@fgiobergia
fgiobergia

and practical

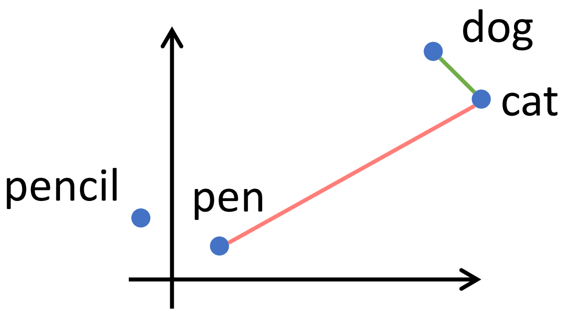
A quick introduction to word embeddings

Flavio Giobergia

Word representations using vectors

V dimensions

Cat	1	0	0	0
Dog	0	1	0	0
Pen	0	0	1	0
Pencil	0	0	0	1



1-of-V vectors

- Problems
  - Sparse representation
  - High-dimensional representation
  - All vectors are orthogonal
  - No semantic relationships preserved
    - Ideally, dist(cat, dog) < dist(cat, pen)

	doc1	doc2	doc3	doc4
Cat	1.1	0	1.2	0.1
Dog	2.1	0	0	2.3
Pen	0	1.1	0	1.4
Pencil	0	2.2	0.1	0

TF, TF-IDF, LSA, …

What would a machine do?

- 3 easy steps
    1. Create a simple task
    2. Train a model to solve the task
    3. ~~Steal~~ Borrow the internal representation of the model

1. Create a simple task

- Step 1.1
  - Get (a lot of) data (sentences)
    - *Sometimes, dogs chase their tail because…*
    - *When your cat holds her tail high…*
    - …

Task # 1 context → word

Task # 2 word → context

Sometimes, dogs chase their tail because…

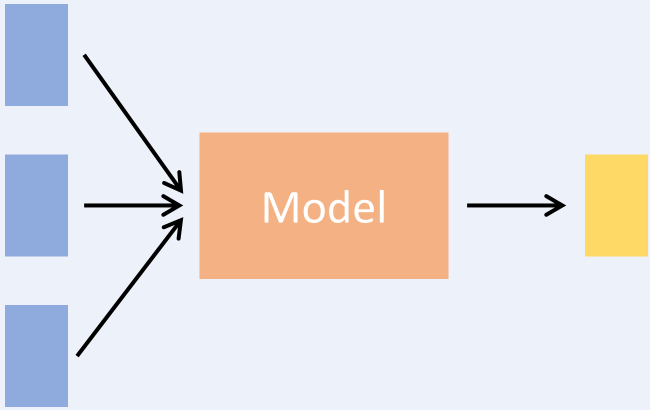
Sometimes, <blank> chase their tail because → *dogs*

