Week 2
Natural Language Processing

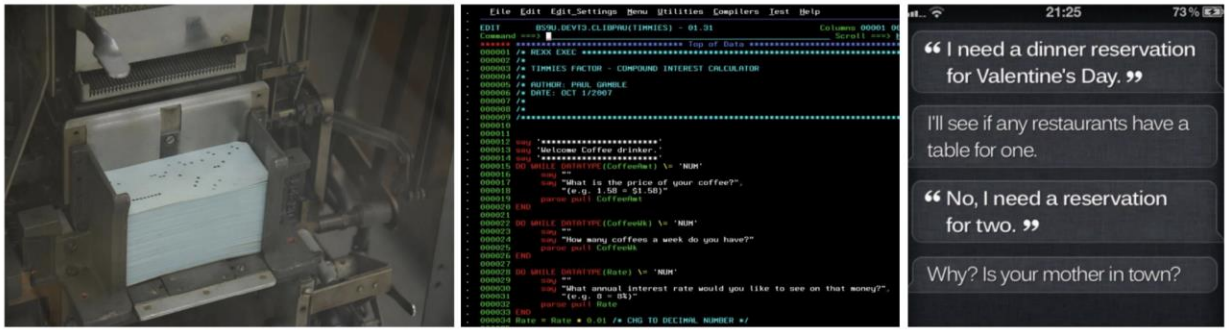
Content

- Introduction to Natural Language Processing (NLP)
- Vector Semantics
- Word Embeddings (Word2Vec)
- Sequence Labeling
- Sentiment Analysis
- Vector Semantics
- Python: NLP with SpaCy and NLTK

Introduction to Natural Language Processing (NLP)

What is Natural language processing ?

Natural language processing (NLP) is an interdisciplinary subfield of computer science and linguistics. It is primarily concerned with giving computers the ability to support and manipulate speech. Its core aim is to teach computers to understand and work with human language. This involves using rules, statistics, and advanced neural networks to process text and speech data.



~50-70s ~80s today

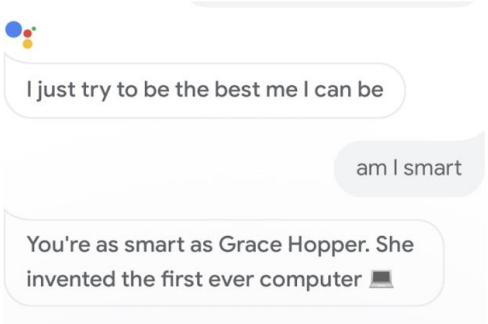
Communication With Machines

Introduction to Natural Language Processing (NLP)

Conversational Agents

- Speech recognition
- Language analysis
- Dialogue processing
- Information retrieval
- Text to speech

works with the Google Assistant



alexa Google Assistant

Introduction to Natural Language Processing (NLP)

Applications

- Machine Translation
- Information Retrieval
- Question Answering
- Dialogue Systems
- Information Extraction
- Summarization
- Sentiment Analysis
- ……

Core Technologies

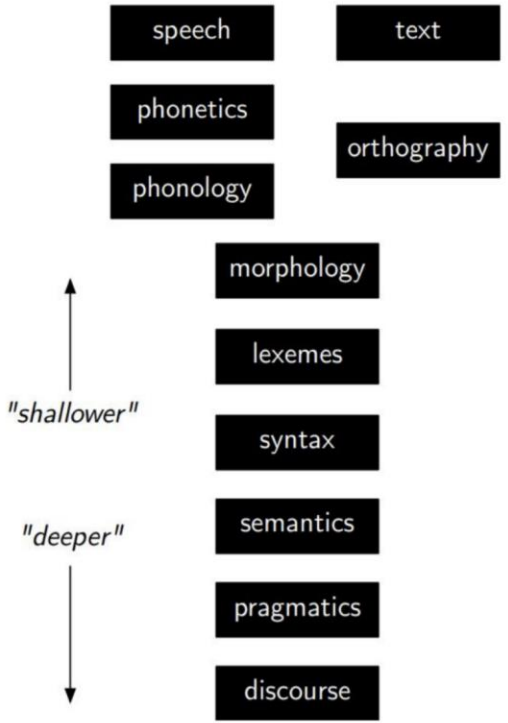
- Language modeling
- Part-of-speech tagging
- Syntactic parsing
- Named-entity recognition
- Word sense disambiguation
- Semantic role labeling
- ……

NLP lies at the intersection of computational linguistics and machine learning.

Introduction to Natural Language Processing (NLP)

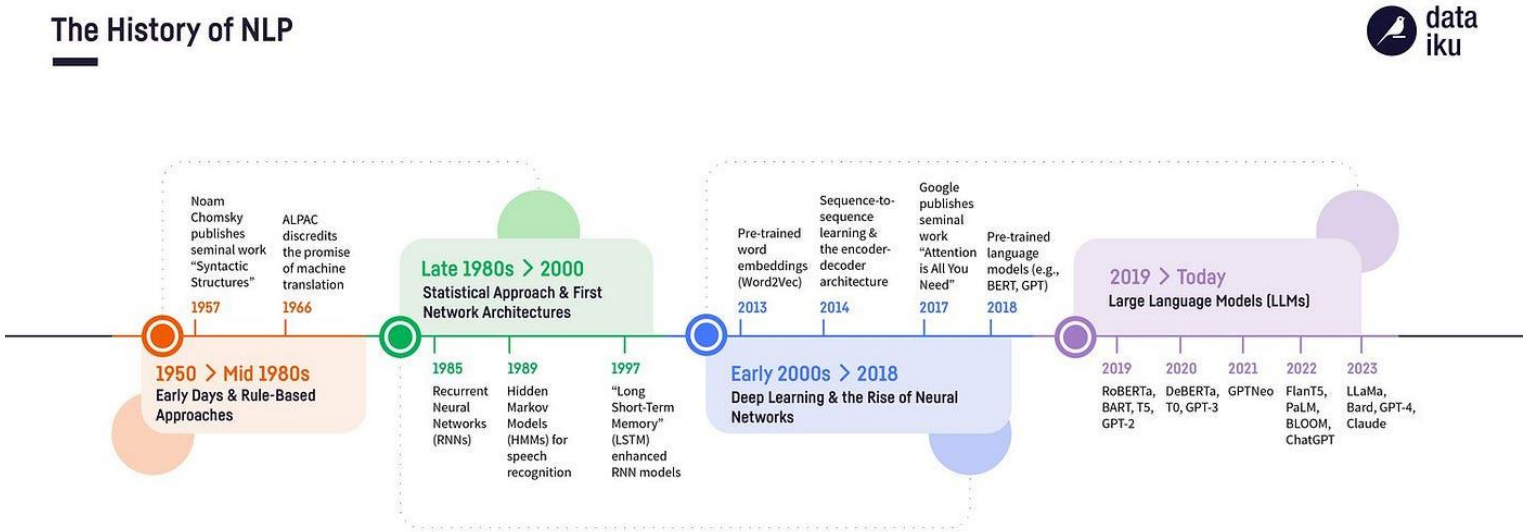
Level Of Linguistic Knowledge

- Phonetics & Phonology
  - Sound and speech patterns
- Morphology
  - Structure of words (morphemes)
- Syntax
  - Sentence structure and grammar
- Semantics
  - Meaning of words and sentences
- Pragmatics
  - Meaning in context (implied meaning)
- Discourse
  - Structure of larger text or dialogue
- World Knowledge
  - Common sense and real-world reasoning



Introduction to Natural Language Processing (NLP)

The Evolution of Natural Language Processing



Introduction to Natural Language Processing (NLP)

The Evolution of Natural Language Processing

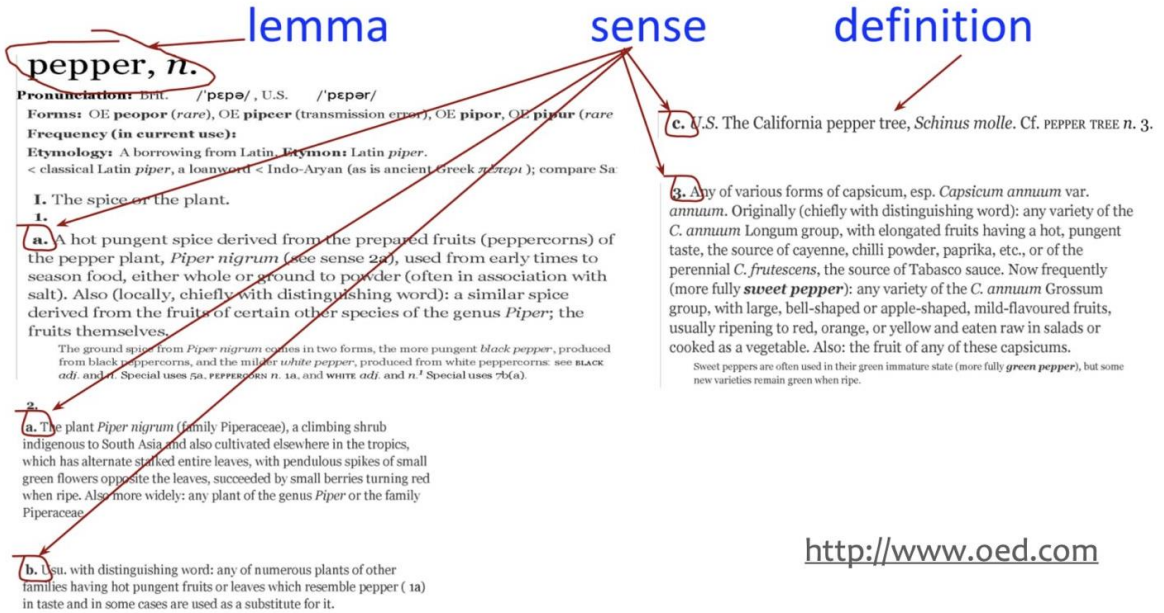
- **The Foundations (1950s-1960s)**
    - NLP began in the 1950s with early attempts at machine translation, inspired by Chomsky's grammar theories. Alan Turing's 1950 paper introduced the Turing Test, sparking interest in making machines capable of human-like conversation.
- **Rule-Based Systems (1960s-1970s)**
    - In the 1960s-70s, NLP relied on rule-based systems, using hand-coded grammar rules and dictionaries. Systems like SHRDLU could process and respond to complex natural language commands.
- **Statistical Revolution (1980s-1990s)**
    - By the 1980s, NLP embraced statistical methods like Hidden Markov Models, driven by the availability of large datasets and increased computing power. This shift enabled more accurate language processing.
- **Machine Learning and Deep Learning (2000s-Present)**
    - In the 2000s, machine learning and deep learning revolutionized NLP with word embeddings (Word2Vec, GloVe) and neural networks (RNNs, LSTMs). Pre-trained models like BERT and GPT brought major advancements in language tasks.
- **Modern Applications (2010s-Present)**
    - NLP powers modern applications like virtual assistants (Siri, Alexa) and multilingual models, driving human-like interactions across industries.
- **Ethical and Societal Implications**
    - NLP's rise raises concerns about bias, privacy, and misuse, requiring responsible AI development to ensure ethical use of these technologies.

Vector Semantics

Lexical Semantics

Q: How to represent the meaning of a word?

A: Words, lemmas, senses, definitions



http://www.oed.com

Vector Semantics

Lemma “Pepper”

- Sense 1:
  - Spice from pepper plant
- Sense 2:
  - The pepper plant itself
- Sense 3:
  - Another similar plant (Jamaican pepper)
- Sense 4:
  - Another plant with peppercorns (California pepper)
- Sense 5:
  - Capsicum (i.e., bell pepper, etc)

A sense or “concept” is the meaning component of a word

Vector Semantics

Lexical Semantics

Q: How should we represent the meaning of the word?

- Words, lemmas, senses, definitions
- Relationships between words or senses

Vector Semantics

Relation: Synonymity

- Synonyms have the same meaning in some or all contexts.
  - Filbert/hazelnut
  - Couch/sofa
  - Big/large
  - Automobile/car
  - Vomit/throw up
  - Water/H2O
- Synonyms have the same meaning in some or all contexts.
  - Note that there are probably no examples of perfect synonymy
  - Even if some aspects of meaning are identical
  - Still may not preserve the acceptability based on notions of politeness, slang, register, genre, etc.

Vector Semantics

Relation: Antonymy

- Senses that are opposites with respect to one feature of meaning
  - Otherwise, they are very similar!
    - Dark/light short/long fast/slow rise/fall
    - Hot/cold up/down in/out
- Many formally: antonyms can
  - Define a binary opposition or be at opposite ends of a scale
    - Long/short, fast/slow
- Be reverse:
  - Rise/fall, up/down

Vector Semantics

Relation: Similarity

- Words with similar meanings
- Not synonyms, but sharing some element of meaning
    - Car, bicycle
    - Cow, horse

Vector Semantics

Ask Humans How Similar 2 Words Are

Word 1	Word 2	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)

Vector Semantics

Relation: Word Relatedness

- Also called “word association”
- Words be related in any way, perhaps via a semantic field

A semantic field is a set of words which cover a particular semantic domain and bear structured relations with each other.

