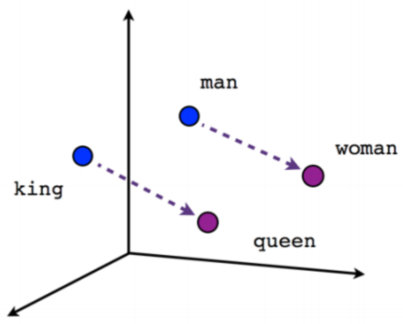
Vector Representation of Text

Word Embedding Technique (word2vec)

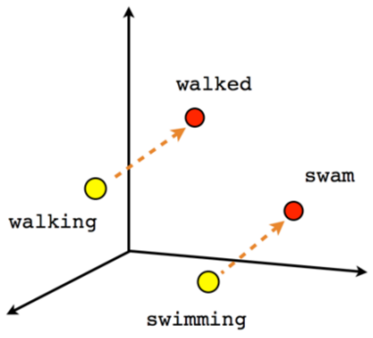
Word to vector (word2vector)

- The more often two words co-occur, the closer their vectors will be

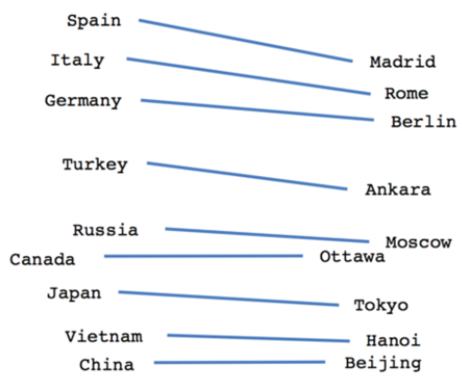
- Two words have close meanings if their local neighborhoods are similar