Sometimes, dogs <blank> their tail because → *chase*

Sometimes, dogs chase <blank> tail because → *their*

Sometimes, dogs chase their <blank> because → *tail*

…



dogs → *Sometimes, chase*

- *dogs* → *Sometimes*
- *dogs* → *chase*

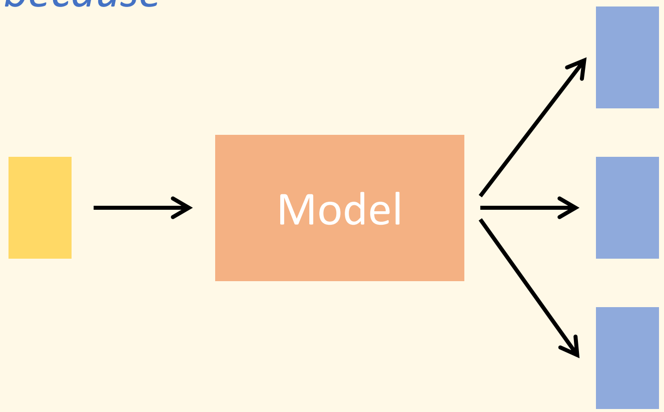
chase → *dogs, their*

- *chase* → *dogs*
- *chase* → *their*

their → *chase, tail*

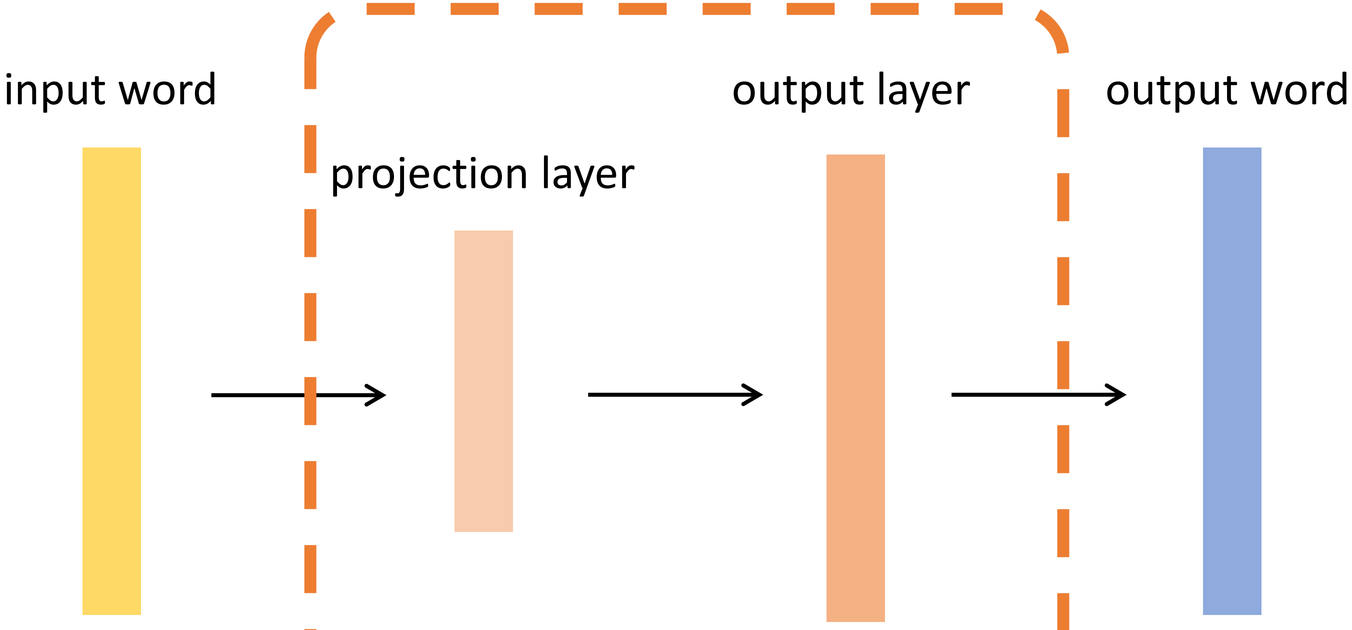
tail → *their, because*

…



Step 2: Train a model to solve the task

(dog → tail)



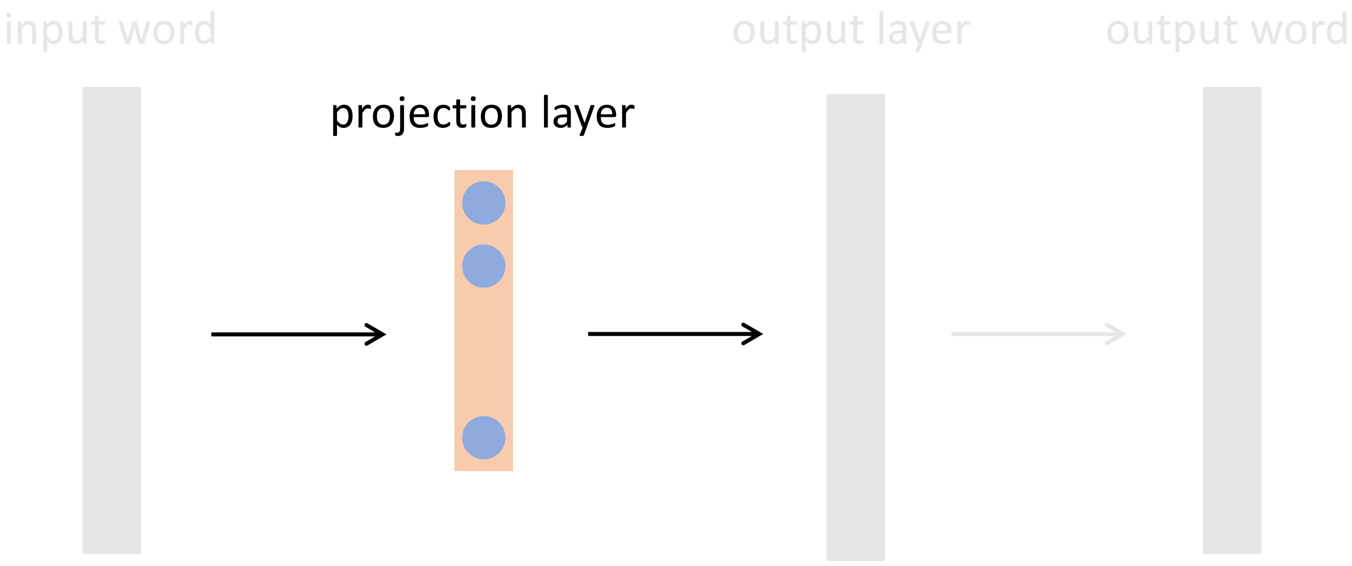
Input word:

