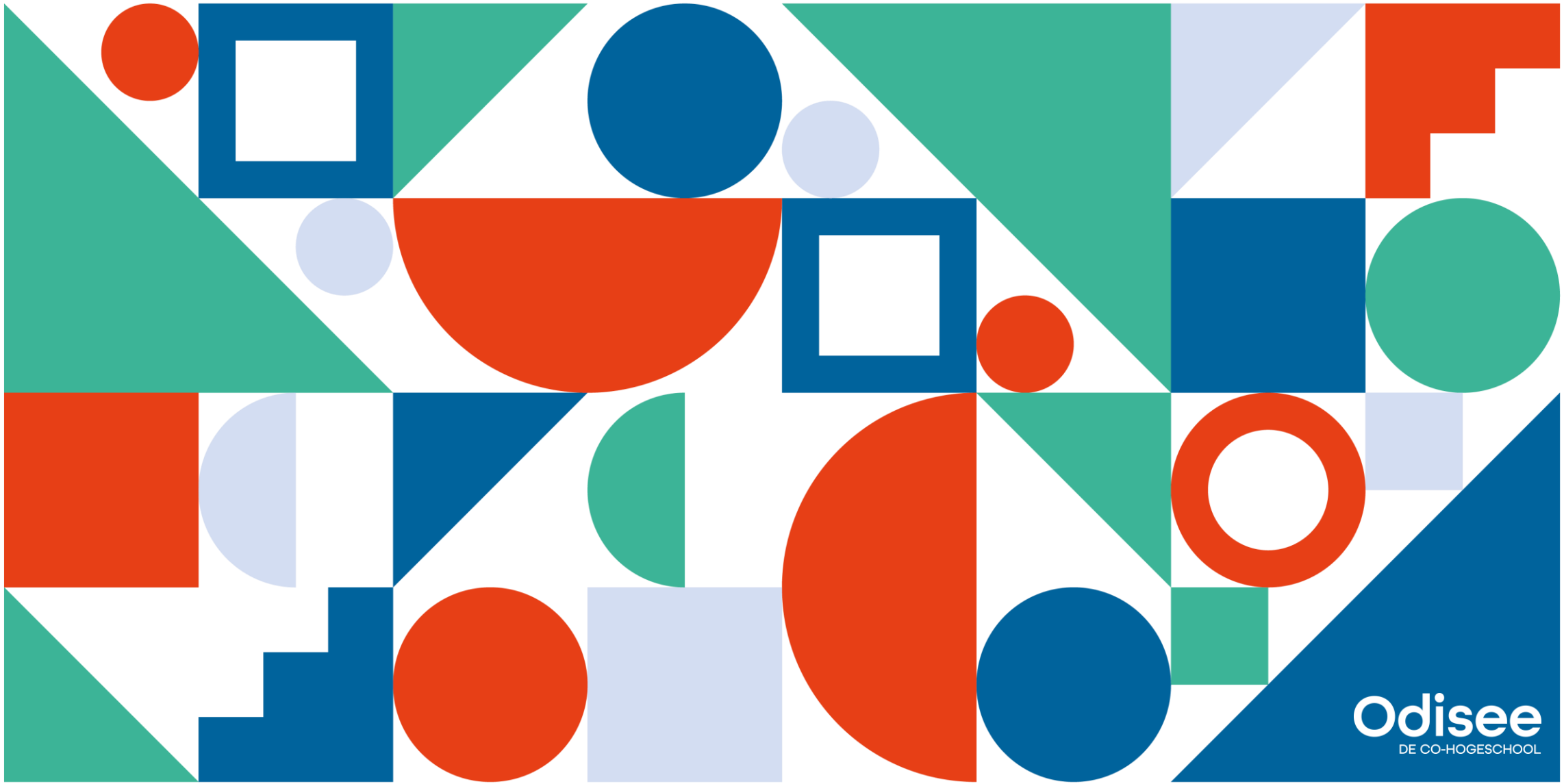

Odisee
DE CO-HOGESCHOOL

Text Processing

Wat zijn de moeilijkheden om te werken met tekstuele data?

Wat zijn de moeilijkheden om te werken met tekstuele data?

- Taal is ambigue: synoniemen, beeldspraak, spreekwoorden, …
- Taal bevat abstracte concepten
  - Subtiele relaties tussen woorden die de betekenis van de zin kunnen verwoorden
- Hoge dimensionaliteit
  - Heel veel woorden (10.000-en) en lengte van de input is variabel.

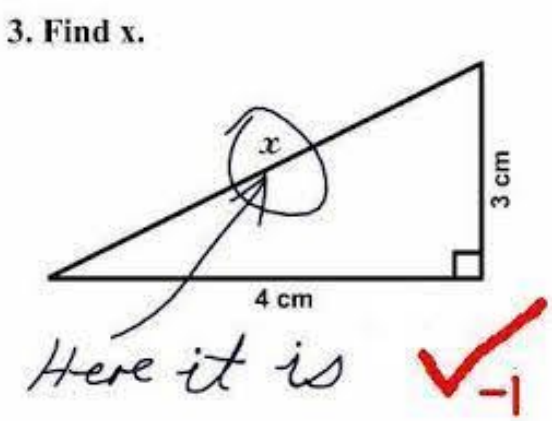
Waar gaan de volgende artikels over?

