Technische Hochschule Rosenheim

Modul - Introduction to AI - part II (AI2)

Bachelor Programme AAI

11 - Word2Vec

Prof. Dr. Marcel Tilly

Faculty of Computer Science, Cloud Computing

Agenda

Technische Hochschule Rosenheim

On the menu for today:

- Word2Vec
  - Vector representation of words
  - Similarity (e.g. Cosine Similarity)

Organizational things

Technische Hochschule Rosenheim

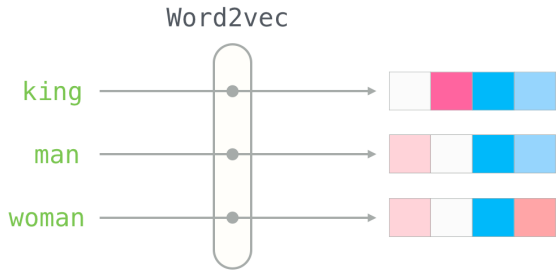
- Exam date: 25.07 at 10am (room tbd!)
- If you have not written AI1, just do exam AI2!
- In case you have written and passed AI1:
  - You can try AI2! You get the better grade at the end!
- Mock exam is available here: https://inf-git.fh-rosenheim.de/aai-ai2/11_exercise
  - We are going to talk about the solution in excerise on Monday, 4.08!

Word2Vec

king – man + woman = queen

Word Embeddings and Word2vec

- **Word embedding** is one of the most popular representation of document vocabulary. It is capable of capturing context of a word in a document, semantic and syntactic similarity, relation with other words, etc.
  - Loosely speaking, they are vector representations of a particular word.
- **Word2Vec** is one of the most popular technique to learn word embeddings using shallow neural network. It was developed by Tomas Mikolov in 2013 at Google.



Task 1: Lets think about *color*

Technische Hochschule Rosenheim

- How is *color* represented in computer?

Task 1: Lets think about *color*

Technische Hochschule Rosenheim

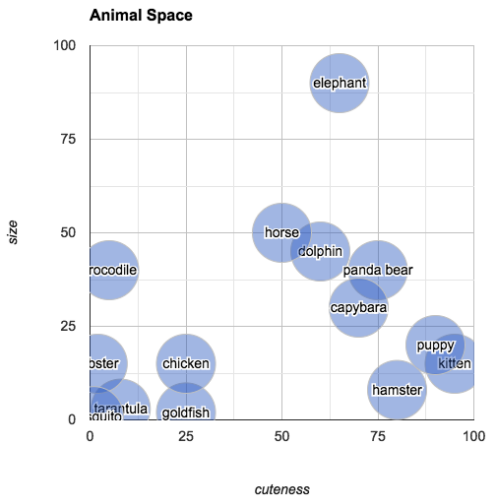
- How is *color* represented in computer?

```
class RGB():

def __init__(self, r, g, b):
    self.r = r
    self.g = g
    self.b = b

def rgb_format(self):
  return '(r{:02},g{:02},b{:02})'.format(self.r,self.g,self.b)
BLUE = RGB(0, 0, 255)
BLUEVIOLET = RGB(138, 43, 226)
BURNTSIENNA = RGB(138, 54, 15)
CADETBLUE3 = RGB(122, 197, 205)
CORNSILK1 = RGB(255, 248, 220)
DARKGOLDENROD4 = RGB(139, 101, 8)
colors = [BLUE, BLUEVIOLET, BURNTSIENNA, CADETBLUE3, CORNSILK1, DARKGOLDENROD4]
for c in colors:
print(c.rgb_format())
```

Similarity and simple linear algebra

- We'll begin by considering a small subset of English: words for animals. Our task is to be able to write computer programs to find similarities among these words and the creatures they designate. To do this, we might start by making a spreadsheet of some animals and their characteristics and display them in an *aninmal_space*.

	cuteness (0–100)	size (0–100)
kitten	95	15
hamster	80	8
tarantula	8	3
puppy	90	20
crocodile	5	40
dolphin	60	45
panda bear	75	40
lobster	2	15
capybara	70	30
elephant	65	90
mosquito	1	1
goldfish	25	2
horse	50	50
chicken	25	15



Distance

Technische Hochschule Rosenheim

One way of calculating how "far apart" two points are is to find their Euclidean distance. (This is simply the length of the line that connects the two points.) For points in two dimensions, Euclidean distance can be calculated with the following Python function:

```
import math
def distance2d(x1, y1, x2, y2):
    return math.sqrt((x1 - x2)**2 + (y1 - y2)**2)
```

So, the distance between "capybara" (70, 30) and "panda" (74, 40):

```
distance2d(70, 30, 75, 40) # panda and capybara
```

... is less than the distance between "tarantula" and "elephant":

```
distance2d(8, 3, 65, 90) # tarantula and elephant
```

Distance

Technische Hochschule Rosenheim

Modeling *words* in this way has a few other interesting properties.

