DIPARTIMENTO DI MATEMATICA E INFORMATICA
UNIVERSITÀ degli STUDI di CATANIA

Word and Document Embeddings

Alessandro Ortis, PhD
ortis@dmi.unict.it

IMAGE PROCESSING LABORATORY
http//iplab.dmi.unict.it

word2vec

Word2vec is a neural network architecture used to produce word embeddings.

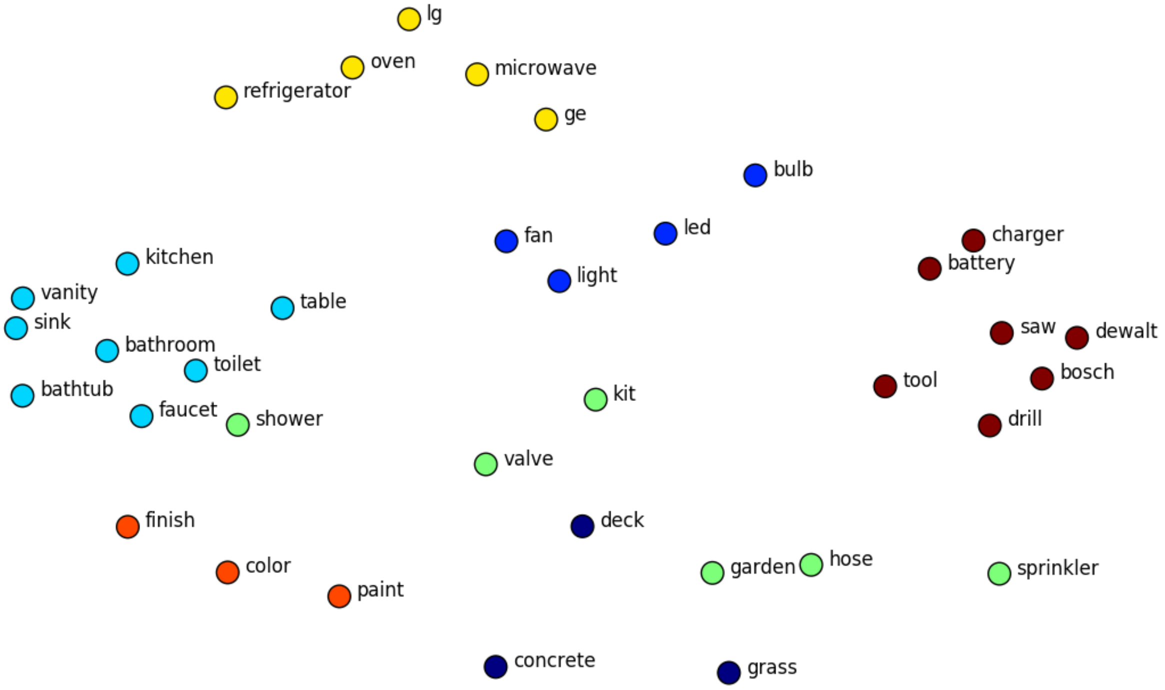
word $\longrightarrow$ **vector**

Word2vec takes as its input a large corpus of text and produces a vector space in which each word is assigned to a corresponding vector in the space.

Word vectors are positioned in the vector space such that words that share *common contexts* in the corpus are located in close proximity to one another in the space.

word2vec

Words that share ***common contexts*** in the corpus are located in close proximity to one another.



word2vec

We need a way to compare words so that if two words are similar, then their representations are close.

…	**…**
armchair	**man**
bench	**woman**
chair	**child**

Next to the **sofa** is a desk, and a **person** is sitting behind it.

deck chair	**girl**
seat	**boy**
…	**…**

word2vec

apple	[0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0]
orange	[0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 ... 0 0 0 0 0 0]
car	[0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0]

1-hot encoding

word2vec

apple [0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0]

orange [0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 0 0 ... 0 0 0 0 0 0]

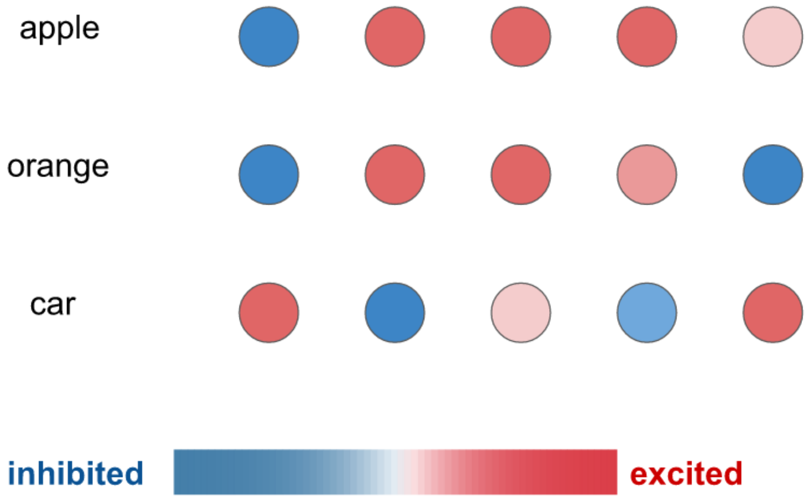
car [0 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 ... 0 0 0 0 0 0]

1-hot encoding

apple juice → ____ juice ?

word2vec

Word is represented as continuous level of activations



word2vec

But we do not have word similarities (how to encode that the words *orange* and *apple* share the ''context'' *juice*?).

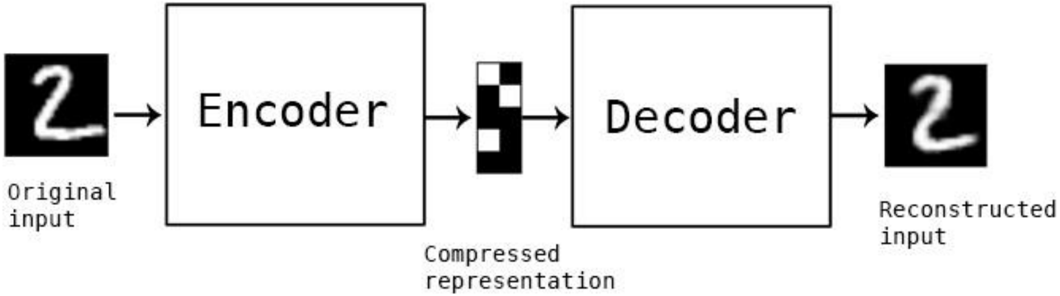
How do we learn the vectors? Use an ***auto-encoder***

word2vec

But we do not have word similarities (how to encode that the words *orange* and *apple* share the ''context'' *juice*?).

How do we learn the vectors? Use an ***auto-encoder***

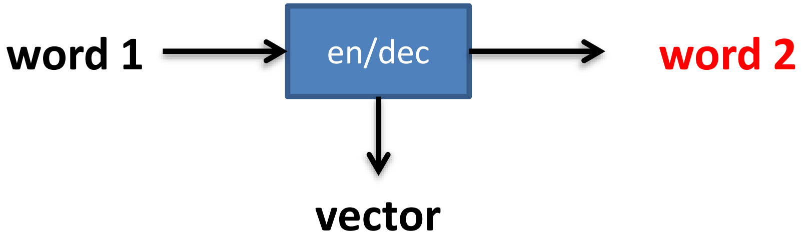
An auto-encoder learns a representation (encoding) for a set of data with the aim to use the representation to reconstruct the original data.



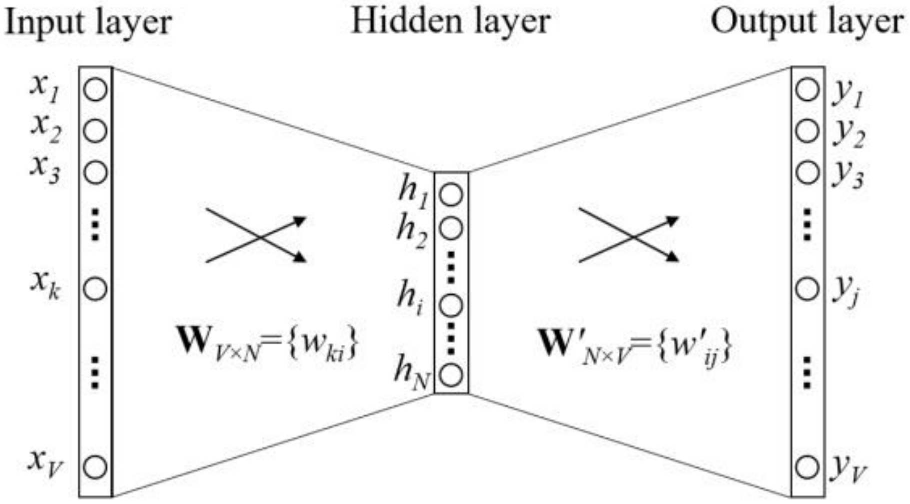
word2vec

Like auto-encoders, we can train a system to perform a different task, and hope that similarity will emerge…

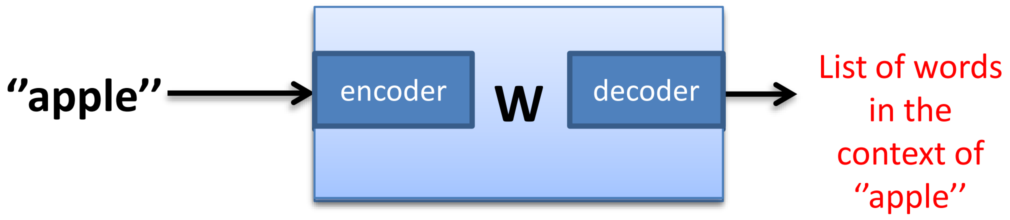
We will train a classifier to predict the *words surrounding each word*

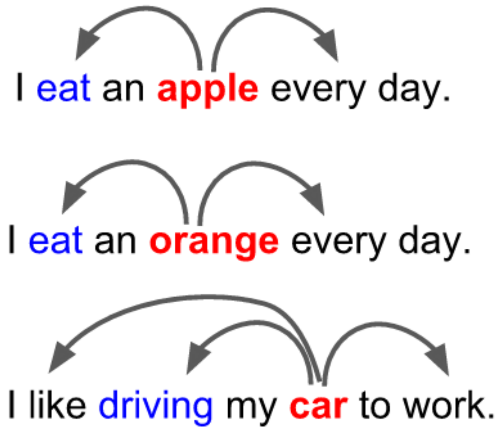


word2vec


Input layer
Hidden layer
Output layer
x_1
x_2
x_3
x_k
x_V
h_1
h_2
h_i
h_N
y_1
y_2
y_3
y_j
y_V
$\mathbf{W}_{V\times N}=\{w_{ki}\}$
$\mathbf{W}'_{N\times V}=\{w'_{ij}\}$

