Applied Natural Language Processing

Dr Jeff Mitchell, University of Sussex

Autumn 2025

Lexical semantics

Previously

- Ambiguity and word senses
- Lexical semantic relationships
  - Synonymy, antonymy, hypernymy, hyponymy

This time

- Distributional semantics
  - What is it?
  - How do we compute it?
  - What are the applications?
  - What are the advantages and disadvantages?

Distributional Semantics

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

Firth (1957)

The Distributional Hypothesis: "Words that occur in the same contexts tend to have similar meanings."

Harris (1954)

What does *tezguino* mean?

1. A bottle of ***tezguino*** is on the table.
2. Everyone likes ***tezguino***.
3. ***Tezguino*** makes you drunk.
4. We make ***tezguino*** out of corn.

(Lin, 1998)

Bootstrapping the Semantics of Unknown words

- The **contexts** in which *tezguino* is used suggest that *tezguino* may be:
  - *A kind of alcoholic beverage made from corn mash*
- Similarity plays an important role in word acquisition (Gentner, 1982)
- Can we use corpora to infer similarity between words i.e., infer that *tezguino* is similar to *beer*, *wine*, *vodka* etc?

Applications of Distributional Semantics

- Automatic thesaurus construction
  - For any language, genre, domain … where we have a corpus
- Overcoming data sparseness in models which require labelled training data

Distributional Semantics in Document classification

- Imagine we have built a Naïve Bayes document relevancy classifier using a relatively small training sample (e.g., 500 documents)
- A test document contains the word *tezguino* which has not been seen in the training sample
  - so it cannot contribute to the relevancy classification
- But by applying distributional semantics to a very large unlabeled corpus (e.g., the web), we know that *tezguino* is very similar to *beer*
  - *beer* has been seen in the training sample
  - Assume **P(*tezguino* | class) ≈ P(*beer* | class)**

Why Does it Work?

- Tigers **eat** meat.
- The monkey **ate** a banana.
- X17 likes to **eat** falafel.
- My son does not **eat** courgettes.
- The machine **ate** my credit card.

From these examples we can learn:

- What can be **eaten**?
- What **eats** things?
- *Meat, banana, falafel, courgettes* and *credit card* all share 1 facet of meaning – that they can be eaten
- *Tigers, monkey, X17, son* and *machine* all share 1 facet of meaning – that they eat things

Features for Capturing Facets of Meaning

- Dependency relationships between words:
  - "is subject of *eat*"
  - "is object of *eat*"
- Proximity between words
  - "occurs within a **window of +- m words** either side of the word *eat*"
- In both cases, feature values can be Boolean but are usually real-valued (more later)
- Dependency parsing is difficult
- Windows are easy to construct
- Window size can be varied to capture different types of semantic relationships
  - More in the lab exercises

Context windows

window size around target word = +-1

Features added per target word

The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card

the: {machine: 1}

machine: {the: 1, ate: 1}

ate: {machine: 1, my: 1}

What features will be added for *my*, *credit* and *card*?

Context windows

window size around target word = +-2

The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card
The	machine	ate	my	credit	card

Features added per target word

the: {machine: 1, ate: 1}

machine: {the: 1, ate: 1, my: 1}

ate: {the: 1, machine: 1, my: 1, credit: 1}

What features will be added for *my*, *credit* and *card*?

Distributional representations based on frequency

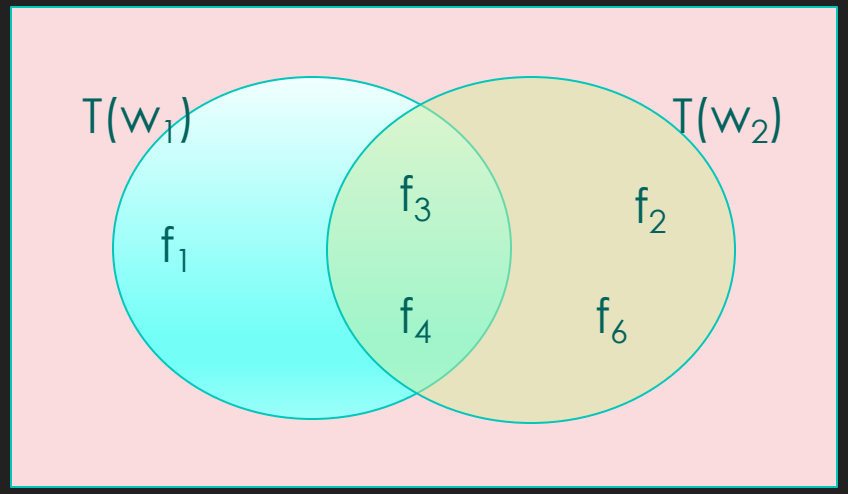
Use windowing to extract and count features for all words in a large corpus i.e., distributional representations or vectors

