Word Vectors

What is "king" - "man" + "woman"?

Recap

- Statistical based approaches treated words as atomic symbols:

Animal	Dog	Truck

- Or in vector space:

$$[\mathbf{0\ 0\ 0\ 0\ 0\ 1\ 0\ 0\ 0\ 0\ 0\ 0\ 0\ 0} \ldots]$$

- Also known as "one hot" representation

- Animal: [0 0 0 0 0 1 0 0 0 0 0 0 0 0 ...] = V_1
- Dog: [0 0 0 0 0 0 0 0 0 1 0 0 0 0 ...] = V_2
- Truck: [0 0 0 0 0 0 0 0 0 0 0 1 0 0 ...] = V_3

$$\boldsymbol{V_1.V_2 = V_1.V_3 = V_2.V_3 = 0}$$
$$\boldsymbol{Dist}(\textbf{Animal, Dog}) = \sqrt{\mathbf{2}}$$
$$\boldsymbol{Dist}(\textbf{Animal, Truck}) = \sqrt{\mathbf{2}}$$
$$\boldsymbol{Dist}(\textbf{Truck, Dog}) = \sqrt{\mathbf{2}}$$

Distributional Representation

"You shall know a word by the company it keeps" – John Rupert Firth

- One of the most successful ideas of modern NLP
- Linguistic units with similar distributions have similar meaning

- A bank is a **financial** institution that accepts **deposits** from the public and creates **credit.**
- The northern bank of the **river** is flooded.

Co-Occurrence Matrix

- A co-occurrence matrix is a terms $\times$ terms matrix which captures the number of times a term appears in the context of another term

- The context is defined as a window of k words around the terms

Text Corpus

- "ChatGPT is a language model."
- "Language models like ChatGPT are powerful."
- "The language model ChatGPT is based on GPT-3.5 architecture."

- Vocabulary: “ChatGPT”, “language”, “model”, “is”, “a”, “like”, “are”, “powerful”, “The”, “based”, “on”, “GPT-3.5”, “architecture”

Co-Occurrence Matrix

1. "ChatGPT is a language model.“
2. "Language models like ChatGPT are powerful.“
3. "The language model ChatGPT is based on GPT-3.5 architecture."

✓ **window size = 1**

	ChatGPT	language	model	like	powerful	based	GPT-3.5	architecture
ChatGPT	0	1	1	1	0	0	0	0
language	1	0	1	0	0	0	0	0
model	1	1	0	1	0	0	0	0
like	1	0	1	0	0	0	0	0
powerful	1	0	0	0	0	0	0	0
based	1	0	0	0	0	0	1	0
GPT-3.5	0	0	0	0	0	1	0	1
architecture	0	0	0	0	0	0	1	0

Distributed Representations

- Compact, dense, low-dimensional, and real-valued representations
- Also known as word embedding, each single component of the vector representation does not have any meaning of its own
- The interpretable features are hidden and distributed among uninterpretable vector components

$$\boldsymbol{animal} = \begin{pmatrix} \mathbf{0.286} \\ \mathbf{0.792} \\ \mathbf{-0.112} \\ \mathbf{-0.143} \\ \mathbf{0.341} \\ \mathbf{0.512} \\ \mathbf{\ldots} \end{pmatrix}$$

Word Embedding

- They are numerical representations of words in a continuous vector space, where each word is mapped to a dense vector of real numbers
- These vectors capture the semantic and contextual meaning of words based on their usage in a large corpus of text.
- They are trained using unsupervised learning techniques on large text corpora and words are represented based on their surrounding context, which means that words are defined by the words they frequently appear with
- Results in dense vectors with real-number values capturing nuanced relationships between words
- There are pre-trained word embeddings available that have been trained on large text corpora. Examples include Word2Vec, GloVe, FastText, and BERT.

Word Embeddings

- Word Embedding for the word “king” (GloVe vector trained on Wikipedia):

```
[ 0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 ,
0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 , -0.034189, -0.98173 , 0.68229
, 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 ,
-0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505
, 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 ,
-0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 , -1.0137 , -0.21585 , -0.15155 ,
0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042 ]
```

