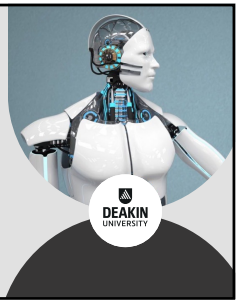
SIT330-770: Natural Language Processing

Week 6 - Vector Semantics and Embeddings

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

reda.bouadjenek@deakin.edu.au



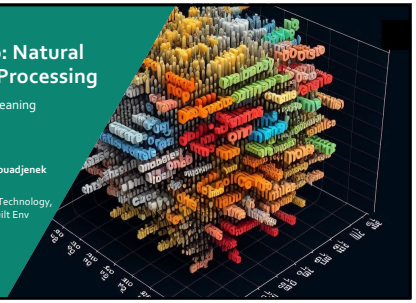
1

SIT330-770: Natural Language Processing

Week 6.1 - Word Meaning

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env



2

What do words mean?

- N-gram or text classification methods we've seen so far
  - Words are just strings (or indices w_i in a vocabulary list)
  - That's not very satisfactory!
- Introductory logic classes:
  - The meaning of "dog" is DOG; cat is CAT
    $\forall x\ DOG(x) \rightarrow MAMMAL(x)$
- Old linguistics joke by Barbara Partee in 1967:
  - Q: What's the meaning of life?
  - A: LIFE
- That seems hardly better!

3

Desiderata

- What should a theory of word meaning do for us?
- Let's look at some desiderata
- From lexical semantics, the linguistic study of word meaning

4

Lemmas and senses

lemma

mouse (N)

sense

1. any of numerous small rodents...
2. a hand-operated device that controls a cursor...

Modified from the online thesaurus WordNet

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

5

Relations between senses: Synonymy

- Synonyms have the same meaning in some or all contexts.
  - filbert / hazelnut
  - couch / sofa
  - big / large
  - automobile / car
  - vomit / throw up
  - water / H_2o

6

Relations between senses: Synonymy

- Note that there are probably no examples of perfect synonymy.
  - Even if many aspects of meaning are identical
  - Still may differ based on politeness, slang, register, genre, etc.

7

Relation: Synonymy?

water/H_2o

"H_2o" in a surfing guide?

big/large

my big sister != my large sister

8

The Linguistic Principle of Contrast

- Difference in form → difference in meaning

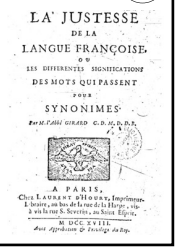
9

Abbé Gabriel Girard 1718

Re: "exact" synonyms

je ne crois pas qu'il y ait de-
mot fynonime dans aucune
Langue. Je le dis par con-

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


LA JUSTESSE
DE LA
LANGUE FRANÇOISE,
OU
LES DIFFERENTES SIGNIFICATIONS
DES MOTS QUI PASSENT
POUR
SYNONIMES.
A PARIS,
M. DCC. XVIII.

Thanks to Mark Aronoff!

10

Relation: Similarity

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

`car, bicycle`

`cow, horse`

11

Ask humans how similar 2 words are

word1	word2	similarity
vanish	disappear	9.8
behave	obey	7.3
belief	impression	5.95
muscle	bone	3.65
modest	flexible	0.98
hole	agreement	0.3

SimLex-999 dataset (Hill et al., 2015)

12

Relation: Word relatedness

- Also called "word association"
- Words can be related in any way, perhaps via a semantic frame or field
  - coffee, tea: **similar**
  - coffee, cup: **related**, not similar

13

13

Semantic field

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

hospitals
surgeon, scalpel, nurse, anaesthetic, hospital
restaurants
waiter, menu, plate, food, menu, chef
houses
door, roof, kitchen, family, bed

14

14

Relation: Antonymy

- Senses that are opposites with respect to only one feature of meaning
- Otherwise, they are very similar!

