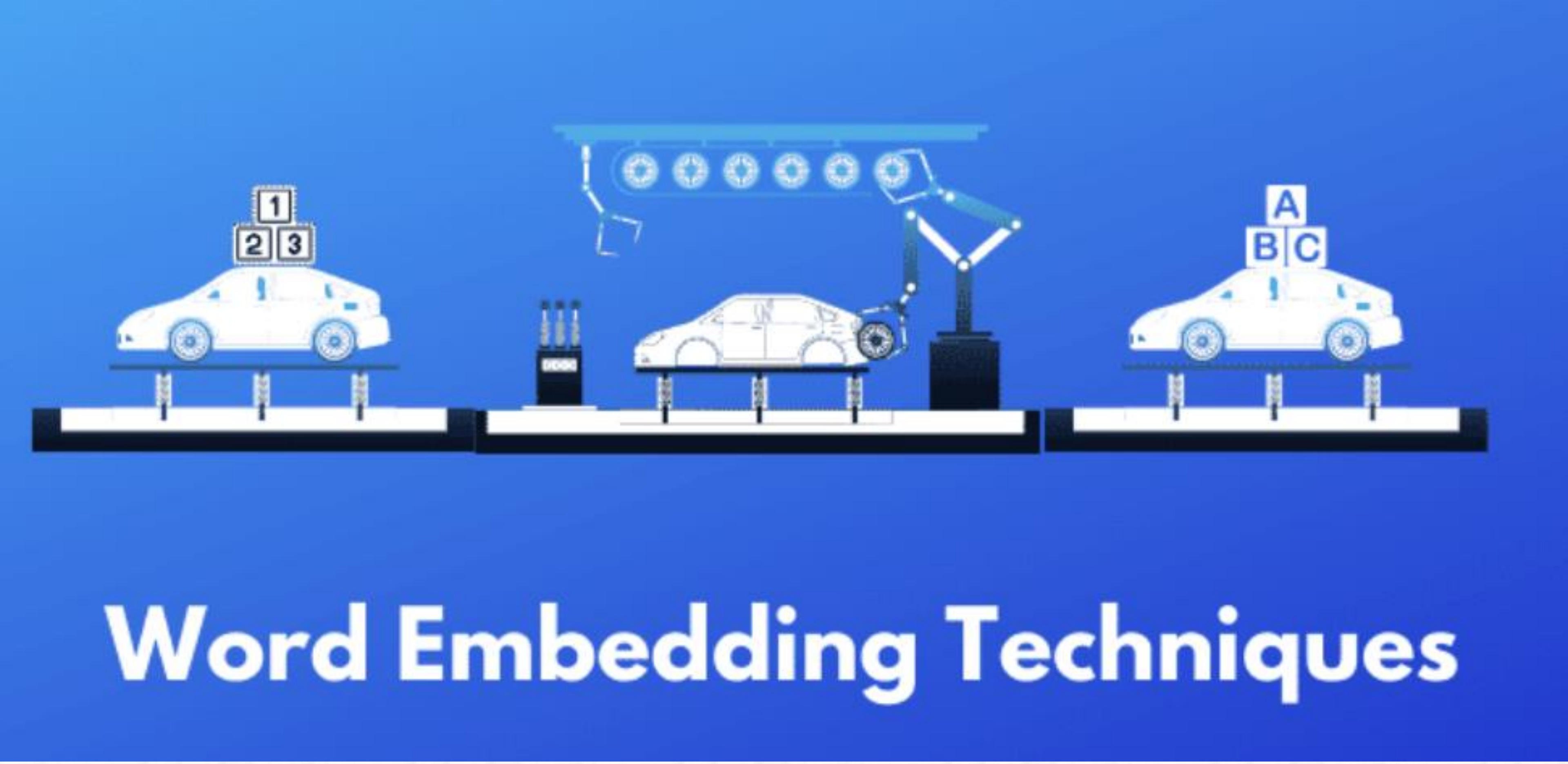
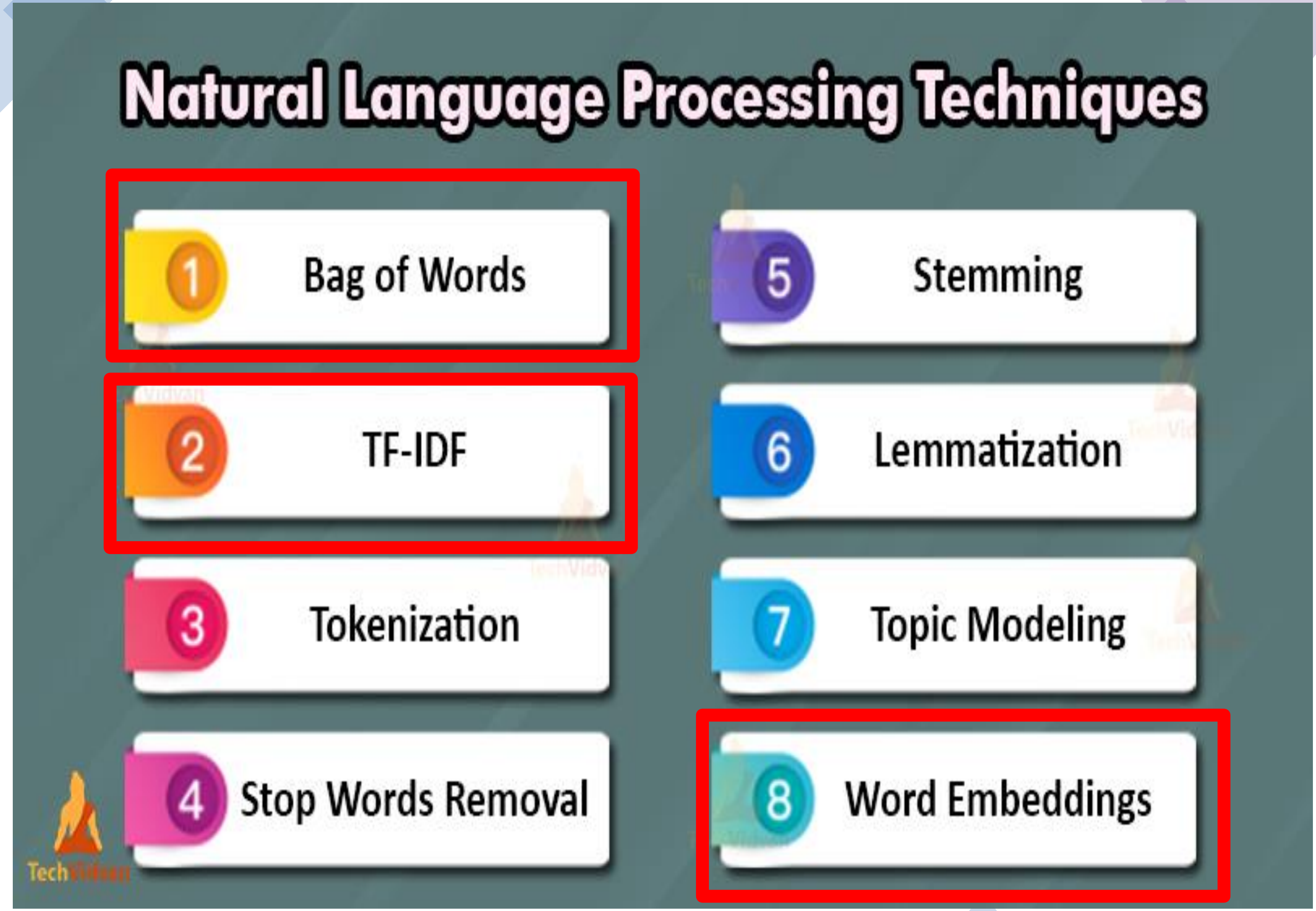


Word embedding

Instructor : Dr. Hanaa Bayomi Ali
Mail : h.mobarz @ fci-cu.edu.eg



Natural Language Processing Techniques

1 Bag of Words

2 TF-IDF

3 Tokenization

4 Stop Words Removal

5 Stemming

6 Lemmatization

7 Topic Modeling

8 Word Embeddings

Introduction

- **BUT, Most important is**

How we **represent words**

as input for all the NLP tasks.

Introduction

- **BUT, Most important is**

How we **represent** ***meaning of*** **words**

as input for all the NLP tasks.

Vector Embedding of Words

- A word is represented as a **vector**.
- Word embeddings depend on a notion of ***word similarity***.
  - Similarity is computed using cosine.
- A very useful definition is paradigmatic similarity:
  - ***Similar words*** occur in ***similar contexts.*** They are ***exchangeable.***
  - Yesterday called a press conference.

    POTUS

    The President

    Trump
  - "POTUS: President of the United States."

Vector Embedding of Words

Traditional Method - Bag of Words Model (no cotext)

- 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 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
- 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
- Context information is not utilized.
- One-hot , BOW, and TF-IDF

Word Embeddings (with context)

- Stores each word in as a point in space, where it is represented by a dense 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.
- Word 2 vec ,fasttext , GLOVE, Elmo

What is a Bag-of-Words?

- A bag-of-words model, or BoW for short, is a **way of extracting features from text** for use in modeling.

- The approach is very simple and flexible, and can be used in many ways for extracting features from documents.

- A bag-of-words is a representation of text that describes the occurrence of words within a document. It involves two things:

    1. A vocabulary of known words.
    2. A measure of the presence of known words.

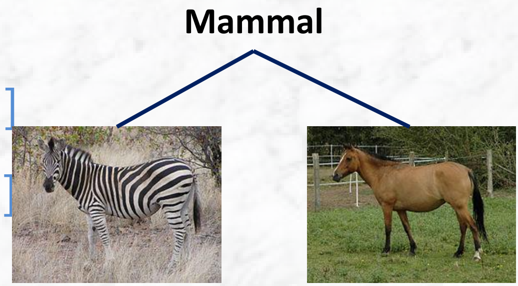
- It is called a "*bag*" of words, because any information about the order or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document.

One-Hot Vector

- Approach 1: Use ***One-Hot Vector***
- Each word in the vocabulary is represented by one bit position in a **HUGE** vector.
- The encoding of a given word is simply the vector in which the corresponding element is set to one, and all other elements are zero.

