Vector Semantics and Embeddings

Natural Language Processing

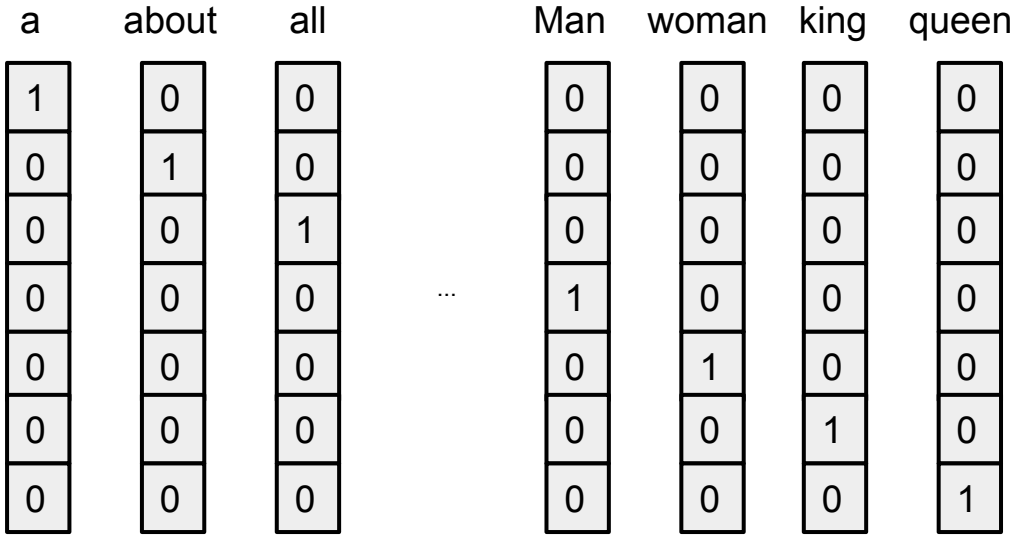
Younes Samih and David Arps

Word Representations in NLP

- Neural Network
- Distributed Word representations
  - Raw text
  - Representing words as low-dimensional vectors
  - These vectors capture Semantics and Syntax
- Word2Vec
- GloVe
- Fasttext

Word Representation

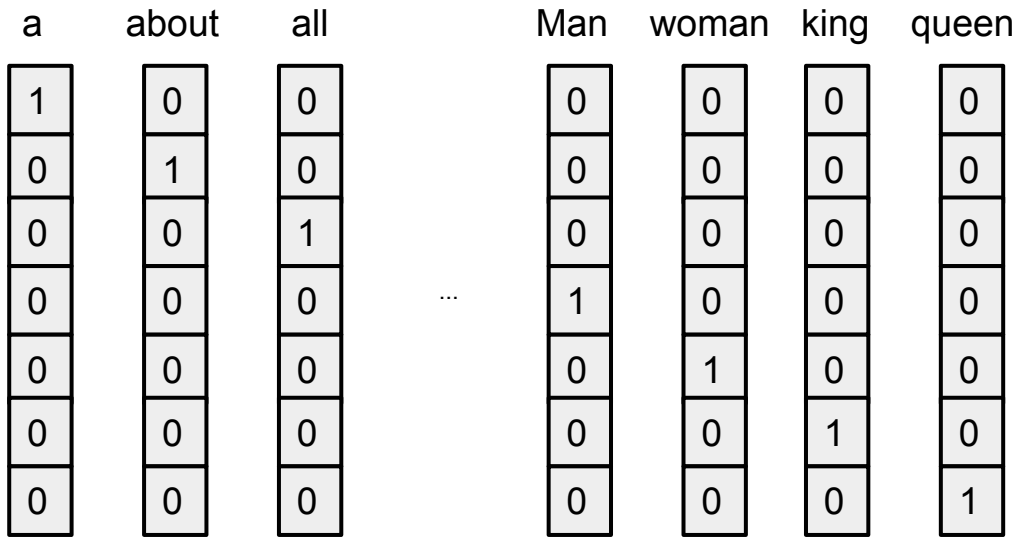
One-hot encoding



Word Representation

Weaknesses One-hot encoding

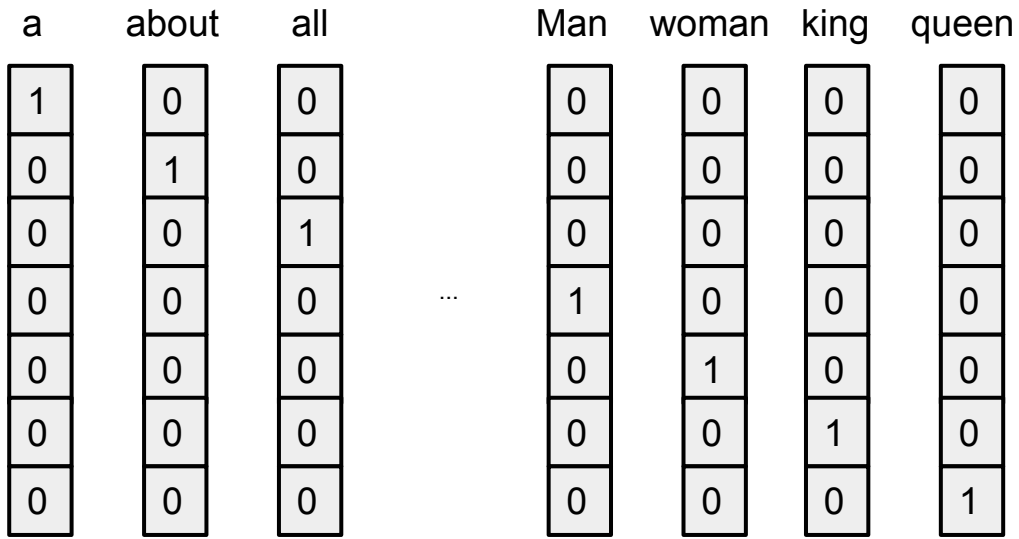
- The **distance** between any pair of words is the same