cat	0
tail	0
dog	1
pen	0

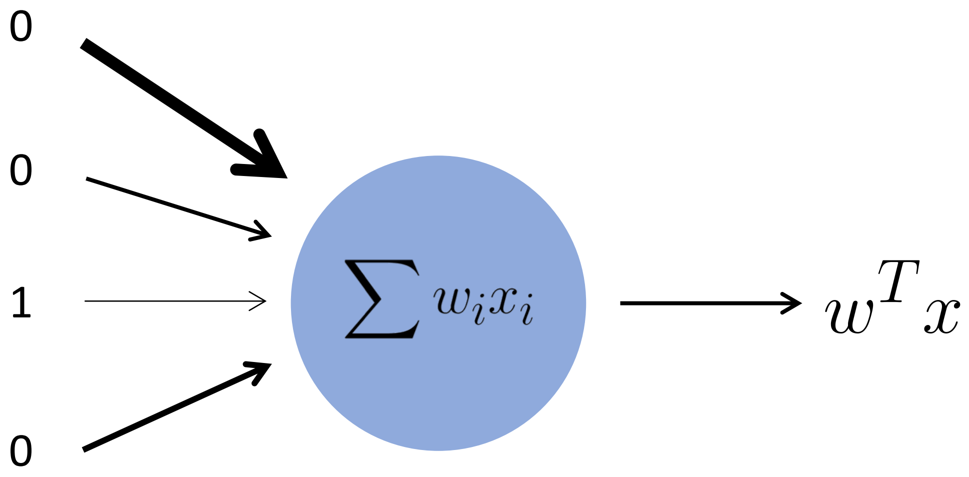
Output word:

cat	0
tail	1
dog	0
pen	0

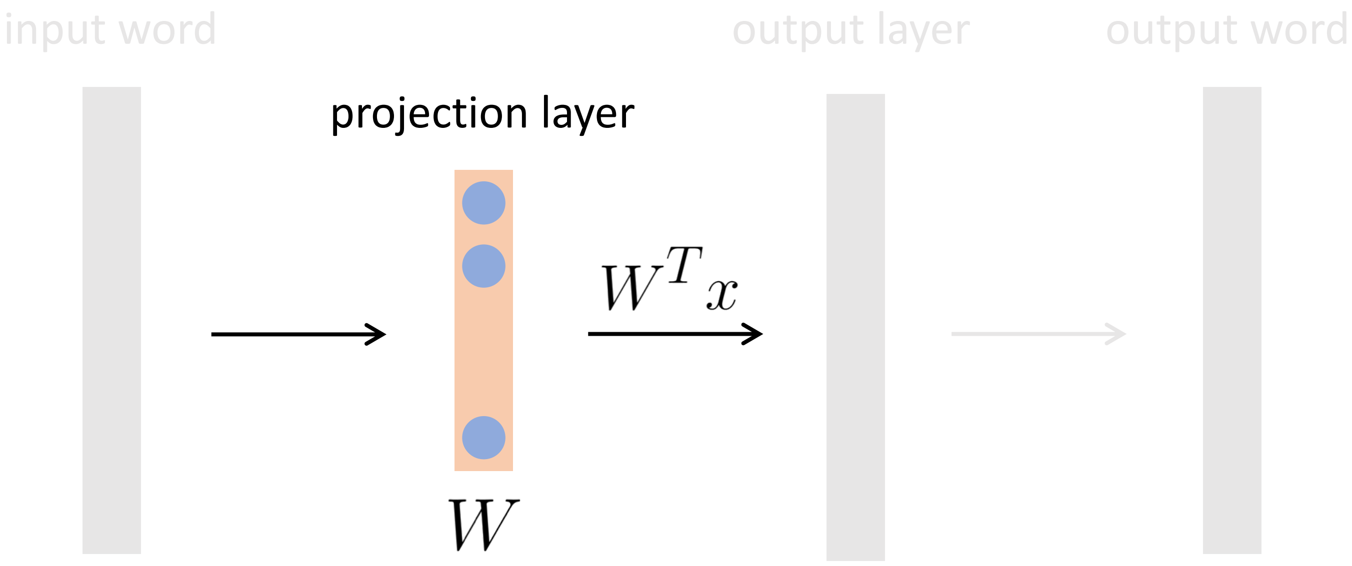
Step 2: Train a model to solve the task



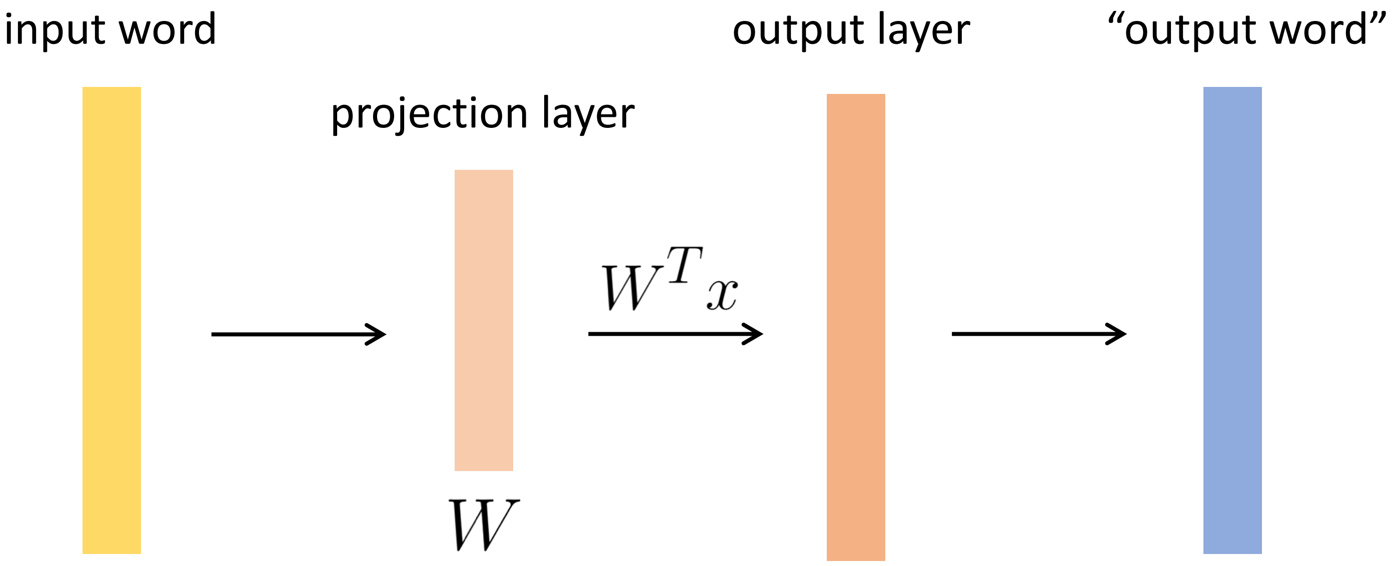
Step 2: Train a model to solve the task



Step 2: Train a model to solve the task