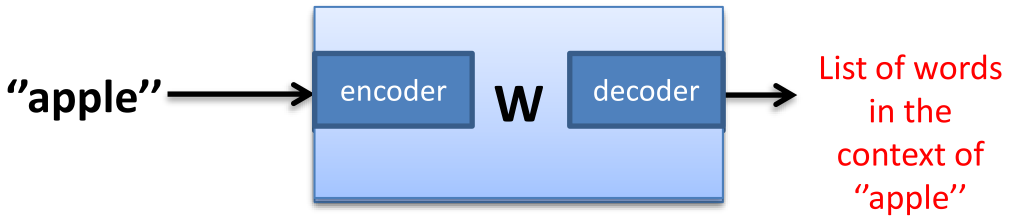
word2vec

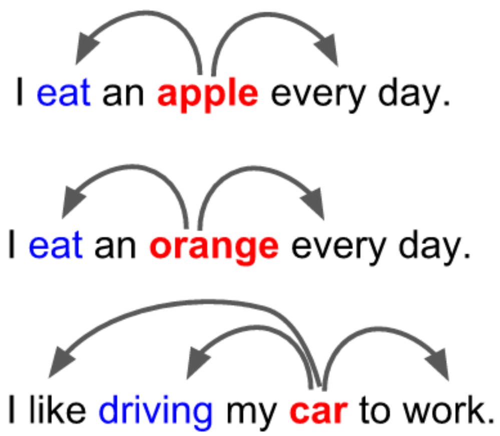




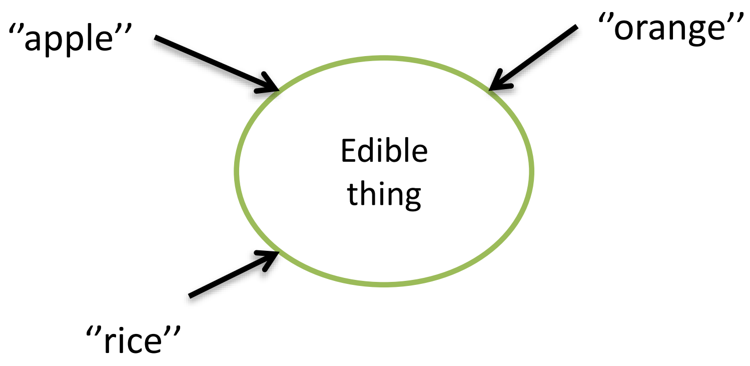
Word is represented by context in use

word2vec

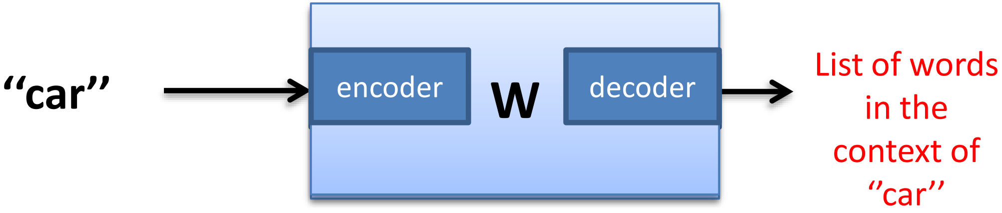


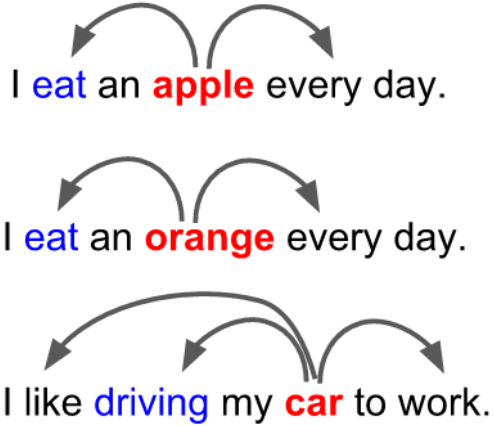


Word is represented by context in use

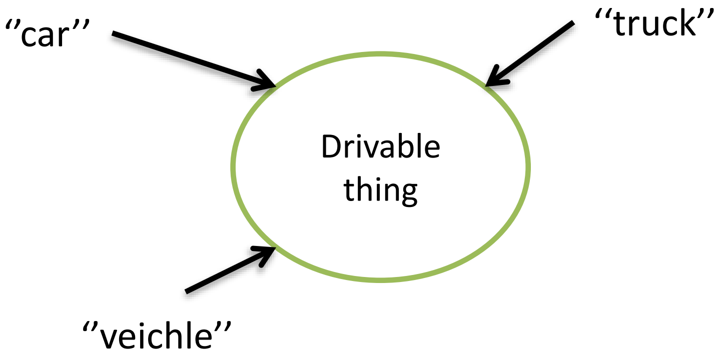


word2vec

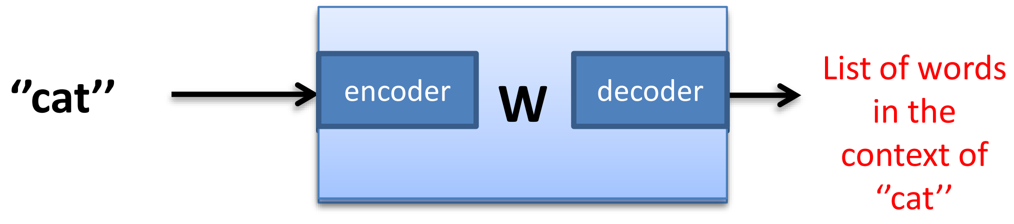




Word is represented by context in use



word2vec

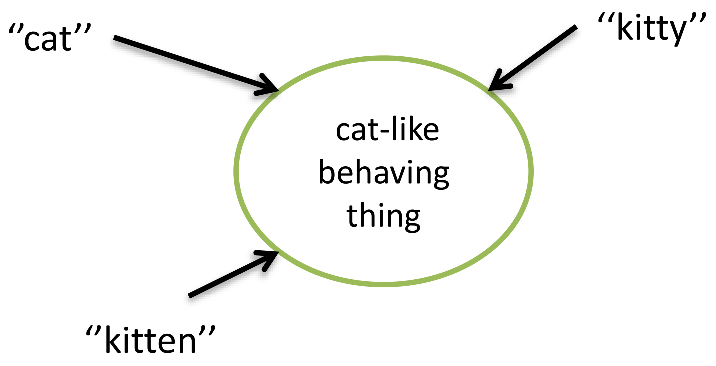


I eat an apple every day.

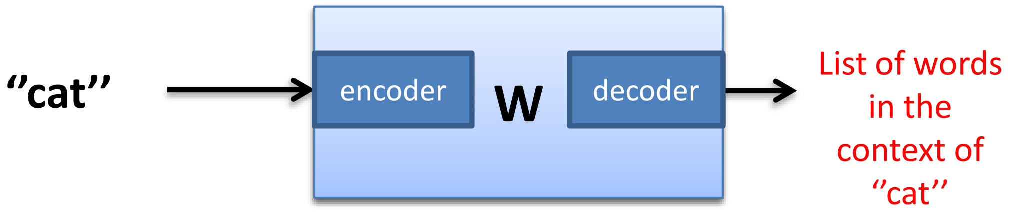
I eat an orange every day.

I like driving my car to work.

Word is represented by context in use

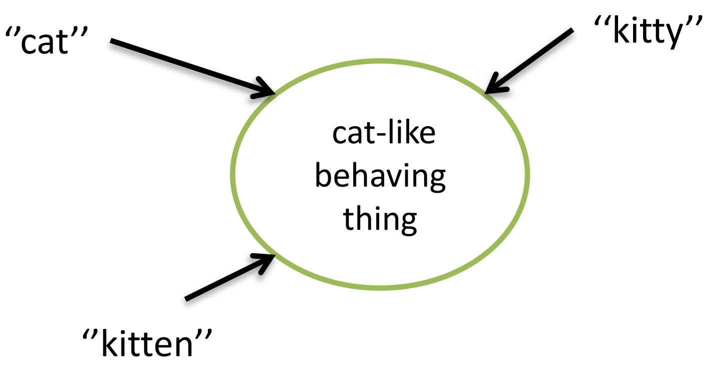


word2vec

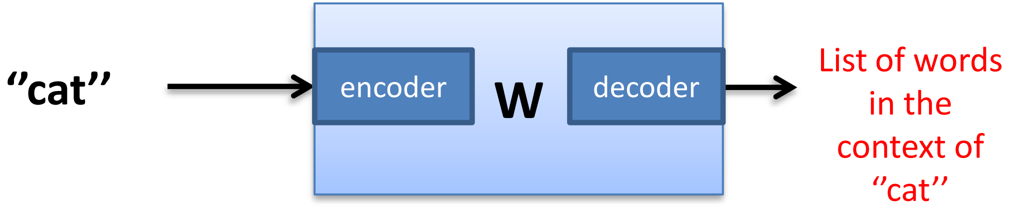


- Don't worry about the meaning of "cat", "kitty" or "kitten",
- The ***context*** gives you a strong idea that those words are similar
- You have to be "cat-like" to purr and hunt mice regardless whether you are a cat, a kitty or a kitten

Word is represented by context in use



word2vec



- Don't worry about the meaning of "cat", "kitty" or "kitten",
- The ***context*** gives you a strong idea that those words are similar
- You have to be "cat-like" to purr and hunt mice regardless whether you are a cat, a kitty or a kitten