Horse = [0 0 0 0 1 0 0 0 0 0 0 0 0 0]

Zebra = [0 0 0 0 0 0 0 0 0 0 0 1 0 0]

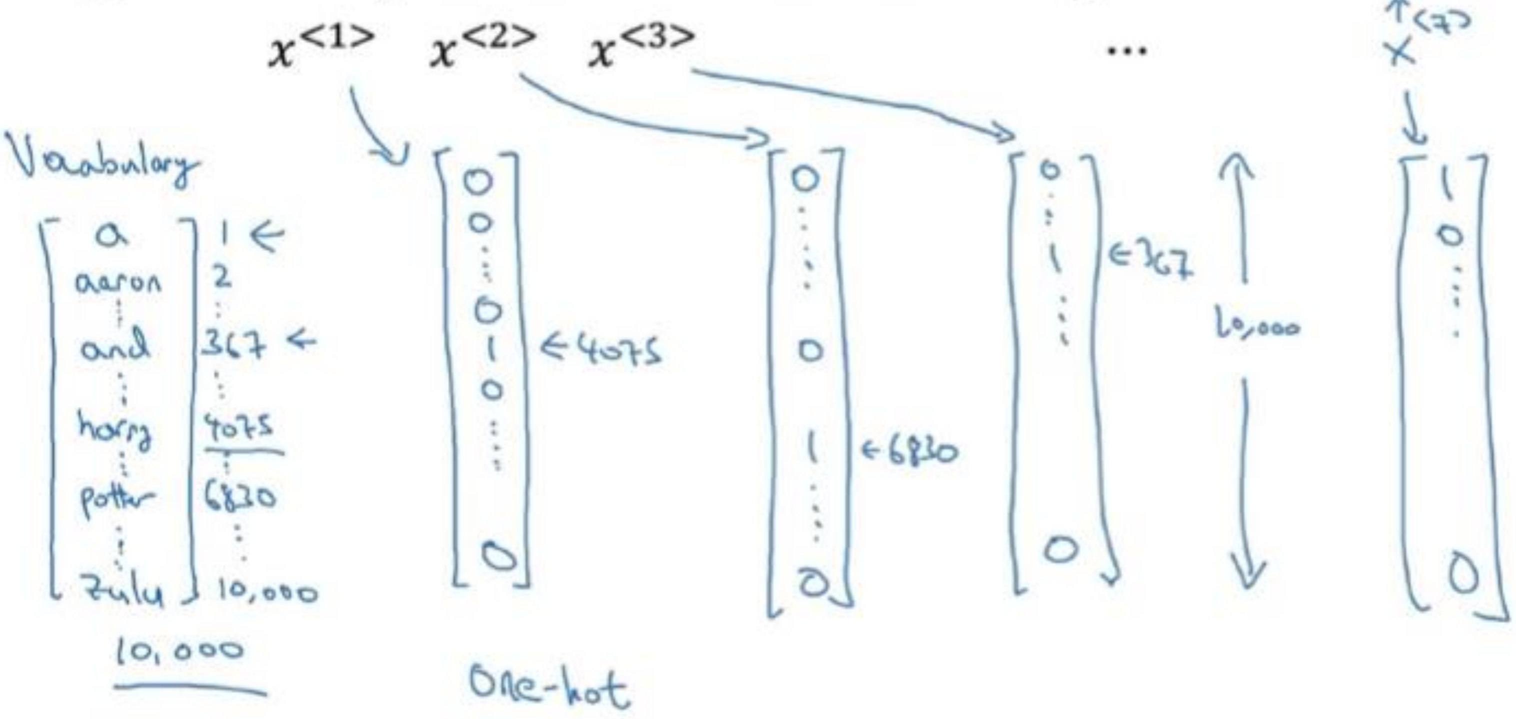


- One-Hot Vector limitations:
  - Horse ∩ Zebra = ??
  - There’s no meaningful comparison
  - Context information is not utilized

Representing words

x: Harry Potter and Hermione Granger invented a new spell.

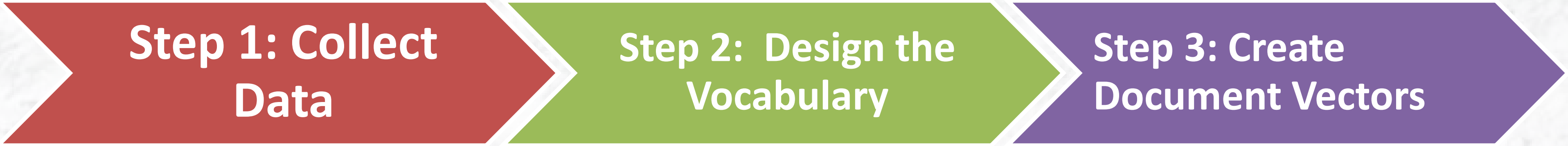
$x^{<1>}$ $x^{<2>}$ $x^{<3>}$ … $x^{<9>}$



Andrew N

Document representation

- Approach 2: Use ***Document representation***



Step 1: Collect Data

Step 2: Design the Vocabulary

Step 3: Create Document Vectors

Consider the following corpus that contains four documents

D1: It was the best of times,

D2: ***it was the worst of times,***

D3: ***it was the age of wisdom,***

D4: ***it was the age of foolishness,***

1. **“it”**
2. **“was”**
3. **“the”**
4. **“best”**
5. **“of”**
6. **“times”**
7. **“worst”**
8. **“age”**
9. **“wisdom”**
10. **“foolishness”**

That is a vocabulary of 10 words from a corpus containing 24 words.

	D1	D2	D3	D4
it				
was				
the				
best				
of				
Times				
Worst				
age				
wisdom				
foolishness				

Document representation

- Approach 2: Use ***Document representation***

1. Binary Vector

- *D1: It was the best best of times,*
  D2: *it was the worst of times,*
  D3: *it was the age of wisdom,*
  D4: *it was the age of foolishness,*

	it	was	the	best	of	Times	Worst	age	wisdom	foolishness
D1	1	1	1	1	1	1	0	0	0	0
D2	1	1	1	0	1	1	1	0	0	0
D3	1	1	1	0	1	0	0	1	1	0
D4	1	1	1	0	1	0	0	1	0	1

Document representation

- Approach 2: Use ***Document representation***

2. Scoring Words

➢ **Counts.** Count the number of times each word appears in a document.

➢ **Frequencies**. Calculate the frequency that each word appears in a document out of all the words in the document.

D1: It was the best best of times,
D2: ***it was the worst of times,***
D3: ***it was the age of wisdom,***
D4: ***it was the age of foolishness,***

	it	was	the	best	of	Times	Worst	age	wisdom	foolishness
D1	1/7	1/7	1/7	2/7	1/7	1/7	0	0	0	0
D2	1/6	1/6	1/6	0	1/6	1/6	1/6	0	0	0
D3	1/6	1/6	1/6	0	1/6	0	0	1/6	1/6	0
D4	1/6	1/6	1/6	0	1/6	0	0	1/6	0	1/6

Document representation

- Approach 2: Use ***Document representation***

3. Term Frequency Inverse Document Frequency (TFIDF)

A problem with scoring word frequency is that highly frequent words start to dominate in the document (e.g. larger score), but may not contain as much "informational content" to the model as rarer but perhaps domain specific words.

- Two-fold heuristics based on frequency
  - TF (Term frequency)
    - More frequent ***within*** a document → more relevant to semantics
    - e.g., "query" vs. "commercial"
  - IDF (Inverse document frequency)
    - Less frequent ***among*** documents → more discriminative
    - e.g. "algebra" vs. "science"

Term Weights: Term Frequency

- More frequent terms in a document are more important, i.e. more indicative of the topic.

  f_{ij} = frequency of term i in document j

- May want to normalize *term frequency* (*tf*) by dividing by the frequency of the most common term in the document:

  $tf_{ij} = f_{ij} / max_i\{f_{ij}\}$

Term Weights: Inverse Document Frequency

- Terms that appear in many *different* documents are *less* indicative of overall topic.

  df_i = document frequency of term i

  = number of documents containing term i

  idf_i = inverse document frequency of term i,

  $= \log_{10} (N/ df_i)$

  (N: total number of documents)

- An indication of a term’s *discrimination* power.

TF-IDF Weighting

- A typical combined term importance indicator is *tf-idf weighting*:

$$w_{ij} = tf_{ij}\, idf_i = tf_{ij} \log_2 (N / df_i)$$

- A term occurring frequently in the document but rarely in the rest of the collection is given high weight.
- Many other ways of determining term weights have been proposed.
- Experimentally, *tf-idf* has been found to work well.

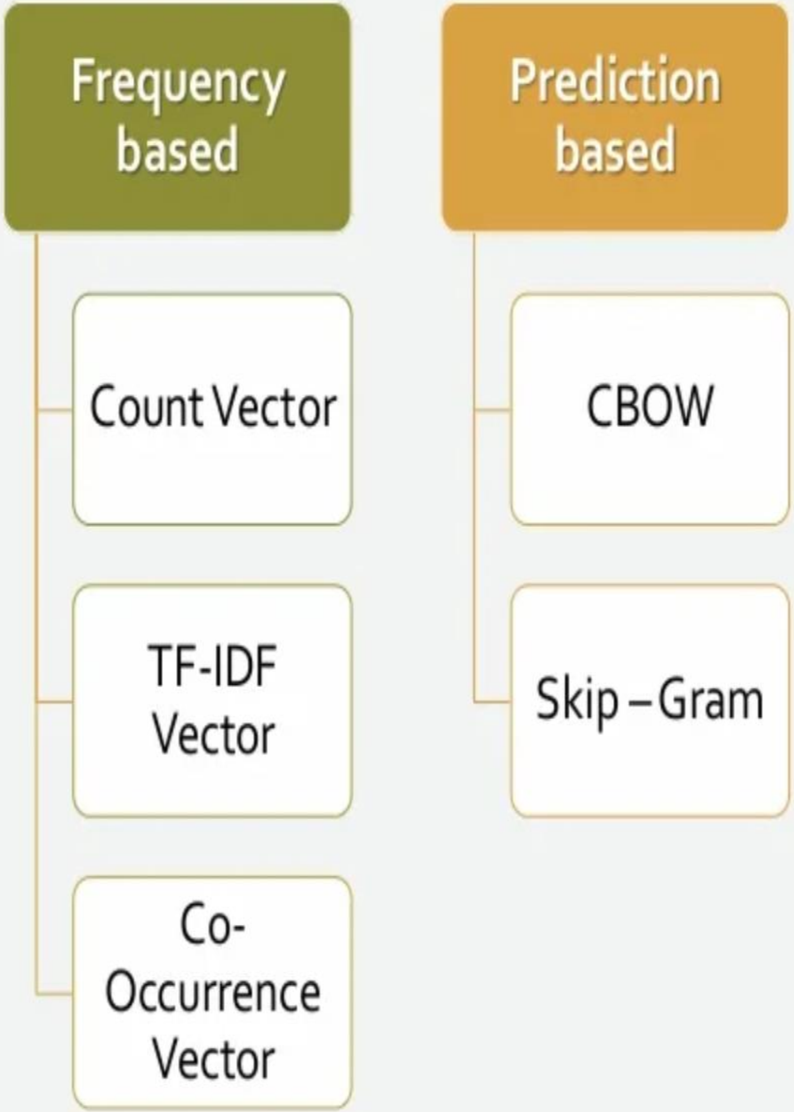
Limitation of Bag-of-Words?

Vocabulary: The vocabulary requires careful design, most specifically in order to manage the size, which impacts the sparsity of the document representations.

Sparsity: Sparse representations are harder to model both for computational reasons (space and time complexity) and also for information reasons, where the challenge is for the models to use so little information in such a large representational space.

