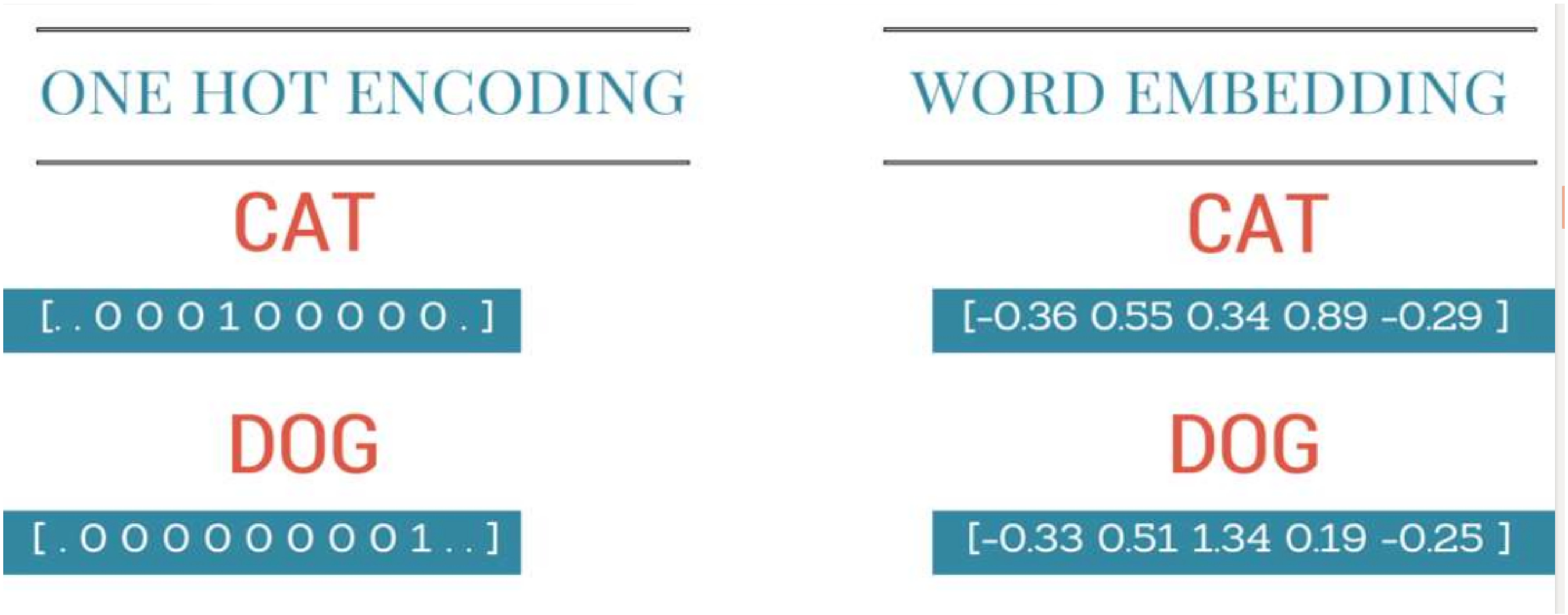
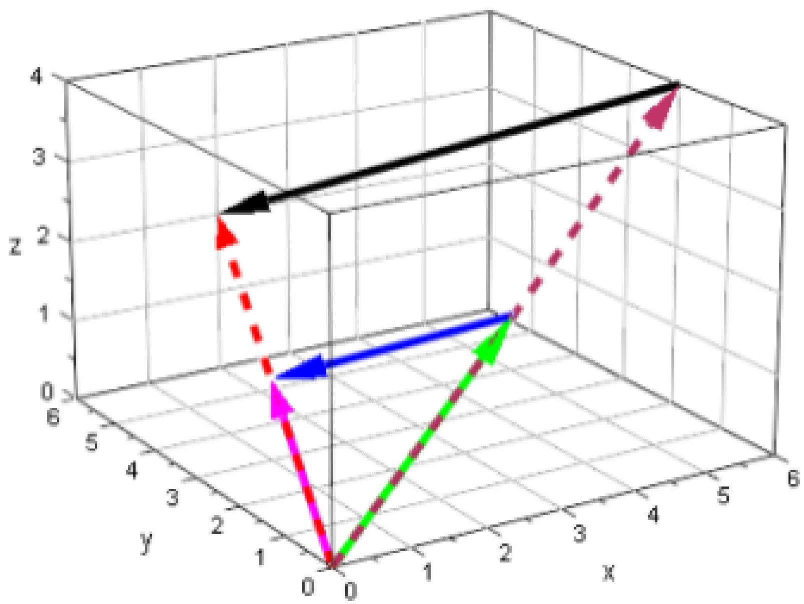
One Hot Encoding vs. Word Vector