- You can pick an arbitrary point in "animal space" and then find the animal closest to that point. If you imagine an animal of size 25 and cuteness 30, you can easily look at the space to find the animal that most closely fits that description: the **chicken**.
- You can also answer questions like: what's halfway between a chicken and an elephant? Simply draw a line from "elephant" to "chicken," mark off the midpoint and find the closest animal. (According to our chart, halfway between an elephant and a chicken is a horse.)
- You can also ask: what's the difference between a hamster and a tarantula? According to our plot, it's about seventy five units of cute (and a few units of size).
- The relationship of "difference" is an interesting one, because it allows us to reason about analogous relationships.

```
Tarantulas are to hamsters as chickens are to kittens.
```

One-hot encoding

Technische Hochschule Rosenheim

Consider the following similar sentences: "Have a good day" and "Have a great day".

- They hardly have different meaning.
- If we construct an exhaustive vocabulary (let's call it V), it would have V = {Have, a, good, great, day}.

Now, let us create a **one-hot encoded** vector for each of these words in V.

- Length of our one-hot encoded vector would be equal to the size of V (=5).
- We would have a vector of zeros except for the element at the index representing the corresponding word in the vocabulary.

e.g.

```
Have = [1,0,0,0,0] ; a=[0,1,0,0,0] ;
good=[0,0,1,0,0] ; great=[0,0,0,1,0] ;
day=[0,0,0,0,1]
```

Task 1: What is Cosine-Similarity

Technische Hochschule Rosenheim

What the Cosine-Similarity of "Have a good day" and "Have a great day"?

with

```
Have = [1,0,0,0,0]
a    = [0,1,0,0,0]
good = [0,0,1,0,0]
great= [0,0,0,1,0]
day  = [0,0,0,0,1]
```

How does Word2Vec work?

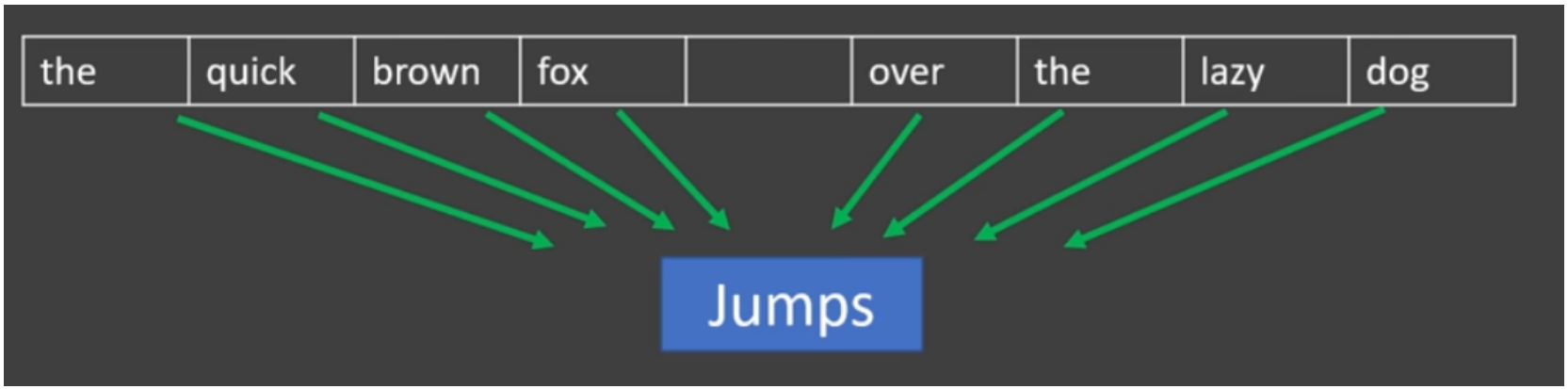
Technische Hochschule Rosenheim

- Word2Vec is a method to construct such an embedding.
- It can be obtained using two methods (both involving Neural Networks):
  - Skip Gram
  - Common Bag Of Words (CBOW)

CBOW Model

Technische Hochschule Rosenheim

This method takes the context of each word as the input and tries to predict the word corresponding to the context.



CBOW Model

Technische Hochschule Rosenheim

Consider our example: "Have a great day."

- Let the input to the Neural Network be the word: **great**.
  - Notice that here we are trying to predict a target word (day) using a single context input word great.
  - More specifically, we use the one hot encoding of the input word and measure the output error compared to one hot encoding of the target word (day).
  - In the process of predicting the target word, we learn the vector representation of the target word.