Male-Female

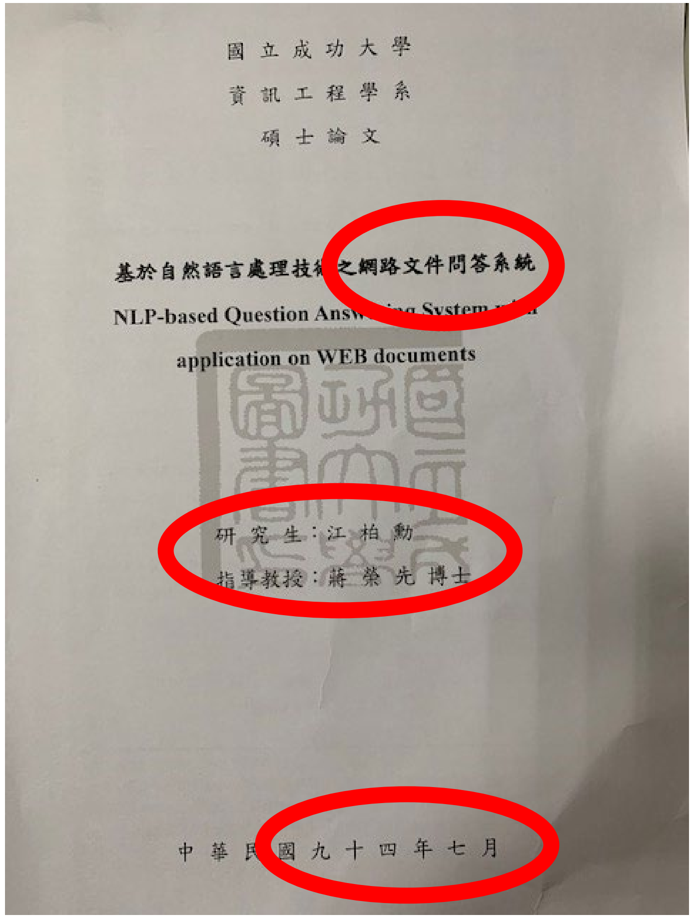


Verb tense



Country-Capital

我們IIR Lab的團隊



國 立 成 功 大 學

資 訊 工 程 學 系

碩 士 論 文

基於自然語言處理技術之網路文件問答系統

NLP-based Question Answering System with

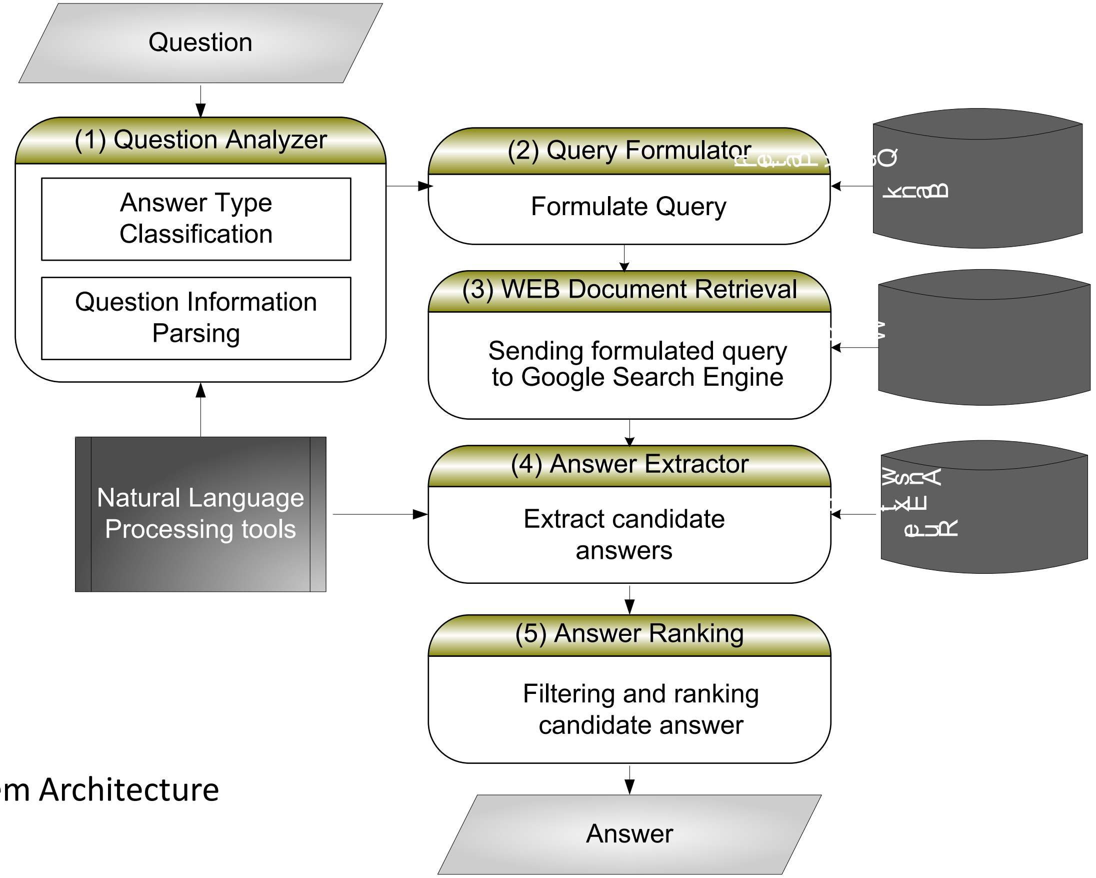
application on WEB documents

研 究 生：江 柏 勳

指導教授：蔣 榮 先 博士

中 華 民 國 九 十 四 年 七 月

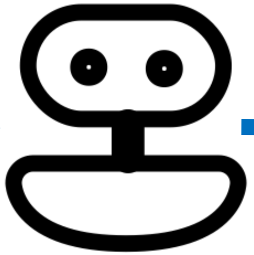
研究架構與做法



CKQA System Architecture

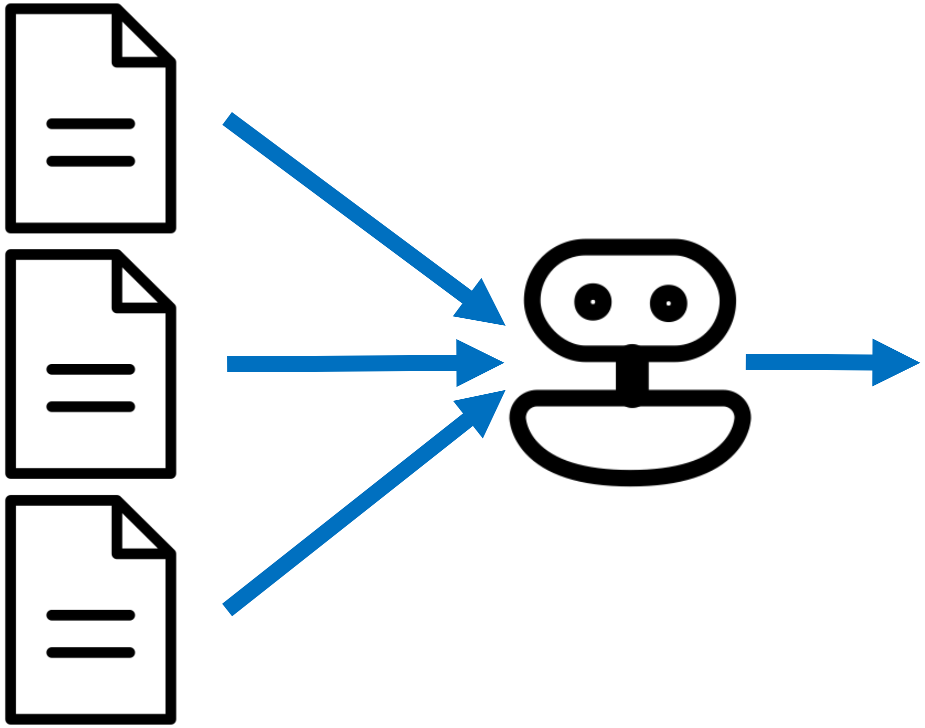
Bag of Words

I love dogs



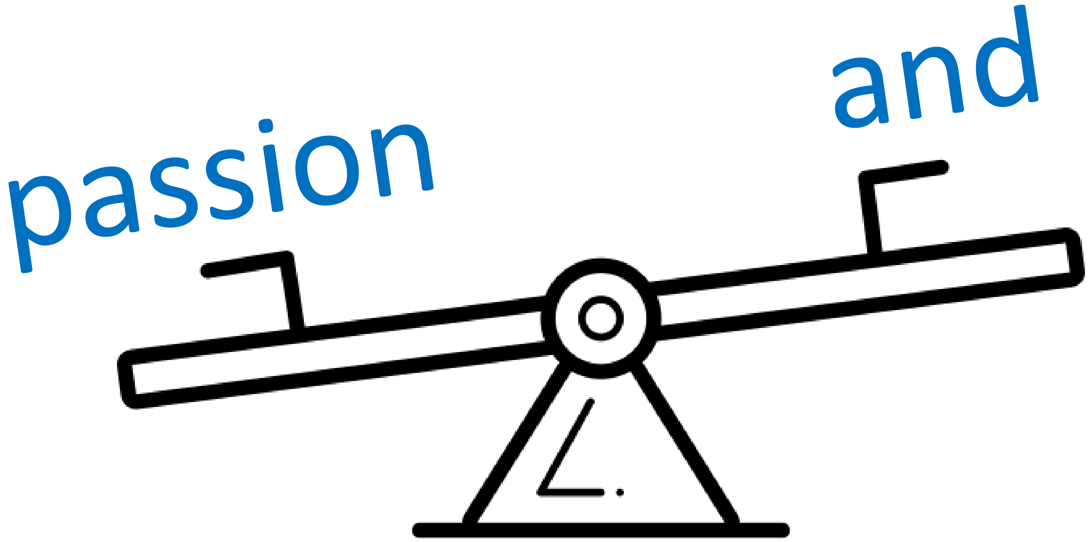
	I	Love	Dogs
Doc 1	1	1	1

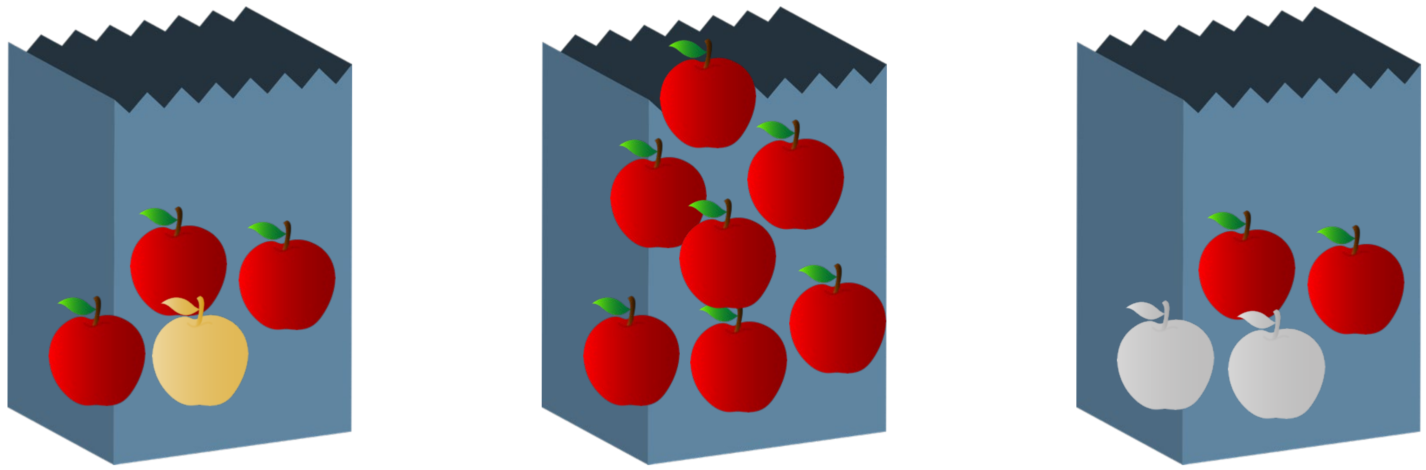
Bag of Words

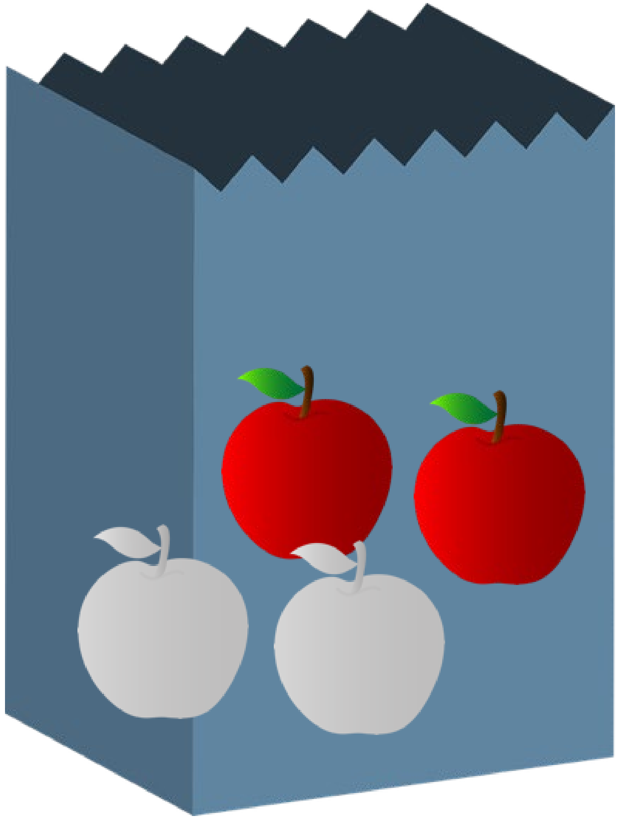


	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

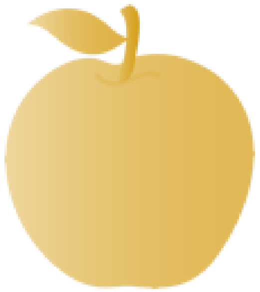
	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	5	1				
Doc 3					1	1	1	2	1	1


passion
and



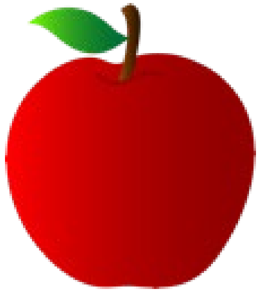


銀蘋果
對於這袋子
有多重要？

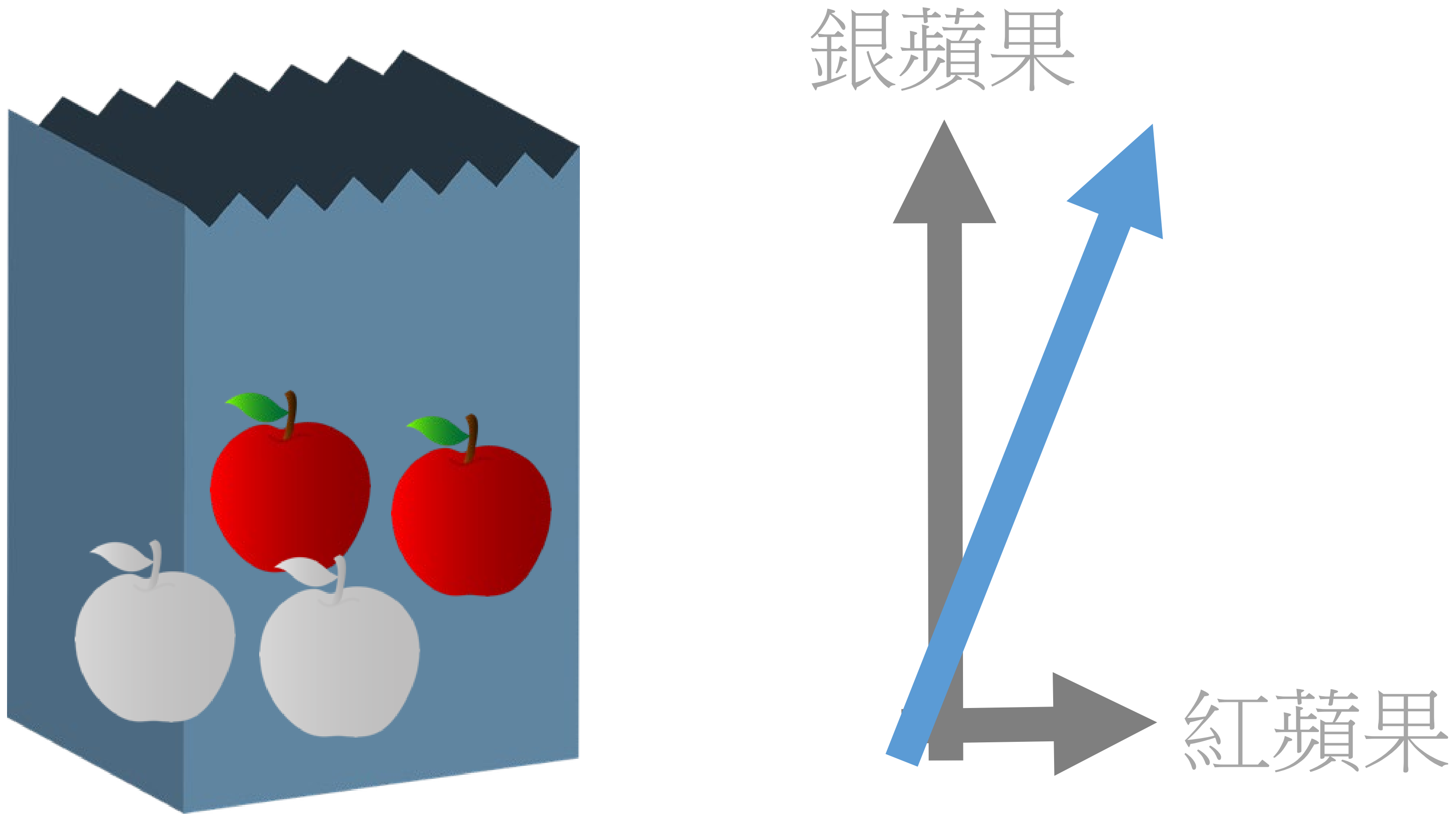


1/10000

1/1000



998.9/1000


銀蘋果
紅蘋果

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

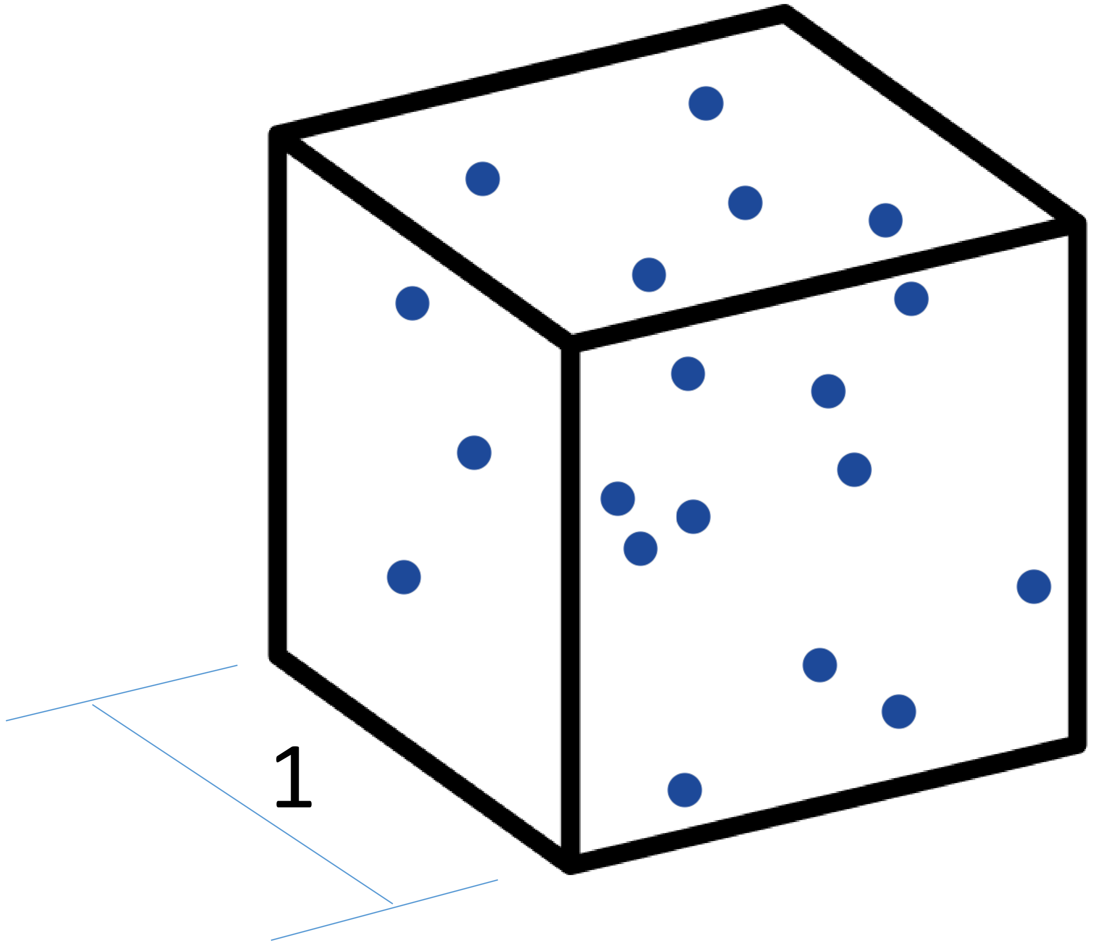
$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

