Лекция 7
Обработка текста

Проектирование интеллектуальных систем

Терехов Валерий Игоревич

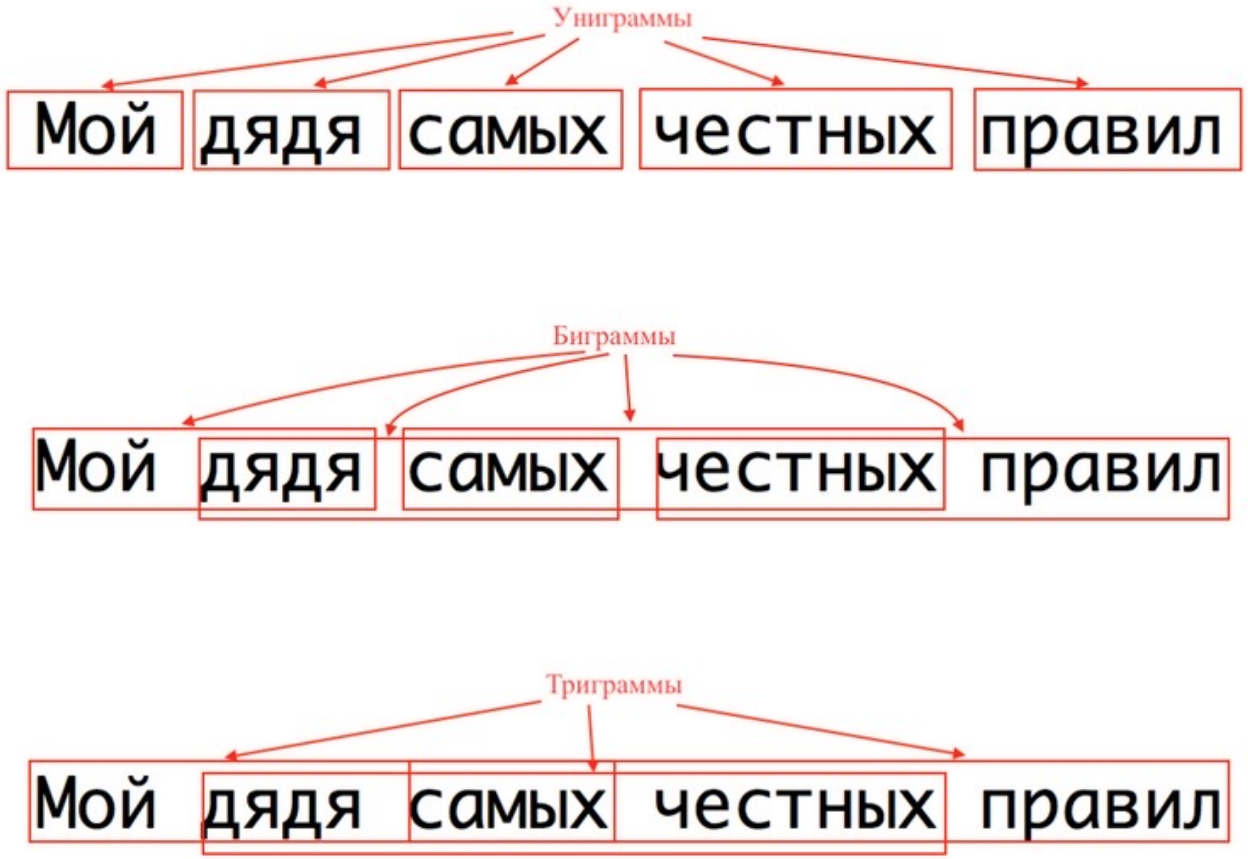
Канев Антон Игоревич

Предобработка текста

Три варианта предобработки текста:

- **Стемминг**. Заключается в отбрасывании окончания
- **N-граммы**. Разделение текста на последовательности по n-слов (word2vec), либо на n-символов (вместо морфологии).
- **Морфологический анализ**. Нахождение начальной формы слова (леммы) и грамматических категорий (число, род, падеж и тд)

N-граммы



- N-граммы слов

Character n-grams

- Building a good stemmer is hard
- Cheap alternative:
  - take every n-character substring of the word
  - related words ➔ many of the same n-grams
  - n=4,5 works well for European languages

document will describe marketing strategies by ...

docu ocum cume umen ment will desc escr scri crib ribe ...

description: desc escr scri crip ript ipti ptio tion
prescribing: pres resc escr scri crib ribi ibin bing
descent: desc esce scen cent
cribbage: crib ribb ibba bbag bage

- N-граммы символов

Стеммер Портера

- Простой стеммер ищет флективную форму в таблице поиска. Недостаток – нужно перечислить все формы в таблице, поэтому незнакомые слова не обработаются
- Алгоритмы усечения окончаний хранят список «правил», по которым отбрасываются окончания, чтобы найти его основу
- Алгоритм стеммера Портера опубликован в 1980 году Мартином Портером. Он по правилам отсекает окончания и суффиксы

Bag-of-words

- Для Bag-of-words составляется словарь из слов текста и указывается, какое количество раз каждое из них употребляется.
- В примере три рецензии. Необходимо подсчитать количество слов в каждой

	1 This	2 movie	3 is	4 very	5 scary	6 and	7 long	8 not	9 slow	10 spooky	11 good	Length of the review(in words)
Review 1	1	1	1	1	1	1	1	0	0	0	0	7
Review 2	1	1	2	0	0	1	1	0	1	0	0	8
Review 3	1	1	1	0	0	0	1	0	0	1	1	6

TF-IDF

Далее вычисляем метрику TF – частота употребления слова в документе

Term	Review 1	Review 2	Review 3	TF (Review 1)	TF (Review 2)	TF (Review 3)
This	1	1	1	1/7	1/8	1/6
movie	1	1	1	1/7	1/8	1/6
is	1	2	1	1/7	1/4	1/6
very	1	0	0	1/7	0	0
scary	1	1	0	1/7	1/8	0
and	1	1	1	1/7	1/8	1/6
long	1	0	0	1/7	0	0
not	0	1	0	0	1/8	0
slow	0	1	0	0	1/8	0
spooky	0	0	1	0	0	1/6
good	0	0	1	0	0	1/6

$$\mathrm{tf}(t,d) = \frac{n_t}{\sum_k n_k}$$

где n_t есть число вхождений слова t в документ, а в знаменателе — общее число слов в данном документе.

TF-IDF

Далее вычисляется метрика IDF, и ее значение умножается на TF

Term	Review 1	Review 2	Review 3	IDF	TF-IDF (Review 1)	TF-IDF (Review 2)	TF-IDF (Review 3)
This	1	1	1	0.00	0.000	0.000	0.000
movie	1	1	1	0.00	0.000	0.000	0.000
is	1	2	1	0.00	0.000	0.000	0.000
very	1	0	0	0.48	0.068	0.000	0.000
scary	1	1	0	0.18	0.025	0.022	0.000
and	1	1	1	0.00	0.000	0.000	0.000
long	1	0	0	0.48	0.068	0.000	0.000
not	0	1	0	0.48	0.000	0.060	0.000
slow	0	1	0	0.48	0.000	0.060	0.000
spooky	0	0	1	0.48	0.000	0.000	0.080
good	0	0	1	0.48	0.000	0.000	0.080

$$\mathrm{idf}(t, D) = \log \frac{|D|}{|\{ d_i \in D \mid t \in d_i \}|}$$

- |D| — число документов в коллекции;
- $|\{ d_i \in D \mid t \in d_i \}|$ — число документов из коллекции D, в которых встречается t (когда $n_t \neq 0$).