CBOW

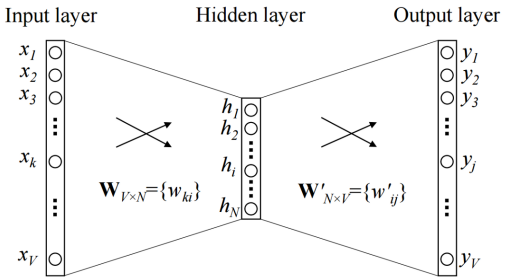
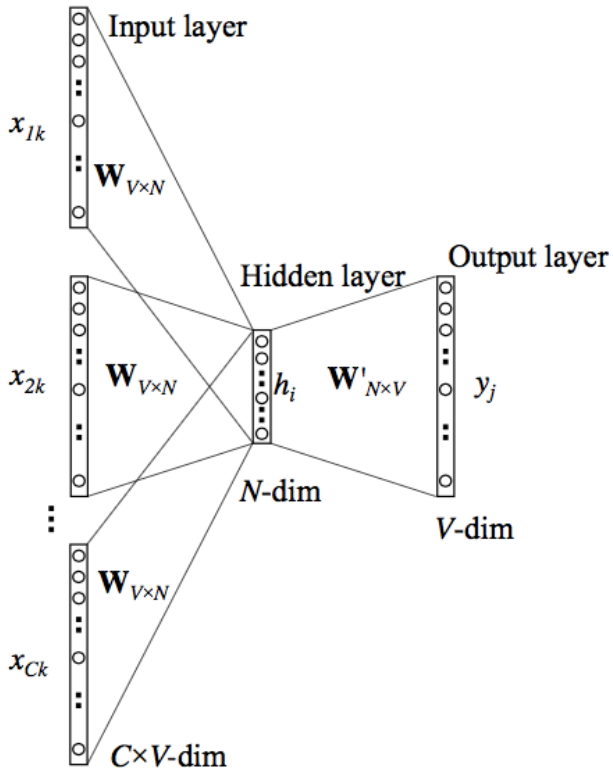


Figure 1: A simple CBOW model with only one word in the context

The input or the context word is a one hot encoded vector of size V. The hidden layer contains N neurons and the output is again a V length vector with the elements being the softmax values.

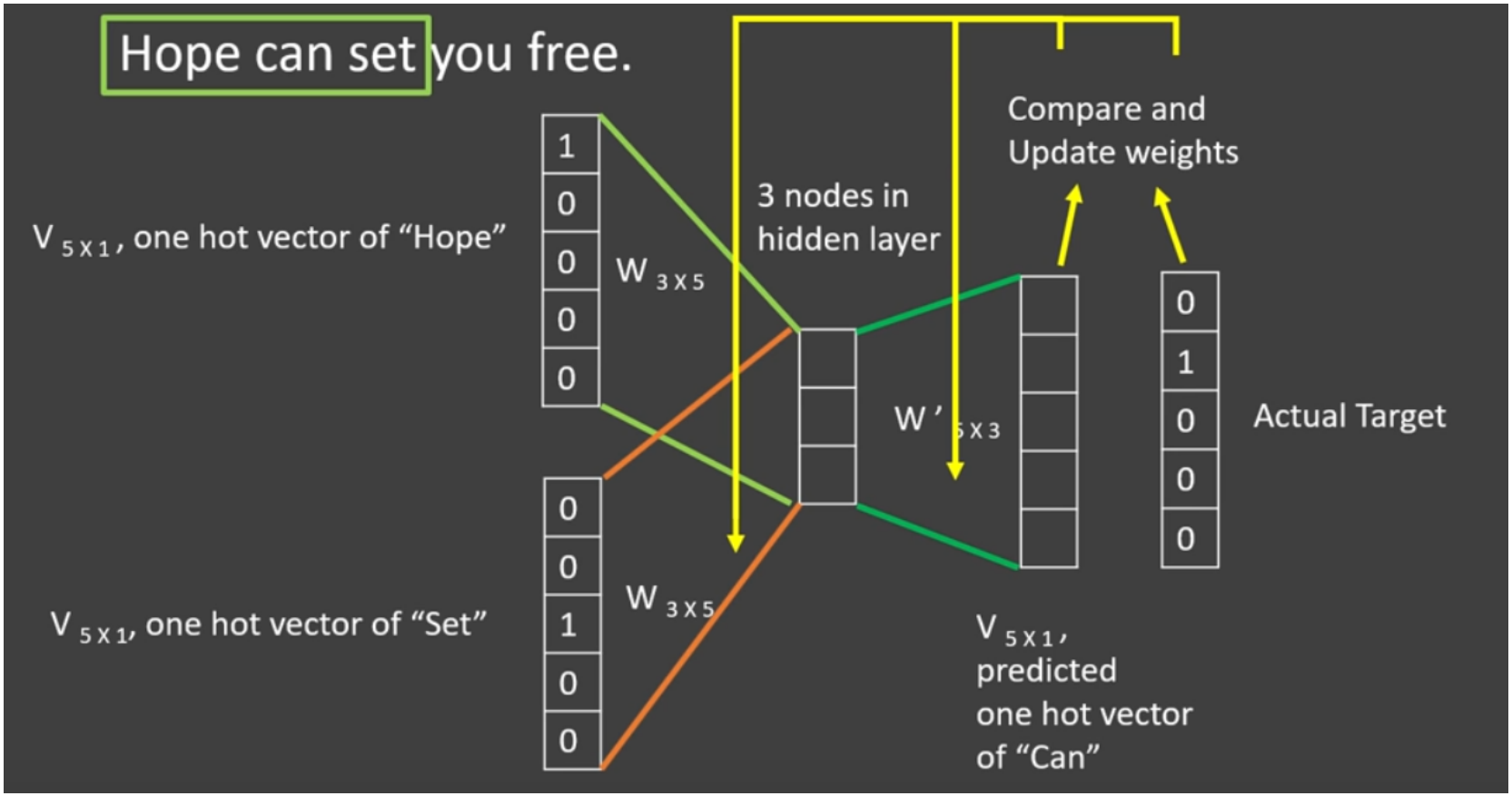
- W_{VxN} is the weight matrix that maps the input x to the hidden layer (V*N dimensional matrix)
- W'_{NxV} is the weight matrix that maps the hidden layer outputs to the final output layer (N*V dimensional matrix)