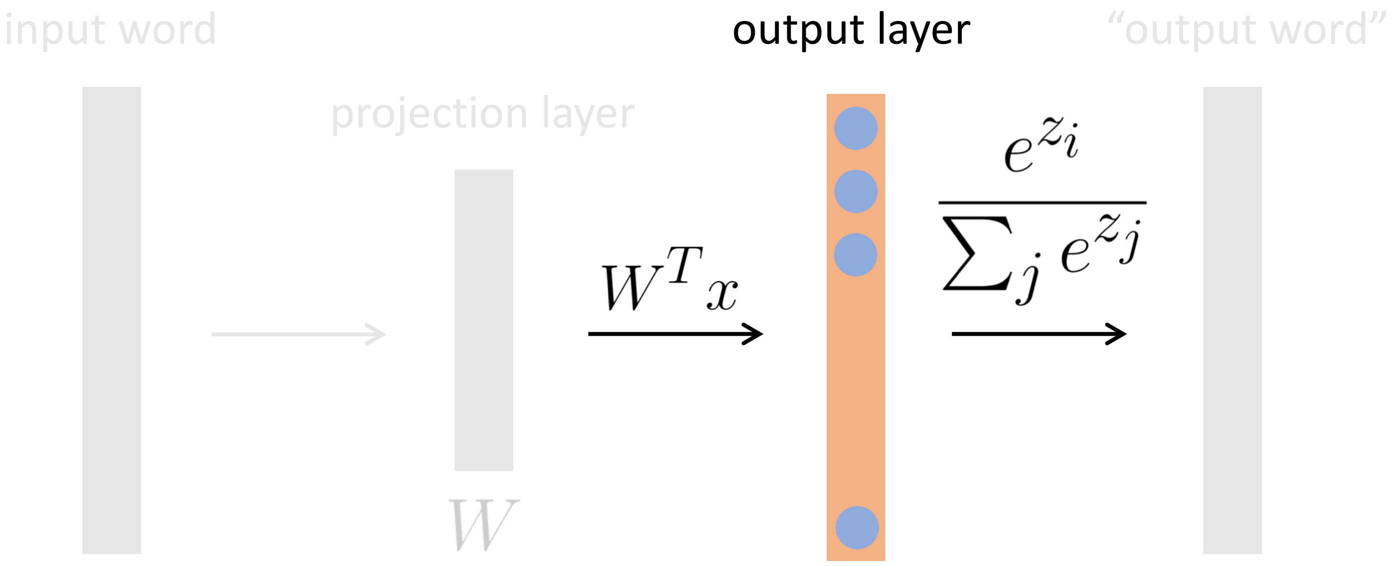
Step 2: Train a model to solve the task



cat	0
tail	0
dog	1
pen	0

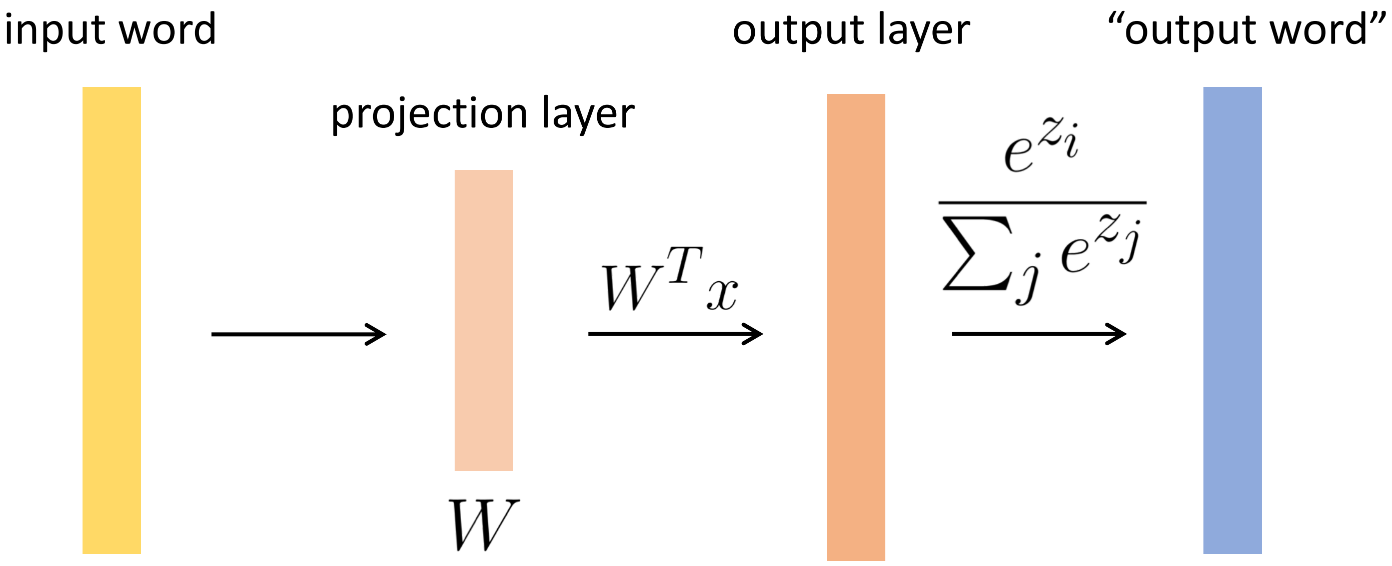
cat	0.15
tail	0.75
dog	0.09
pen	0.01

Step 2: Train a model to solve the task