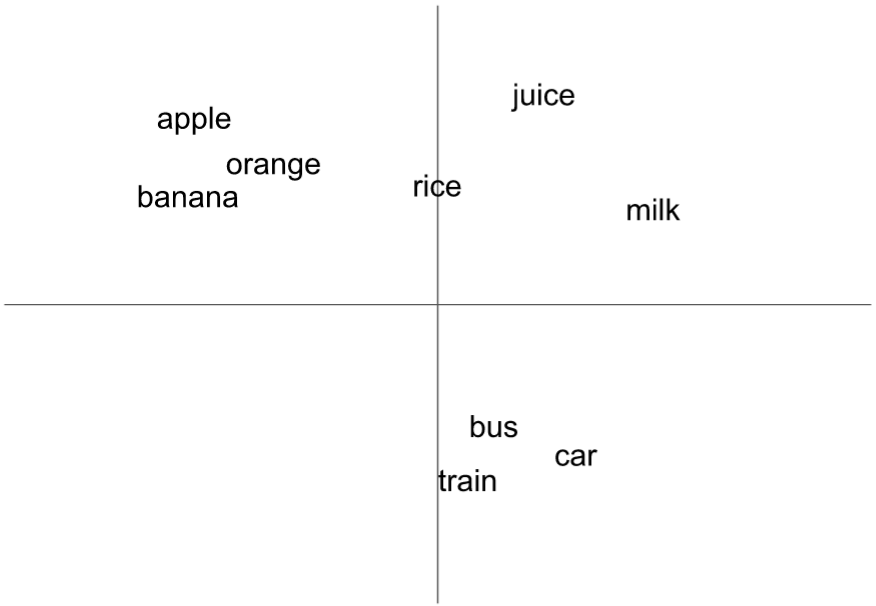
Word is represented by context in use

"the … meows"

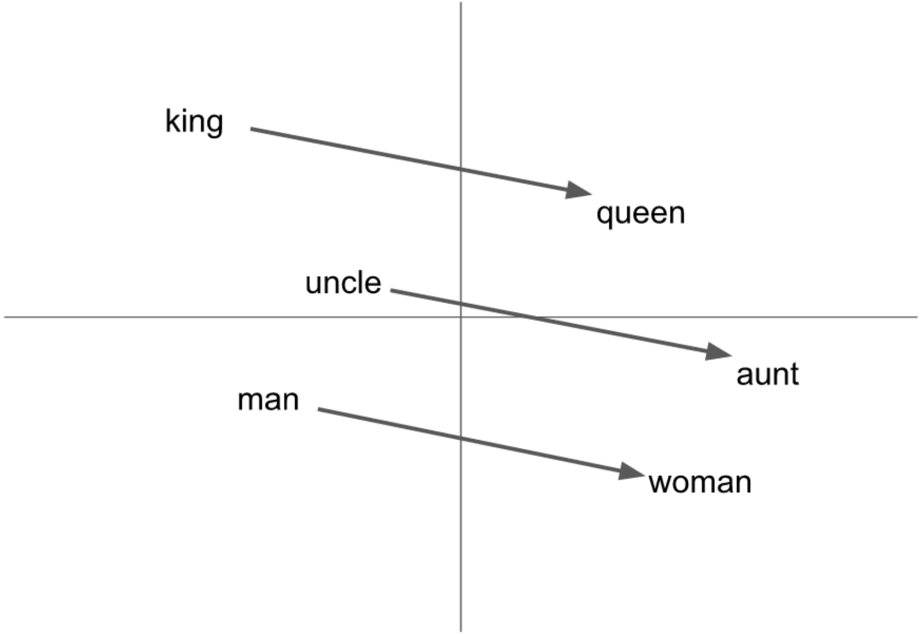
"the … purrs"

"the … hunts mice"

