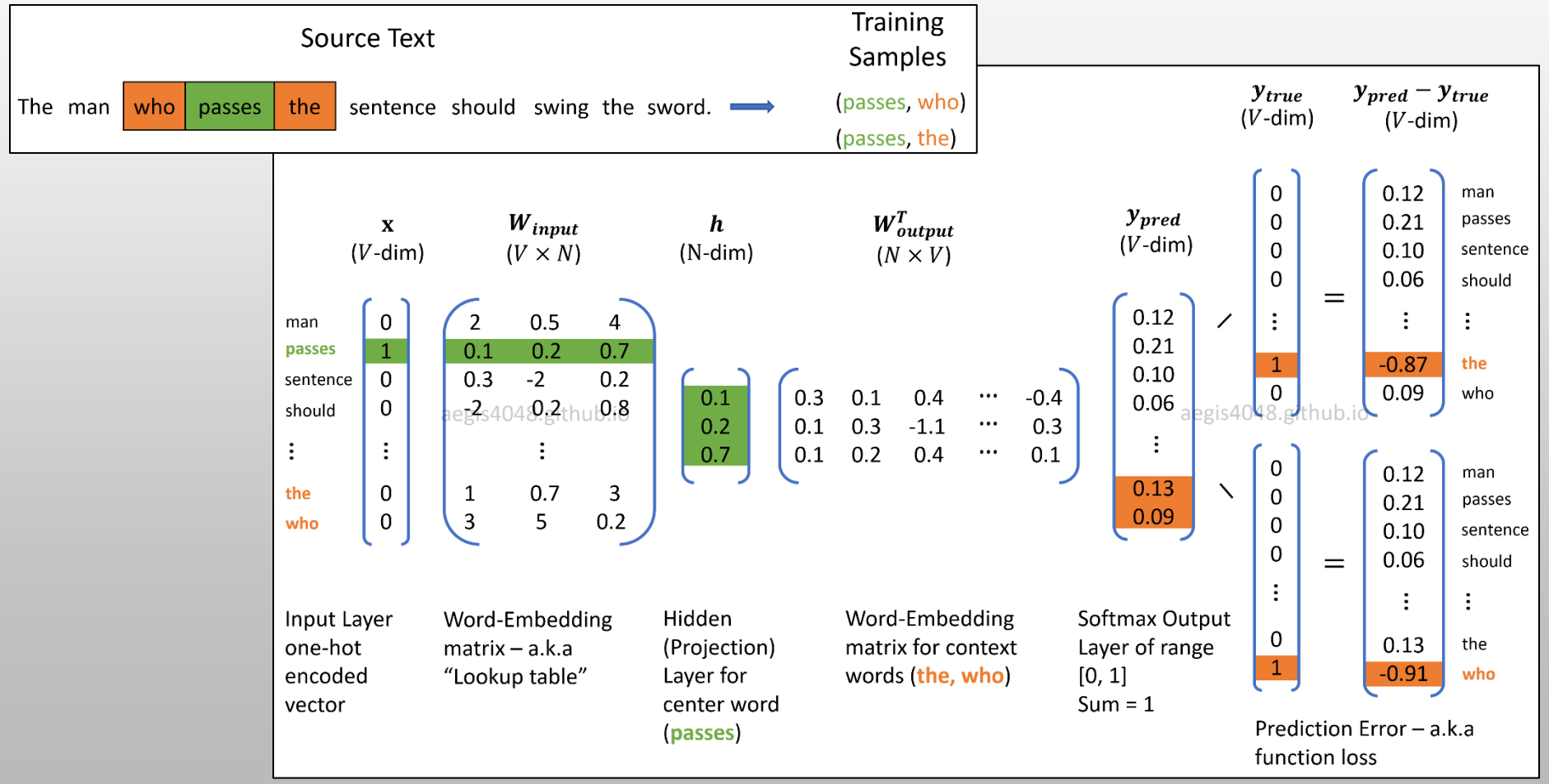


Skip-Gram Classifier

- Given a tuple (w, c), return the probability that c is a real context word for w
  - $P(+|w, c)$
  - $P(-|t, c) = 1 - P(+|t, c)$

- Base this probability on similarity via dot product (a word is likely to occur near the target if they have similar embeddings)
  - $Similarity(w, c) \approx c \cdot w$

Skip-Gram



https://aegis4048.github.io/demystifying_neural_network_in_skip_gram_language_modeling

Skip-Gram with Negative Sampling

- Using softmax in the output layer is computationally expensive: need to go through $|V|$ words!

$$p(w_{context}|w_{center}) = \frac{exp(W_{output_{(context)}} \cdot h)}{\sum_{i=1}^{V} exp(W_{output_{(i)}} \cdot h)}$$