Word embedding

keras.layers.Embedding

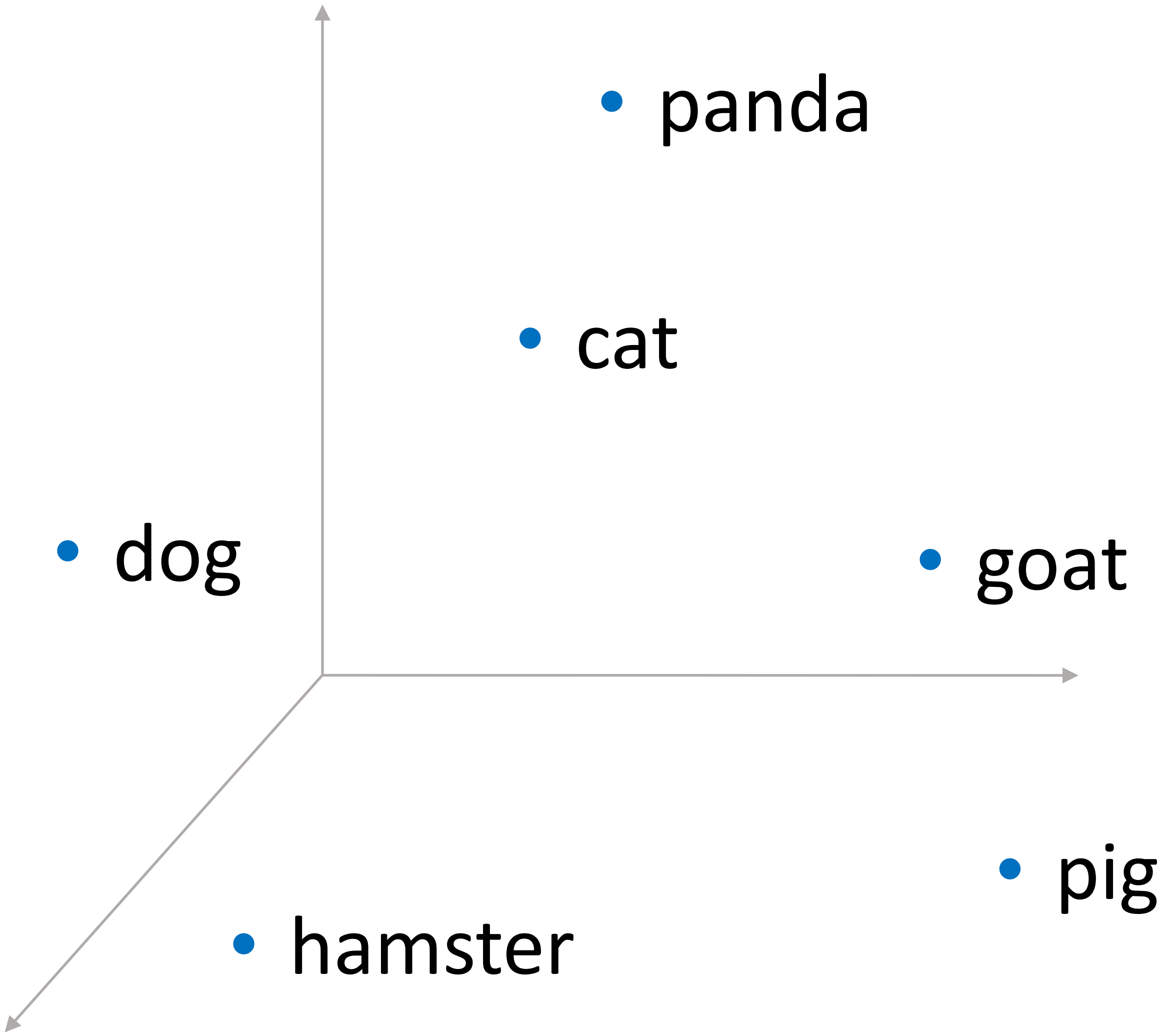


passion

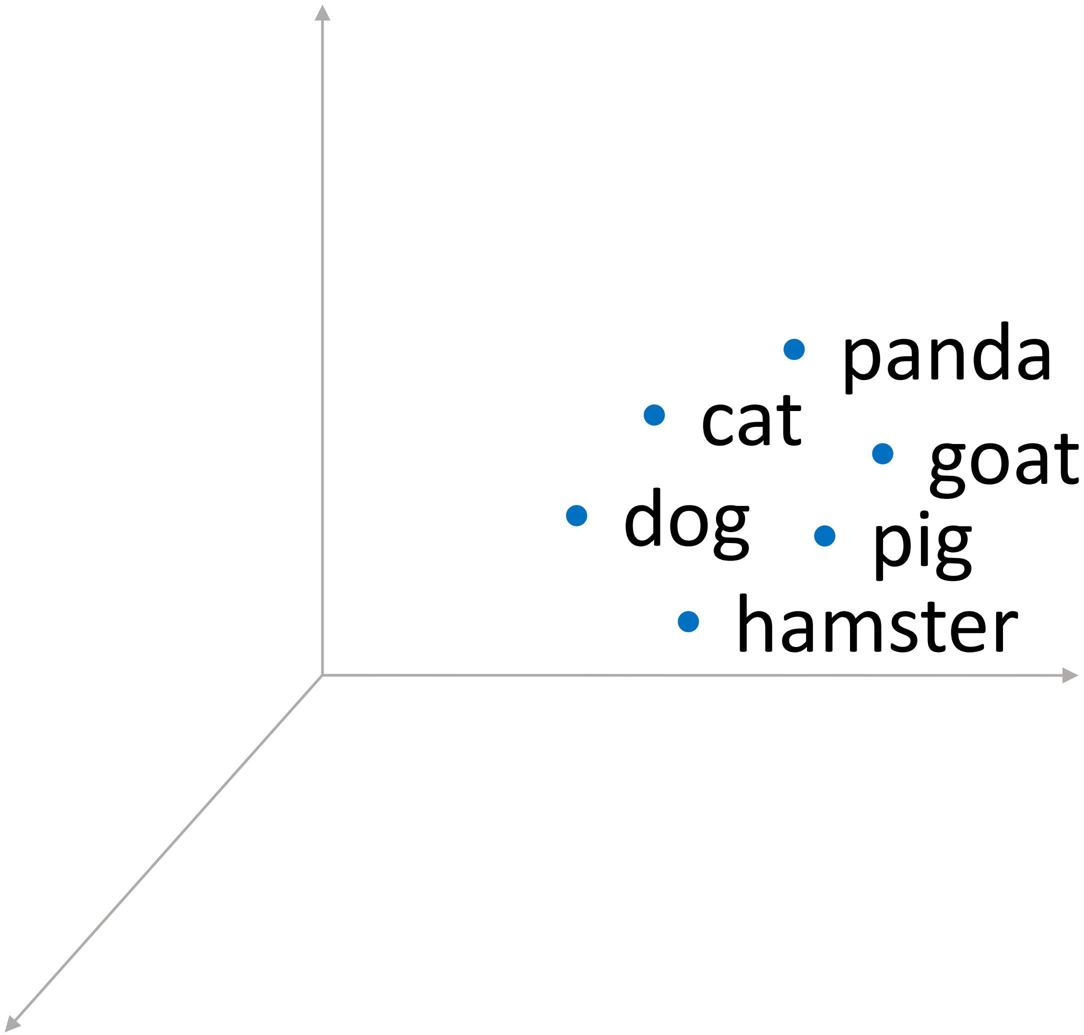
x: 0.119

y: 0.212

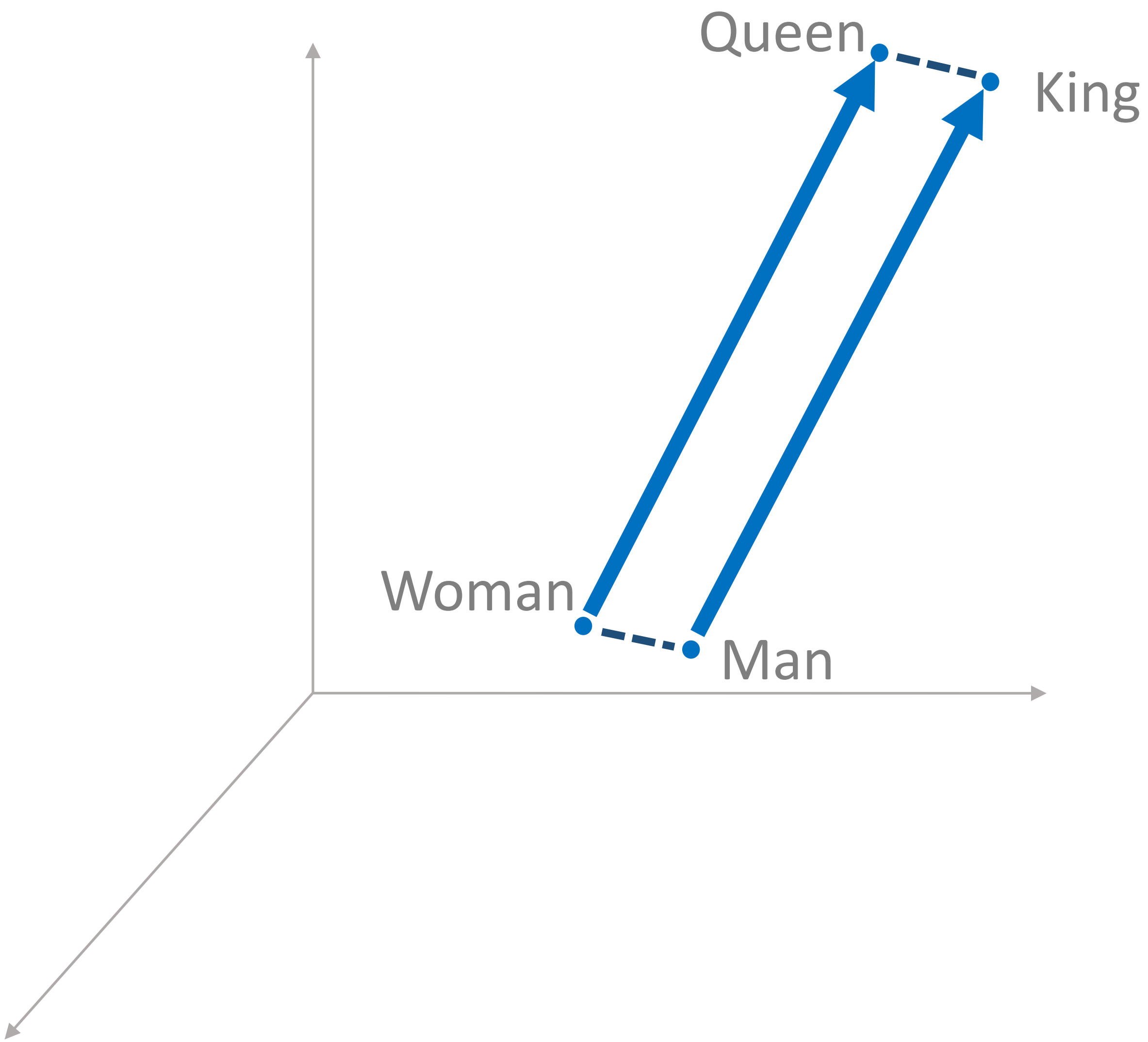
z: 0.010


panda
cat
dog
goat
pig
hamster

Word2Vec

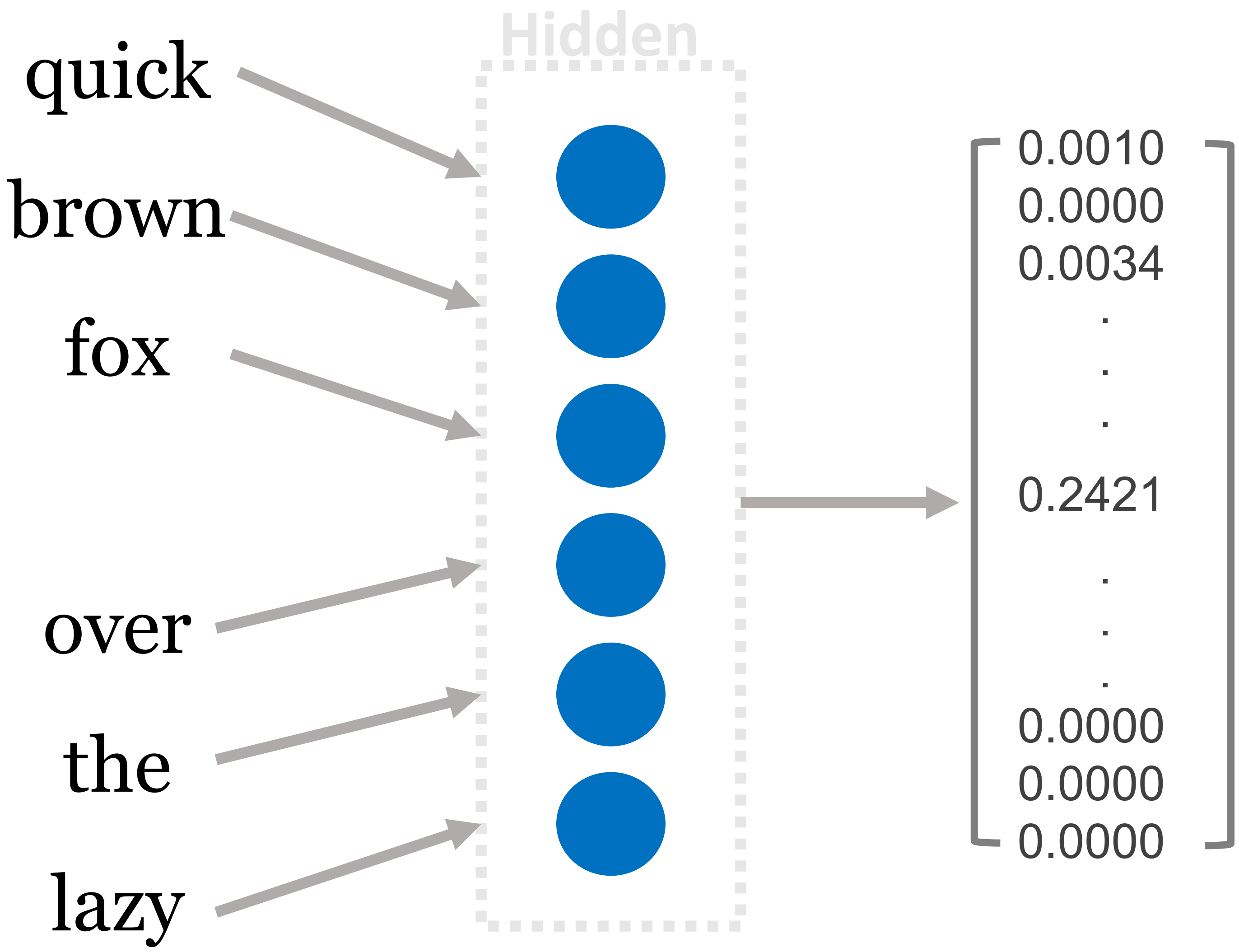