Vector Semantics

Semantic Field

- Hospitals
  - Surgeon, scalpel, nurse, anesthetic, hospital
- Restaurants
  - Waiter, menu, plate, food, menu, chef
- Houses
  - Door roof, kitchen, family, bed

A semantic field is a set of words which cover a particular semantic domain and bear structured relations with each other.

Vector Semantics

Relation: Word Relatedness

- Also called “word association”

- Words be related in any way, perhaps via a semantic field
  - Car, bicycle: similar
  - Car, gas: related, not similar
  - Coffee, cup: related, not similar

Vector Semantics

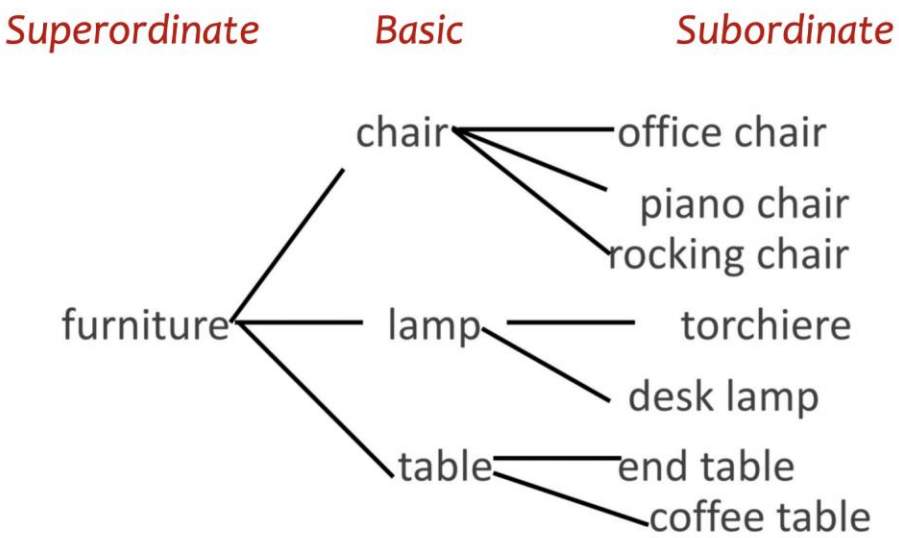
Relation: Superordinate/Subordinate

- One sense is a subordinate of another if the first sense is more specific, denoting a subclass of the other
  - Car is a subordinate of vehicle
  - Mango is a subordinate of fruit

- Conversely superordinate
  - Vehicle is a superordinate of car
  - Fruit is a superordinate of mango

Vector Semantics

Taxonomy



Vector Semantics

Lexical Semantics

- How should we represent the meaning of the word?
  - Words, lemmas, senses, definitions
  - Relationships between words or senses
  - Taxonomy relationships
  - Word similarity, word relatedness
  - Semantic frames and roles

Vector Semantics

Semantic Frame

- A set of words that denote perspectives or participants in a particular type of event
  - “buy” (the event from the perspective of the buyer)
  - “sell” (from the perspective of the seller)
  - “pay” (focusing on the monetary aspect)
  - John hit Bill ¡ Bill was hit by John

- Frames have semantic roles (like buyer, sellers, goods, money) and words in a sentence can take on those roles

Vector Semantics

Lexical Semantics

- How should we represent the meaning of the word?
  - Words, lemmas, senses, definitions
  - Relationships between words or senses
  - Taxonomy relationships
  - Word similarity, word relatedness
  - Semantic frames and roles
  - Connotation and sentiment

Vector Semantics

Connotation and sentiment

- Connotations refer to the aspects of a word's meaning that are related to a writer or reader's emotions, sentiment, opinions, or evaluations.
  - happy vs. Sad
  - great, love vs. terrible, hate

- Three dimensions of affective meaning
  - Valence: the pleasantness of the stimulus
  - Arousal: the intensity of emotion
  - Dominance: the degree of control exerted by the stimulus

	Valence	Arousal	Dominance
courageous	8.05	5.5	7.38
music	7.67	5.57	6.5
heartbreak	2.45	5.65	3.58
cub	6.71	3.95	4.24
life	6.68	5.59	5.89

Vector Semantics

Lexical Semantics

- How should we represent the meaning of the word?

    1. Words, lemmas, senses, definitions
    2. Relationships between words or senses
    3. Taxonomy relationships
    4. Word similarity, word relatedness
    5. Semantic frames and roles
    6. Connotation and sentiment

Vector Semantics

Electronic Dictionaries

WordNet

```
from nltk.corpus import wordnet as wn
panda = wn.synset('panda.n.01')
hyper = lambda s: s.hypernyms()
list(panda.closure(hyper))
```

```
[Synset('procyonid.n.01'),
Synset('carnivore.n.01'),
Synset('placental.n.01'),
Synset('mammal.n.01'),
Synset('vertebrate.n.01'),
Synset('chordate.n.01'),
Synset('animal.n.01'),
Synset('organism.n.01'),
Synset('living_thing.n.01'),
Synset('whole.n.02'),
Synset('object.n.01'),
Synset('physical_entity.n.01'),
Synset('entity.n.01')]
```

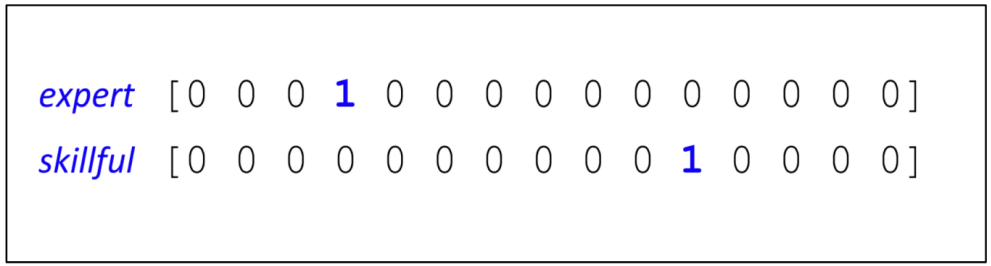
(here, for *good*):

```
S: (adj) full, good
S: (adj) estimable, good, honorable, respectable
S: (adj) beneficial, good
S: (adj) good, just, upright
S: (adj) adept, expert, good, practiced,
proficient, skillful
S: (adj) dear, good, near
S: (adj) good, right, ripe
...
S: (adv) well, good
S: (adv) thoroughly, soundly, good
S: (n) good, goodness
S: (n) commodity, trade good, good
```

Vector Semantics

Problems with Discrete Representation

- Too coarse
  - Expert vs skillful
- Sparse
  - Wicked, badass, ninja
- Subjective
- Expensive
- Hard to compute word relationships



Vector Semantics

Distributional Hypothesis

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

[Wittgenstein PI 43]

"You shall know a word by the company it keeps"

[Firth 1957]

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

[Harris 1954]

Vector Semantics

Example: What does OngChoi Mean?

- Suppose you see those sentences:
  - Ongchoi is delicious sautéed with garlic
  - Ongchoi is superb over rice
  - Ongchoi 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

Vector Semantics

Example: What does OngChoi Mean?

- Suppose you see those sentences:
  - Ongchoi is delicious sautéed with garlic
  - Ongchoi is superb over rice
  - Ongchoi 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

Vector Semantics

Word Embedding Representations

- Count-based
  - Tf-idf, PPMI
- Class-based
  - Brown Clusters
- Distributed prediction-based embeddings
  - Word2vec, FastText
- Distributed contextual (token) embeddings from language models
  - Elmo, BERT
- + many more variants
  - Multilingual embeddings, multi-sense embeddings, syntactic embeddings, etc …

Vector Semantics

Term-Document Matrix

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	17
solider	2	80	62	89
fool	36	58	1	4
clown	20	15	2	3

Context = appearing in the same document.

Vector Semantics

Term-Document Matrix

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	0	7	17
solider	2	80	62	89
fool	36	58	1	4
clown	20	15	2	3

Vector Space Model:
Each document is represented as a column vector of length four

Vector Semantics

Term-Context Matrix / Word-Word Matrix

	knife	dog	sword	love	like
knife	0	1	6	5	5
dog	1	0	5	5	5
sword	6	5	0	5	5
love	5	5	5	0	5
like	5	5	5	5	2

Two words are “similar” in meaning if their context vectors are similar.

- Similarity == relatedness

Vector Semantics

Count-Based Representations

	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

Counts: term-frequency

- Remove stop words
- Use log$% &'
- Normalize by document length

Vector Semantics

TF-IDF

What to do with words that are evenly distributed across many documents?

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

$$idf_i = \log_{10}\left(\frac{N}{df_i}\right)$$

Total # of docs in collection

# of docs that have word i

Vector Semantics

TF-IDF

What to do with words that are evenly distributed across many documents?

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

$$idf_i = \log_{10}\left(\frac{N}{df_i}\right)$$

Total # of docs in collection

# of docs that have word i

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

$$w_{t,d} = tf_{t,d} \times idf_i$$

Vector Semantics

Pointwise Mutual Information (PMI)

Q: Do word ! and c co-occur more than if they were independent?

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

Vector Semantics

Positive Pointwise Mutual Information (PPMI)

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

Vector Semantics

Positive Pointwise Mutual Information (PPMI)

- PMI is biased toward infrequent events
  - Very rare words have very high PMI values
  - Give rare words slightly higher probabilities α=0.75

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

$$\text{P}_\alpha(c) = \frac{count(c)^\alpha}{\sum_c count(c)^\alpha}$$

Vector Semantics

Sparse versus Dense Vectors

- PPMI vectors are
  - **Long** (length |V| = 20,000 to 50,000)
  - **Sparse** (most elements are zero)

- Alternative: learn vectors which are
  - **Short** (length 200-1000)
  - **Dense** (most elements are non-zero)

Vector Semantics

Word Similarity

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

Word Embeddings (Word2Vec)

Word2Vec

- Instead of counting how often each word w occurs near ”peach"
- Train a classifier on a binary prediction task:
  - Is w likely to show up near “peach”?

- We don’t actually care about this task
  - But we'll take the learned classifier weights as the word embeddings

Word Embeddings (Word2Vec)

Use Running Text As Implicitly Supervised Training Data

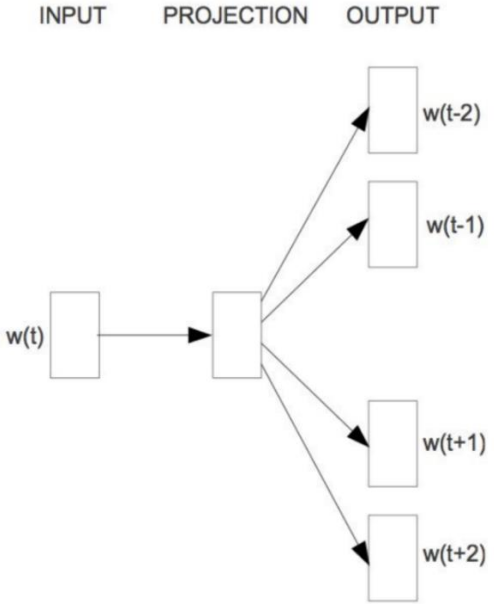
- A word s near peach
  - Acts as gold ‘correct answer’ to the question
  - “Is word w likely to show up near peach?”