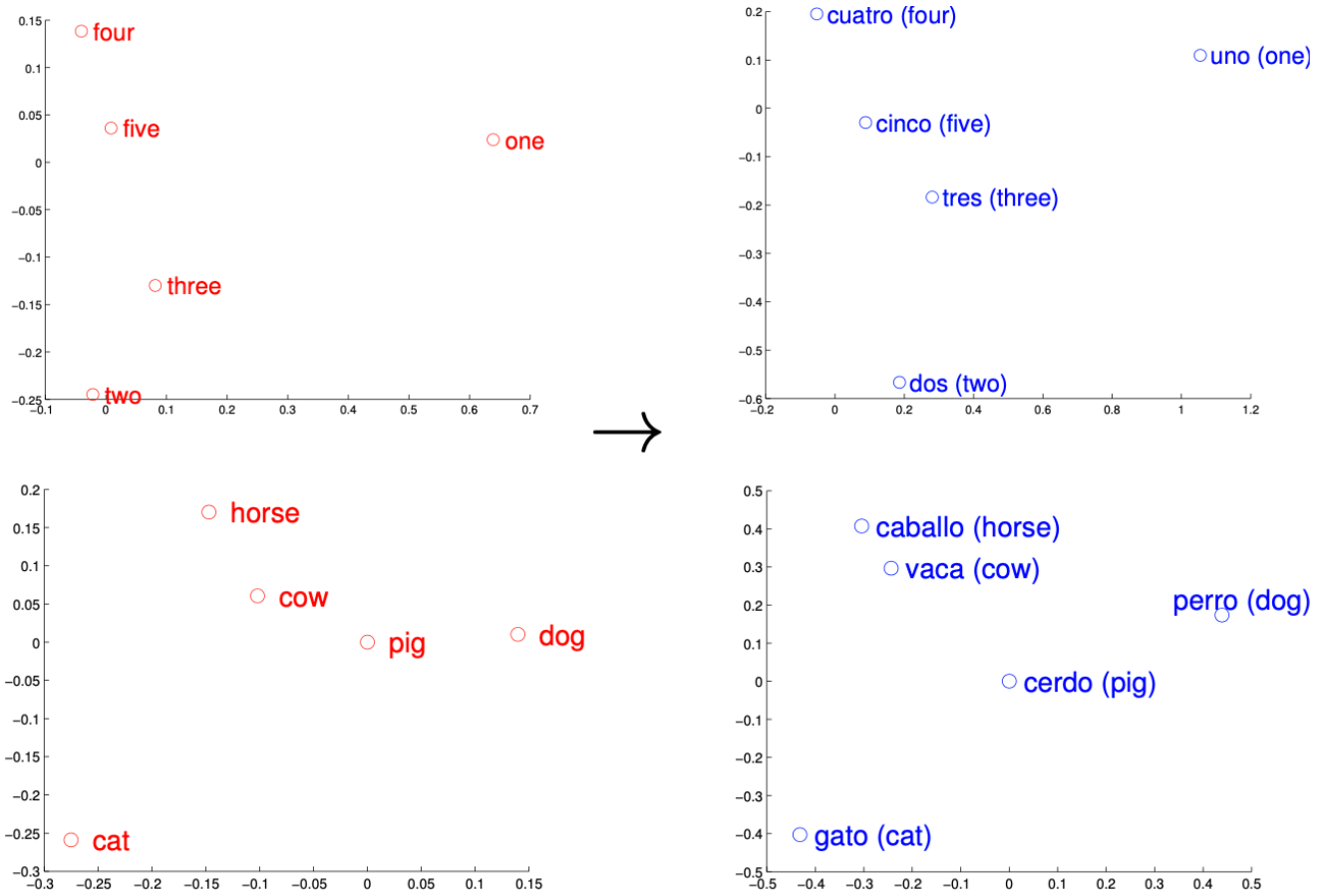
$$\text{tf-idf}(t, d, D) = \mathrm{tf}(t, d) \times \mathrm{idf}(t, D)$$

Word embedding

- Классическое представление bag-of-words плохо подходит для представления данных для обучения нейронных сетей.
- Размерность такого пространства признаков оказывается очень большой, равной количество слов в словаре.
- Поэтому оказывается очень полезным использовать векторное представление слов (embedding). Помимо сокращения пространства признаков это позволяет близкие к друг друг слова располагать ближе в этом пространстве.

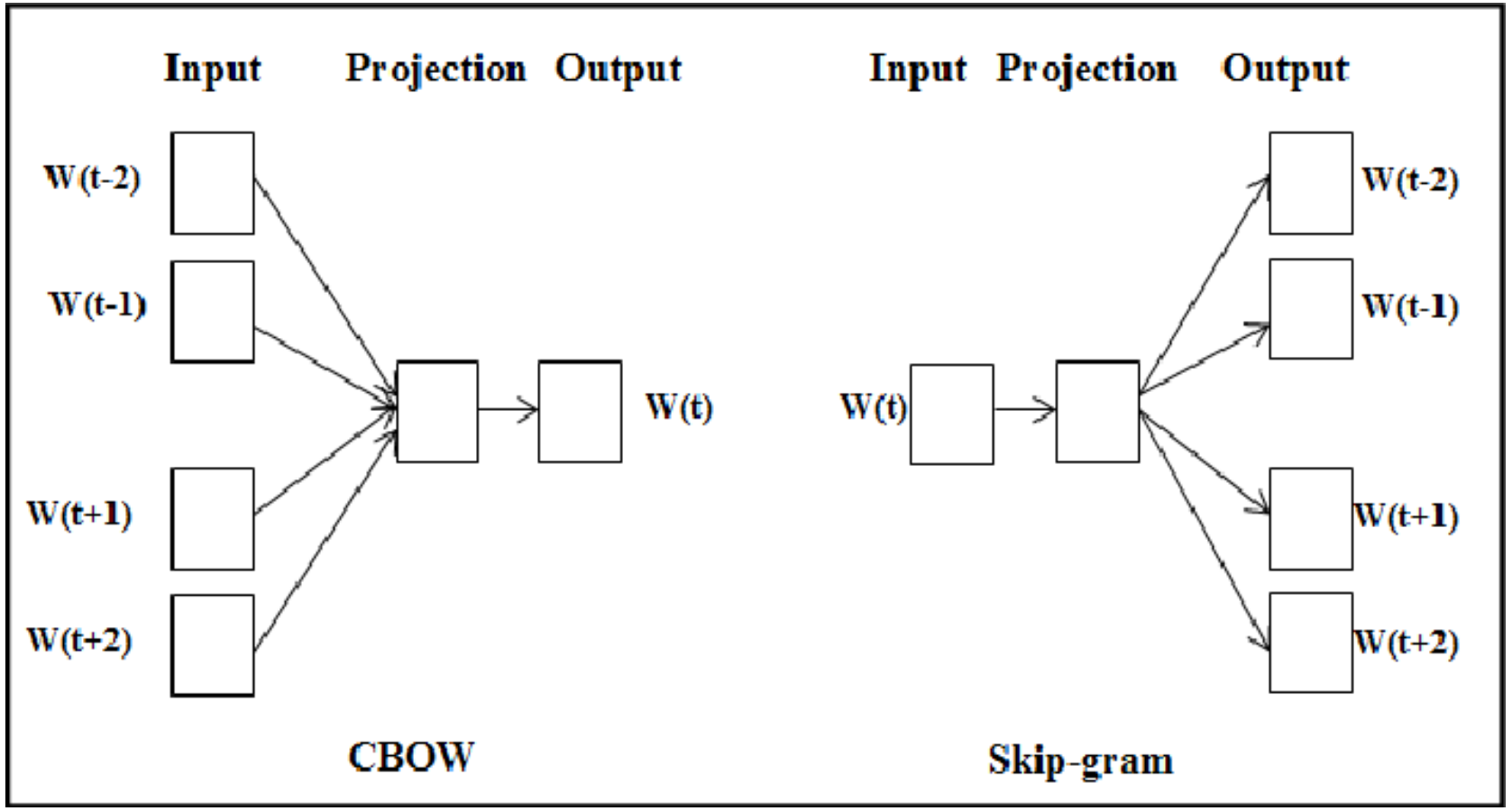
Word2vec

- Разработано в Google в 2013 году. По большому корпусу данных вычисляется векторное представление слов (embedding), обучаясь на этих данных.
- Каждому слову соответствует вектор в этом пространстве. Схожие по смыслу слова находятся в этом пространстве рядом.
- Используется модель из одного скрытого слоя.
- Между представлениями для разных языков также наблюдается зависимость



