



A quick ^{and practical} introduction to word embeddings

Flavio Giobergia

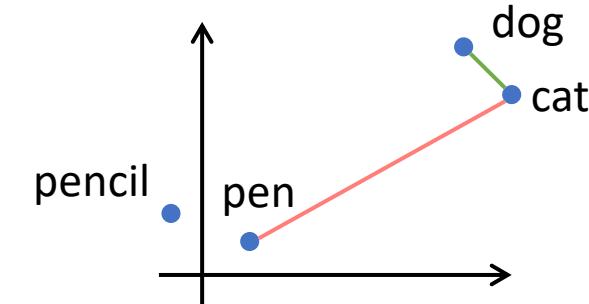
Word representations using vectors

1-of-V vectors

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

V dimensions

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



TF, TF-IDF, LSA, ...

	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

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...

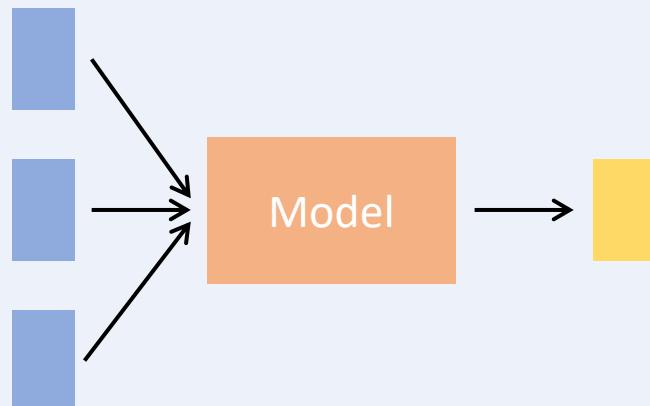
Sometimes, <blank> chase their tail because → dogs