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

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

15

15

Connotation (sentiment)

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

16

16

Connotation

Osgood et al. (1957)

- Words seem to vary along 3 affective dimensions:
  - **valence**: the pleasantness of the stimulus
  - **arousal**: the intensity of emotion provoked by the stimulus
  - **dominance**: the degree of control exerted by the stimulus

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

Values from NRC VAD Lexicon (Mohammad 2018)

17

17

So far

- **Concepts** or word senses
  - Have a complex many-to-many association with **words** (homonymy, multiple senses)
- Have relations with each other
  - Synonymy
  - Antonymy
  - Similarity
  - Relatedness
  - Connotation

18

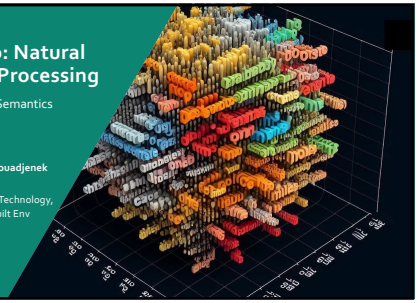
18

SIT330-770: Natural Language Processing

Week 6.2 - Vector Semantics

Dr. Mohamed Reda Bouadjenek

School of Information Technology,
Faculty of Sci Eng & Built Env



19

Computational models of word meaning

- Can we build a theory of how to represent word meaning, that accounts for at least some of the desiderata?
- We'll introduce **vector semantics**
  - The standard model in language processing!
  - Handles many of our goals!

20

Ludwig Wittgenstein

- PI #43:

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

21

Let's define words by their usages

- One way to define "usage":
- words are defined by their environments (the words around them)
- Zellig Harris (1954):
- **If A and B have almost identical environments we say that they are synonyms**.

22

What does recent English borrowing *ongchoi* mean?

- Suppose you see these sentences:
  - Ong choi is delicious **sautéed with garlic**.
  - Ong choi is superb **over rice**
  - Ong choi **leaves** with salty sauces
- And you've also seen these:
  - …spinach **sautéed with garlic over rice**
  - Chard stems and **leaves** are **delicious**
  - Collard greens and other **salty** leafy greens
- Conclusion:
  - Ongchoi is a leafy green like spinach, chard, or collard greens
    - We could conclude this based on words like "leaves" and "delicious" and "sauteed"

23

Ongchoi: *Ipomoea aquatica "Water Spinach"*

空心菜
kangkong
rau muống
…

Yamaguchi, Wikimedia Commons, public domain

24

Idea 1: Defining meaning by linguistic distribution

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

25

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

- 3 affective dimensions for a word
  - **valence**: pleasantness
  - **arousal**: intensity of emotion
  - **dominance**: the degree of control exerted

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

NRC VAD Lexicon (Mohammad 2018)

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

26

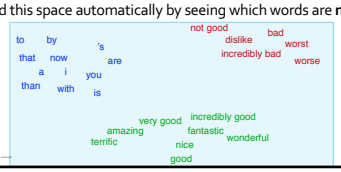
Idea 1: Defining meaning by linguistic distribution

Idea 2: Meaning as a point in multidimensional space

27

Defining meaning as a point in space based on distribution

- Each word = a vector (not just "good" or "w_{45}")
- Similar words are "**nearby in semantic space**"
- We build this space automatically by seeing which words are **nearby in text**



28

We define meaning of a word as a vector

- Called an "embedding" because it's embedded into a space (see textbook)
- The standard way to represent meaning in NLP
- **Every modern NLP algorithm uses embeddings as the representation of word meaning**
- Fine-grained model of meaning for similarity

29

Intuition: why vectors?

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

30

We'll discuss 2 kinds of embeddings

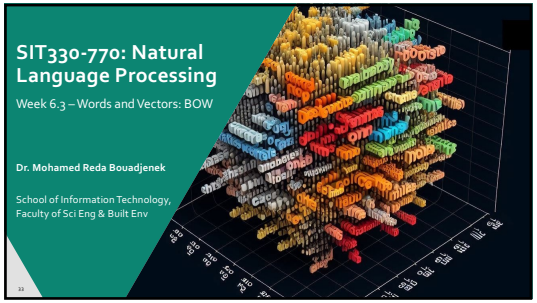
- tf-idf
  - Information Retrieval workhorse!
  - A common baseline model
  - **Sparse** vectors
  - Words are represented by (a simple function of) the **counts** of nearby words
- Word2vec
  - **Dense** vectors
  - Representation is created by training a classifier to **predict** whether a word is likely to appear nearby
  - Later we'll discuss extensions called **contextual embeddings**