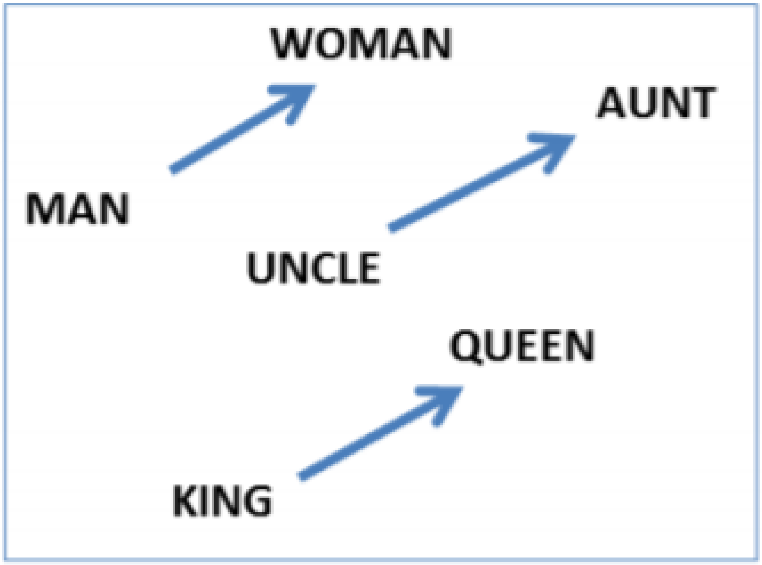


Vector Space



$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

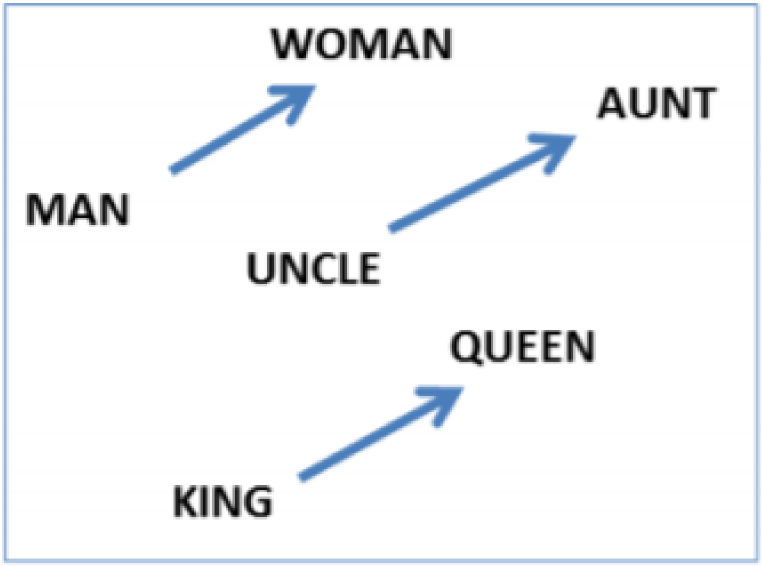
Vector Space



(Mikolov et al., NAACL HLT, 2013)