word2vec

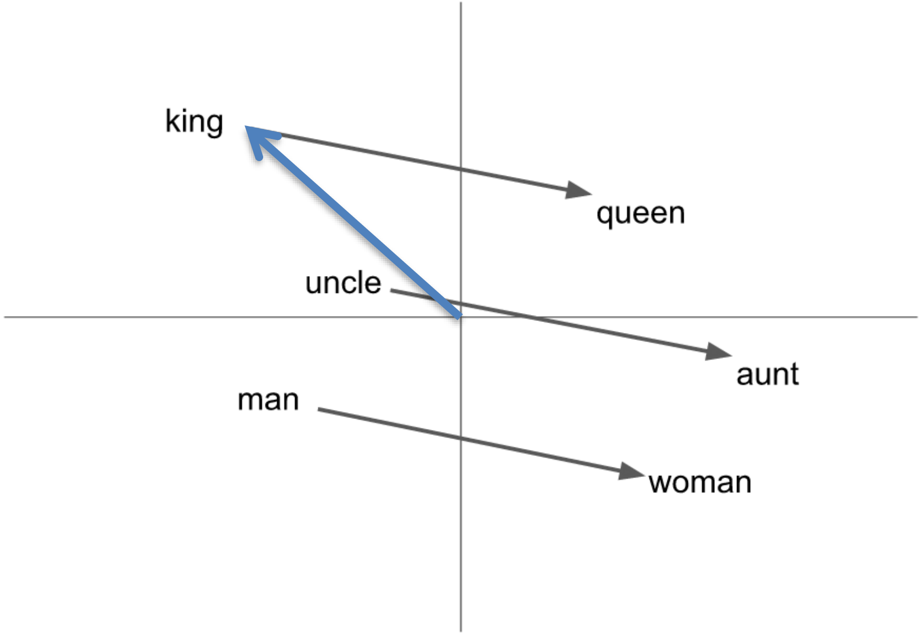


word2vec



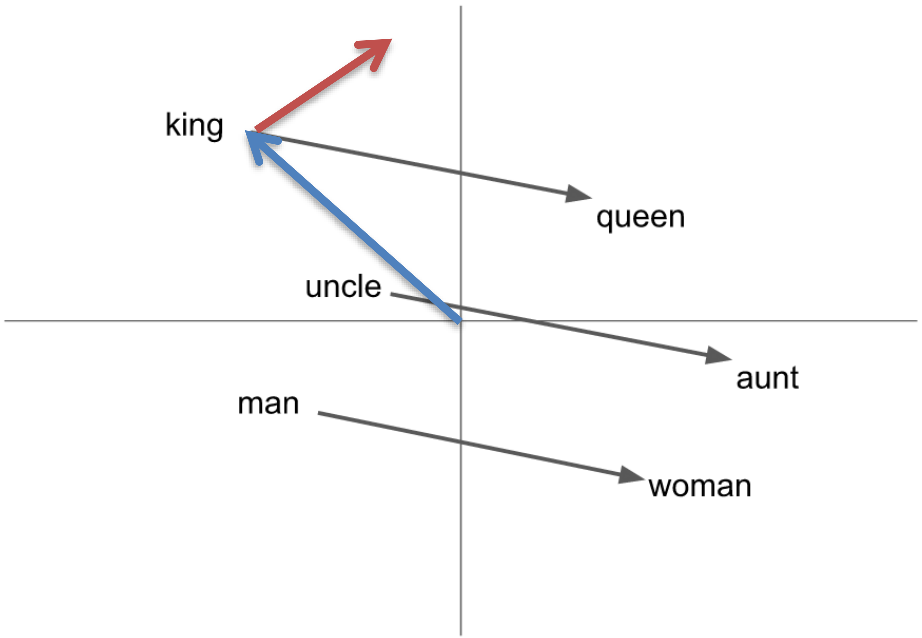
word2vec

King



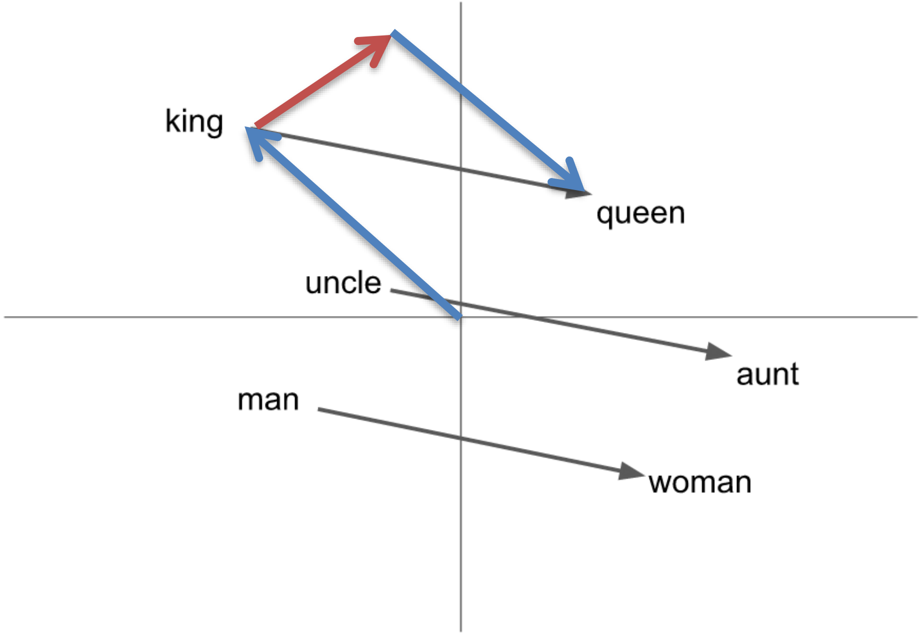
word2vec

King - man



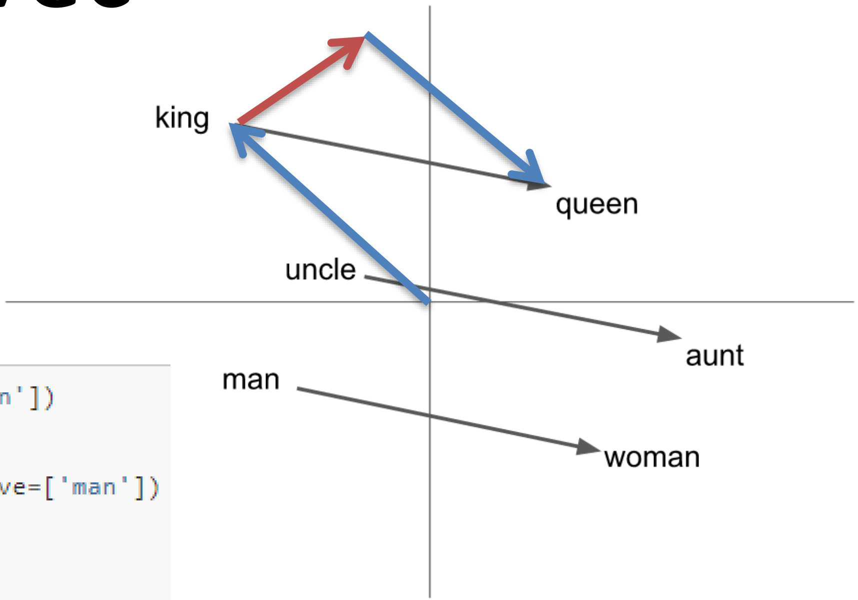
word2vec

King – man + woman → queen



word2vec

King – man + woman → queen



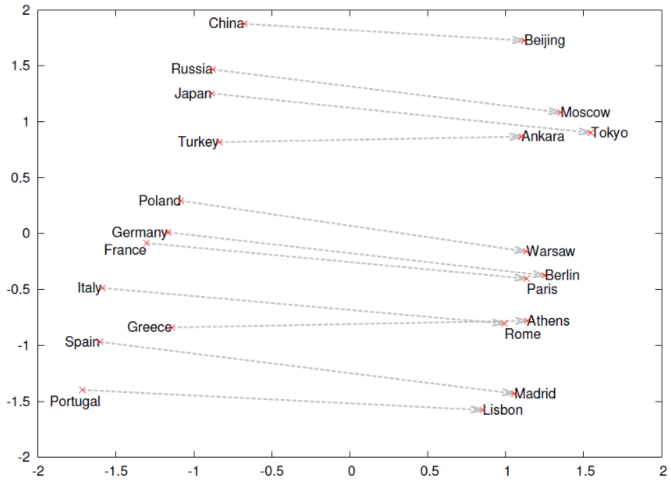

```
>>> model.wv.most_similar(positive=['woman', 'king'], negative=['man'])
[('queen', 0.50882536), ...]

>>> model.wv.most_similar_cosmul(positive=['woman', 'king'], negative=['man'])
[('queen', 0.71382287), ...]


>>> model.wv.doesnt_match("breakfast cereal dinner lunch".split())
'cereal'

>>> model.wv.similarity('woman', 'man')
0.73723527
```

[Link to the python library]()

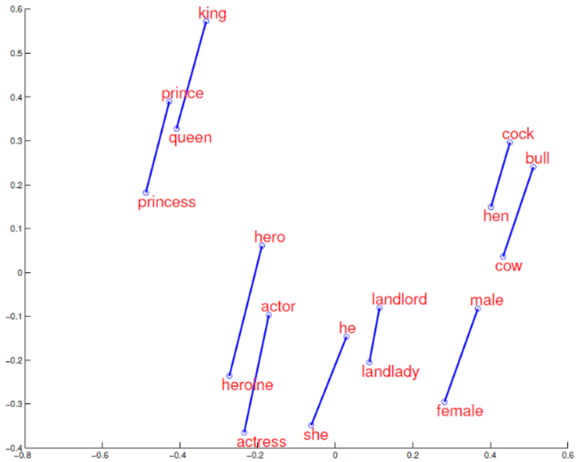
word2vec

Expression	*Nearest token*
Paris - France + Italy	Rome
bigger - big + cold	colder
sushi - Japan + Germany	bratwurst
Cu - copper + gold	Au
Windows - Microsoft + Google	Android
Montreal Canadiens - Montreal + Toronto	Toronto Maple Leafs

Expression	*Nearest tokens*
Czech + currency	koruna, Czech crown, Polish zloty, CTK
Vietnam + capital	Hanoi, Ho Chi Minh City, Viet Nam, Vietnamese
German + airlines	airline Lufthansa, carrier Lufthansa, flag carrier Lufthansa
Russian + river	Moscow, Volga River, upriver, Russia
French + actress	Juliette Binoche, Vanessa Paradis, Charlotte Gainsbourg



China
Beijing
Russia
Japan
Moscow
Tokyo
Turkey
Ankara
Poland
Germany
France
Warsaw
Berlin
Paris
Italy
Greece
Spain
Athens
Rome
Madrid
Portugal
Lisbon



king
prince
queen
princess
cock
bull
hen
cow
hero
actor
heroine
actress
he
she
landlord
landlady
male
female

fallen
fall
fell
drawn
draw
drew
taken
take
took
given
give
gave

doc2vec

Word2vec provides a method to learn representations of words.

Doc2vec implements the same mechanism to learn representations of *documents* (named *Paragraph Vectors*).

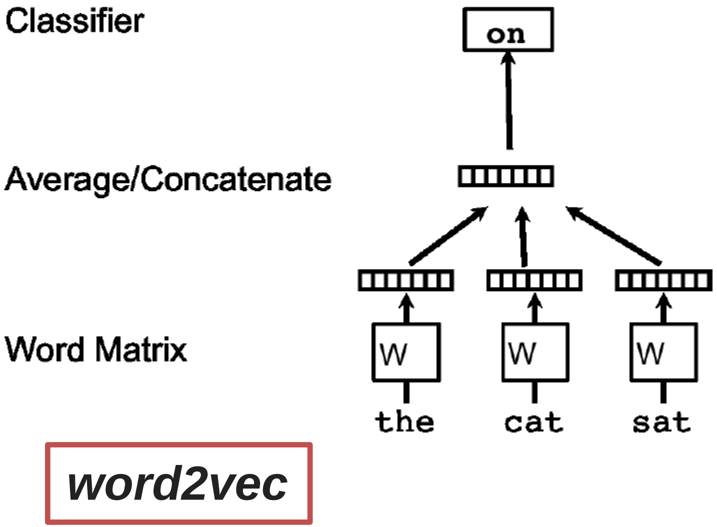
In particular, Doc2vec learns fixed-length feature representations of pieces of texts, such as:

- Sentences
- Paragraphs
- Documents
- Comments
- Tweets
- ….

Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." *International Conference on Machine Learning*. 2014.

word2vec → doc2vec

Input: *"the cat sat on the sofa"*



"on" (predicted word)

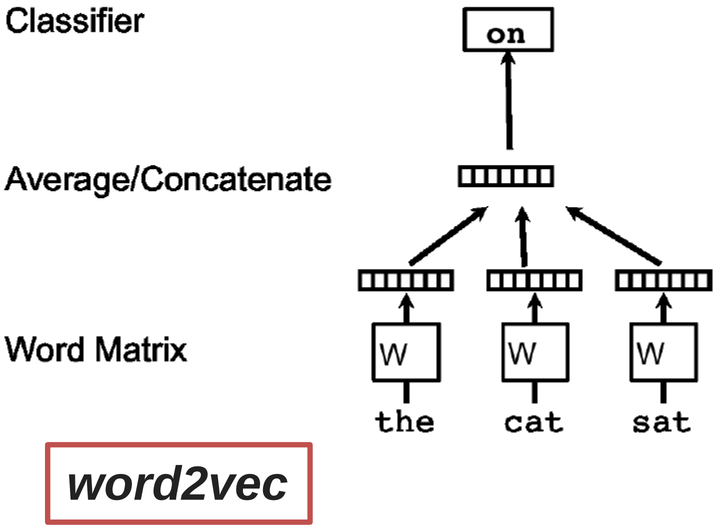
$$\frac{1}{T}\sum_{t=k}^{T-k}\log p(w_t \mid w_{t-k},\ldots,w_{t+k})$$

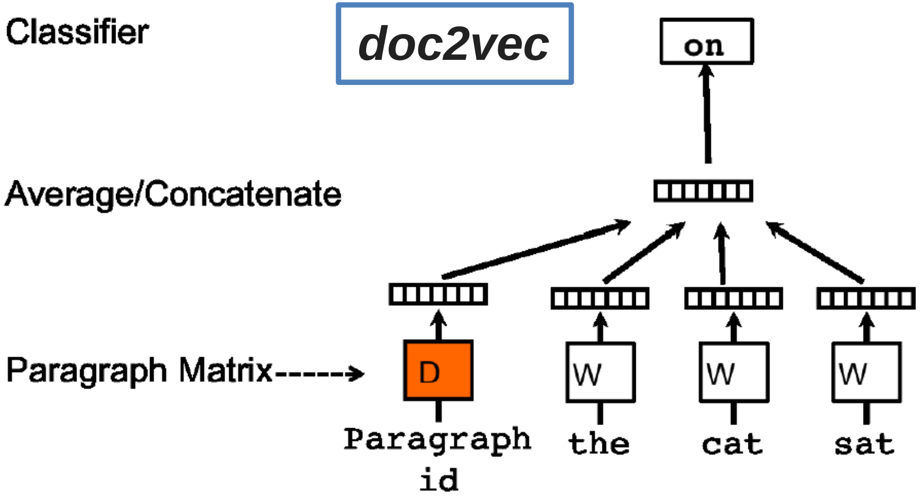
"the" "cat" "sat" (context words)

Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." *International Conference on Machine Learning*. 2014.

word2vec → doc2vec

Input: *“the cat sat on the sofa”*



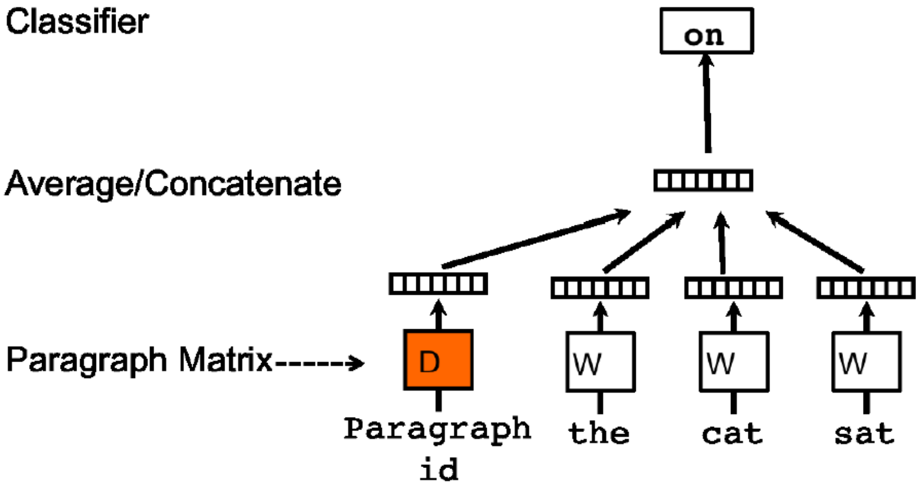


Le, Quoc, and Tomas Mikolov. "Distributed representations of sentences and documents." *International Conference on Machine Learning*. 2014.

doc2vec

Training:

- A set of words (context) are used to predict one word in the same context
- The ***document*** gives a further context indication to the input words
- The model learns ***shared representations*** for words and ***unique paragraph vectors*** for documents



doc2vec

Training:

- A set of words (context) are used to predict one word in the same context
- The ***document*** gives a further context indication to the input words
- The model learns ***shared representations*** for words and ***unique paragraph vectors*** for documents

Inference:

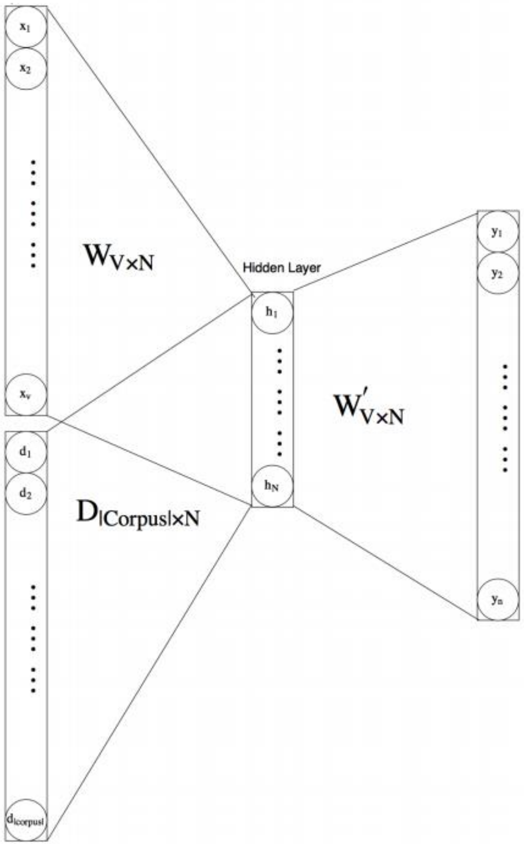
- The paragraph vectors are inferred by ***fixing the word vectors*** and training the new paragraph vector until converge