Word Representation

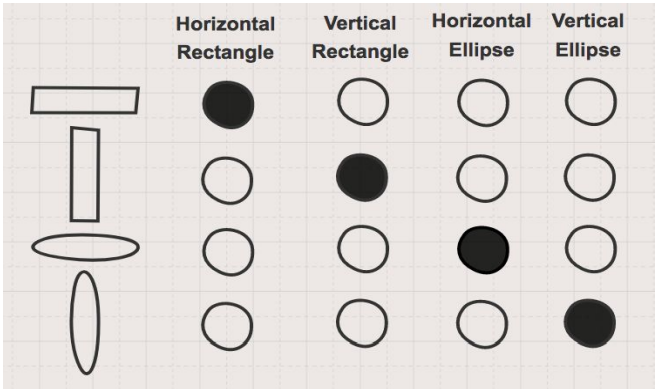
Weaknesses One-hot encoding

- The **distance** between any pair of words is the same
- Can not capture word similarity

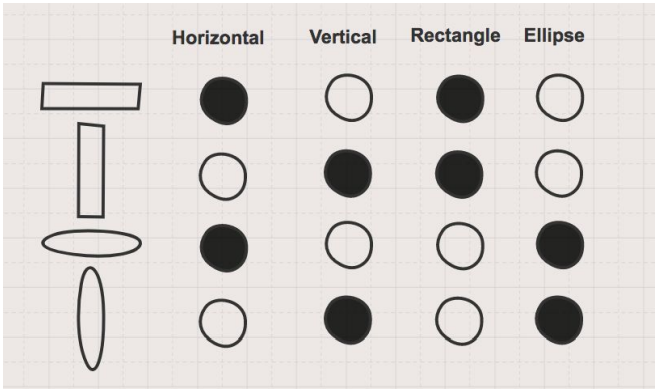


Word Representation

Distributed representation



Non-distributed representation



distributed representation

Image source: https://www.oreilly.com/content/how-neural-networks-learn-distributed-representations/

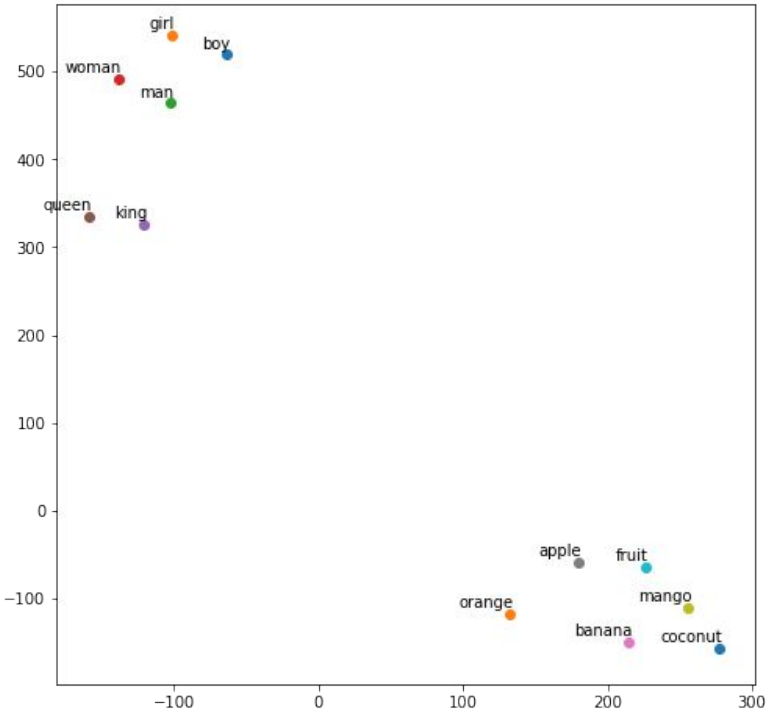
Word Representation

Distributed representation of words

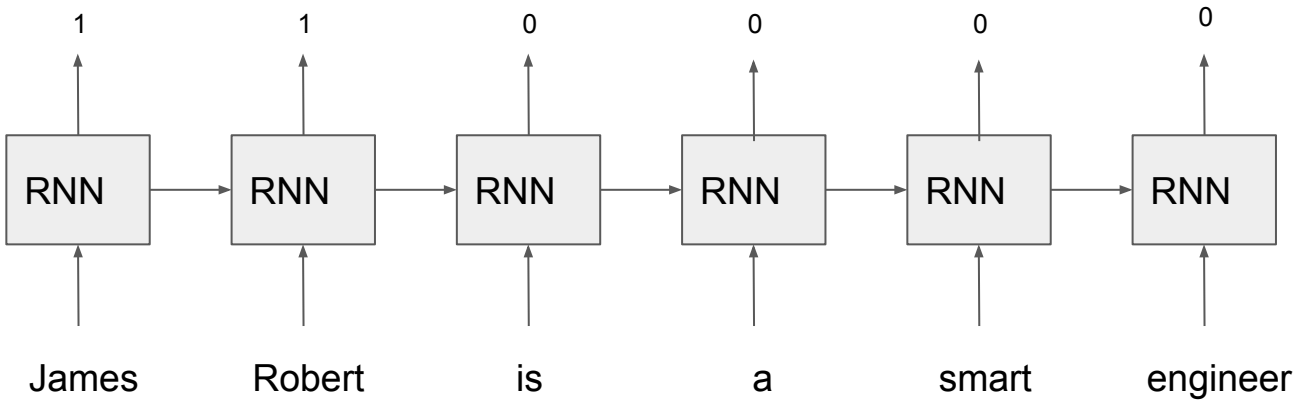
	Man	Women	Knig	Queen	Apple	Orange
Gender	-1	1	-0.98	0.97	0.00	-0.01
Royal	0.01	0.02	0.93	0.98	-0.01	0.00
age	0.03	0.02	0.72	0.56	0.03	0.02
Food	0.00	0.00	0.01	0.02	0.96	0.93