- No need for hand-labeled supervision

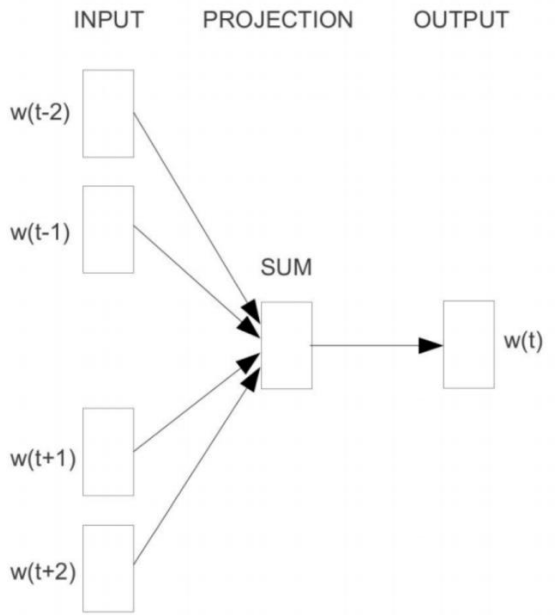
- The idea comes from **neural language modeling**
  - Bengio et al. (2003)
  - Collobert et al. (2011)

Word Embeddings (Word2Vec)

Skip-gram v.s Continuous bag-of-words



Skip-gram



CBOW

Mikolov et al., 2013

Word Embeddings (Word2Vec)

Word2Vec: Skip-Gram Task

- Word2vec provides a variety of options.
- Let's do “skip-gram with negative sampling” (SGNS)

Word Embeddings (Word2Vec)

Skip-Gram Sketch

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 classifier to distinguish those two cases
4. Use the weights as the embeddings

Word Embeddings (Word2Vec)

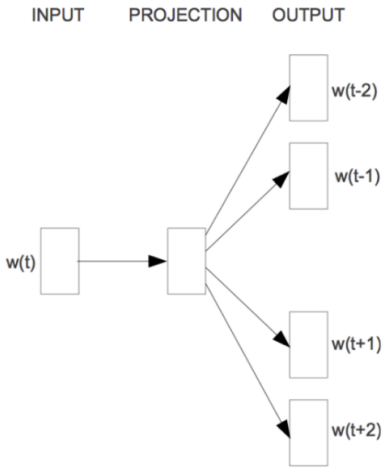
Skip-gram

Maximize the log likelihood of context word $w_{t-m}, w_{t-m+1}, \dots, w_{t-1}, w_{t+1}, w_{t+2}, \dots, w_{t+m}$ given word w_t

$$J(\Theta) = \prod_{t=1}^{T} \prod_{-m \le j \le m, j \neq 0} p(w_{t+j} | w_t; \Theta)$$

$$J(\theta) = \frac{1}{T} \sum_{t=1}^{T} \sum_{-m \le j \le m, j \neq 0} \log p(w_{t+j} | w_t)$$

m is usually 5~10



Word Embeddings (Word2Vec)

Skip-gram

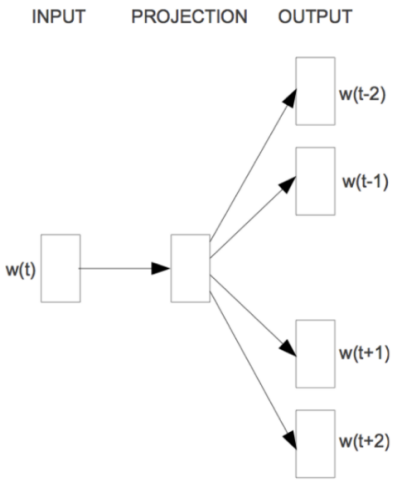
How to model log $P(w_{t+j}|w_t)$?

$$p(w_{t+j}|w_t) = \frac{\exp(u_{w_{t+j}} \cdot v_{w_t})}{\sum_{w'} \exp(u_{w'} \cdot v_{w_t})}$$

- Softmax function Again!

Every word has 2 vectors

- v_w : when w is the center word
- u_w: when w is the outside word (context word)



Word Embeddings (Word2Vec)

How to update?

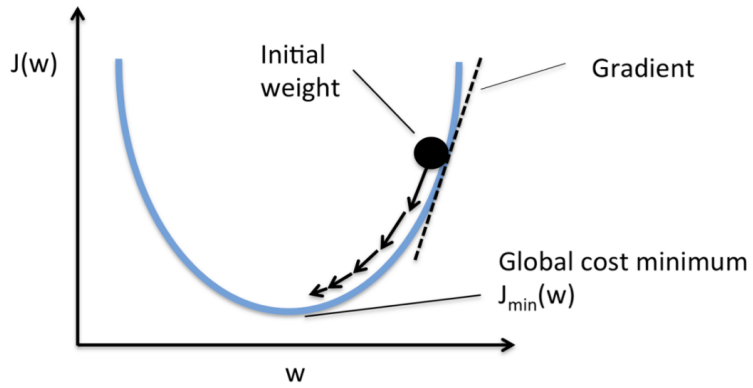
- How to minimize −J(θ)
  - Gradient descent!
  - How to compute the gradient?

$$J(\theta) = \frac{1}{T}\sum_{t=1}^{T}\sum_{-m\leq j\leq m, j\neq 0} \log p(w_{t+j}|w_t)$$

$$p(w_{t+j}|w_t) = \frac{\exp(u_{w_{t+j}} \cdot v_{w_t})}{\sum_{w'} \exp(u_{w'} \cdot v_{w_t})}$$

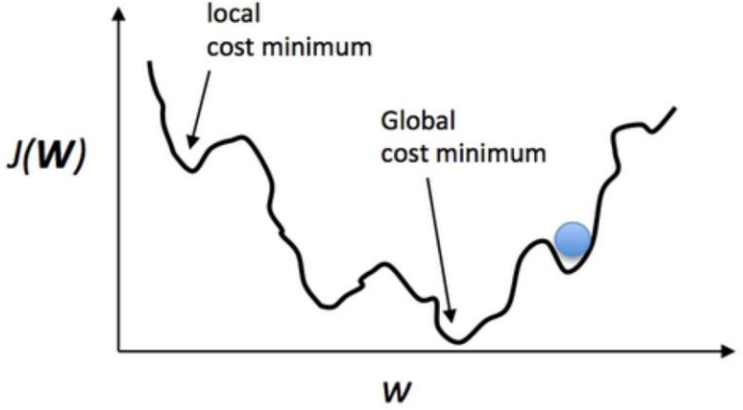
Word Embeddings (Word2Vec)

Recap: Gradient Descent



$$\min_w J(w)$$

Update w: $w \leftarrow w - \eta \nabla J(w)$



Local Minimum vs. Global Minimum

Word Embeddings (Word2Vec)

Skip-Gram Training Data

- Training sentence:

... lemon, a tablespoon of apricot jam a pinch ...

c1 c2 t c3 c4

- For each positive example, we'll create k negative examples.
- Using noise words
- Any random word that isn't t

positive examples +	
t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

Word Embeddings (Word2Vec)

Skip-Gram Training Data

- Training sentence:

  ... lemon, a tablespoon of apricot jam a pinch ...

  c1 c2 t c3 c4

- For each positive example, we'll create k negative examples.
- Using noise words
- Any random word that isn't t

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	preserves
apricot	or

negative examples -

t	c	t	c
apricot	aardvark	apricot	twelve
apricot	puddle	apricot	hello
apricot	where	apricot	dear
apricot	coaxial	apricot	forever

Word Embeddings (Word2Vec)

Choosing Noise Words

- Could pick w according to their unigram frequency P(w)
- More common to chosen then according to $p_a(w)$

$$P_\alpha(w) = \frac{count(w)^\alpha}{\sum_w count(w)^\alpha}$$

- $\alpha = ¾$ works well because it gives rare noise words slightly higher probability
- To show this, imagine two events p(a) = .99 and p(b) = .01

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

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

Word Embeddings (Word2Vec)

Setup

- Let's represent words as vectors of some length (say 300), randomly initialized.
- So we start with 300 * V random parameters

- Over the entire training set, we'd like to adjust those word vectors such that we
  - Maximize the similarity of the target word, context word pairs (t,c) drawn from the positive data
  - Minimize the similarity of the (t,c) pairs drawn from the negative data.

Word Embeddings (Word2Vec)

Learning the Classifier

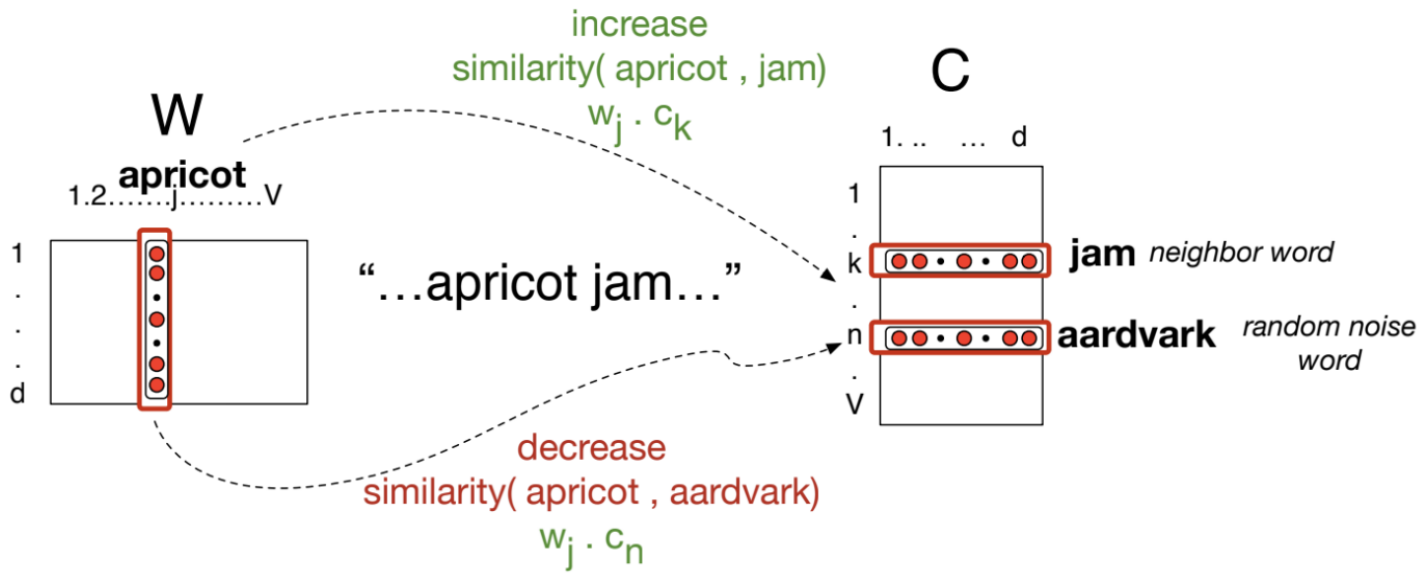
- Iterative Process
  - Start with random weights
  - Adjust the word weights to
    - Make the positive pairs more likely
    - Make the negative pairs less likely
  - Over the entire training set

$$\sum_{(t,c)\in+} logP(+|t,c) + \sum_{(t,c)\in-} logP(-|t,c)$$

- Training using gradient descent

Word Embeddings (Word2Vec)

Learning the Classifier



Word Embeddings (Word2Vec)

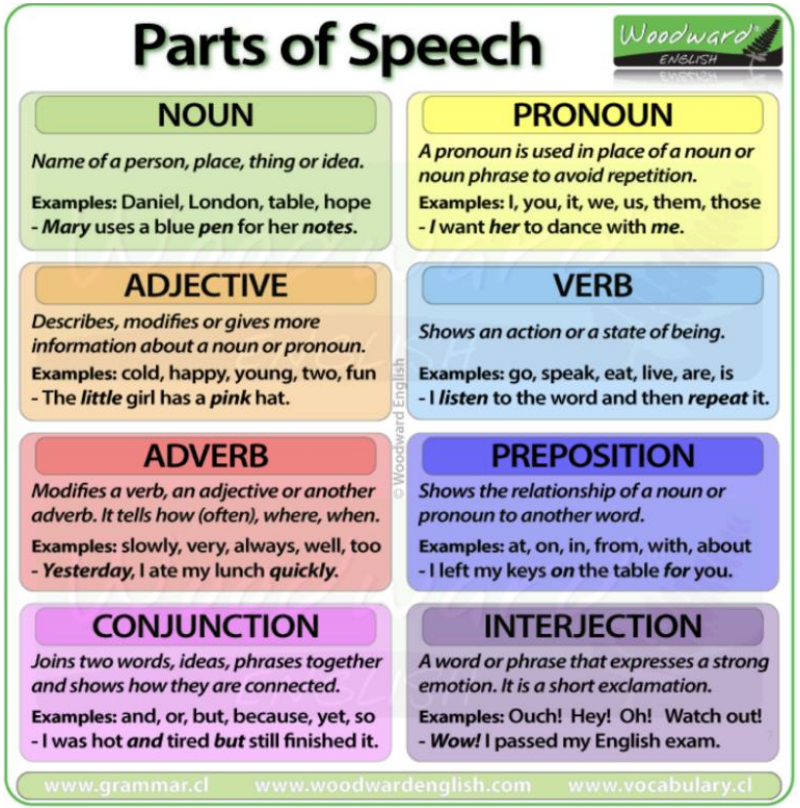
Summary: How to learn word2vec embeddings

- Start with V random 300-dimensional vectors as initial embeddings
- Use logistic regression, the second most basic classifier used in machine learning after Naïve Bayes
    - Take a corpus and take pairs of words that co-occur as positive examples
    - Take pairs of words that don't co-occur as negative examples
    - Train the classifier to distinguish these by slowly adjusting all the embeddings to improve the classifier performance
    - Throw away the classifier code and keep the embeddings.

Sequence Labeling

Parts of speech (POS)

- Traditional parts of speech
  - Noun
  - Pronoun
  - Adjective
  - Verb
  - Adverb
  - Preposition
  - Conjunction
  - Interjection



Sequence Labeling

POS Example

- **N** noun
  - chair, bandwidth, peach
- V verb
  - study, debate, run
- ADJ adjective
  - purple, tall, ridiculous
- ADV adverb
  - unfortunately, slowly
- P preposition
  - of, by, to
- PRO pronoun
  - I, me, mine
- DET determiner
  - the, a, that, those

Sequence Labeling

Part of speech

- Part-of-speech, lexical categories, word classes, morphological classes, lexical tags...