CBOW



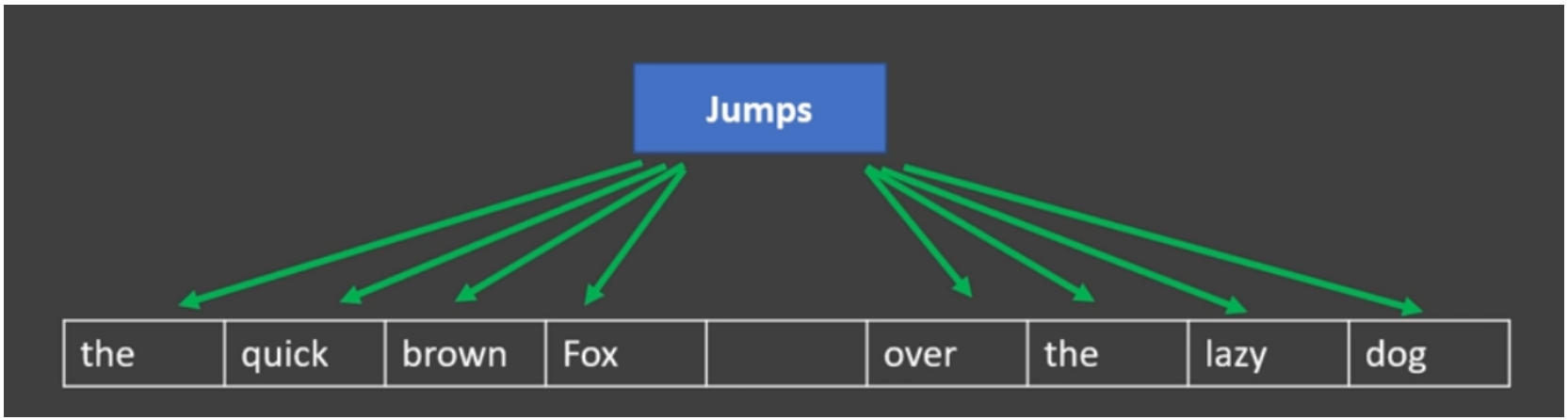
- We can use multiple context words to do the same.
- The above model takes C context words.
- When W_{VxN} is used to calculate hidden layer inputs, we take an average over all these C context word inputs.

CBOW in Action

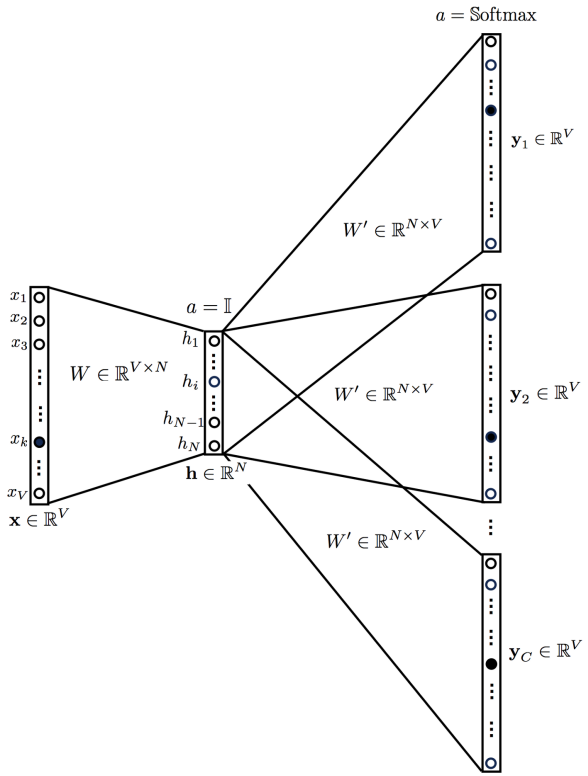


Skip-Gram Model

- Predicts the context words from target.