- Rely instead on negative sampling

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

Skip-Gram with Negative Sampling

- Use a logistic (or sigmoid) function on the dot product: $P(+|w,c) = \sigma(c \cdot w) = \frac{1}{1+e^{-c \cdot w}}$
- Learning the embeddings:
  - Start with random embedding vectors for each word
  - Through SGD, shift the embedding of each word to be more like the context words and less like the negative examples

$$
\begin{aligned}
L_{CE} &= -\log\left[P(+|w,c_{pos})\prod_{i=1}^{k}P(-|w,c_{neg_i})\right] \\
&= -\left[\log P(+|w,c_{pos}) + \sum_{i=1}^{k}\log P(-|w,c_{neg_i})\right]
\end{aligned}
$$

Skip-Gram with Negative Sampling

- Context window in ±2:

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

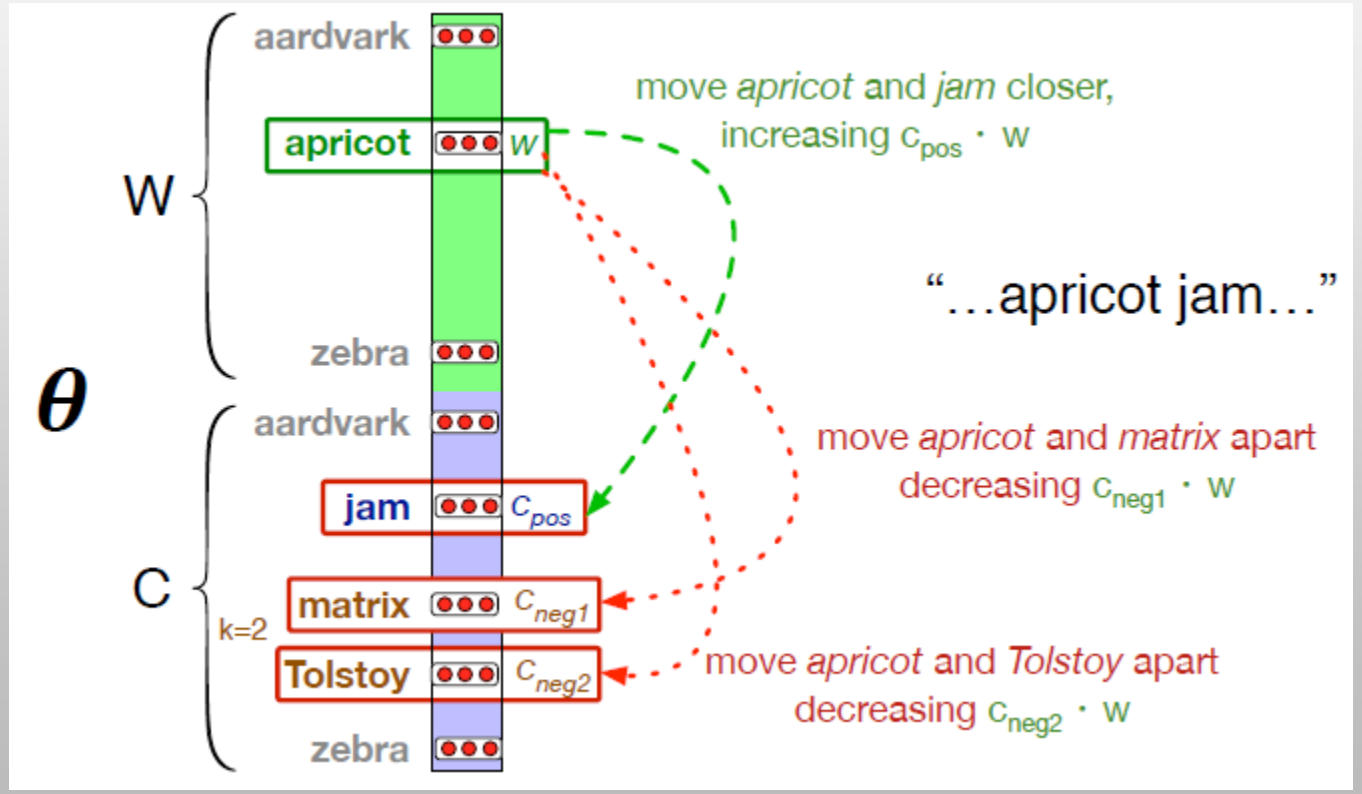
positive examples +

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

negative examples -

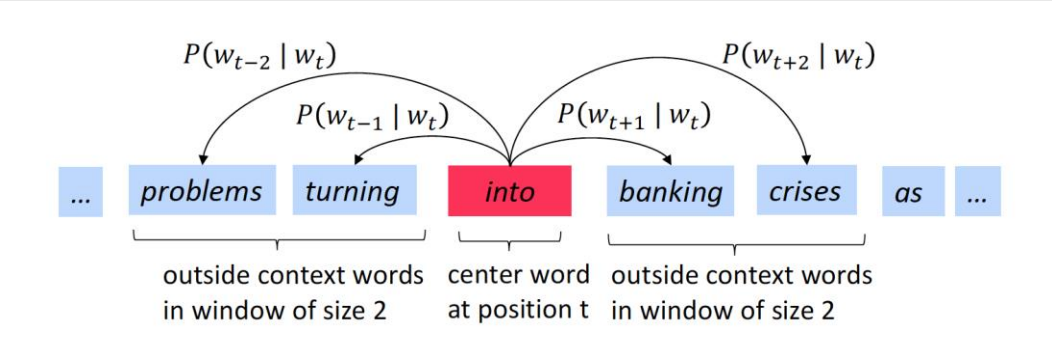