Twee agenten en brandweerman gewond bij betoging tegen circulatieplan in Schaarbeek: “Dit kunnen we niet tolereren”

Wie betoogde?

Thibaut Courtois boos na nederlaag in Leipzig: “We sliepen nog op het veld”

Is dit letterlijk?



Hoe hebben we deze moeilijkheden opgelost in vorige vakken?

- Tokenize de string
- Doe eventueel wat data cleaning op de tokens
- Geef elk woord een aparte klasse
- Tel hoeveel keer elk woord voorkomt
  - Dit geeft een vector/tensor van de features/inputs
- Deze tensor kan gebruikt worden voor classificatie/regressie

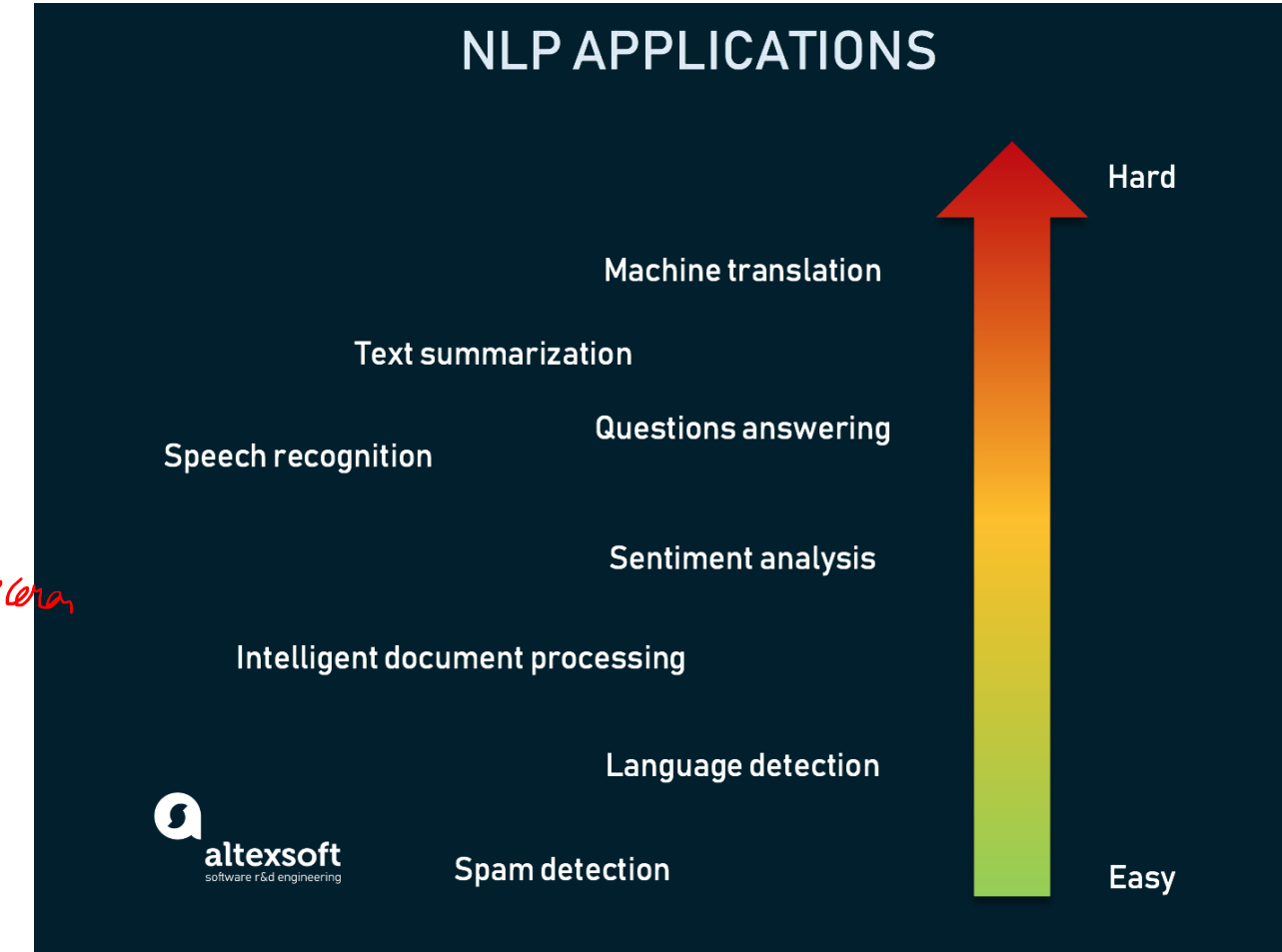
Gaat deze aanpak werken voor alle NLP-problemen?

- Enkel voor eenvoudigere toepassingen
- Voor complexere toepassingen
  - Meer informatie nodig over de context
  - Volgorde van woorden ook belangrijk

Sequenties

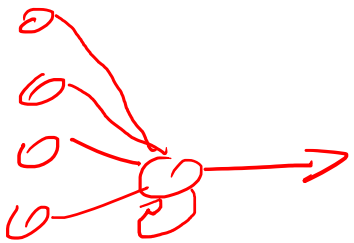
Niet heel de zin in 1 Keer Classificeren

→ woord per woord



Hoe kunnen we deze problemen oplossen?