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

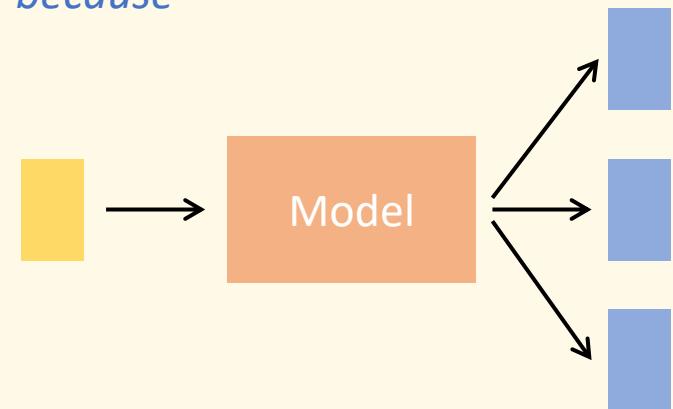
chase → dogs

chase → their

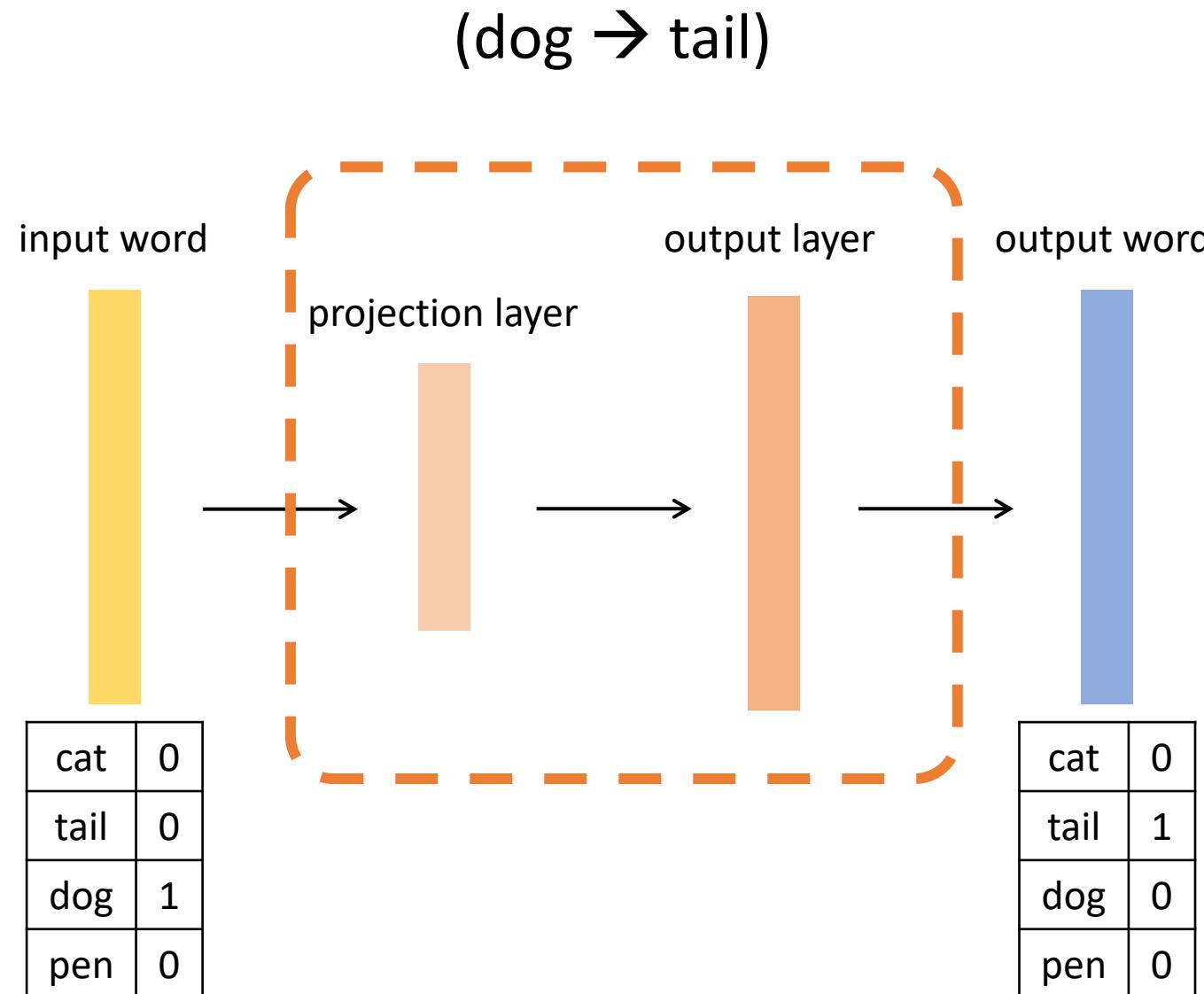
their → chase, tail

tail → their, because

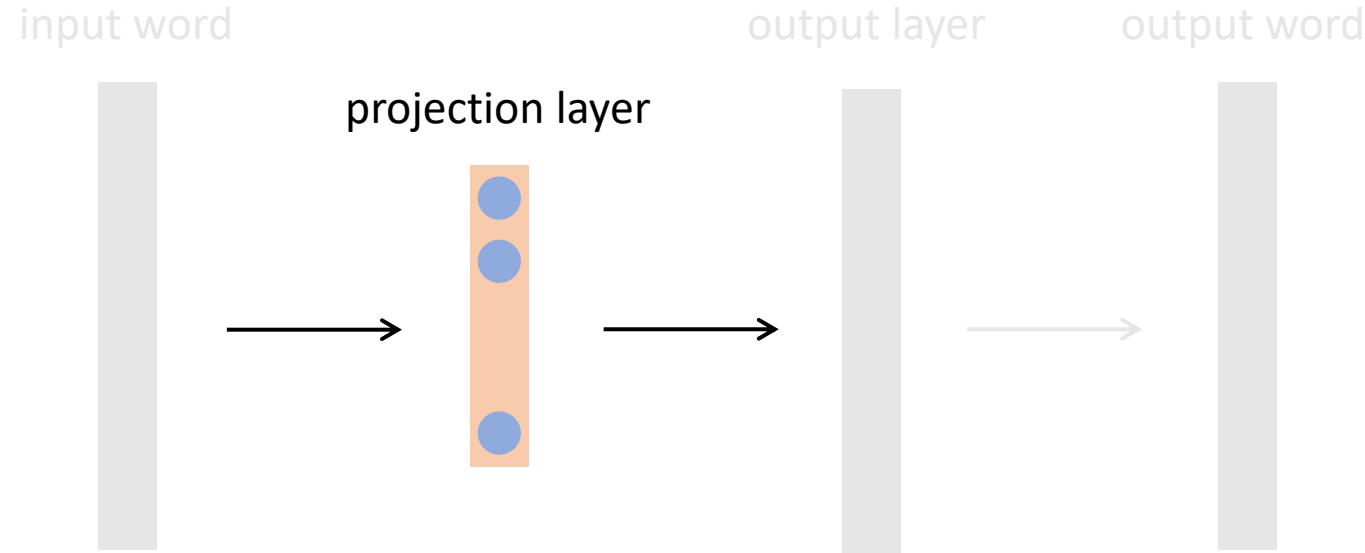
...