Word Representation

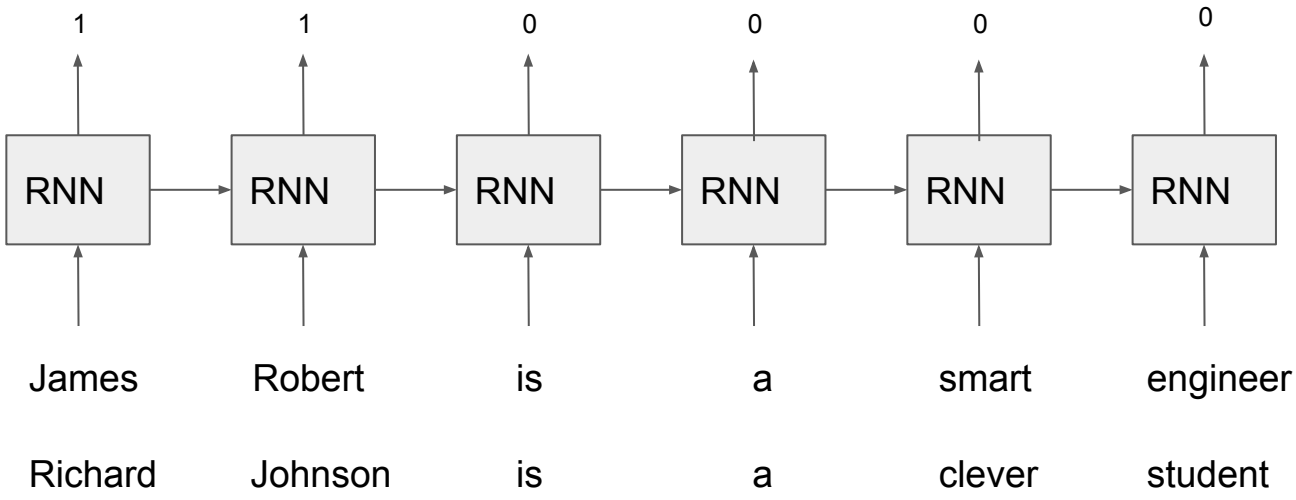
Visualizing word embeddings



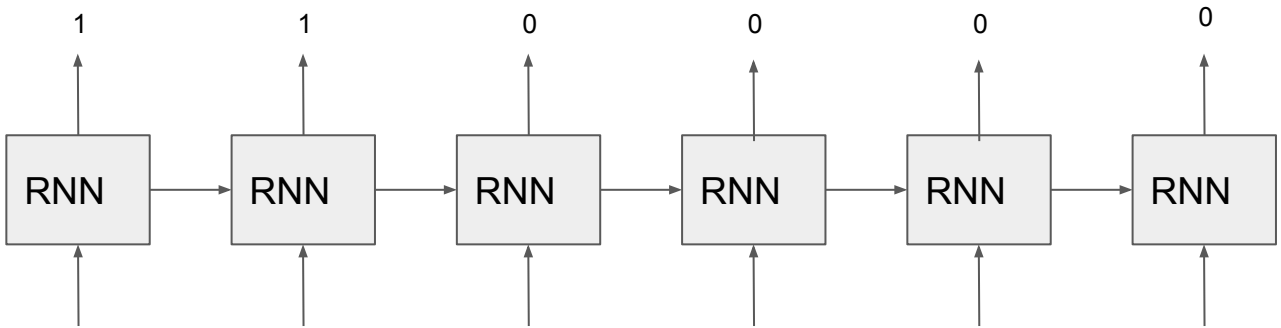
Application of Word Embeddings



Application of Word Embeddings



Application of Word Embeddings



James	Robert	is	a	smart	engineer
Richard	Johnson	is	a	clever	student
Jone	clark	is	a	nuclear	scientist

Transfer Learning

- Train word embeddings using large text corpus

Transfer Learning

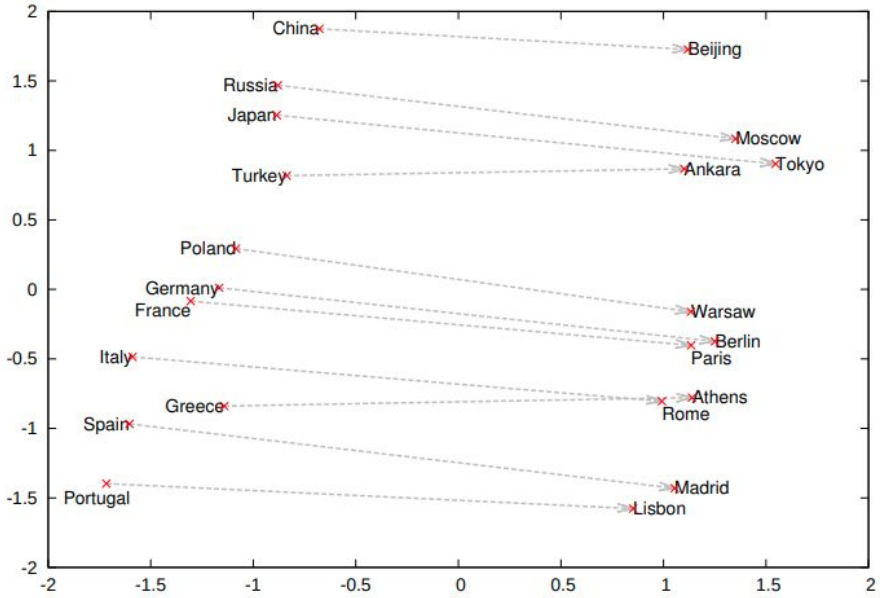
- Train word embeddings using large text corpus
- Use pre-trained word embeddings