- Complexere preprocessingen stappen nodig voor volgorde en context
- Er moet een vorm van geheugen zijn voor de context bij te houden tussen verschillende inputs
  - Gebruik de output / state van een neural network als input
  - Recurrentie



- Goed artikel: https://medium.com/analytics-vidhya/natural-language-processing-from-basics-to-using-rnn-and-lstm-ef6779e4ae66

Preprocessing

Transformaties van tekst

- Tokenization
  - Text naar woorden (typisch op spatie)
- Stemming
  - Brute manier om het einde van een woord af te kappen
  - Vervoeging/achtervoegsel te verwijderen
  - Niet noodzakelijk een correct woord
- Lemmatization
  - Intelligentere manier om het einde van een woord te verwijderen
  - Bijvoorbeeld meervouden verwijderen
  - Basis woord is een correct woord

Transformaties van tekst

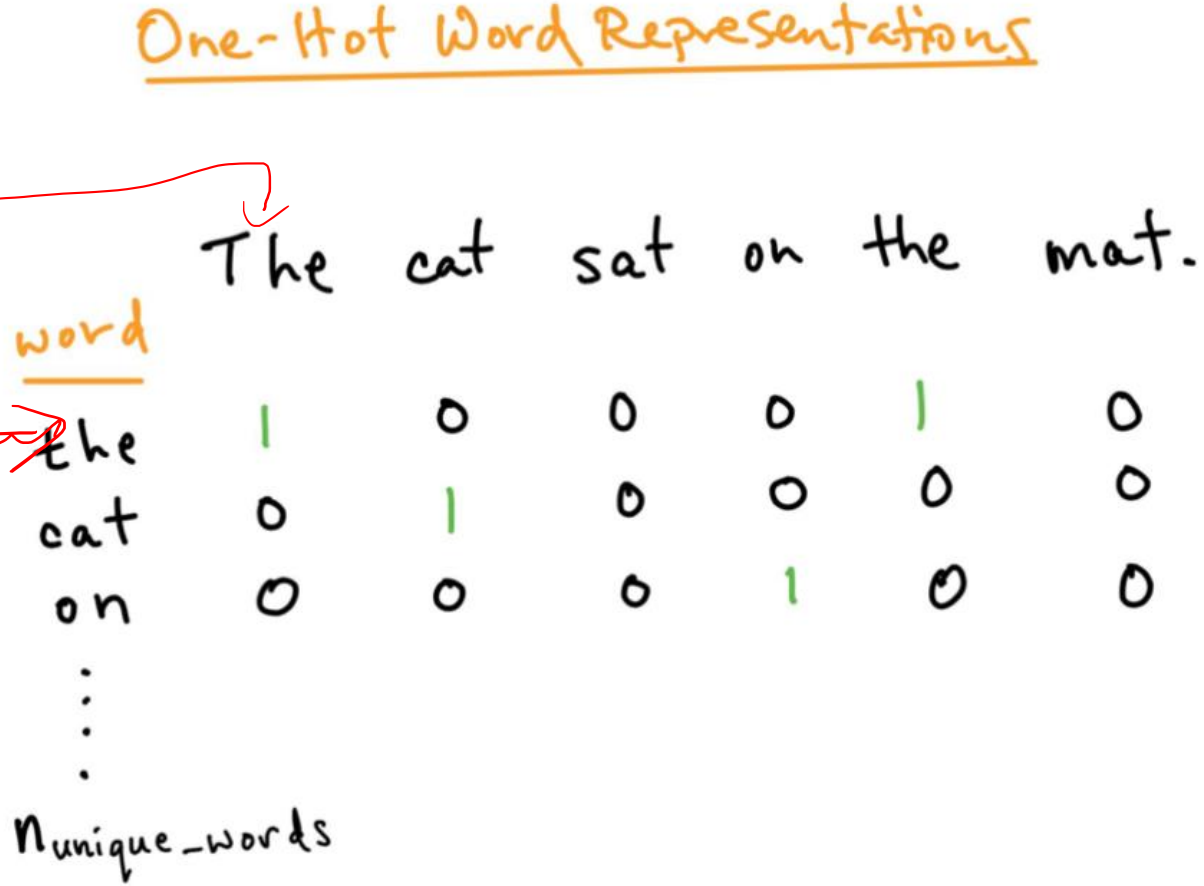
- N-Grams
  - Combineer nabijgelegen woorden in groepen
  - Natural Language Processing is essential to Computer Science.
    - 1-gram = tokenize
      - Natural, language, Processing, is, essential, to, Computer, Science
    - 3-gram
      - Natural Language Processing, Language Processing is, Processing is essential, is essential to, essential to Computer, to Computer Science