Vectors are directions in space
Vectors can encode relationships

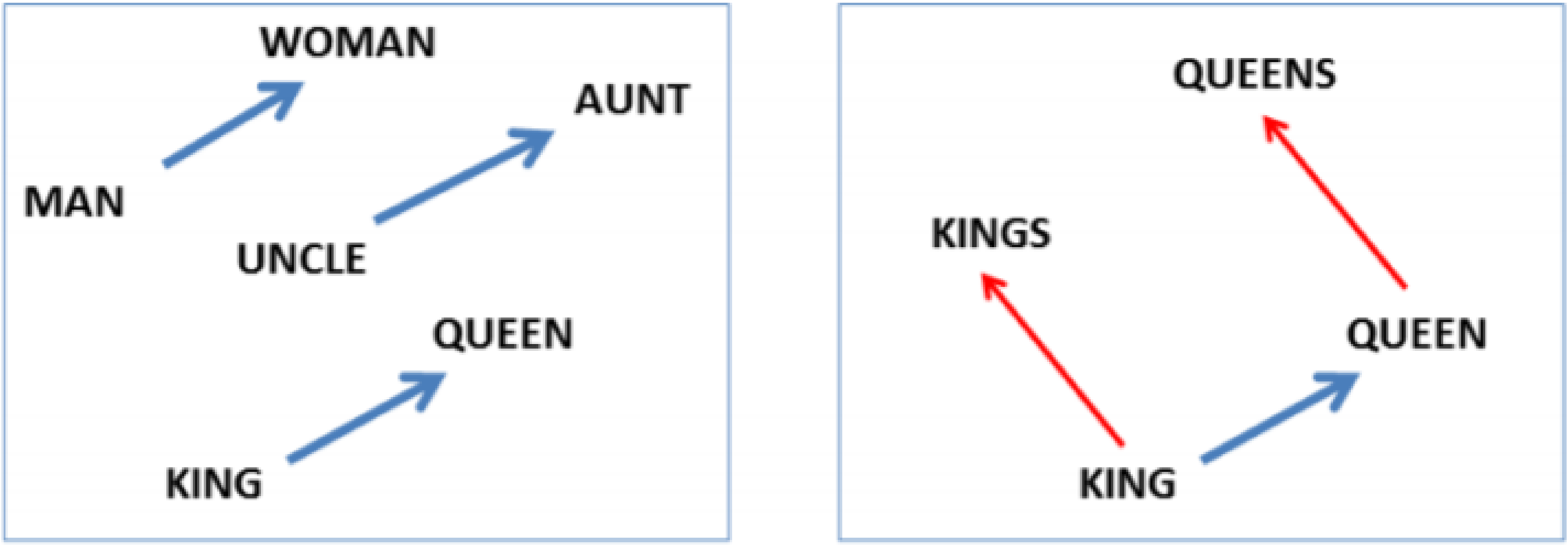
Vector Space



(Mikolov et al., NAACL HLT, 2013)

man is to **woman** as **king** is to **?**

Vector Space



(Mikolov et al., NAACL HLT, 2013)

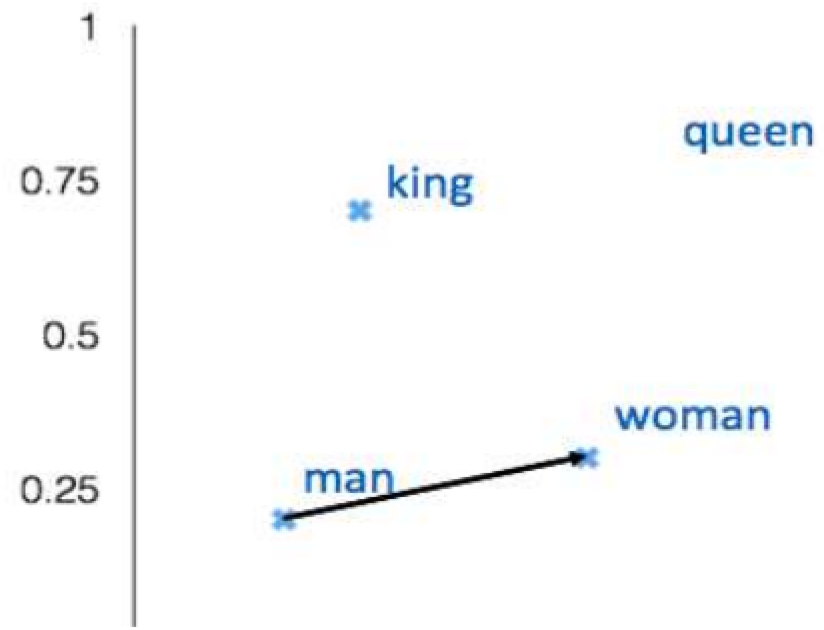
Computation Relationship

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]



Similarity Visualization

FRANCE	JESUS	XBOX	REDDISH	SCRATCHED	MEGABITS
AUSTRIA	GOD	AMIGA	GREENISH	NAILED	OCTETS
BELGIUM	SATI	PLAYSTATION	BLUISH	SMASHED	MB/S
GERMANY	CHRIST	MSX	PINKISH	PUNCHED	BIT/S
ITALY	SATAN	IPOD	PURPLISH	POPPED	BAUD
GREECE	KALI	SEGA	BROWNISH	CRIMPED	CARATS
SWEDEN	INDRA	PSNUMBER	GREYISH	SCRAPED	KBIT/S
NORWAY	VISHNU	HD	GRAYISH	SCREWED	MEGAHERTZ
EUROPE	ANANDA	DREAMCAST	WHITISH	SECTIONED	MEGAPIXELS
HUNGARY	PARVATI	GEFORCE	SILVERY	SLASHED	GBIT/S
SWITZERLAND	GRACE	CAPCOM	YELLOWISH	RIPPED	AMPERES

What words have embeddings closest to a given word? From Collobert *et al.* (2011)