CBOW и Skip-gram

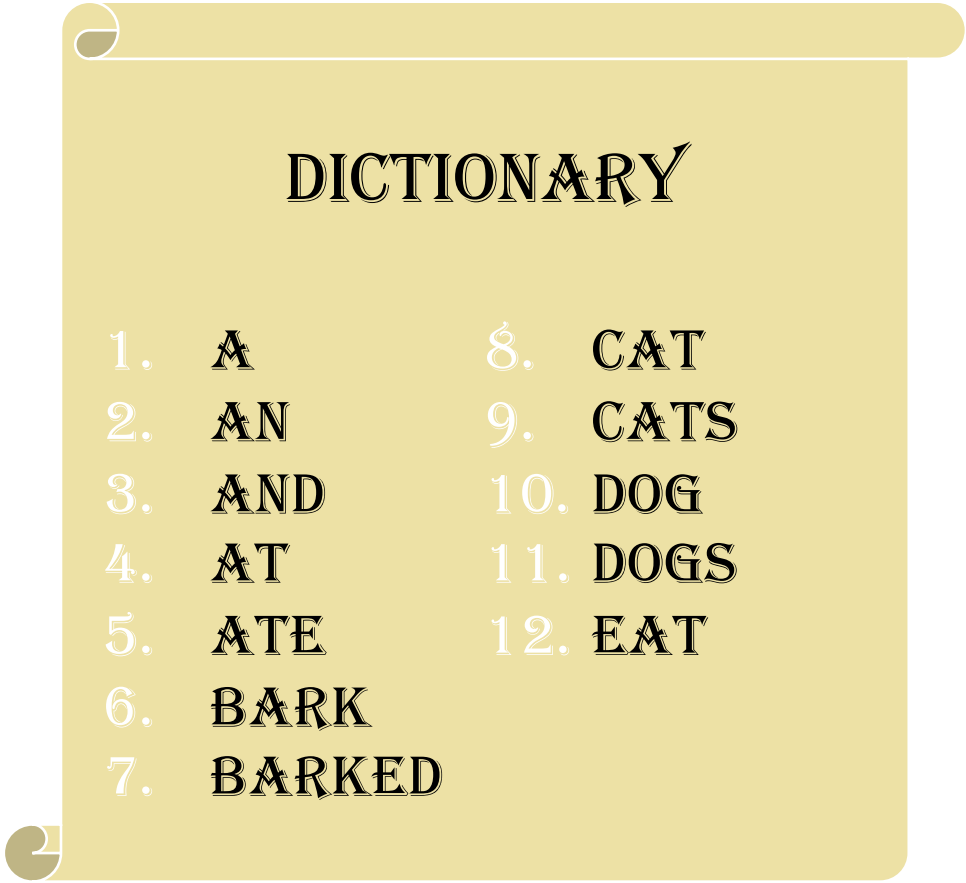
- Используется два вида представления: CBOW (Continuous Bag of Words) и Skip-Gram. Эти два представления оперируют с определенным окном входных данных.
- CBOW предсказывает слово исходя из контекста.
- Skip-Gram предлагает список вероятного контекста в рамках окна для выбранного слова.
- В обоих случаях порядок слов не анализируется.



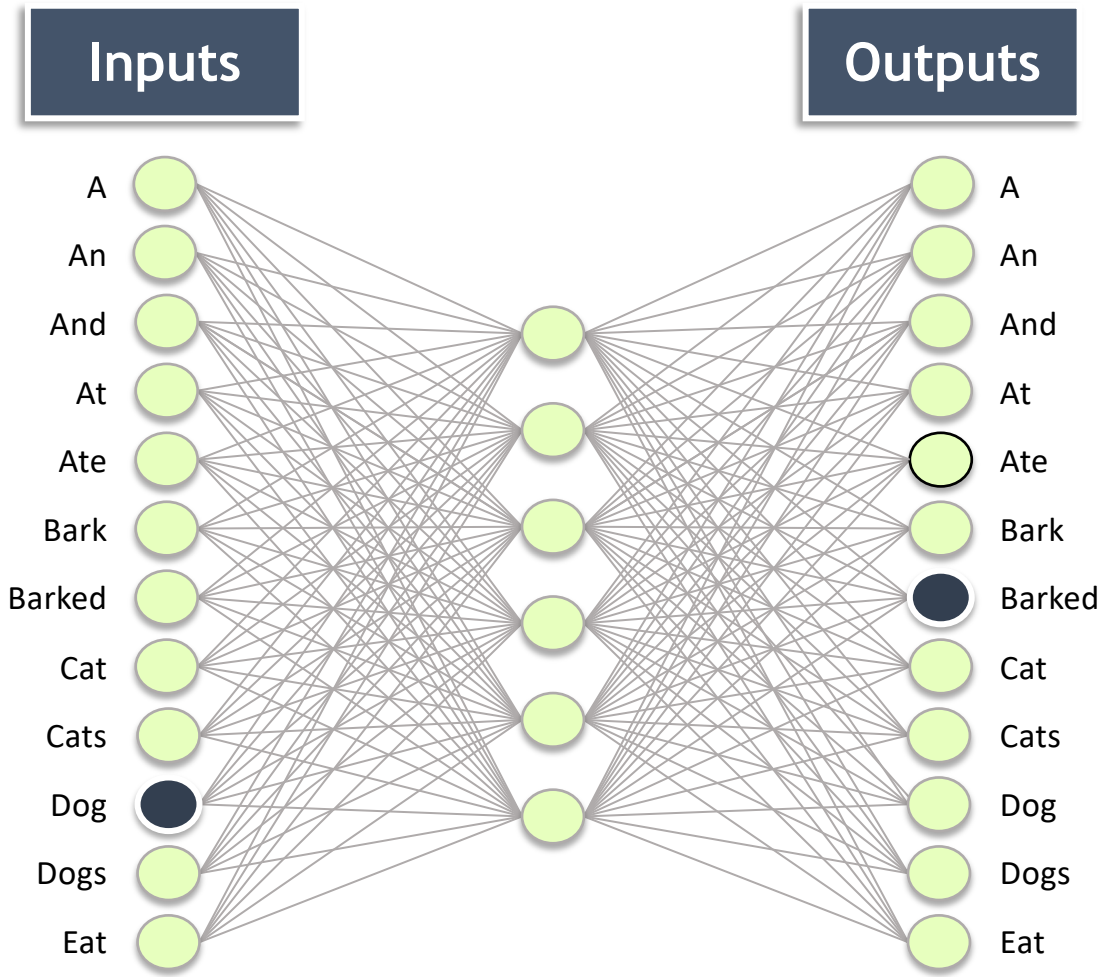
From Words to Numbers

"A dog barked at a cat."

[1, 10, 7, 4, 1, 8]