- N-grams zijn heel belangrijk voor traditionele wiskunde processen in NLP

Hoe data voorstellen

multi-hot: | the 1 | cat 1 | on 1 | floor 0 | . . . . |

- Tf-idf
- One-hot encodings
- Multi-hot encodings

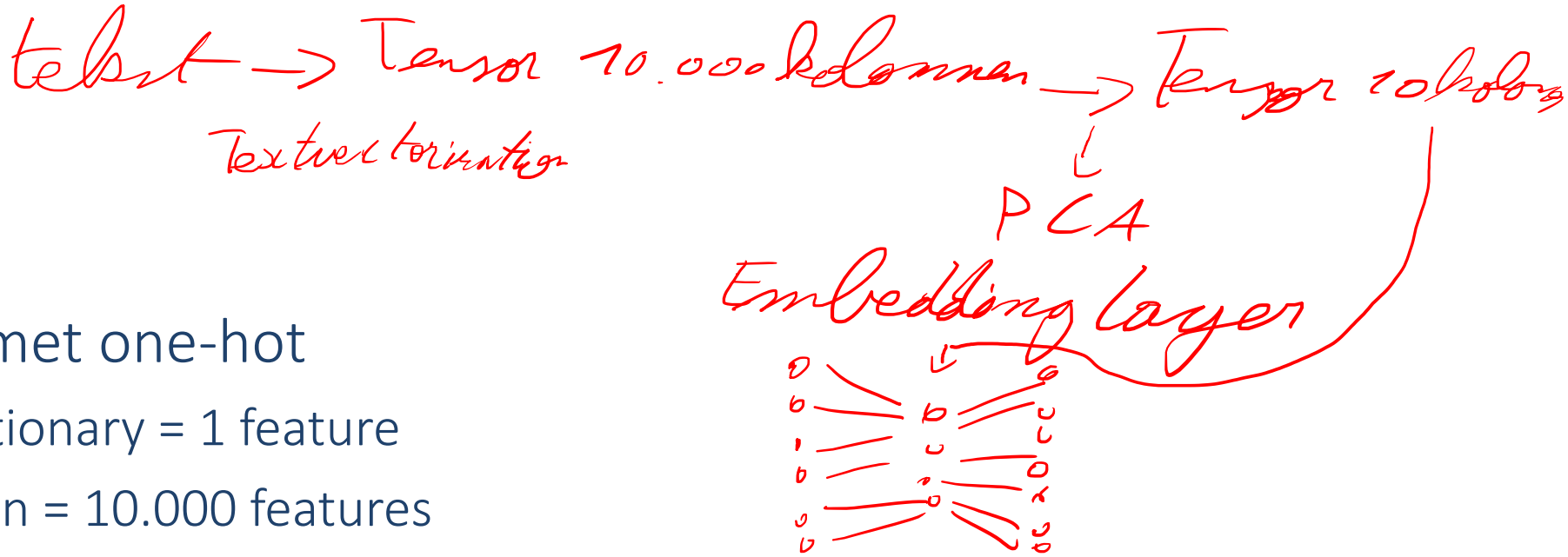


Transformaties in Tensorflow

- tf.keras.layers.TextVectorization
  - Kan gebruikt worden voor tokenization
  - Standaardisatie van de tekst

  - Geen lemmatization/stemming -> geen informatieverlies
  - Maakt ook n-grams mogelijk