Word2Vec

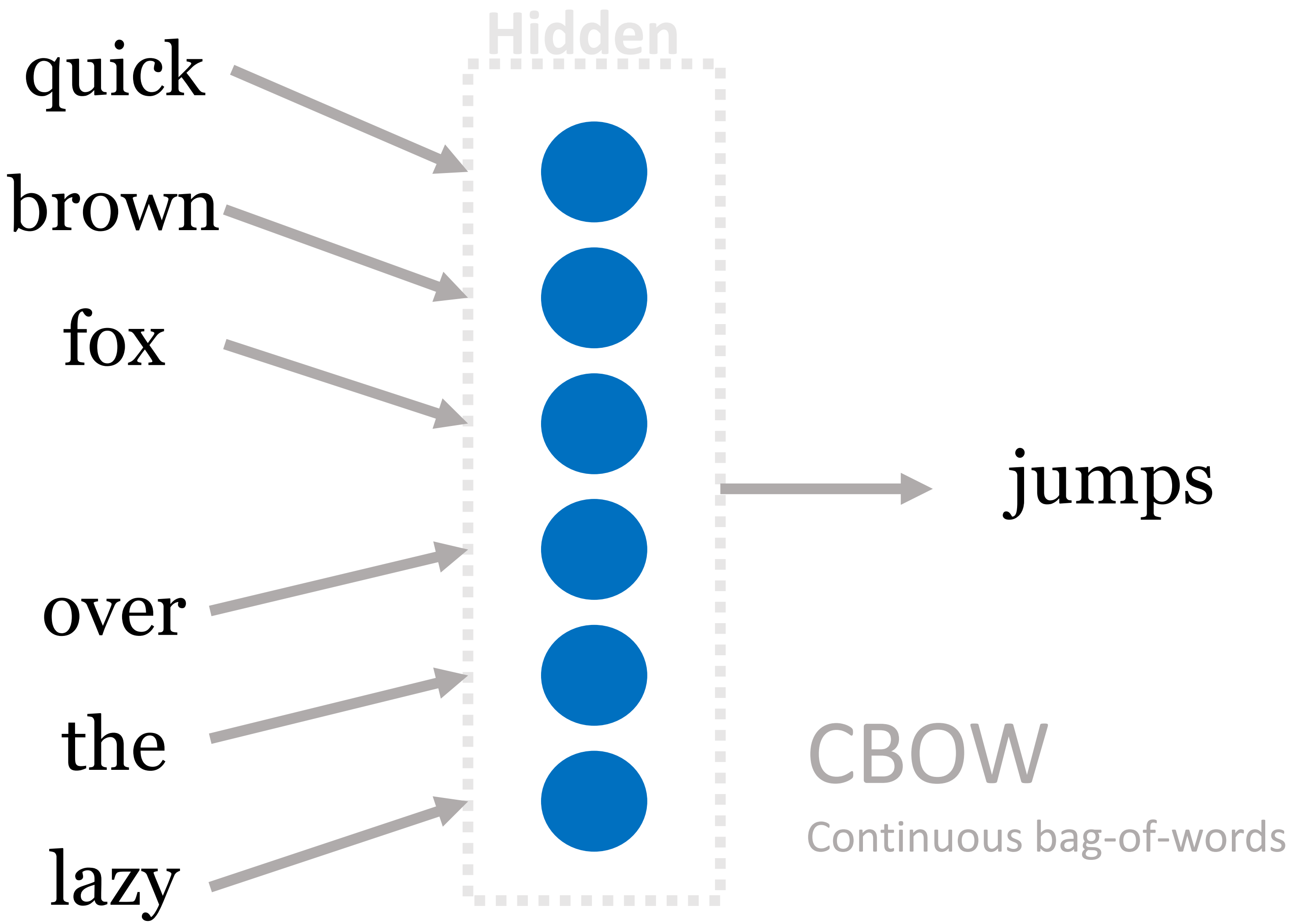


“ The quick brown fox ___?___ over the lazy dog ”

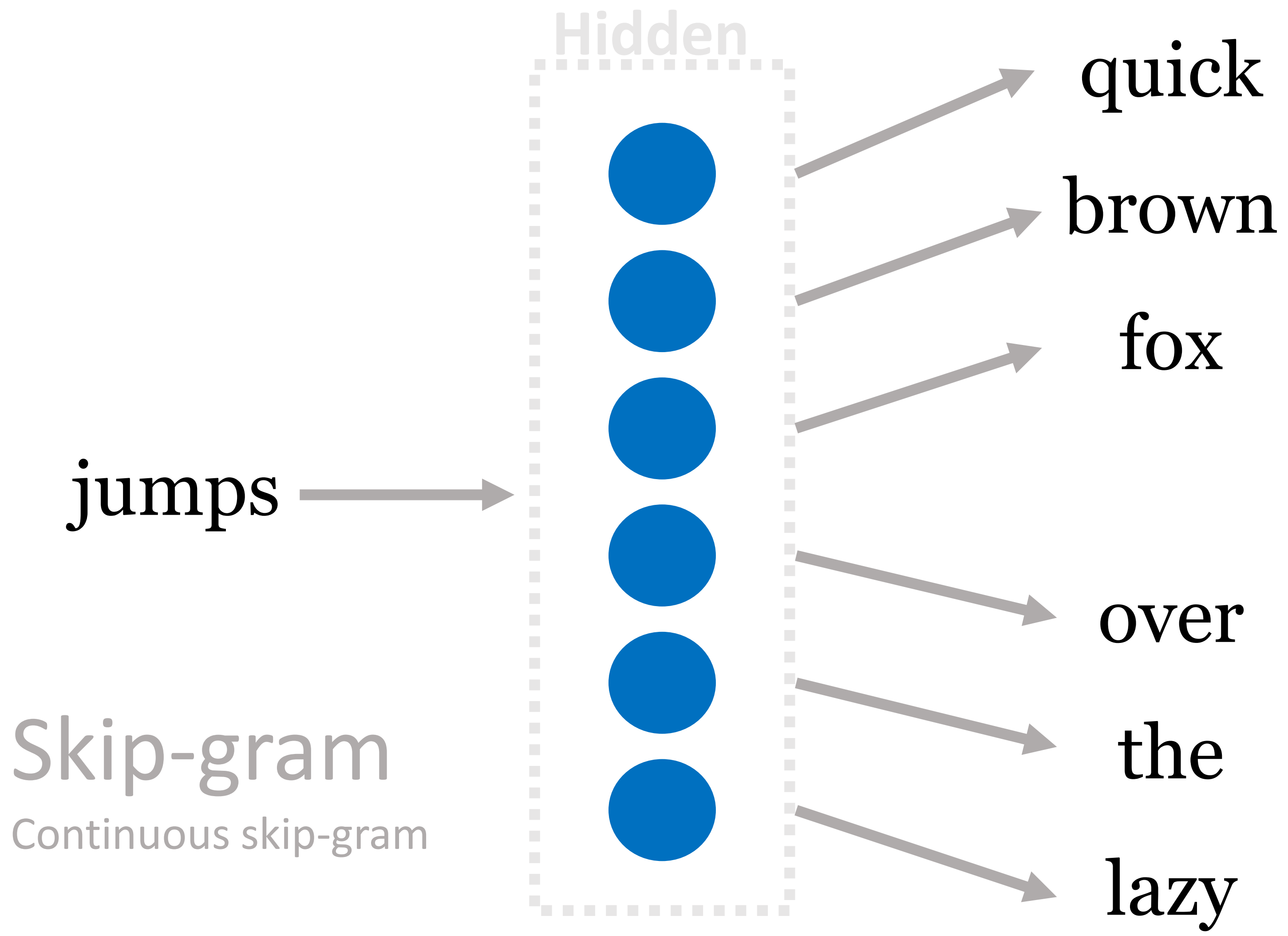
“ The quick brown fox _______ over the lazy dog ”



quick
brown
fox
over
the
lazy
Hidden
0.0010
0.0000
0.0034
.
.
.
0.2421
.
.
.
0.0000
0.0000
0.0000