Transfer Learning

- Train word embeddings using large text corpus
- Use pre-trained word embeddings
- Leverage word embedding for new tasks with smaller training data set

Analogies and Word Embeddings

Analogy Reasoning



Reference: [Mikolov et. al., 2013, Distributed Representations of Words and Phrases and their Compositionality]

Word Embeddings

Word Similarity

- Similarity is calculated using cosine similarity :

Word Embeddings

Word Similarity

- Similarity is calculated using cosine similarity :

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

Word Embeddings

Word Similarity

- Similarity is calculated using cosine similarity :

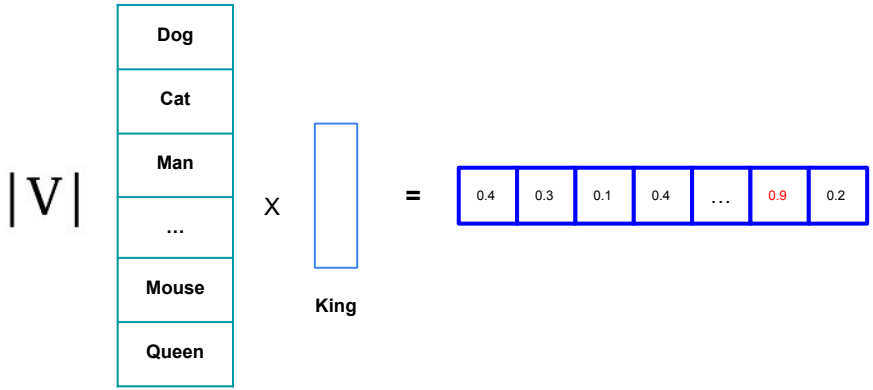
$$\cos\theta = \frac{\vec{a} \cdot \vec{b}}{\|\vec{a}\| \cdot \|\vec{b}\|}$$

- vectors need to normalised when loading them

Similarity and Word Embeddings

Finding the most similar word to **King**

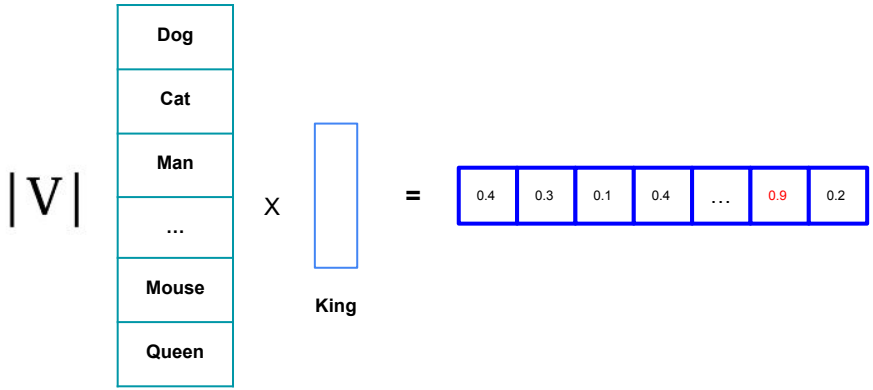
- Compute the similarity from word **King** to all other words.



Similarity and Word Embeddings

Finding the most similar word to **King**

- Compute the similarity from word **King** to all other words.

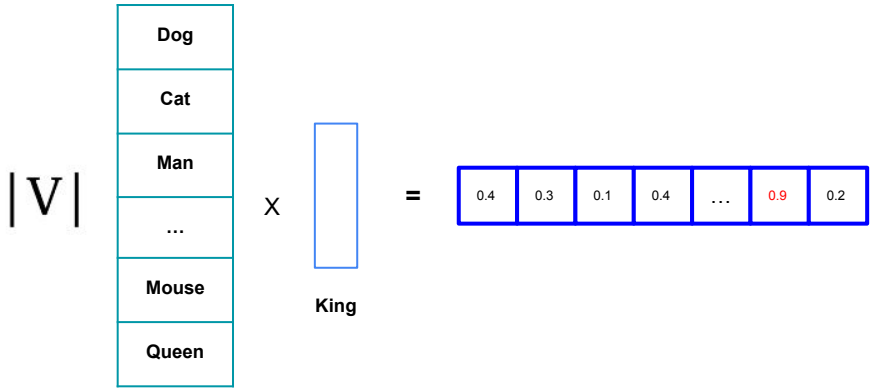


- This is a single matrix-vector product

Similarity and Word Embeddings

Finding the most similar word to **King**

- Compute the similarity from word **King** to all other words.