feature	banana	meat	credit	Total
yellow	10	2	3	**15**
red	2	14	19	**35**
eat	20	9	1	**30**
spend	1	2	27	**30**
card	3	2	50	**55**
the	25	25	50	**100**
is	20	20	40	**80**
tiger	3	17	0	**20**
man	6	9	10	**25**
monkey	10	0	0	**10**
Total	**100**	**100**	**200**	**400**

Similarity Measures: Jaccard's measure

- Set-theoretic similarity measure
- Intuitively measures the amount of overlap between feature sets i.e., the intersection, with respect to their potential for overlap.

Let ***T(w)*** be the set of features of w



$$Jaccard(w_1, w_2) = \frac{|T(w_1) \cap T(w_2)|}{|T(w_1) \cup T(w_2)|}$$

$$= \frac{|T(w_1) \cap T(w_2)|}{|T(w_1)| + |T(w_2)| - |T(w_1) \cap T(w_2)|}$$

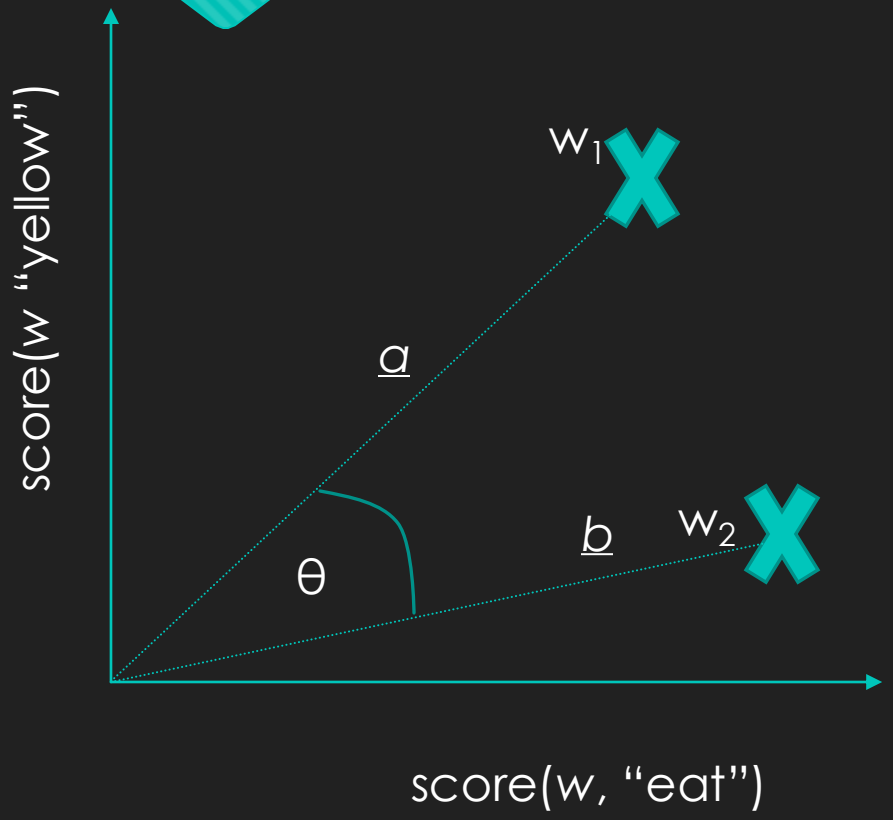
$$=?$$

Calculating Jaccard

feature	banana	meat	|*intersection*|	|union|
yellow	10	2	2	10
red	2	14	2	14
eat	20	9	9	20
spend	1	2	1	2
card	3	2	2	3
the	25	25	25	25
is	20	20	20	20
tiger	3	17	3	17
man	6	9	6	9
monkey	10	0	0	10
Total	**100**	**100**	**70**	**130**