31

From now on: Computing with meaning representations instead of string representations

荃者所以在鱼，得鱼而忘荃　Nets are for fish;
Once you get the fish, you can forget the net.
言者所以在意，得意而忘言　Words are for meaning;
Once you get the meaning, you can forget the words
庄子(Zhuangzi), Chapter 26

32



33

Bag of Words

- A document is represented as vector of words.
  - One dimension per word.
  - Vector size is the vocabulary size, e.g., English may contain 100k words.
  - Different weighting schemas can be used, e.g., tf, log(tf), tf-idf, Boolean, etc.
  - Sparse vector, e.g., almost all values are zeros.

34

Order matters for NLP tasks!

- Assumes independence between words:
  - The sentences "**John likes Mary**" has the same representation as "**Mary likes John**" – even though the semantic is different).
- May work well for Information Retrieval tasks, but not for NLP tasks!
  - Sentiment analysis:
    "Ah **no**, there are good movies on Netflix!" vs. "Ah, there are **no** good movies on Netflix!"

35

Order matters for NLP tasks!

"Ah **no**, there are good movies on Netflix!" vs. "Ah, there are **no** good movie on Netflix!"

- Use N-grams.
  - Dimensionality grows exponentially V^N.
  - 3-grams with English: $(10^5)^3 = 10^{15} = 1{,}000{,}000{,}000{,}000{,}000$ entries.
  - Too expensive!

36

Computing word similarity: Dot product and cosine

- The dot product between two vectors is a scalar:

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

- The dot product tends to be high when the two vectors have large values in the same dimensions
- Dot product can thus be a useful similarity metric between vectors

37

37

Problem with raw dot-product

- Dot product favors long vectors
- Dot product is higher if a vector is longer (has higher values in many dimension)
- Vector length:

- Frequent words (of, the, you) have long vectors (since they occur many times with other words).
- So dot product overly favors frequent words

38

38

Alternative: cosine for computing word similarity

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

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

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

39

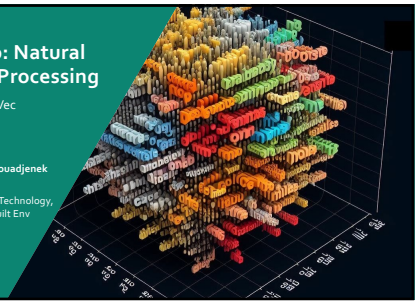
39

SIT330-770: Natural Language Processing

Week 6.4 – Word2Vec

Dr. Mohamed Reda Bouadjenek

School of Information Technology,
Faculty of Sci Eng & Built Env



40

40

Sparse versus dense vectors

- tf-idf (or PMI) vectors are
  - **long** (length |V|= 20,000 to 50,000)
  - **sparse** (most elements are zero)
- Alternative: learn vectors which are
  - **short** (length 50-1000)
  - **dense** (most elements are non-zero)

41

41

Sparse versus dense vectors

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

42

42

Common methods for getting short dense vectors

- "Neural Language Model"-inspired models
  - Word2vec (skipgram, CBOW), GloVe
- Singular Value Decomposition (SVD)
  - A special case of this is called LSA – Latent Semantic Analysis
- Alternative to these "static embeddings":
  - Contextual Embeddings (ELMo, BERT)
  - Compute distinct embeddings for a word in its context
  - Separate embeddings for each token of a word

43

43

Simple static embeddings you can download!