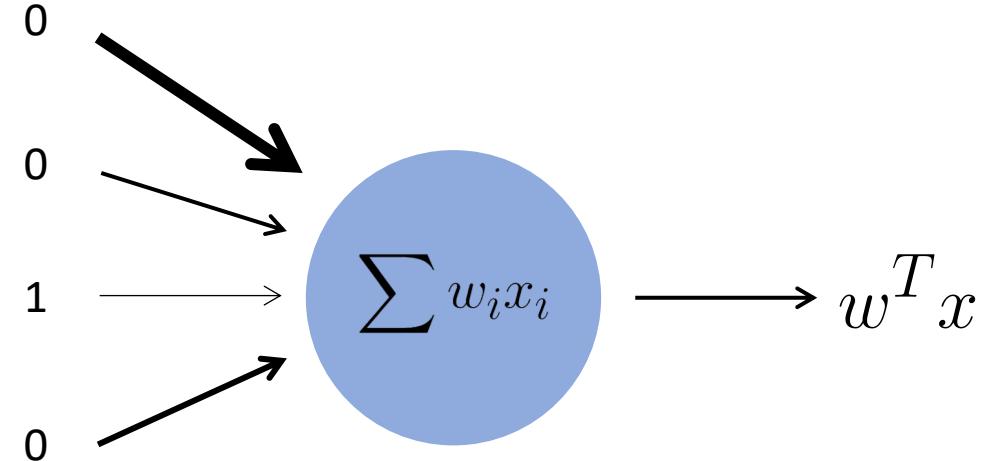
Step 2: Train a model to solve the task



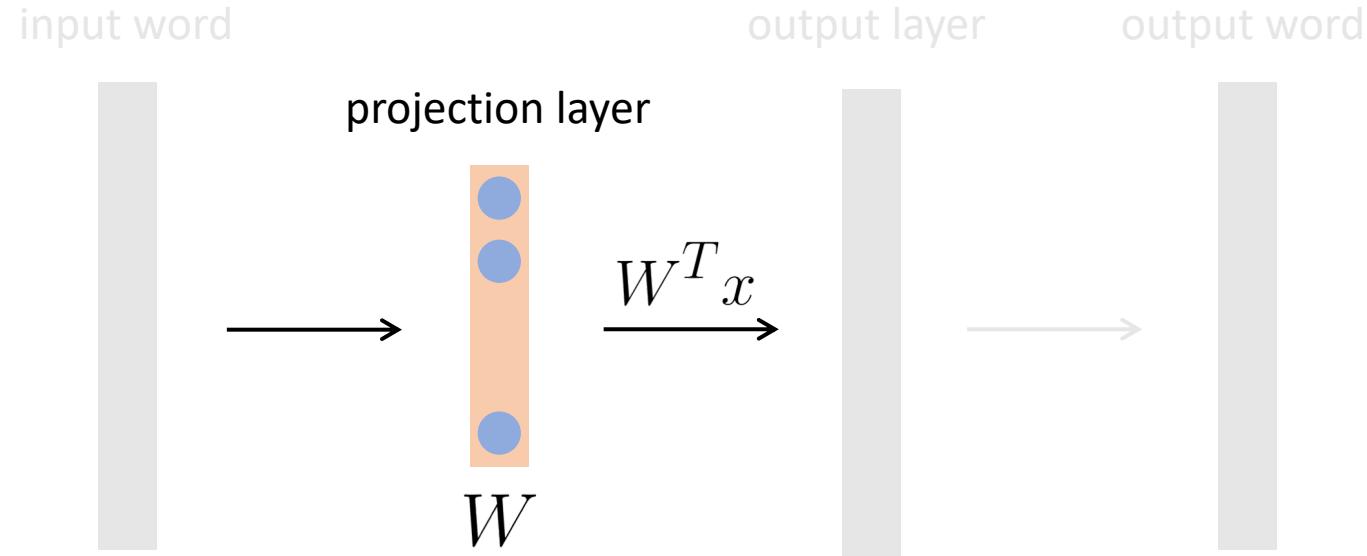
Step 2: Train a model to solve the task



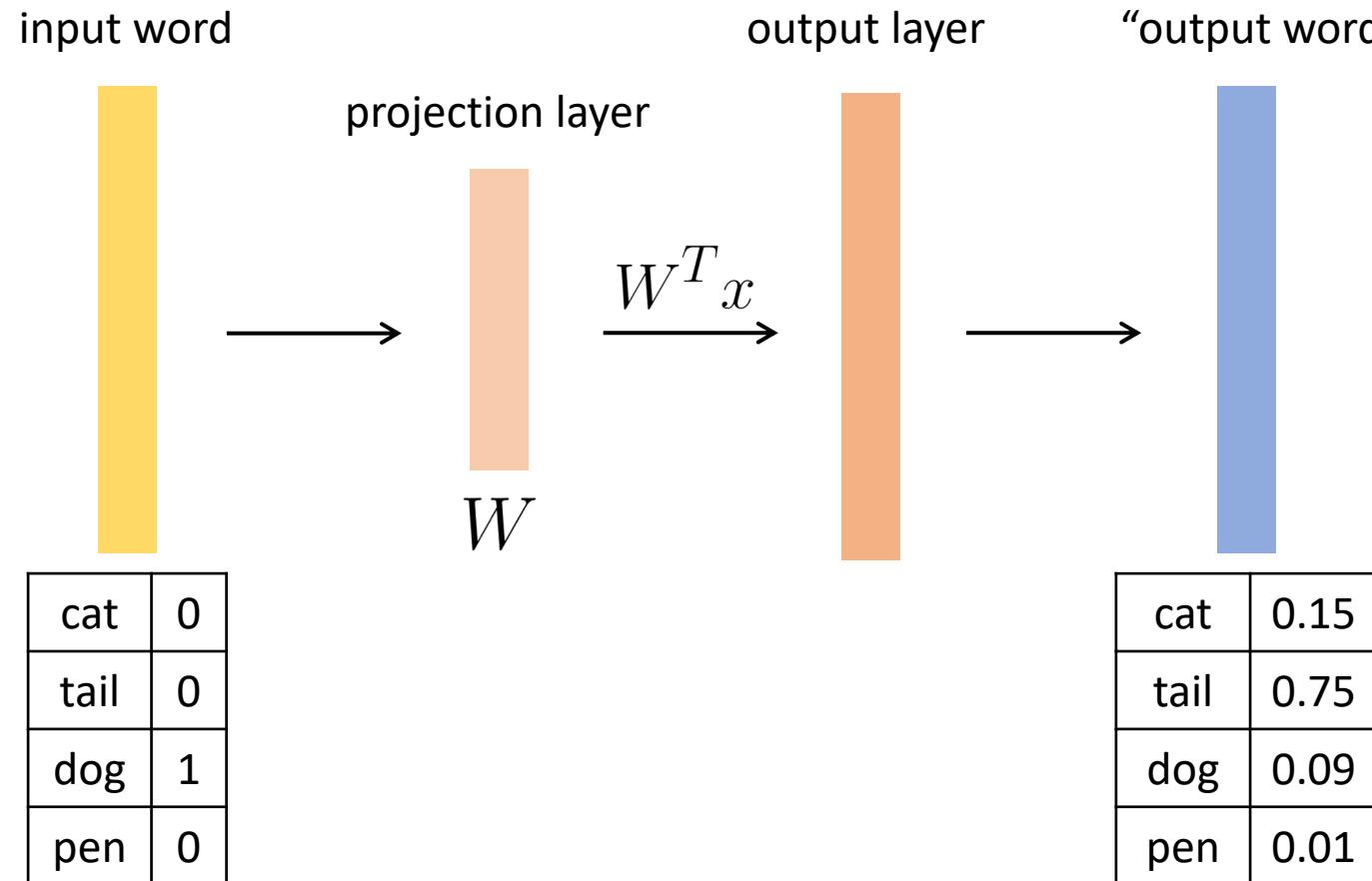
Step 2: Train a model to solve the task



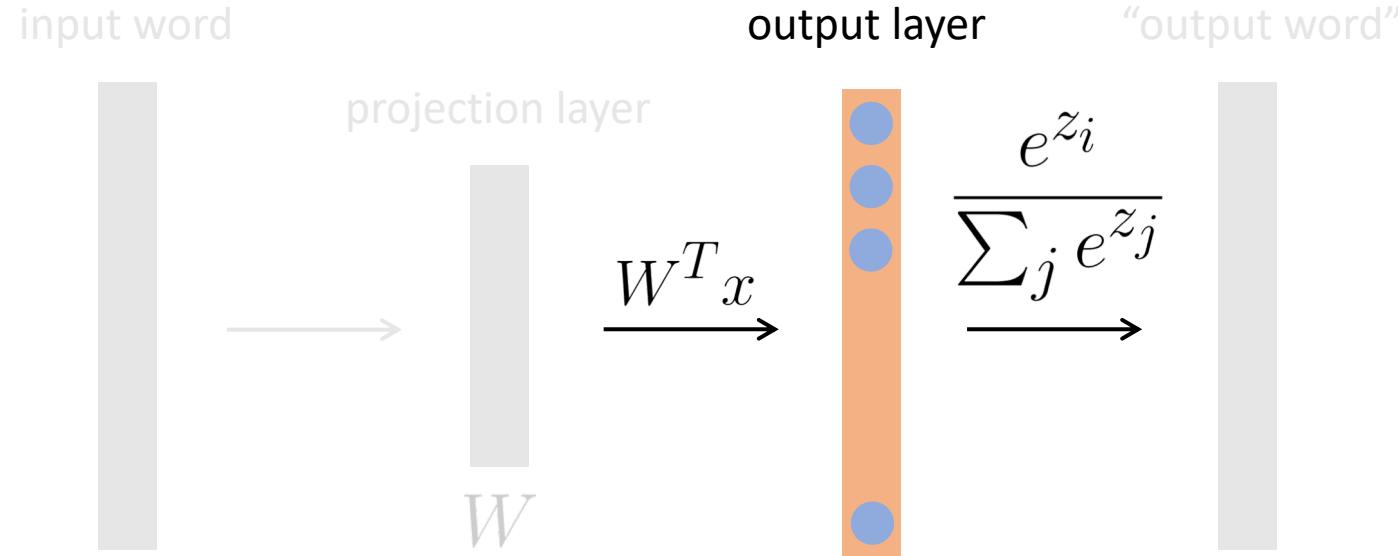
Step 2: Train a model to solve the task



