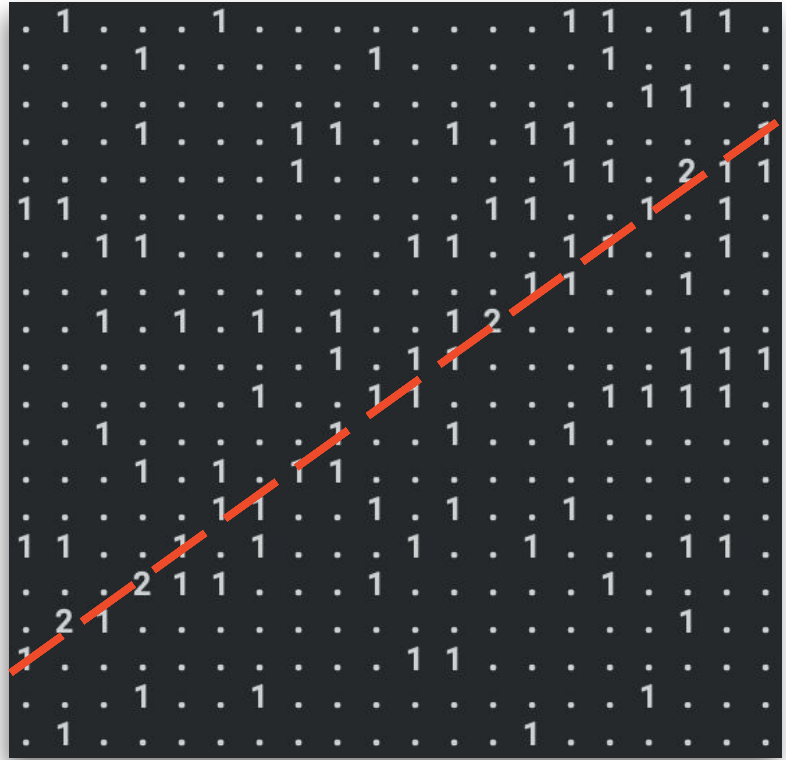
Track Finding Using a Neural Language Model

Jakapat Kannika
Forschungszentrum Jülich

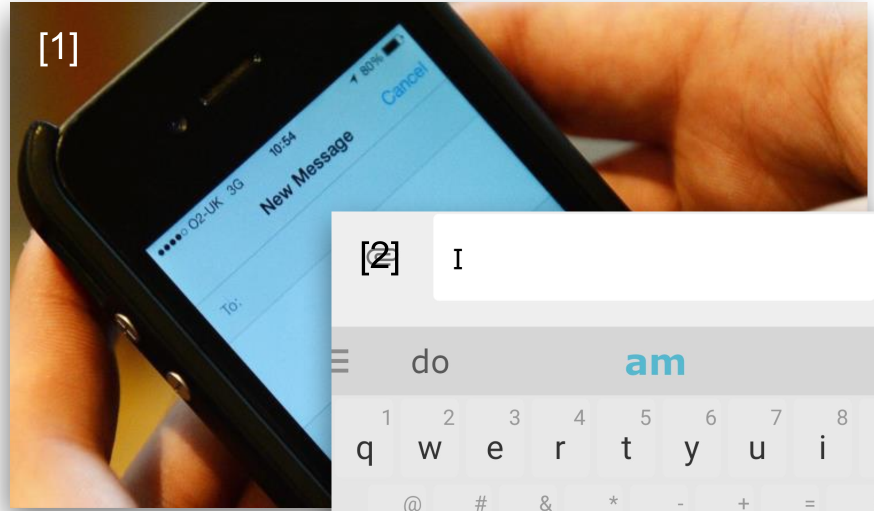


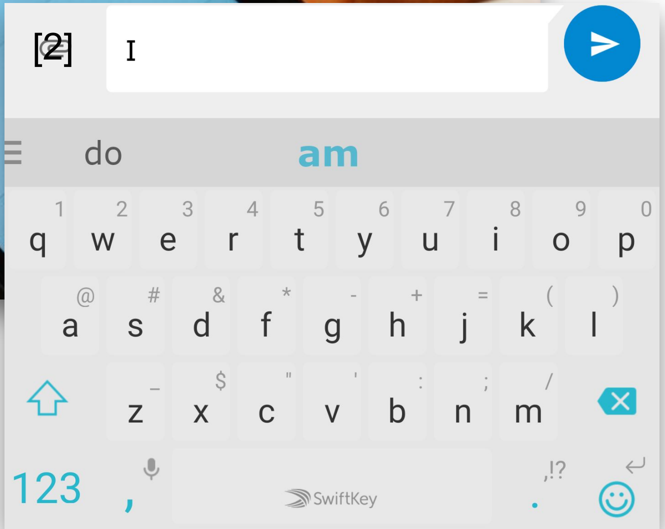
How to find a track from continuous hits in the presence of noise?

Language Models

Example: Next word prediction

“I am Sam”
“Sam I am”
“I do not like green eggs and ham”





Sources:
[1] http://www.androidpolice.com
[2] http://qocall.com

<s>: 3   I: 3   ···

<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

Unigram Model

Word	**Count**
<s>	3
I	3
am	2
</s>	3
Sam	2
do	1

Word	**Count**
not	1
like	1
green	1
eggs	1
and	1
ham	1

Frequency distribution for a unigram model

<s> I: 1   I am: 2   ···

<s> I am Sam </s>

<s> Sam I am </s>

<s> I do not like green eggs and ham </s>

Bigram Model

Word	Count
<s> I	2
I am	2
am Sam	1
Sam </s>	1
<s> Sam	1
Sam I	1
am </s>	1
I do	1

Word	Count
do not	1
not like	1
like green	1
green eggs	1
eggs and	1
and ham	1
Ham </s>	1

Frequency distribution for a bigram model.