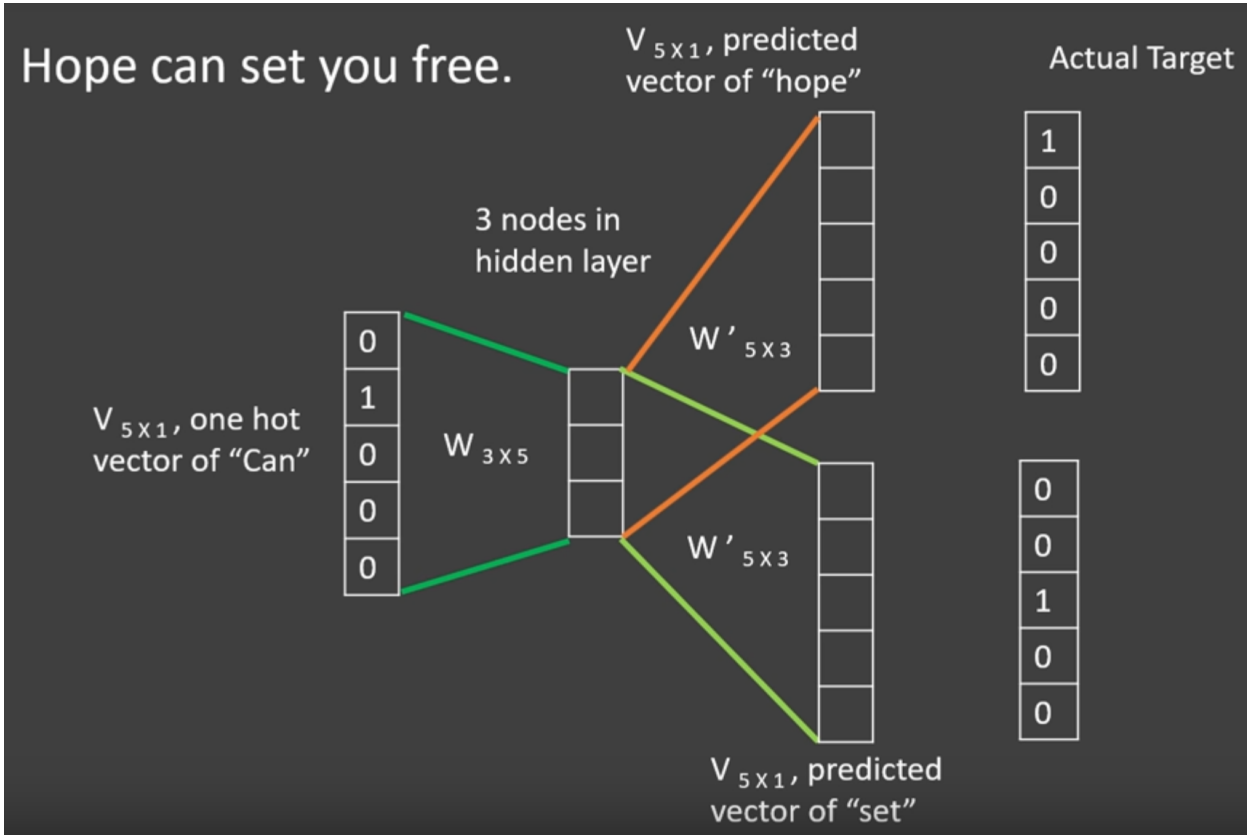


Skip-Gram Model

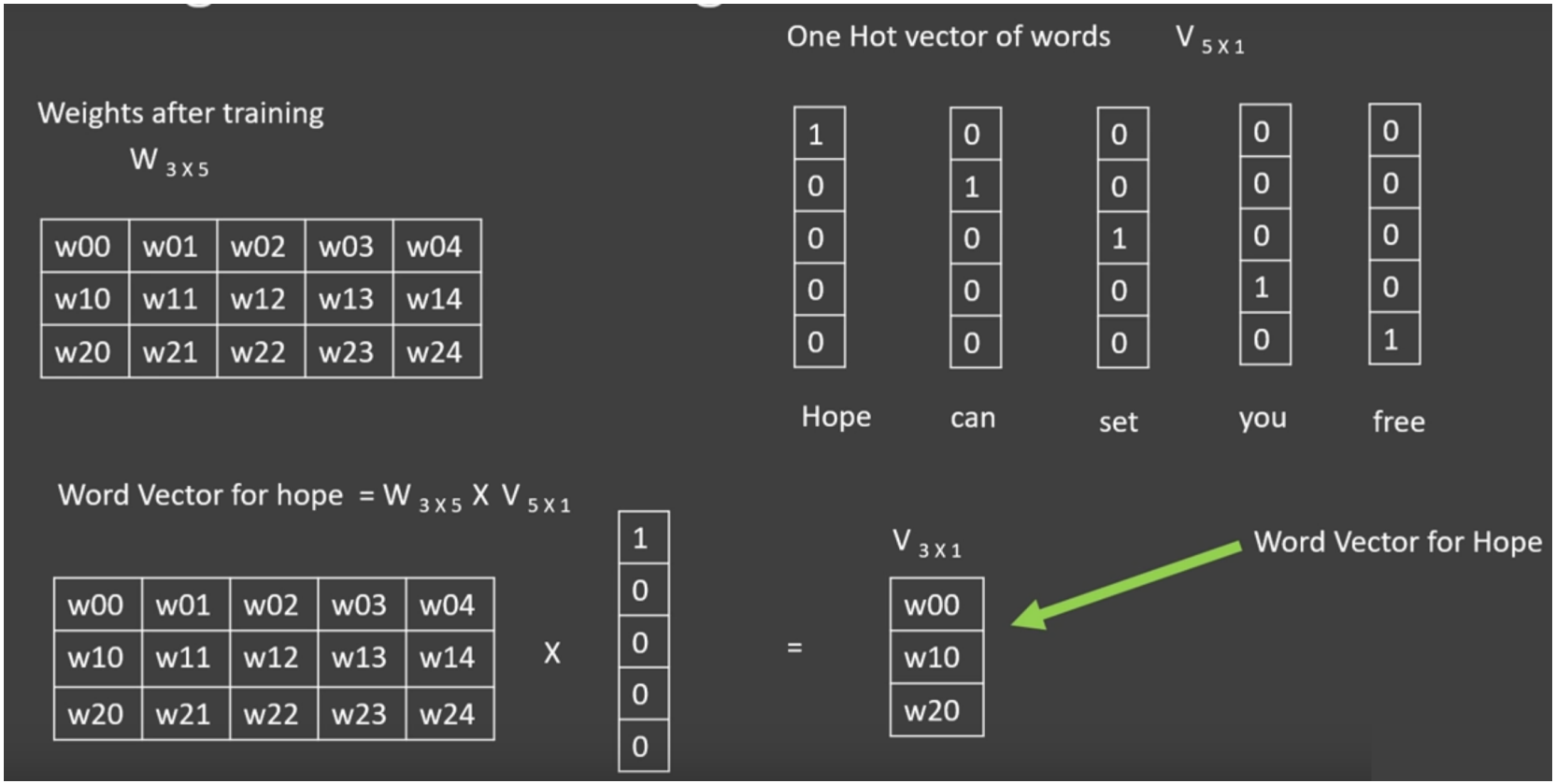


- This looks like multiple-context CBOW model just got flipped. To some extent that is true.
- We input the target word into the network. The model outputs C probability distributions.
- For each context position, we get C probability distributions of V probabilities, one for each word.

Skip-Gram in Action



Towards Word Embeddings



Conclude

Technische Hochschule Rosenheim

- Both have their own advantages and disadvantages.
- According to Mikolov, Skip Gram works well with small amount of data and is found to represent rare words well.
- On the other hand, CBOW is faster and has better representations for more frequent words.
- Advantages
  - It scales

  – Train on billion word corpora in limited time

  – Paralleization is possible
- Word embeddings trained by one can be used by others
- Incremental training
  - Train on one piece of data, save results, continnue training later on

– Gensim word2vec: https://radimrehurek.com/gensim/models/word2vec.html