$$\text{jaccard}(banana, meat) = \frac{70}{130}$$

Cosine similarity



- The more similar two words are, the smaller the angle θ between their vectors will be.
- So:

$$sim(w_1, w_2) = \cos(\theta)$$

$$= \frac{\underline{a}.\underline{b}}{\sqrt{\underline{a}.\underline{a} \times \underline{b}.\underline{b}}}$$

dot product

Where:

$$\underline{a}.\underline{b} = \sum_{i}^{m} a_i b_i$$

m=number of dimensions

Calculating cosine

feature	banana	meat	*a.b*	a.a	b.b
yellow	10	2	20	100	4
red	2	14	28	4	196
eat	20	9	180	400	81
spend	1	2	2	1	4
card	3	2	6	9	4
the	25	25	625	625	625
is	20	20	400	400	400
tiger	3	17	51	9	289
man	6	9	54	36	81
monkey	10	0	0	100	0
Total	**100**	**100**	**1366**	**1684**	**1684**

$$\cos(banana, meat) = \frac{1366}{1684} = 0.81$$

Pointwise Mutual information (PMI)

- Frequency and/or simple conditional probability do not capture the intuition that some features are more informative than others
- *the* and *is* appear relatively frequently with all of the words
- so their contribution to similarity should be smaller
- PMI measures the amount of information gained by seeing a word and a feature together
- A feature which co-occurs with a target word more than we would expect (if words and features occurred independently) has more weight in the similarity calculation

Calculating PMI

$$I(w,f) = \log \frac{P(f \mid w)}{P(f)} \quad = \log \frac{P(f \square w)}{P(f) \square P(w)}$$

$$I(w,f) = \log \frac{\text{freq}(f,w) \times \text{freq}(*,*)}{\text{freq}(f,*) \times \text{freq}(*,w)}$$

grand total

row total

column total

Representations based on PMI

feature	banana	meat	credit	Total
yellow	10	2	3	15
red	2	14	19	35
eat	20	9	1	30
spend	1	2	27	30
card	3	2	50	55
the	25	25	50	100
is	20	20	40	80
tiger	3	17	0	20
man	6	9	10	25
monkey	10	0	0	10
Total	100	100	200	400

$$log \frac{10 \times 400}{15 \times 100}$$

feature	banana	meat	credit
yellow	1.42		
red			
eat			
spend			
card			
the			
is			
tiger			
man			
monkey			

Positive PMI (PPMI)

- What happens when frequency of co-occurrence is 0?
- PMI = negative infinity!!!
- positive PMI avoids this problem and similarity calculations on shared features rather than the sharing of absent features

$$\text{PPMI}(w,f) = \begin{cases} I(w,f) & I(w,f) > 0 \\ 0 & \text{otherwise} \end{cases}$$

Representations based on PPMI

feature	banana	meat	credit	Total
yellow	10	2	3	15
red	2	14	19	35
eat	20	9	1	30
spend	1	2	27	30
card	3	2	50	55
the	25	25	50	100
is	20	20	40	80
tiger	3	17	0	20
man	6	9	10	25
monkey	10	0	0	10
Total	**100**	**100**	**200**	**400**

$$log \frac{10 \times 400}{15 \times 100}$$

feature	banana	meat	credit
yellow	1.42	**0**	**0**
red	**0**	0.68	0.12
eat	1.42	0.26	**0**
spend			
card			
the			
is			
tiger			**0**
man			
monkey		**0**	**0**

Question for you

- The table shows the number of times tennis and player have co-occurred together in a small corpus, along with the respective totals for other co-occurrences.

	tennis	…	TOTAL
player	1	…	500
….	…	…	
TOTAL	250		100000

What is the PPMI(tennis,player)?

Assuming the dictionary representations on the right exist, which of the following would correctly convert them into PPMI scores?

```
import math

reps={'tennis':{'player':1}}
wordtotals={'tennis':250}
feattotals={'player':500}
grandtotal=100000
```

A

```
ppmi={word:{feat:min(0,math.log((value*grandtotal)/(wordtotals[word]*feattotals[feat]),2))
          for feat,value in rep.items()} for word,rep in reps.items()}
```

B

```
ppmi={word:{feat:math.log((value*grandtotal)/(wordtotals[word]*feattotals[feat]))
          for feat,value in rep.items()} for word,rep in reps.items()}
```