Finding a probability distribution of the bigram model

$$P(w_n | w_{n-1}) = C(w_{n-1}w_n) / C(w_{n-1})$$

Probability of which w_{n-1} could be followed by w_n

Bigram count for the word $w_{n-1}w_n$

Unigram count for the word w_{n-1}

Word	Prob.
P(I | <s>)	2/3 = 0.67
P(am | I)	2/3 = 0.67
P(Sam | am)	1/2 = 0.5
P(</s> | Sam)	1/2 = 0.5
P(Sam | <s>)	1/3 = 0.33
P(I | Sam)	1/2 = 0.5
P(</s> | am)	1/2 = 0.5
P(do| I)	1/3 = 0.33

Word	Prob.
P(not | do)	1/1 = 1
P(like | not)	1/1 = 1
P(green | like)	1/1 = 1
P(eggs | green)	1/1 = 1
P(and | eggs)	1/1 = 1
P(ham | and)	1/1 = 1
P(</s> | Ham)	1/1 = 1

Probability distribution for a bigram model.

What is the next word after ‘I’?

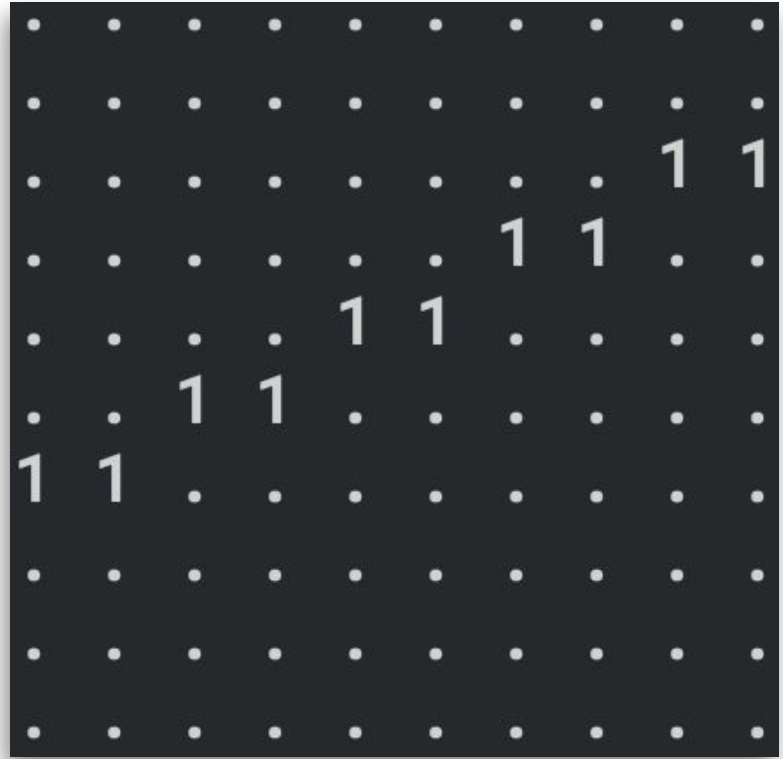
<s> I am Sam </s>
<s> Sam I am </s>
<s> I do not like green eggs and ham </s>

Word	**Prob.**
P(am | I)	0.67
P(do | I)	0.33

How can we apply the language models to the track finding task?

(*x*, *y*) information:

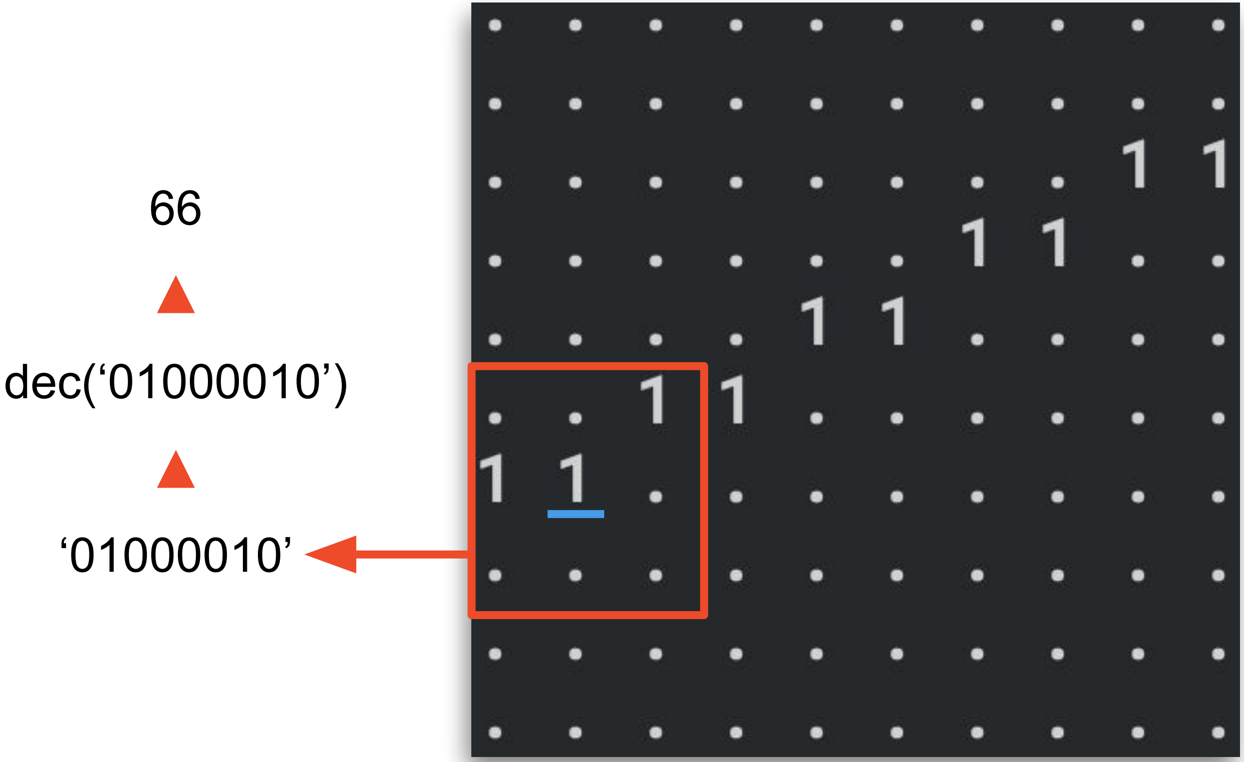
[0, 3], [1, 3], [2, 4], [3, 4], [4, 5], [5, 5], [6, 6], [7, 6], [8, 7], [9, 7]