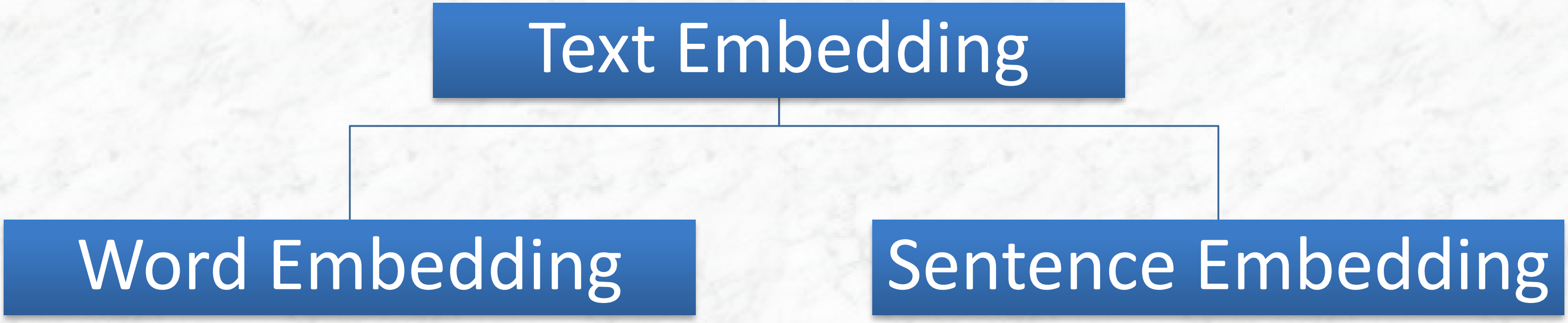
Meaning: Discarding word order ignores the context, and in turn meaning of words in the document (semantics). Context and meaning can offer a lot to the model, that if modeled could tell the difference between the same words differently arranged (“this is interesting” vs “is this interesting”), synonyms (“old bike” vs “used bike”), and much more.

Types of Word Embeddings



- Frequency based
  - Count Vector
  - TF-IDF Vector
  - Co-Occurrence Vector
- Prediction based
  - CBOW
  - Skip – Gram

Text Representation



Text Embedding

Word Embedding

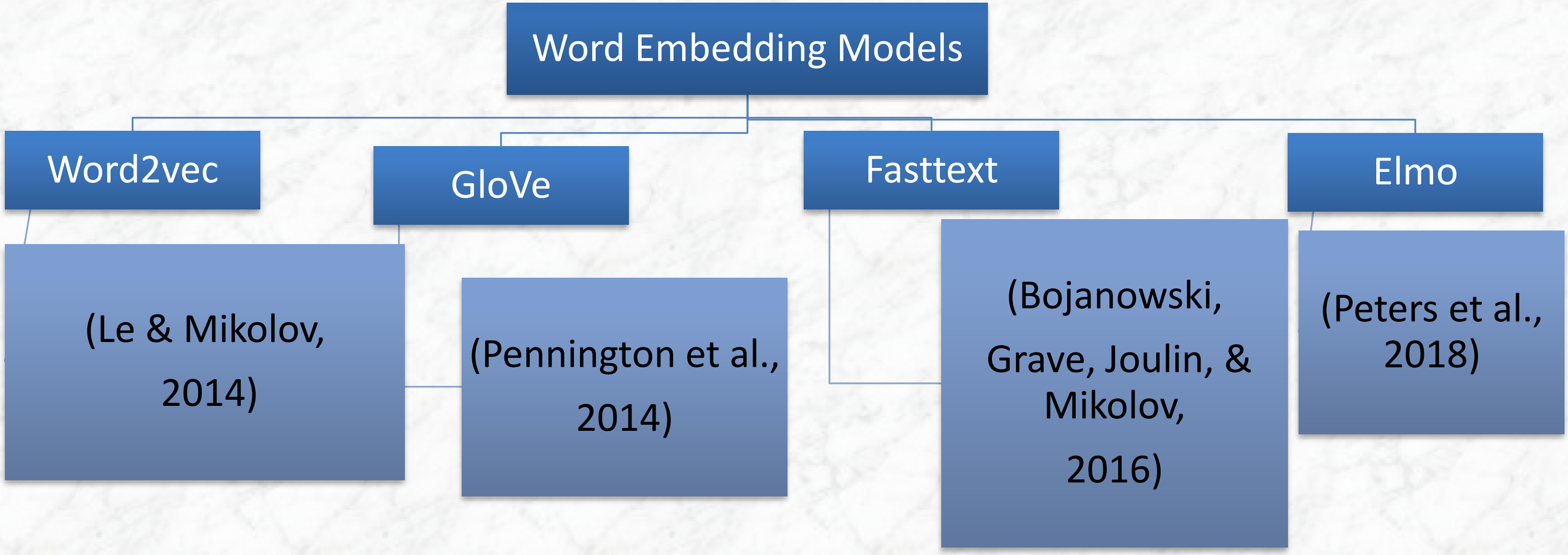
Sentence Embedding

- ML model is trained on sequences of words from large corpora to generate **fixed-length vector representation for each word**

- ML model is trained on sequences of sentences to generate **fixed-length vector representation for variable length sentences**

Text Representation

- Word embeddings
  - Techniques for learning distributional vector representations for words existing in low dimensional space



Word Embedding Models

Word2vec

(Le & Mikolov, 2014)

GloVe

(Pennington et al., 2014)

Fasttext

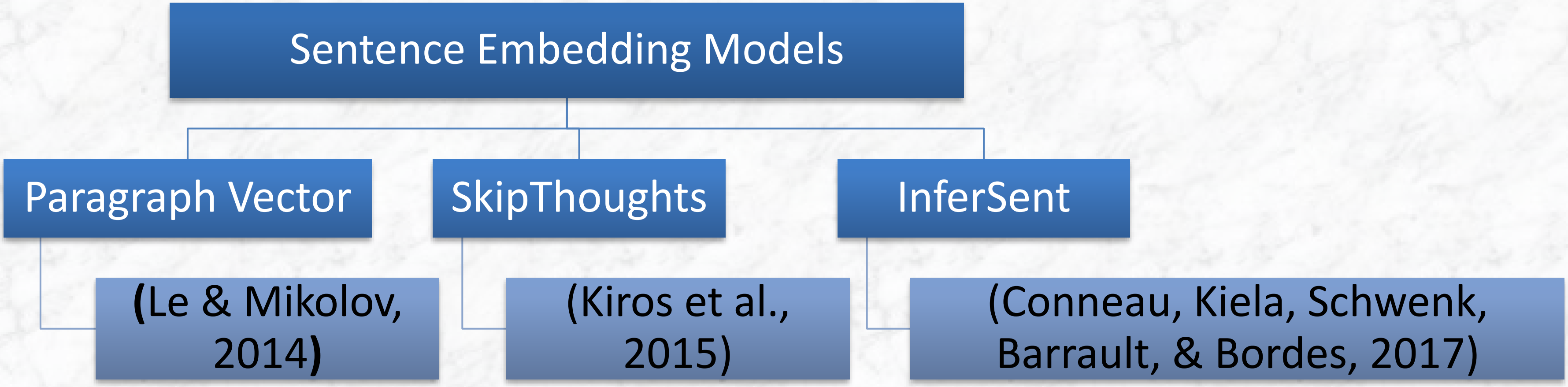
(Bojanowski, Grave, Joulin, & Mikolov, 2016)

Elmo

(Peters et al., 2018)

Text Representation (4)

- Sentence Embeddings
  - Techniques for generating numeric fixed-length vector representations for variable length pieces of texts
  - Used to embed sequences from variable lengths ranging from a phrase, a sentence, a paragraph, or even a document to fixed length vector.


Sentence Embedding Models
Paragraph Vector
SkipThoughts
InferSent
(Le & Mikolov, 2014**)**
(Kiros et al., 2015)
(Conneau, Kiela, Schwenk, Barrault, & Bordes, 2017)

Word2vec

Word2vec

- Represent each word with a low-dimensional vector
- Word similarity = Vector Similarity
- Key idea: ***Don't Count , Predict***
  - Predict surrounding words of every word
- Faster and can easily incorporate a new sentence/document or add a word to the vocabulary
- Word meaning and relationships between words are encoded spatially (context)
- Introduced by Tomas Mikolov 2013 (then at Google 2014)
- Neural Network approach
- It is **NOT** Deep Learning

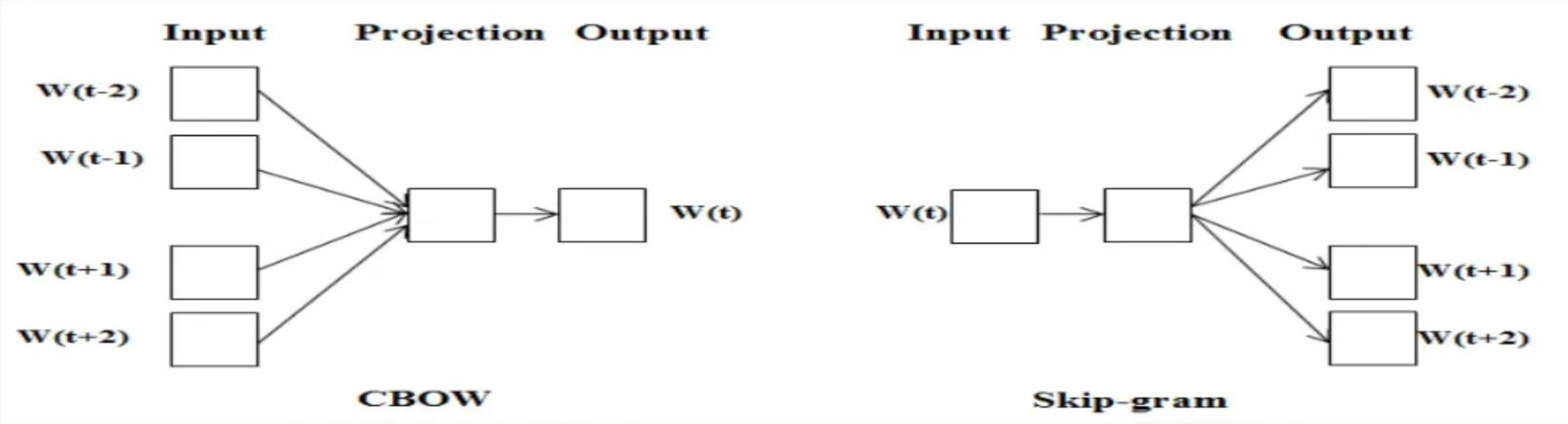
Word2vec

- Two architectures are proposed:
  - *Continuous* ***Bag-of-Words*** model
    - ***predicting the word given its context***
  - *Continuous* ***Skip-gram*** model
    - ***predicting the context given a word***

CBOW

(Continuous Bag of words)

Skip-gram



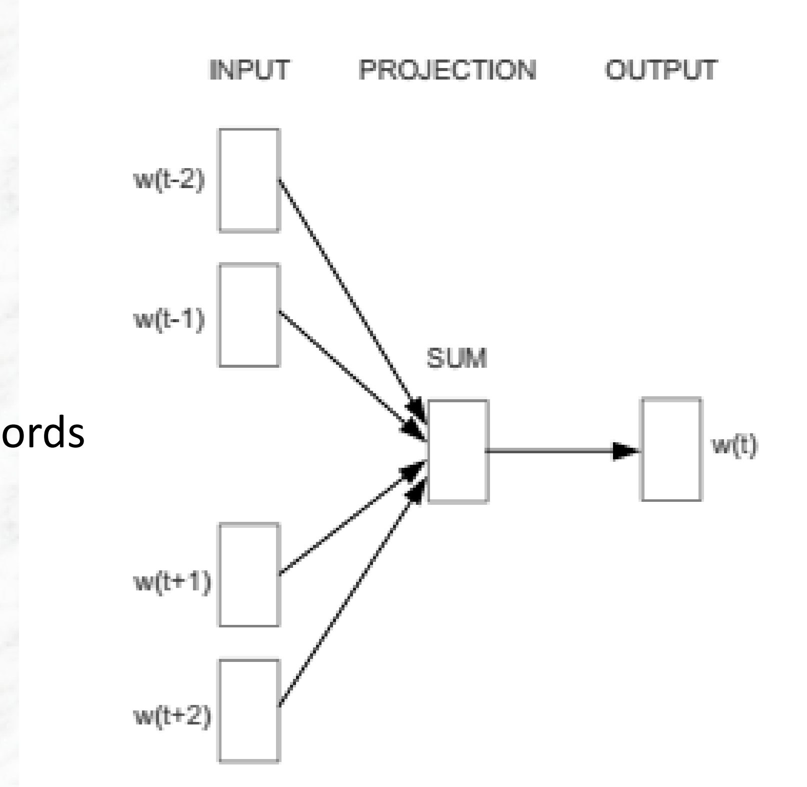
Syntactic relation

Semantic relation

CBOW	Skip-gram
•In this context or neighbor words are input and output is the center word. •Works good with large datasets. •Better representation for frequent words than rarer.	•In this input is centre word and output is context words (neighbour words). •Works well with small datasets. •Skip-gram identifies rarer words better.