- Word2vec (Mikolov et al)
- https://code.google.com/archive/p/word2vec/

- GloVe (Pennington, Socher, Manning)
- http://nlp.stanford.edu/projects/glove/

44

44

Word2vec

- Popular embedding method
- Very fast to train
- Code available on the web
- Idea: **predict** rather than **count**
- Word2vec provides various options. We'll do:
- **skip-gram with negative sampling (SGNS)**

45

45

Word2Vec

- Instead of **counting** how often each word *w* occurs near "*apricot*"
  - Train a classifier on a binary **prediction** task:
  - Is *w* likely to show up near "*apricot*"?
- We don't actually care about this task
  - But we'll take the learned classifier weights as the word embeddings
- Big idea: **self-supervision**:
  - A word c that occurs near apricot in the corpus cats as the gold "correct answer" for supervised learning
  - No need for human labels
  - Bengio et al. (2003); Collobert et al. (2011)

46

46

Approach: predict if candidate word *c* is a "neighbor"

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

47

47

Skip-Gram Training Data

- (assuming a +/- 2 word window)
  ...lemon, a [tablespoon of apricot jam, a] pinch...
  c1 c2 [target] c3 c4
- Goal: train a classifier that is given a candidate (**w**ord, **c**ontext) pair
  (apricot, jam)
  (apricot, aardvark)
  ...
- And assigns each pair a probability:
  - $P(+|w, c)$
  - $P(-|w, c) = 1 - P(+|w, c)$

48

48

Similarity is computed from dot product

- Remember: two vectors are similar if they have a high dot product
  - Cosine is just a normalized dot product
- So:
  - Similarity(w,c) $\propto w \cdot c$
- We'll need to normalize to get a probability
  - (cosine isn't a probability either)

49

Turning dot products into probabilities

- $\text{Sim}(w,c) \approx w \cdot c$
- To turn this into a probability
- We'll use the sigmoid from logistic regression:

$$P(+|w,c) = \sigma(c \cdot w) = \frac{1}{1+\exp(-c \cdot w)}$$

$$P(-|w,c) = 1 - P(+|w,c)$$
$$= \sigma(-c \cdot w) = \frac{1}{1+\exp(c \cdot w)}$$

50

How Skip-Gram Classifier computes *P(+|w, c)*

$$P(+|w,c) = \sigma(c \cdot w) = \frac{1}{1+\exp(-c \cdot w)}$$

- This is for one context word, but we have lots of context words.
- We'll assume independence and just multiply them:

$$P(+|w,c_{1:L}) = \prod_{i=1}^{L} \sigma(c_i \cdot w)$$

$$\log P(+|w,c_{1:L}) = \sum_{i=1}^{L} \log \sigma(c_i \cdot w)$$

51

Skip-gram classifier: summary

- A probabilistic classifier, given
  - a test target word *w*
  - its context window of *L* words $c_{1:L}$
- Estimates probability that w occurs in this window based on similarity of w (embeddings) to $c_{1:L}$ (embeddings).
- To compute this, we just need embeddings for all the words.

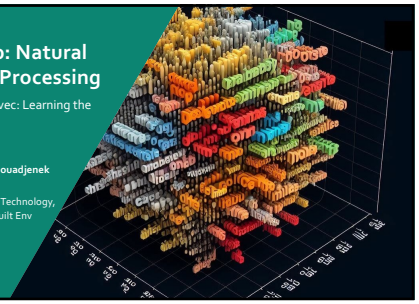
52

SIT330-770: Natural Language Processing

Week 6.5 – Word2vec: Learning the embeddings

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env



53

Skip-Gram Training data

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

c1 c2 [target] c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

54

Skip-Gram Training data

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

c1 c2 [target] c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

For each positive example we'll grab k negative examples, sampling by frequency

55

Skip-Gram Training data

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

c1 c2 [target] c3 c4

positive examples +

t	c
apricot	tablespoon
apricot	of
apricot	jam
apricot	a

negative examples -

t	c	t	c
apricot	aardvark	apricot	seven
apricot	my	apricot	forever
apricot	where	apricot	dear
apricot	coaxial	apricot	if

56

Word2vec: how to learn vectors

- Given the set of positive and negative training instances, and an initial set of embedding vectors
- The goal of learning is to adjust those word vectors such that we:
  - **Maximize** the similarity of the target word, context word pairs (w, c_{pos}) drawn from the positive data
  - **Minimize** the similarity of the (w, c_{neg}) pairs drawn from the negative data.