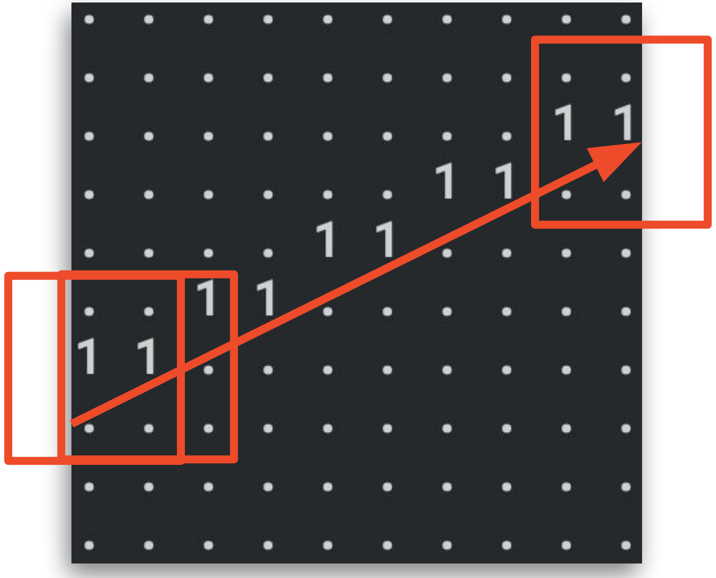
Features:

- neighbor pattern,
- moving direction.

Neighbor Pattern Feature



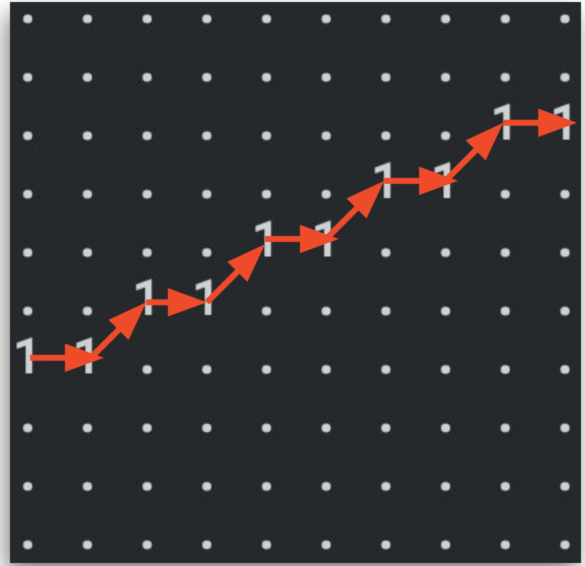
- 66 is a *neighbor pattern id*.
- There are 255 patterns (excluding one that has zero neighbor point).



Neighbor pattern tokens:

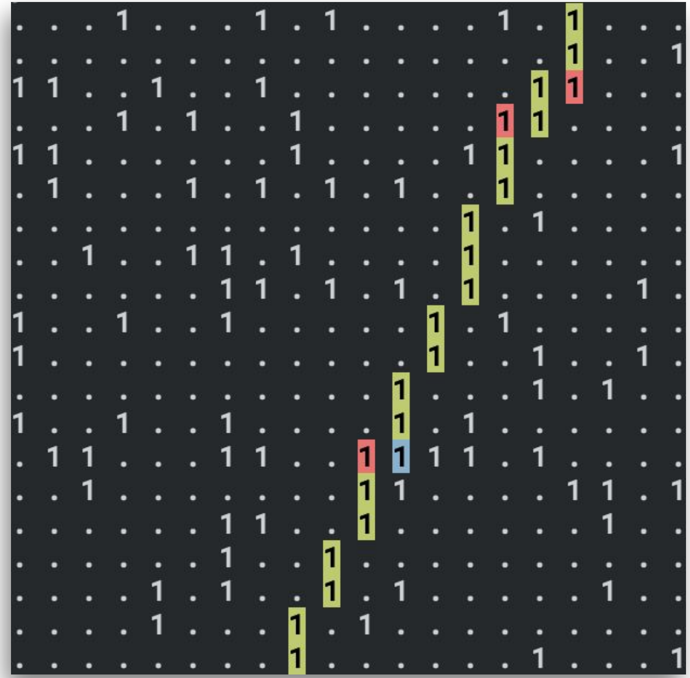
32 66 36 66 36 66 36 66 36 2

Moving Direction Feature

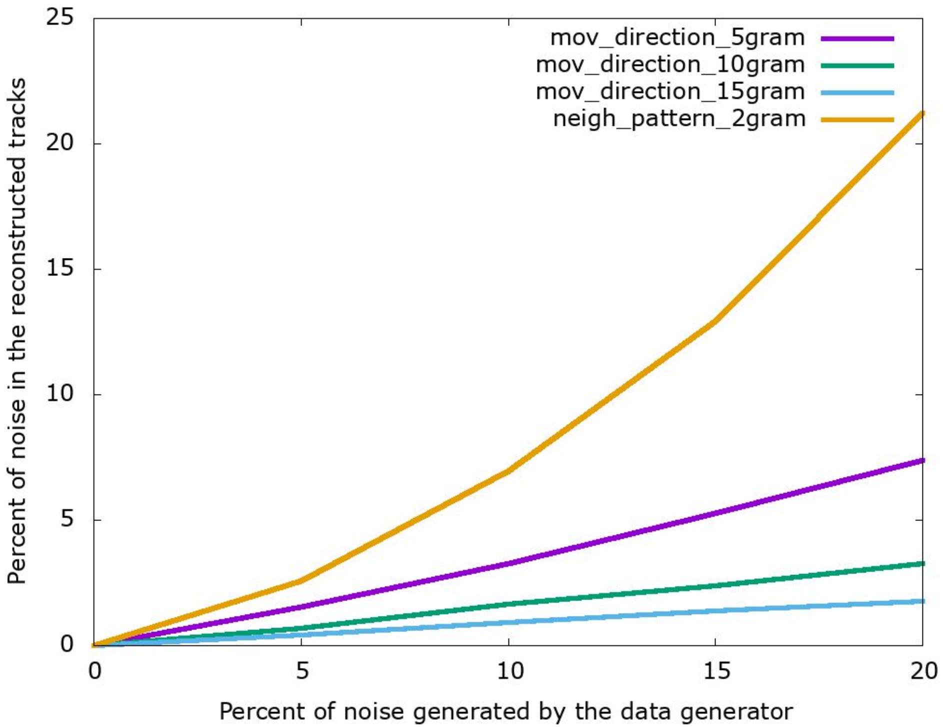


Moving direction tokens:

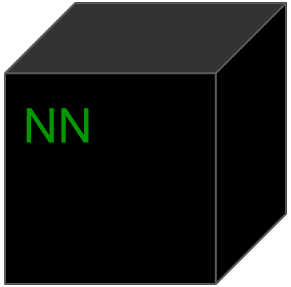
90 45 90 45 90 45 90 45 90



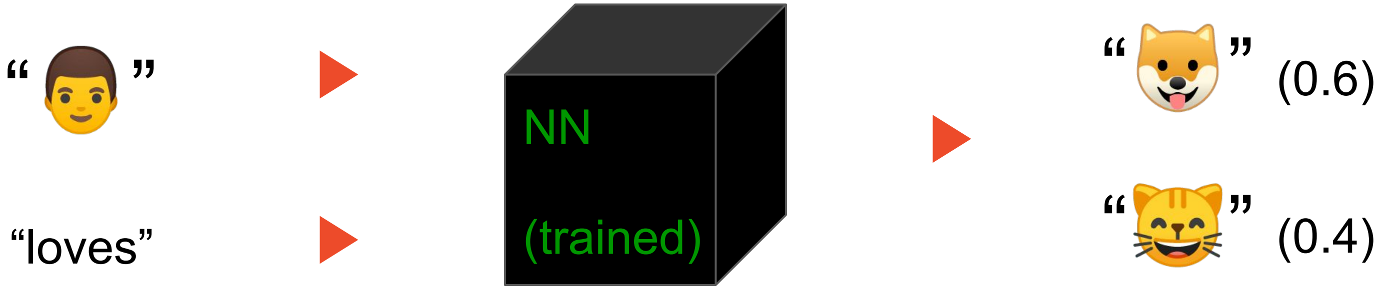
GREEN: correct predicted hit,
RED: incorrect predicted hit,
BLUE: missed correct hit.