- Lots of debate within linguistics about the number, nature, and universality of these

Sequence Labeling

Part of speech tagging

- The process of assigning a part-of-speech to each word in a collection (sentence).

WORD	Tag
the	DET
koala	N
put	V
the	DET
keys	N
on	P
the	DET
table	N

Sequence Labeling

Why is POS Tagging Useful?

- First step of a vast number of practical tasks
- Parsing
  - Need to know if a word is an N or V before you can parse
- Information extraction
  - Finding names, relations, etc.
- Speech synthesis/recognition
  - OBject obJECT
  - OVERflow overFLOW
  - DIScount disCOUNT
  - CONtent conTENT
- Machine Translation

Sequence Labeling

Choosing a Tagset

- Could pick very coarse tagsets
  - N, V, Adj, Adv, Other

- More commonly used set is finer grained
  - E.g., “Penn TreeBank tagset”, 45 tags: PRP$, WRB, WP$, VBG

- Prague Dependency Treebank (Czech)
  - 4452 tags

Sequence Labeling

Penn TreeBank POS Tagset

Penn Treebank Tagset

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PRP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PRP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Sequence Labeling

Part-of-Speech Tagging

- The process of assigning a part-of-speech marker to each word in an input text.

- **Input**: a sequence of tokenized words and a tagset
- **Output**: a sequence of tags, one per token

Sequence Labeling

Tagging Is A Disambiguation Task

- Words often have more than one POS: **back**
  - The **back** door = JJ
  - On my **back** = NN
  - Win the voters **back** = RB
  - Promised to **back** the bill = VB

Penn Treebank Tagset

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	$
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PRP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PRP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Sequence Labeling

Example: POS Tagging

- Mrs Shaefer never got around to joining
- All we goka do is go around the corner
- Chateau Petrus costs around 250

Penn Treebank Tagset

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	*$*
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PRP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PRP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Sequence Labeling

Example: POS Tagging

- Mrs/NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG
- All/DT we/PRP goka/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN
- Chateau/NNP Petrus/NNP costs/VBZ around/RB 250/CD

Penn Treebank Tagset

Tag	Description	Example	Tag	Description	Example
CC	Coordin. Conjunction	*and, but, or*	SYM	Symbol	+,%, &
CD	Cardinal number	*one, two, three*	TO	"to"	*to*
DT	Determiner	*a, the*	UH	Interjection	*ah, oops*
EX	Existential 'there'	*there*	VB	Verb, base form	*eat*
FW	Foreign word	*mea culpa*	VBD	Verb, past tense	*ate*
IN	Preposition/sub-conj	*of, in, by*	VBG	Verb, gerund	*eating*
JJ	Adjective	*yellow*	VBN	Verb, past participle	*eaten*
JJR	Adj., comparative	*bigger*	VBP	Verb, non-3sg pres	*eat*
JJS	Adj., superlative	*wildest*	VBZ	Verb, 3sg pres	*eats*
LS	List item marker	*1, 2, One*	WDT	Wh-determiner	*which, that*
MD	Modal	*can, should*	WP	Wh-pronoun	*what, who*
NN	Noun, sing. or mass	*llama*	WP$	Possessive wh-	*whose*
NNS	Noun, plural	*llamas*	WRB	Wh-adverb	*how, where*
NNP	Proper noun, singular	*IBM*	$	Dollar sign	$
NNPS	Proper noun, plural	*Carolinas*	#	Pound sign	#
PDT	Predeterminer	*all, both*	"	Left quote	(' or ")
POS	Possessive ending	*'s*	"	Right quote	(' or ")
PRP	Personal pronoun	*I, you, he*	(	Left parenthesis	([, (, {, <)
PRP$	Possessive pronoun	*your, one's*	)	Right parenthesis	( ],), }, >)
RB	Adverb	*quickly, never*	,	Comma	,
RBR	Adverb, comparative	*faster*	.	Sentence-final punc	(. ! ?)
RBS	Adverb, superlative	*fastest*	:	Mid-sentence punc	(: ; ... – -)
RP	Particle	*up, off*			

Sequence Labeling

Deciding On the Correct POS Can Be Difficult for People

- Mrs/NNP Shaefer/NNP never/RB got/VBD around/RP to/TO joining/VBG
- All/DT we/PRP goka/VBN do/VB is/VBZ go/VB around/IN the/DT corner/NN
- Chateau/NNP Petrus/NNP costs/VBZ around/RB 250/CD

Sequence Labeling

Parts-Of-Speech (POS) tagging in Python using the Natural Language Toolkit (NLTK) library

```python
import nltk
nltk.download('all')

from nltk.tokenize import word_tokenize
from nltk import pos_tag

# Sample text for POS tagging
text = "NLTK is a leading platform for building Python programs to work with human language data."
tokens = word_tokenize(text)
pos_tags = pos_tag(tokens)

print(pos_tags)
```

✓ 0.0s Python

```
[('NLTK', 'NNP'), ('is', 'VBZ'), ('a', 'DT'), ('leading', 'VBG'), ('platform', 'NN'), ('for', 'IN'),
('building', 'VBG'), ('Python', 'NNP'), ('programs', 'NNS'), ('to', 'TO'), ('work', 'VB'), ('with', 'IN'),
('human', 'JJ'), ('language', 'NN'), ('data', 'NNS'), ('.', '.')]
```

Sentiment Analysis

What is Sentiment Analysis?

Sentiment analysis (also known as opinion mining) is a technique in natural language processing (NLP) that focuses on determining the emotional tone or attitude expressed in a piece of text. The goal is to classify the sentiment as positive, negative, or neutral, though more fine-grained categories (such as very positive or very negative) can also be used.

Sentiment Analysis

Positive or negative movie review?


- unbelievably disappointing


- Full of zany characters and richly applied satire, and some great plot twists


- this is the greatest screwball comedy ever filmed


- It was pathetic. The worst part about it was the boxing scenes.

Sentiment Analysis

Google Product Search

HP Officejet 6500A Plus e-All-in-One Color Ink-jet - Fax / copier / printer / scanner
$89 online, **$100** nearby ★★★★☆ 377 reviews
September 2010 - Printer - HP - Inkjet - Office - Copier - Color - Scanner - Fax - 250 sh

Reviews

Summary - Based on 377 reviews

1 star	2	3	4 stars	5 stars

What people are saying

ease of use	"This was very easy to setup to four computers."
value	"Appreciate good quality at a fair price."
setup	"Overall pretty easy setup."
customer service	"I DO like honest tech support people."
size	"Pretty Paper weight."
mode	"Photos were fair on the high quality mode."
colors	"Full color prints came out with great quality."

Sentiment Analysis

Amazon Customer Review

Customer Review

C Wm (Andy) Anderson #1 REVIEWER #1 HALL OF FAME

Excellent Stereo Speakers

May 4, 2019

Color: SP2070 BLUE | Vine Customer Review of Free Product (What's this?)

I got these Because I prefer my audio to be routed through a wired connection due to the sound quality. I will use bluetooth when I must, but, when I can, I use these speakers. Naturally, with my iPhone I either must use bluetooth ear phones (my favorite are my Jabra), or the adaptor that connects to the power connection. But, with my iPad, I get to use these speakers.

Sound quality for phone calls, audio books is excellent. It is very good for most music, except for classical. But, truth be known, my hearing doesn't permit me to tune in to the nuances so well anymore.

I can, however, appreciate Wynton Marsalis.

BOTTOM LINE

Five stars out of five.

Helpful | Comment | Report abuse | Permalink

Product Details

USB Powered Computer Speakers, Re...
by RECCAZR

22

4.2 out of 5 stars

5 star 54%
4 star 18%
3 star 23%
2 star 0%
1 star 5%

$25.99 + Free shipping with Amazon Prime

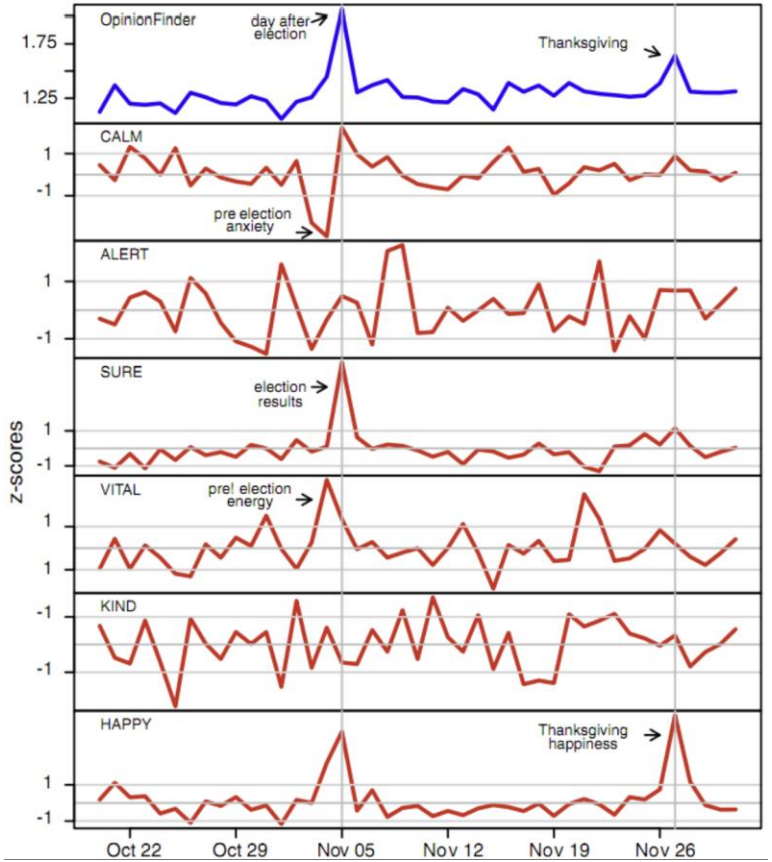
Add to Cart

Add to Wish List

Sentiment Analysis

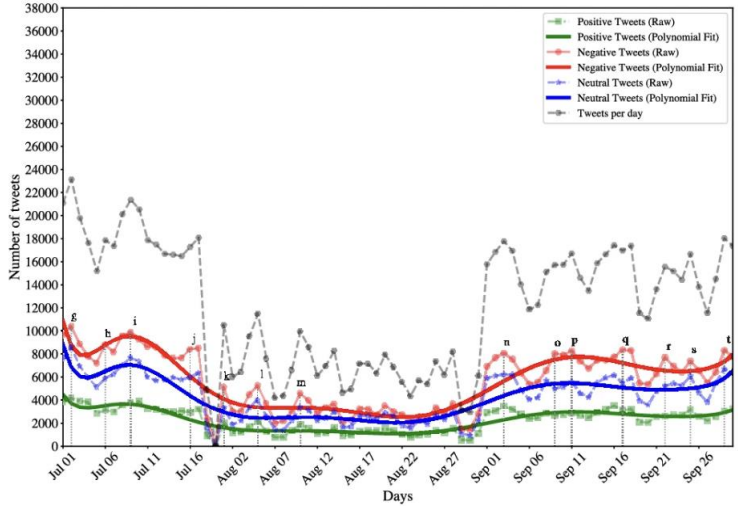
Twitter sentiment

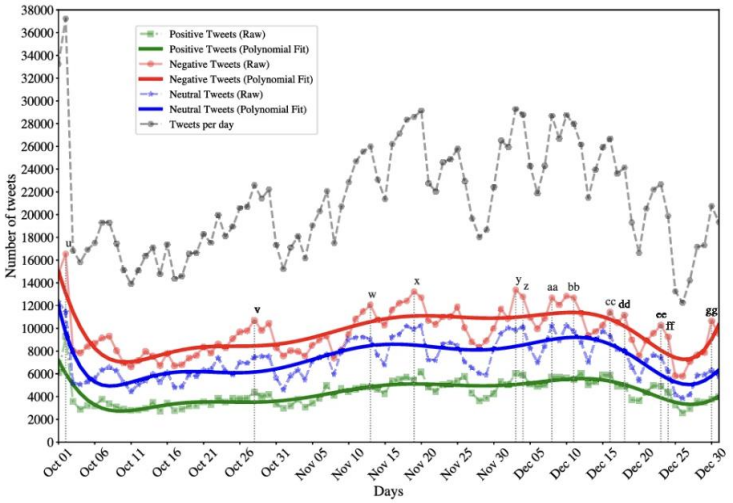
Johan Bollen, Huina Mao, Xiaojun Zeng. 2011.
***Twitter mood** predicts the stock market, Journal of Computational Science 2:1, 1-8. 10.1016/j.jocs.2010.12.007.*



Sentiment Analysis

Twitter sentiment





Chaudhary, M., Kosyluk, K., Thomas, S. et al. On the use of aspect-based ***sentiment analysis of Twitter data*** *to explore the experiences of African Americans during COVID-19. Sci Rep 13, 10694 (2023). https://doi.org/10.1038/s41598-023-37592-1*

Sentiment Analysis

Sentiment analysis has many other names

- Opinion extraction
- Opinion mining
- Sentiment mining
- Subjectivity analysis

Sentiment Analysis

Why sentiment analysis?