List of 50 numbers

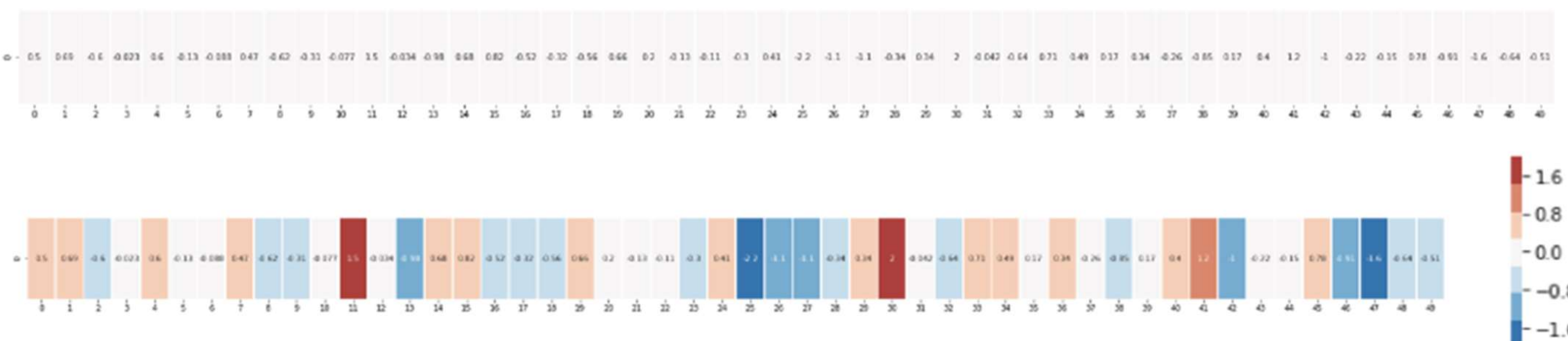


Image Reference: https://jalammar.github.io/illustrated-word2vec/

Word Embeddings

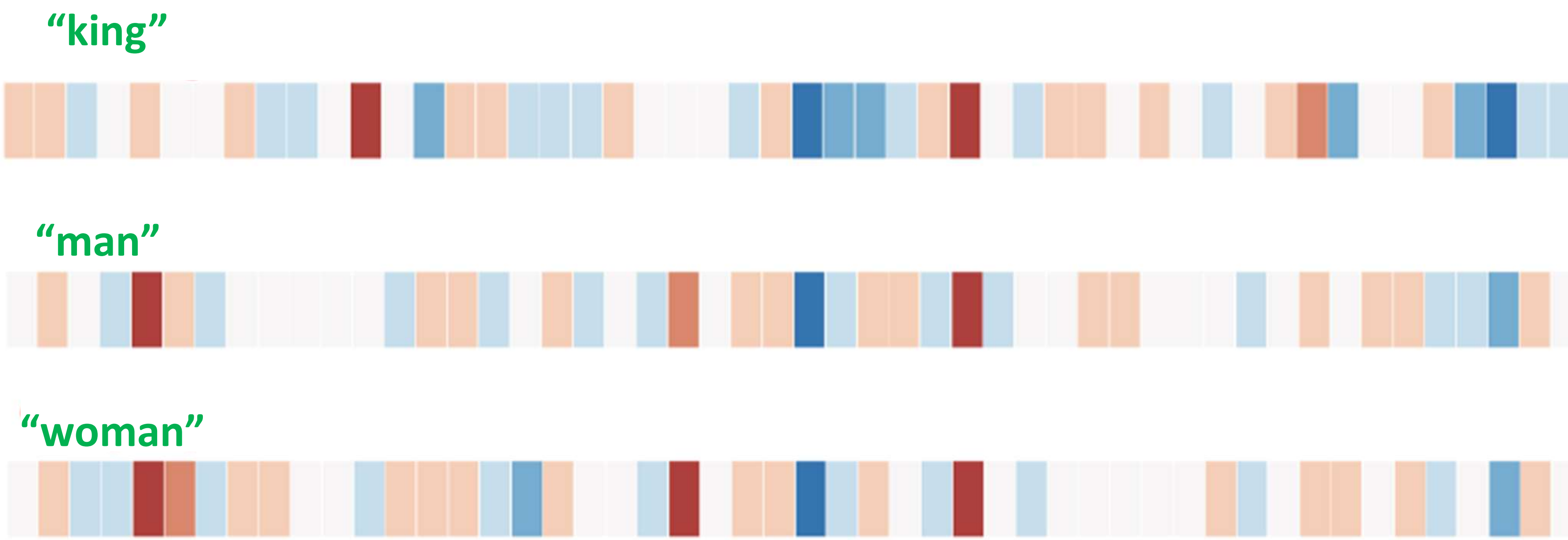


Image Reference: https://jalammar.github.io/illustrated-word2vec/

Word Embeddings

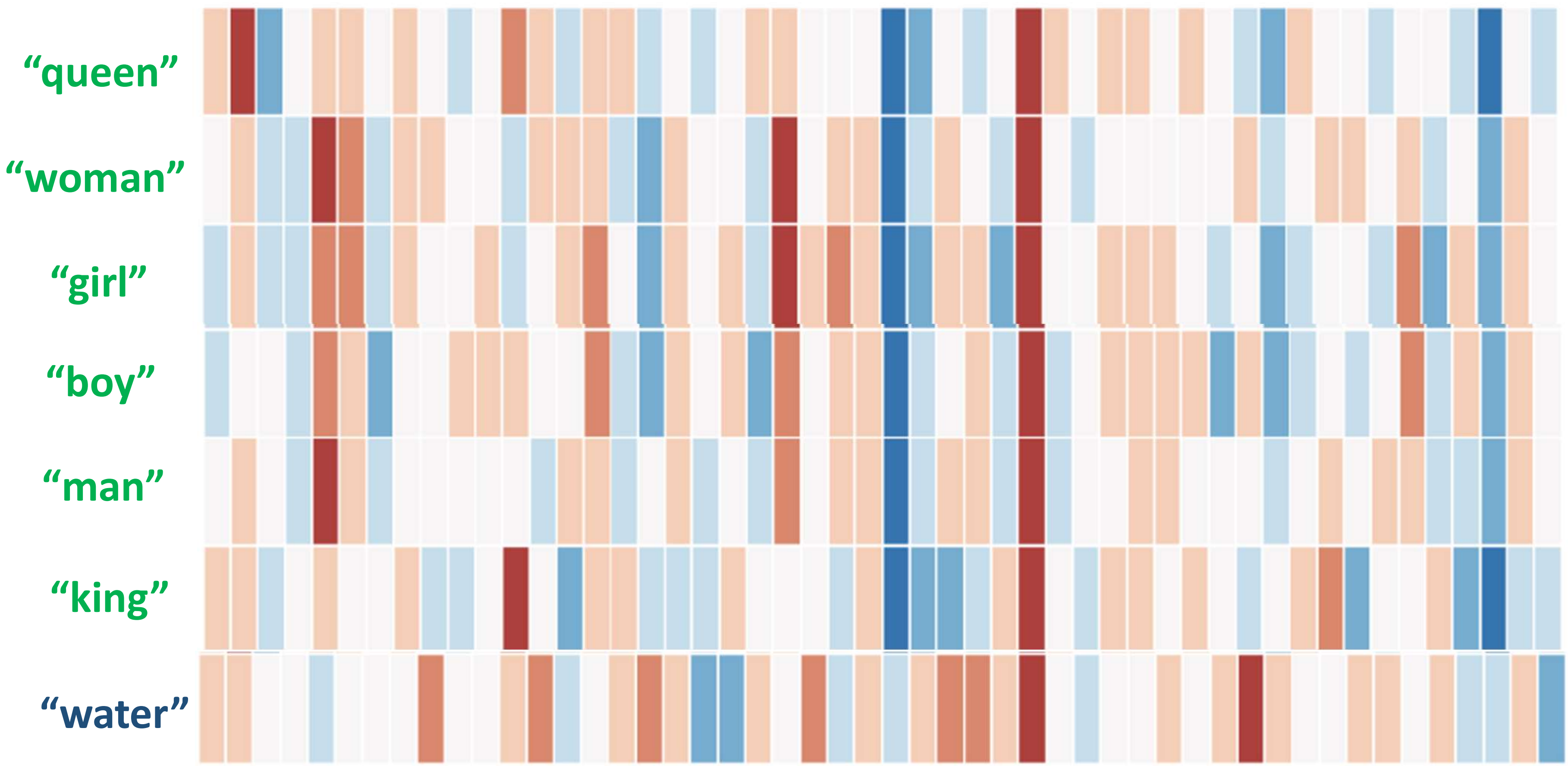


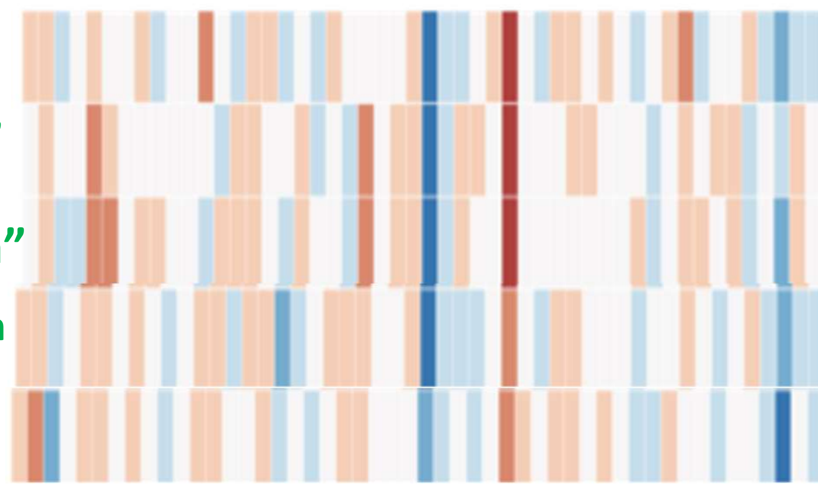
Image Reference: https://jalammar.github.io/illustrated-word2vec/

Analogies

- You can add and subtract word embeddings and see the concept of analogies

king – man + woman ≅ queen

“king”
“man”
“woman”
“king-man +woman”
“queen”