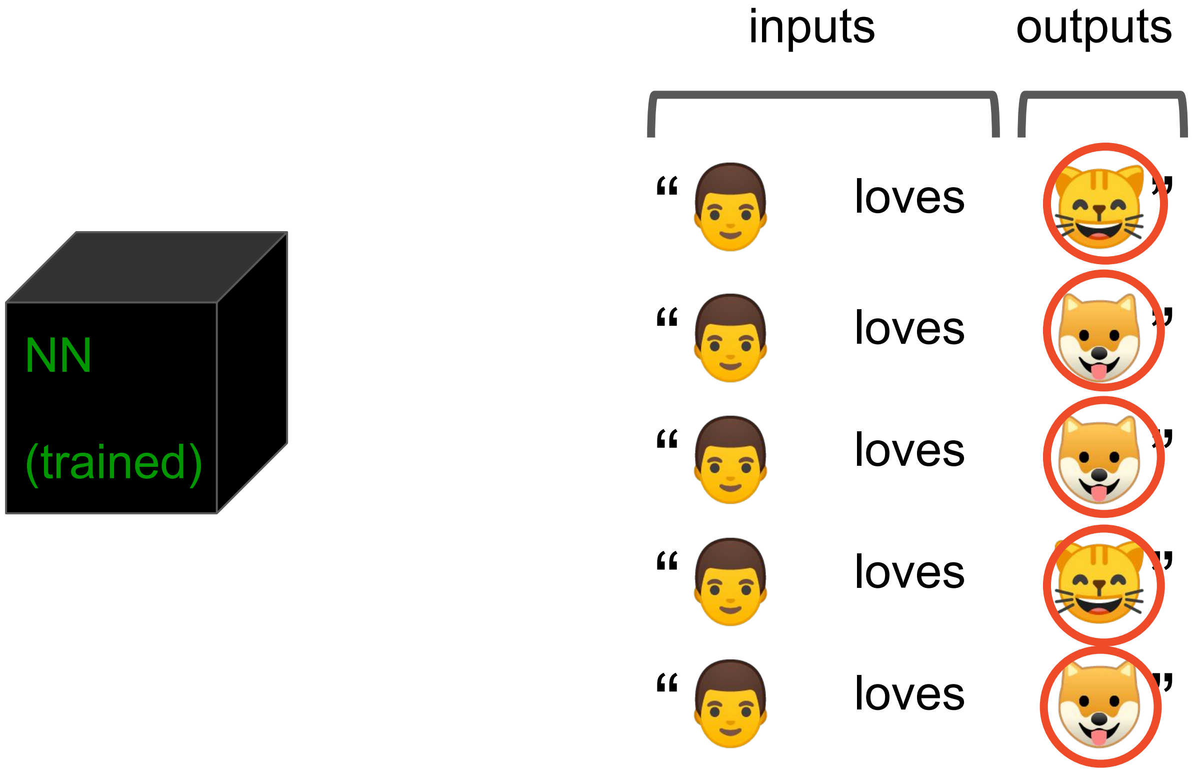
Neural Language Model



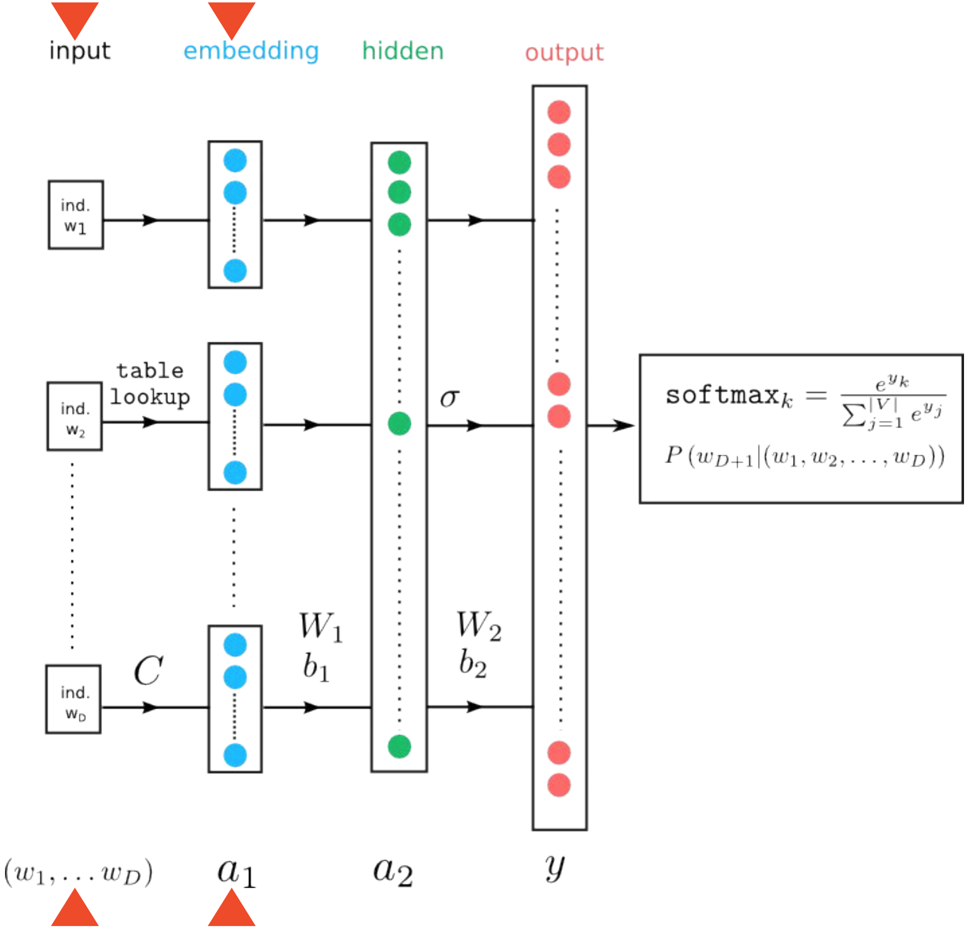
Neural language model: training



Neural language model: testing



Neural language model: training



src: https://burakhimmetoglu.com/2016/12/16/deciphering-the-neural-language-model/

The network architecture for the neural language model vocabs = ["👨", "🐶", "😸", "loves", "hates"]

"👨" = [1, 0, 0, 0, 0]

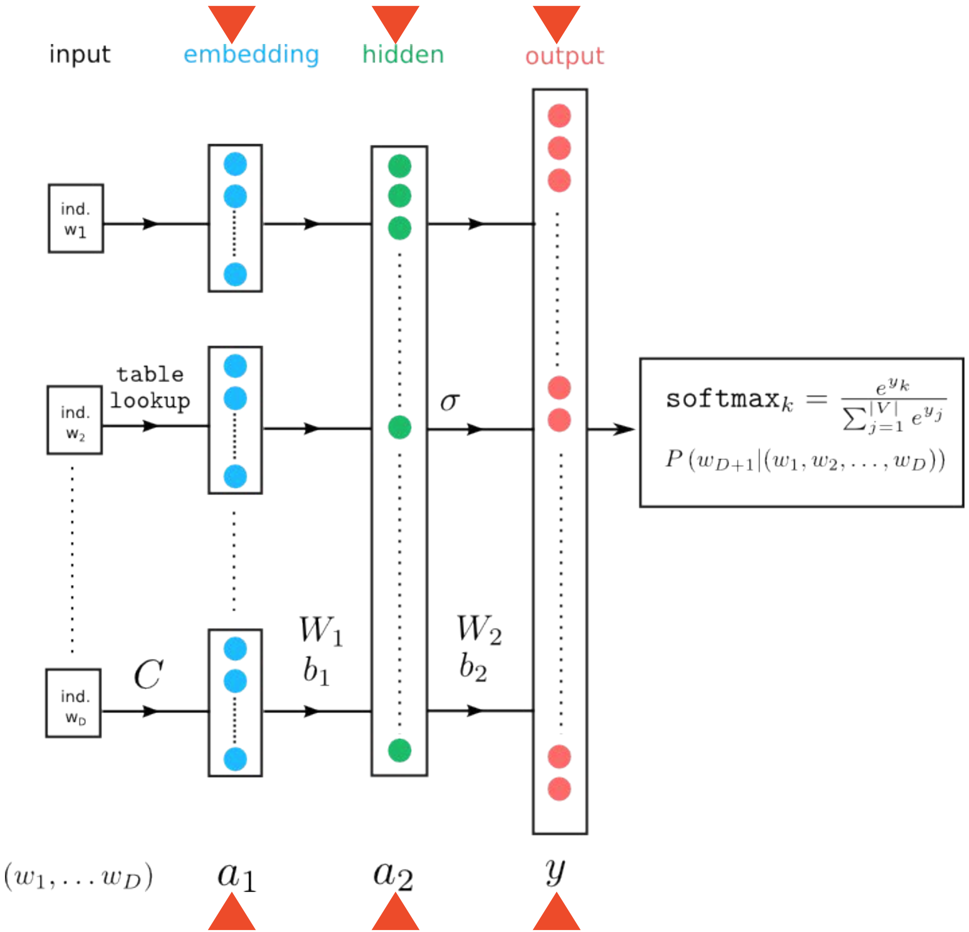
"🐶" = [0, 1, 0, 0, 0]

