

Computer-Linguistische Anwendungen

CLA | B.Sc. | LMU



Vorlesung: Neuronale Netze

Philipp Wicke, PhD
Centrum für Sprach- und Informationsverarbeitung
Ludwig-Maximilians-Universität München
pwicke@cis.lmu.de



Übersicht

Themen der Vorlesung

- Einführung Neuronale Netze
- Anwendungsbeispiel
- **Recurrent Neural Networks (RNNs)**
- Convolutional Neural Networks (CNNs)



Rekurrente Neuronale Netze

Motivation

Wie kann man ...

... am besten eine Sequenz von Wörtern als Vektor repräsentieren?

... die gelernten Wort-Vektoren effektiv kombinieren?

... die für eine bestimmte Aufgabe relevanten Informationen (bestimmte Merkmale, bestimmter Wörter) behalten, unwesentliches unterdrücken?



Rekurrente Neuronale Netze

Motivation

Bei kurzen Phrasen ließe sich ein Durchschnitts Vektor bilden:

$$\begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix} = 1/3 \left(\begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \\ \bullet \end{bmatrix} \right)$$

London Symphony Orchestra

→ Was aber bei: *London Orchestra Symphony*?

Bei langen Phrasen ist dies nun nicht mehr ohne Probleme möglich:

$$\begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} = 1/18 \left(\begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} + \begin{bmatrix} \bullet \\ \bullet \\ \bullet \end{bmatrix} \right)$$

The sopranos was probably the last best show to air in the 90's. It's sad that it's over

Probleme: 1) Reihenfolge geht verloren. 2) Es gibt keine Parameter, die schon bei der Kombination zwischen wichtiger und unwichtiger Information (stopwords) unterscheiden können. (Erst der Klassifikator kann dies versuchen)

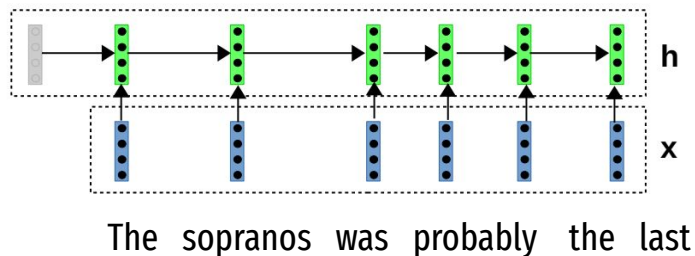


Rekurrente Neuronale Netze

Idee

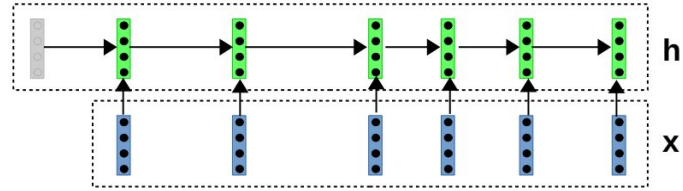
- Berechne für jede Position (jeden **time step**) im Text eine Repräsentation, die alle wesentlichen Informationen bis zu dieser Position zusammenfasst.
- Für eine Position t ist diese Repräsentation ein Vektor $\mathbf{h}^{(t)}$ (hidden representation)
- $\mathbf{h}^{(t)}$ wird rekursiv aus dem Wortvektor $\mathbf{x}^{(t)}$ und dem hidden Vektor der vorhergehenden Position berechnet:

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)})$$



Rekurrente Neuronale Netze

Idee



The sopranos was probably the last

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)})$$

- Der hidden Vektor im letzten Zeitschritt $\mathbf{h}^{(n)}$ kann dann zur Klassifikation verwendet werden (zum Beispiel zur Bewertung des Sentiment des Satzes)
- Als Vorgänger-Repräsentation des ersten Zeitschritts wird der **0-Vektor** verwendet.