Hidden
quick
brown
fox
over
the
lazy
jumps
CBOW
Continuous bag-of-words


Hidden
quick
brown
fox
jumps
over
the
lazy
Skip-gram
Continuous skip-gram

Dimension coordination

Probability list

$$\begin{bmatrix} 0.0010 \\ 0.0000 \\ 0.0034 \\ \vdots \\ 0.2421 \\ \vdots \\ 0.0000 \\ 0.0000 \\ 0.0000 \end{bmatrix}$$

“ The quick brown fox ______ over the lazy dog ”

grammatical, semantical similarity

Word Embedding Choices

1. Learnable embedding
2. Word2Vec
3. GloVe
4. FastText

Word2Vec

- Check genism
- Get the training data on the same page (text8)
- https://radimrehurek.com/gensim/models/word2vec.html

Text8

Large text compression benchmark

- First 10^9 bytes of the English Wikipedia dump on Mar. 3, 2006.
- Remove tags, digits, punctations
- Lower cases
- Leaving a-z, unrepeated spaces
- Truncate first 10^8 bytes
- About 1700 articles
- http://mattmahoney.net/dc/textdata.html

```
  enwik8        Step
-----------   ------------
100,000,000   Original size
 96,829,911   Discard all outside <text...> ... </text>
 96,604,864   Discard #REDIRECT text
 96,210,439   Discard XML tags (<text...> and </text>)
 95,287,203   URL-decode &lt; &gt; and &amp; to < > and &
 95,087,290   Remove <ref> ... </ref> (citations)
 93,645,338   Remove other XHTML tags
 91,399,021   Replace [http:... anchor text] with [anchor text]
 90,868,662   Replace [[Image:...|thumb|left/right|NNNpx|caption]] with caption
 90,770,617   Replace [[category:text|title]] with [[text]]
 88,385,654   Remove  [[language:link]]  (links to same page in other languages)
 85,443,983   Replace [[Wiki link|anchor text]] with [[anchor text]]
 83,420,173   Remove  {{...}}  (icons and special symbols)
 80,943,967   Remove  { ... }  (tables)
 77,732,609   Remove  [ and ]
 75,053,443   Replace &...; with space (URL-encoded chars)
 70,007,945   Convert to lower case, replace all sequences not in a-z,0-9 with a single space
 74,090,640   Spell digits, leaving a-z and unrepeated spaces
```

$$O = E \times T \times Q$$

O	*Training complexity*
E	*Training epochs*
T	*Word count in corpus*
Q	*<Model related factor>*

$$O = E \times T \times Q$$

O	*Training complexity*
E	*Training epochs*
T	*Word count in corpus*
Q	*<Model related factor>*

CBOW

Continuous bag-of-words

N	*Window size for input*
D	*Dimension of P layer*
V	*Size of vocabulary*

$$Q = N \times D + D \times log_2(V)$$

$$O = E \times T \times Q$$

O	*Training complexity*
E	*Training epochs*
T	*Word count in corpus*
Q	*<Model related factor>*

Skip-gram

Continuous skip-gram

C	*Maximum word distance*
D	*Dimension of P layer*
V	*Size of vocabulary*

$$Q = C \times (D + D \times log_2(V))$$

Word count

- Try Wikipedia Statistics page https://en.wikipedia.org/wiki/Wikipedia:Size_in_volumes

Vocabulary size

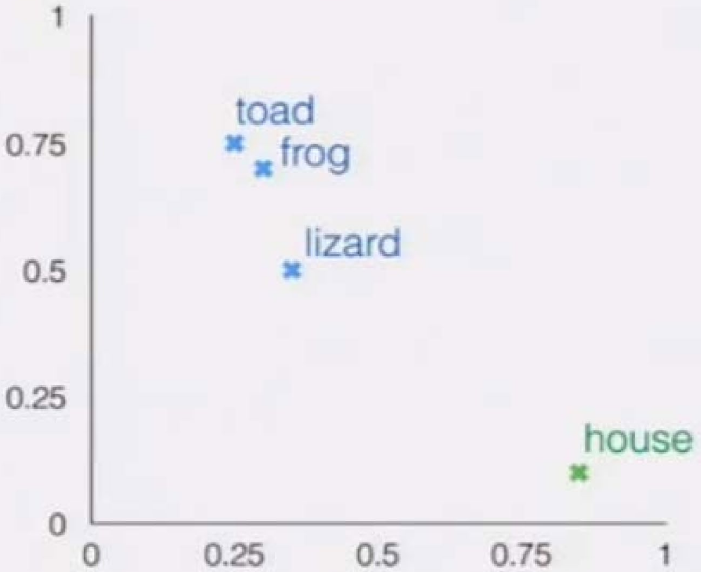
- Try Heaps' law
- https://en.wikipedia.org/wiki/Heaps%27_law

Problem?

Distributed representations

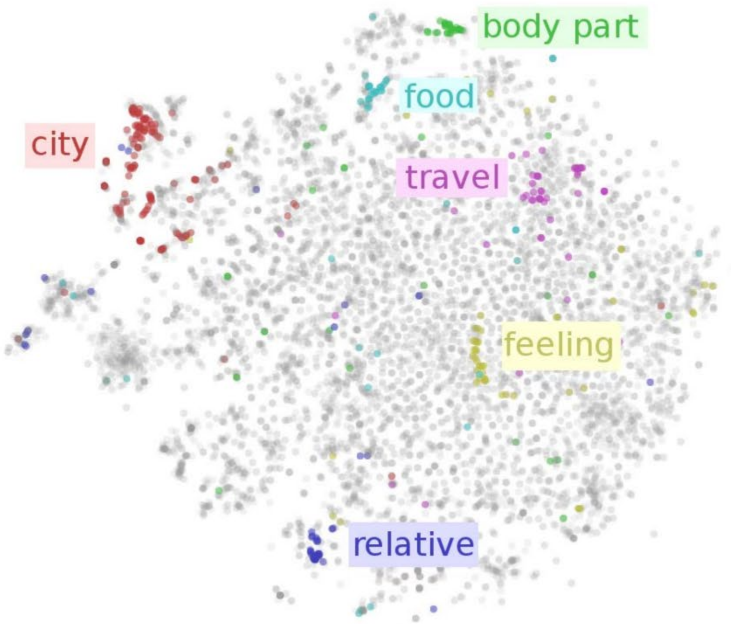
Word vectors aren't guaranteed to encode any linguistic relationships between words, but many models produce vectors that do

frog	[0.30 0.70]
toad	[0.25 0.75]
lizard	[0.35 0.50]
house	[0.85 0.10]



Example

Any technique mapping a word (or phrase) from it's original high-dimensional input space (the body of all words) to a lower-dimensional numerical vector space - so one *embeds* the word in a different space



Points: original word space

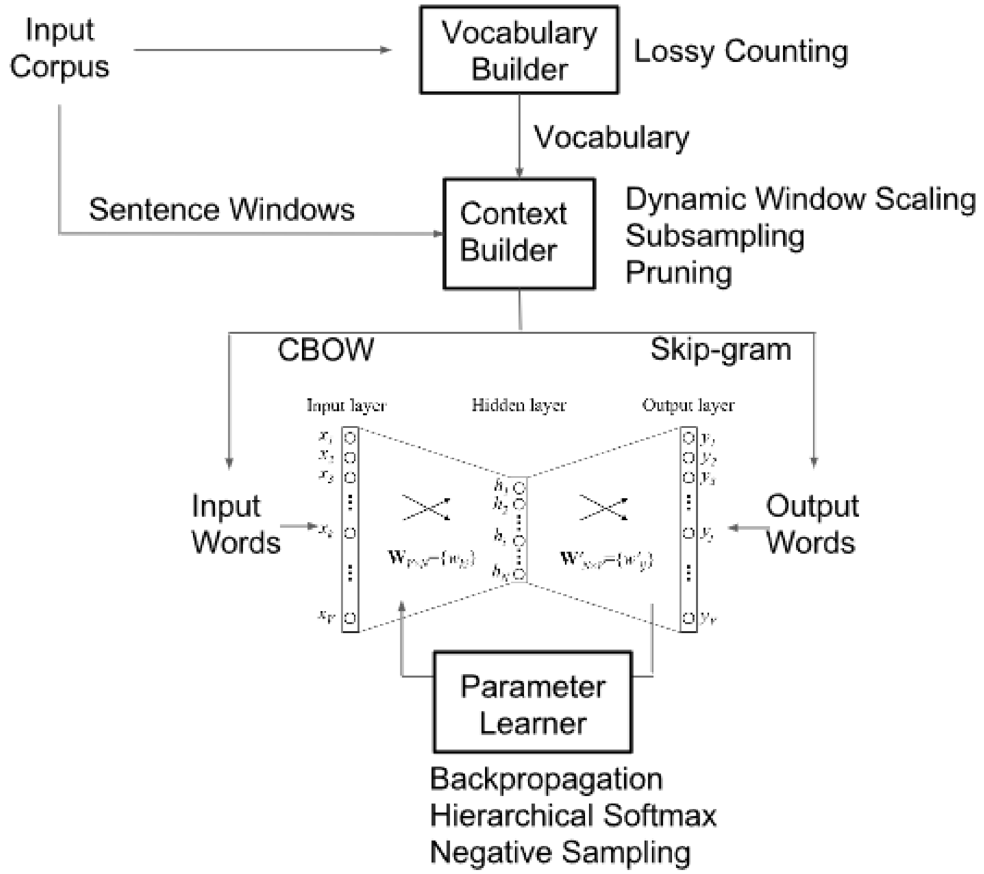
Colored points / clusters: Word embedding

Source: http://sebastianruder.com/content/images/2016/04/word_embeddings_colah.png

Word Representations

Traditional Method - Bag of Words Model	Word Embeddings
• 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 10000 words, and “Hello” is the 4th word in the dictionary, it would be represented by: 0 0 0 1 0 0 . . . . 0 0 0 0 • Context information is not utilized	• Stores each word in as a point in space, where it is represented by a vector of fixed number of dimensions (generally 300) • Unsupervised, built just by reading huge corpus • For example, “Hello” might be represented as : [0.4, -0.11, 0.55, 0.3 . . . 0.1, 0.02] • Dimensions are basically projections along different axes, more of a mathematical concept.