“Final” architecture

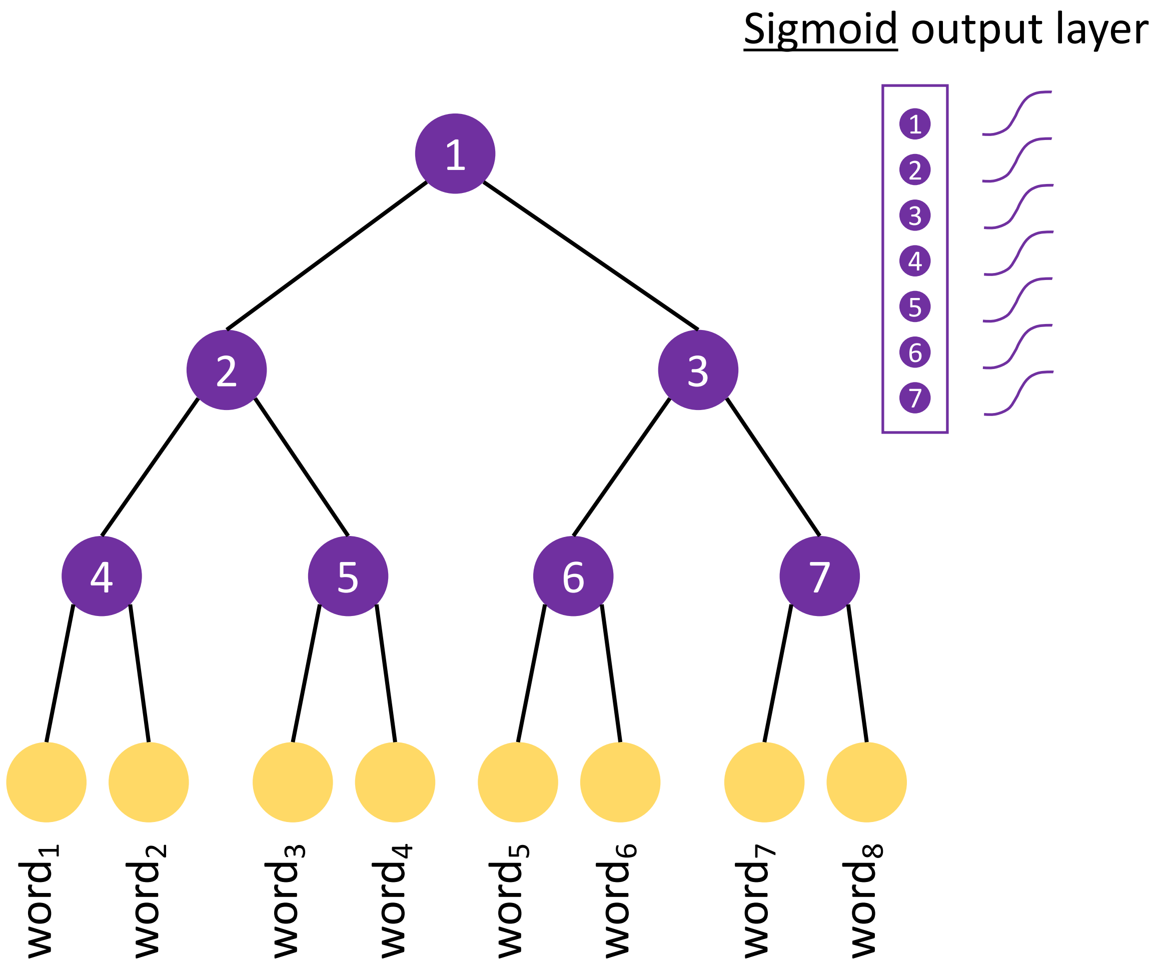
Bengio, Yoshua, et al. "A neural probabilistic language model." *Journal of machine learning research* 3.Feb (2003): 1137-1155.
(sort of)