C

```
ppmi={word:{feat:max(0,math.log(value*grandtotal/wordtotals[word]*feattotals[feat],2))
          for feat,value in rep.items()} for word,rep in reps.items()}
```

D

```
ppmi={word:{feat:max(0,math.log((value*grandtotal)/(wordtotals[word]*feattotals[feat]),2))
          for feat,value in rep.items()} for word,rep in reps.items()}
```

E

```
ppmi={word:{feat:math.log((value*grandtotal)/(wordtotals[word]*feattotals[feat]))
          for feat,value in rep} for word,rep in reps.items()}
```

Distributional Semantics: Part 2

Distributional semantics

So far

- The distributional hypothesis: words that occur in the same contexts tend to have similar meanings
- Distributional representations based on context
- Positive Pointwise Mutual Information
- Cosine similarity
- Applications

Now

- Challenges
  - evaluation
  - word ambiguity
  - distinguishing semantic relationships
  - sparseness

Automatic Thesaurus generation

- Extract feature representations based on corpus co-occurrence frequencies
- Convert representations to PPMI
- Calculate cosine similarities for all pairs of words
  - computationally very expensive
  - may want to reduce the number of words considered in vocab
  - e.g., top 10,000 words
- Find nearest neighbours of each word

Evaluation

- Difficult – why?
- Intrinsic evaluation
  - human synonymy judgements
  - manually compiled thesauruses
- Extrinsic evaluation
  - performance gain in an application

Word ambiguity

bow	
ribbon	0.09
machete	0.07
spear	0.07
hull	0.07
sword	0.07
knife	0.07
arrow	0.06
scarf	0.06
rope	0.06
streamer	0.06

Here is the distributional thesaurus entry for the noun *bow* (derived using nltk.lin_thesaurus)

- What different senses of the word bow do you think are captured by the thesaurus entry?
- Are the neighbours distributed evenly between the senses or do some senses have more neighbours than others?
- Why do you think this is?

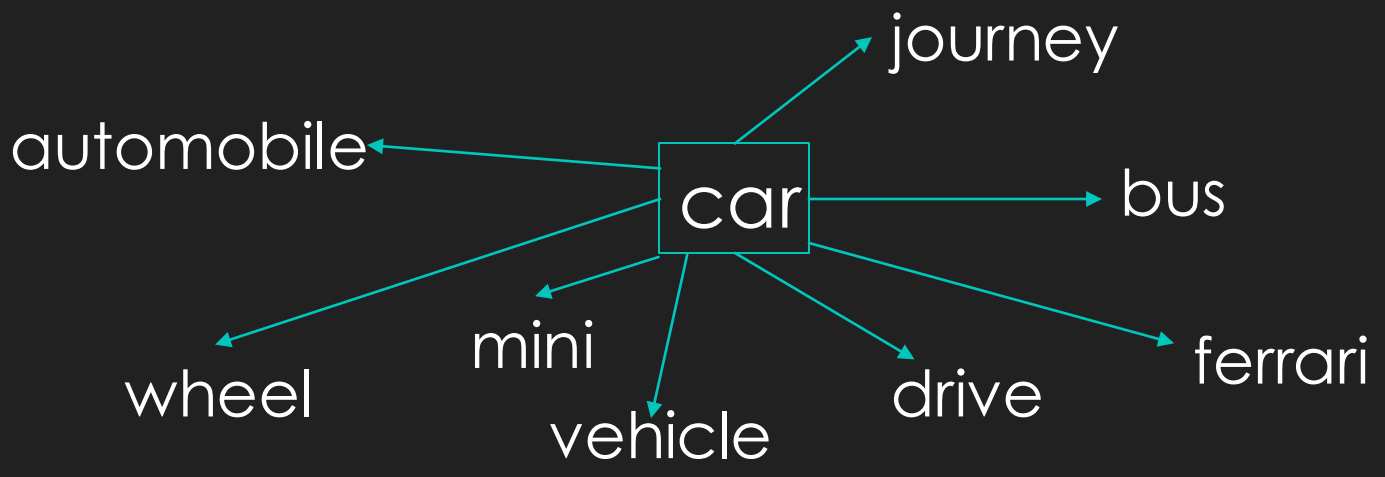
Senses in Distributional Semantics

- Distributional representations are of words not senses
  - mixture of senses in distributional neighbourhoods
  - this can be a problem in some applications. ….?