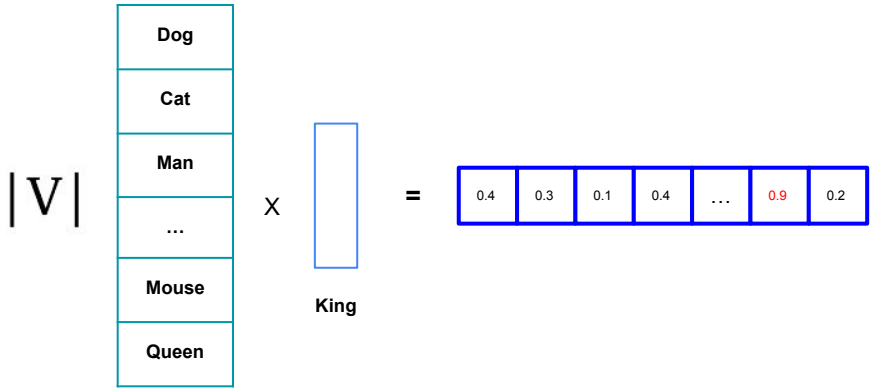


- This is a single matrix-vector product
- Result is a | V | sized vector of similarities

Similarity and Word Embeddings

Finding the most similar word to **King**

- Compute the similarity from word **King** to all other words.



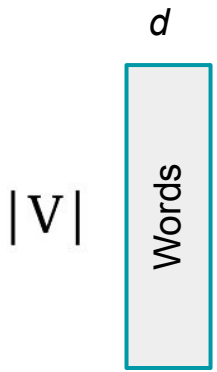
- This is a single matrix-vector product
- Result is a | V | sized vector of similarities
- Get the k -highest values
- But where do these vectors come from?

Word2Vec

Text

word2vec

Cat	=	0.8	0.4	0.3	0.2	…	0.4	0.8	0.2	0.3	0.4
Dog	=	0.5	0.9	0.7	0.3	…	0.5	0.7	0.8	0.4	0.3
King	=	0.4	0.3	0.1	0.4	…	0.4	0.2	0.6	0.3	0.2



Word2Vec

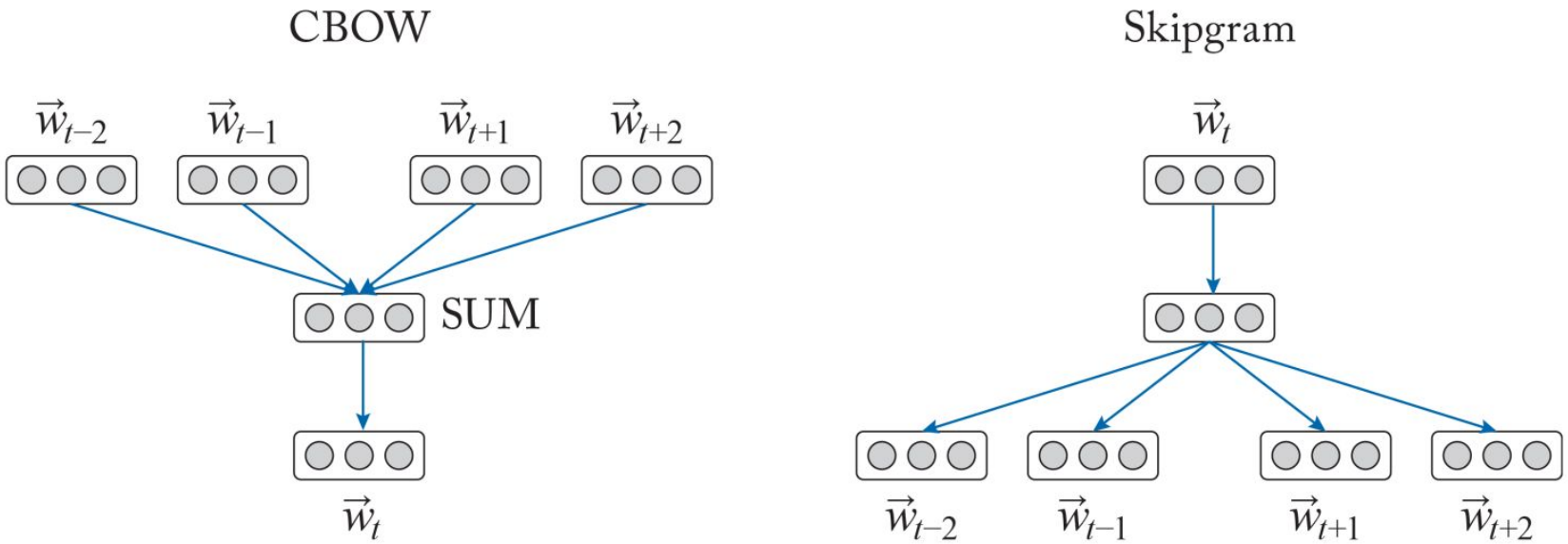


Image:[Pilehvar, Mohammad Taher, and Jose Camacho-Collados. "Embeddings in natural language processing: theory and advances in vector representations of meaning"]

Word2Vec

- Word2vec [Mikolov et al., 2013d]

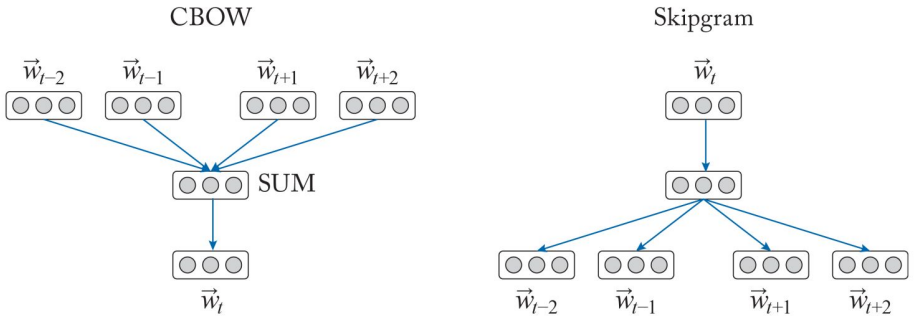


Image:[Pilehvar, Mohammad Taher, and Jose Camacho-Collados. "Embeddings in natural language processing: theory and advances in vector representations of meaning"]

Word2Vec

- Word2vec [Mikolov et al., 2013d]
- FFNN architecture trained with language modeling objective.