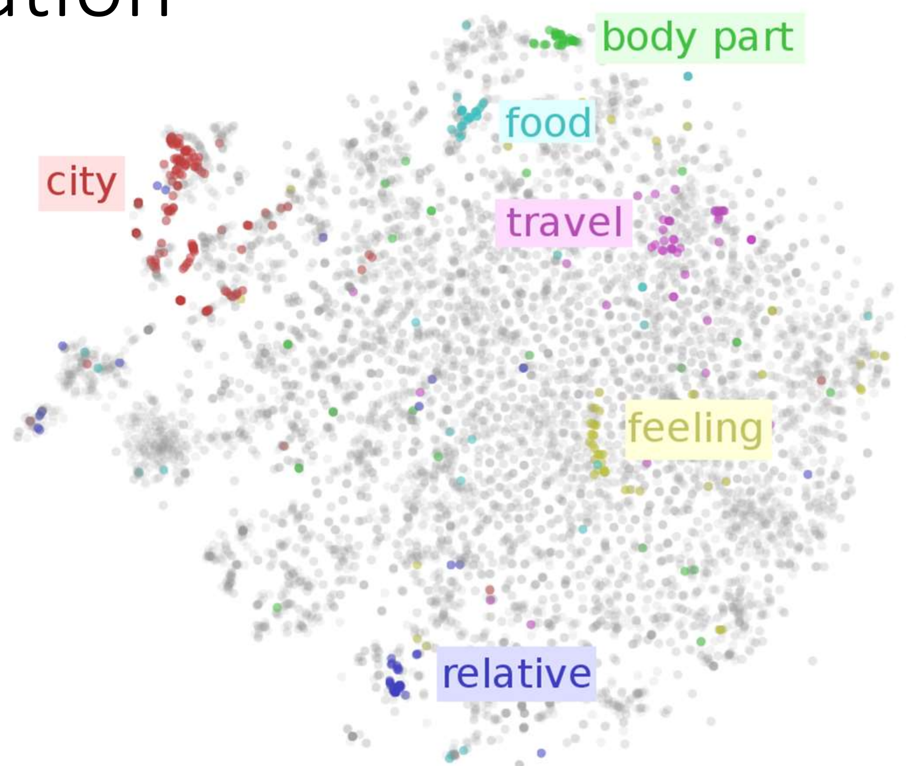


```
model.most_similar(positive=["king","woman"], negative=["man"])
```

```
[('queen', 0.8523603677749634),
 ('throne', 0.7664333581924438),
 ('prince', 0.7592144012451172),
 ('daughter', 0.7473883032798767),
 ('elizabeth', 0.7460219860076904),
 ('princess', 0.7424570322036743),
 ('kingdom', 0.7337411642074585),
 ('monarch', 0.721449077129364),
 ('eldest', 0.7184862494468689),
 ('widow', 0.7099430561065674)]
```

Image Reference: https://jalammar.github.io/illustrated-word2vec/

T-SNE visualization



Word Embedding Visualized with t-SNE

Image Source: http://colah.github.io/posts/2015-01-Visualizing-Representations/

T-SNE visualization

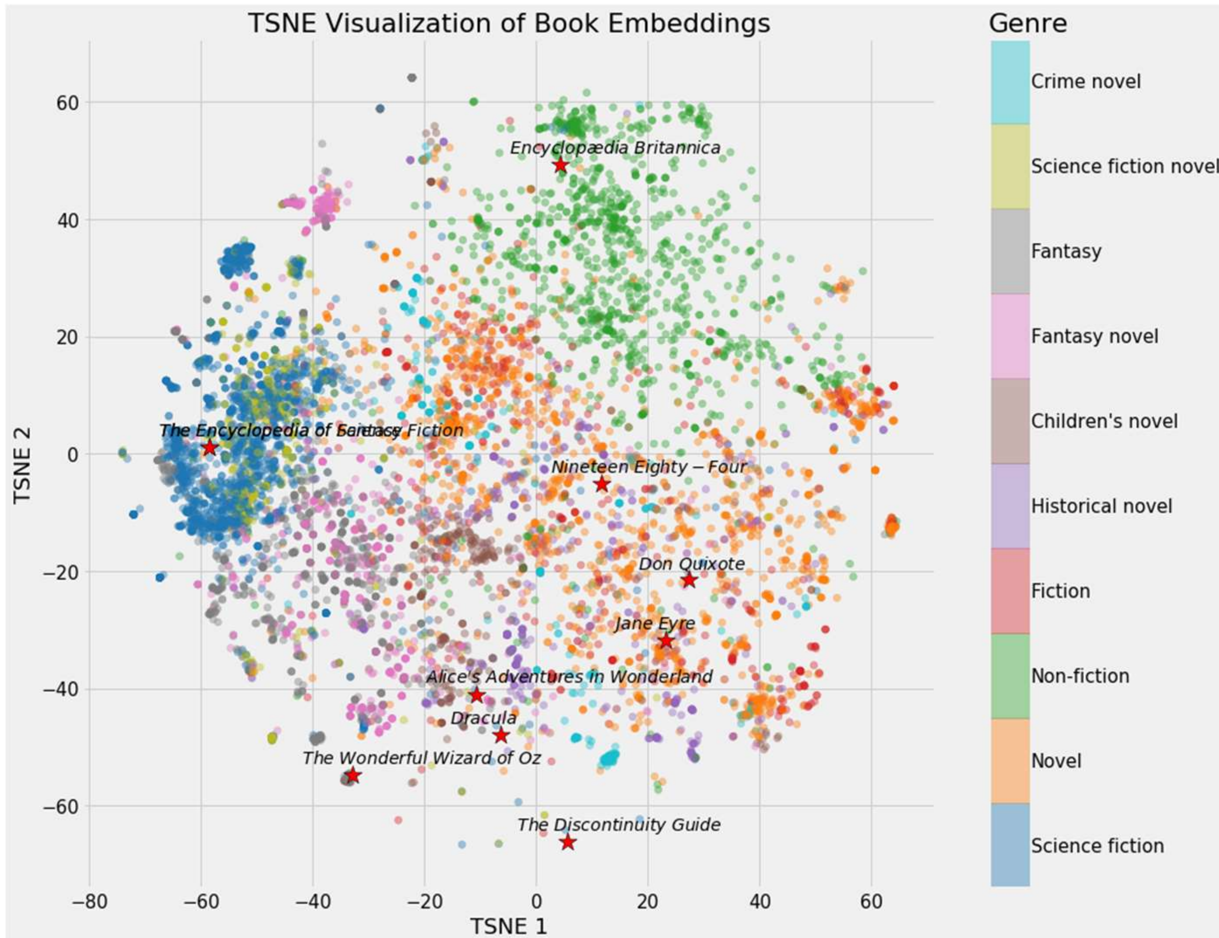


Image Source: https://miro.medium.com/v2/resize:fit:1100/format:webp/1*zAdi7DntawgPsQekPkFxPA.png

Vector Embedding of Words

- We will discuss 2 methods of vector embedding

  - Latent Semantic Analysis/Indexing (1988)
    - Term weighting-based model
    - Consider occurrences of terms at document level.
  - Word2Vec (2013)
    - Prediction-based model.
    - Consider occurrences of terms at context level.

Latent Semantic Analysis

- Latent semantic analysis studies documents in Bag-Of-Words model (1988).
- LSA takes meaningful text documents and recreates them in x different parts where each part expresses a different way of looking at meaning in the text – Dimensionality Reduction Technique

m words

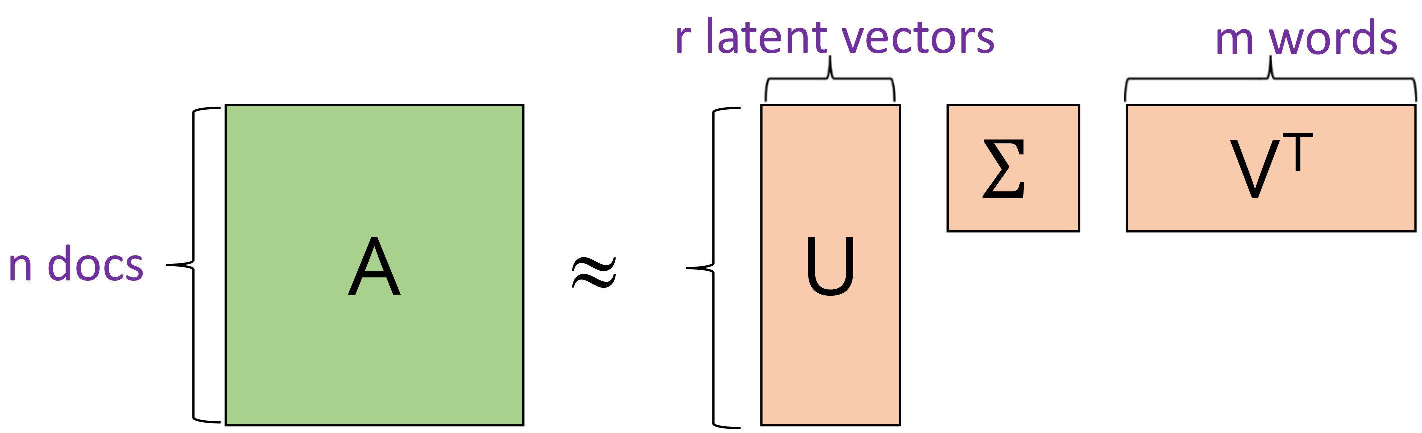
Words / Docs	W_1	W_2	…	W_m
D_1				
D_2	TF-IDF Score of the unique words in the respective documents			
…				
D_n				