doc2vec

Training

Input words

Input document

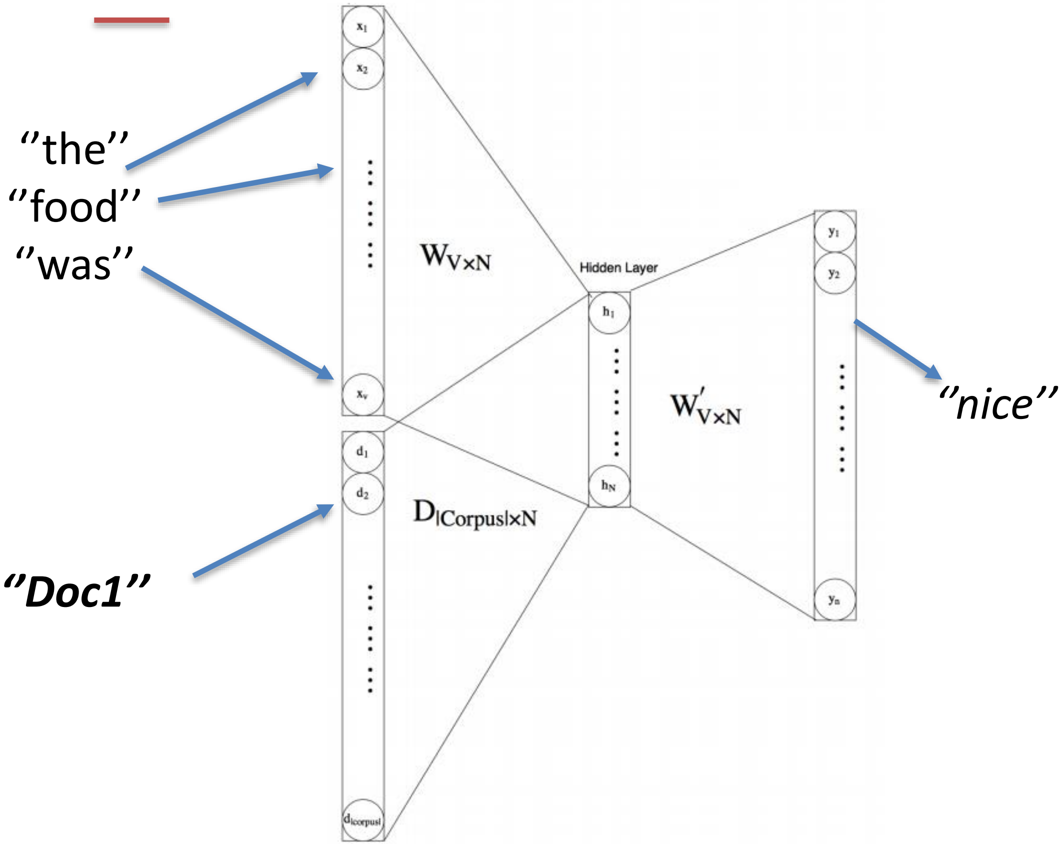


Output layer

doc2vec

Doc1: *"The food was nice"*

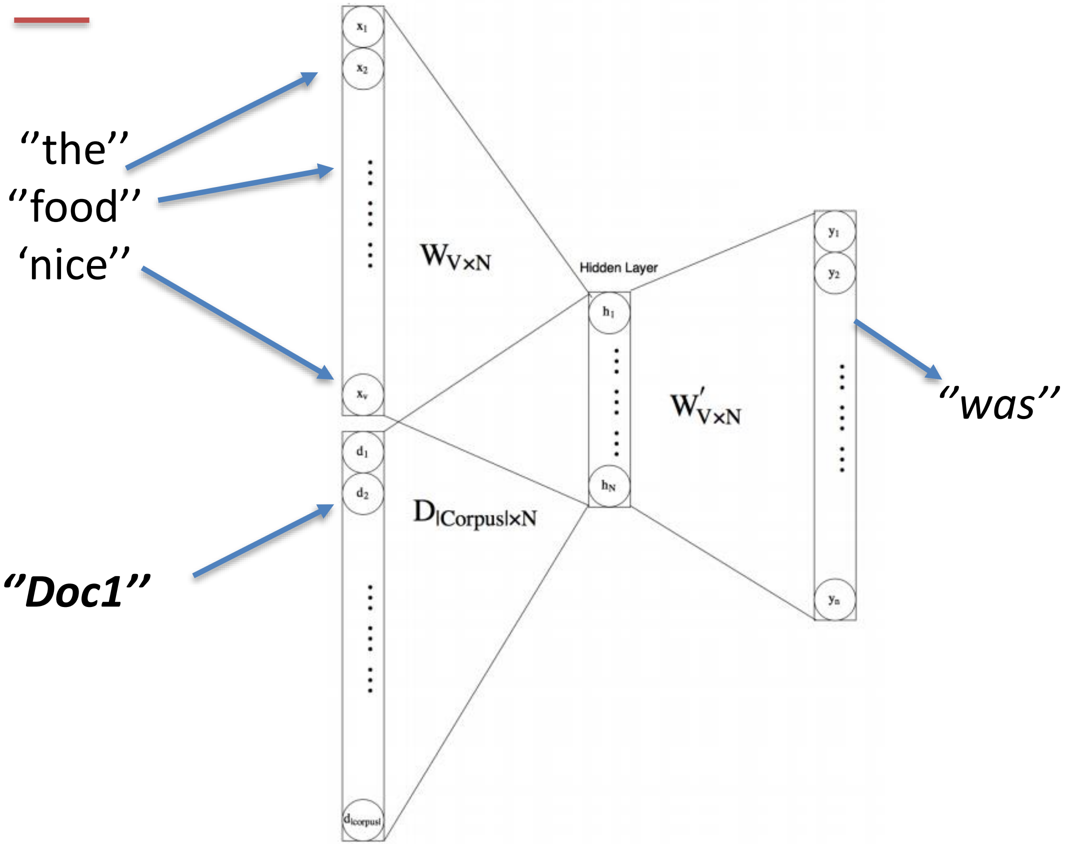
Training



doc2vec

Doc1: *"The food was nice"*

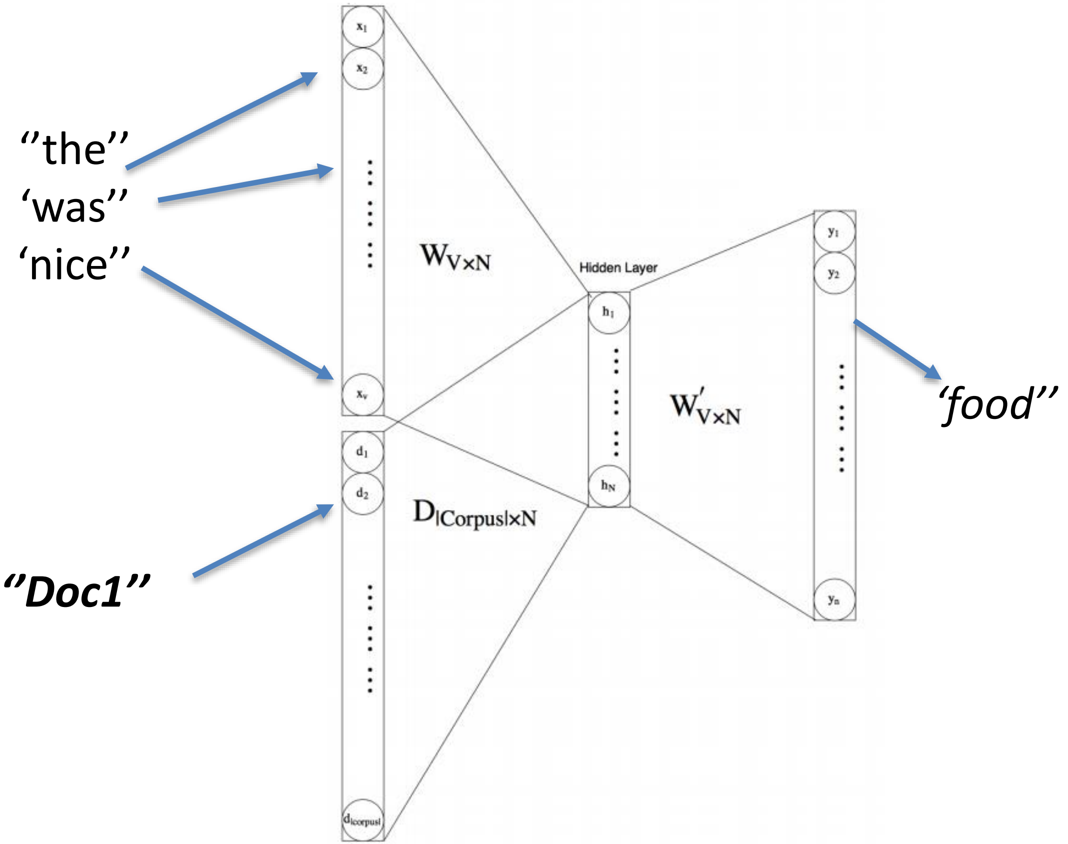
Training



doc2vec

Doc1: *"The food was nice"*

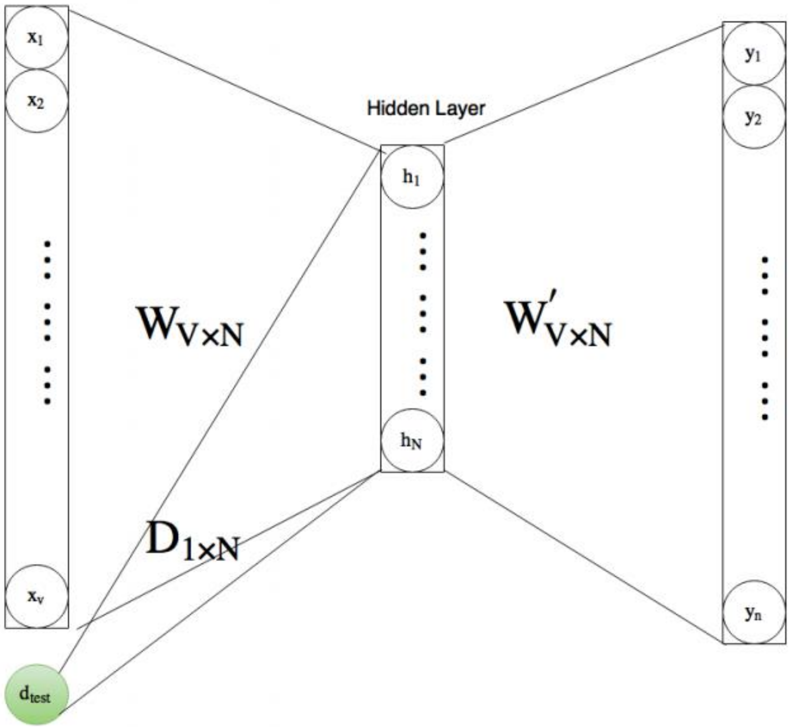