Rekurrente Neuronale Netze

Rekursive Funktion f

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)})$$

- Die Funktion f nimmt zwei Vektoren als Eingabe und gibt einen Vektor aus.
- Die Funktion f ist in den meisten Fällen eine Kombination aus:
 - **Vektor-Matrix-Multiplikation**
 - Und einer **nicht-linearen Funktion** (z.B. logistic Sigmoid), die auf alle Komponenten des Ergebnisvektors angewendet wird.

$$\mathbf{h}^{(t)} = \sigma(\mathbf{W}[\mathbf{h}^{(t-1)}; \mathbf{x}^{(t)}] + \mathbf{b})$$

- Oft wird ein Bias-Vektor \mathbf{b} hinzu addiert, den wir zur Vereinfachung auch weglassen können



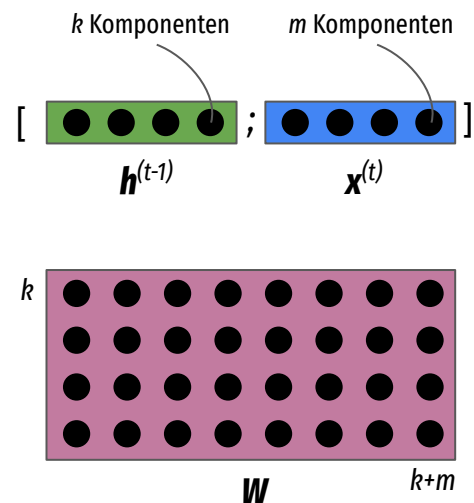
Rekurrente Neuronale Netze

Rekursive Funktion f

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)})$$

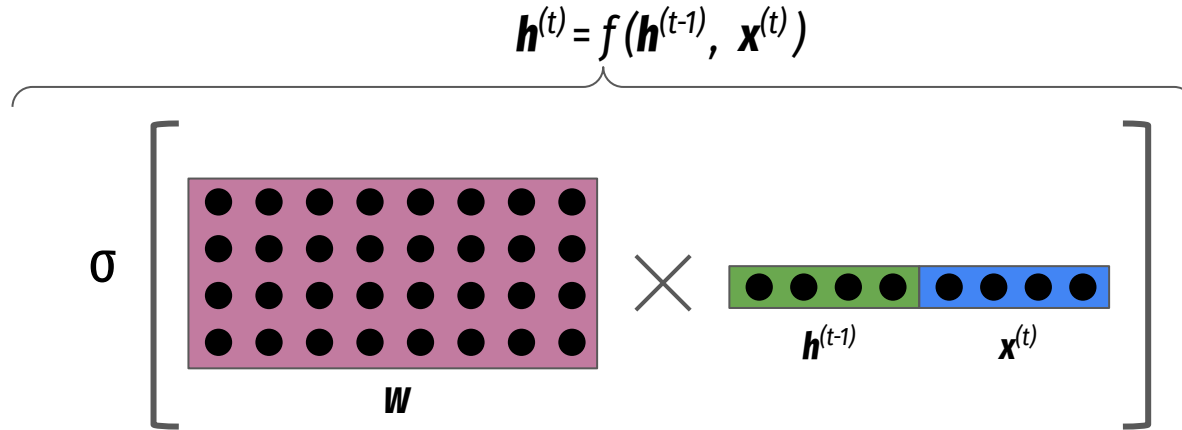
- **Vektor-Matrix-Multiplikation**

- Einfachste Form einen Vektor auf einen Vektor abzubilden
- Zunächst werden die Vektoren $\mathbf{h}^{(t-1)}$ (mit k Komponenten) und $\mathbf{x}^{(t)}$ (m Komponenten) aneinander gehängt (konkateniert):
 - Ergebnis $[\mathbf{h}^{(t-1)}; \mathbf{x}^{(t)}]$ hat Größe $k + m$ (Anzahl Vektor-Komponenten)
- Gewichtsmatrix \mathbf{W} (Größe: $k * (k + m)$)
 - dieselbe Matrix für alle Zeitschritte (*weight sharing*)
 - wird beim Trainieren des RNN optimiert