Step 2: Train a model to solve the task

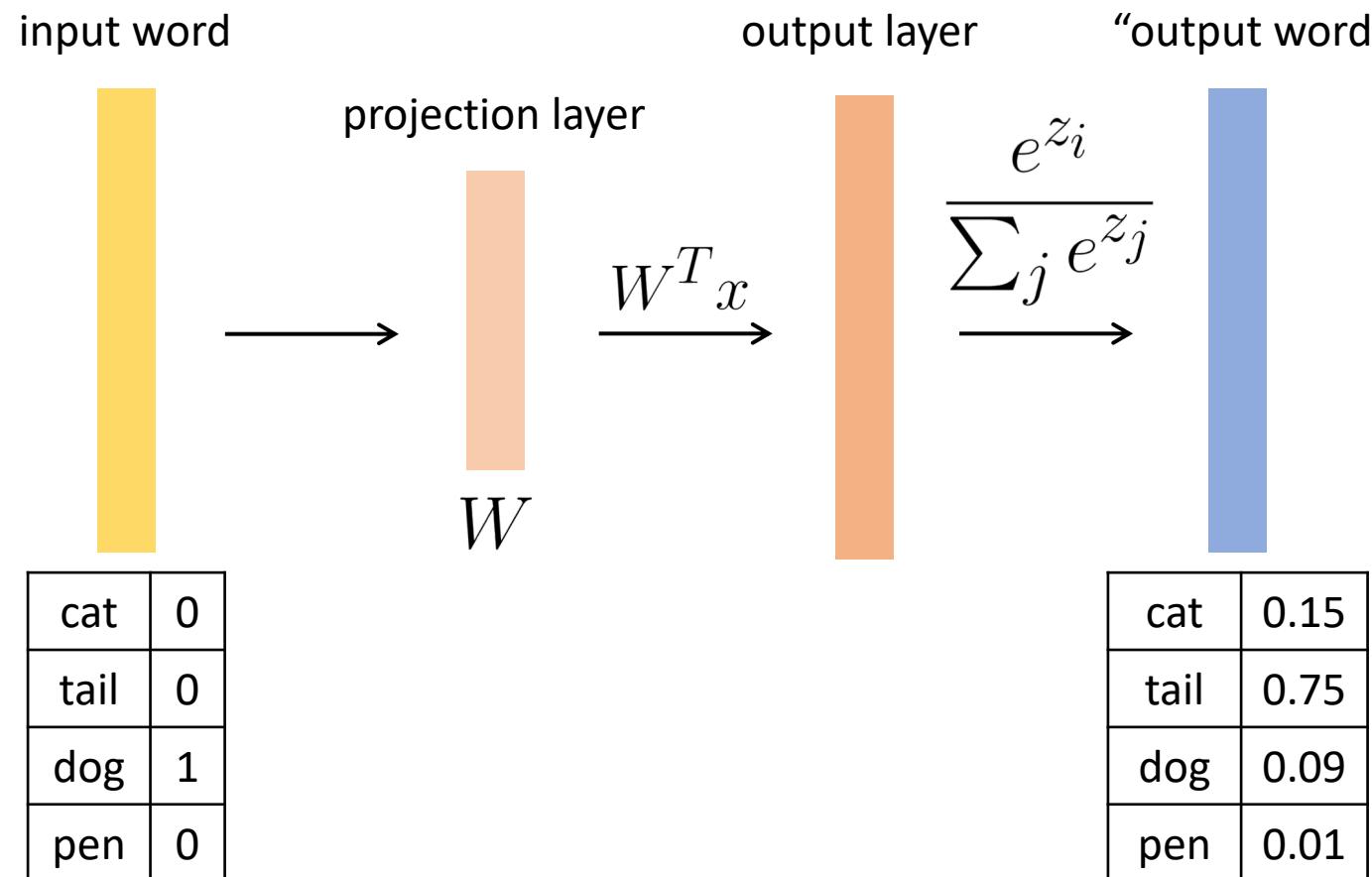


Step 2: Train a model to solve the task



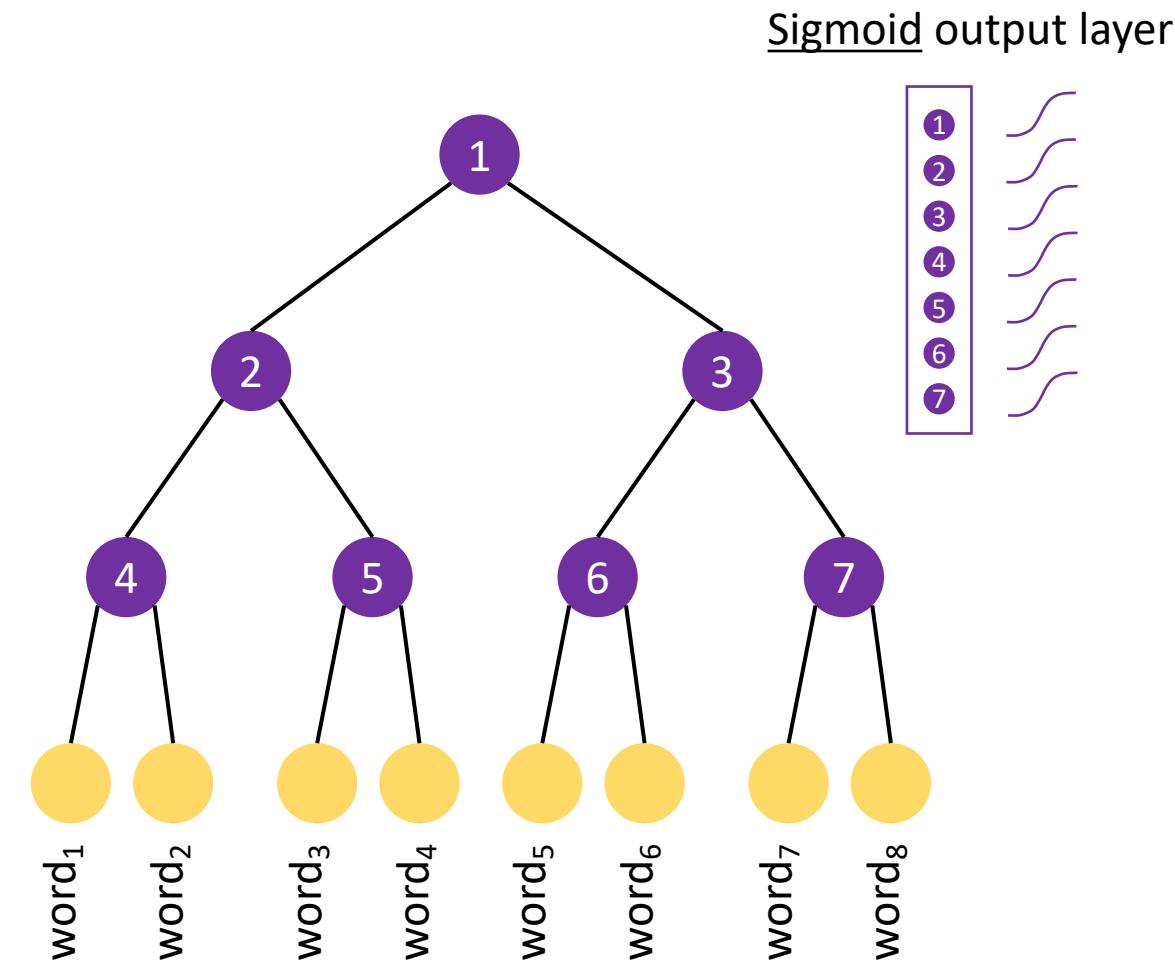
“Final” architecture

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

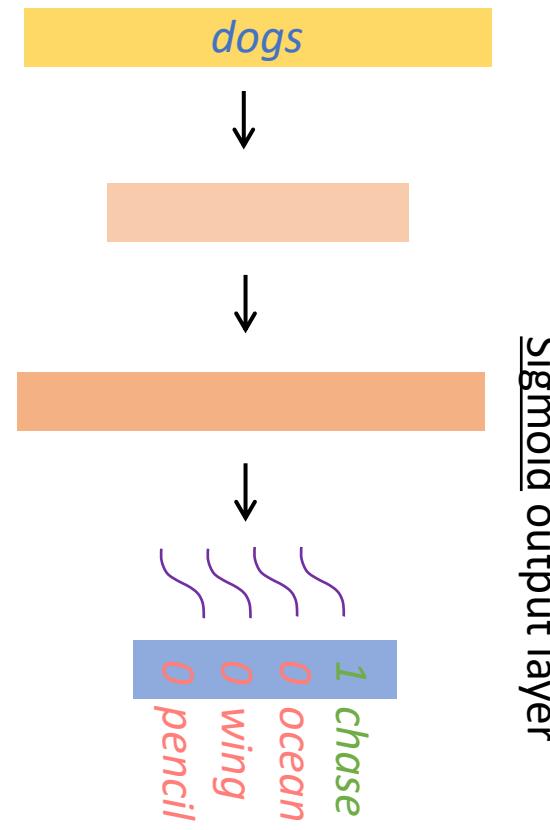