Capture Relationship

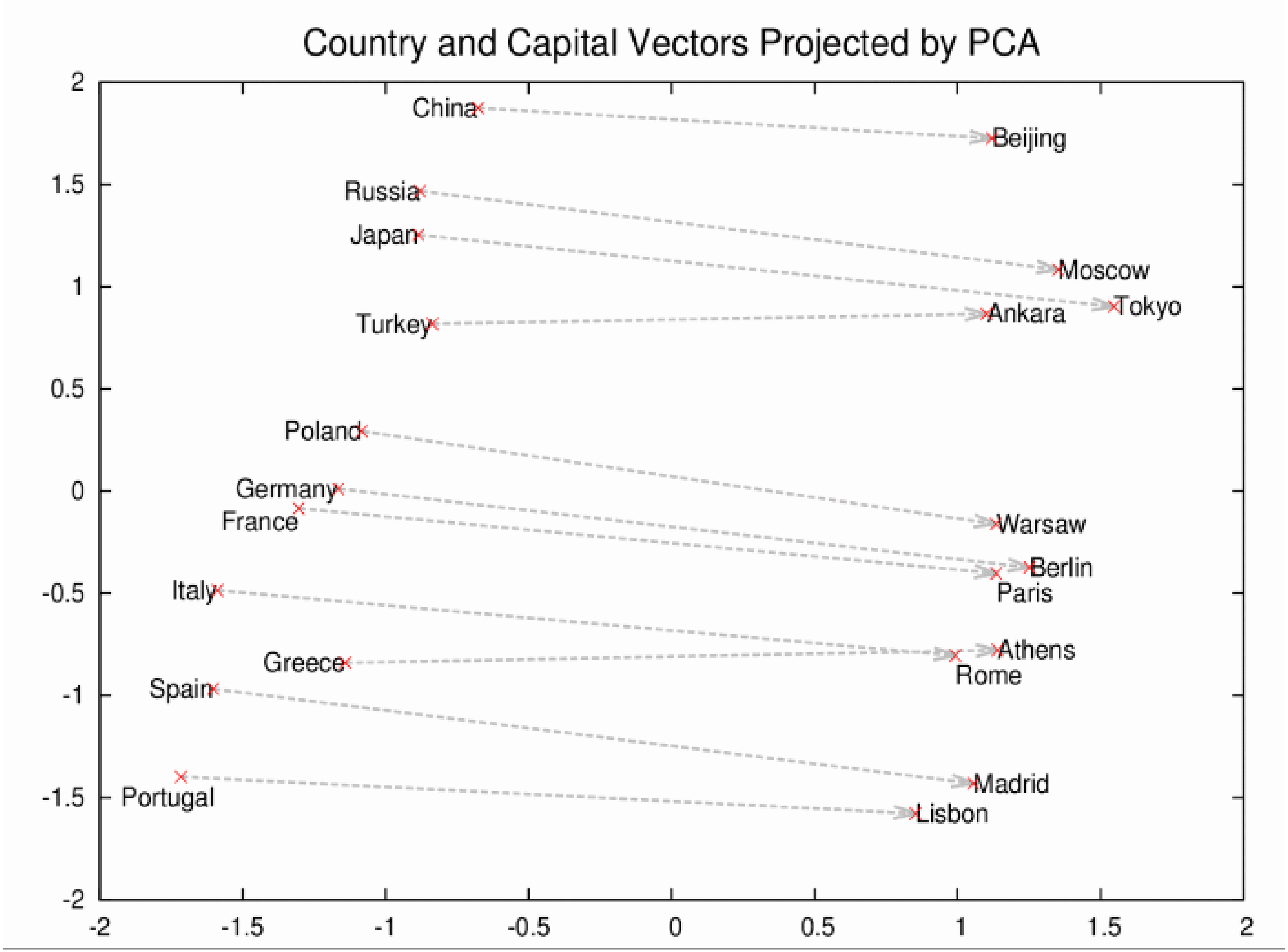
$$W(\text{``woman''}) - W(\text{``man''}) \simeq W(\text{``aunt''}) - W(\text{``uncle''})$$

$$W(\text{``woman''}) - W(\text{``man''}) \simeq W(\text{``queen''}) - W(\text{``king''})$$

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

Relationship pairs in a word embedding. From Mikolov *et al.* (2013b).

Relationship Visualization



Computation Sentence Vector

word	vector
The	(0.12, 0.23, 0.56)
Cardinals	(0.24, 0.65, 0.72)
will	(0.38, 0.42, 0.12)
win	(0.57, 0.01, 0.02)
the	(0.53, 0.68, 0.91)
world	(0.11, 0.27, 0.45)
series	(0.01, 0.05, 0.62)

⇒

sentence vector

(0.28, 0.33, 0.49)

Comparing 2 sentences

Take the two sentences:

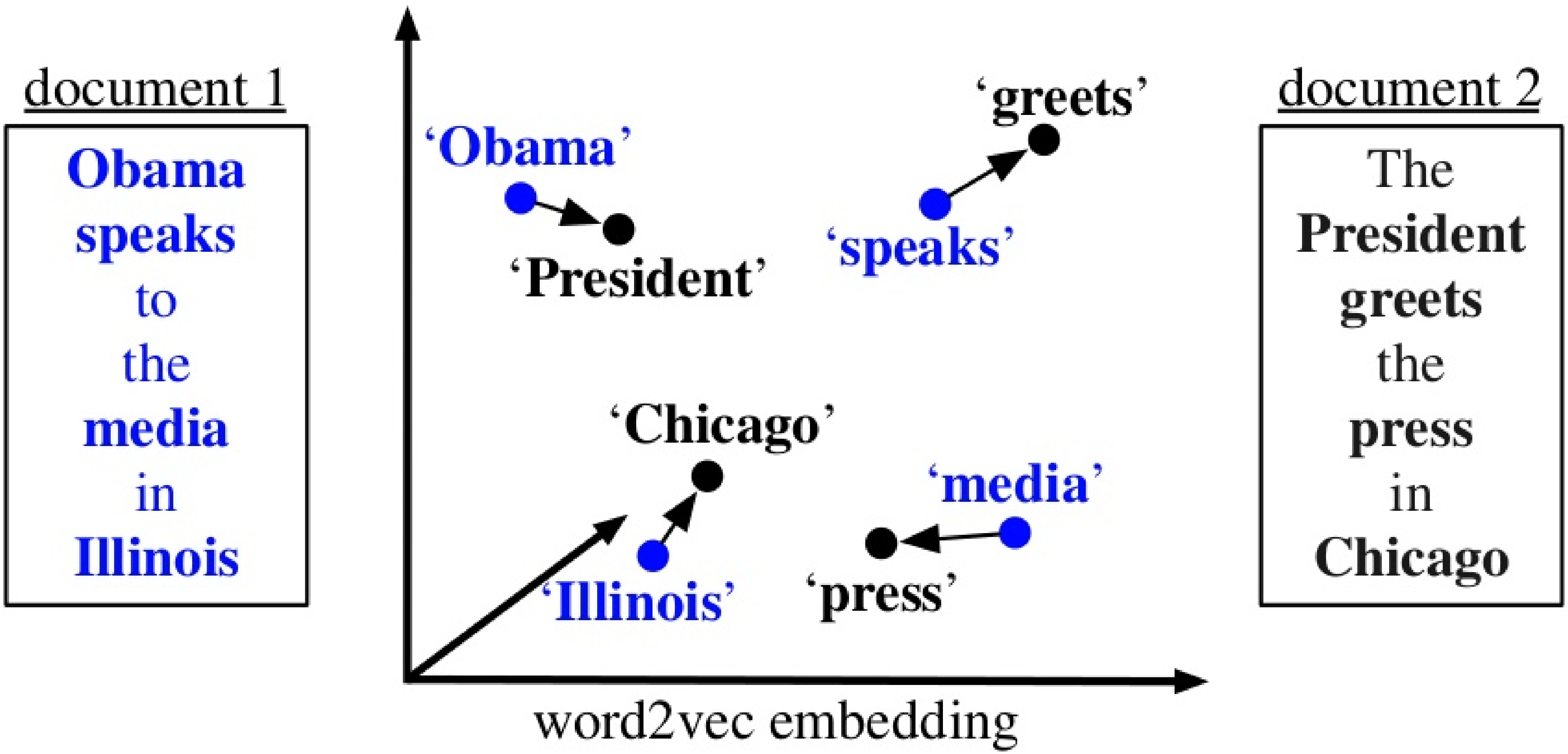
"Obama speaks to the media in Illinois"

and

"The President greets the press in Chicago"

While these sentences have no words in common, they convey nearly the same information, a fact that cannot be represented by the BOW model (*Kusner, Matt J. et al. 2015*).

Comparing 2 sentences



Source: From Word Embeddings To Document Distances. Kusner, Matt J. et al. 2015.

How to compare 2 sentences

?

How to build Wordvector?