"😸" = [0, 0, 1, 0, 0]

"loves" = [0, 0, 0, 1, 0]

"hates" = [0, 0, 0, 0, 1]

Numeric representations using the one-hot encoding



src: https://burakhimmetoglu.com/2016/12/16/deciphering-the-neural-language-model/

The network architecture for the neural language model

"👨": 0.15

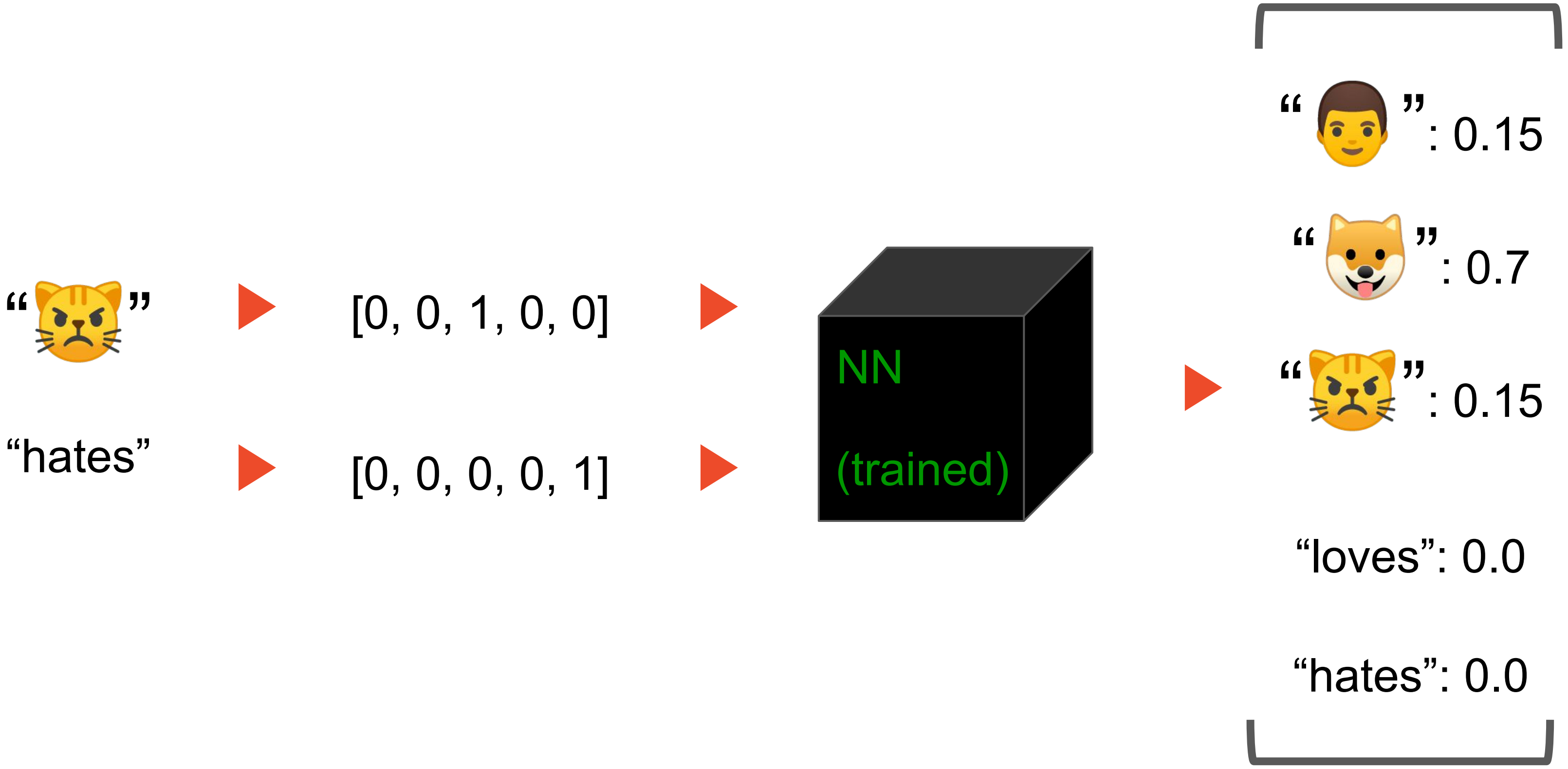
"🐶": 0.70

"😸": 0.15

"loves": 0.00

"hates": 0.00

Output probabilities of the neural language model



Example of the neural language model

What is the difference between the conventional language model and the neural language model?