Word2vec: CBOW

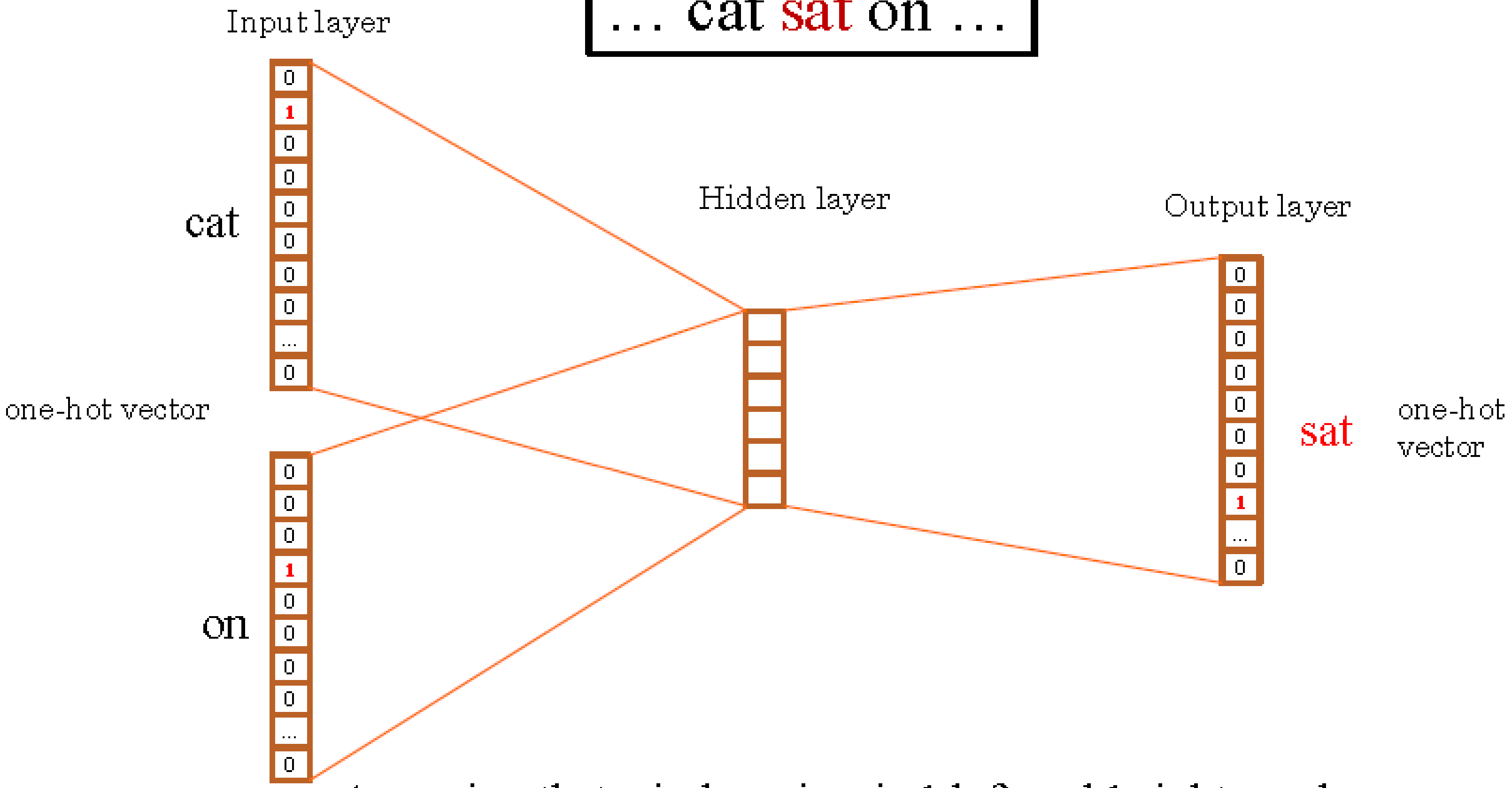
Context words



Center word

Distributed Representations of Words and Phrases and their Compositionality
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, NIPS 2013

… cat sat on …

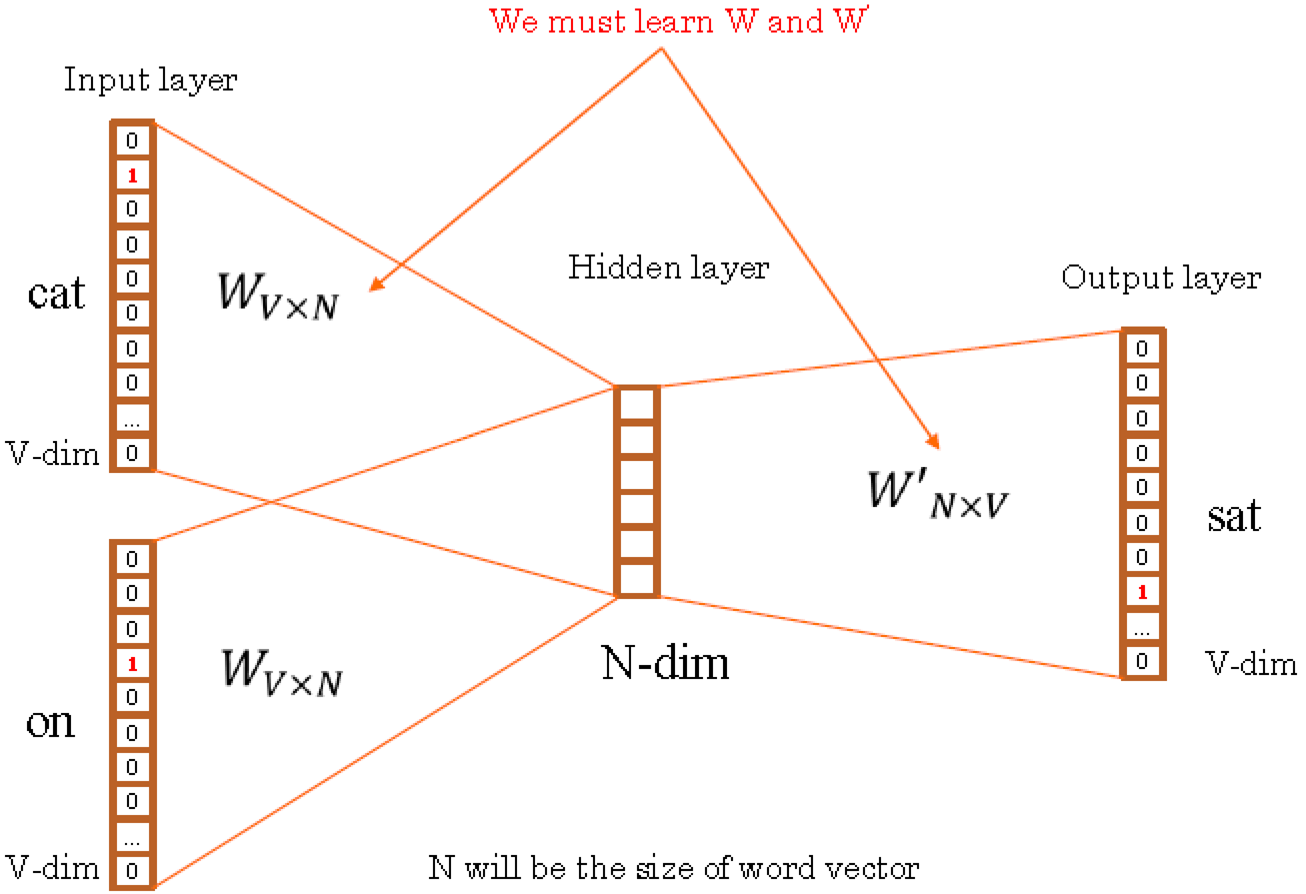


Assuming that window size is 1 left and 1 right , and:

Index of **cat** in vocabulary is 2

index of **on** in vocabulary is 4

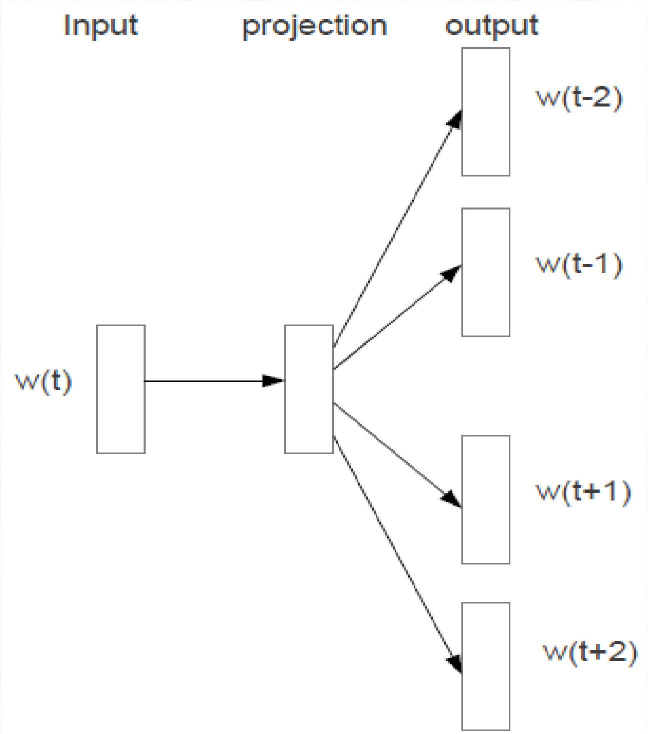
Index of **sat** in vocabulary is 8


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

Word2vec: Skip-gram

The pairs of center word/context word are called **"skip-grams."** Typical distances are 3-5 word positions. Skip-gram model:

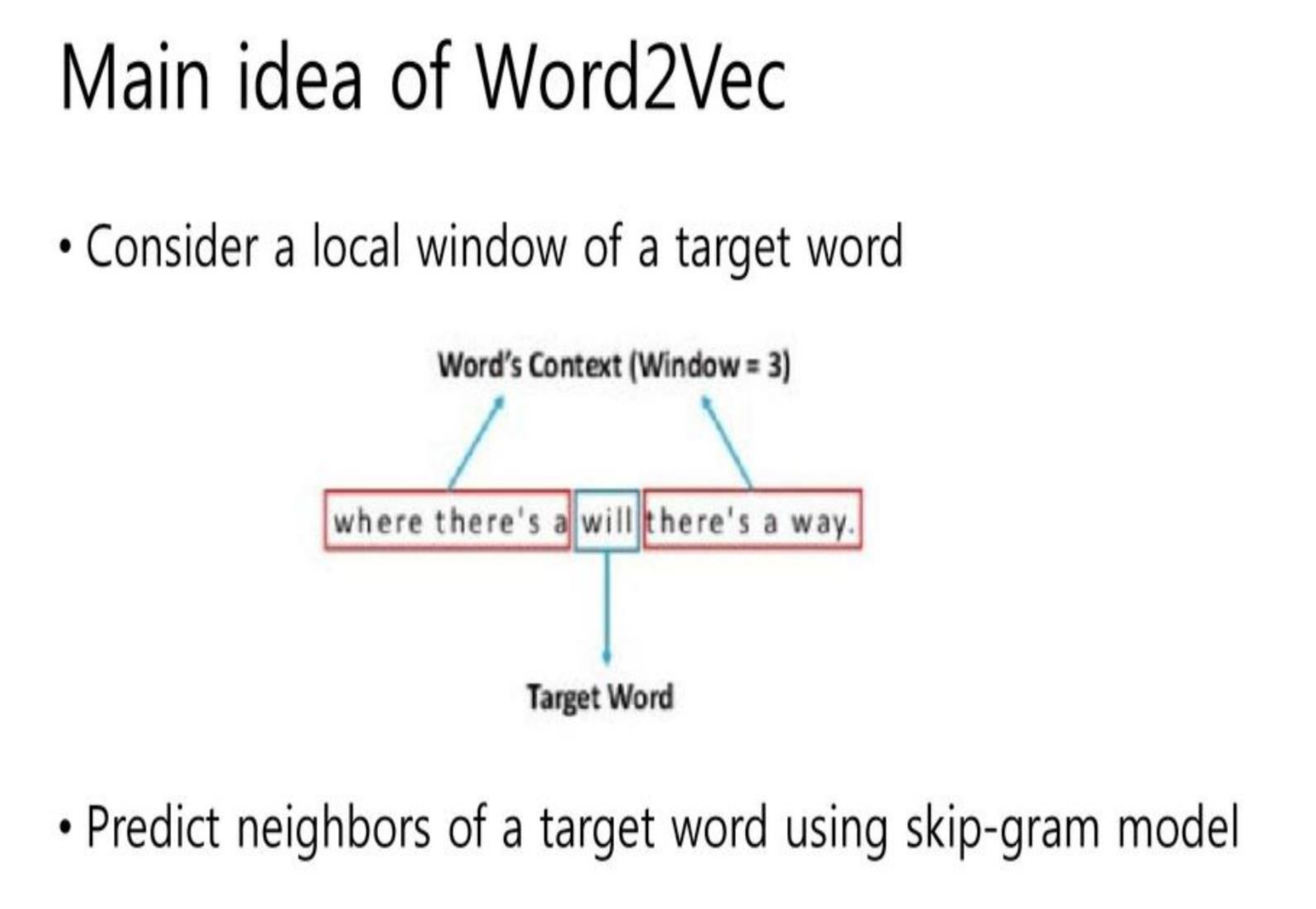
Center word



Context words

Distributed Representations of Words and Phrases and their Compositionality
Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, Jeffrey Dean, NIPS 2013

Word2vec: Skip-gram

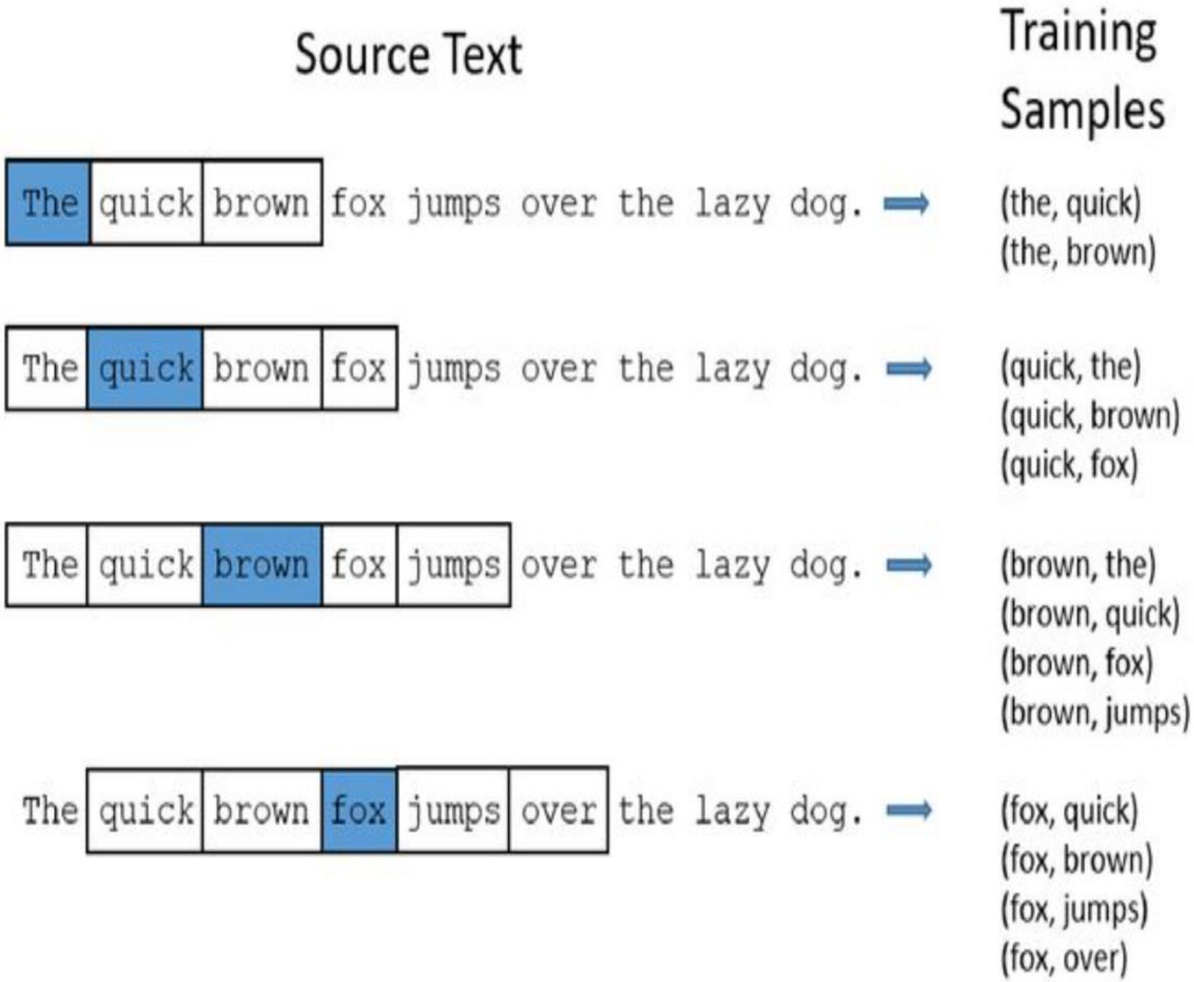


Main idea of Word2Vec

- Consider a local window of a target word

- Predict neighbors of a target word using skip-gram model

Collection of Training samples with a window of size 2



Source Text	Training Samples
The quick brown fox jumps over the lazy dog.	(the, quick) (the, brown)
The **quick** brown fox jumps over the lazy dog.	(quick, the) (quick, brown) (quick, fox)
The quick **brown** fox jumps over the lazy dog.	(brown, the) (brown, quick) (brown, fox) (brown, jumps)
The quick brown **fox** jumps over the lazy dog.	(fox, quick) (fox, brown) (fox, jumps) (fox, over)

Word Embedding

- Build a vocabulary of words from training documents
  - E.g.) a vocabulary of 10,000 unique words

- Represent an input word like ″ants″ as a one-hot vector
  - This vector will have 10,000 components (one for every word in our vocabulary)
  - This vector will have “1” in the position corresponding to the word, say “ants”, and 0s in all of the other positions.

Word Embedding

- No activation function for hidden layer neurons, but output neurons use softmax

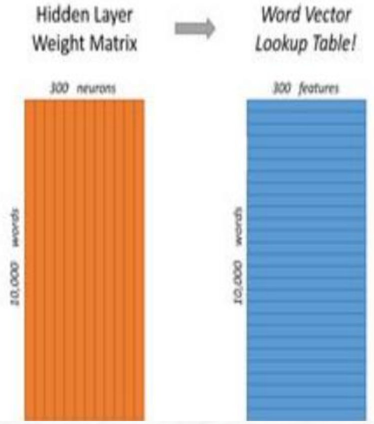
- When *training* the network with word pairs, the input is a one-hot vector representing the input word and output *is also a one-hot vector* representing the output word.

- when evaluating the trained network on an input word, the output vector will actually be a probability distribution (i.e., a bunch of floating point values, *not* a one-hot vector).

Word Embedding

- Hidden layer is represented by a weight matrix with 10,000 rows (one for every word in our vocabulary) and 300 columns (one for every hidden neuron)
  - 300 features for Google News Dataset

- *rows* of this weight matrix are actually what will be our word vectors!