Architecture



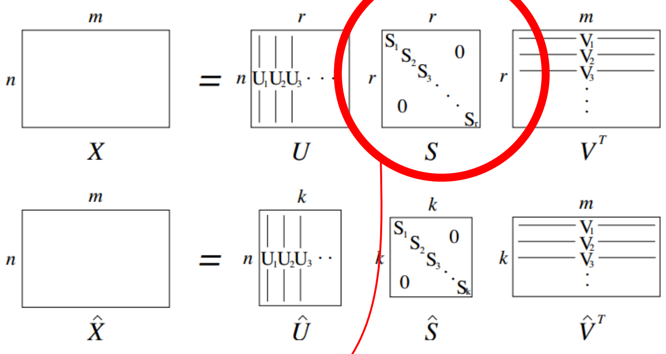
To compare pieces of text

- We need effective representation of
  - Words
  - Sentences
  - Text
- Approach 1: Use existing thesauri or ontologies like WordNet and Snomed CT (for medical).

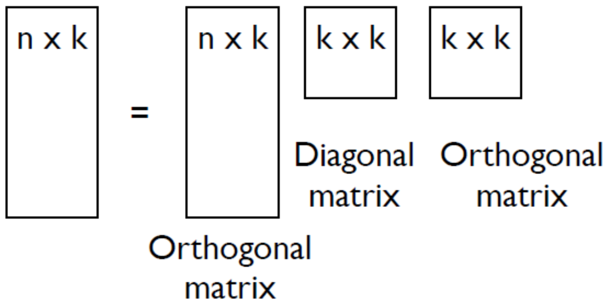
  Drawbacks:
  - Manual
  - Not context specific
- Approach 2: Use co-occurrences for word similarity. Drawbacks:
  - Quadratic space needed
  - Relative position and order of words not considered

Approach 3: low dimensional vectors

- Store only “important” information in fixed, low dimensional vector.
- Singular Value Decomposition (SVD) on co-occurrence matrix
  - $\hat{X}$ is the best rank *k* approximation to X , in terms of least squares
  - Motel = [0.286, 0.792, -0.177, -0.107, 0.109, -0.542, 0.349, 0.271]
- m = n = size of vocabulary
- $\hat{S}$ is the same matrix as S except that it contains only the top largest singular values

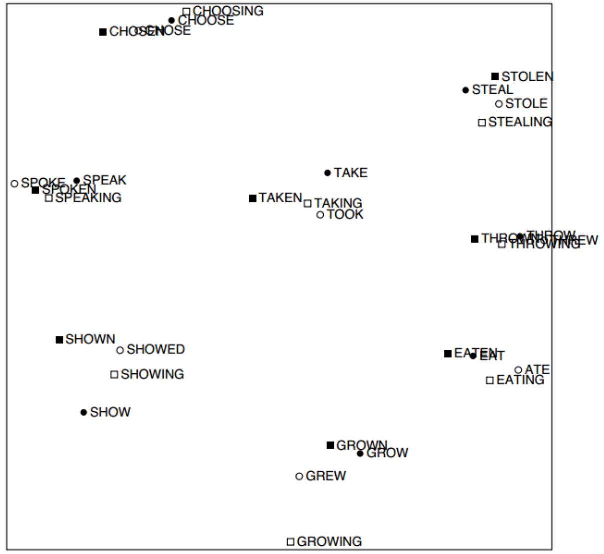


$$\mathbf{X} = \mathbf{U}\mathbf{D}\mathbf{V}^{\mathsf{T}}$$



Example of Approach 3: low dimensional vectors

- An Improved Model of Semantic Similarity Based on Lexical Co-Occurrence [Rohde et al. 2005]



Problems with SVD

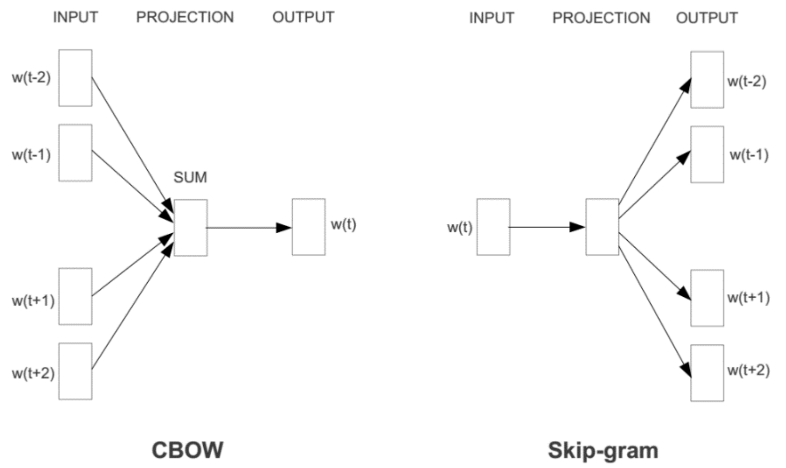
- Computational cost scales quadratically for n x m matrix: $O(mn^2)$ flops (when $n<m$)
- Hard to incorporate new words or documents
- Does not consider order of words

word2vec approach to represent the meaning of word

- Represent each word with a low-dimensional vector
- Word similarity = vector similarity
- Key idea: Predict surrounding words of every word
- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary

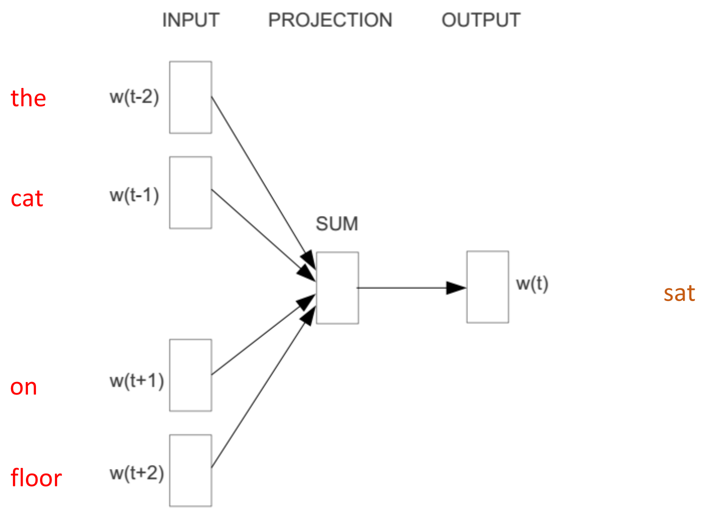
Represent the meaning of **word** – word2vec

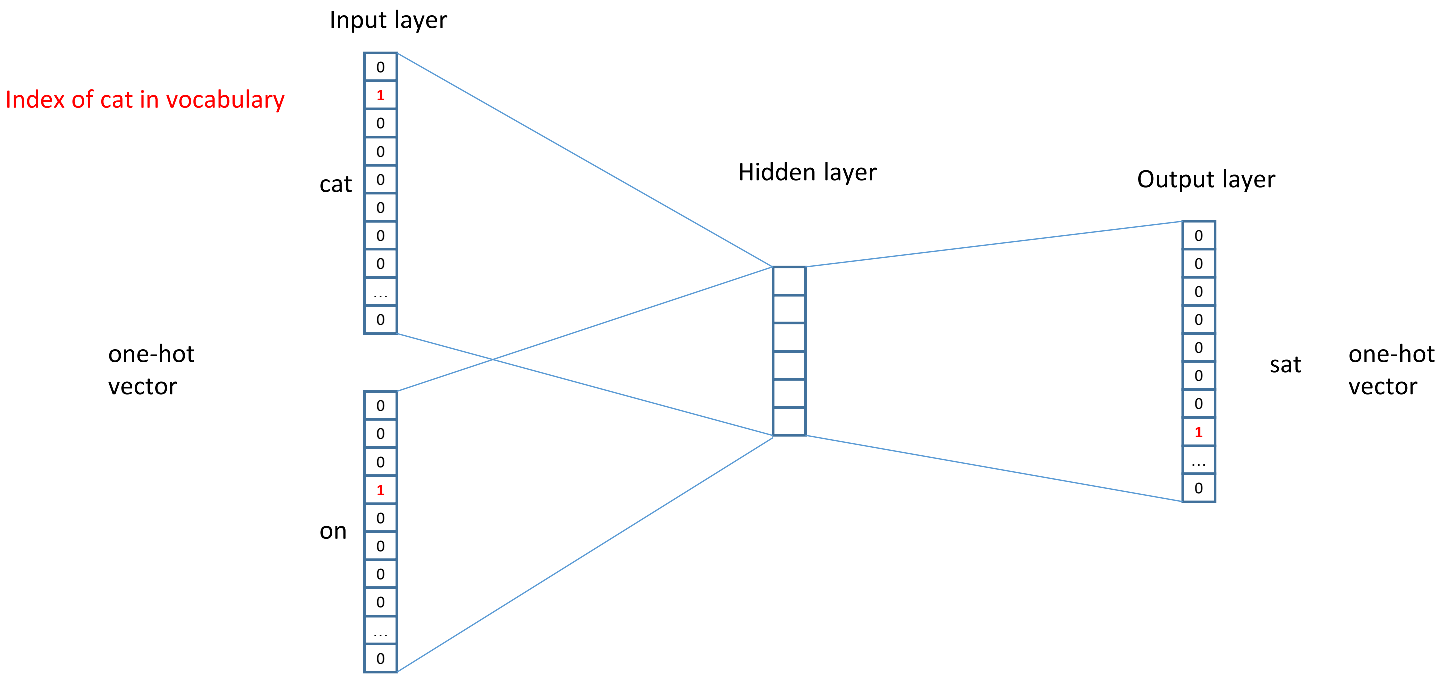
- 2 basic neural network models:
  - Continuous Bag of Word (CBOW): use a window of word to predict the middle word
  - Skip-gram (SG): use a word to predict the surrounding ones in window.