n documents

$A = n \times m$ **Matrix**

Latent Semantic Analysis

- Low rank SVD decomposition: $A_{n\times m} = U_{n\times r}\,\Sigma_{r\times r}\,(V_{m\times r})^T$
  - $\boldsymbol{U}$: document-to-concept similarities matrix (orthogonal matrix), with the n documents in the rows and r concepts in the columns
  - **V**: word-to-concept similarities matrix (orthogonal matrix).
  - $\boldsymbol{\Sigma}$: strength of each concept, diagonal *(r,r)* matrix.



Latent Semantic Analysis

- Low rank SVD decomposition: $A_{n \times m} = U_{n \times r} \, \Sigma_{r \times r} \, (V_{m \times r})^T$
- Then given a word ***w*** (column of ***A***):
  - $\boldsymbol{\varsigma} = \boldsymbol{w}^{\boldsymbol{T}} \times \boldsymbol{U}$ is the embedding (encoding) of the word ***w*** in the latent space.
  - $\boldsymbol{w} \approx \boldsymbol{U} \times \boldsymbol{\varsigma}^{\boldsymbol{T}} = \boldsymbol{U} \times (\boldsymbol{w}^{\boldsymbol{T}} \times \boldsymbol{U})^{\boldsymbol{T}}$ is the decoding of the word ***w*** from its embedding.
- An SVD factorization gives the **best possible reconstructions** of the word ***w*** from its embedding.
- **SVD is computationally expensive and restrictive**.
- **Models are based on linear algebra, so they are limited in their ability to capture non-linear relationships between words or concepts.**

Prediction Based Models

- Mikolov et al. in 2013 originally proposed two new model architectures, the architectures being computationally less expensive compared to SVD
    - Continuous Bag-of-Words Model (CBoWM)
    - Continuous Skip-gram Model (CSgM)
- CBoWM uses words ***n*** positions away from each center word.
    - These words are called **context words**.
- Example for ***k=3***:
    - "It has been raining heavily today since morning, and my umbrella is missing".
    - Center word: red (also called focus word).
    - Context words: blue (also called target words).
- These models consider all words as center words, and all their corresponding context words.

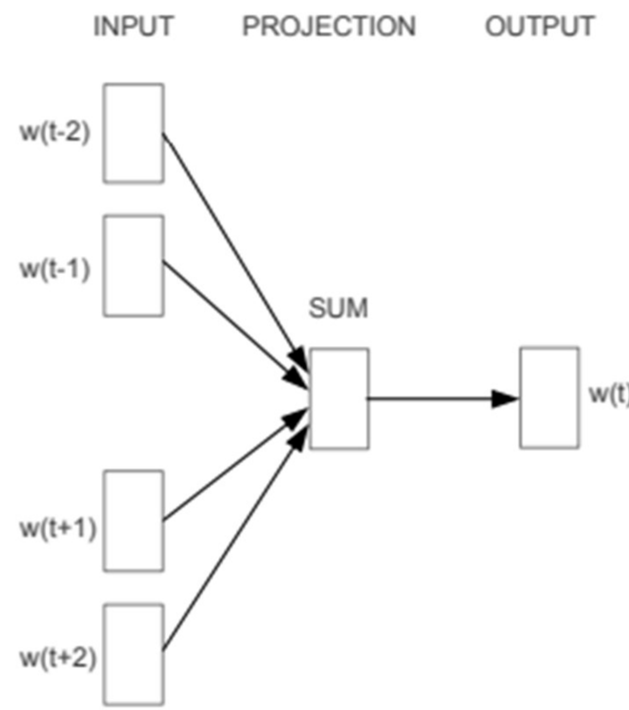
CBoWM

- CBoWM focusses on guessing a word based on its context.

- Predicting n[th] word given previous n-1 words
  Example: She sat on the **chair**

- Training data: All n-word windows in the corpus

Modern humans arrived the Indian subcontinent from Africa no later than 55,000 years ago. Their long occupation, initially in varying forms of isolation as hunter-gatherers, has made the region highly diverse, second only to Africa in human genetic diversity. Settled life emerged on the subcontinent in western margins of the Indus river basin 9,000 years ago, evolving gradually into the Indus Valley Civilisation of the third millennium BCE.



Continuous bag-of-words (Mikolov et al., 2013)

$$J_\theta = \frac{1}{T}\sum_{t=1}^{T} \log p(w_t | w_{t-n}, \dots, w_{t-1}, w_{t+1}, \dots, w_{t+n})$$

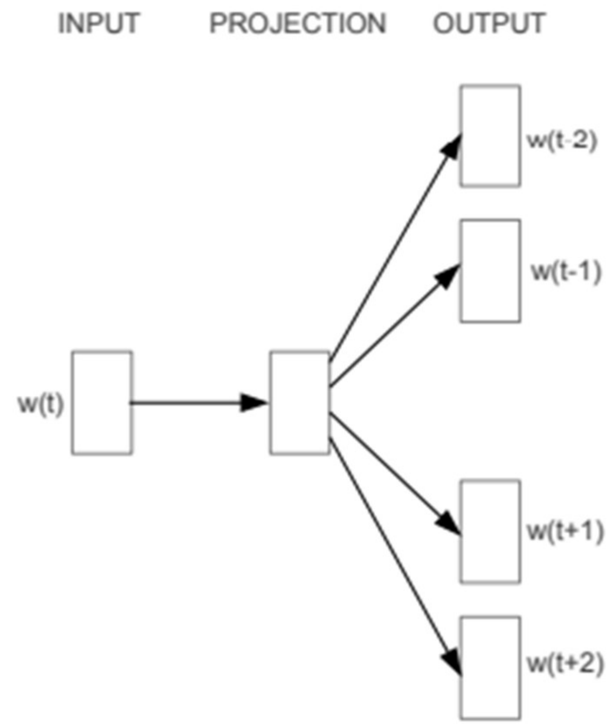
Skip-gram

- Unlike CBoW which uses surrounding words to predict the centre word, skip-gram uses the centre word to predict the surrounding words

Doubt kills more dreams than failure ever will

- CBoW: Doubts kills more __dreams__ than failure ever will
- Ski-gram:

Input Word	Target Word
dreams	kills
dreams	more
dreams	than
dreams	failure



Skip-gram - Pipeline

- Context-Target Pairs: For each word in the corpus, consider a fixed window of words around it. This window defines the context for that word. Create pairs of (target word, context word) based on the words within this window.