What can you do with it?

Technische Hochschule Rosenheim

```
A is to B as C is to ?
```

This is the famous example:

```
vector(king) - vector(man) + vector(woman) = vector(queen)
```

If you substract the vector for "man" from the one for "king" and add the vector for "woman", the vector closest to the one you end up with turns out to be the one for "queen".

More on Word Embeedings

Technische Hochschule Rosenheim

This is a word embedding for the word “king” (trained on Wikipedia):

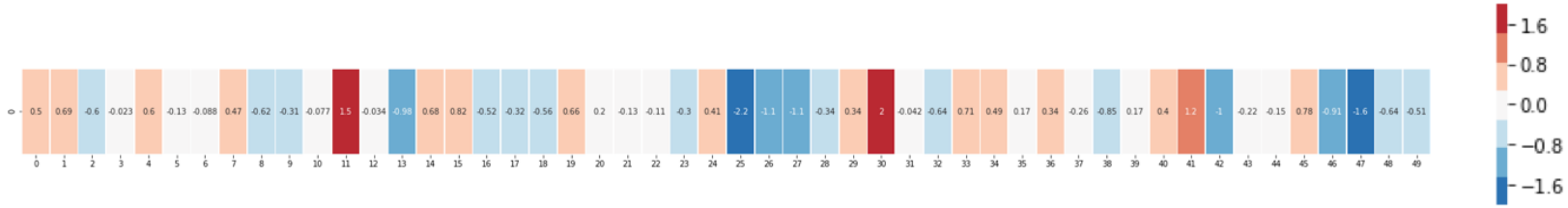
[0.50451 , 0.68607 , -0.59517 , -0.022801, 0.60046 , -0.13498 , -0.08813 , 0.47377 , -0.61798 , -0.31012 , -0.076666, 1.493 , -0.034189, -0.98173 , 0.68229 , 0.81722 , -0.51874 , -0.31503 , -0.55809 , 0.66421 , 0.1961 , -0.13495 , -0.11476 , -0.30344 , 0.41177 , -2.223 , -1.0756 , -1.0783 , -0.34354 , 0.33505 , 1.9927 , -0.04234 , -0.64319 , 0.71125 , 0.49159 , 0.16754 , 0.34344 , -0.25663 , -0.8523 , 0.1661 , 0.40102 , 1.1685 , -1.0137 , -0.21585 , -0.15155 , 0.78321 , -0.91241 , -1.6106 , -0.64426 , -0.51042]

> It’s a list of 50 numbers.