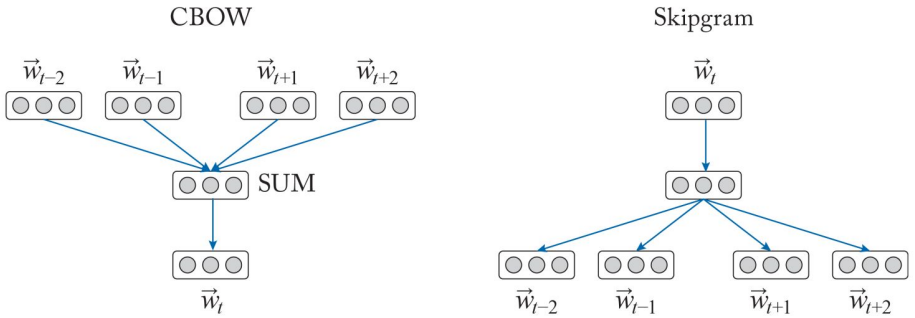


Image:[Pilehvar, Mohammad Taher, and Jose Camacho-Collados. "Embeddings in natural language processing: theory and advances in vector representations of meaning"]

Word2Vec

- Word2vec [Mikolov et al., 2013d]
- FFNN architecture trained with language modeling objective.
- Two different Word2vec models were proposed:
  - Continuous Bag-Of-Words (CBOW)
  - The Skip-gram model

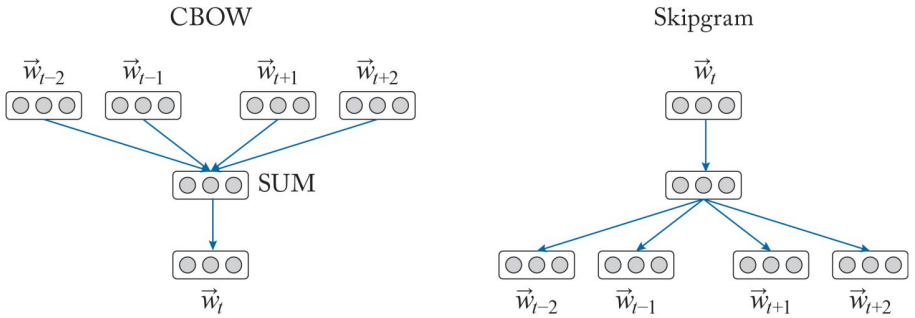


Image:[Pilehvar, Mohammad Taher, and Jose Camacho-Collados. "Embeddings in natural language processing: theory and advances in vector representations of meaning"]

Word2Vec

- The **CBOW** model aims at predicting the current word using its surrounding context
- The **Skip-gram** model aims at predicting the words in the surrounding context given the target word

- Two training methods:
  - Negative sampling
  - Hierarchical Softmax

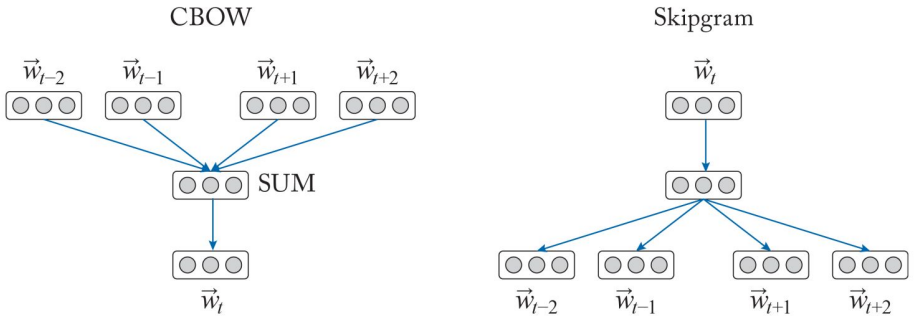


Image:[Pilehvar, Mohammad Taher, and Jose Camacho-Collados. "Embeddings in natural language processing: theory and advances in vector representations of meaning"]

Word2Vec

- The **CBOW** model aims at predicting the current word using its surrounding context
- The **Skip-gram** model aims at predicting the words in the surrounding context given the target word

- Two training methods:
  - Negative sampling
  - Hierarchical Softmax

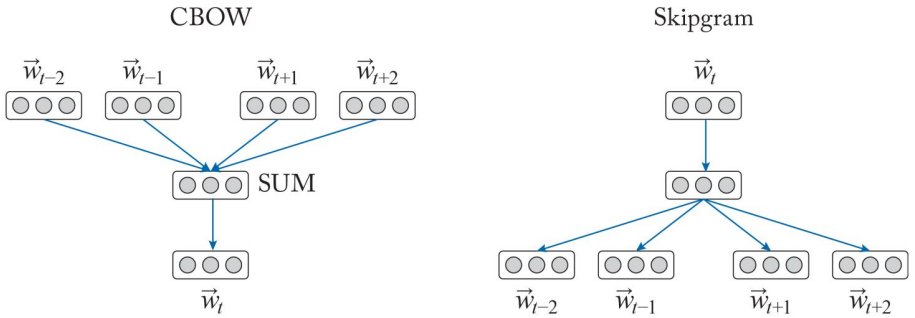
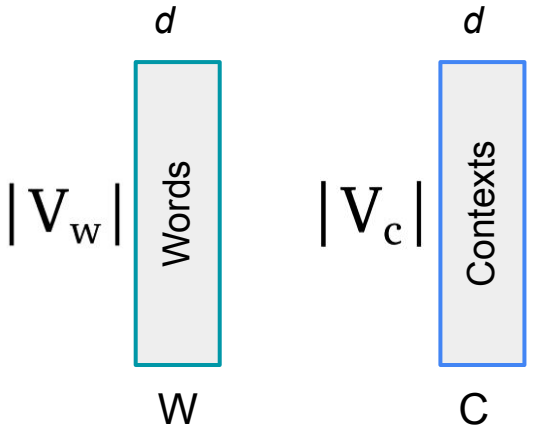