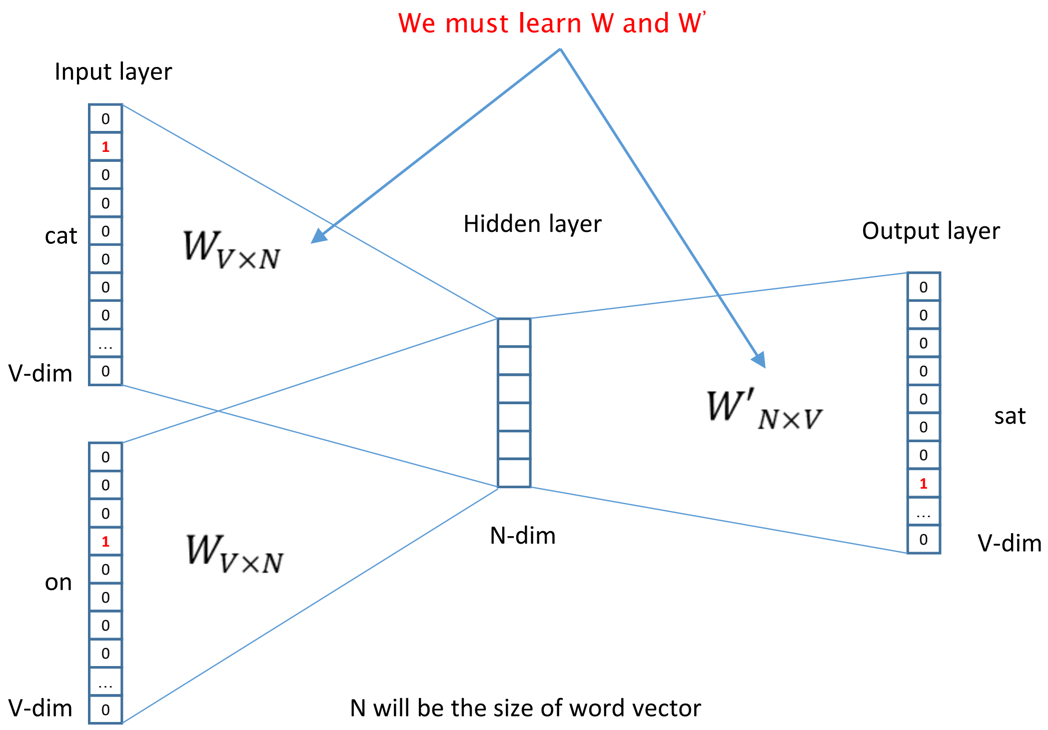
Word2vec – Continuous Bag of Word

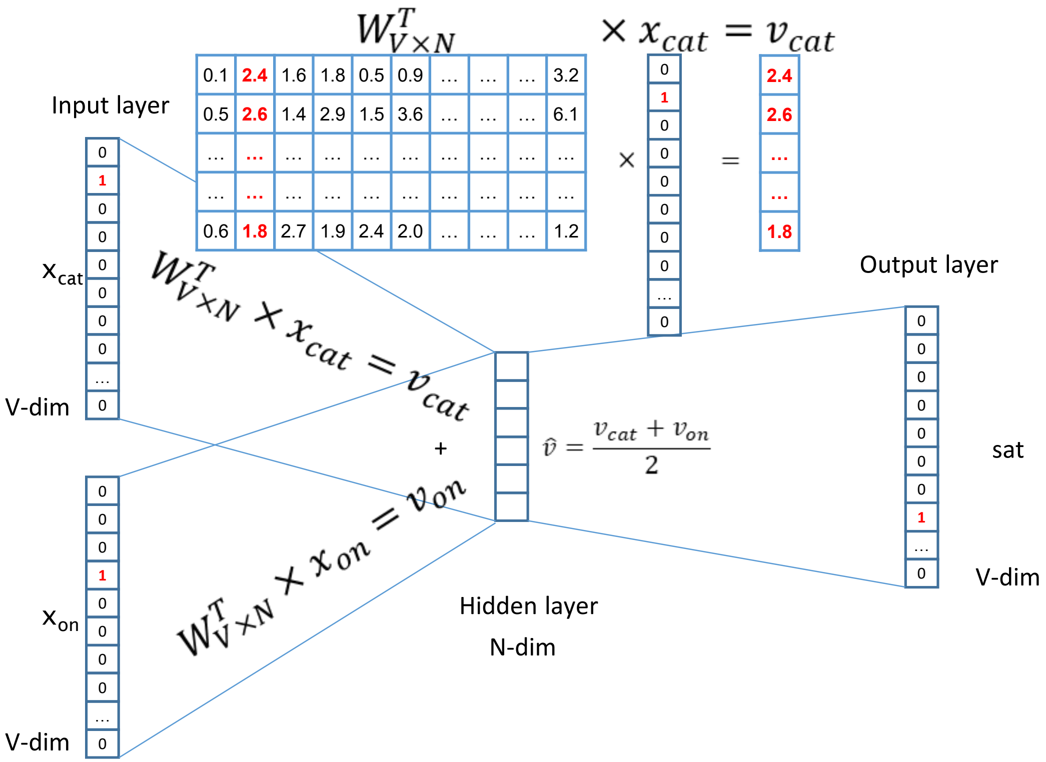
- E.g. “The cat sat on floor”
  - Window size = 2

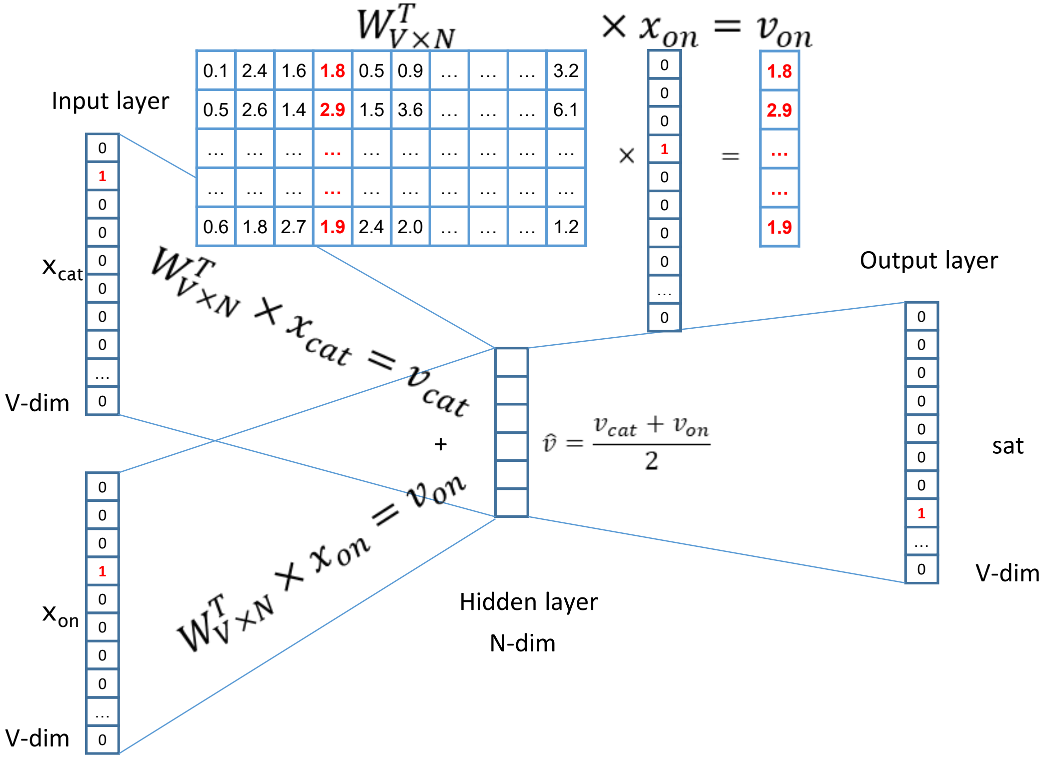




Input layer
Index of cat in vocabulary
0
1
0
0
cat
0
0
0
0
…
0
Hidden layer
Output layer
0
0
0
0
0
0
0
1
…
0
one-hot
vector
sat
one-hot
vector
0
0
0
1
on
0
0
0
0
…
0

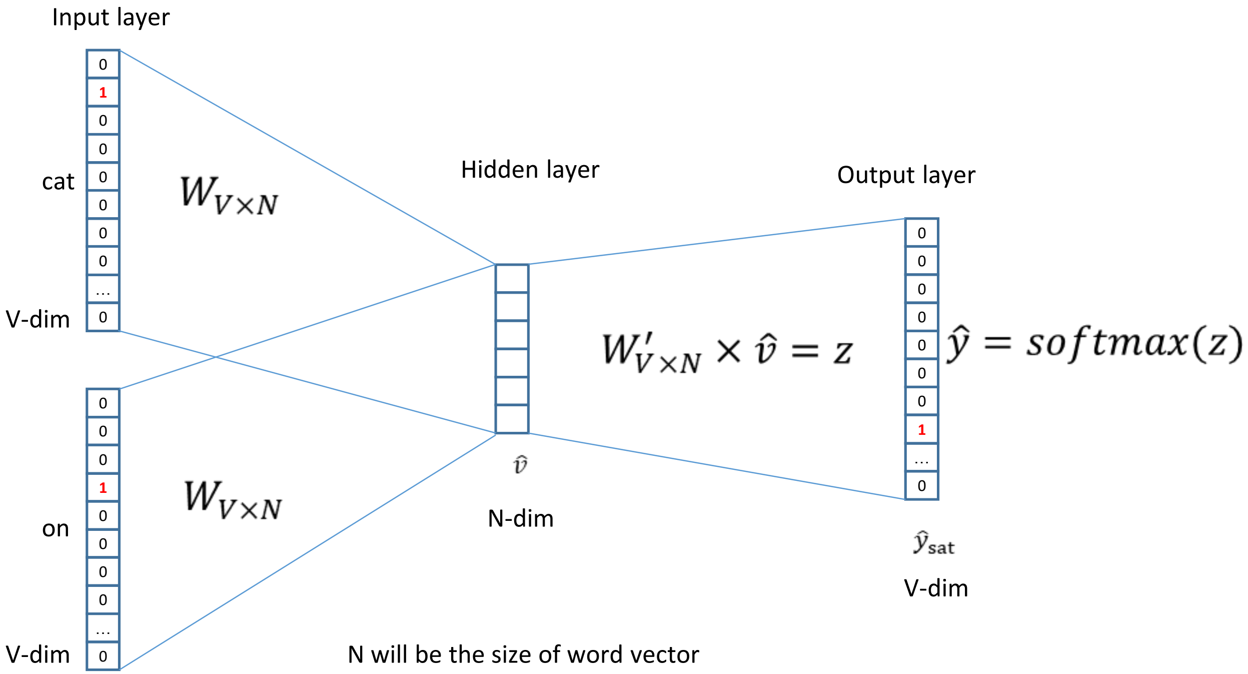

We must learn W and W'
Input layer
Hidden layer
Output layer
cat
V-dim
on
V-dim
$W_{V\times N}$
$W_{V\times N}$
$W'_{N\times V}$
N-dim
sat
V-dim
0
1
0
0
0
0
0
0
...
0
N will be the size of word vector


$W^T_{V\times N} \times x_{cat} = v_{cat}$
0.1 2.4 1.6 1.8 0.5 0.9 … … … 3.2
0.5 2.6 1.4 2.9 1.5 3.6 … … … 6.1
0.6 1.8 2.7 1.9 2.4 2.0 … … … 1.2
2.4
2.6
1.8
Input layer
x_{cat}
V-dim
x_{on}
V-dim
$W^T_{V\times N} \times x_{cat} = v_{cat}$
+
$W^T_{V\times N} \times x_{on} = v_{on}$
$\hat{v} = \frac{v_{cat} + v_{on}}{2}$
Hidden layer
N-dim
Output layer
sat
V-dim