Doubt kills more dreams than failure ever will

Target Word	Context Word
more	doubt
more	kills
more	dreams
more	than
dreams	kills
dreams	more
dreams	than
dreams	failure

Skip-gram: window size = 2

- Convert the target word in each (target, context) pair into a one-hot encoded vector. Example X1="large brown dog", X2 ="small white cat"

Word	Word OHE	Neighbour	Neighbour OHE
large	[1 0 0 0 0 0]	brown	[0 1 0 0 0 0]
large	[1 0 0 0 0 0]	dog	[0 0 1 0 0 0]
brown	[0 1 0 0 0 0]	large	[1 0 0 0 0 0]
brown	[0 1 0 0 0 0]	dog	[0 0 1 0 0 0]
dog	[0 0 1 0 0 0]	large	[1 0 0 0 0 0]
dog	[0 0 1 0 0 0]	brown	[0 1 0 0 0 0]
small	[0 0 0 1 0 0]	white	[0 0 0 0 1 0]
small	[0 0 0 1 0 0]	cat	[0 0 0 0 0 1]
white	[0 0 0 0 1 0]	small	[0 0 0 1 0 0]
white	[0 0 0 0 1 0]	cat	[0 0 0 0 0 1]
cat	[0 0 0 0 0 1]	small	[0 0 0 1 0 0]
cat	[0 0 0 0 0 1]	white	[0 0 0 0 1 0]

Word2Vec: window size = 2

Example X1="large brown dog", X2 ="small white cat"

Word	Word OHE	Neighbour	Neighbour OHE
large	[1 0 0 0 0 0]	brown	[0 1 0 0 0 0]
		dog	[0 0 1 0 0 0]
brown	[0 1 0 0 0 0]	large	[1 0 0 0 0 0]
		dog	[0 0 1 0 0 0]
dog	[0 0 1 0 0 0]	large	[1 0 0 0 0 0]
		brown	[0 1 0 0 0 0]
small	[0 0 0 1 0 0]	white	[0 0 0 0 1 0]
		cat	[0 0 0 0 0 1]
white	[0 0 0 0 1 0]	small	[0 0 0 1 0 0]
		cat	[0 0 0 0 0 1]
cat	[0 0 0 0 0 1]	small	[0 0 0 1 0 0]
		white	[0 0 0 0 1 0]

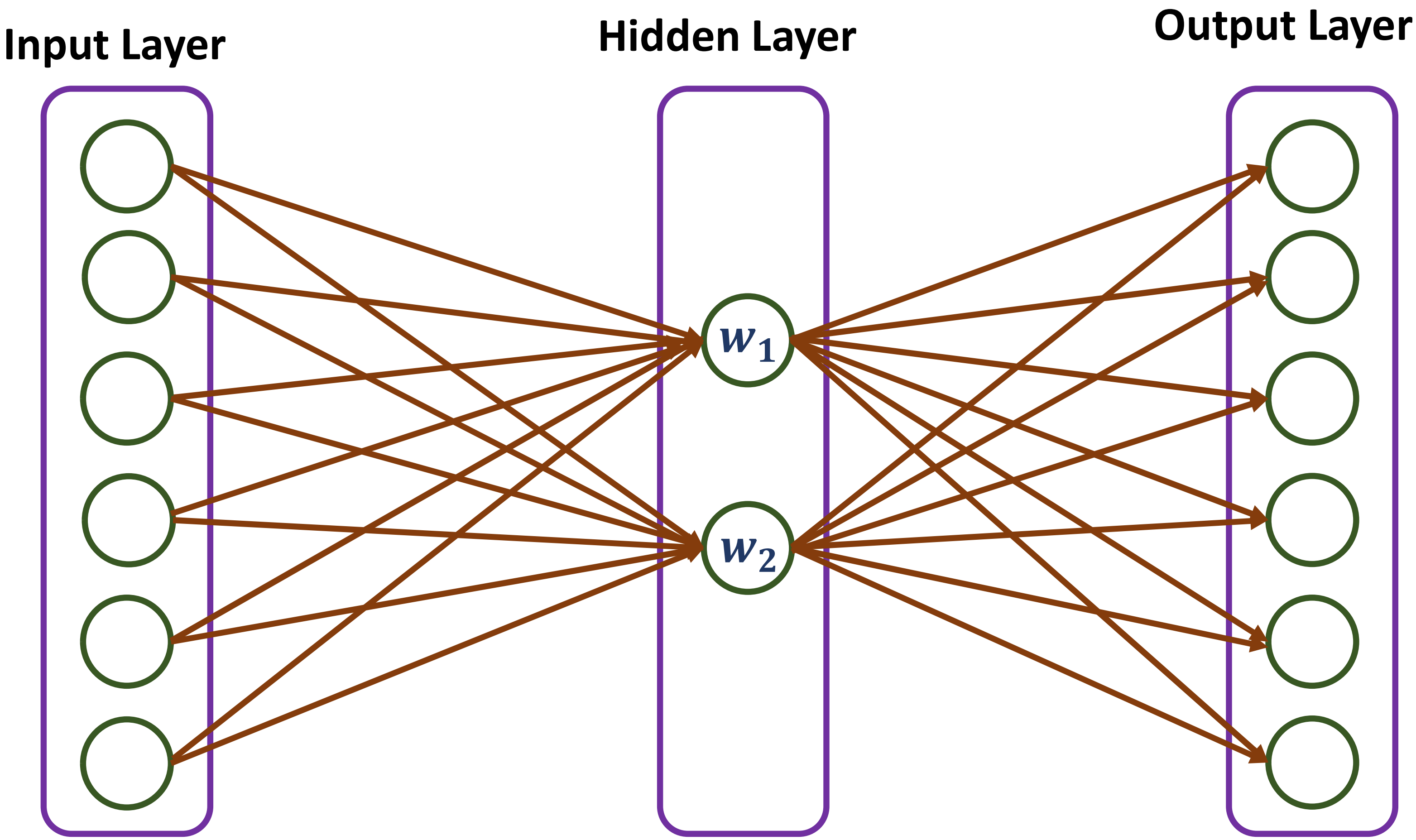
Skip-gram Neural Network Architecture

- Train a neural network with a single hidden layer
- The input to the neural network is the one-hot encoded vector of the target word
- The output layer has as many neurons as there are words in the vocabulary
- The weights of this neural network represent the word embeddings you want to learn