57

Loss function for one *w* with c_{pos}, c_{neg1} ...c_{negk}

- Maximize the similarity of the target with the actual context words, and minimize the similarity of the target with the *k* negative sampled non-neighbor words.

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

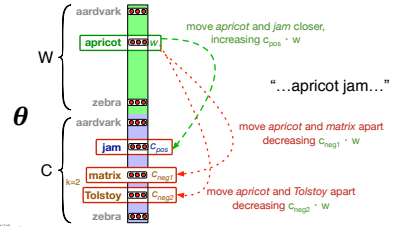
58

Learning the classifier

- How to learn?
  - Stochastic gradient descent!
- We'll adjust the word weights to
  - make the positive pairs more likely
  - and the negative pairs less likely,
  - over the entire training set.

59

Intuition of one step of gradient descent



60

Reminder: gradient descent

- At each step
  - Direction: We move in the reverse direction from the gradient of the loss function
  - Magnitude: we move the value of this gradient $\frac{d}{dw}L(f(x;w),y)$ weighted by a **learning rate** η
  - Higher learning rate means move *w* faster

$$w^{t+1} = w^t - \eta \frac{d}{dw} L(f(x;w),y)$$

61

The derivatives of the loss function

$$L_{CE} = -\left[\log \sigma(c_{pos} \cdot w) + \sum_{i=1}^{k} \log \sigma(-c_{neg_i} \cdot w)\right]$$

$$\frac{\partial L_{CE}}{\partial c_{pos}} = [\sigma(c_{pos} \cdot w) - 1]w$$

$$\frac{\partial L_{CE}}{\partial c_{neg}} = [\sigma(c_{neg} \cdot w)]w$$

$$\frac{\partial L_{CE}}{\partial w} = [\sigma(c_{pos} \cdot w) - 1]c_{pos} + \sum_{i=1}^{k} [\sigma(c_{neg_i} \cdot w)]c_{neg_i}$$

62

Update equation in SGD

- Start with randomly initialized C and W matrices, then incrementally do updates

$$c_{pos}^{t+1} = c_{pos}^t - \eta[\sigma(c_{pos}^t \cdot w^t) - 1]w^t$$

$$c_{neg}^{t+1} = c_{neg}^t - \eta[\sigma(c_{neg}^t \cdot w^t)]w^t$$

$$w^{t+1} = w^t - \eta\left[[\sigma(c_{pos} \cdot w^t) - 1]c_{pos} + \sum_{i=1}^{k} [\sigma(c_{neg_i} \cdot w^t)]c_{neg_i}\right]$$

63

Two sets of embeddings

- SGNS learns two sets of embeddings
  - Target embeddings matrix W
  - Context embedding matrix C
- It's common to just add them together, representing word *i* as the vector $w_i + c_i$

64

Summary: How to learn word2vec (skip-gram) embeddings

- Start with V random d-dimensional vectors as initial embeddings
- Train a classifier based on embedding similarity
  - 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.

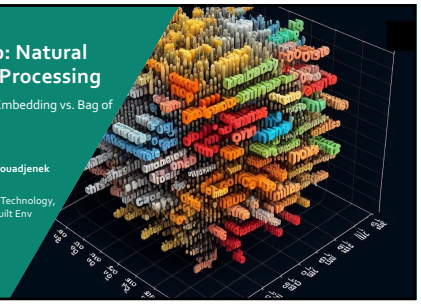
65

SIT330-770: Natural Language Processing

Week 6.6 – Word Embedding vs. Bag of Words

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env



66

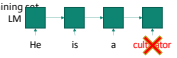
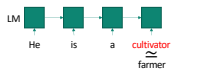
Word Embedding vs. Bag of Words