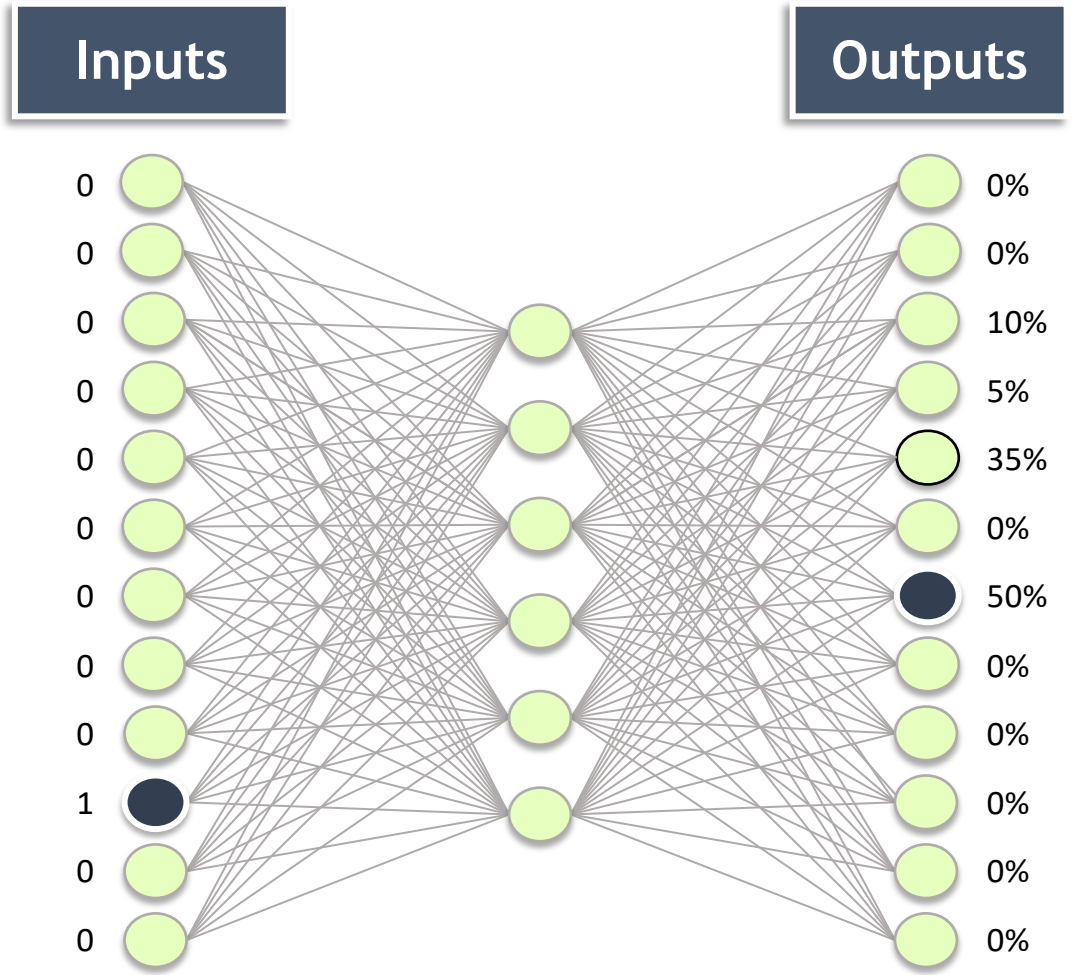
From Words to Numbers



DICTIONARY

1. A
2. AN
3. AND
4. AT
5. ATE
6. BARK
7. BARKED
8. CAT
9. CATS
10. DOG
11. DOGS
12. EAT

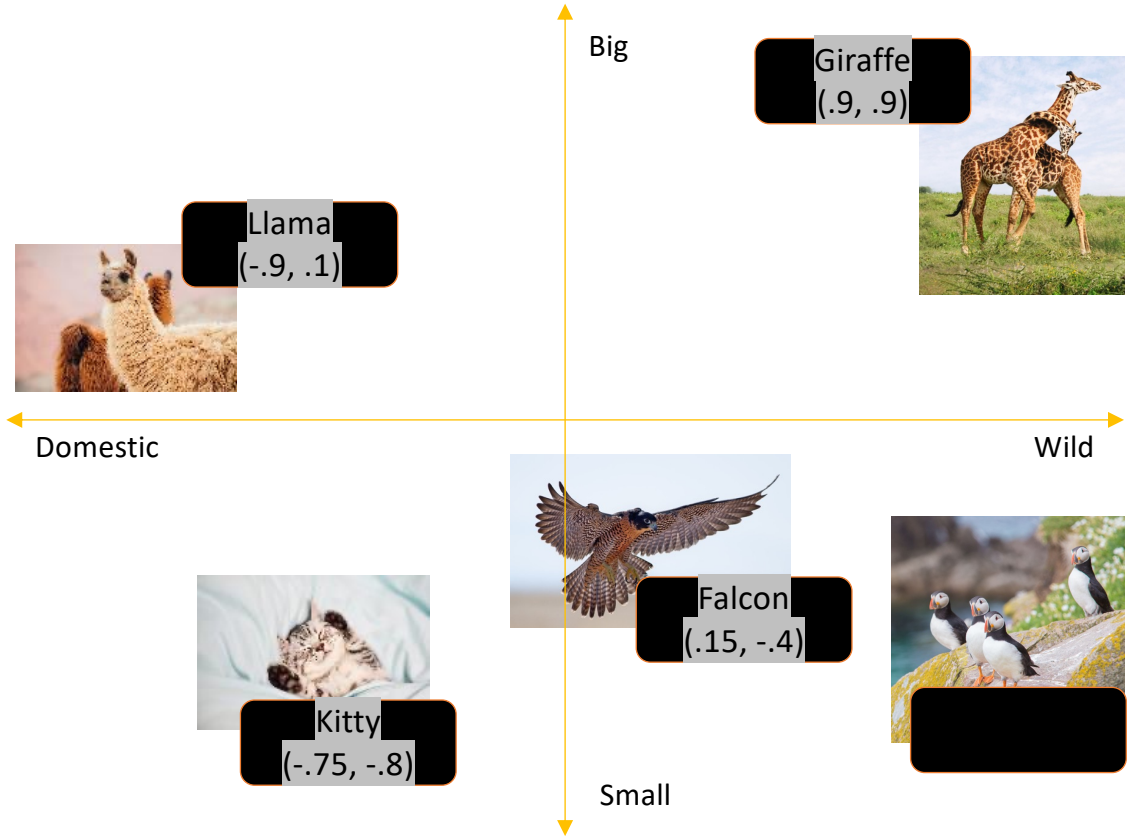
From Words to Numbers



DICTIONARY

1. A
2. AN
3. AND
4. AT
5. ATE
6. BARK
7. BARKED
8. CAT
9. CATS
10. DOG
11. DOGS
12. EAT

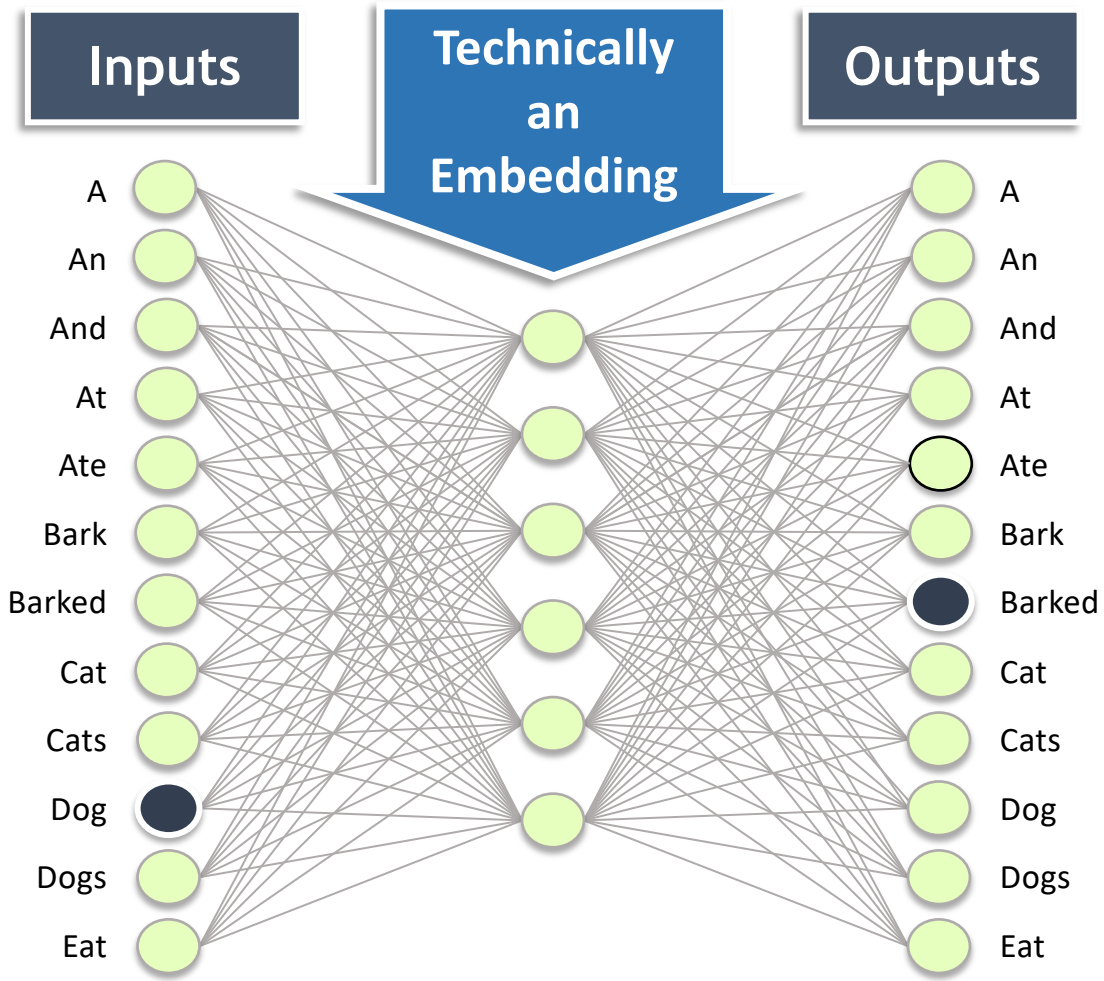
From Words to Numbers



BIGGER DICTIONARY

1.	A	34.	CAT	67.	AN
2.	AN	35.	CATS	68.	AND
3.	AND	36.	DOG	69.	AT
4.	AT	37.	DOGS	70.	ATE
5.	ATE	38.	EAT	71.	BARK
6.	BARK	39.	EATEN	72.	BARKED
7.	BARKED	40.	A	73.	CAT
8.	CAT	41.	AN	74.	CATS
9.	CATS	42.	AND	75.	DOG
10.	DOG	43.	AT	76.	DOGS
11.	DOGS	44.	ATE	77.	EAT
12.	EAT	45.	BARK	78.	EATEN
13.	EATEN	46.	BARKED	79.	...
14.	A	47.	CAT	80.	...
15.	AN	48.	CATS	81.	...
16.	AND	49.	DOG	82.	...
17.	AT	50.	DOGS		
18.	ATE	51.	EAT		
19.	BARK	52.	EATEN		
20.	BARKED	53.	A		
21.	CAT	54.	AN		
22.	CATS	55.	AND		
23.	DOG	56.	AT		
24.	DOGS	57.	ATE		
25.	EAT	58.	BARK		
26.	EATEN	59.	BARKED		
27.	A	60.	CAT		
28.	AN	61.	CATS		
29.	AND	62.	DOG		
30.	AT	63.	DOGS		
31.	ATE	64.	EAT		
32.	BARK	65.	EATEN		
33.	BARKED	66.	A		