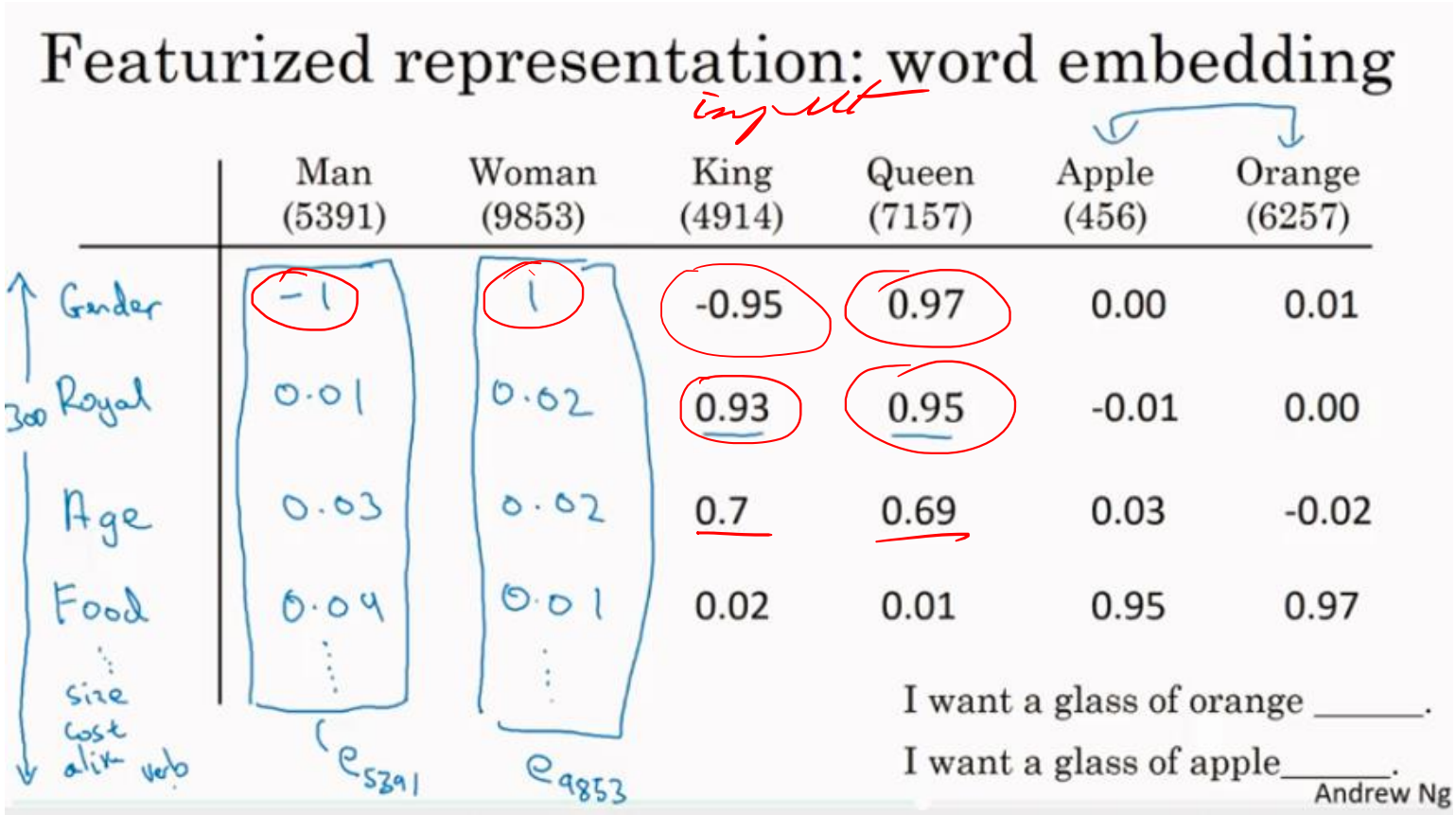
Embedding layer



- Text vectorization met one-hot
  - Elk woord in de dictionary = 1 feature
  - Dus 10.000 woorden = 10.000 features
  - Dit is niet bruikbaar omdat de opslag te groot is en te veel rekenkracht vraagt
- Embedding layer
  - Zet deze 10.000-en features om naar een tensor van vaste (kleinere) lengte
    - Gelijke woorden naar gelijke vectors
  - Er bestaan voorgetrainde lagen hiervoor
  - Kan je ook zelf trainen maar kan redelijk wat geheugen in beslag nemen

Embedding layer - resultaat

- Gelijkaardige woorden resulteren in gelijkaardige vectors
  - Soort van feature extraction



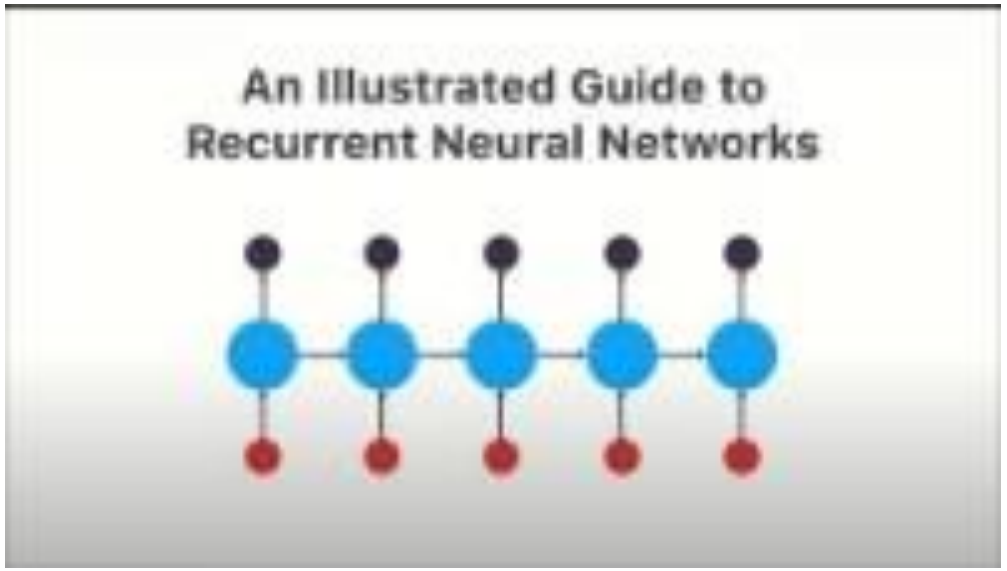
Een zin is een sequentie van woorden

- De lengte van de sequentie wordt dus bepaald door de zin
  - Training runs/Tensors verwachten gelijke dimensies voor elk voorbeeld
    - Dit maakt het mogelijk om alles in batches uit te voeren wat de uitvoering versneld
    - Er zijn mogelijkheden om te werken met dynamische sequentie lengtes maar vereisen extra werk
  - Padding/Truncating kan gebruikt worden om een zin om te zetten naar een vaste lengte
    - https://www.tensorflow.org/api_docs/python/tf/keras/utils/pad_sequences
    - Preprocessing (niet in sequentieel model)
    - RNN moet weten dat er padding is
      - Gebruik een Masking layer of gebruik de mask_zero parameter in de Embedding layer

Geheugen in neuraal netwerk

Hoe kan een neuraal netwerk een geheugen hebben?