  - possible solutions: carry out WSD
    - before finding distributional neighbours
    - after finding distributional neighbours
- Distributional neighbours tend to reflect predominant sense of word
  - how could this be useful?

Semantic relationships

- Similar words are not necessarily synonyms
- Neighbourhoods typically contain:
  - synonyms, antonyms, hypernyms, hyponyms, co-hyponyms, meronyms, topically related words ….



The nearest neighbour of a word is often an antonym (or co-hyponym). Why might this be a problem?

Distinguishing relationships

neighbour	sim
run	0.15
game	0.15
season	0.14
play	0.14
quarter	0.13
kick	0.13
match	0.12
inning	0.12
minute	0.12
goal	0.11

- The relationships between a word and its neighbours are well-known to depend on the definition of context used.
- One of these sets of noun neighbours of *ball* uses dependency based contexts.
- The other defines a co-occurrence as being in a window of +-7 words.
- Which do you think is which?
- And why?

neighbour	sim
shot	0.17
stick	0.16
wheel	0.15
shell	0.15
piece	0.14
puck	0.14
bullet	0.14
barrel	0.14
ring	0.14
projectile	0.14

Synonyms, antonyms, hypernyms and co-hyponyms

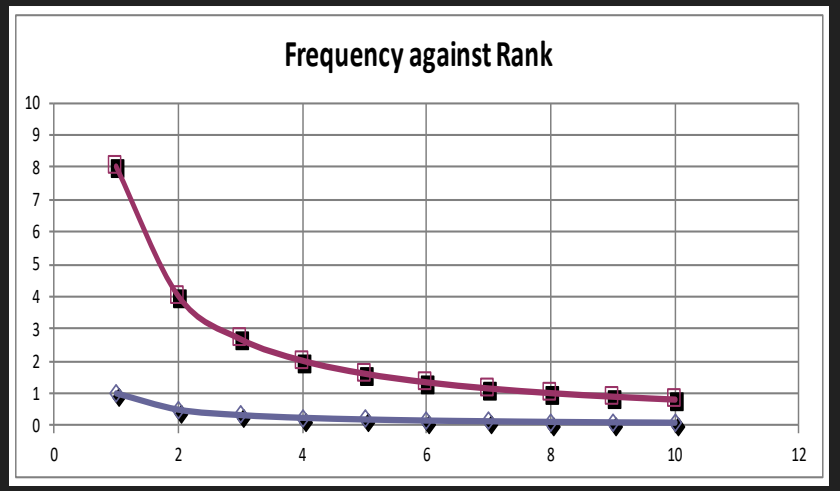
- Can be discovered by learning syntactic patterns for individual relations (Snow et al.2005) e.g., "X and other Y"

- Unsupervised hypernym detection often relies on some form of the distributional inclusion hypothesis (Weeds et al. 2004,Geffet and Dagan 2005) : *if w_2 is a hypernym of w_1, it follows that the contexts of w_2 will* ***include*** *the contexts of w_1*
- Supervised methods (e.g., Roller et al. 2014, Weeds et al. 2014, Shwartz et al. 2016) typically leverage lists of related words (e.g., derived from WordNet) and apply machine learning (e.g., SVM) to the concatenation of the distributional vectors.

Sparsity

- Remember Zipf's Law: "**The product of the frequency of a word and its rank is approximately constant.**"

Rank	1/Rank	Freq
1	8	8
2	4	4
3	2.667	3
4	2	2
5	1.6	2
6	1.333	1
7	1.143	1
8	1	1
9	0.889	1
10	0.8	1
	23.43	24



Hapax Legomena : words which only occur once. However large the corpus, these make up approximate half the vocabulary.

Consequences of Zipf's Law

- 82k dimensional co-occurrence vectors will be very sparse (lots of zeros)
- difficult to compare vectors because of all of this unseen stuff
- What can we do?