Hierarchical softmax

Negative sampling



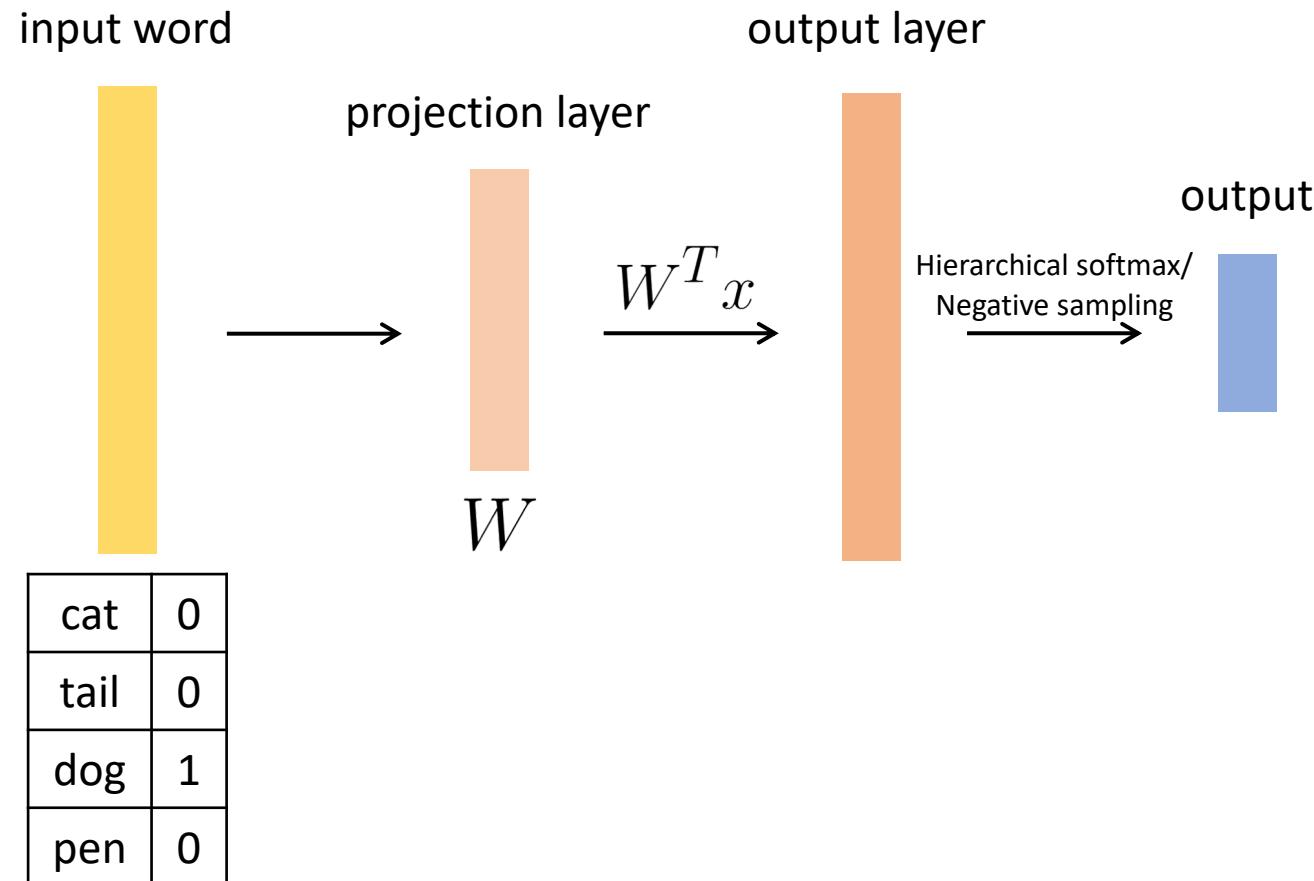
dogs → *chase, ocean, wing, pencil*



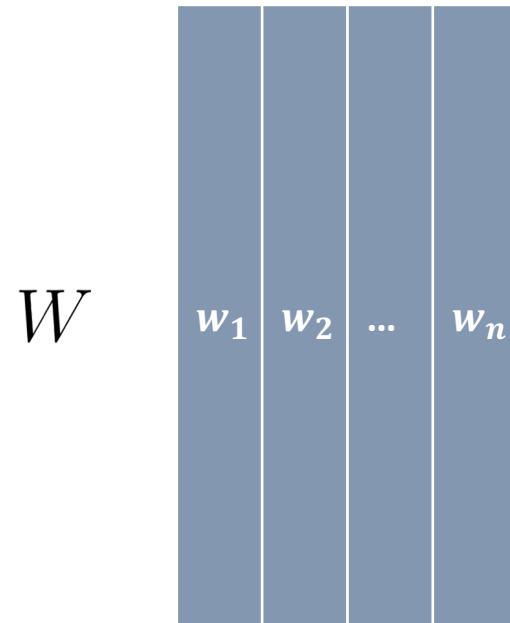
Morin, Frederic, and Yoshua Bengio. "Hierarchical probabilistic neural network language model." *Aistats*. Vol. 5. 2005.

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.