w	c_{neg}	w	c_{neg}
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

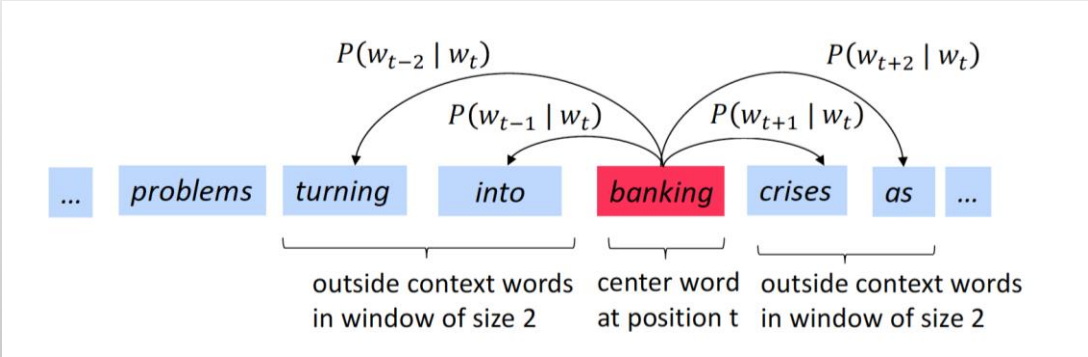
Skip-Gram with Negative Sampling



Two Embeddings for each Word

- Skip-gram learns two embeddings for each word: as target and as context





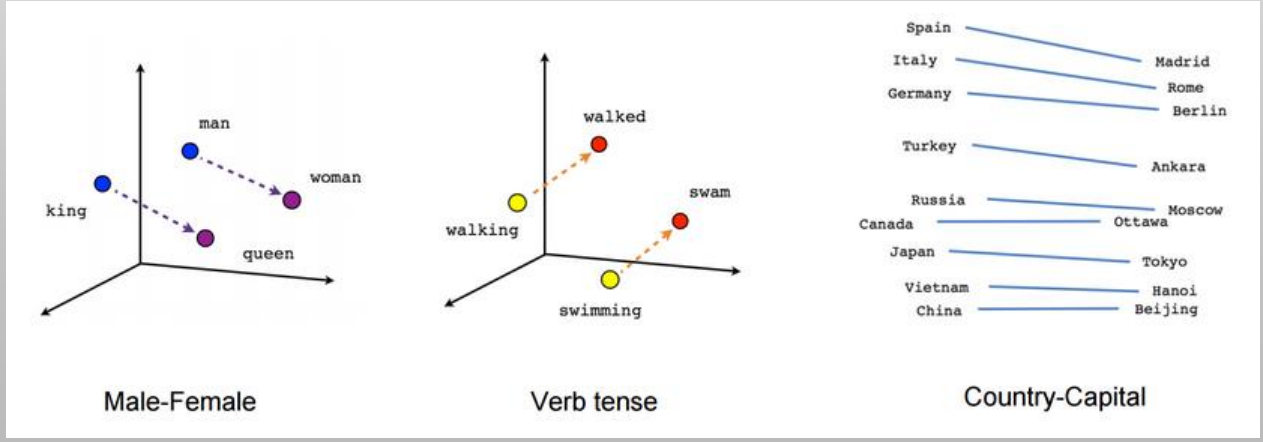
- Add them together, representing word i with the vector $w_i + c_i$
- Or simply throw away c_i

Semantic Properties of Embeddings

- The size of the context window matters
  - Narrower windows lead to more syntactic representations: the most semantically similar words tend to have the same POS (aka attributional similarity)
    - *Hogwarts* ±2: *Sunnydale*, *Evernight*
    - Potentially more suitable for lexical substitution tasks
  - Wider windows lead to topically related (but not necessarily similar) words having the highest cosine similarity
    - *Hogwarts* ±5: *Dumbledore*, *Malfoy*, *half-blood*