Image:[Pilehvar, Mohammad Taher, and Jose Camacho-Collados. "Embeddings in natural language processing: theory and advances in vector representations of meaning"]

Word2Vec

Training the **Skip-gram** model

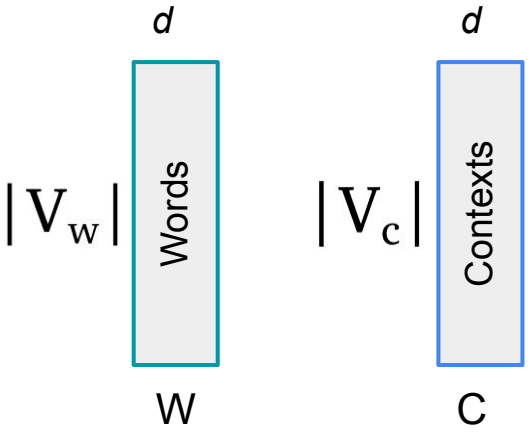
1. Each word is represented as a ***d*** dimensional vector.



Word2Vec

Training the **Skip-gram** model

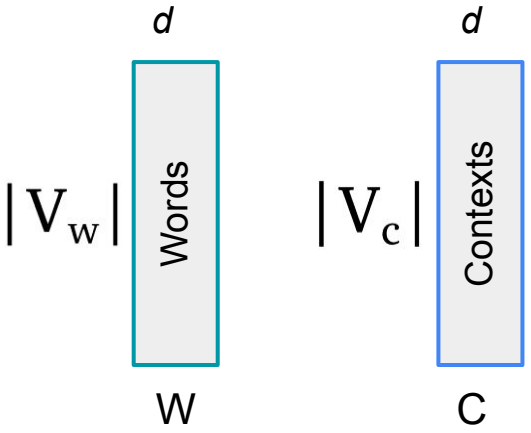
1. Each word is represented as a ***d*** dimensional vector.
2. Each context is represented as a ***d*** dimensional vector.



Word2Vec

Training the **Skip-gram** model

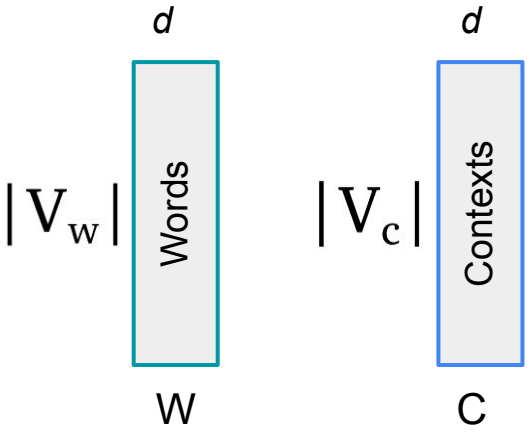
1. Each word is represented as a ***d*** dimensional vector.
2. Each context is represented as a ***d*** dimensional vector.
3. All vectors weights are randomly initialized .



Word2Vec

Training the **Skip-gram** model

1. Each word is represented as a ***d*** dimensional vector.
2. Each context is represented as a ***d*** dimensional vector.
3. All vectors weights are randomly initialized .
4. Arrange vectors in two matrices, *W* and *C*



Word2Vec

Training the **Skip-gram** model

1. Data

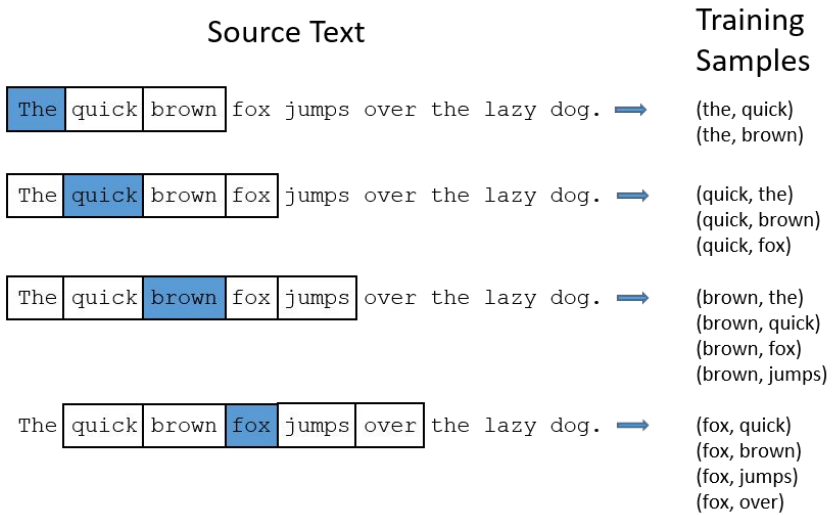


Image:[McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model. Retrieved from http://www.mccormickml.com]