Using Contextual information

Contextual representation

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

One of the most successful ideas of modern statistical NLP

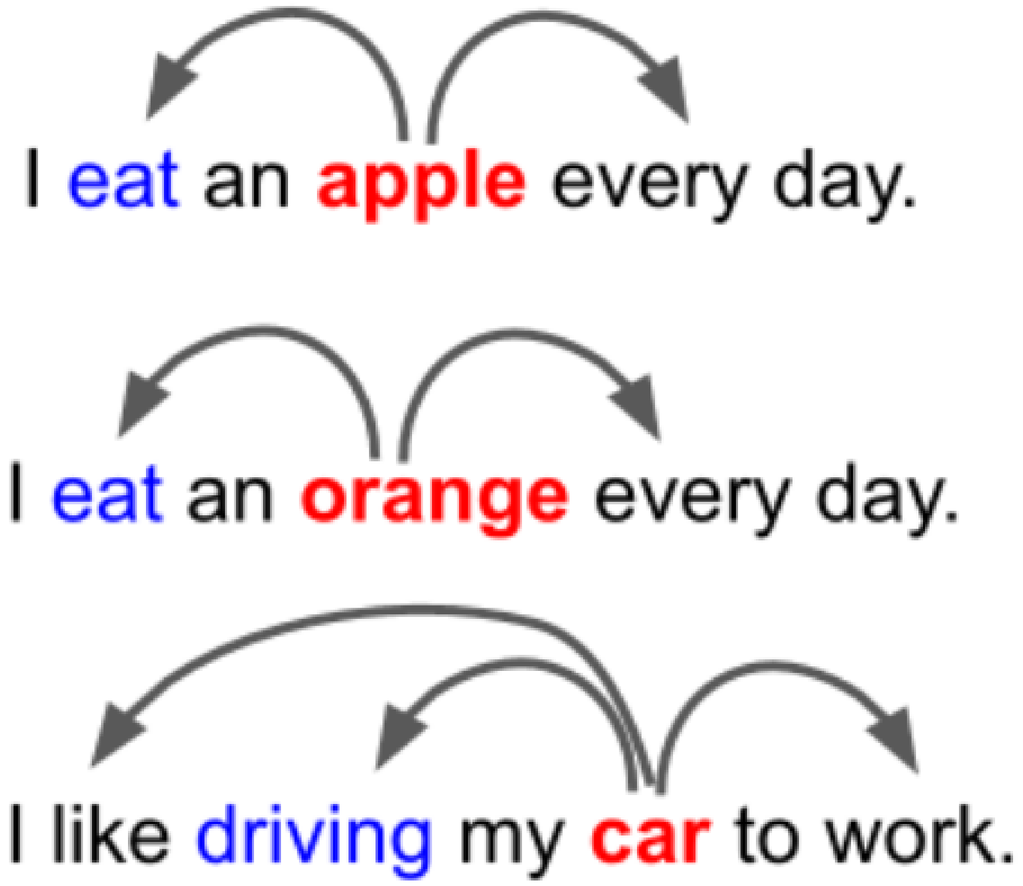
government debt problems turning into **banking** crises as has happened in

saying that Europe needs unified **banking** regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