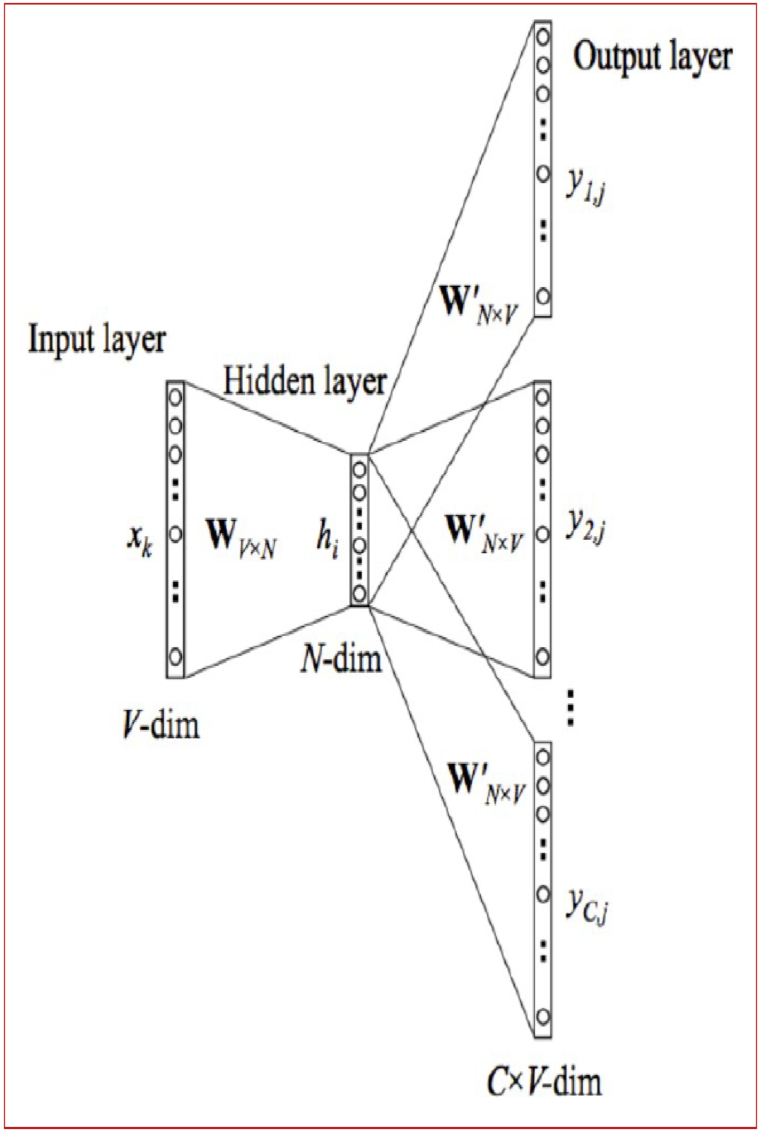
Word Embedding

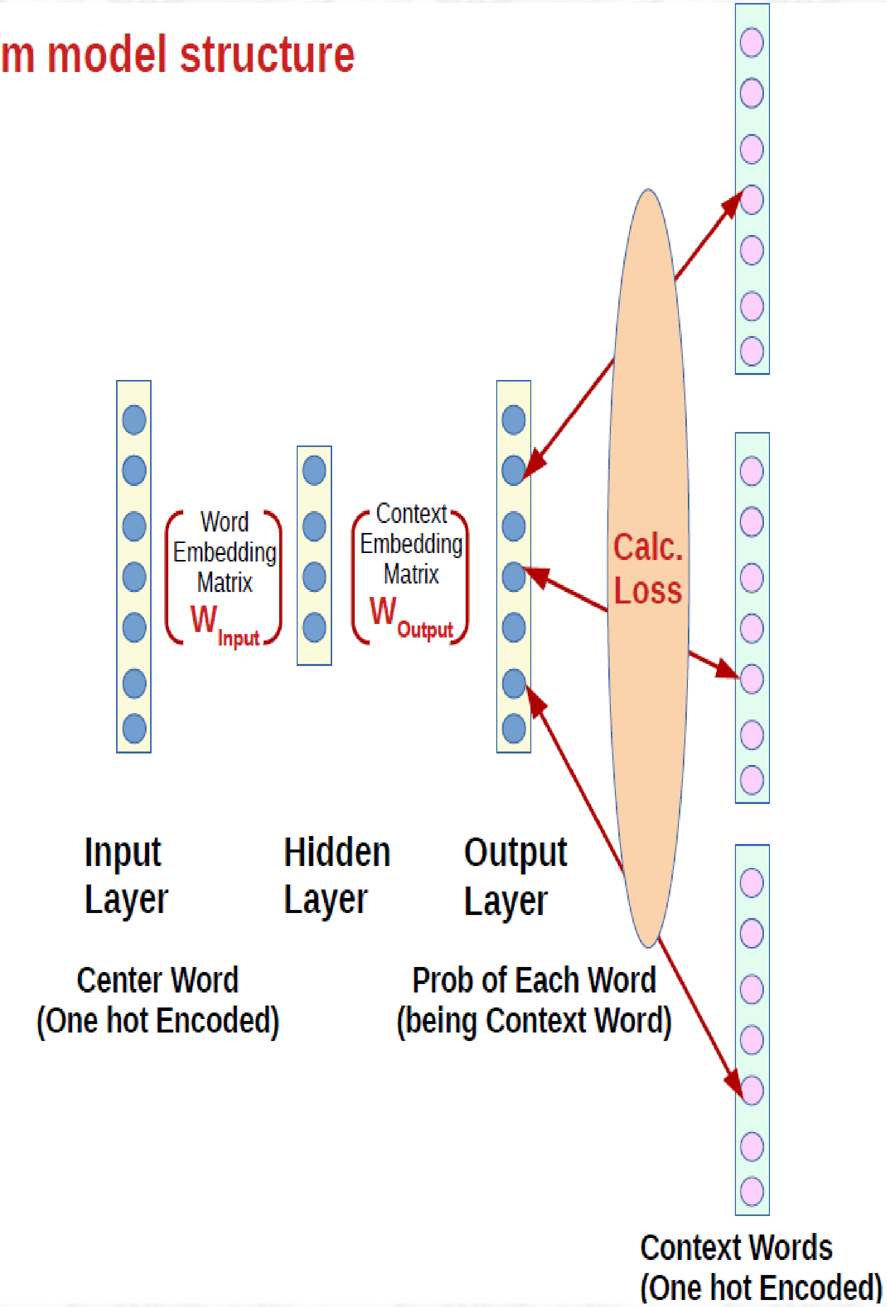
- If we multiply a 1 x 10,000 one-hot vector by a 10,000 x 300 matrix, it will effectively just *select* the matrix row corresponding to the "1"

$$\begin{bmatrix} 0 & 0 & 0 & 1 & 0 \end{bmatrix} \times \begin{bmatrix} 17 & 24 & 1 \\ 23 & 5 & 7 \\ 4 & 6 & 13 \\ 10 & 12 & 19 \\ 11 & 18 & 25 \end{bmatrix} = \begin{bmatrix} 10 & 12 & 19 \end{bmatrix}$$

- Hidden layer is really just operating as a lookup table !!
- The output of the hidden layer is just the "word vector" for the input word

Skip-Gram model structure





Word Embedding Matrix W_{Input}

Context Embedding Matrix W_{Output}

Calc. Loss

Input Layer

Hidden Layer

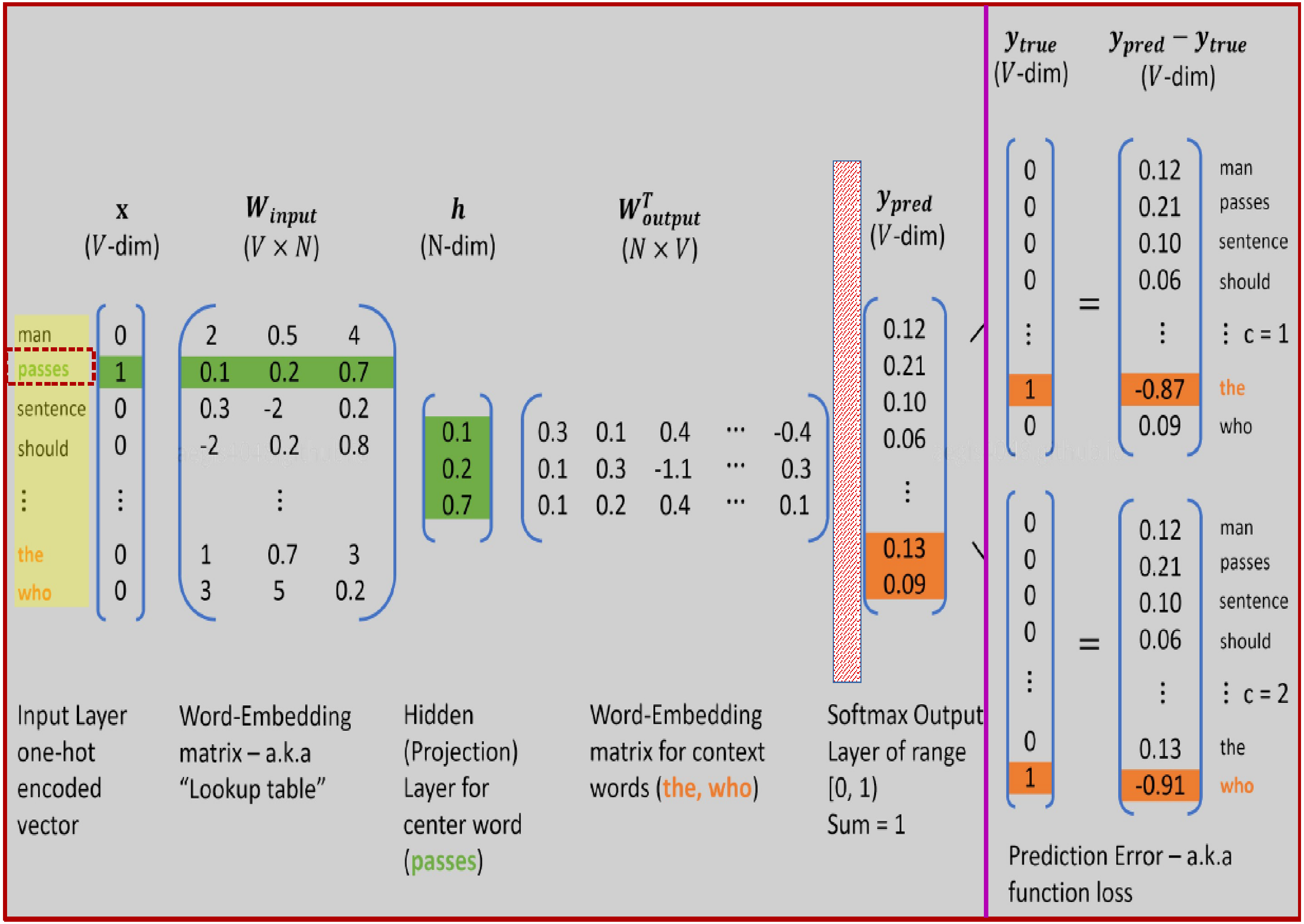
Output Layer

Center Word (One hot Encoded)

Prob of Each Word (being Context Word)

Context Words (One hot Encoded)

Skip-Gram model structure, Current center word is "passes"



$\mathbf{x}$ (V-dim) $\quad W_{input}$ ($V \times N$) $\quad \boldsymbol{h}$ (N-dim) $\quad W_{output}^{T}$ ($N \times V$) $\quad y_{pred}$ (V-dim)

$$\begin{array}{l} \text{man} \\ \text{passes} \\ \text{sentence} \\ \text{should} \\ \vdots \\ \text{the} \\ \text{who} \end{array} \begin{pmatrix} 0 \\ 1 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} \quad \begin{pmatrix} 2 & 0.5 & 4 \\ 0.1 & 0.2 & 0.7 \\ 0.3 & -2 & 0.2 \\ -2 & 0.2 & 0.8 \\ & \vdots & \\ 1 & 0.7 & 3 \\ 3 & 5 & 0.2 \end{pmatrix} \quad \begin{pmatrix} 0.1 \\ 0.2 \\ 0.7 \end{pmatrix} \quad \begin{pmatrix} 0.3 & 0.1 & 0.4 & \cdots & -0.4 \\ 0.1 & 0.3 & -1.1 & \cdots & 0.3 \\ 0.1 & 0.2 & 0.4 & \cdots & 0.1 \end{pmatrix} \quad \begin{pmatrix} 0.12 \\ 0.21 \\ 0.10 \\ 0.06 \\ \vdots \\ 0.13 \\ 0.09 \end{pmatrix}$$

Input Layer one-hot encoded vector

Word-Embedding matrix – a.k.a "Lookup table"

Hidden (Projection) Layer for center word (passes)

Word-Embedding matrix for context words (the, who)

Softmax Output Layer of range [0, 1) Sum = 1

y_{true} (V-dim) $\quad y_{pred} - y_{true}$ (V-dim)