Rekurrente Neuronale Netze

Rekursive Funktion f

$$\mathbf{h}^{(t)} = f(\mathbf{h}^{(t-1)}, \mathbf{x}^{(t)})$$


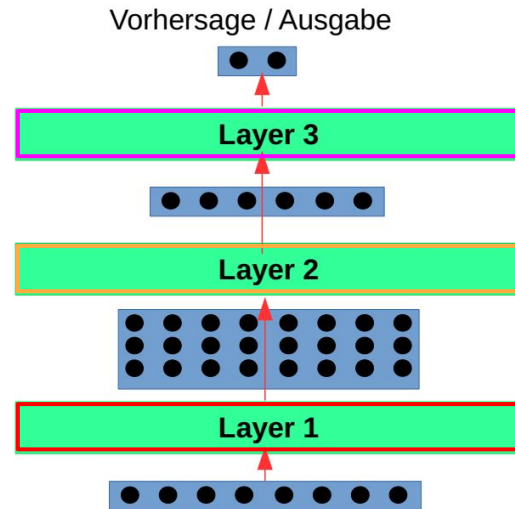
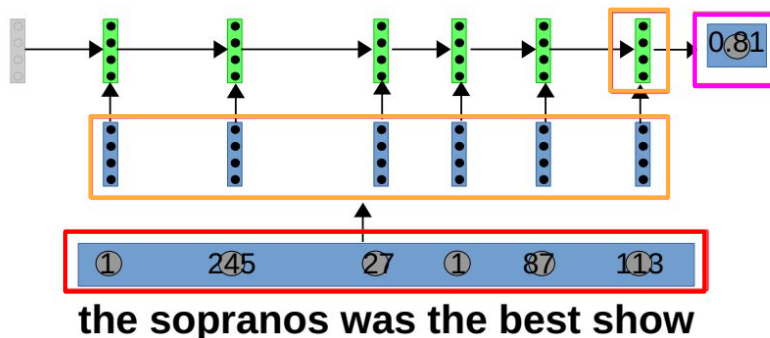
Nicht-linearen Funktion

- Beispiele: Sigmoid, Tanh (=skalierter Sigmoid, zwischen -1 . . . 1), Softmax, ReLu (=max(0, x))
- Wird auf alle Komponenten des Ergebnisvektors angewendet.
- Notwendig, damit das Netzwerk qualitativ etwas anderes als eine Lineare Kombination der Eingaben (~Durchschnitts-Vektor) berechnen kann.

Rekurrente Neuronale Netze

Beispiel: Vorhersage -> Sentiment mit einfachem RNN

- Eingabe: Vektor mit Word-ids
- Layer 1 (Embedding): Lookup von Wort-Vektoren für ids (Vektor → Matrix)
- Layer 2 (RNN): Berechnung des Satz-Vektors aus Wort-Vektoren (Matrix → Vektor)
- Layer 3: Berechnung der Wahrschlkt. für pos. Sentiment aus Satz-Vektor (Vektor → Reelle Zahl, dargestellt als Vektor mit 1 Element)



Rekurrente Neuronale Netze

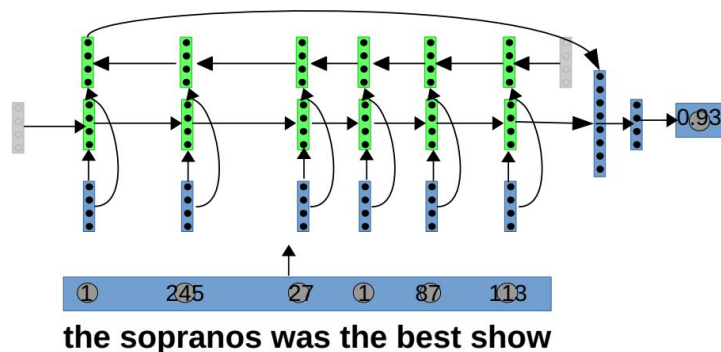
Beispiel: Vorhersage -> Sentiment mit einfachem RNN, erweitert

Vor der Vorhersage können mehrere Dense-Layers hintereinandergeschaltet werden.

- Eine Dense-Layer (auch: fully connected layer) entspricht einer Matrix-Multiplikation und Anwendung einer Nicht-Linearität

Im einfachen RNN entspricht die **rekursive** Funktion dem **Dense-Layer**

- Eine Dense-Layer “übersetzt” Vektoren und kombiniert Information aus der vorhergehenden Layer.
- Auch die Vorhersage-Layer ist meist eine Dense-Layer. (im Beispiel: Übersetzung in einen Vektor der Größe 1; Nichtlinearität ist die Sigmoid Funktion)



Rekurrente Neuronale Netze

Lernen im RNN

Wie können die Parameter für das RNN gelernt werden? Parameter sind z.B. Wort-Embeddings, Gewichts-Matrizen der Dense-Layers, ... etc. Ein Neuronales Netzwerk ist eine aus einfachen Einheiten aufgebaute Funktion, mit einem Vektor als Eingabe (z.B. Word-Ids eines Satzes), und einem Vektor als Ausgabe (z.B. WK des positiven Sentiment).