From Words to Numbers

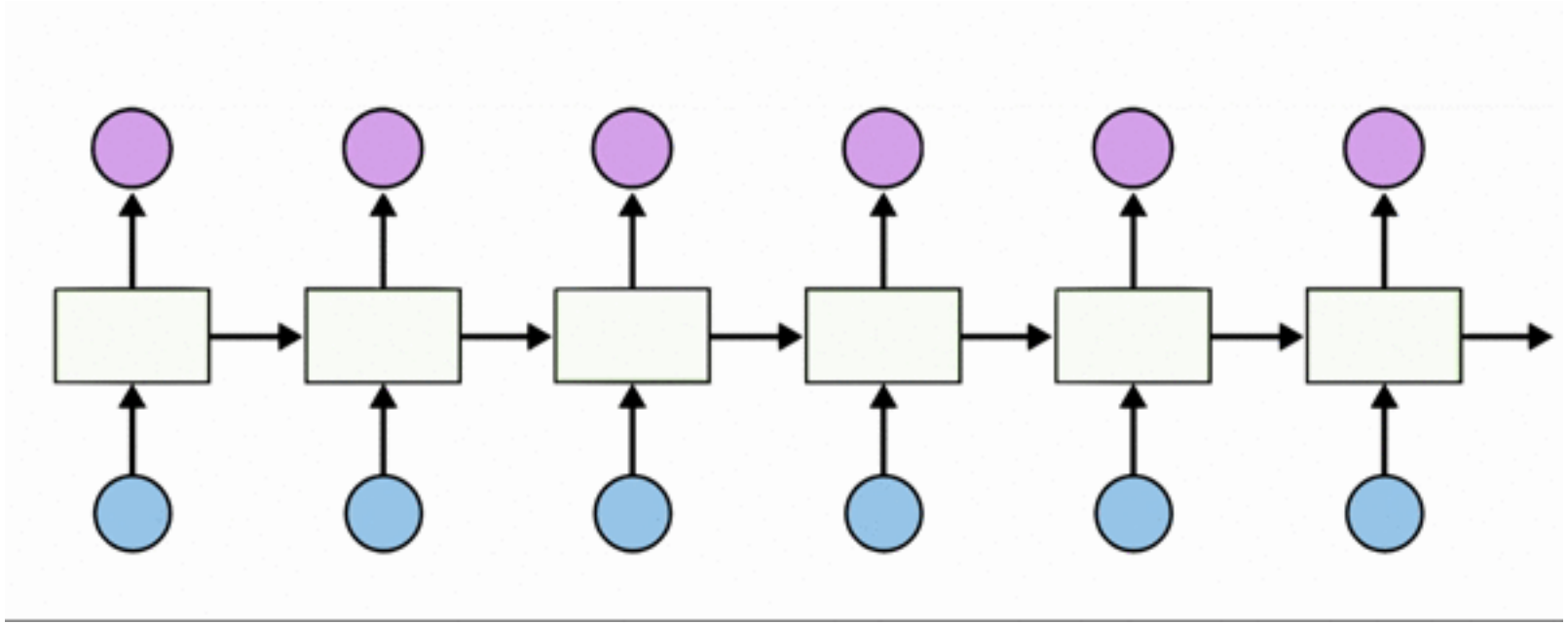


DICTIONARY

1. A
2. AN
3. AND
4. AT
5. ATE
6. BARK
7. BARKED
8. CAT
9. CATS
10. DOG
11. DOGS
12. EAT

RNN

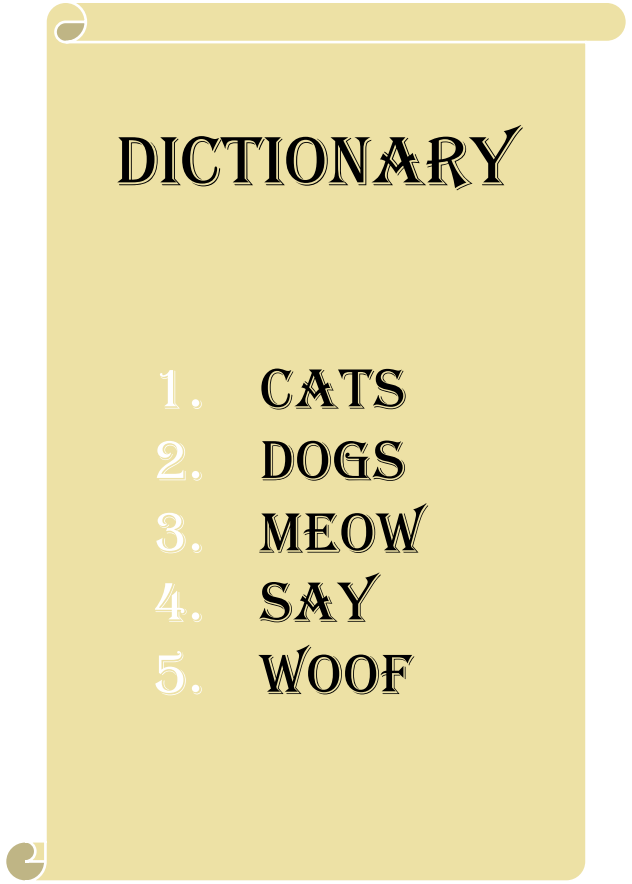
- Рекуррентная нейронная сеть учитывает состояние ячейки