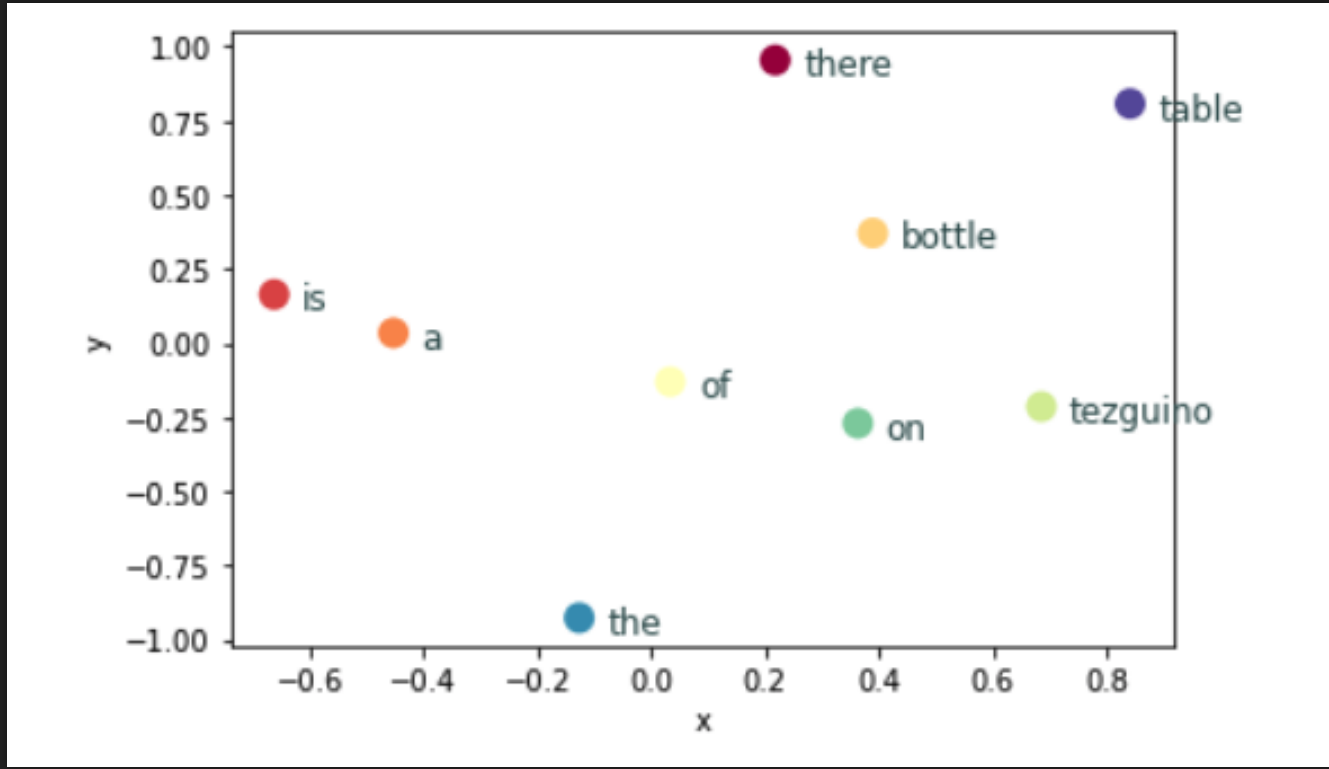
Possible solutions

- Smoothing
- Dimensionality reduction
- Language models with fixed dimensionality e.g., neural network language models (NNLMs)

Neural Network Language Models

- Start with the assumption that each word can be represented by a small fixed set of dimensions
- Learn the best values to optimise performance on some task e.g., the log-likelihood of some training corpus

Initialisation



- Points are randomly initialised in m-dimensional space
- typically m in range [50, 500]
- here m = 2

Prediction

- Use the current model to make a prediction about what word comes next / is masked in a sentence

there is a bottle of _____ on the table

CBOW model

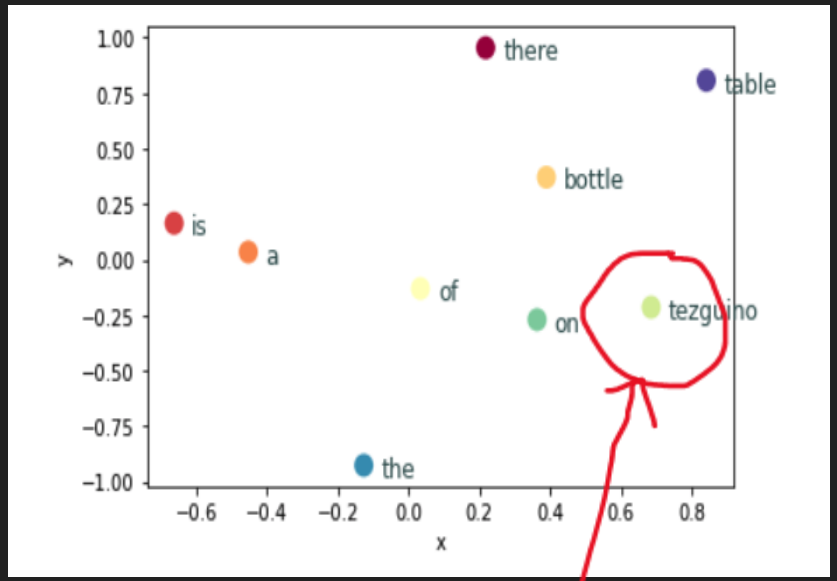
- continuous bag of words model (Word2Vec, Mikolov 2013)
- has two word spaces: one for target words and one for context words
- uses a window of k context words on either side of the target word to make the prediction