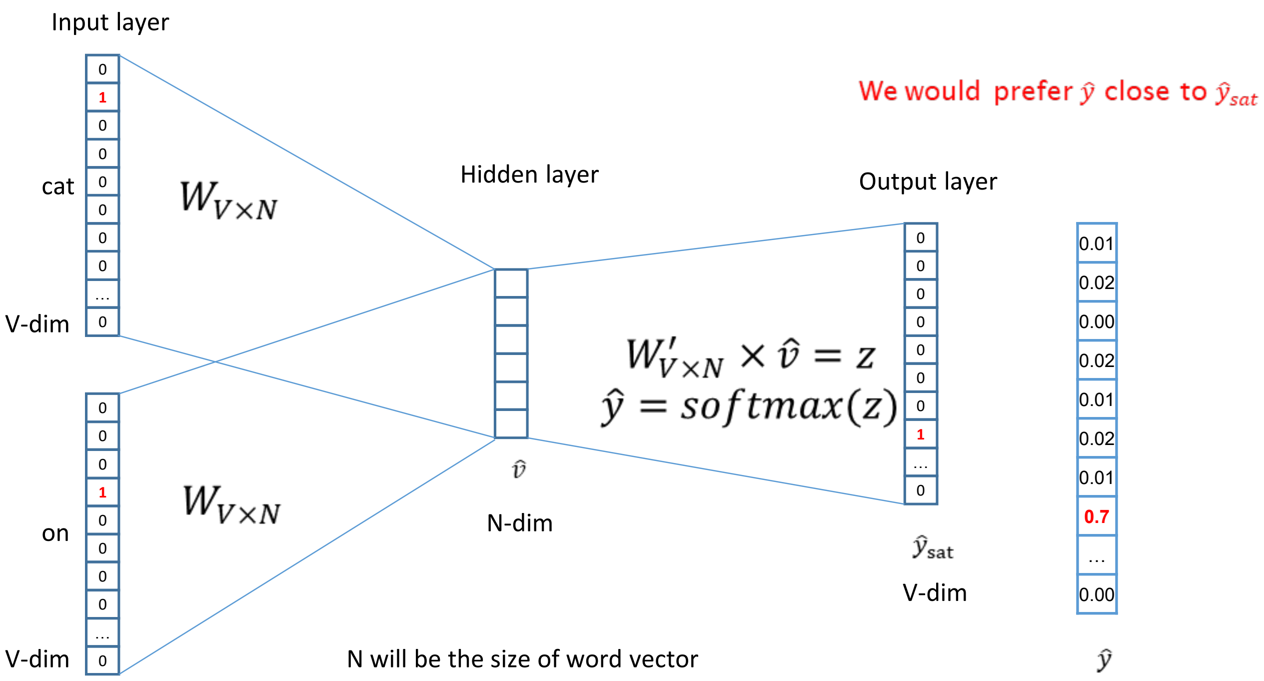


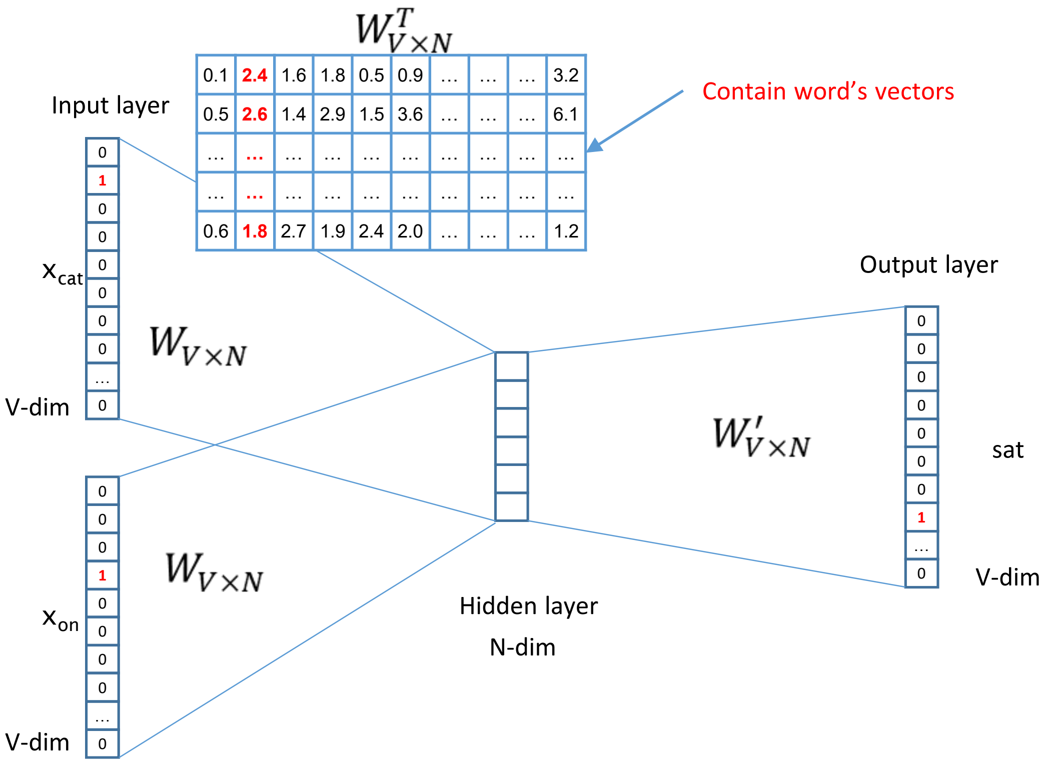
0.1	2.4	1.6	**1.8**	0.5	0.9	…	…	…	3.2
0.5	2.6	1.4	**2.9**	1.5	3.6	…	…	…	6.1
…	…	…	**…**	…	…	…	…	…	…
…	…	…	**…**	…	…	…	…	…	…
0.6	1.8	2.7	**1.9**	2.4	2.0	…	…	…	1.2

$\times$ [0, 0, 0, **1**, 0, 0, 0, 0, …, 0] $=$ [**1.8**, **2.9**, **…**, **…**, **1.9**]



N will be the size of word vector


We would prefer $\hat{y}$ close to $\hat{y}_{sat}$
Input layer
cat
V-dim
on
V-dim
$W_{V\times N}$
$W_{V\times N}$
Hidden layer
$\hat{v}$
N-dim
$W'_{V\times N} \times \hat{v} = z$
$\hat{y} = softmax(z)$
Output layer
$\hat{y}_{sat}$
V-dim
0.01
0.02
0.00
0.02
0.01
0.02
0.01
0.7
…
0.00
$\hat{y}$
N will be the size of word vector



We can consider either W or W' as the word's representation. Or even take the average.

Some interesting results

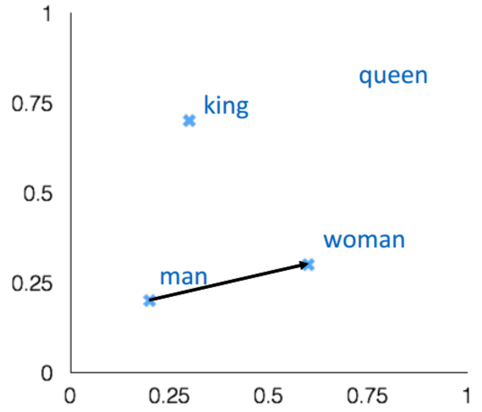
Word Analogies

Test for linear relationships, examined by Mikolov et al. (2014)

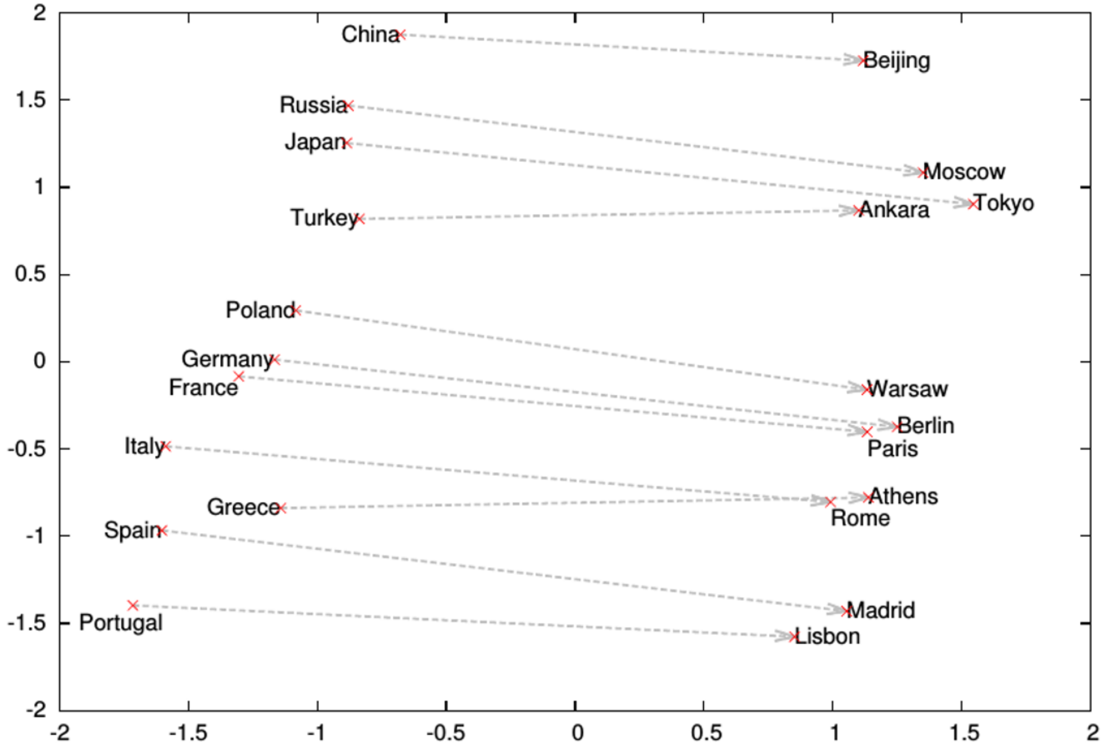
a:b :: c:? → $d = \arg\max_x \frac{(w_b - w_a + w_c)^T w_x}{||w_b - w_a + w_c||}$

man:woman :: king:?

+	king	[0.30 0.70]
-	man	[0.20 0.20]
+	woman	[0.60 0.30]
	queen	[0.70 0.80]

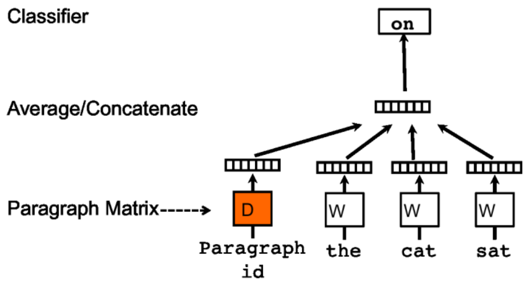


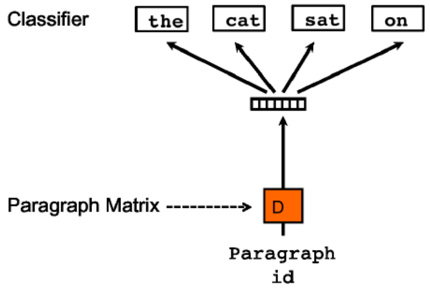
Word analogies



Represent the meaning of **sentence/text**

- Simple approach: take avg of the word2vecs of its words
- Another approach: Paragraph vector (2014, Quoc Le, Mikolov)
  - Extend word2vec to text level
  - Also two models: add paragraph vector as the input





Applications

- Word Similarity: Edit Distance, WordNet, Porter's Stemmer, Lemmatization using dictionaries
- Search, e.g., query expansion
- Machine Translation
- Part-of-Speech and Named Entity Recognition
- Relation extraction
- Sentiment analysis
- Semantic Analysis of Documents
- Clustering