Training



doc2vec

Inference

Input words



Output layer

Test document

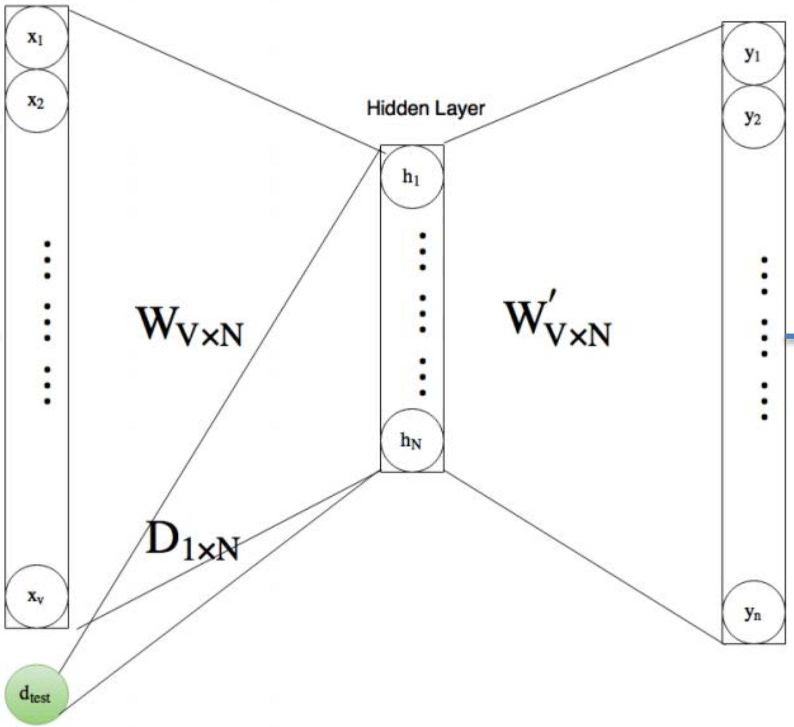
doc2vec

***???:** "The food was nice"*

Inference

"the"
"food"
"was"

"???"

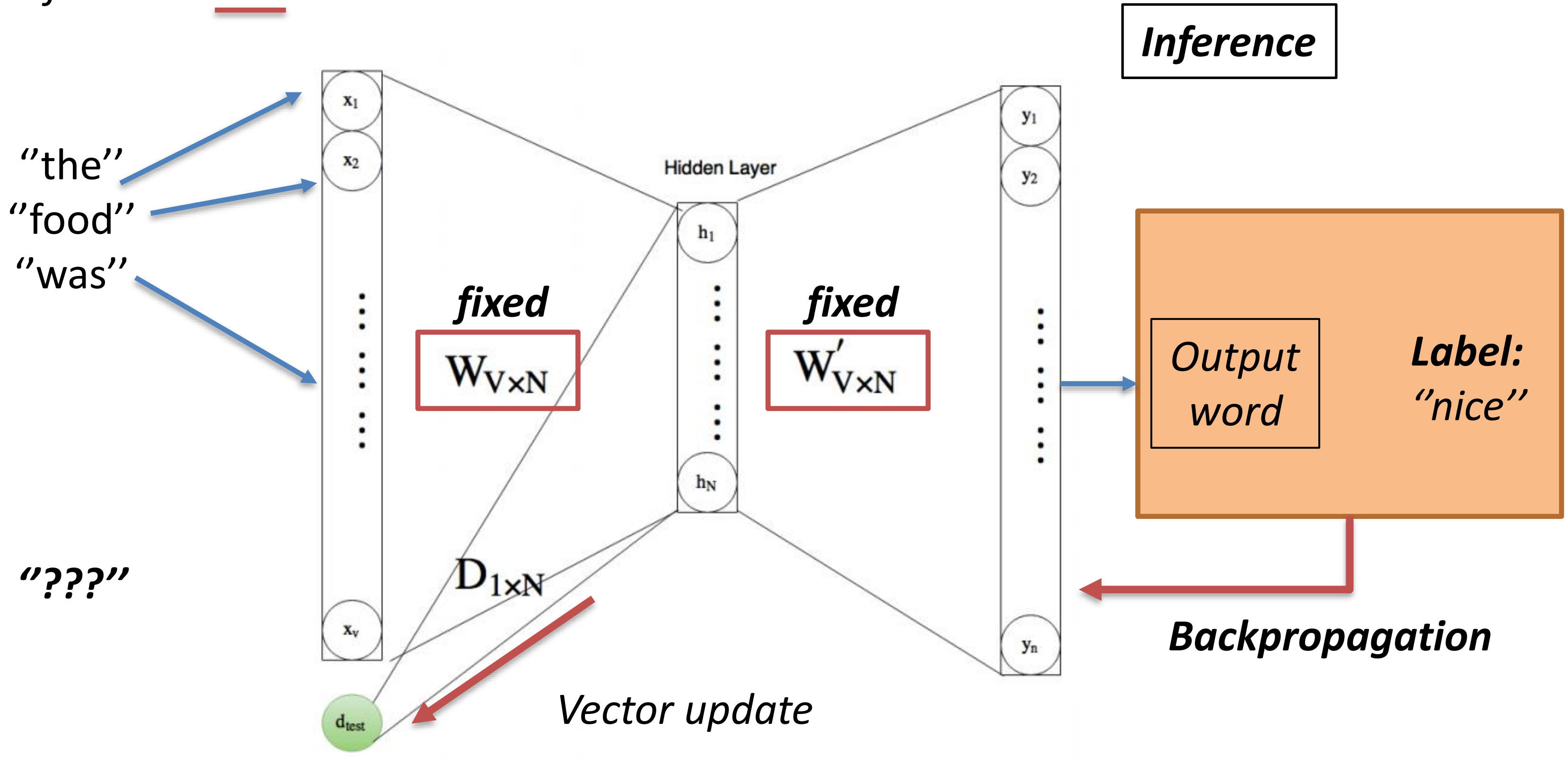


Output word

Label:
"nice"

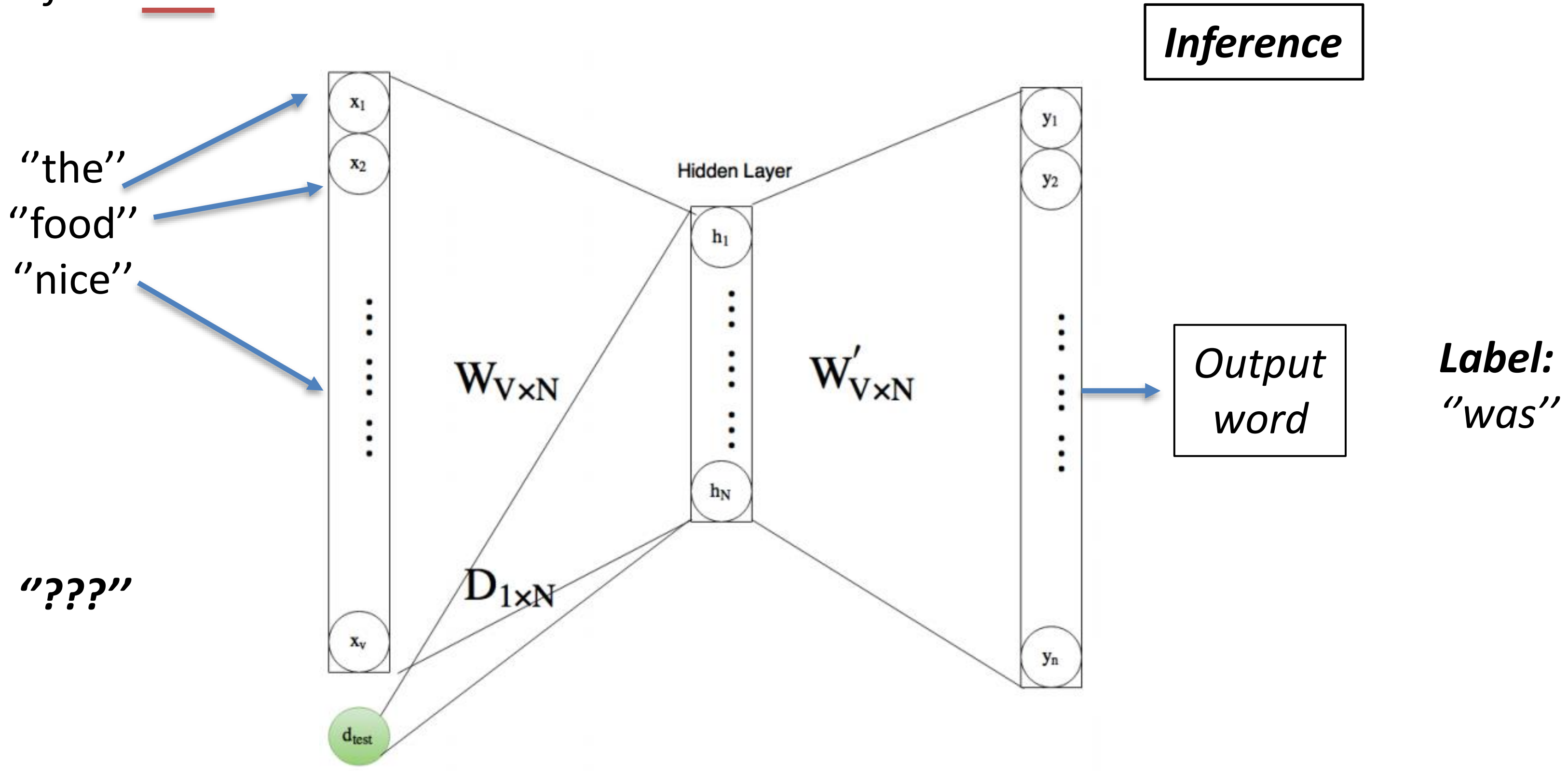
doc2vec

???: "The food was nice"