- We can’t tell much by looking at the values.
- What can we do?

Visualize it

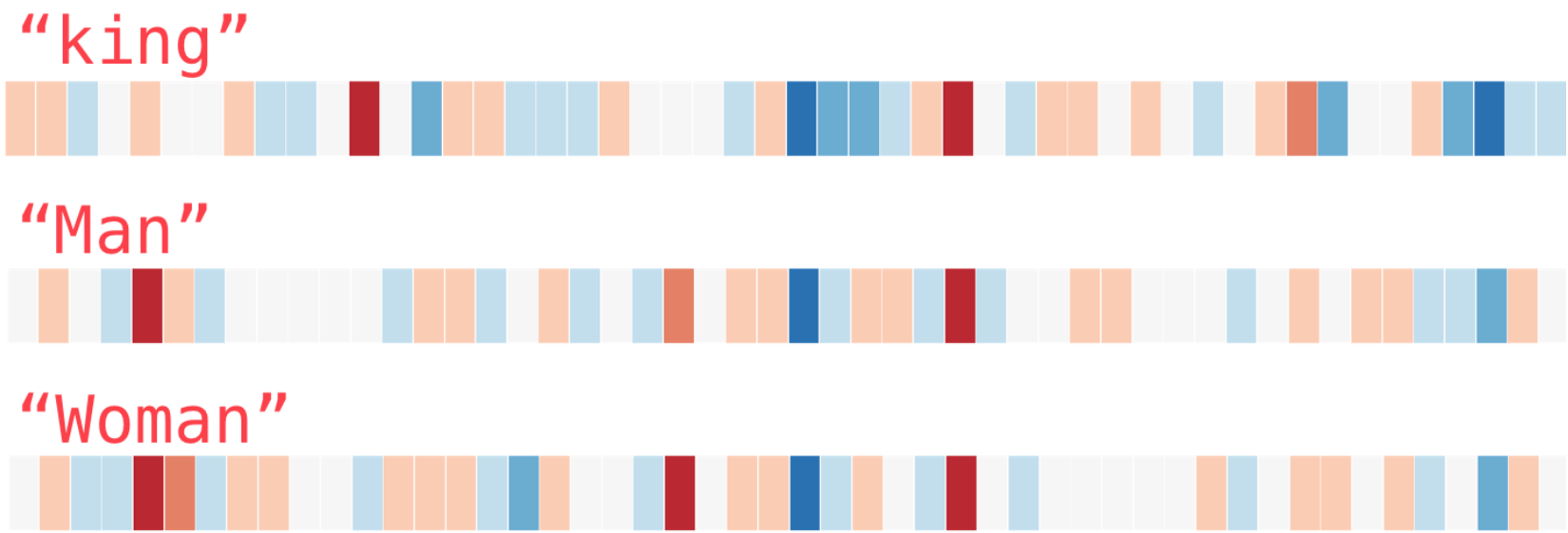
Let's color code the cells based on their values (red if they're close to 2, white if they're close to 0, blue if they're close to -2):