Curse of Dimensionality

vocabs = [a, …, Z, A, …, Z]

len(vocabs) ▶ 52

2-gram model

max_len(prob_dist) = len(vocabs)^2

2,704

3-gram model

max_len(prob_dist) = len(vocabs)^3

140,608

4-gram model

max_len(prob_dist) = len(vocabs)^4

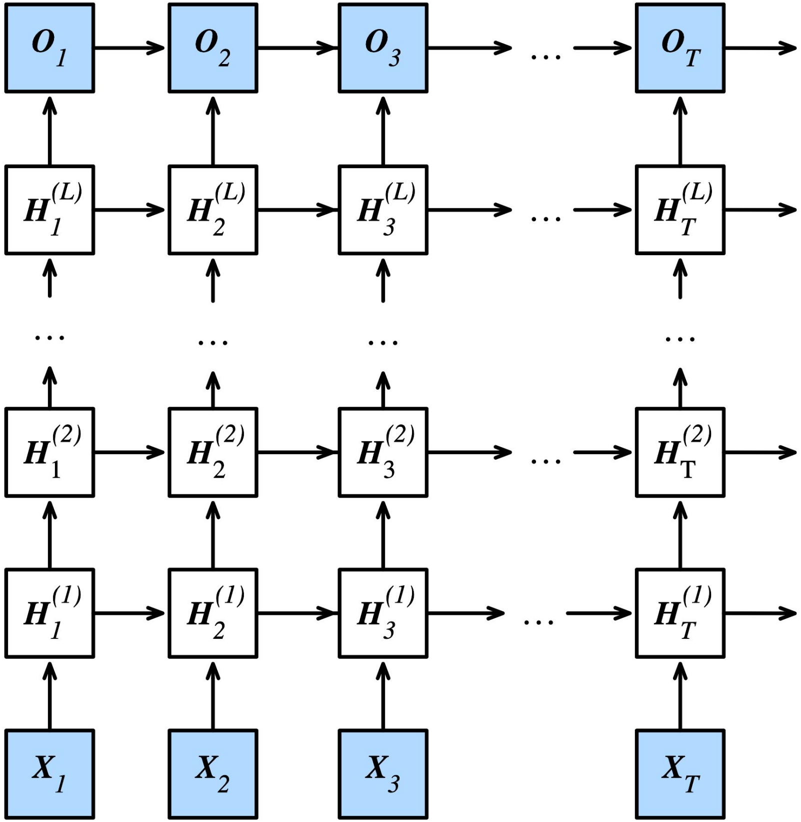
7,311,616

10-gram model

max_len(prob_dist) = len(vocabs)^10

144,555,105,949,057,024

Curse of dimensionality in the conventional language model



"...A neural network language model is a language model based on Neural Networks, exploiting their ability to learn distributed representations to reduce the impact of the curse of dimensionality..."

— Yoshua Bengio (2008), Scholarpedia, 3(1):3881.

src: https://d2l.ai/chapter_recurrent-neural-networks/deep-rnn.html

Network architecture for the neural language model

Multiple N-gram Trainings

Conventional language model

2-gram

seq	prob
...	...
...	...

3-gram

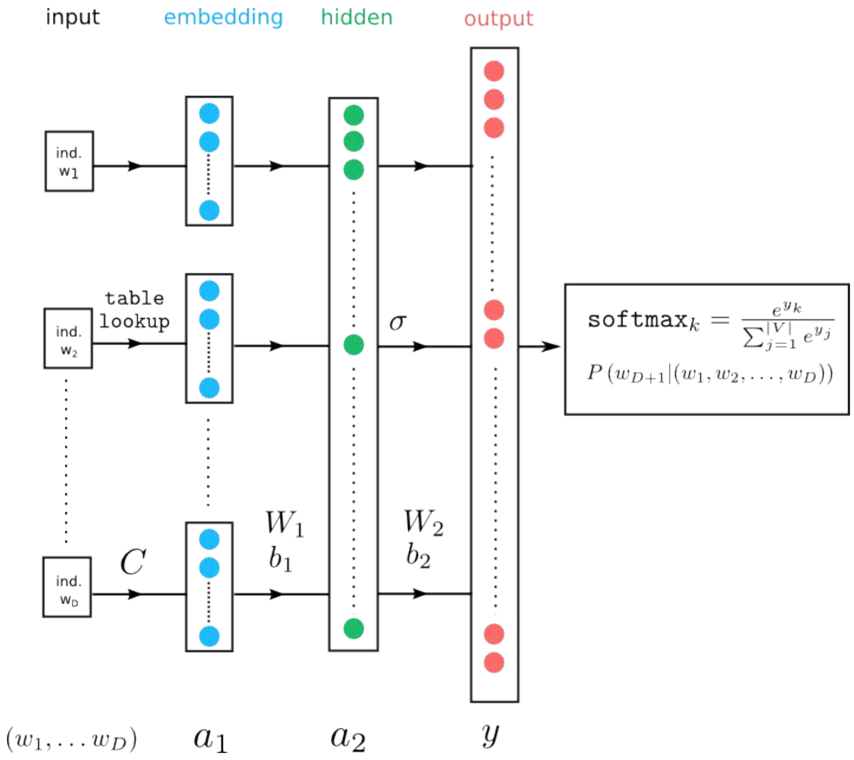
seq	prob
...	...
...	...
...	...

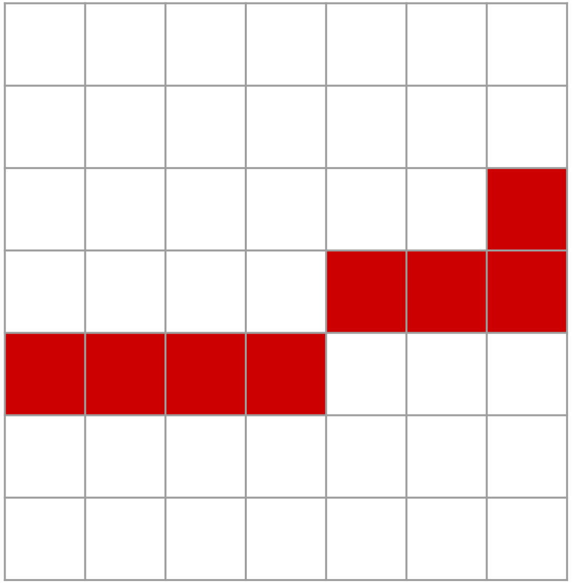
4-gram

seq	prob
...	...
...	...
...	...
...	...