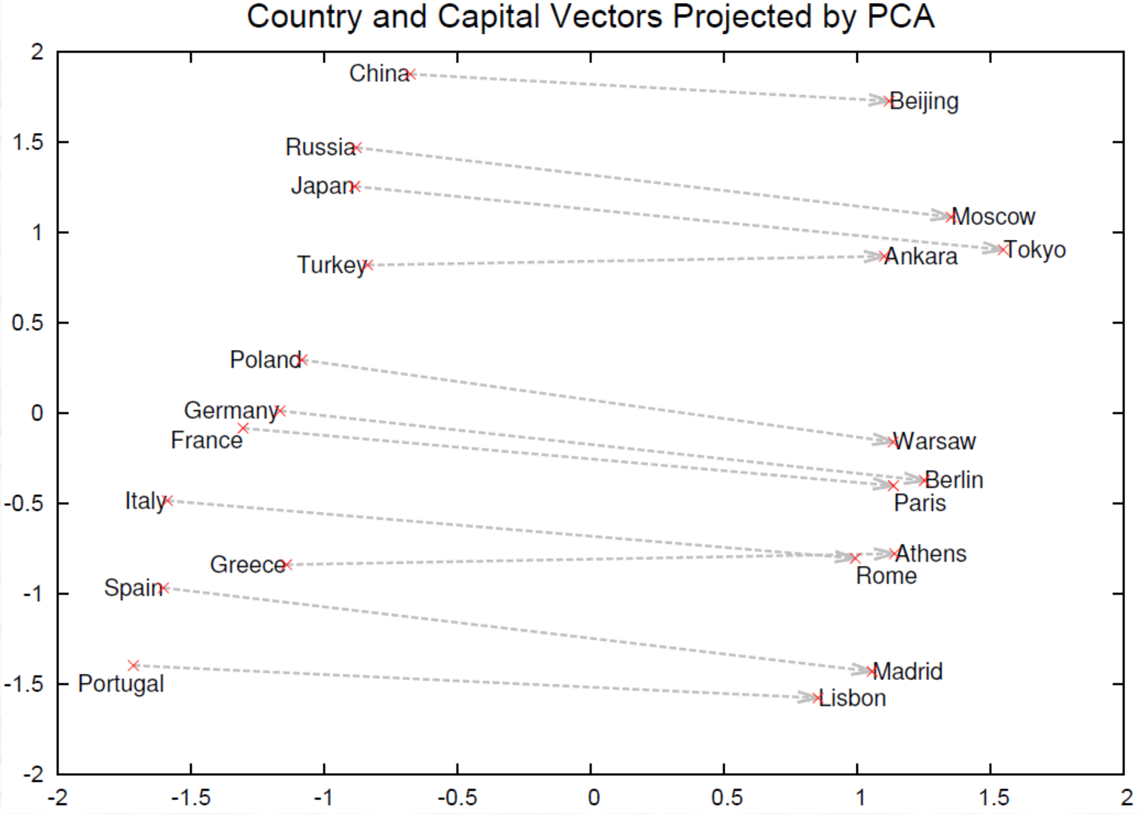
$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 1 \\ 0 \end{pmatrix} = \begin{pmatrix} 0.12 \\ 0.21 \\ 0.10 \\ 0.06 \\ \vdots \\ -0.87 \\ 0.09 \end{pmatrix} \begin{array}{l} \text{man} \\ \text{passes} \\ \text{sentence} \\ \text{should} \\ \vdots \ \ c = 1 \\ \text{the} \\ \text{who} \end{array}$$

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{pmatrix} = \begin{pmatrix} 0.12 \\ 0.21 \\ 0.10 \\ 0.06 \\ \vdots \\ 0.13 \\ -0.91 \end{pmatrix} \begin{array}{l} \text{man} \\ \text{passes} \\ \text{sentence} \\ \text{should} \\ \vdots \ \ c = 2 \\ \text{the} \\ \text{who} \end{array}$$

Prediction Error – a.k.a function loss

Relations Learned by Word embedding

Local contexts capture much more information about relations and properties:

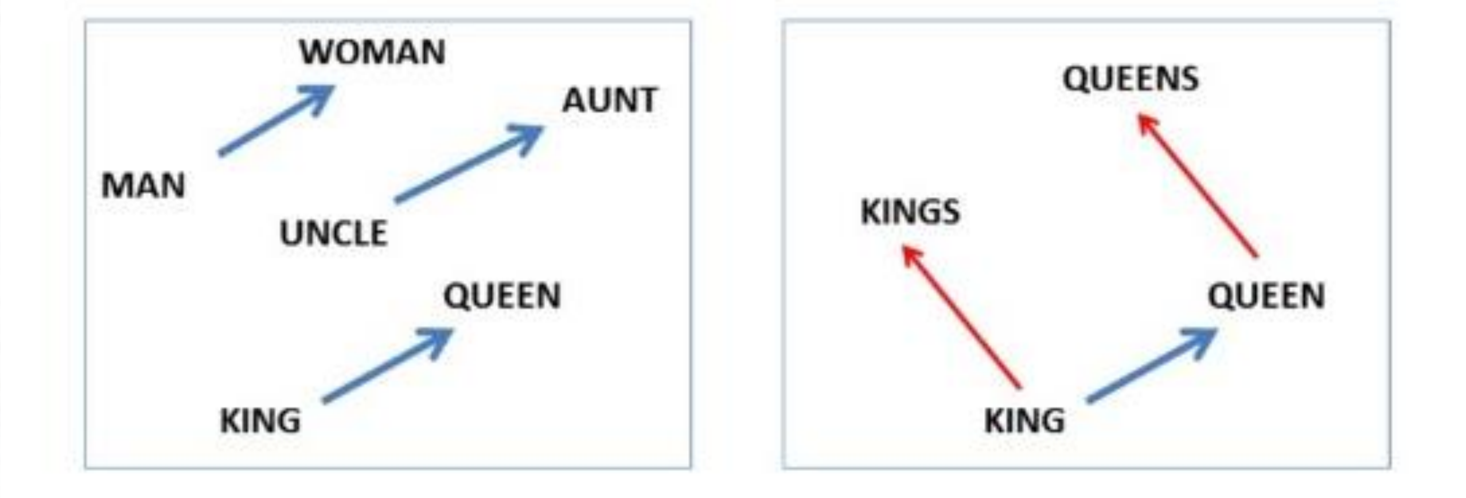


Relations Learned by Word embedding

Algebraic relations:

vec(“woman")−vec(“man") ≃ vec(“aunt")−vec(“uncle")

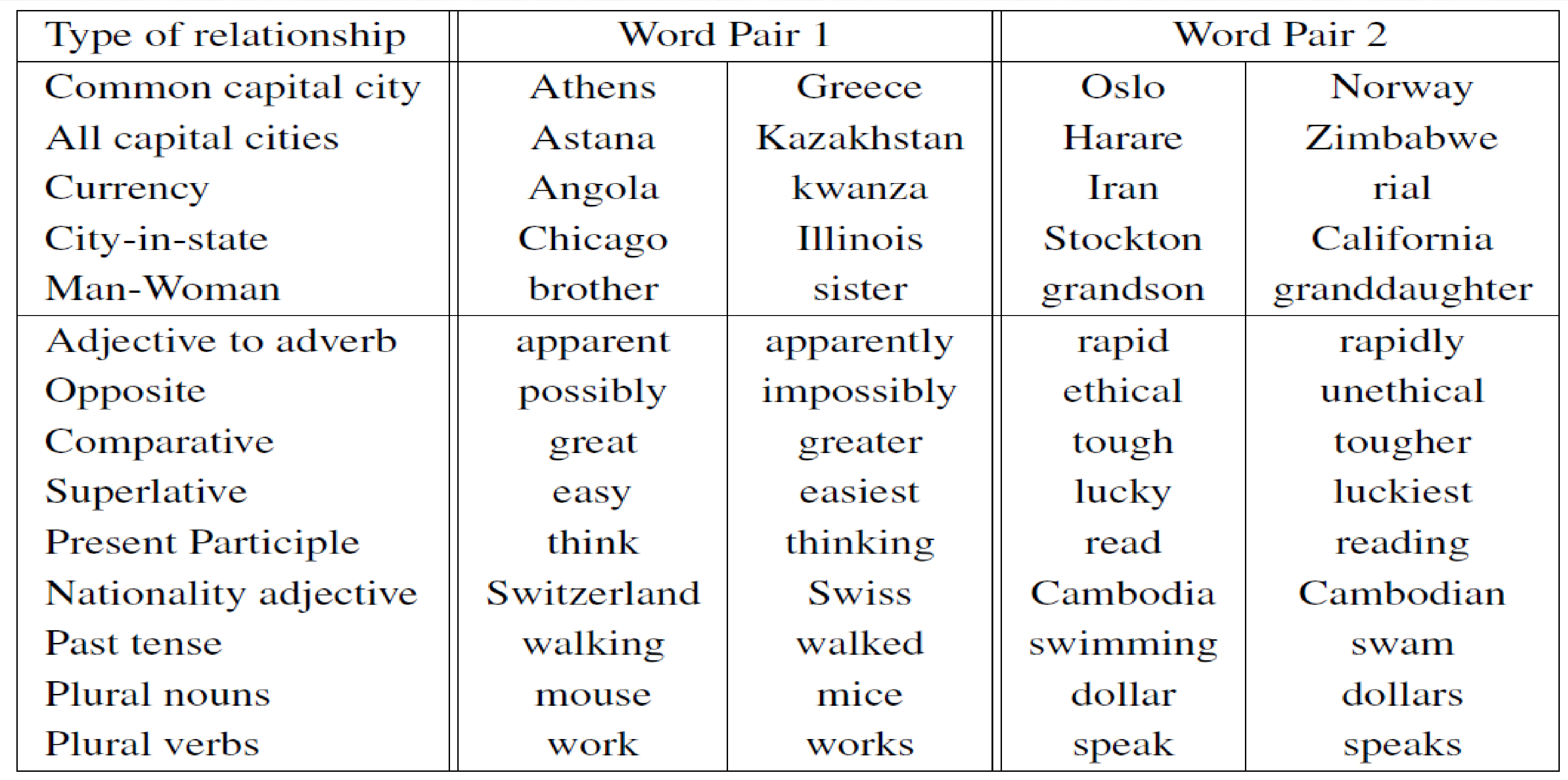
vec (“King”) – vec(“Man”) + vec(“Woman”) ≃ vec (“Queen”)



From “Linguistic Regularities in Continuous Space Word Representations”
Tomas Mikolov , Wen-tau Yih, Geoffrey Zweig, NAACL-HLT 2013