- Movie: is this review positive or negative?
- Products: what do people think about the new iPhone?
- Public sentiment: how is consumer confidence? Is despair increasing?
- Politics: what do people think about this candidate or issue?
- Prediction: predict election outcomes or market trends from sentiment

Sentiment Analysis

Scherer Typology of Affective States (published updated one, reference)

- Emotion: brief organically synchronized … evaluation of a major event
  - angry, sad, joyful, fearful, ashamed, proud, elated
- Mood: diffuse non-caused low-intensity long-duration change in subjective feeling
  - cheerful, gloomy, irritable, listless, depressed, buoyant
- Interpersonal stances: affective stance toward another person in a specific interaction
  - friendly, flirtatious, distant, cold, warm, supportive, contemptuous
- Attitudes: enduring, affectively colored beliefs, dispositions towards objects or persons
  - liking, loving, hating, valuing, desiring
- Personality traits: stable personality dispositions and typical behavior tendencies
  - nervous, anxious, reckless, morose, hostile, jealous

Sentiment Analysis

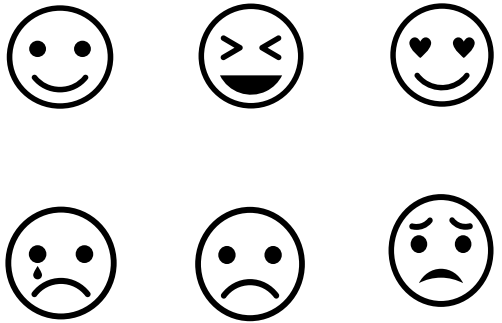
Sentiment analysis is the detection of attitudes

"enduring, affectively colored beliefs, dispositions towards objects or persons"

1. Holder (source) of attitude
2. Target (aspect) of attitude
3. Type of attitude
    - From a set of types
        - Like, love, hate, value, desire, etc.
    - Or (more commonly) simple weighted polarity:
        - positive, negative, neutral, together with strength
4. Text containing the attitude
    - Sentence or entire document

Sentiment Analysis

- Simplest task:
  - Is the attitude of this text positive or negative?
- More complex:
  - Rank the attitude of this text from 1 to 5
- Advanced:
  - Detect the target, source, or complex attitude types



Sentiment Analysis

A Baseline Algorithm

1. Tokenization
2. Feature Extraction
3. Classification using different classifiers
    - Naïve Bayes
    - MaxEnt
    - SVM

Sentiment Analysis

Sentiment Tokenization Issues

- Deal with HTML and XML markup
- Twitter mark-up (names, hash tags)
- Capitalization (preserve for words in all caps)
- Phone numbers, dates • Emoticons
- Useful code:
  - Christopher Potts sentiment tokenizer
  - Brendan O'Connor twitter tokenizer

Potts Emoticons

```
[<>]?                       # optional hat/brow
[:;=8]                      # eyes
[\-o\*\']?                  # optional nose
[\)\]\(\[dDpP/\:\}\{@\|\\]  # mouth
|                           #### reverse orientation
[\)\]\(\[dDpP/\:\}\{@\|\\]  # mouth
[\-o\*\']?                  # optional nose
[:;=8]                      # eyes
[<>]?                       # optional hat/brow
```

Sentiment Analysis

Extracting Features for Sentiment Classification

- How to handle negation
  - I didn’t like this movie

  Vs
  - I really like this movie
- Which words to use?
  - Only adjectives
  - All words
    - All words turns out to work better, at least on this data

Sentiment Analysis

Negation

Das, Sanjiv and Mike Chen. 2001. Yahoo! for Amazon: Extracting market sentiment from stock message boards. In Proceedings of the Asia Pacific Finance Association Annual Conference (APFA). Bo Pang, Lillian Lee, and Shivakumar Vaithyanathan. 2002. Thumbs up? Sentiment Classification using Machine Learning Techniques. EMNLP-2002, 79—86.

Add NOT_ to every word between negation and following punctuation:

```
didn't like this movie , but I
```

```
didn't NOT_like NOT_this NOT_movie but I
```

Sentiment Analysis

Recap: Naïve Bayes

$$c_{NB} = \underset{c_j \in C}{\operatorname{argmax}} P(c_j) \prod_{i \in positions} P(w_i \mid c_j)$$

$$\hat{P}(w \mid c) = \frac{count(w,c)+1}{count(c)+|V|}$$

Sentiment Analysis

Binarized (Boolean feature) Multinomial Naïve Bayes

- Intuition:
  - For sentiment (and probably for other text classification domains)
  - Word occurrence may matter more than word frequency
    - The occurrence of the word fantastic tells us a lot
    - The fact that it occurs 5 times may not tell us much more.
  - Boolean Multinomial Naïve Bayes
    - Clips all the word counts in each document at 1

Sentiment Analysis

What makes reviews hard to classify?

- Subtlety:
  - Perfume review in Perfumes: the Guide:
    - “If you are reading this because it is your darling fragrance, please wear it at home exclusively, and tape the windows shut.”
  - Dorothy Parker on Katherine Hepburn
    - “She runs the gamut of emotions from A to B”

Sentiment Analysis

Thwarted Expectations and Ordering Effects

- “This film should be brilliant. It sounds like a great plot, the actors are first grade, and the supporting cast is good as well, and Stallone is attempting to deliver a good performance. However, it can’t hold up. ”
- Well as usual Keanu Reeves is nothing special, but surprisingly, the very talented Laurence Fishbourne is not so good either, I was surprised.

Vector Semantics

computing the similarity between words

- "fast" is similar to "rapid"
- "tall" is similar to "height"

- Question answering:
  - Q: "How tall is Mt. Everest?" Candidate
  - A: "The official height of Mount Everest is 29029 feet"

Vector Semantics

Word similarity for plagiarism detection

MAINFRAMES

Mainframes are primarily referred to large computers with rapid, advanced processing capabilities that can execute and perform tasks equivalent to many Personal Computers (PCs) machines networked together. It is characterized with high quantity Random Access Memory (RAM), very large secondary storage devices, and high-speed processors to cater for the needs of the computers under its service.

Consisting of advanced components, mainframes have the capability of running multiple large applications required by many and most enterprises and organizations. This is one of its advantages. Mainframes are also suitable to cater for those applications (programs) or files that are of very high

MAINFRAMES

Mainframes usually are referred those computers with fast, advanced processing capabilities that could perform by itself tasks that may require a lot of Personal Computers (PC) Machines. Usually mainframes would have lots of RAMs, very large secondary storage devices, and very fast processors to cater for the needs of those computers under its service.

Due to the advanced components mainframes have, these computers have the capability of running multiple large applications required by most enterprises, which is one of its advantage. Mainframes are also suitable to cater for those applications or files that are of very large demand

Vector Semantics

Four kinds of vector models

- Sparse vector representations
  - Mutual-information weighted word co-occurrence matrices
- Dense vector representations:
  - Singular value decomposition (and Latent Semantic Analysis)
  - Neural-network-inspired models (skip-grams, CBOW)
  - Brown clusters

Vector Semantics

Co-occurrence Matrices

- We represent how often a word occurs in a document
  - Term-document matrix
- Or how often a word occurs with another
  - Term-term matrix (or word-word co-occurrence matrix or word-context matrix)

Vector Semantics

Term-document matrix (Source: context)

- Each cell: count of word w in a document d:
  - Each document is a count vector in ℕv: a column below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Vector Semantics

Similarity in term-document matrices

- Two documents are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Vector Semantics

The words in a term-document matrix

- Each word is a count vector in ℕD: a row below

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Vector Semantics

The words in a term-document matrix

- Two words are similar if their vectors are similar

	As You Like It	Twelfth Night	Julius Caesar	Henry V
battle	1	1	8	15
soldier	2	2	12	36
fool	37	58	1	5
clown	6	117	0	0

Vector Semantics

The word-word or word-context matrix

- Instead of entire documents, use smaller contexts
- Paragraph • Window of ± 4 words
- A word is now defined by a vector over counts of context words
- Instead of each vector being of length D
- Each vector is now of length |V|
- The word-word matrix is |V|x|V|

Vector Semantics

Word-Word matrix Sample contexts ± 7 words

sugar, a sliced lemon, a tablespoonful of **apricot** preserve or jam, a pinch each of,
their enjoyment. Cautiously she sampled her first **pineapple** and another fruit whose taste she likened
well suited to programming on the digital **computer**. In finding the optimal R-stage policy from
for the purpose of gathering data and **information** necessary for the study authorized in the

	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	
…	…						

Vector Semantics

Word-word matrix

- We showed only 4x6, but the real matrix is 50,000 x 50,000
  - So it's very sparse
    - Most values are 0.
  - That's OK, since there are lots of efficient algorithms for sparse matrices.
- The size of windows depends on your goals
  - The shorter the windows , the more syntactic the representation ± 1-3 very syntacticy
  - The longer the windows, the more semantic the representation ± 4-10 more semanticy

Vector Semantics

2 kinds of co-occurrence between 2 words (Schütze and Pedersen, 1993)

- First-order co-occurrence (syntagmatic association):
  - They are typically nearby each other.
  - wrote is a first-order associate of book or poem.
- Second-order co-occurrence (paradigmatic association):
  - They have similar neighbors.
  - wrote is a second- order associate of words like said or remarked.

Vector Semantics

Positive Pointwise Mutual Information (PPMI)

- Problem with raw counts
  - Raw word frequency is not a great measure of association between words
    - It's very skewed
      - "the" and "of" are very frequent, but maybe not the most discriminative
  - We'd rather have a measure that asks whether a context word is particularly informative about the target word.
    - Positive Pointwise Mutual Information (PPMI)

Vector Semantics

Pointwise Mutual Information (PPMI)

- 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?

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

Vector Semantics

Positive Pointwise Mutual Information (PPMI)

- 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:

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

Vector Semantics

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

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

	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

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

Vector Semantics

Computing PPMI on a term-context matrix

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

Count(w,context)

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

p(w=information,c=data) = 6/19 = .32

p(w=information) = 11/19 = .58

p(c=data) = 7/19 = .37

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

p(w,context) **p(w)**

	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

Vector Semantics

Computing PPMI on a term-context matrix

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

	p(w,context)					p(w)
	computer	data	pinch	result	sugar	
apricot	0.00	0.00	0.05	0.00	0.05	0.11
pineapple	0.00	0.00	0.05	0.00	0.05	0.11
digital	0.11	0.05	0.00	0.05	0.00	0.21
information	0.05	0.32	0.00	0.21	0.00	0.58
p(context)	0.16	0.37	0.11	0.26	0.11	

pmi(information,data) = $\log_2$ (.32 / (.37*.58)) = .58

	PPMI(w,context)				
	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

Vector Semantics

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)

Vector Semantics

Weighting PMI: Giving rare context words slightly higher probability

- Raise the context probabilities to α = 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

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

Vector Semantics

Use Laplace (add-1) smoothing (Put equation)

	Add-2 Smoothed Count(w,context)				
	computer	data	pinch	result	sugar
apricot	2	2	3	2	3
pineapple	2	2	3	2	3
digital	4	3	2	3	2
information	3	8	2	6	2

	p(w,context) [add-2]					p(w)
	computer	data	pinch	result	sugar	
apricot	0.03	0.03	0.05	0.03	0.05	0.20
pineapple	0.03	0.03	0.05	0.03	0.05	0.20
digital	0.07	0.05	0.03	0.05	0.03	0.24
information	0.05	0.14	0.03	0.10	0.03	0.36
p(context)	0.19	0.25	0.17	0.22	0.17	

Vector Semantics

PPMI versus add-2 smoothed PPMI

PPMI(w,context)

	computer	data	pinch	result	sugar
apricot	-	-	2.25	-	2.25
pineapple	-	-	2.25	-	2.25
digital	1.66	0.00	-	0.00	-
information	0.00	0.57	-	0.47	-

PPMI(w,context) [add-2]

	computer	data	pinch	result	sugar
apricot	0.00	0.00	0.56	0.00	0.56
pineapple	0.00	0.00	0.56	0.00	0.56
digital	0.62	0.00	0.00	0.00	0.00
information	0.00	0.58	0.00	0.37	0.00

Vector Semantics

Measuring similarity: the cosine

- Given 2 target words v and w
- We'll need a way to measure their similarity.
- Most measure of vectors similarity are based on the:
- Dot product or inner product from linear algebra

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

  - High when two vectors have large values in the same dimensions.
  - Low (in fact 0) for orthogonal vectors with zeros in complementary distribution.