Traditional Method - Bag of Words Model	Word Embeddings

Two approaches:
• Either uses one hot encoding.
  ◦ Each word in the vocabulary is represented by one bit position in a HUGE vector.
  ◦ For example, if we have a vocabulary of 10,000 words, and "aardvark" is the *4th word in the dictionary*, it would be represented by: [0 0 0 1 0 0 . . . . . 0 0 0].
• Or uses document representation.
  ◦ Each word in the vocabulary is represented by its presence in documents.
  ◦ For example, if we have a corpus of 1M documents, and "Hello" is in 1th, 3th and 5th documents *only*, it would be represented by: [1 0 1 0 1 0 . . . . . 0 0 0].
• Assumes independence between words.
	• Stores each word in as a point in space, where it is represented by a dense vector of fixed number of dimensions (generally 300) .
	  ◦ For example, "Hello" might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02].
	  ◦ Dimensions are projections along different axes, more of a mathematical concept.
	• Unsupervised, built just by reading huge corpus.
	• Assumes dependence between words.

67

Word Embedding vs. Bag of Words

Traditional Method - Bag of Words Model	Word Embeddings
• Requires very large weight matrix for 1st layers.  W's size is 10,000x100 = 10^6 • Models not flexible with unseen words in the training set.  	• A compact weight matrix for 1st layers.  W's size is 300x100 = $3x10^4$ • Flexible models with unseen words in the training set. 

68

SIT330-770: Natural Language Processing

Week 6.7 – Properties of Embeddings

Dr. Mohamed Reda Bouadjenek

School of Information Technology, Faculty of Sci Eng & Built Env

69

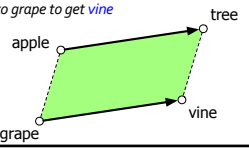
The kinds of neighbors depend on window size

- **Small windows** (C= +/- 2) : nearest words are syntactically similar words in same taxonomy
  - *Hogwarts* nearest neighbors are other fictional schools
  - *Sunnydale, Evernight, Blandings*
- **Large windows** (C= +/- 5) : nearest words are related words in same semantic field
  - *Hogwarts* nearest neighbors are Harry Potter world:
  - *Dumbledore, half-blood, Malfoy*

70

Analogical relations

- The classic parallelogram model of analogical reasoning (Rumelhart and Abrahamson 1973)
- To solve: *"apple is to tree as grape is to _____"*
- *Add tree – apple to grape to get vine*



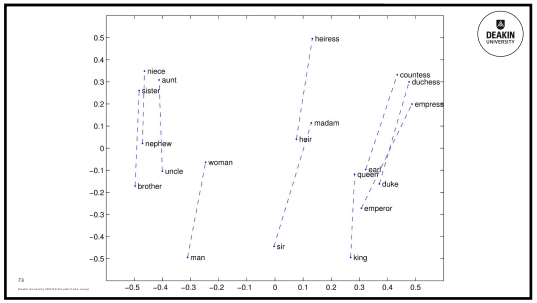
71

Analogical relations via parallelogram

- The parallelogram method can solve analogies with both sparse and dense embeddings (Turney and Littman 2005, Mikolov et al. 2013b)
- king – man + woman is close to queen
- Paris – France + Italy is close to Rome
- For a problem a:a*::b:b*, the parallelogram method is:

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

72



73

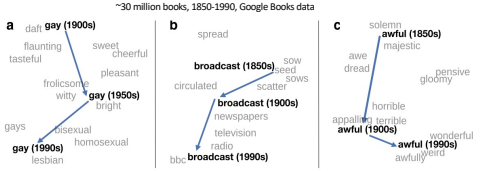
Caveats with the parallelogram method

- It only seems to work for frequent words, small distances and certain relations (relating countries to capitals, or parts of speech), but not others. (Linzen 2016, Gladkova et al. 2016, Ethayarajh et al. 2019a)
- Understanding analogy is an open area of research (Peterson et al. 2020)

74

Embeddings as a window onto historical semantics

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



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

75

Embeddings reflect cultural bias!

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

- Ask "Paris : France :: Tokyo : x"
  - x = Japan
- Ask "father : doctor :: mother : x"
  - x = nurse
- Ask "man : computer programmer :: woman : x"
  - x = homemaker

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

76

Historical embedding as a tool to study cultural biases

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

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

77