Contextual representation

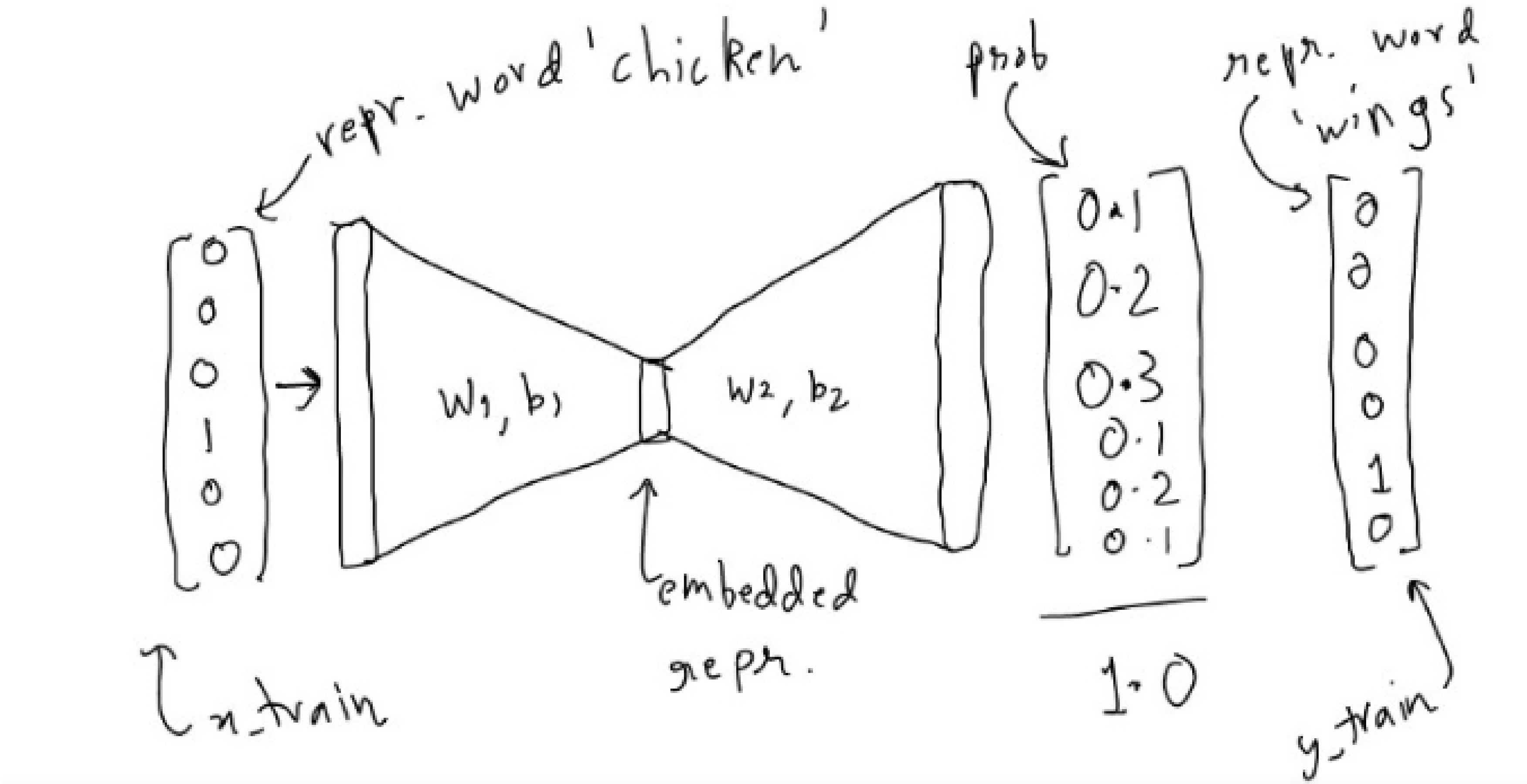
Word is represented by context in use



How to build Word Vector

- Take a 3 layer neural network. (1 input layer + 1 hidden layer + 1 output layer)
- Feed it a word and train it to predict its neighbouring word.
- Remove the last (output layer) and keep the input and hidden layer.
- Now, input a word from within the vocabulary. The output given at the hidden layer is the ‘word embedding’ of the input word.

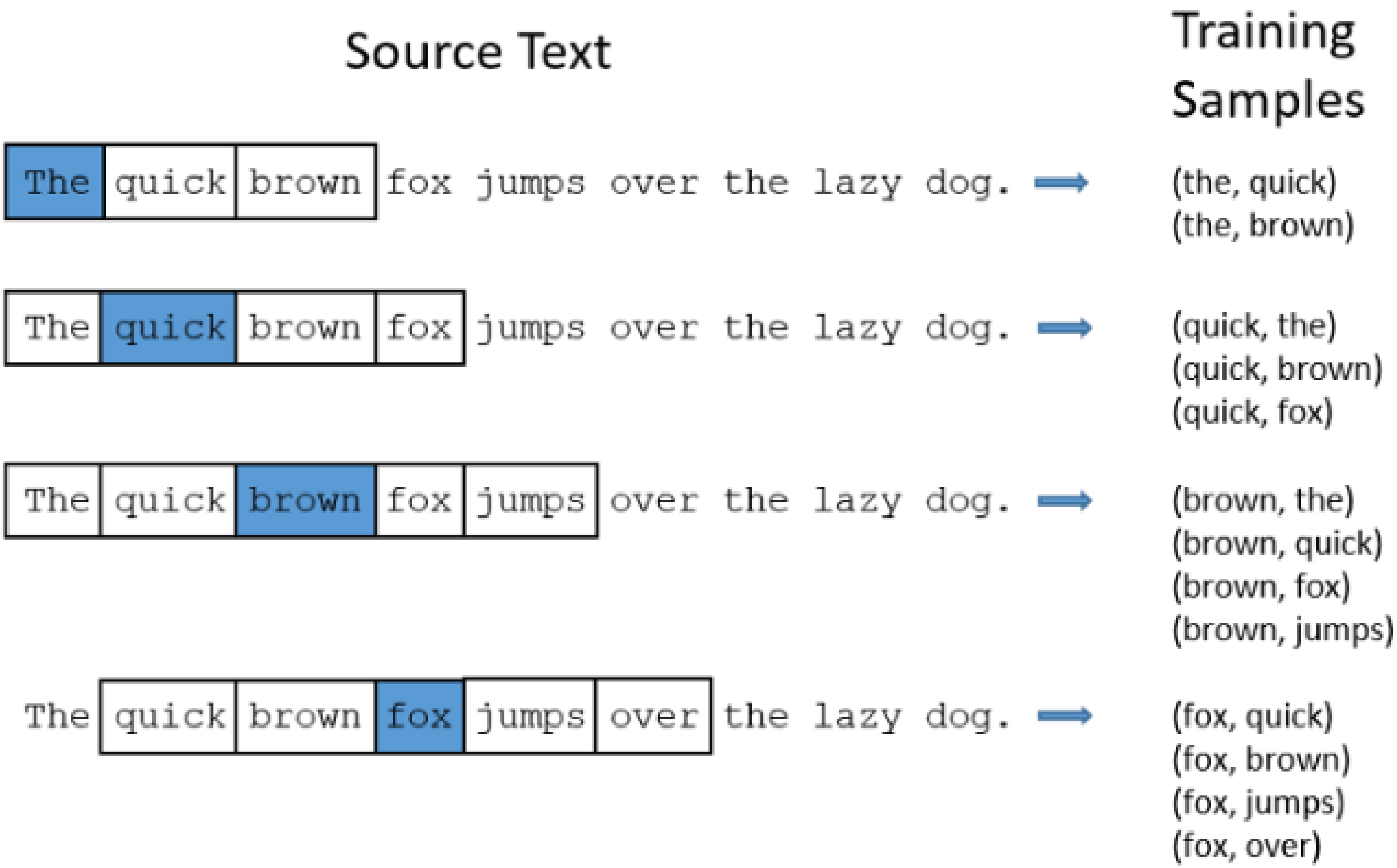
Auto encoder like structure to build Word2vec