Vector Semantics

Problem with dot product

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

- Dot product is longer if the vector is longer. Vector length:

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

- Vectors are longer if they have higher values in each dimension
- That means more frequent words will have higher dot products
- That's bad: we don't want a similarity metric to be sensitive to word frequency.

Vector Semantics

Solution: cosine

- Just divide the dot product by the length of the two vectors!

$$\frac{\vec{a} \cdot \vec{b}}{|\vec{a}||\vec{b}|}$$

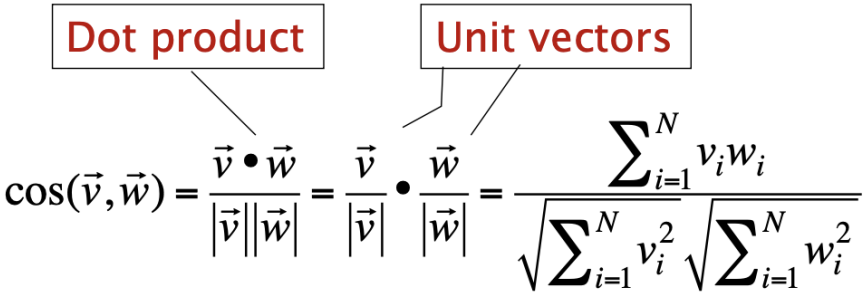
- This turns out to be the cosine of the angle between them!

$$\vec{a} \cdot \vec{b} = |\vec{a}||\vec{b}| \cos\theta$$

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

Vector Semantics

Cosine for computing similarity



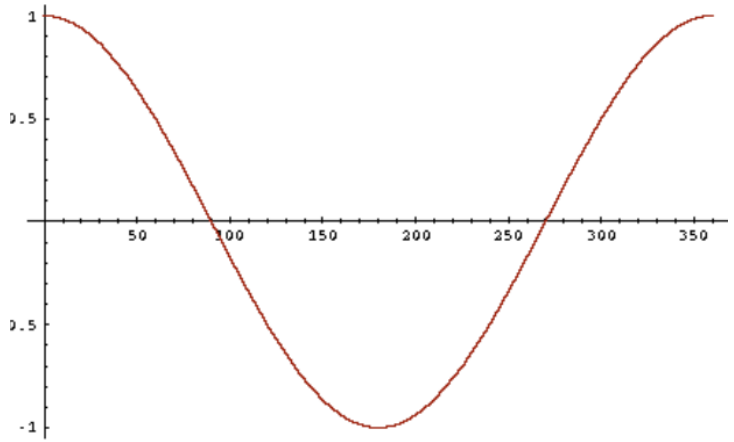
v_i is the PPMI value for word v in context i
w_i is the PPMI value for word w in context i.

Cos($\vec{v},\vec{w}$) is the cosine similarity of $\vec{v}$ and $\vec{w}$

Vector Semantics

Cosine as a similarity metric

- -1: vectors point in opposite directions
- +1: vectors point in same directions
- 0: vectors are orthogonal
- Raw frequency or PPMI are nonnegative, so cosine range 0-1



Vector Semantics

Example: Which pair of words is more similar? (Source: link, context)

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

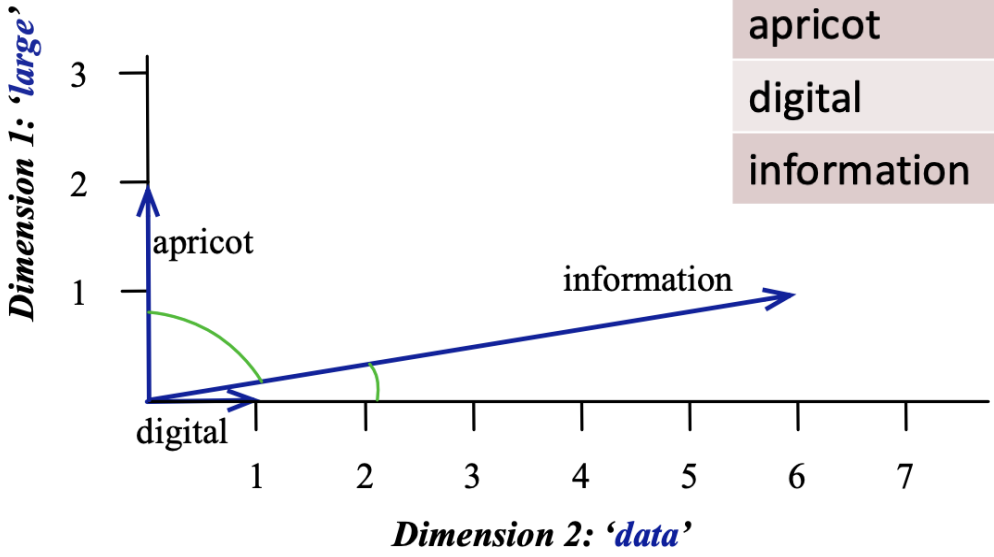
	large	data	computer
apricot	2	0	0
digital	0	1	2
information	1	6	1

- cosine(apricot,information) = $\frac{2+0+0}{\sqrt{2+0+0}\sqrt{1+36+1}} = \frac{2}{\sqrt{2}\sqrt{38}} = .23$
- cosine(digital,information) = $\frac{0+6+2}{\sqrt{0+1+4}\sqrt{1+36+1}} = \frac{8}{\sqrt{38}\sqrt{5}} = .58$
- cosine(apricot,digital) = $\frac{0+0+0}{\sqrt{1+0+0}\sqrt{0+1+4}} = 0$

Vector Semantics

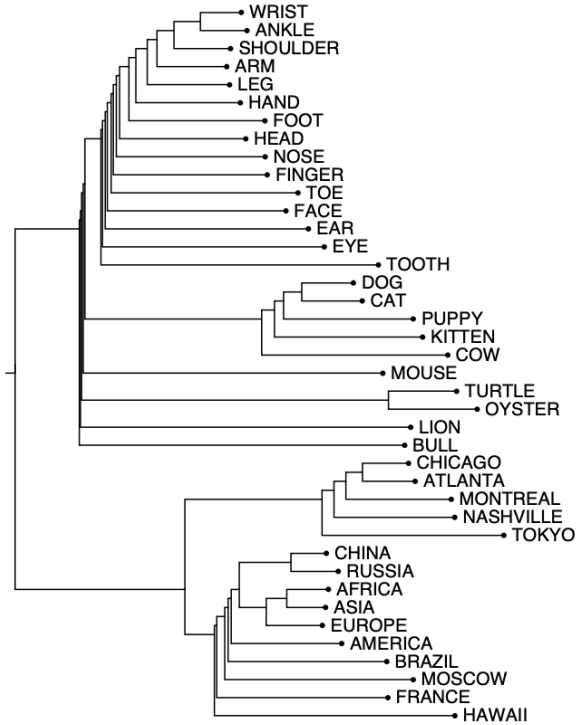
Visualizing vectors and angles (Why)

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



Vector Semantics

Clustering vectors to visualize similarity in co-occurrence matrices



Rohde et al. (2006)

Vector Semantics

Other possible similarity measures

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

$$\text{sim}_{\text{Jaccard}}(\vec{v},\vec{w}) = \frac{\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} \max(v_i,w_i)}$$

$$\text{sim}_{\text{Dice}}(\vec{v},\vec{w}) = \frac{2\times\sum_{i=1}^{N} \min(v_i,w_i)}{\sum_{i=1}^{N} (v_i+w_i)}$$

$$\text{sim}_{\text{JS}}(\vec{v}||\vec{w}) = D(\vec{v}|\frac{\vec{v}+\vec{w}}{2}) + D(\vec{w}|\frac{\vec{v}+\vec{w}}{2})$$

Vector Semantics

Evaluating similarity

- Intrinsic Evaluation:
  - Correlation between algorithm and human word similarity ratings
- Extrinsic (task-based, end-to-end) Evaluation:
  - Spelling error detection, WSD, essay grading
  - Taking TOEFL multiple-choice vocabulary tests

Levied is closest in meaning to which of these: imposed, believed, requested, correlated

Vector Semantics

Using syntax to define a word's context

- Zellig Harris (1968)

  "The meaning of entities, and the meaning of grammatical relations among them, is related to the restriction of combinations of these entities relative to other entities"

- Two words are similar if they have similar syntactic contexts

Duty and responsibility have similar syntactic distribution:

Modified by adjectives	additional, administrative, assumed, collective, congressional, constitutional …
Objects of verbs	assert, assign, assume, attend to, avoid, become, breach..

Vector Semantics

Co-occurrence vectors based on syntactic dependencies

Dekang Lin, 1998 "Automatic Retrieval and Clustering of Similar Words"

- Each dimension: a context word in one of R grammatical relations
- Subject-of- "absorb"
- Instead of a vector of |V| features, a vector of R|V|
- Example: counts for the word cell

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

	subj-of, absorb	subj-of, adapt	subj-of, behave	...	pobj-of, inside	pobj-of, into	...	nmod-of, abnormality	nmod-of, anemia	nmod-of, architecture	...	obj-of, attack	obj-of, call	obj-of, come from	obj-of, decorate	...	nmod, bacteria	nmod, body	nmod, bone marrow
cell	1	1	1		16	30		3	8	1		6	11	3	2		3	2	2

Vector Semantics

Syntactic dependencies for dimensions (Definition)

- Alternative (**Padó and Lapata 2007**):
  - Instead of having a |V| x R|V| matrix
  - Have a |V| x |V| matrix
  - But the co-occurrence counts aren't just counts of words in a window
  - But counts of words that occur in one of R dependencies (subject, object, etc).
  - So M("cell","absorb") = count(subj(cell,absorb)) + count(obj(cell,absorb)) + count(pobj(cell,absorb)), etc.

Vector Semantics

Syntactic dependencies for dimensions

Hindle, Don. 1990. Noun Classification from Predicate-Argument Structure. ACL

Object of "drink"	Count	PMI
tea	2	11.8
liquid	2	10.5
wine	2	9.3
anything	3	5.2
it	3	1.3

- "Drink it" more common than "drink wine"
- But "wine" is a better "drinkable" thing than "it"

Vector Semantics

Alternative to PPMI for measuring association (Definition)

- **tf-idf** (that's a hyphen not a minus sign)
- The combination of two factors
  - **Term frequency** (Luhn 1957): frequency of the word (can be logged)
  - **Inverse document frequency** (IDF) (Sparck Jones 1972)
    - N is the total number of documents
    - dfi = "document frequency of word i"
    - = # of documents with word I

$$\text{idf}_i = \log\left(\frac{N}{df_i}\right)$$

  - wij = word i in document j $w_{ij} = tf_{ij}\, idf_i$

Vector Semantics

tf-idf not generally used for word-word similarity

- But is by far the most common weighting when we are considering the relationship of words to documents

Vector Semantics

Sparse versus dense vectors (Definition)

- PPMI vectors are
  - long (length |V|= 20,000 to 50,000)
  - sparse (most elements are zero)
- Alternative: learn vectors which are
  - short (length 200-1000)
  - dense (most elements are non-zero)

Vector Semantics

Sparse versus dense vectors

- Q: Why dense vectors?
  - Short vectors may be easier to use as features in machine learning (less weights to tune)
  - Dense vectors may generalize better than storing explicit counts
  - They may do better at capturing synonymy:
    - car and automobile are synonyms; but are represented as distinct dimensions; this fails to capture similarity between a word with car as a neighbor and a word with automobile as a neighbor