there is a bottle of __??___ on the table

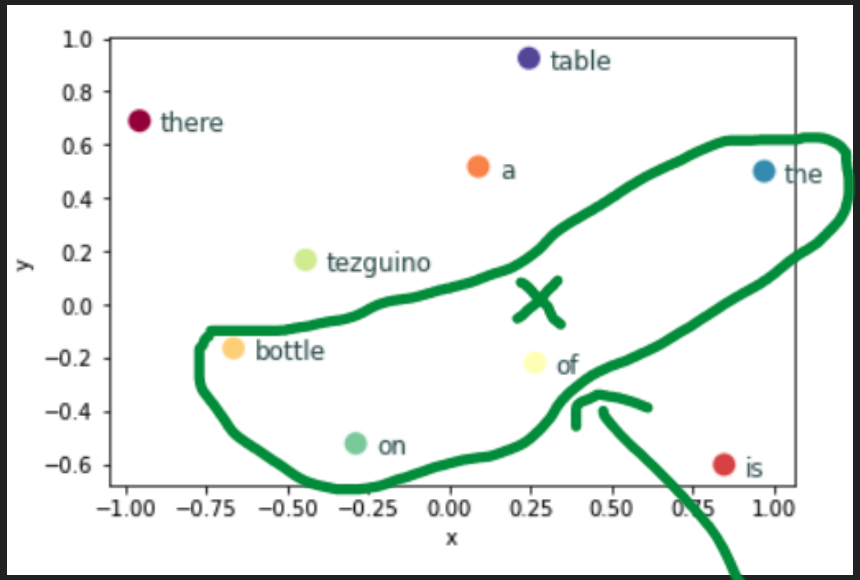
Update

- Find the centroid of the context vectors for these words together and predict that the target vector for the target word should be at this point
- Compare
  - compute the **loss:** the distance between the target vector and the centroid of the context vectors should be small
- Update the vectors to reduce the loss
- Repeat for more training examples

target word space



context word space



target word (t)

context words, centroid (c)

- loss will attempt to minimise distance between t and c
- calculates dot products which we know maximise similarity
- uses differentiation to make update in the "right" direction
- learn more in Advanced NLP

Making progress

- This week there is only a single notebook Lab_7_1.ipynb this week.

Bibliography

- Dagan, Pereira and Lee (ACL, 1994): Similarity-based estimation of word co-occurrence probabilities
- Geffet and Dagan (ACL, 2005): Improving Hypernymy Detection with an Integrated Path-Based and Distributional Model
- Levy and Goldberg (NIPS, 2014): Neural Word Embeddings as Implicit Matrix Factorization
- Levy, Goldberg and Dagan (TACL, 2015): Improving Distributional Similarity with Lessons Learned from Word Embeddings
- Landauer, Foltz and Laham (Discourse Processes, 1998): An Introduction to Latent Semantic Analysis
- Mikolov, Chen, Corrado and Dean (arXiv pre-print, 2013): Efficient Estimation of Word Representations in Vector Space
- Polajnar and Clark (EACL, 2014): Improving Distributional Semantic Vectors through Context Selection and Normalisation
- Snow, Jurafsky and Ng (NIPS, 2005): Learning syntactic patterns for automatic hypernym discovery
- Weeds, McCarthy and Weir (COLING, 2004): Characterising Measures of Lexical Distributional Similarity
- Weeds, Clarke, Reffin, Weir and Keller (COLING, 2014): Learning to Distinguish Hypernyms and Co-Hyponyms