Step 3: What does W contain?



Step 3: What does W contain?



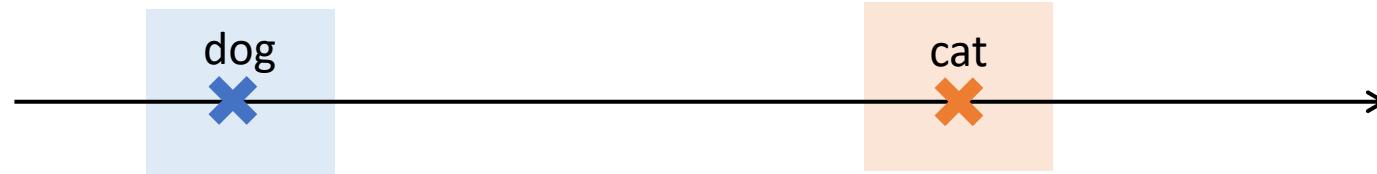
Some intuition

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



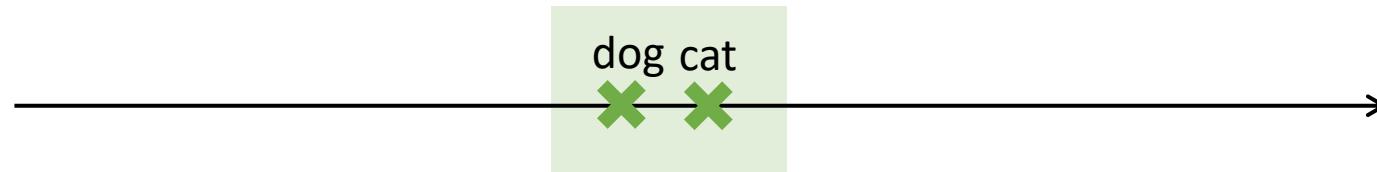
Some intuition

- 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. “ $\text{king} - \text{man} + \text{woman} = \text{queen}$ ”... Kind of
2. Huge splash in NLP world
3. Learns from raw text
4. Pretty simple algorithm
5. Comes pretrained [GloVe](#)
[fastText](#)

There can be a leaner machine learning!

Demo!

```
# king - man + woman = queen??  
vec_king = wv.get_vector("king")  
vec_man = wv.get_vector("man")  
vec_woman = wv.get_vector("woman")  
wv.similar_by_vector(vec_king - vec_man + vec_woman, topn=5)  
[('king', 0.7955734729766846),  
 ('queen', 0.680290043354034),  
 ('princess', 0.59751874240845032),  
 ('regnant', 0.5888890624046326),  
 ('princesses', 0.5710146427154541)]
```