Relations Learned by Word embedding

Word2vec model computed from **6 billion word** corpus of news articles

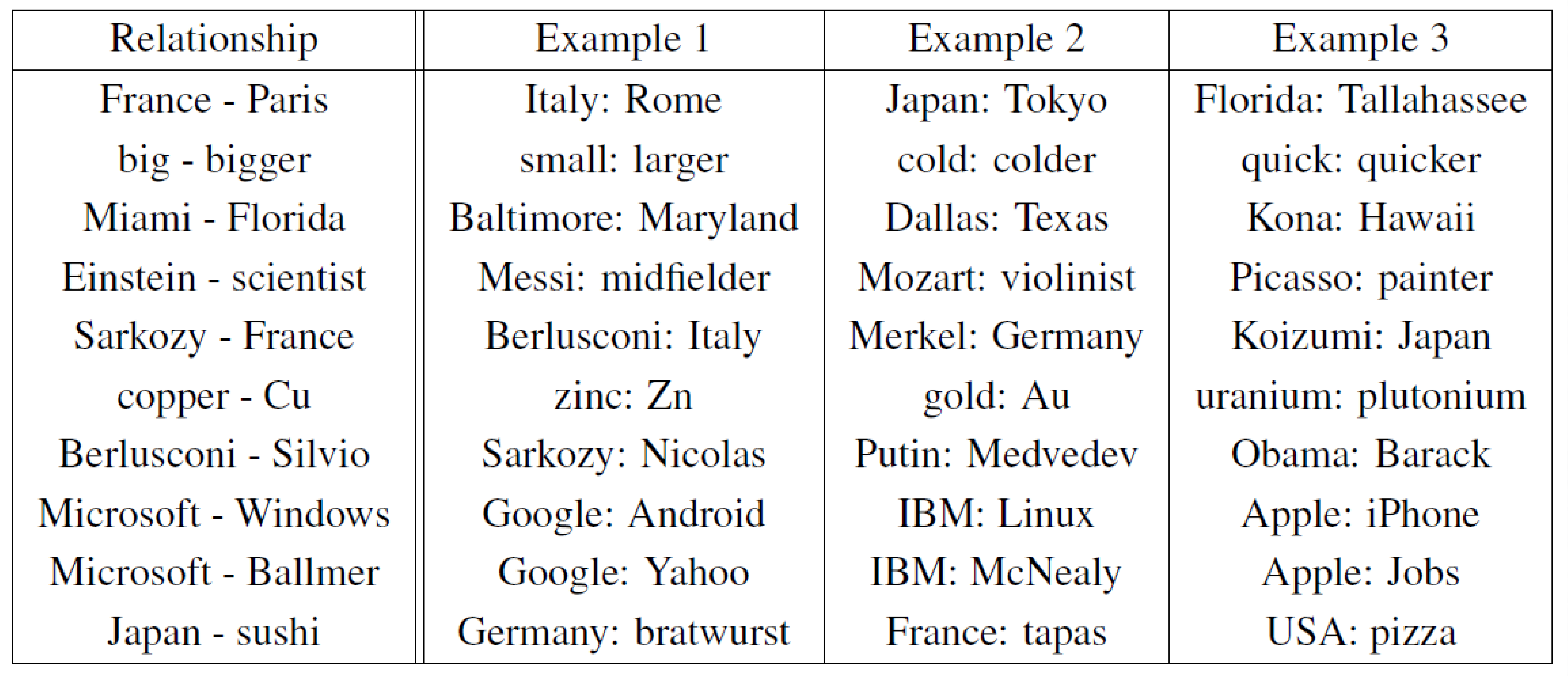


Type of relationship	Word Pair 1		Word Pair 2	
Common capital city	Athens	Greece	Oslo	Norway
All capital cities	Astana	Kazakhstan	Harare	Zimbabwe
Currency	Angola	kwanza	Iran	rial
City-in-state	Chicago	Illinois	Stockton	California
Man-Woman	brother	sister	grandson	granddaughter
Adjective to adverb	apparent	apparently	rapid	rapidly
Opposite	possibly	impossibly	ethical	unethical
Comparative	great	greater	tough	tougher
Superlative	easy	easiest	lucky	luckiest
Present Participle	think	thinking	read	reading
Nationality adjective	Switzerland	Swiss	Cambodia	Cambodian
Past tense	walking	walked	swimming	swam
Plural nouns	mouse	mice	dollar	dollars
Plural verbs	work	works	speak	speaks

“Efficient Estimation of Word Representations in Vector Space” Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

Relations Learned by Word embedding

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

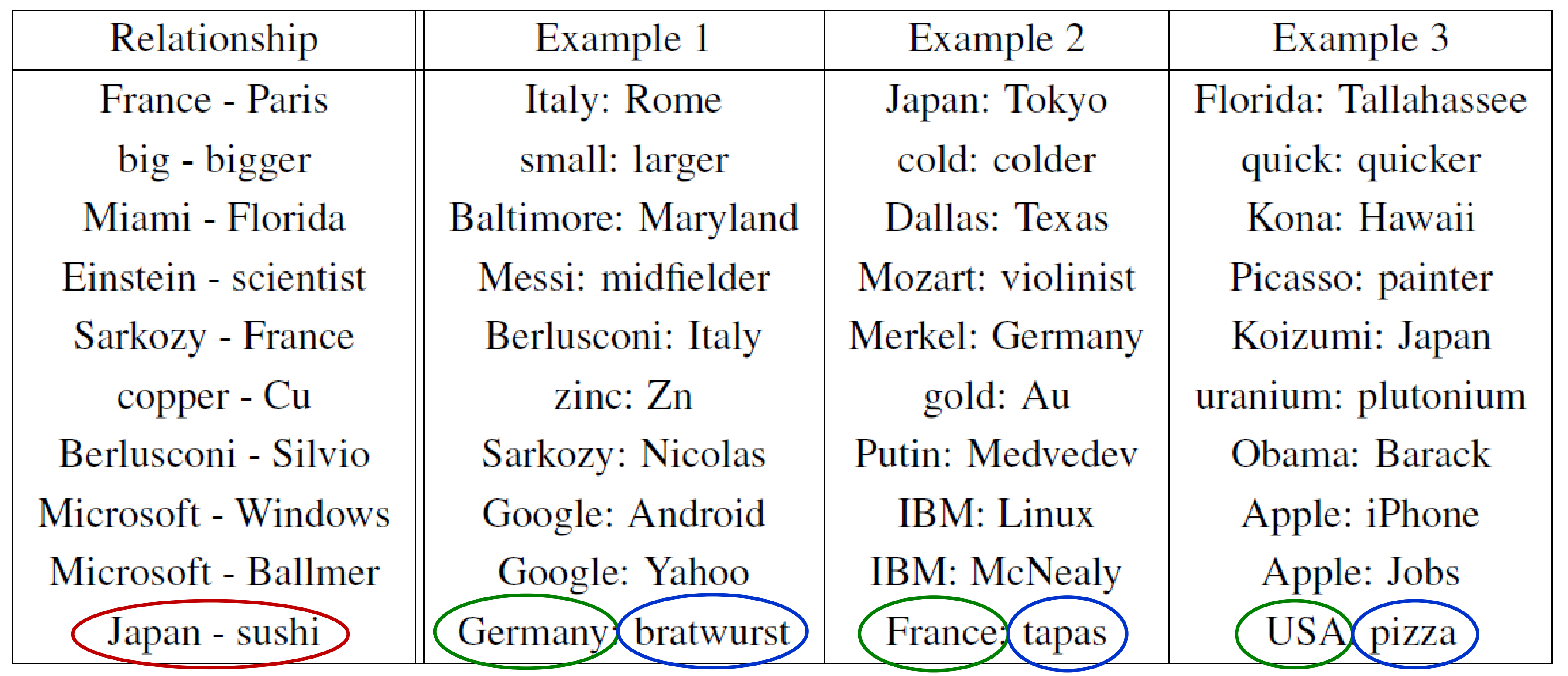


Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - 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

"Efficient Estimation of Word Representations in Vector Space" Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

Relations Learned by Word embedding

A relation is defined by the vector displacement in the first column. For each start word in the other column, the closest displaced word is shown.

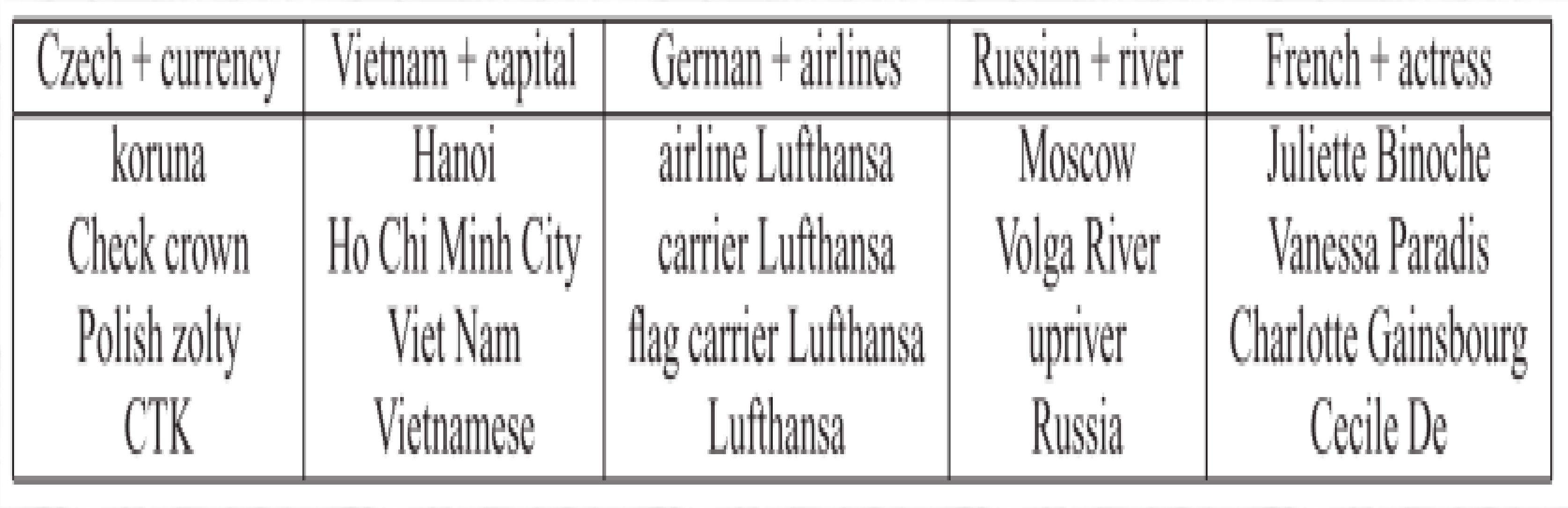


Relationship	Example 1	Example 2	Example 3
France - Paris	Italy: Rome	Japan: Tokyo	Florida: Tallahassee
big - bigger	small: larger	cold: colder	quick: quicker
Miami - Florida	Baltimore: Maryland	Dallas: Texas	Kona: Hawaii
Einstein - scientist	Messi: midfielder	Mozart: violinist	Picasso: painter
Sarkozy - 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

“Efficient Estimation of Word Representations in Vector Space” Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean, Arxiv 2013

Relations Learned by Word embedding

- We can also use element-wise addition of vector elements to ask questions such as ‘German + airlines’ and by looking at the closest tokens to the composite vector come up with impressive answers:



Czech + currency	Vietnam + capital	German + airlines	Russian + river	French + actress
koruna	Hanoi	airline Lufthansa	Moscow	Juliette Binoche
Check crown	Ho Chi Minh City	carrier Lufthansa	Volga River	Vanessa Paradis
Polish zolty	Viet Nam	flag carrier Lufthansa	upriver	Charlotte Gainsbourg
CTK	Vietnamese	Lufthansa	Russia	Cecile De

Word2vec Advantages

- It scales
  - Train on billion word corpora
  - In limited time
  - Mikolov mentions parallel training
- Incremental Training
  - Train on one piece of data, save results, continue training later on
- Pre-trained Word2vec Models

  https://code.google.com/archive/p/word2vec/

  https://github.com/dav/word2vec