- Maak een lus
  - Gebruik de output (of een deel ervan) als input

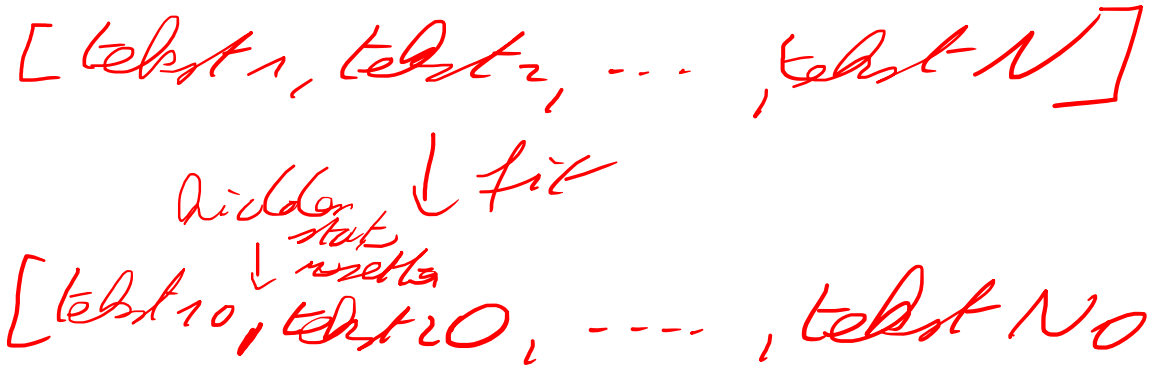


https://towardsdatascience.com/illustrated-guide-to-recurrent-neural-networks-79e5eb8049c9

	ANN	CNN	RNN
Basics	One of the simplest types of neural networks.	One of the most popular types of neural networks.	The most advanced and complex neural network.
Structural Layout	Its simplicity comes from its feed forward nature — information flows in one direction only.	Its structure is based on multiple layers of nodes including one or more convolutional layers.	Information flows in different directions, which gives it its memory and self-learning features.
Data Type	Fed on tabular and text data.	Relies on image data.	Trained with sequence data.
Complexity	Simple in contrast with the other two models.	Considered more powerful than the other two.	Fewer features than CNN but powerful due to its self-learning & memory potential.
Commendable Feature	Ability to work with incomplete knowledge and high fault tolerance.	Accuracy in recognizing images.	Memory and self-learning.
Feature type: spatial recognition	No	Yes	No
Feature type: Recurrent connections	No	No	Yes
Main Drawback	Hardware dependence.	Large training data required.	Slow and complex training and gradient concerns.
Uses	Complex problem solving such as predictive analysis.	Computer vision including image recognition	Natural language processing including sentiment analysis and speed recognition.

https://levity.ai/blog/neural-networks-cnn-ann-rnn

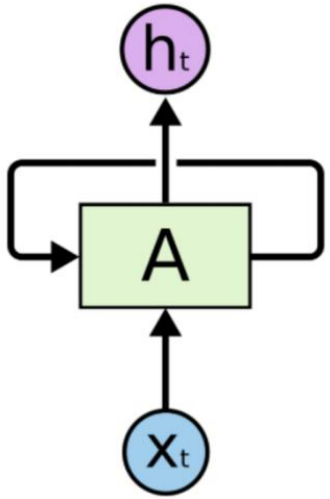
Recurrente Neurale Netwerken



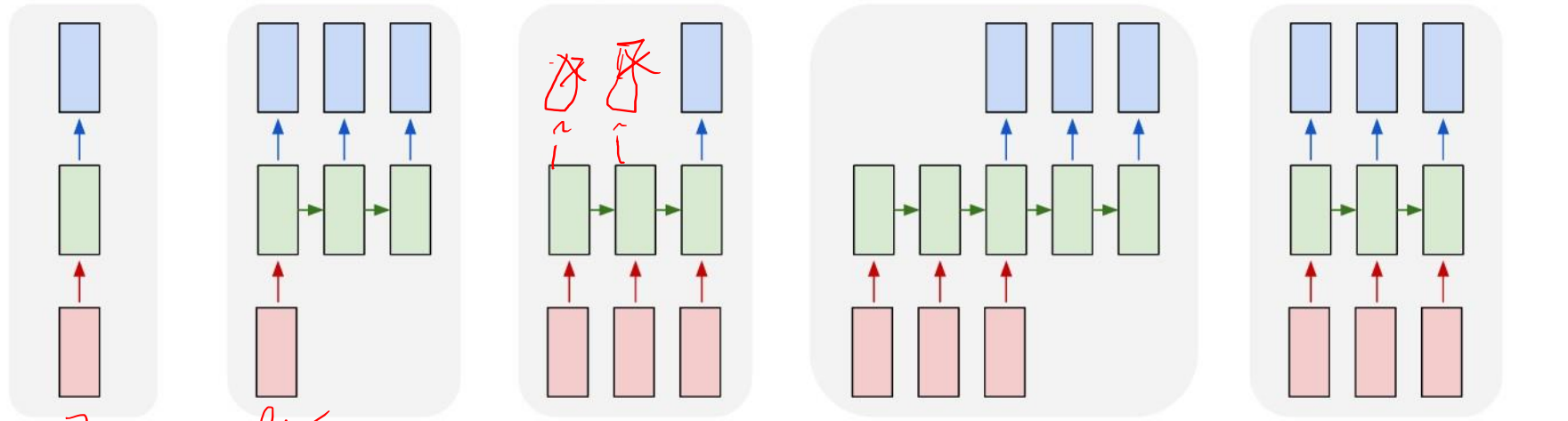
- Dit soort modellen worden ook Seq2Seq genoemd
  - Staat voor Sequence to Sequence
  - Ipv 1 input heb je een sequentie van inputs die 1 voor 1 aan het network gegeven wordt
    - 1 input = 1 tijdsstap
  - De output kan dan een sequentie zijn of een enkele waarde
    - Afhankelijk van waarin we geinteresseerd zijn
  - Tussen verschillende input sequenties wordt de state van het rnn gereset naar een start toestand