taken from https://jalammar.github.io/illustrated-word2vec/

Compare

Technische Hochschule Rosenheim

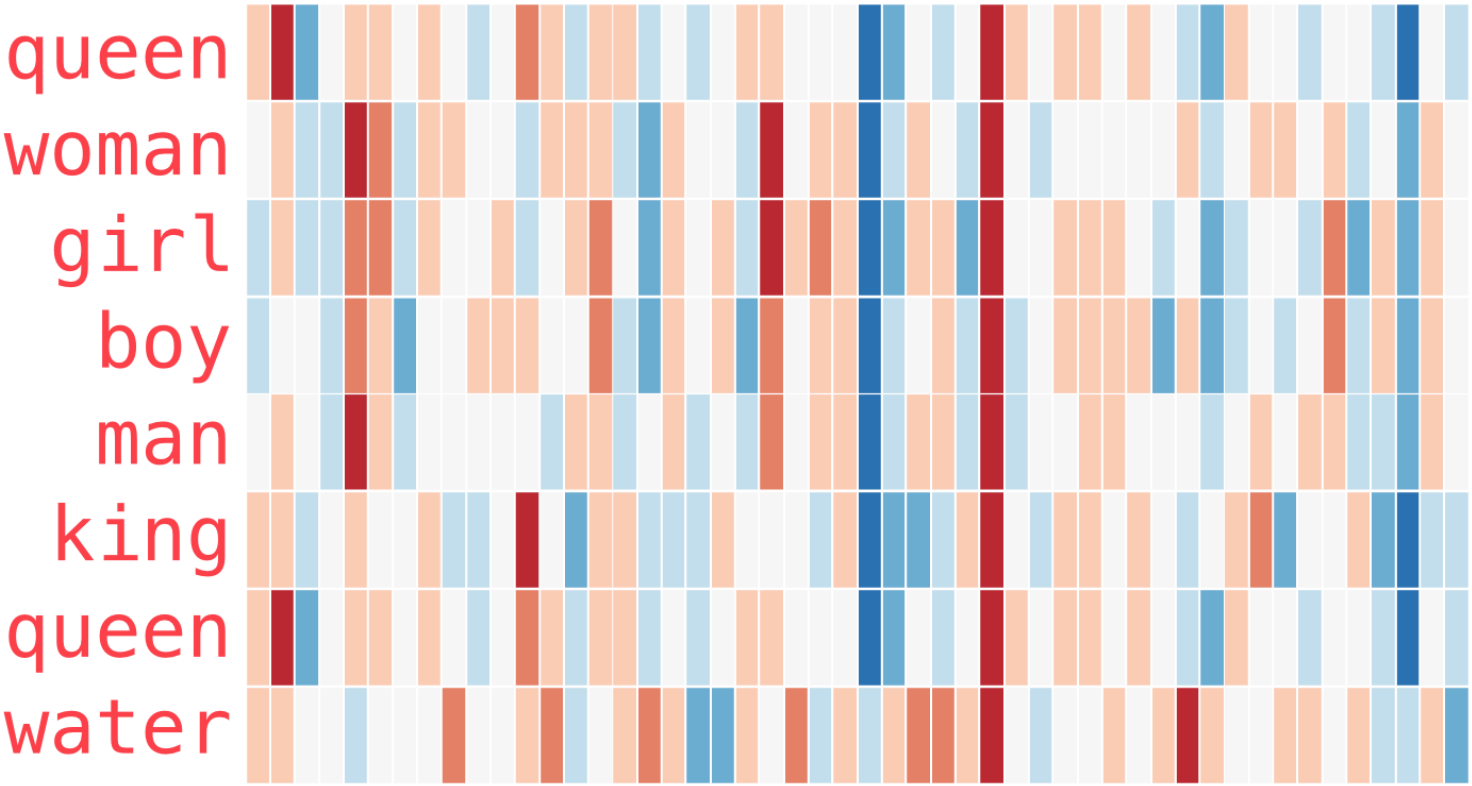


> See how “Man” and “Woman” are much more similar to each other than either of them is to “king”? This tells you something. These vector representations capture quite a bit of the information/meaning/associations of these words.

Task

Technische Hochschule Rosenheim

What do you observe?



Observations

Technische Hochschule Rosenheim

A few things to point out:

- There's a straight red column through all of these different words. They're similar along that dimension (and we don't know what each dimensions codes for).
- You can see how "woman" and "girl" are similar to each other in a lot of places. The same with "man" and "boy", "boy" and "girl" also have places where they are similar to each other, but different from "woman" or "man". Why?
- All but the last word are words representing people. There is an object (water) to show the differences between categories. You can, for example, see that blue column going all the way down and stopping before the embedding for "water".
- There are clear places where "king" and "queen" are similar to each other and distinct from all the others.

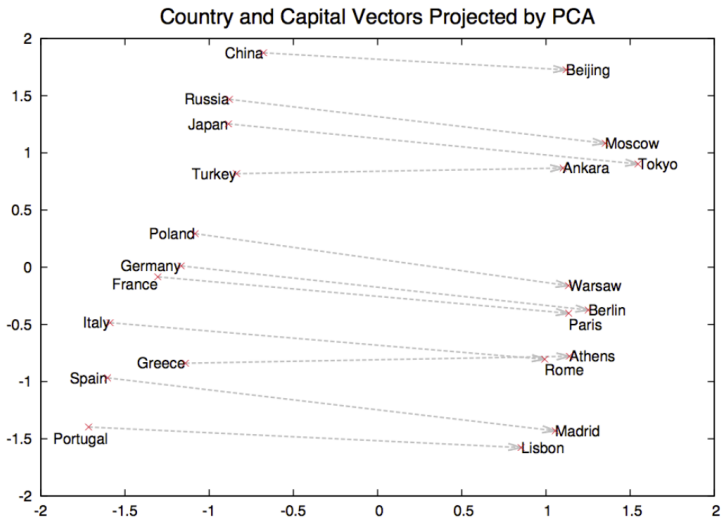
What can you do with it?

Technische Hochschule Rosenheim

`A is to B as C is to ?`

More interesting: Which country does a particular city belong to?

`France is to Paris as Germany is to Berlin.`



taen from T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. 2013.

Summary

Technische Hochschule Rosenheim

Lessons learned today:

- Word Embeddings
- Word2Vec
  - CBOW Model
  - Skip-gram Model