Word2Vec

Training the **Skip-gram** model

1. Data
2. Architecture

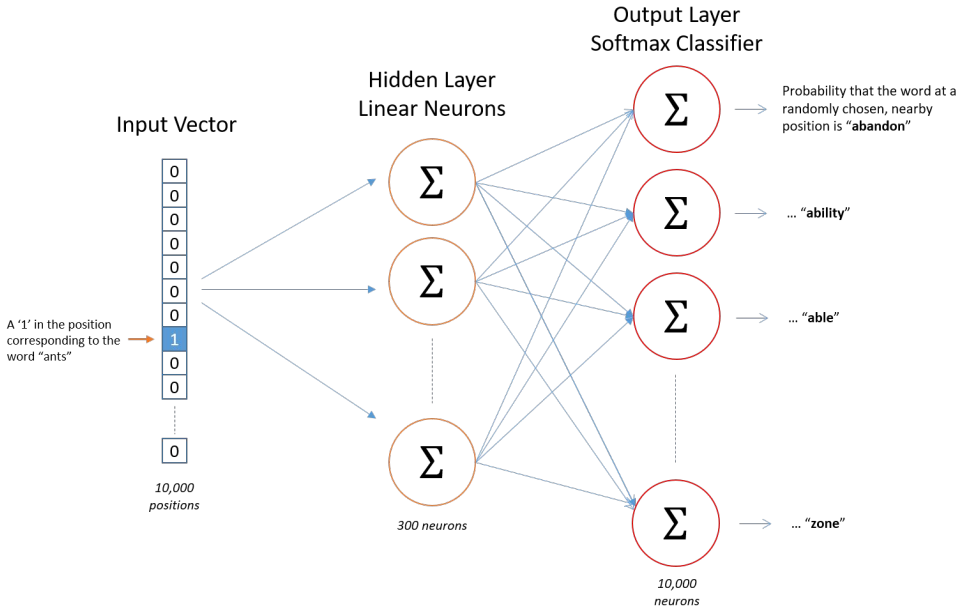


Image:[McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model. Retrieved from http://www.mccormickml.com]

Word2Vec

Training the **Skip-gram** model

1. Data
2. Architecture
3. The Hidden Layer

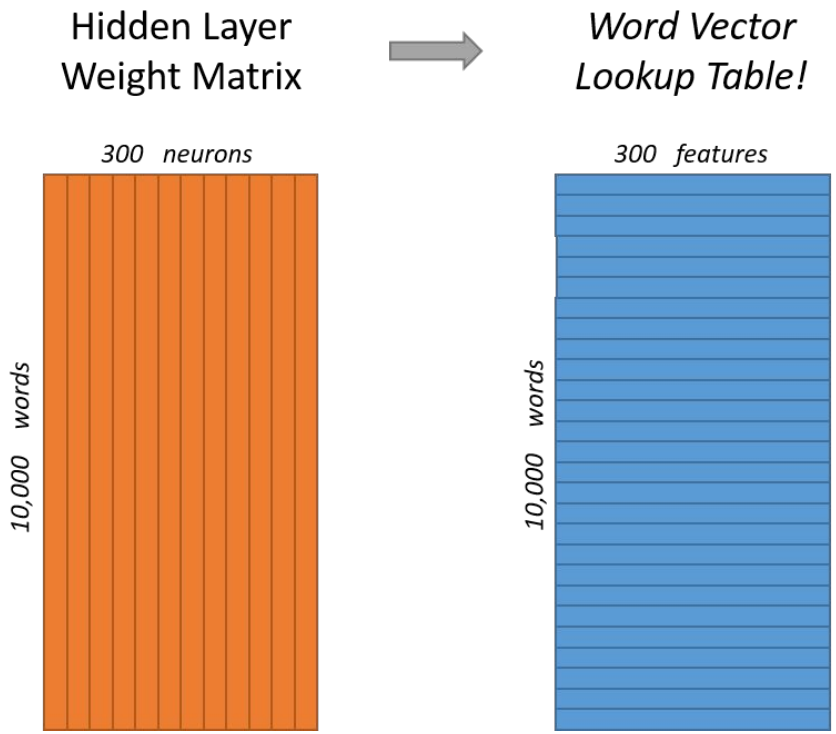


Image:[McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model. Retrieved from http://www.mccormickml.com]

Word2Vec

Training the **Skip-gram** model

1. Data
2. Architecture
3. The Hidden Layer
4. The output Layer

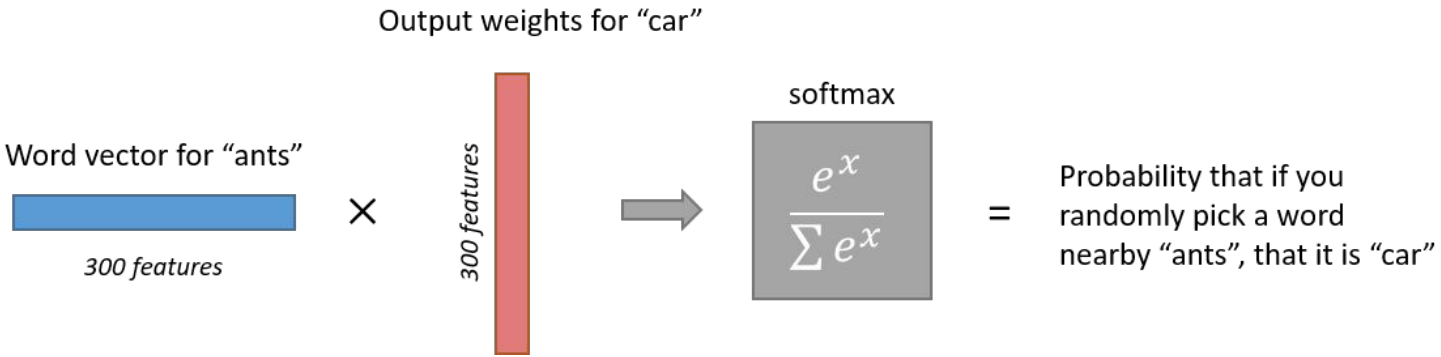


Image:[McCormick, C. (2016, April 19). Word2Vec Tutorial - The Skip-Gram Model. Retrieved from http://www.mccormickml.com]