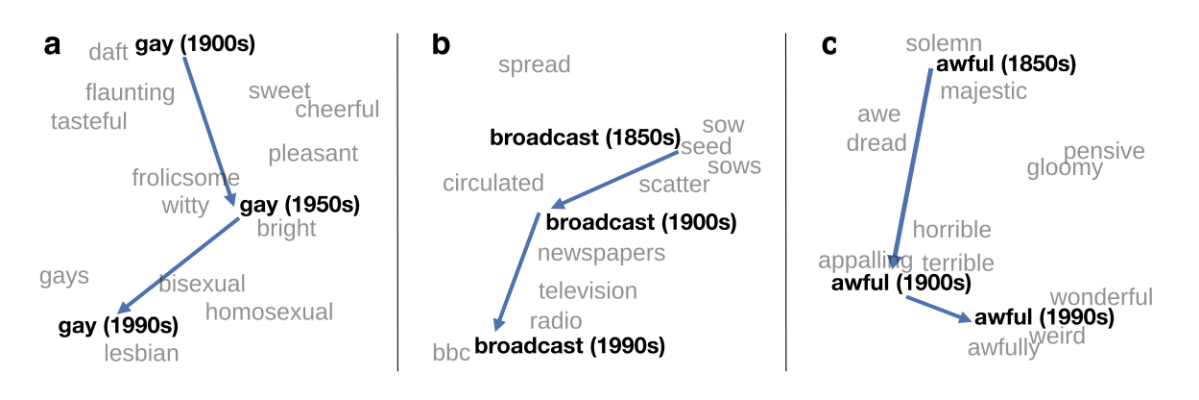
Semantic Properties of Embeddings

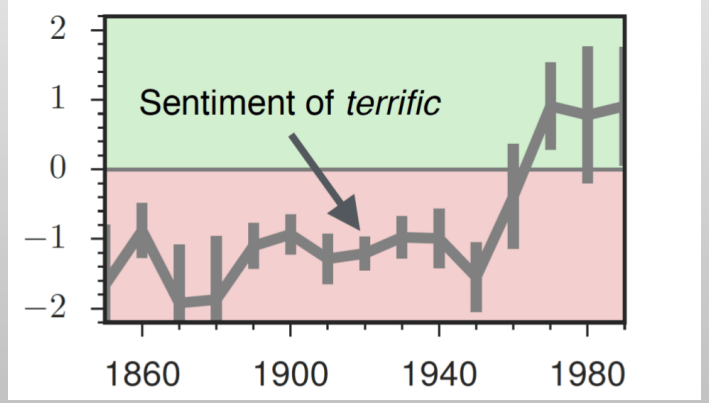
- Analogy: ability to capture relational meanings
  - The offsets between vector embeddings can capture some analogical relations between words
    - *king – man + woman ≈ queen*
    - *Paris – France + Italy ≈ Rome*



Historical Semantics

- How word meaning changes over time [Hamilton et al., 2016]





Bias in Embeddings

- Embeddings reproduce the implicit biases and stereotypes that are latent in the text [Bolukbasi et al., 2016]
  - 'man' is to 'computer programmer' as 'woman' is to 'homemaker'
  - 'father' is to 'doctor' as 'mother' is to 'nurse'

Extreme *she* occupations		
1. homemaker	2. nurse	3. receptionist
4. librarian	5. socialite	6. hairdresser
7. nanny	8. bookkeeper	9. stylist
10. housekeeper	11. interior designer	12. guidance counselor
Extreme *he* occupations		
1. maestro	2. skipper	3. protege
4. philosopher	5. captain	6. architect
7. financier	8. warrior	9. broadcaster
10. magician	11. figher pilot	12. boss

Gender stereotype *she-he* analogies.		
sewing-carpentry	register-nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	hairdresser-barber
Gender appropriate *she-he* analogies.		
queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

Evaluating Embeddings

- Extrinsic evaluation
  - Check improvements on the use of embeddings as features in an NLP task
- Intrinsic evaluation: compare embedding scores with human ratings (similarity)
  - WordSim-353: ratings for noun pairs (e.g. plane-car)
  - SimLex-999: quantifying similarity (e.g. cup-mug) rather than relatedness (e.g. cup-coffee)
  - TOEFL dataset: 80 questions on synonymy, each with a target word and 4 word choices
  - With context:
    - Stanford Contextual Word Similarity (SCWS): human judgments on pairs of words in their sentential context
    - Word-in-Context (WiC): pairs of sentential contexts for the same target word, with the same/different senses
    - Analogy tasks: *Athens is to Greece as Oslo is to* _____

The Python Notebook

word-embeddings_.ipynb

- Training word2vec embeddings
- Exploring embeddings
- Visualizing embeddings through tSNE

https://pypi.org/project/gensim/

word-embeddings-for-classification_.ipynb

- Using word embeddings as features: input truncation and averaging

https://scikit-learn.org/