cat	0
tail	0
dog	1
pen	0

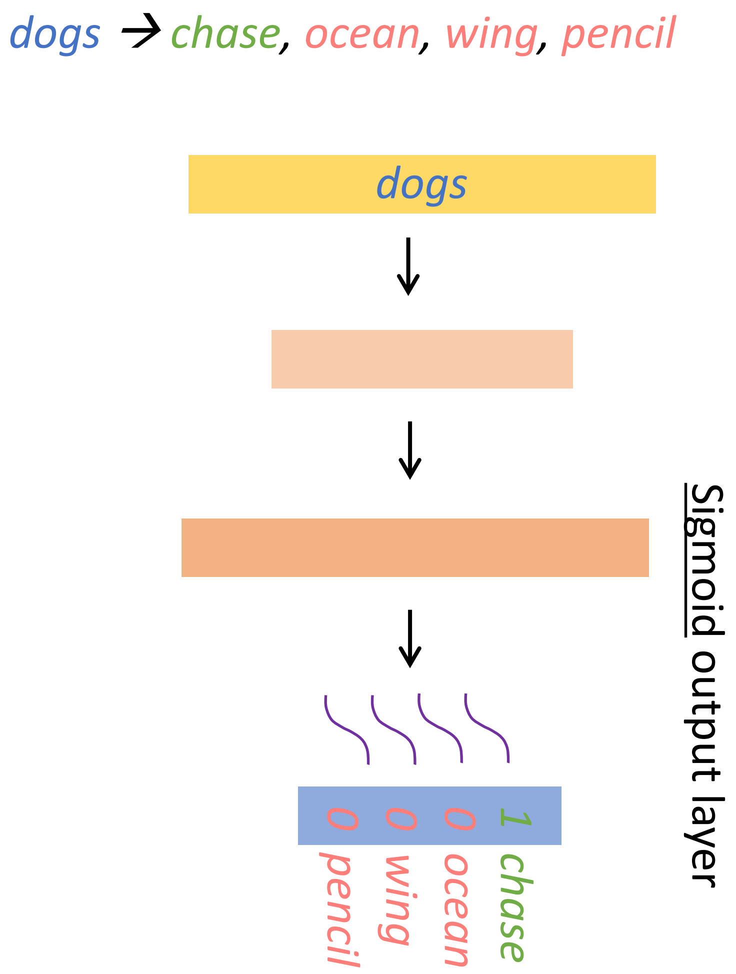
cat	0.15
tail	0.75
dog	0.09
pen	0.01

Hierarchical softmax



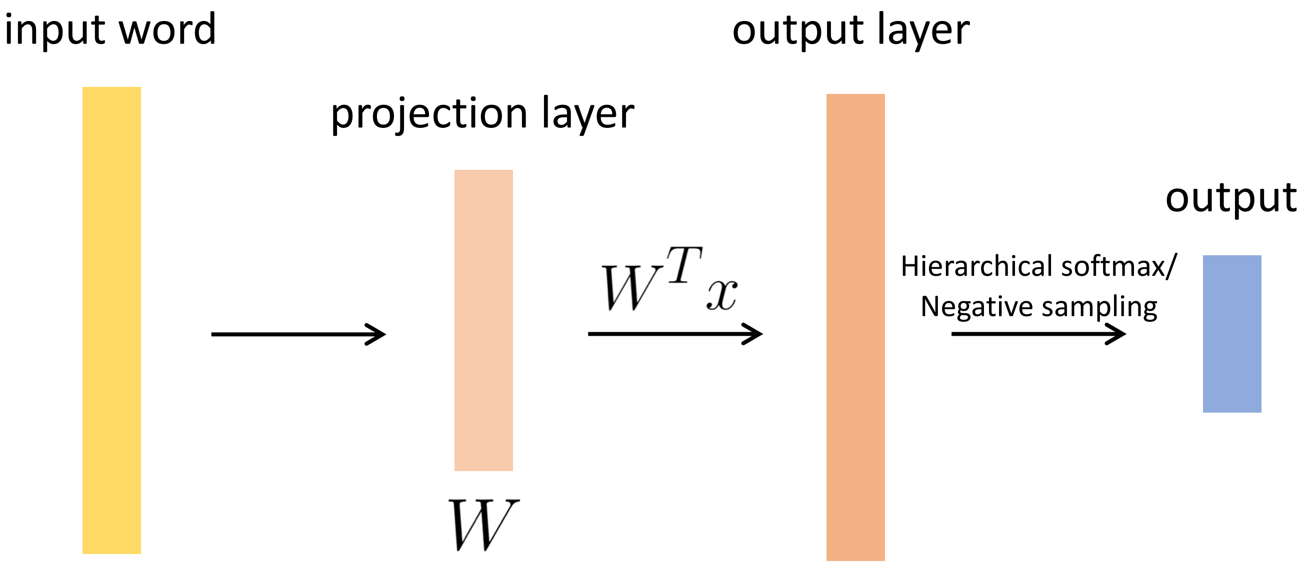
Morin, Frederic, and Yoshua Bengio. "Hierarchical probabilistic neural network language model." *Aistats*. Vol. 5. 2005.