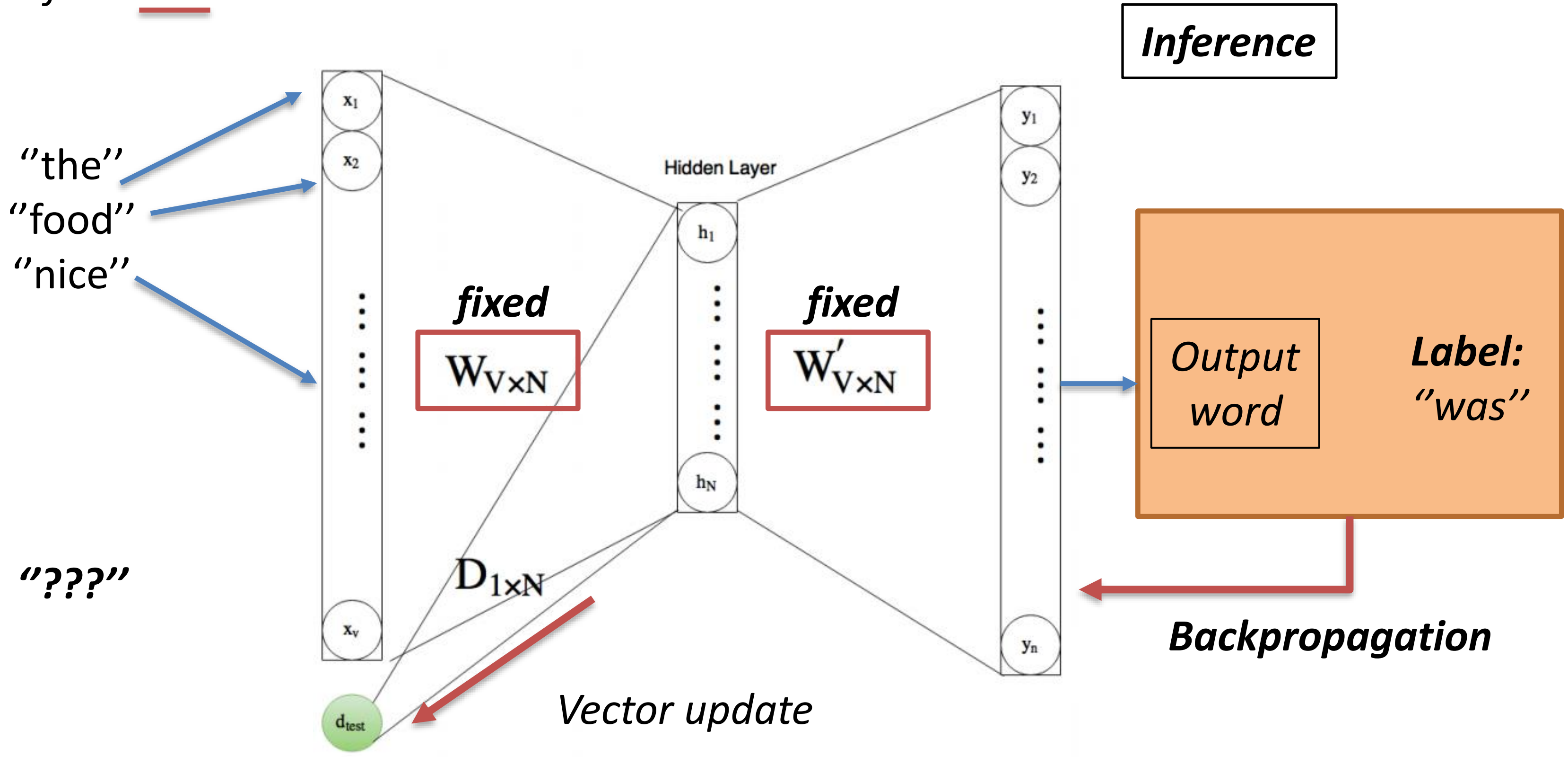
doc2vec

???: *"The food was nice"*



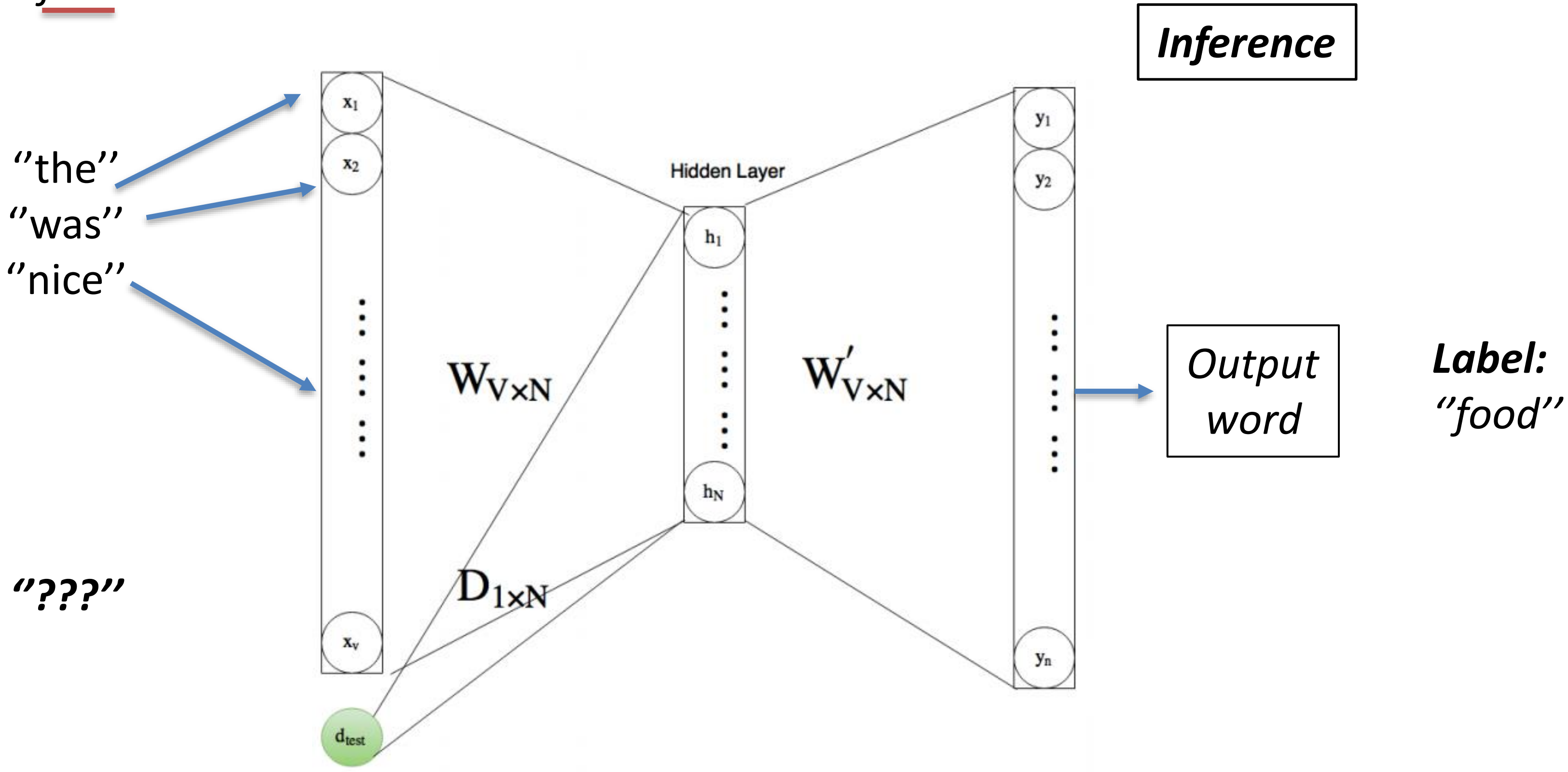
doc2vec

???: *"The food was nice"*



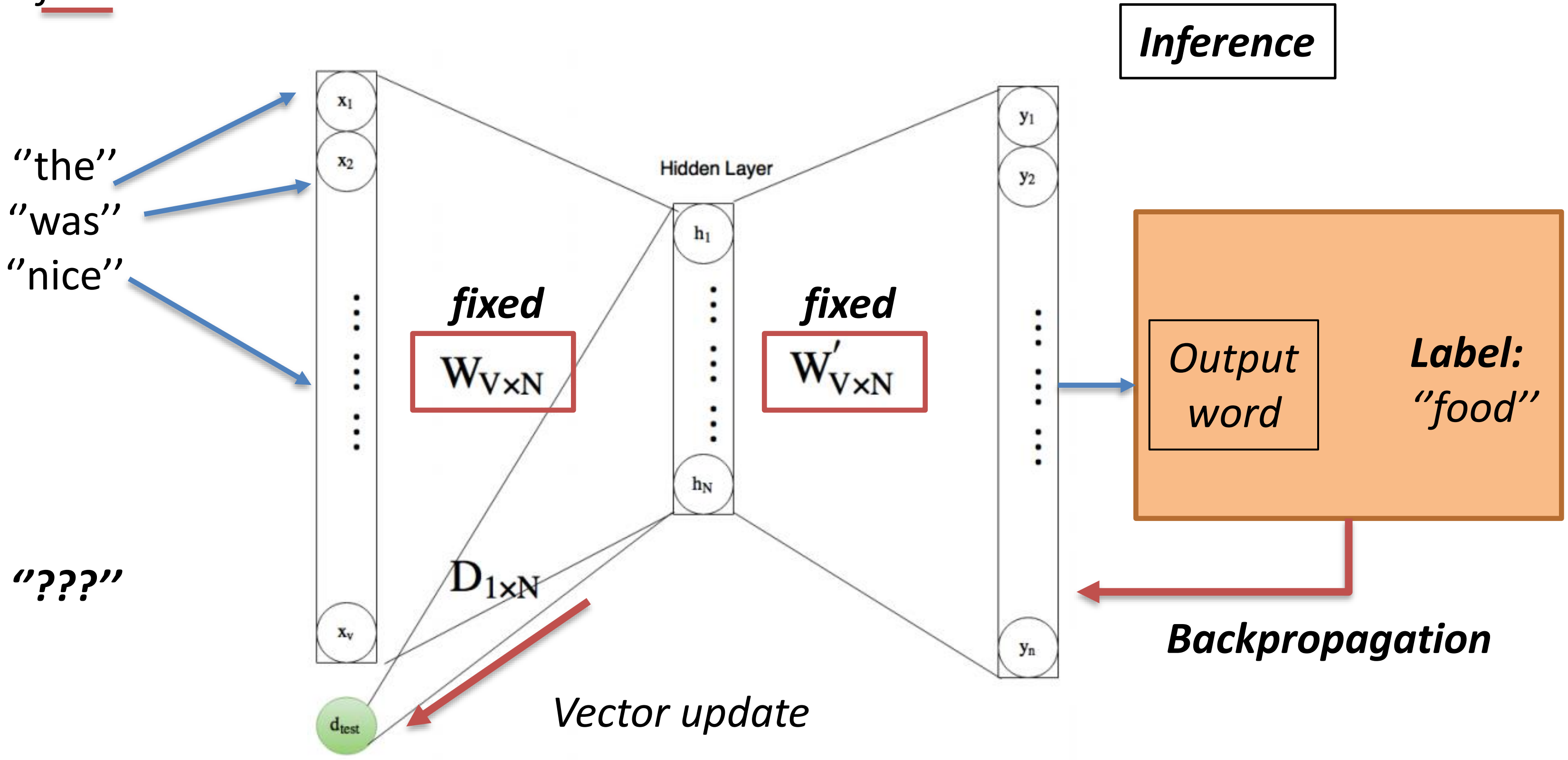
doc2vec

???: *"The food was nice"*



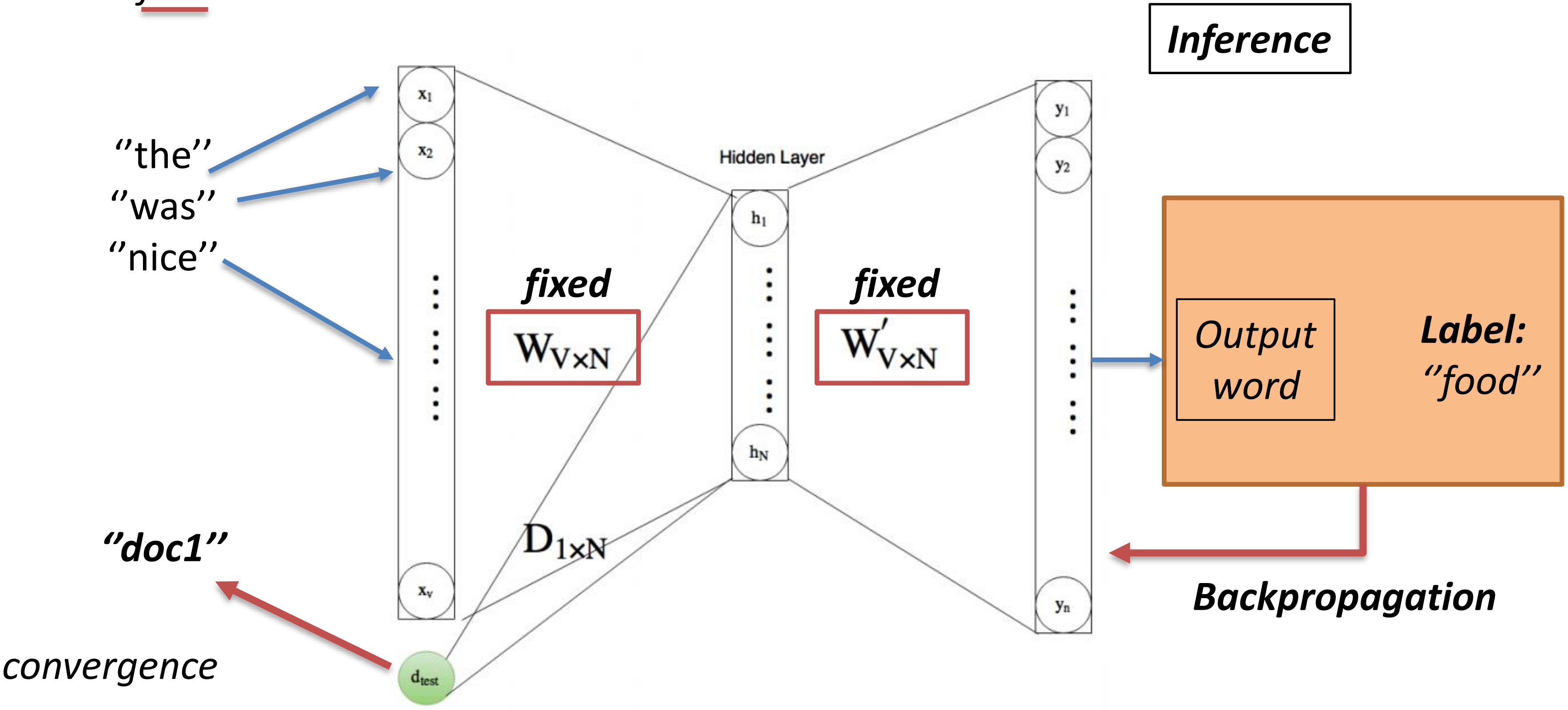
doc2vec

***???:** "The food was nice"*



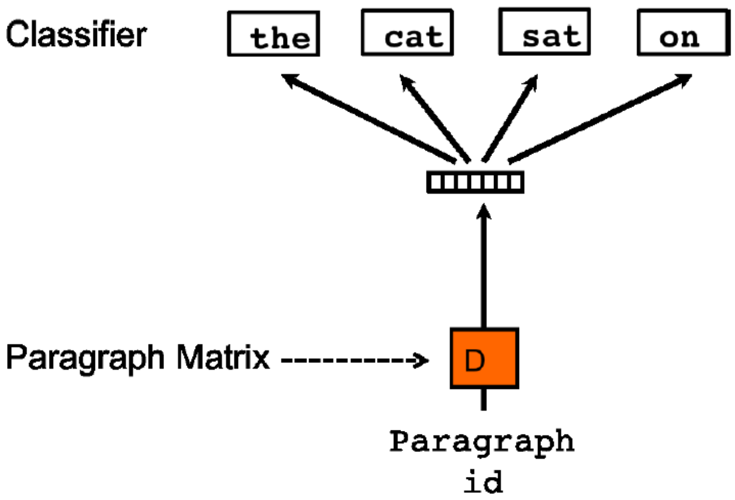
doc2vec

???: *"The food was nice"*

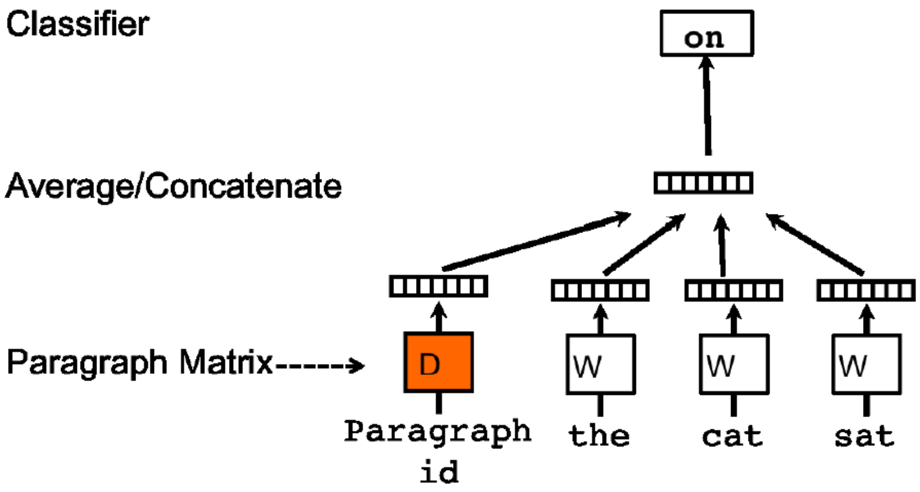


doc2vec

Distributed bag of words (PV-DBOW)



Distributed Memory Model of Paragraph Vectors (PV-DM)



doc2vec

To sum up:

- A model able to learn fixed-length representations from variable-length pieces of texts
- Word representations (shared) are learned in conjunction with paragraph representations (unique)
- ***PV-DM:*** predicts a word given a words+doc context
- ***PV-DBOW:*** predicts a set of words given a document as a context
- Document inference: the model is trainied fixing the word vectors until convergence.

word2vec & doc2vec

Genism Python Library: https://radimrehurek.com/gensim/

```
>>> from gensim.test.utils import common_texts, get_tmpfile
>>> from gensim.models import Word2Vec
>>>
>>> path = get_tmpfile("word2vec.model")
>>>
>>> model = Word2Vec(common_texts, size=100, window=5, min_count=1, workers=4)
>>> model.save("word2vec.model")
```

```
>>> from gensim.test.utils import common_texts
>>> from gensim.models.doc2vec import Doc2Vec, TaggedDocument
>>>
>>> documents = [TaggedDocument(doc, [i]) for i, doc in enumerate(common_texts)]
>>> model = Doc2Vec(documents, vector_size=5, window=2, min_count=1, workers=4)
```