...

Neural language model





Moving direction:
["0" "0" "0" "45" "0" "0" "90"]

Training the neural network with variable length sequences

Moving direction:
[“0” “0” “0” “45” “0” “0” “90”]

2gram

inputs	outputs
“0”	“0”
“0”	“0”
“0”	“45”
“45”	“0”
“0”	“0”
“0”	“90”

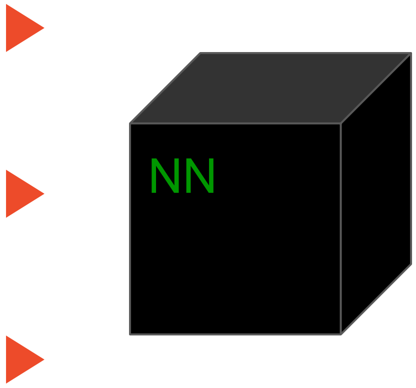
3gram

inputs	outputs
“0”, “0”	“0”
“0”, “0”	“45”
“0”, “45”	“0”
“45”, “0”	“0”
“0”, “0”	“90”

4gram

inputs	outputs
“0”, “0”, “0”	“45”
“0”, “0”, “45”	“0”
“0”, “45”, “0”	“0”
“45”, “0”, “0”	“90”

Training the neural network with variable length sequences



Training the neural network with variable length sequences

Moving direction:
["0" "0" "0" "45" "0" "0" "90"]

2gram

inputs	outputs
"0"	"0"
"0"	"0"
"0"	"45"
"45"	"0"
"0"	"0"
"0"	"90"

3gram

inputs	outputs
"0", "0"	"0"
"0", "0"	"45"
"0", "45"	"0"
"45", "0"	"0"
"0", "0"	"90"

4gram

inputs	outputs
"0", "0", "0"	"45"
"0", "0", "45"	"0"
"0", "45", "0"	"0"
"45", "0", "0"	"90"

Training the neural network with variable length sequences

Moving direction:
[“0” “0” “0” “45” “0” “0” “90”]

2gram

inputs	outputs
0, *0*, “0”	“0”
0, *0*, “0”	“0”
0, *0*, “0”	“45”
0, *0*, “45”	“0”
0, *0*, “0”	“0”
0, *0*, “0”	“90”

3gram

inputs	outputs
0, “0”, “0”	“0”
0, “0”, “0”	“45”
0, “0”, “45”	“0”
0, “45”, “0”	“0”
0, “0”, “0”	“90”

4gram

inputs	outputs
“0”, “0”, “0”	“45”
“0”, “0”, “45”	“0”
“0”, “45”, “0”	“0”
“45”, “0”, “0”	“90”

Note: *0* is [0, 0, 0, 0, 0, 0, 0] in one-hot encoding.

Current works

Neighbor pattern feature:

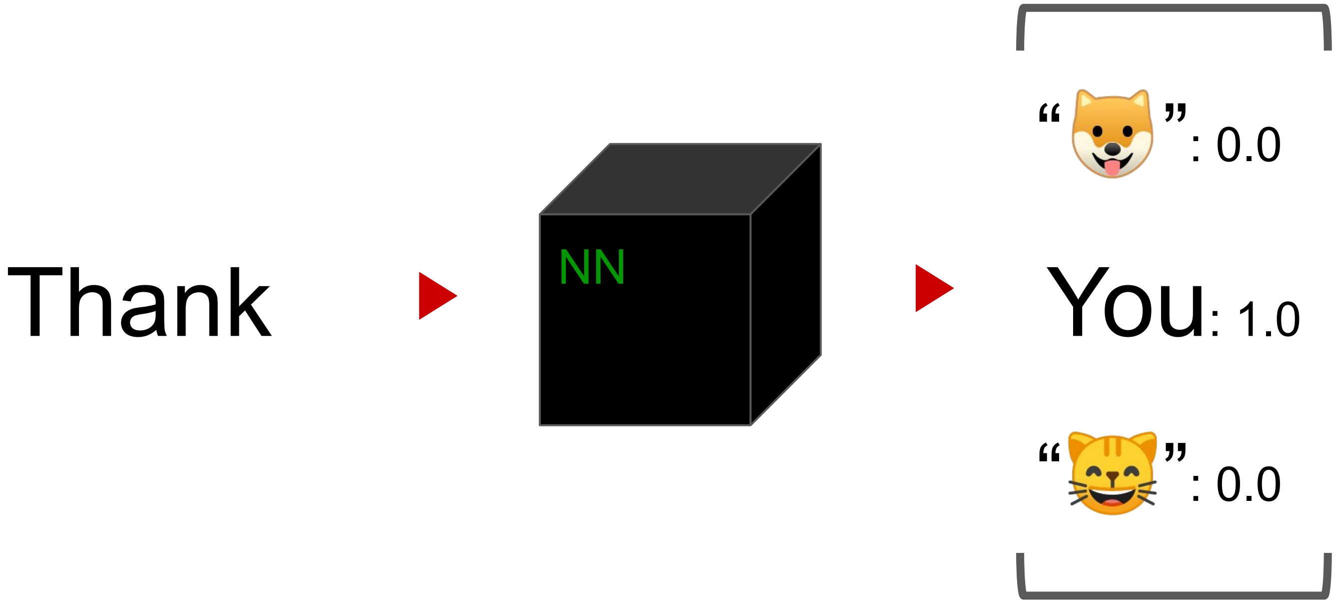
- 2-gram,
- 1-skip-bigram,
- 2-skip-bigram.

Moving direction feature:

- 5-gram,
- 10-gram,
- 15-gram.

Summary

- The neural language model can use less space in the memory than the conventional language model especially in the higher ngram models,
- The neural language model can be as accurate as the conventional language model,
- The neural language model can recognize multiple language models in a single network.



Thank
NN
“ ”: 0.0
You: 1.0
“ ”: 0.0