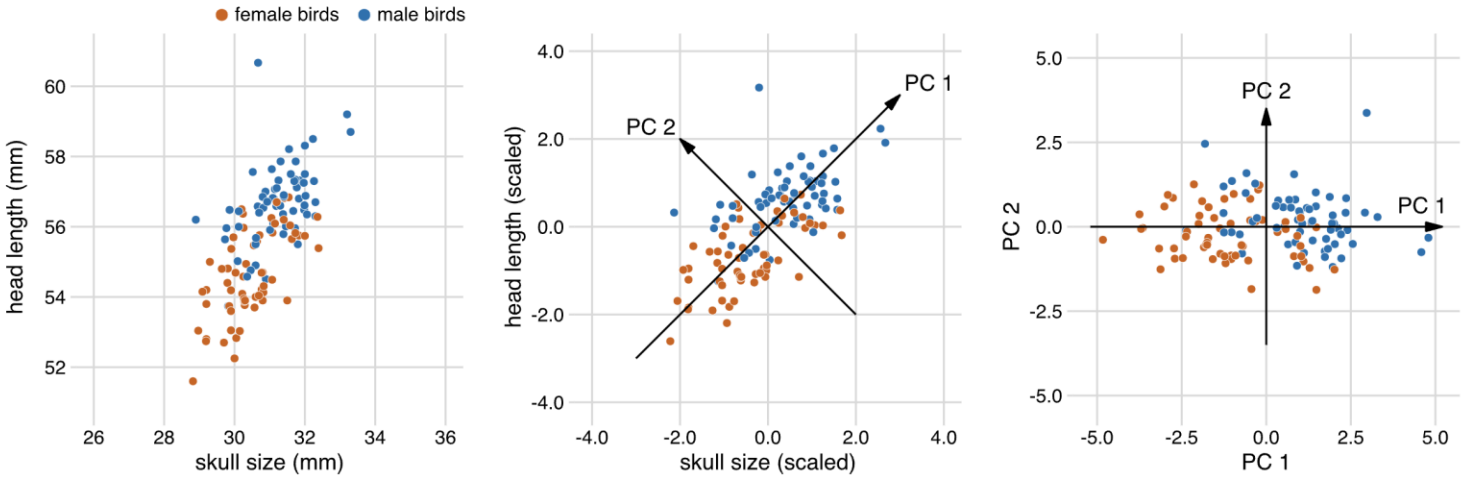
Vector Semantics

Three methods for getting short dense vectors

- Singular Value Decomposition (**SVD**)
  - A special case of this is called LSA – Latent Semantic Analysis
- "Neural Language Model"-inspired predictive models
  - **skip-grams** and **CBOW**
- Brown clustering

Vector Semantics

Dimensionality reduction



PCA aligns the major axes with directions of maximum variation in the data

Vector Semantics

Singular Value Decomposition

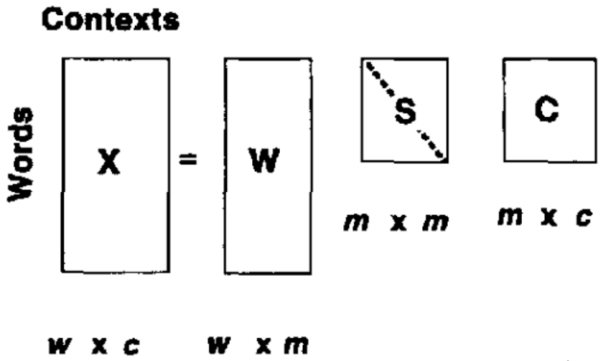
Any rectangular w x c matrix X equals the product of 3 matrices:

W: rows corresponding to original but m columns represents a dimension in a new latent space, such that

- M column vectors are orthogonal to each other
- Columns are ordered by the amount of variance in the dataset each new dimension accounts for

S: diagonal m x m matrix of singular values expressing the importance of each dimension.

C: columns corresponding to original but m rows corresponding to singular values



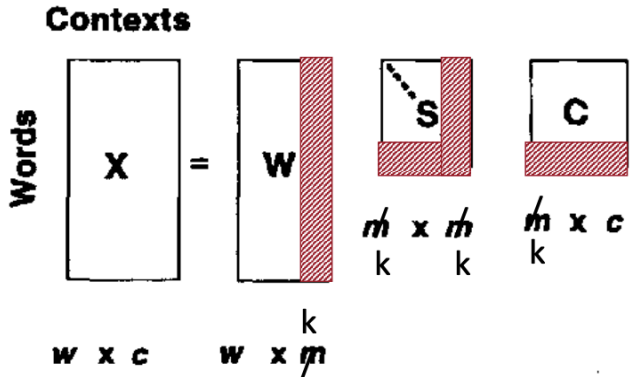
Landuaer and Dumais 1997

Vector Semantics

SVD applied to term-document matrix: Latent Semantic Analysis (Definition)

Deerwester et al (1988)

- If instead of keeping all m dimensions, we just keep the top k singular values. Let's say 300.
- The result is a least-squares approximation to the original X
- But instead of multiplying, we'll just make use of W.
- Each row of W:
  - A k-dimensional vector
  - Representing word W



Vector Semantics

LSA more details

- 300 dimensions are commonly used
- The cells are commonly weighted by a product of two weights
  - Local weight: Log term frequency
  - Global weight: either idf or an entropy measure

Vector Semantics

SVD applied to term-term matrix

- 300 dimensions are commonly used
- The cells are commonly weighted by a product of two weights
  - Local weight: Log term frequency
  - Global weight: either idf or an entropy measure

Vector Semantics

Dense Vectors via SVD

$$\underset{|V|\times|V|}{\begin{bmatrix} \\ X \\ \\ \\ \end{bmatrix}} = \underset{|V|\times|V|}{\begin{bmatrix} \\ W \\ \\ \\ \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_V \end{bmatrix}} \underset{|V|\times|V|}{\begin{bmatrix} \\ C \\ \\ \\ \end{bmatrix}}$$

Assume the matrix has rank |V|)

Vector Semantics

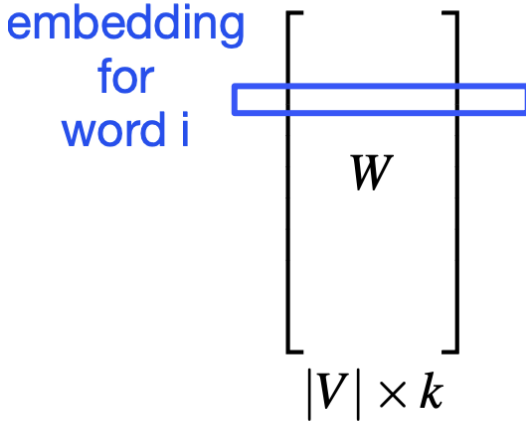
Truncated SVD on term-term matrix

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

Vector Semantics

Truncated SVD produces embeddings

- Each row of W matrix is a k-dimensional representation of each word w
- K might range from 50 to 1000
- Generally we keep the top k dimensions, but some experiments suggest that getting rid of the top 1 dimension or even the top 50 dimensions is helpful (Lapesa and Evert 2014).



Vector Semantics

Truncated SVD produces embeddings Embeddings versus sparse vectors

- Dense SVD embeddingssometimes work better than sparse PPMI matrices at tasks like word similarity
  - Denoising: low-order dimensions may represent unimportant information
  - Truncation may help the models generalize better to unseen data.
  - Having a smaller number of dimensions may make it easier for classifiers to properly weight the dimensions for the task.
  - Dense models may do better at capturing higher order cooccurrence.

Vector Semantics

Prediction-based models: An alternative way to get dense vectors

- **Skip-gram** (Mikolov et al. 2013a) **CBOW** (Mikolov et al. 2013b)
- Learn embeddings as part of the process of word prediction.
- Train a neural network to predict neighboring words
  - Inspired by neural net language models.
  - In so doing, learn dense embeddings for the words in the training corpus.
- Advantages:
  - Fast, easy to train (much faster than SVD)
  - Available online in the word2vec package
  - Including sets of pretrained embeddings!

Vector Semantics

Skip-grams

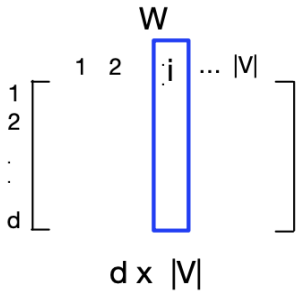
- Predict each neighboring word
  - in a context window of 2C words
  - from the current word.
- So for C=2, we are given word wt and predicting these 4 words

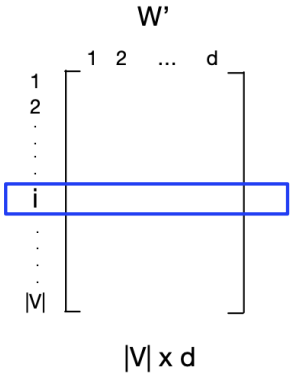
$$[w_{t-2}, w_{t-1}, w_{t+1}, w_{t+2}]$$

Vector Semantics

Skip-grams learn 2 embeddings for each w

- **Input embedding** v, in the input matrix W
  - Column i of the input matrix W is the 1×d embedding vi for word i in the vocabulary.
- **Output embedding** v′, in output matrix W’
  - Row i of the output matrix W′ is a d × 1 vector embedding v′ i for word i in the vocabulary.





Vector Semantics

Setup

- Walking through corpus pointing at word w(t), whose index in the vocabulary is j, so we'll call it wj ($1 < j < |V|$).
- Let's predict w(t+1) , whose index in the vocabulary is k ($1 < k < |V|$). Hence our task is to compute P(wk|wj).

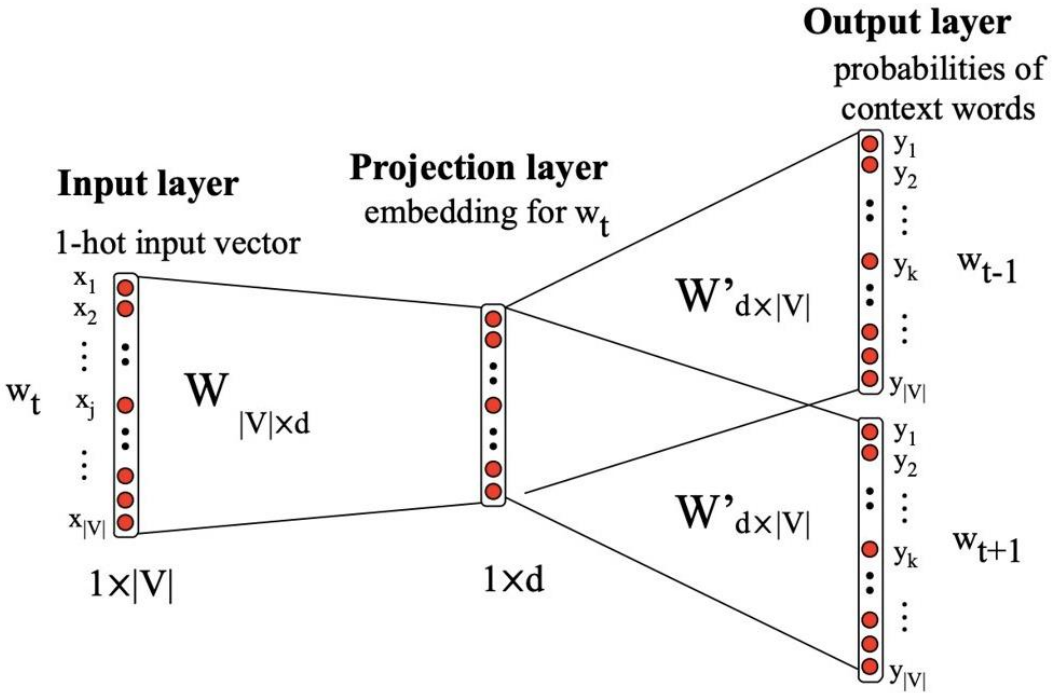
Vector Semantics

One-hot vectors

- A vector of length |V|
- 1 for the target word and 0 for other words
- So if “popsicle” is vocabulary word 5
- The one-hot vector is
- [0,0,0,0,1,0,0,0,0…….0]