Negative sampling



Revised architecture (word2vec)

Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems* 26 (2013): 3111-3119.



cat	0
tail	0
dog	1
pen	0

Step 3: What does W contain?

$$W \quad \begin{bmatrix} \boldsymbol{w_1} & \boldsymbol{w_2} & \dots & \boldsymbol{w_n} \end{bmatrix}$$

Step 3: What does W contain?

W

w_1	w_2	…	w_n

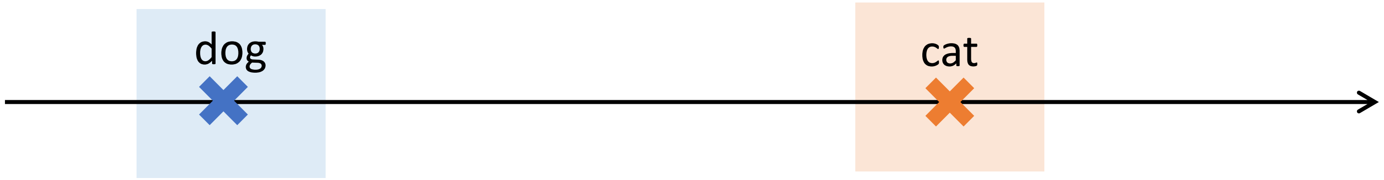
r_1
r_2
r_3
r_4
.
.
.
r_V

Some intuition

- How do we map *cat* and *dog* in 1D so that they both output *tail*?
  - (cat → tail), (dog → tail)

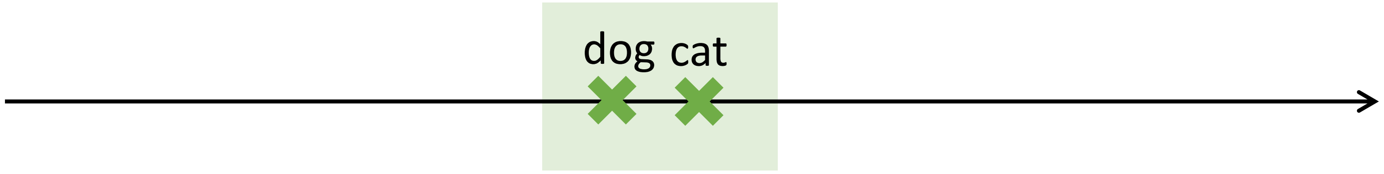
Some intuition

- How do we map *cat* and *dog* in 1D so that they both output *tail*?
  - (cat → tail), (dog → tail)

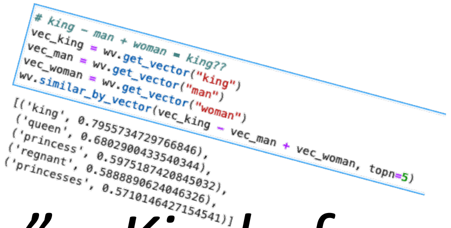


Some intuition

- How do we map *cat* and *dog* in 1D so that they both output *tail*?
  - (cat → tail), (dog → tail)



word2vec



1. *"king – man + woman = queen"... Kind of*
2. Huge splash in NLP world
3. Learns from raw text
4. Pretty simple algorithm There can be a leaner machine learning!
5. Comes pretrained [GloVe] [fastText]

Slide based on Christopher Moody (@chrisemoody)'s intro to word2vec