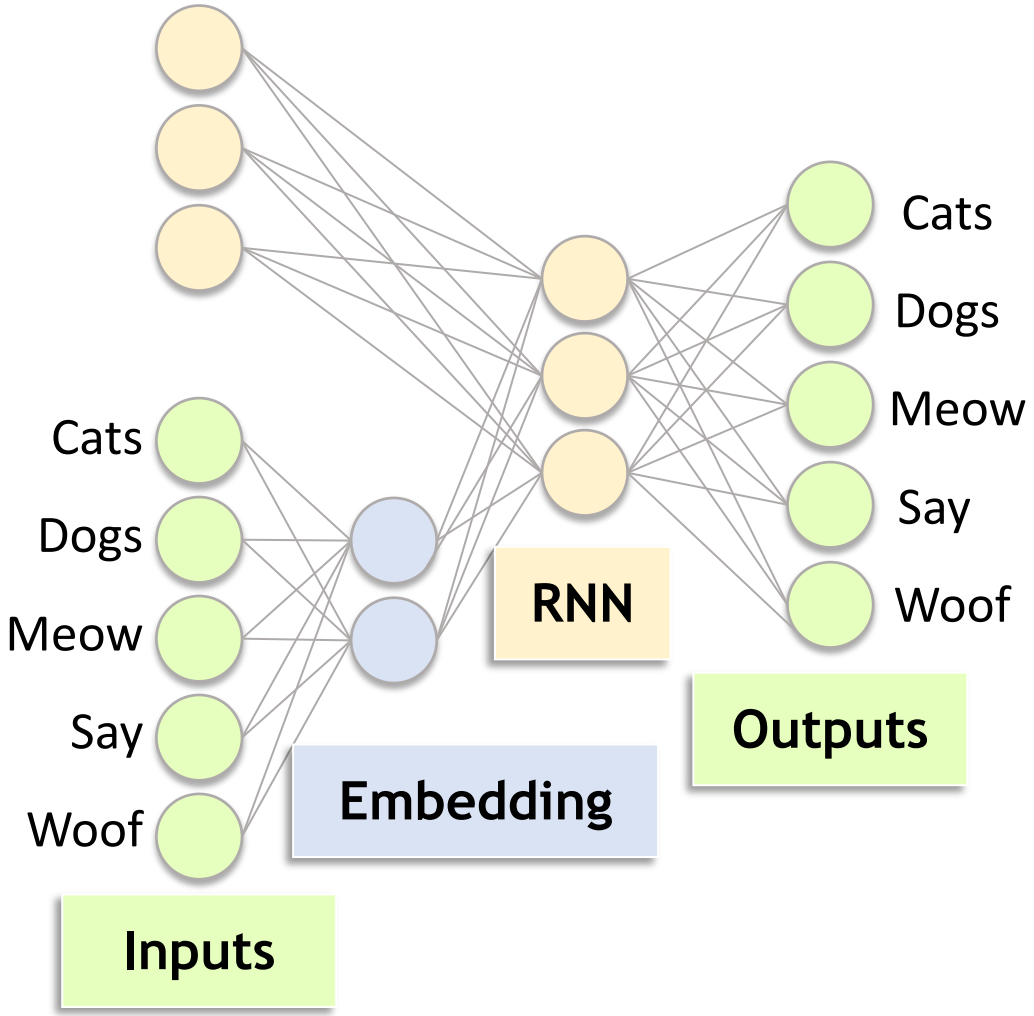
Recurrent Neural Networks

“Cats say ___.”

“Dogs say ___.”

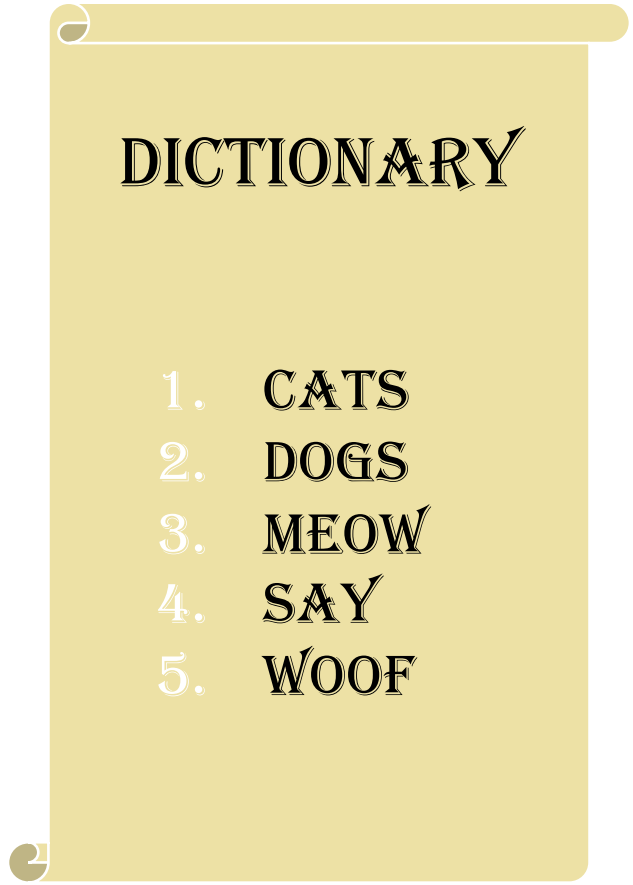


Recurrent Neural Networks

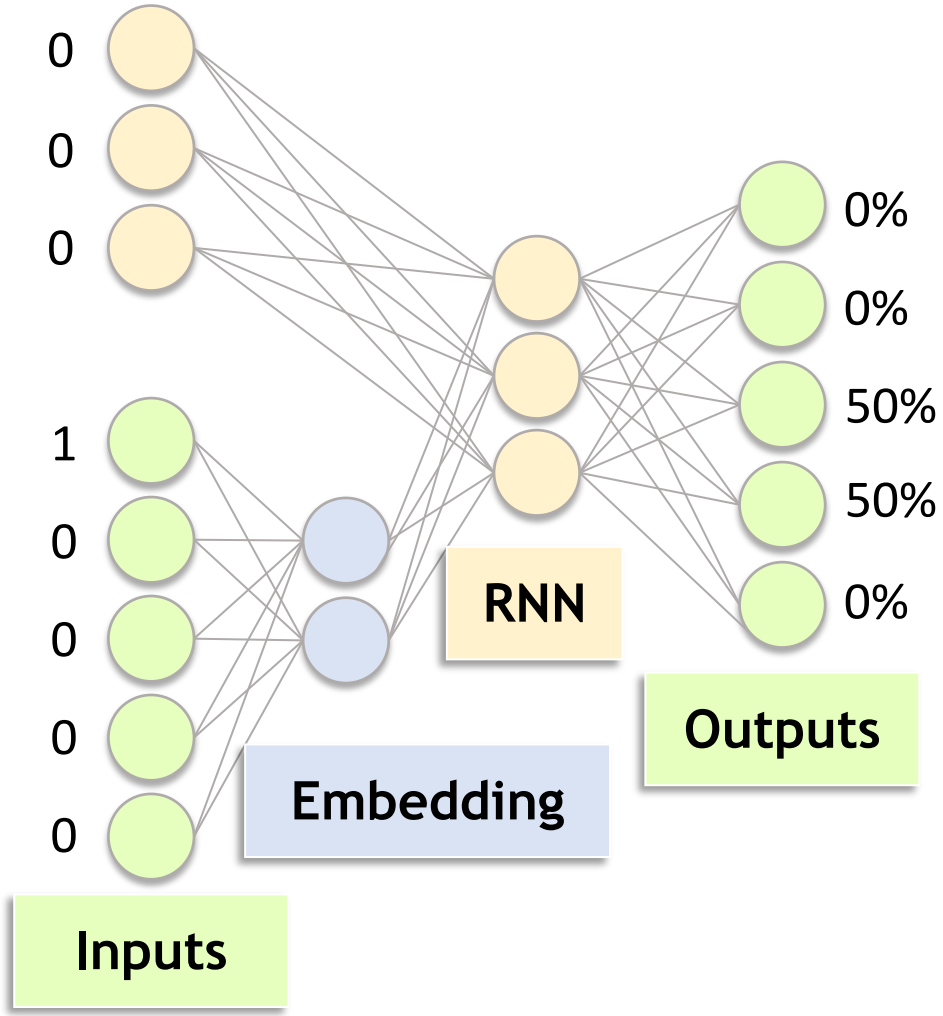


"Cats say ___."

"Dogs say ___."