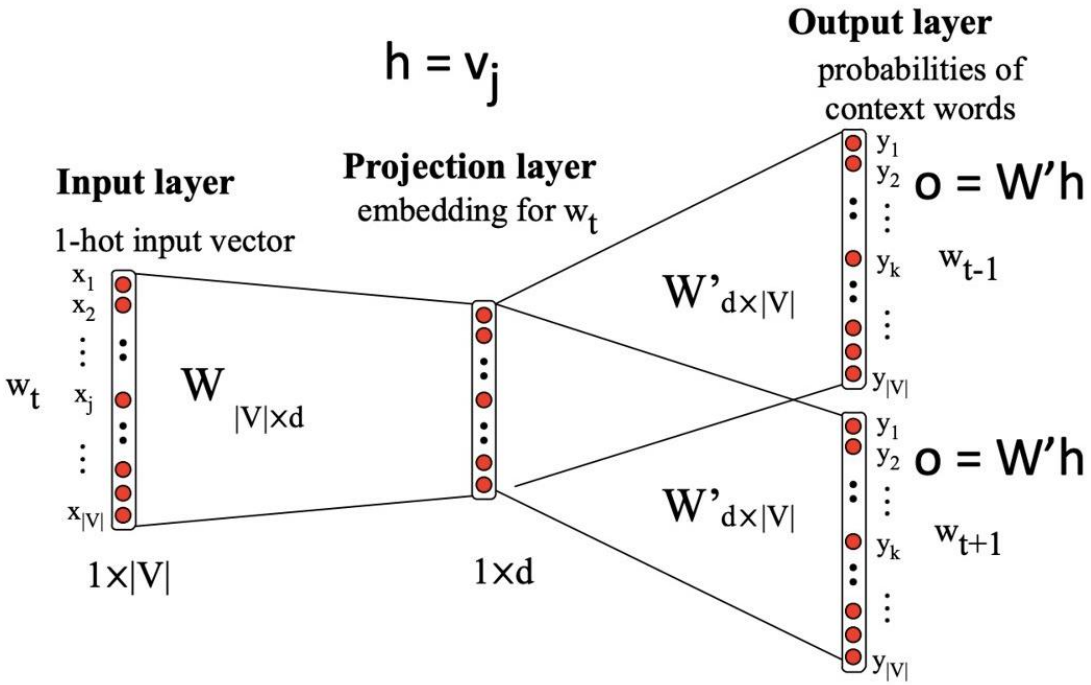
Vector Semantics

Skip-gram



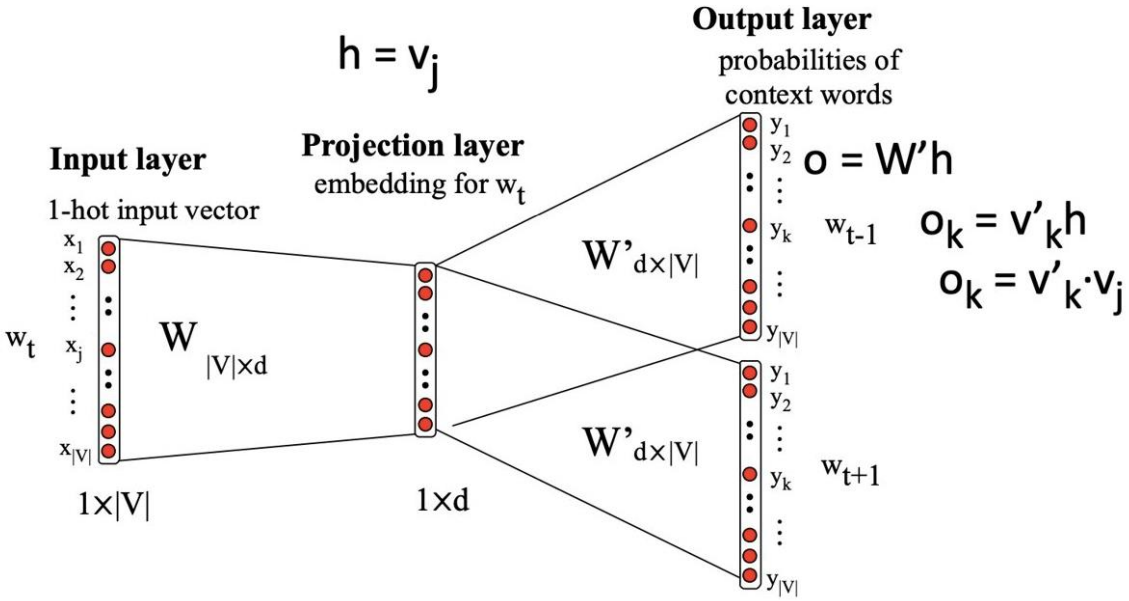
Vector Semantics

Skip-gram



Vector Semantics

Skip-gram



Vector Semantics

Turning outputs into probabilities

- $O_k = v'_k \cdot v_j$
- We use softmax to turn into probabilities

$$p(w_k|w_j) = \frac{exp(v'_k \cdot v_j)}{\sum_{w' \in |V|} exp(v'_w \cdot v_j)}$$

Vector Semantics

Embeddings from W and W’

- Since we have two embeddings, v_j and v'_j for each word w_j
- We can either:
  - Just use v_j
  - Sum them
  - Concatenate them to make a double-length embedding

Vector Semantics

Q: How do we learn the embeddings?

$$\underset{\theta}{\text{argmax}} \ \log \ p(\text{Text})$$

$$\underset{\theta}{\text{argmax}} \log \prod_{t=1}^{T} p(w^{(t-C)}, \ldots, w^{(t-1)}, w^{(t+1)}, \ldots, w^{(t+C)})$$

$$\underset{\theta}{\text{argmax}} \sum_{-c \le j \le c, j \ne 0} \log p(w^{(t+j)} | w^{(t)})$$

$$= \underset{\theta}{\text{argmax}} \sum_{t=1}^{T} \sum_{-c \le j \le c, j \ne 0} \log \frac{exp(v'^{(t+j)} \cdot v^{(t)})}{\sum_{w \in |V|} exp(v'_w \cdot v^{(t)})}$$

$$= \underset{\theta}{\text{argmax}} \sum_{t=1}^{T} \sum_{-c \le j \le c, j \ne 0} \left[v'^{(t+j)} \cdot v^{(t)} - \log \sum_{w \in |V|} exp(v'_w \cdot v^{(t)}) \right]$$

Vector Semantics

Relation between skip-grams and PMI!

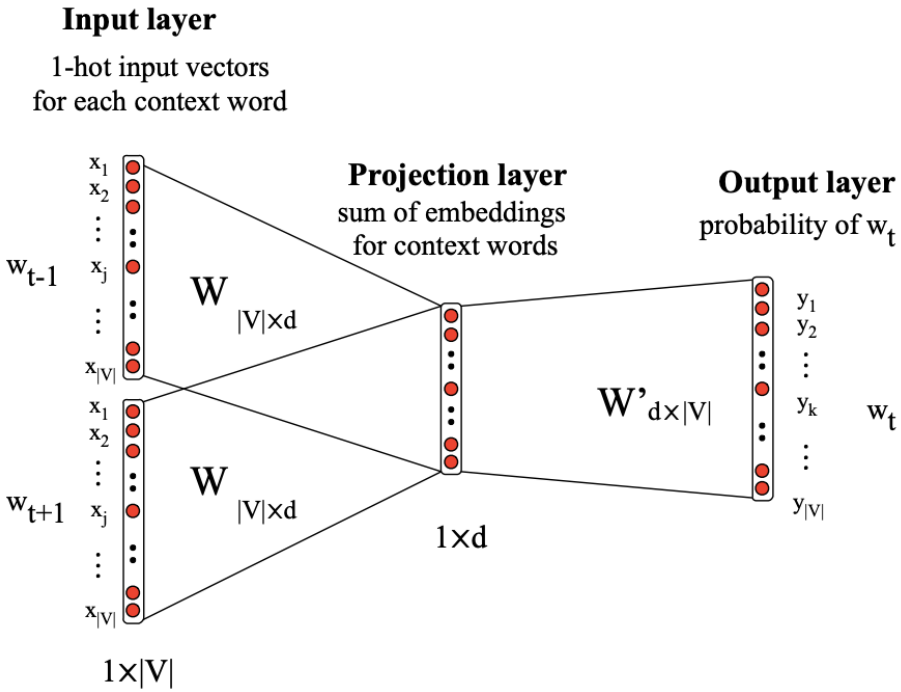
- If we multiply WW'T
- We get a |V|x|V| matrix M , each entry m_{ij} corresponding to some association between input word i and output word j
- Levy and Goldberg (2014b) show that skip-gram reaches its optimum just when this matrix is a shifted version of PMI:

$$WW'^{T} = M^{\text{PMI}} - \log k$$

- So skip-gram is implicitly factoring a shifted version of the PMI matrix into the two embedding matrices.

Vector Semantics

CBOW (Continuous Bag of Words)



Vector Semantics

Properties of embeddings

target:	Redmond	Havel	ninjutsu	graffiti	capitulate
	Redmond Wash.	Vaclav Havel	ninja	spray paint	capitulation
	Redmond Washington	president Vaclav Havel	martial arts	grafitti	capitulated
	Microsoft	Velvet Revolution	swordsmanship	taggers	capitulating

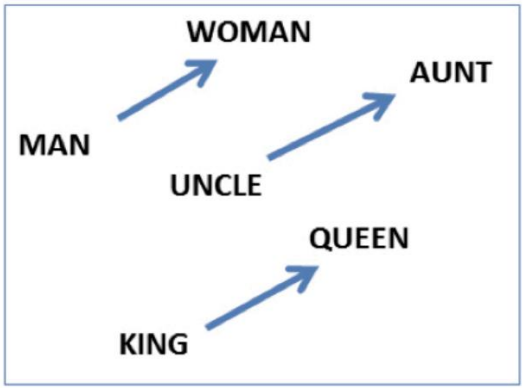
Nearest words to some embeddings (Mikolov et al. 20131)

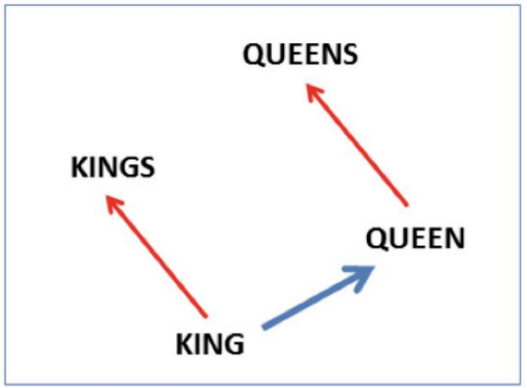
Vector Semantics

Embeddings capture relational meaning!

vector(*'king'*) - vector(*'man'*) + vector(*'woman'*) ≈ vector('queen')

vector(*'Paris'*) - vector(*'France'*) + vector(*'Italy'*) ≈ vector('Rome')





Vector Semantics

Evaluating similarity

- Extrinsic (task-based, end-to-end) Evaluation:
  - Question Answering
  - Spell Checking
  - Essay grading
- Intrinsic Evaluation:
  - Correlation between algorithm and human word similarity ratings
    - Wordsim353: 353 noun pairs rated 0-10. sim(plane,car)=5.77
  - Taking TOEFL multiple-choice vocabulary tests
    - Levied is closest in meaning to: imposed, believed, requested, correlated

Python: NLP with SpaCy and NLTK

1. **SpaCy**:
    - **Tokenization**: Breaks down the text into individual tokens (words, punctuation).
    - **POS Tagging**: Identifies the part of speech (POS) for each word in the sentence.
    - **NER**: Extracts entities like names, locations, dates, etc.
    - **Dependency Parsing**: Shows grammatical relationships between words.
2. **NLTK**:
    - **Tokenization**: Splits the text into tokens.
    - **POS Tagging**: Tags each token with its grammatical role.
    - **NER**: Identifies named entities (e.g., people, organizations).

```python
import spacy
import nltk

# Load SpaCy model
nlp = spacy.load("en_core_web_sm")

# Example text
text = "SpaCy is an excellent library for NLP, and NLTK offers great features for linguistic analysis."

### Using SpaCy for NLP ###

# Processing text with SpaCy
doc = nlp(text)

# Tokenization with SpaCy
print("Tokenization (SpaCy):")
for token in doc:
    print(f'{token.text} -> {token.pos_}')

# Named Entity Recognition (NER) with SpaCy
print("\nNamed Entities (SpaCy):")
for ent in doc.ents:
    print(f'{ent.text} -> {ent.label_}')

# Dependency Parsing with SpaCy
print("\nDependency Parsing (SpaCy):")
for token in doc:
    print(f'{token.text} ({token.dep_}) <- {token.head.text}')

```

✓ 0.4s

```
Tokenization (SpaCy):
SpaCy -> PROPN
is -> AUX
an -> DET
excellent -> ADJ
library -> NOUN
for -> ADP
NLP -> PROPN
, -> PUNCT
and -> CCONJ
NLTK -> PROPN
offers -> VERB
great -> ADJ
features -> NOUN
for -> ADP
linguistic -> ADJ
analysis -> NOUN
. -> PUNCT

Named Entities (SpaCy):
SpaCy -> PERSON
NLP -> ORG
NLTK -> ORG

Dependency Parsing (SpaCy):
...
for (prep) <- features
linguistic (amod) <- analysis
analysis (pobj) <- for
. (punct) <- offers
```

Python: NLP with SpaCy and NLTK

```
Tokenization (NLTK):
['SpaCy', 'is', 'an', 'excellent', 'library', 'for', 'NLP', ',', 'and', 'NLTK', 'offers', 'great',
'features', 'for', 'linguistic', 'analysis', '.']

POS tagging (NLTK):
[('SpaCy', 'NNP'), ('is', 'VBZ'), ('an', 'DT'), ('excellent', 'JJ'), ('library', 'NN'), ('for', 'IN'),
('NLP', 'NNP'), (',', ','), ('and', 'CC'), ('NLTK', 'NNP'), ('offers', 'VBZ'), ('great', 'JJ'),
('features', 'NNS'), ('for', 'IN'), ('linguistic', 'JJ'), ('analysis', 'NN'), ('.', '.')]
```

```python
### Using NLTK for NLP ###

# Tokenization with NLTK
tokens = nltk.word_tokenize(text)
print("\nTokenization (NLTK):")
print(tokens)

# POS tagging with NLTK
pos_tags = nltk.pos_tag(tokens)
print("\nPOS tagging (NLTK):")
print(pos_tags)

# Named Entity Recognition (NER) with NLTK
nltk.download('maxent_ne_chunker')
nltk.download('words')
print("\nNamed Entities (NLTK):")
named_entities = nltk.ne_chunk(pos_tags)
print(named_entities)
```

✓ 0.2s