Werken met RNNs in Tensorflow

- Via een SimpleRNN layer
  - Verwerkt een volledige batch
- Via een RNNCell in een RNN layer
  - Stap per stap
  - Kan je ook gebruiken om complexere zaken te doen
    - Zoals LSTM (long short term memory) of GRU (gated recurrent unit)
- Inputs: 3 dimensies
  - Batch size, Sequence length, Sequence width

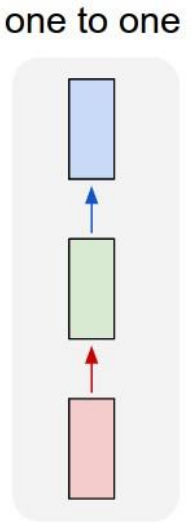


Architectuur van een Recurrent Neuraal Netwerk



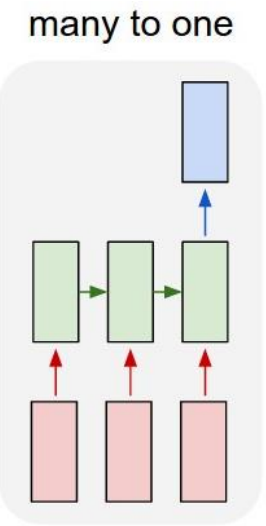
One-to-One RNN

- Dit is wat je reeds gedaan hebt (standaard neurale netwerken)
- Enkel de laatste output wordt bewaard
  - Classificatie van images, …



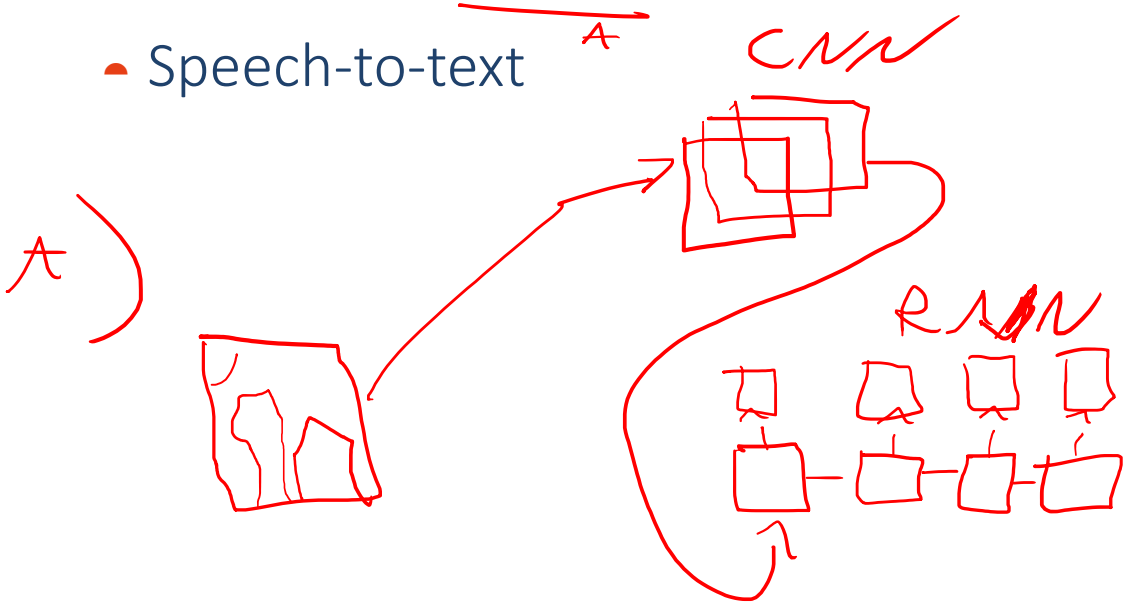
Many-To-One RNN

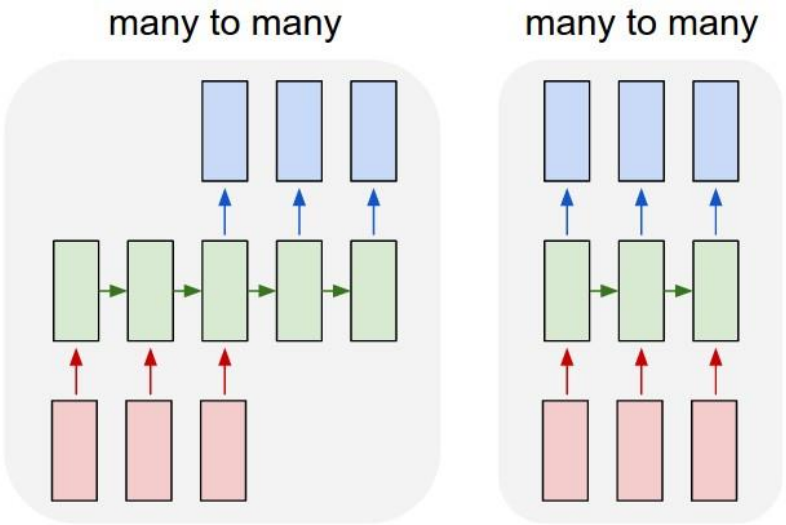
- Sequenties als input en slechts op 1 moment een output
- Applicaties
  - Text classification stuk per stuk / met geheugen
  - Sentiment analysis