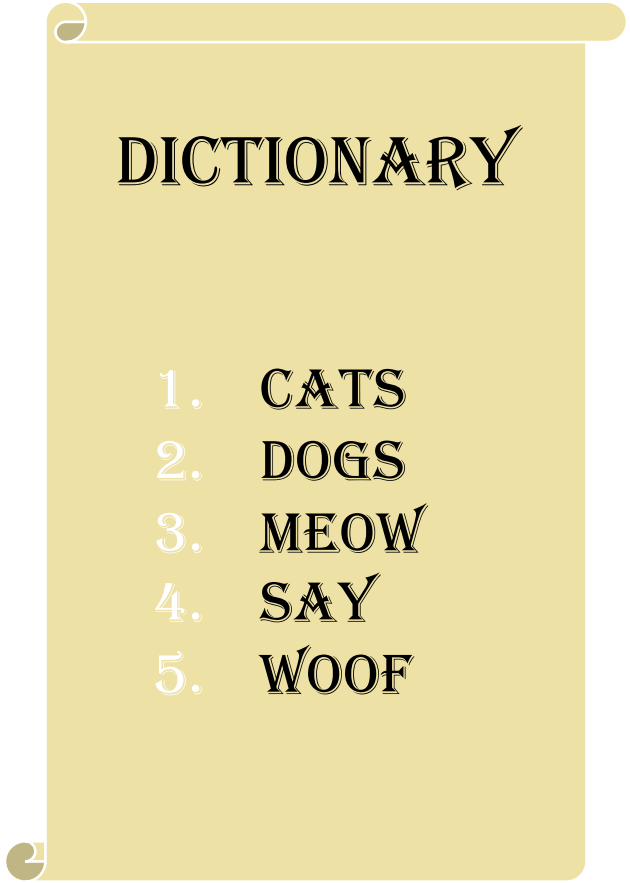


Recurrent Neural Networks

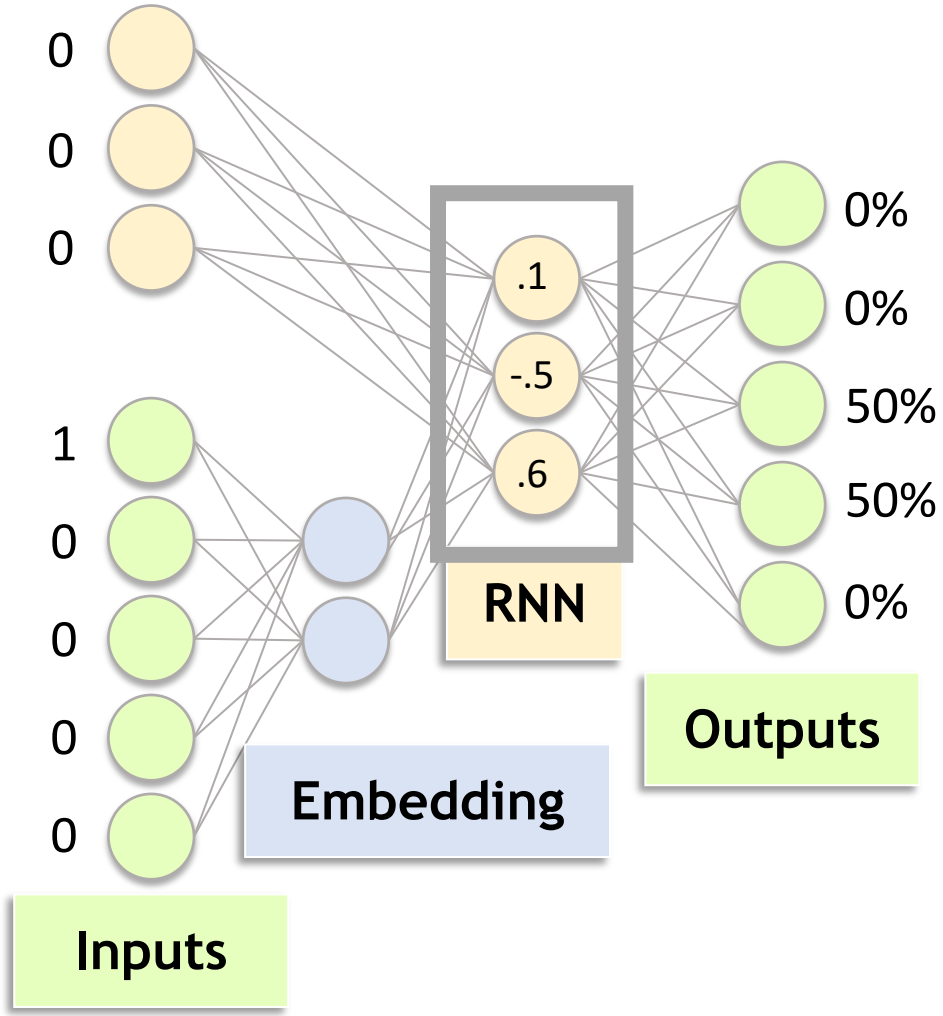


"Cats say ___."

"Dogs say ___."



Recurrent Neural Networks



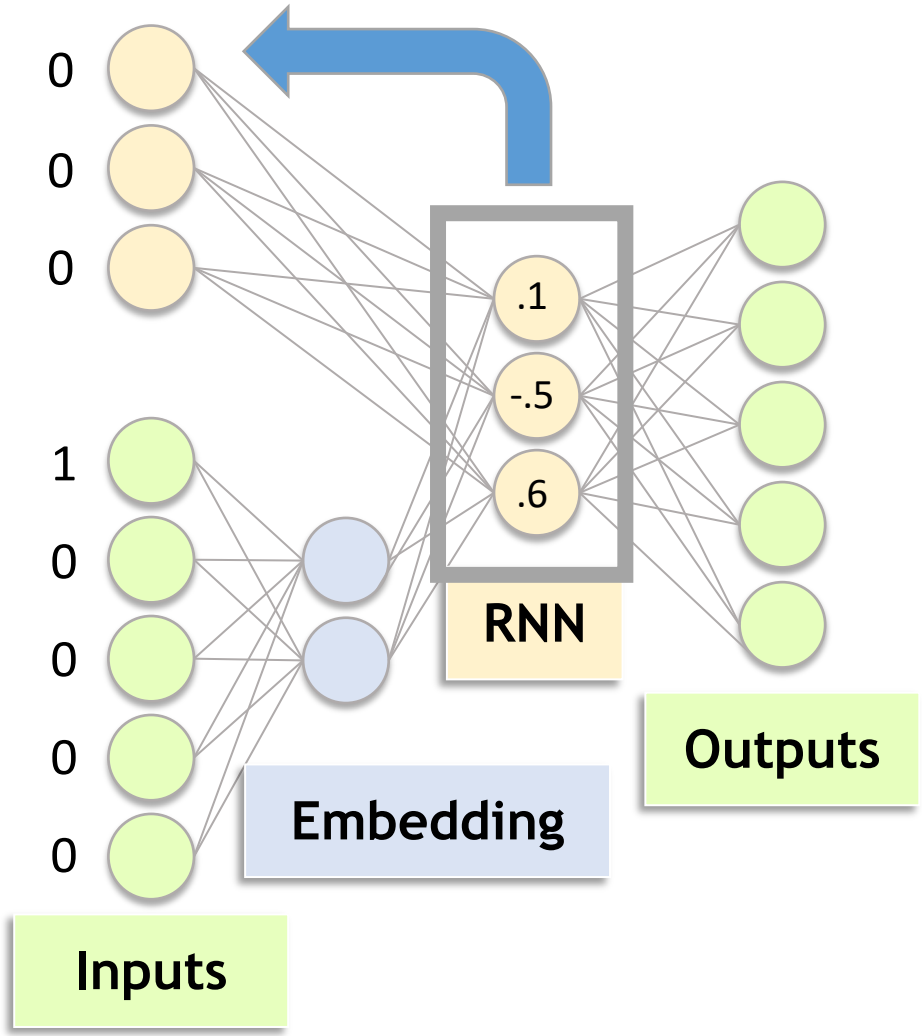
"Cats say ___."

"Dogs say ___."