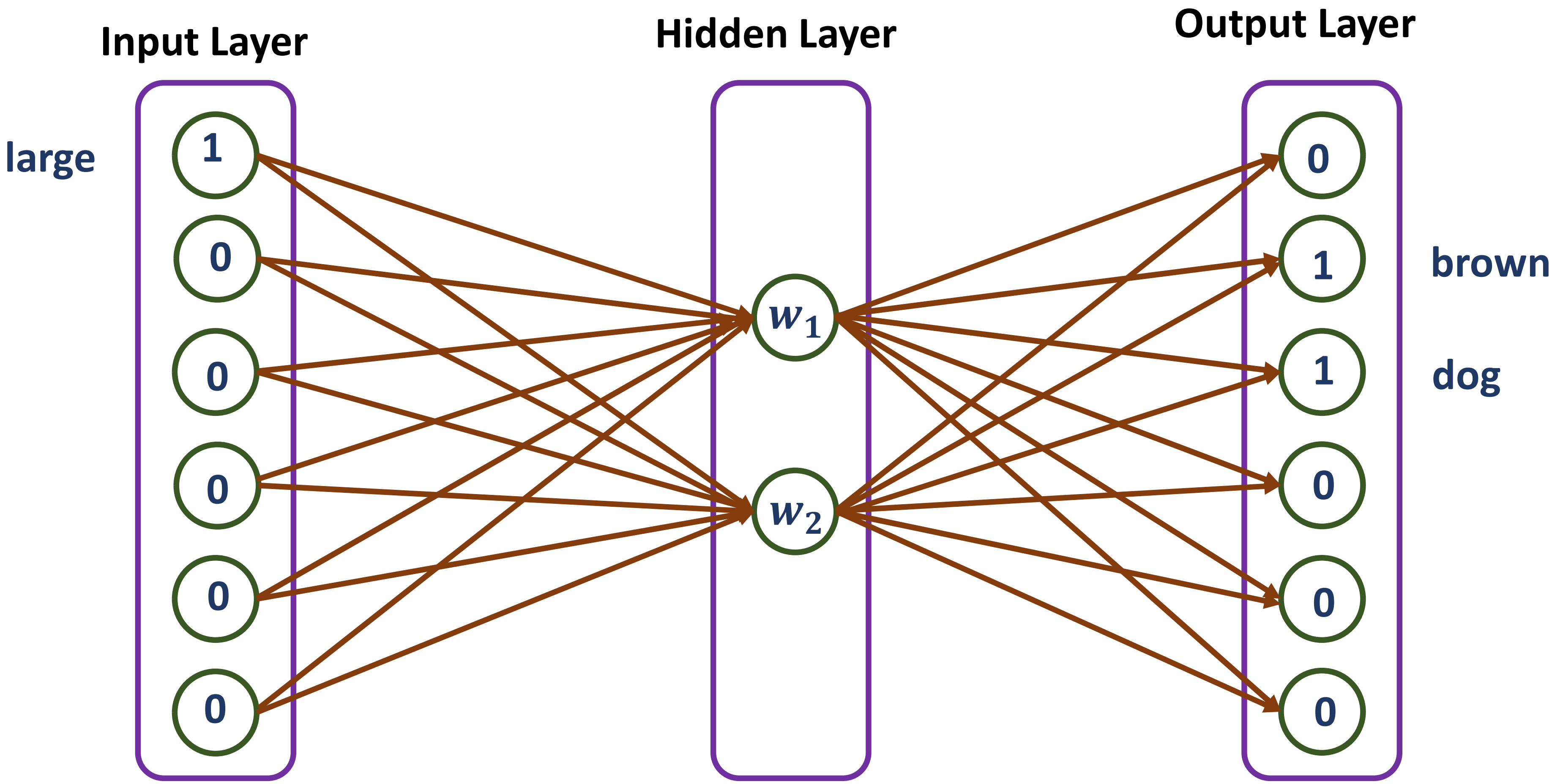
Example X1="large brown dog", X2 ="small white cat"

- To learn the word-embedding for **brown**
- The input layer will have input [0 1 0 0 0 0], i.e., 6 neurons
- The output layer will have 6 neurons, as there are 6 words in the vocabulary