Many-To-Many

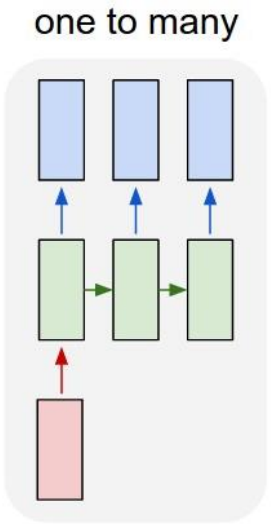
- Applicaties
  - Vertalen van taal naar taal
  - Frames van video classificeren
  - Speech-to-text



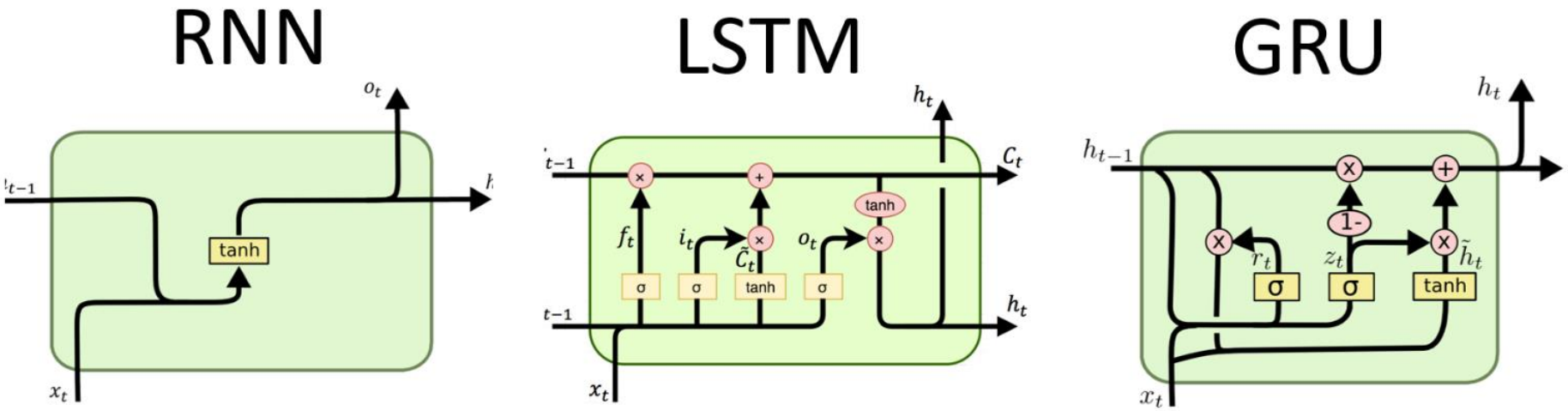


One-To-Many RNN

- 1 input en meerdere outputs
- Mogelijke applicatie:
  - Auto captioning van een beeld

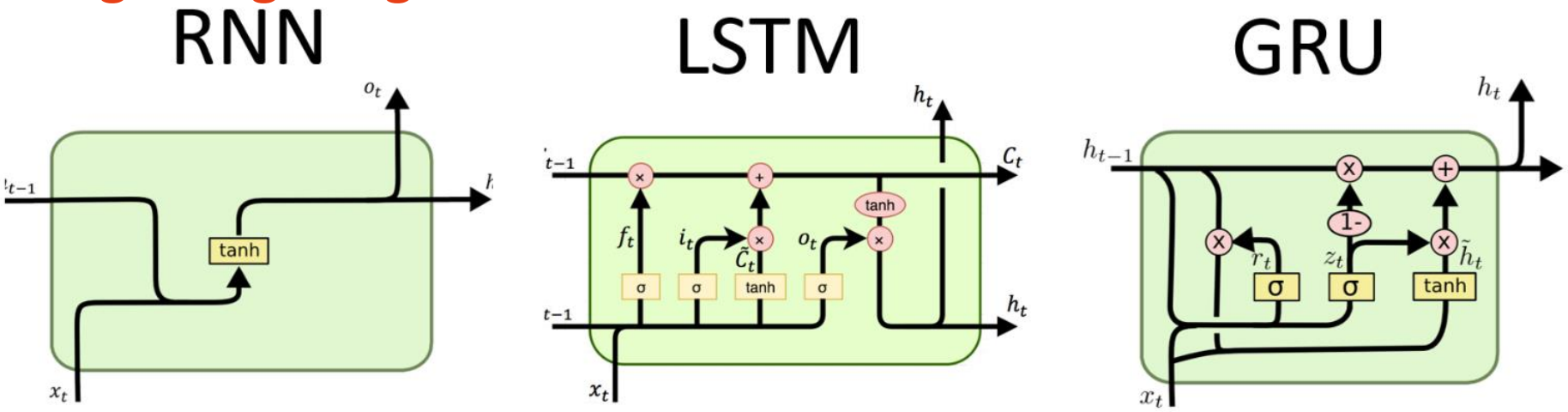


Complexere geheugenlagen dan RNN



- Long – short term memory (LSTM)
  - Langer geheugen dan standaard RNN
    - Hoeveel van de state behouden wordt wordt bepaald door de cell
  - Meer rekenintensief
  - Neiging tot overfitten maar dropout is moeilijk toe te passen

Complexere geheugenlagen dan RNN



- Gated Recurrent Units(GRU)
  - Minder popular dan LSTM
  - Minder opties voor het finetunen van de state die onthouden wordt