DICTIONARY

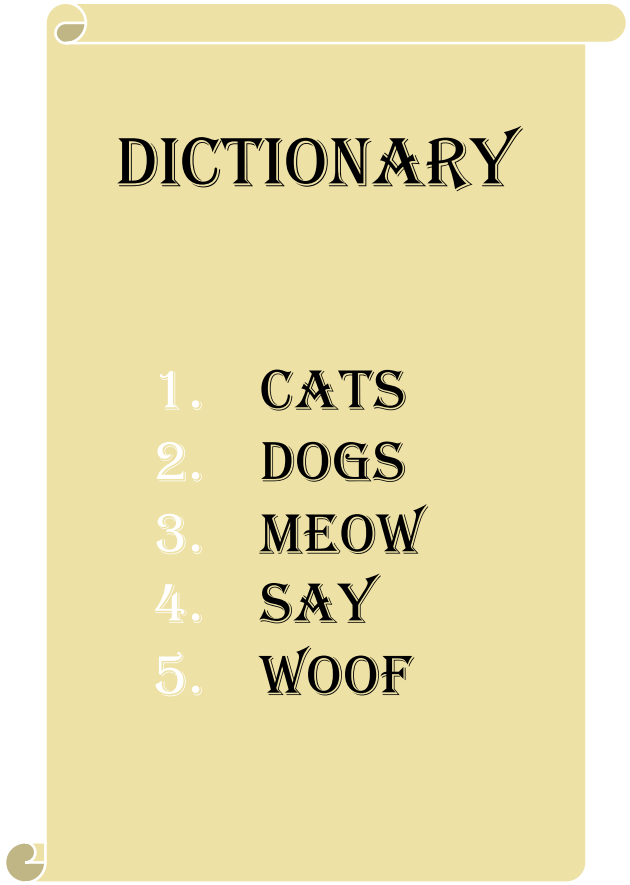
1. CATS
2. DOGS
3. MEOW
4. SAY
5. WOOF

Recurrent Neural Networks

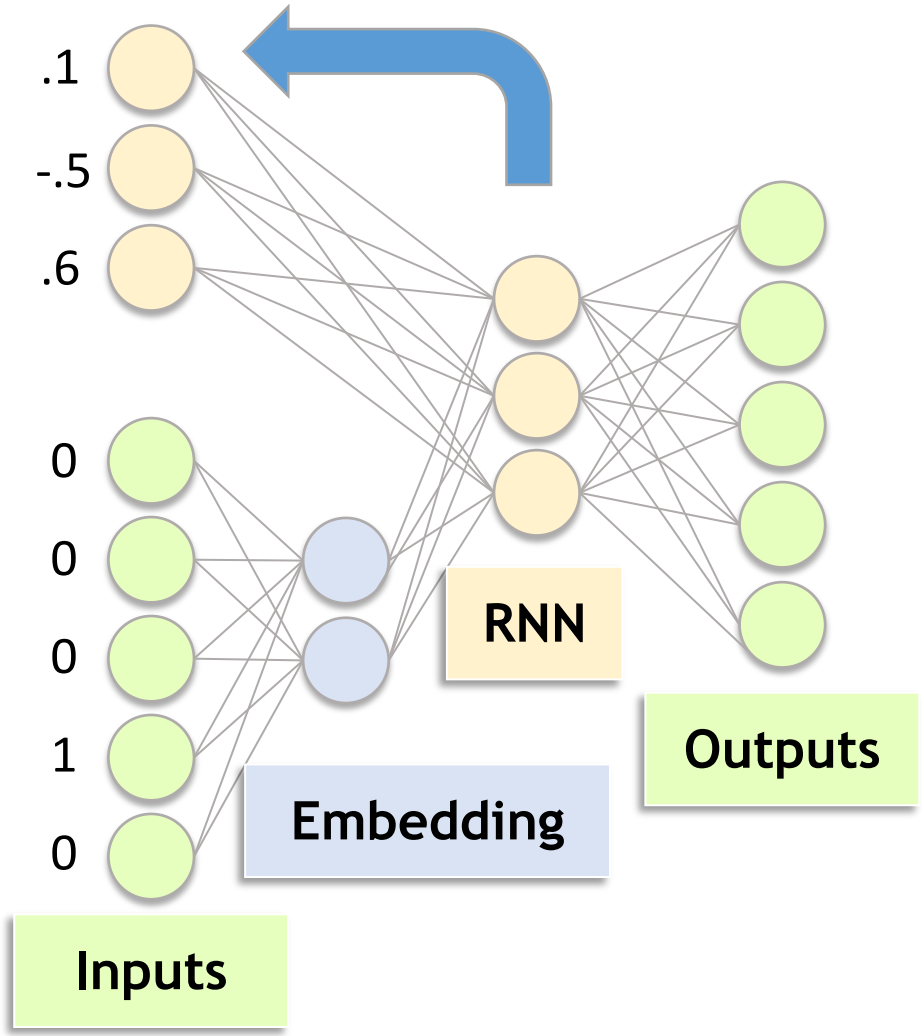


"Cats say ___."

"Dogs say ___."

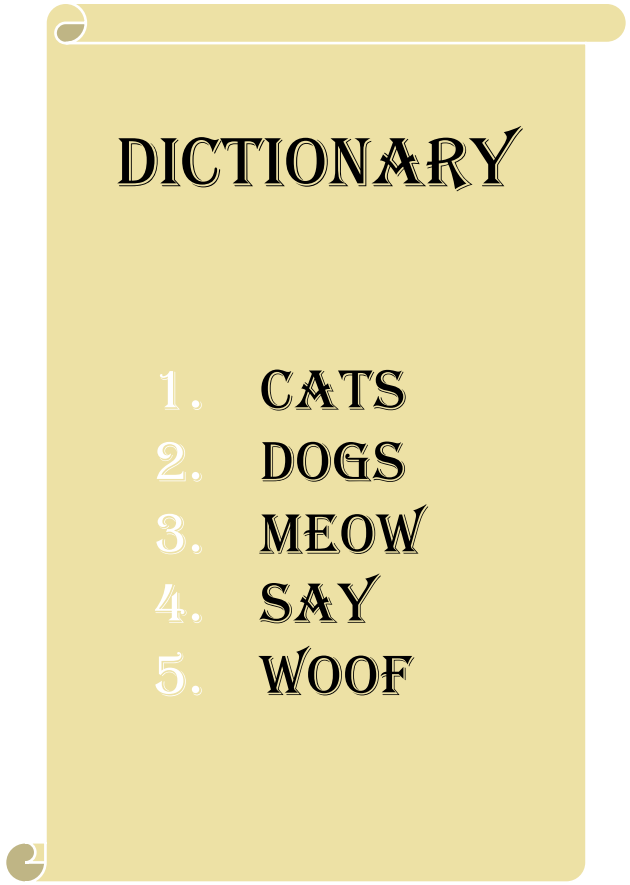


Recurrent Neural Networks

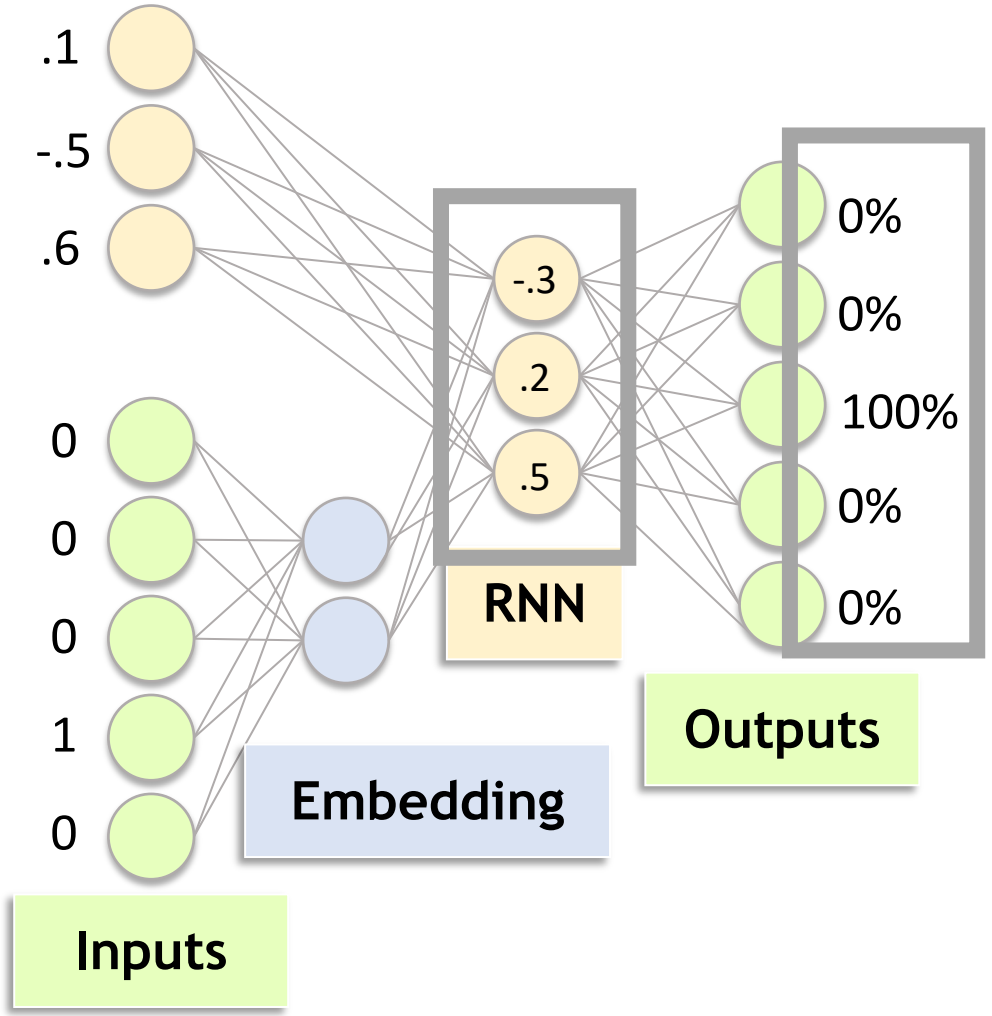


"Cats say ___."

"Dogs say ___."



Recurrent Neural Networks



"Cats say ___."

"Dogs say ___."

DICTIONARY

1. CATS
2. DOGS
3. MEOW
4. SAY
5. WOOF

Seq2seq

Для машинного перевода