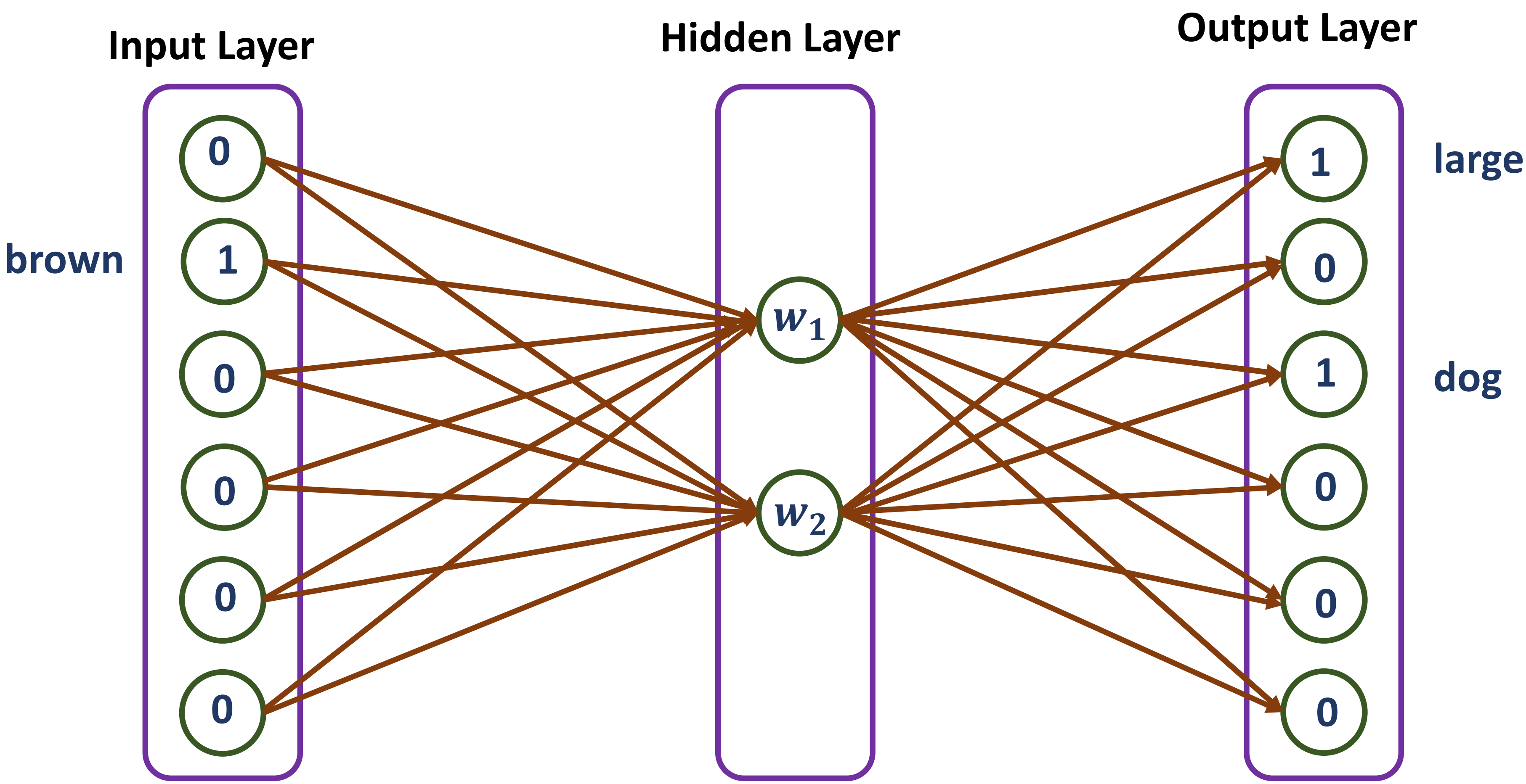
NN architecture



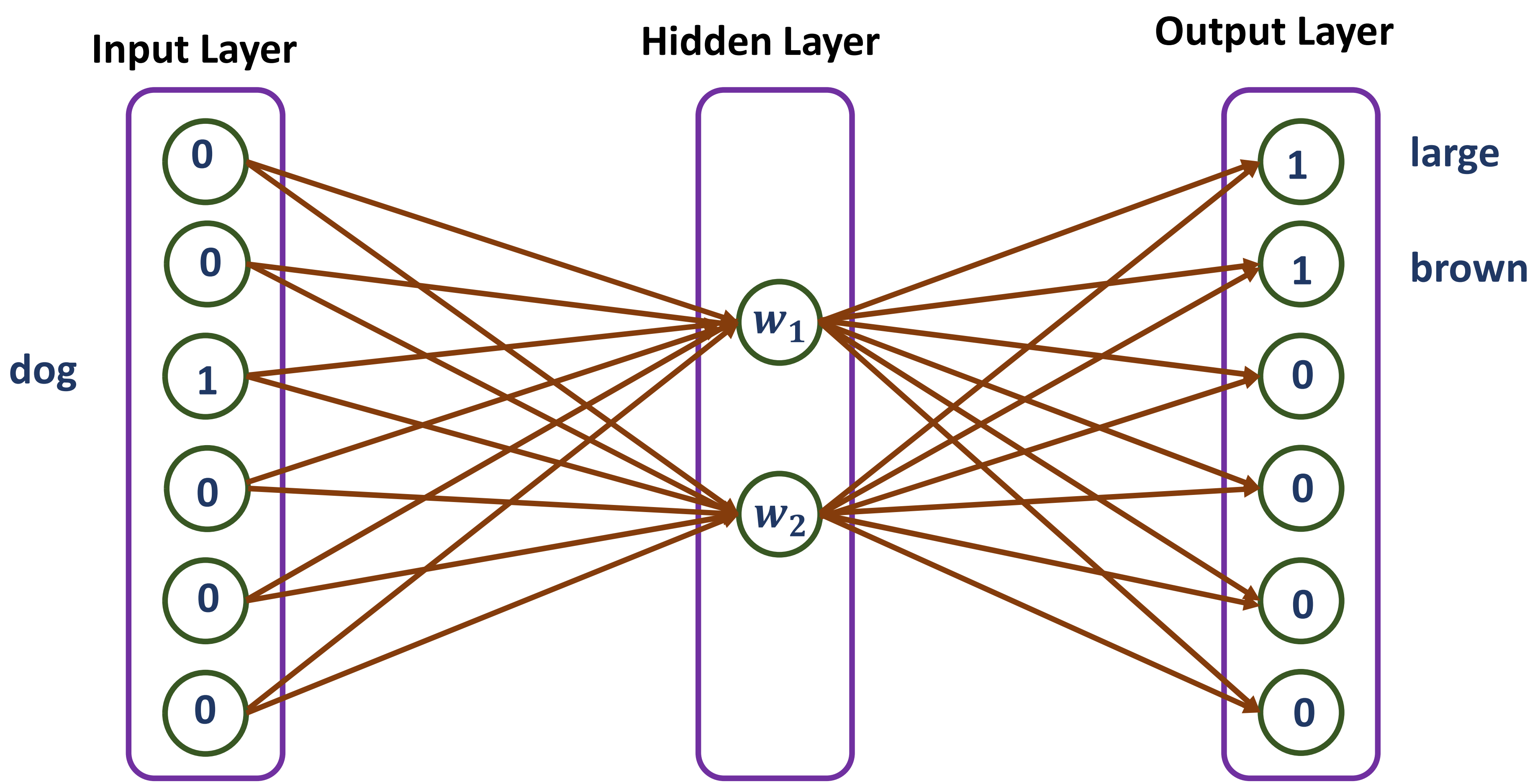
NN architecture



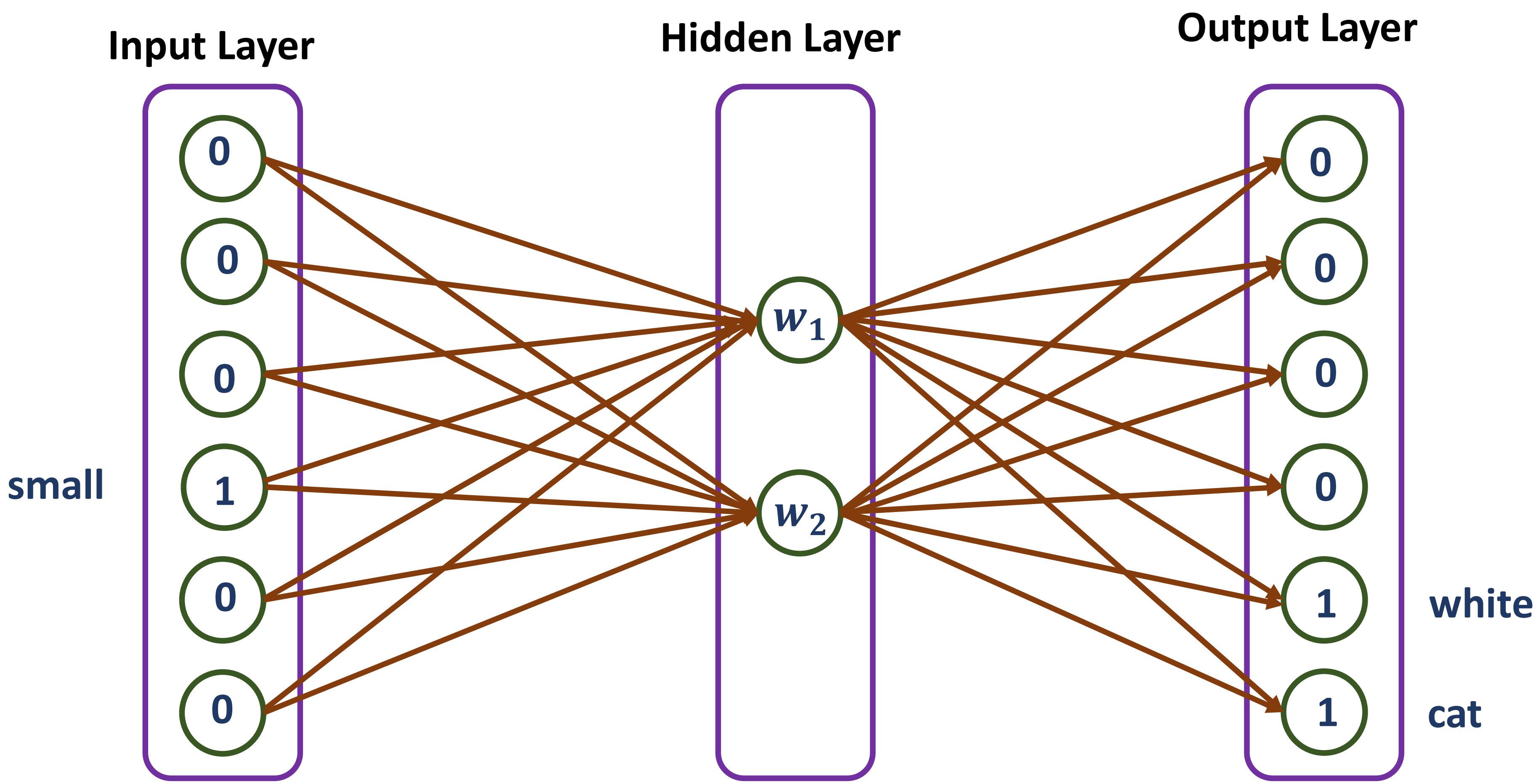
NN architecture



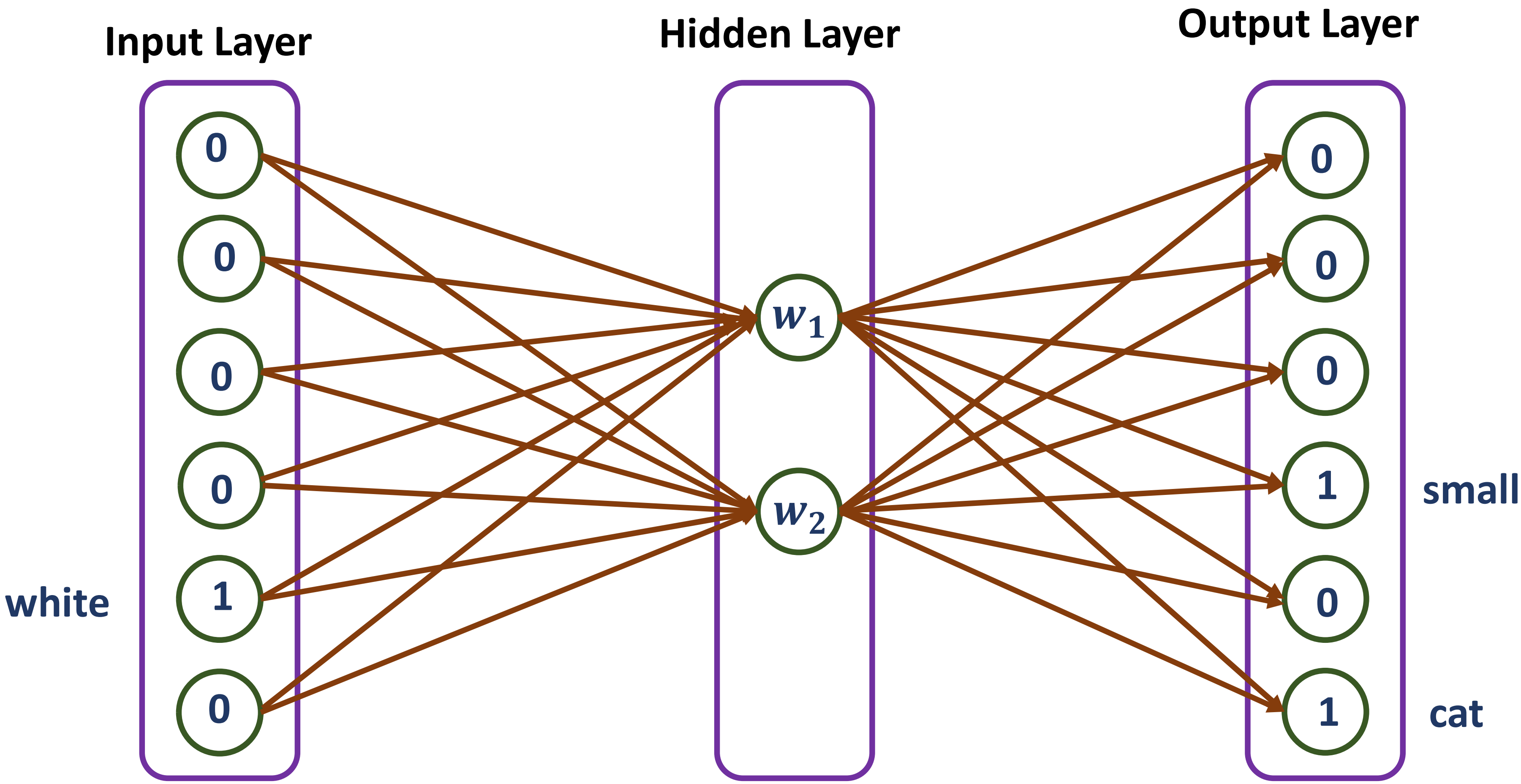
NN architecture



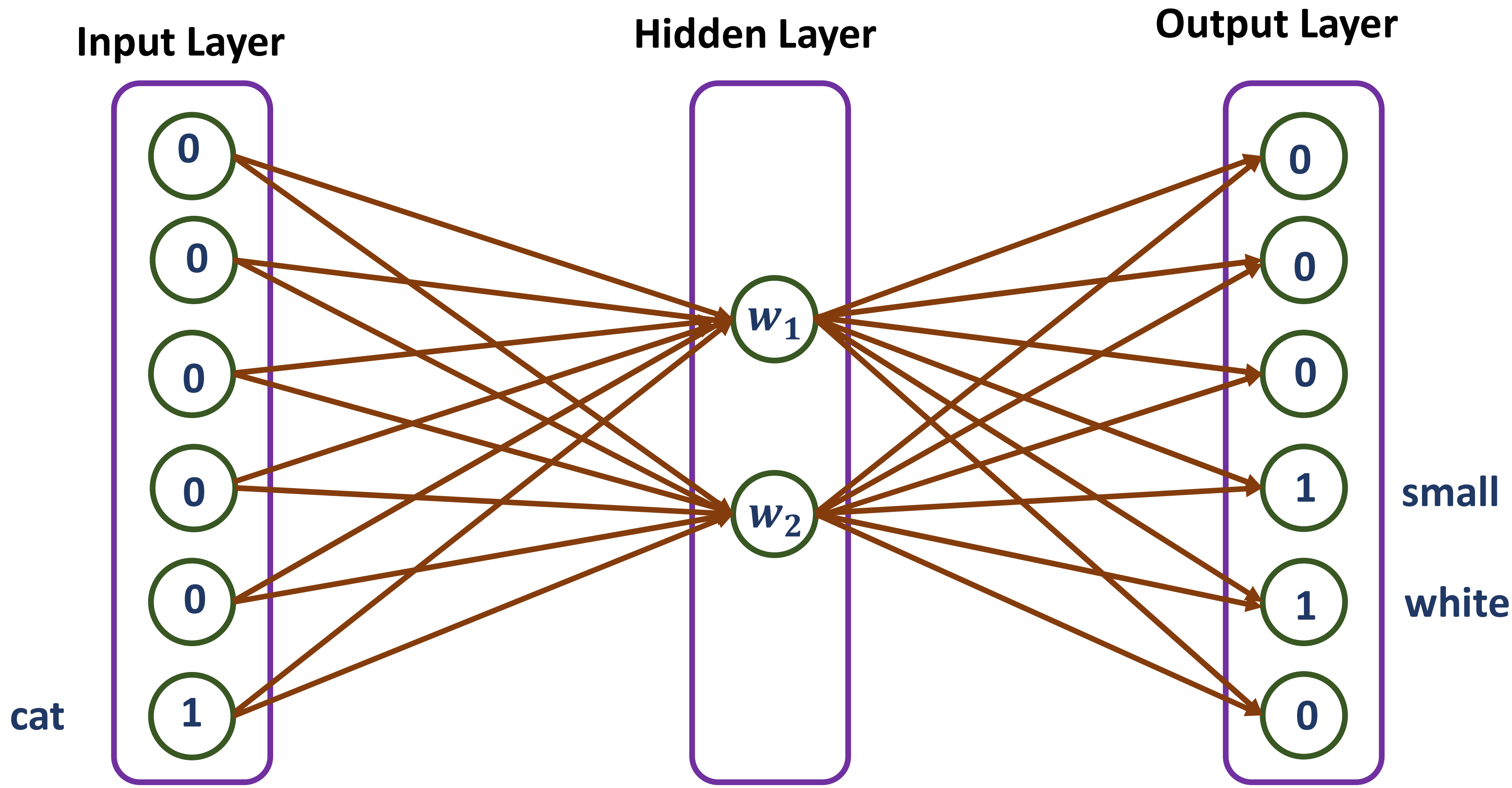
NN architecture



NN architecture



NN architecture



Skip-gram

- The objective of the neural network is to predict the context words given the target word (or vice versa).
- The network is trained to minimize the cross-entropy loss between the predicted and actual context words.
- Once the model is trained, the weights of the hidden layer of the neural network represent the word embeddings.
- These learned embeddings capture semantic and contextual information about words based on the patterns in the training corpus.

Relations learned by Word2Vec

Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - France	Berlusconi: Italy	Merkel: Germany	Koizumi: Japan
copper - Cu	zinc: Zn	gold: Au	uranium: plutonium
Berlusconi - Silvio	Sarkozy: Nicolas	Putin: Medvedev	Obama: Barack
Microsoft - Windows	Google: Android	IBM: Linux	Apple: iPhone
Microsoft - Ballmer	Google: Yahoo	IBM: McNealy	Apple: Jobs
Japan - sushi	Germany: bratwurst	France: tapas	USA: pizza

Skip-gram model trained on 783M words with 300 dimensionality, using the relationship defined as *"Paris" – "France" + "Italy" = "Rome"*, Mikolov et.al.