Für einen Datensatz kann nun eine Kostenfunktion berechnet werden, z.B. die negative Loglikelihood.

- ▶ (negative log-) Wahrscheinlichkeit, die das Modell den Labels der Trainingsdaten zuweist.
- ▶ Manchmal wird auch die Bezeichnung **Cross-Entropy** verwendet

Die Parameter können dann (ähnlich wie bei Word2Vec) mit **Stochastic Gradient Descent** optimiert werden.

- ▶ Anders als bei Word2Vec wird bei NN ein Parameter-Update meist für eine Mini-Batch von 10-500 Trainings-Instanzen durchgeführt.
- ▶ Es stehen mehrere Erweiterungen von SGD zur Verfügung (RMS-Prop, Adagrad, Adam, ...)

Rekurrente Neuronale Netze

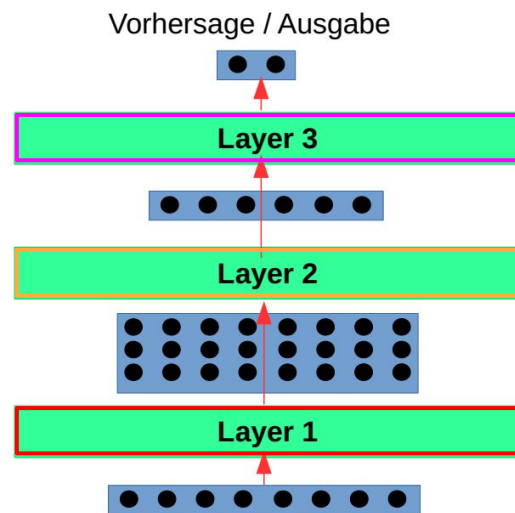
Embedding Layer

Stellt Wortvektoren der Größe `input_dim` für ein Vokabular der Größe `output_dim` bereit.

- Oft die erste Layer in einem Modell.
- Eingabe pro Instanz: Vektor mit Wort id's
- Ausgabe pro Instanz: Matrix; Sequenz von Wortvektoren.

Die Parameter (Wortvektoren) der Embedding Layer

- ... können mit vortrainierten Vektoren (Word2Vec), oder zufällig initialisiert werden.
- ... bei vortrainierten Vektoren kann oft auf ein weiteres Optimieren der Wortvektoren verzichtet werden.



Rekurrente Neuronale Netze

Embedding Layer: Vor- und Nachteile?

Vorteile / Nachteile wenn man vortrainierte Wortvektoren benutzt, und diese nicht weiter optimiert?

Vorteil: Für einen bestimmten Task, wie z.B. Sentiment-Analyse hat man oft vergleichsweise wenige Trainingsdaten. Wortvektoren kann man unüberwacht auf großen Korpora trainieren, diese haben deshalb eine bessere Abdeckung. Außerdem hat das Modell weniger Parameter zu optimieren, weshalb weniger Gefahr des *Overfitting* besteht.

Nachteil: Die verwendeten Wortvektoren passen u.U. nicht gut zum Task, die relevanten Eigenschaften wurden beim unüberwachten Lernen der Vektoren nicht berücksichtigt \Rightarrow *Underfitting*

Hinweis: Ein guter Mittelweg ist oft, die Vektoren mit vortrainierten Vektoren zu initialisieren, und diese trotzdem noch weiter Task-spezifisch zu optimieren.

Rekurrente Neuronale Netze

Masking

- Eingabe ist oft eine Matrix, mit der maximalen berücksichtigten Satzlänge als Spaltenanzahl (Zeilen: Sätze, Spalten: Wort-Id an jeweiliger Position)
- Sätze haben unterschiedliche Länge, lange Sätze müssen abgeschnitten werden, kürzere müssen aufgefüllt werden (mit einem pad-Symbol)
- Das neuronale Netz soll in allen folgenden Layers die entsprechenden Positionen ignorieren! (Das heißt, nicht in die Berechnungen einbeziehen)
- **Mask:** Tensor aus Einsen und Nullen, die für eine Eingabe die zu ignorierenden Stellen anzeigt

Short sentence	pad	pad	pad	
1	1	0	0	0
<hr/>				
This is a long sentence.				
1	1	1	1	

- Ein RNN überspringt maskierte Positionen
- Eine Loss-Funktion ignoriert maskierte Positionen