Step 1: Define the corpus

```
import numpy as np
import tensorflow as tf
corpus_raw = 'He is the king . The king is royal .
She is the royal  queen '
# convert to lower case
corpus_raw = corpus_raw.lower()
```

Step 2: Rule for building the training data



Step 2_1: Building The word2int and int2word functions

```
words = []
for word in corpus_raw.split():
    if word != '.': # because we don't want to treat . as a word
        words.append(word)
words = set(words) # so that all duplicate words are removed
word2int = {}
int2word = {}
vocab_size = len(words) # gives the total number of unique words
for i,word in enumerate(words):
    word2int[word] = i
    int2word[i] = word

# test functions
print(word2int['queen'])
print(int2word[42])
```

Step 2_2: Build tokenized sentences

```
# raw sentences is a list of sentences.
raw_sentences = corpus_raw.split('.')
sentences = []
for sentence in raw_sentences:
    sentences.append(sentence.split())

print(sentences)
```

Step 2_3: generate training data

```
data = []
WINDOW_SIZE = 2
for sentence in sentences:
    for word_index, word in enumerate(sentence):
        for nb_word in sentence[max(word_index - WINDOW_SIZE, 0) : min(word_index + WINDOW_SIZE, len(sentence)) + 1] :
            if nb_word != word:
                data.append([word, nb_word])

print(data)
```

Step 2_4: Building xtrain, ytrain

```
# function to convert numbers to one hot vectors
def to_one_hot(data_point_index, vocab_size):
    temp = np.zeros(vocab_size)
    temp[data_point_index] = 1
    return temp
x_train = [] # input word
y_train = [] # output word
for data_word in data:
    x_train.append(to_one_hot(word2int[ data_word[0] ], vocab_size))
    y_train.append(to_one_hot(word2int[ data_word[1] ], vocab_size))
# convert them to numpy arrays
x_train = np.asarray(x_train)
y_train = np.asarray(y_train)


print(x_train)
print(x_train.shape, y_train.shape)
```

Step 3: Build tensorflow model

```
# making placeholders for x_train and y_train
x = tf.placeholder(tf.float32, shape=(None, vocab_size))
y_label = tf.placeholder(tf.float32, shape=(None, vocab_size))

EMBEDDING_DIM = 5 # you can choose your own number
W1 = tf.Variable(tf.random_normal([vocab_size, EMBEDDING_DIM]))
b1 = tf.Variable(tf.random_normal([EMBEDDING_DIM])) #bias
hidden_representation = tf.add(tf.matmul(x,W1), b1)

W2 = tf.Variable(tf.random_normal([EMBEDDING_DIM, vocab_size]))
b2 = tf.Variable(tf.random_normal([vocab_size]))
prediction = tf.nn.softmax(tf.add( tf.matmul(hidden_representation, W2), b2))


sess = tf.Session()
init = tf.global_variables_initializer()
sess.run(init) #make sure you do this!

# define the loss function:
cross_entropy_loss = tf.reduce_mean(-tf.reduce_sum(y_label * tf.log(prediction),
reduction_indices=[1]))

# define the training step:
train_step = tf.train.GradientDescentOptimizer(0.1).minimize(cross_entropy_loss)
```

Step 3: train the tensorflow model

```
n_iters = 10000
# train for n_iter iterations

for _ in range(n_iters):
    sess.run(train_step, feed_dict={x: x_train, y_label: y_train})
    print('loss is : ', sess.run(cross_entropy_loss, feed_dict={x: x_train, y_label: y_train}))

vectors = sess.run(W1 + b1)
```

Step 4: define nlp functions

```
def euclidean_dist(vec1, vec2):
    return np.sqrt(np.sum((vec1-vec2)**2))

def find_closest(word_index, vectors):
    min_dist = 10000 # to act like positive infinity
    min_index = -1
    query_vector = vectors[word_index]
    for index, vector in enumerate(vectors):
        if euclidean_dist(vector, query_vector) < min_dist and not np.array_equal(vector, query_vector):
            min_dist = euclidean_dist(vector, query_vector)
            min_index = index
    return min_index
```

Step 3_1: test the model

```
print(vectors[ word2int['queen'] ])

print(int2word[find_closest(word2int['king'], vectors)])
print(int2word[find_closest(word2int['queen'], vectors)])
print(int2word[find_closest(word2int['royal'], vectors)])
```