Anwendungen von Wortvektoren

Benjamin Roth

Centrum für Informations- und Sprachverarbeitung
Ludwig-Maximilian-Universität München
beroth@cis.uni-muenchen.de

Quiz

- `sli.do`

Wort-Vektoren

- *"Wortvektoren":*
  - Sparse:
    - ★ aus PPMI-gewichteter Co-okkurrenz-Matrix
    - ★ aus TF-IDF-gewichteter Term-Dokument-Matrix
  - Dense:
    - ★ durch Singular Value Decomposition der PPMI (oder TF-IDF) Matrix
    - ★ durch gradienten-basierte maschinelle Lernverfahren (Word2Vec, GloVe, ...)
- **Was sind Vorteile und Anwendungsmöglichkeiten von Wort-Vektoren?**

Vorteil: Universelle Merkmale

- Wortvektoren repräsentieren alle Wörter im selben Merkmalsraum.
- Diese Merkmale können zur Vorhersage von Wort-Eigenschaften verwendet werden, und vom Klassifikator je nach Aufgabe gewichtet werden.
- Beispiele:
  - Wortarten
  - Eigennamen-Typen (Person, Location, Organization, ...)
  - Fein-granulare Nomen-Typisierung (software, award, politician, food, ...)
  - Wort-Sentiment (*"great"* vs. *"lame"*)
  - ...

Vorteile von Dense-Repräsentationen: Generalisierung

- Dense-Repräsentationen: 50-1000 Dimensionen (SVD, Word2Vec, Glove, ...)
- Indirekte Ähnlichkeit: Weil das Modell die Ko-okkurrenz-Information komprimieren muss, werden Wörter ähnlich repräsentiert die wiederum mit *ähnlichen* (aber nicht unbedingt denselben) Wörtern Co-okkurrieren.
  $\Rightarrow$ Bessere Generalisierung
- Werden nur wenige (50-1000) Merkmale benutzt, besteht weniger Gefahr des *Overfitting* eines Klassifikators (im Vergleich zur Verwendung der PPMI-Vektoren)

Vorteil: Unsupervised (Nicht-Überwacht)

- Um Wort-Vektoren zu berechnen, benötigt man keinerlei Annotationen, es reicht eine genügend große Textmenge (z.B. Wikipedia).
- Klassifikatoren können dann mit wenigen annotierten Daten trainiert werden, unter Benutzung der zuvor gewonnenen Wort-Vektoren.

Beispiel: Wort-Sentiment

Wort	**Vektor**	**Label**
absurd	[-0.4, 0.2,0.2,. . .]	NEG
accurate	[-0.1,-1.2,0.1,. . .]	POS
proper	[0.2,-0.1,0.2,. . .]	POS
racist	[-0.5, 0.5,0.1,. . .]	NEG

...

- Einfacher Anwendungsfall:
  - Der Klassifikator kann auf einem annotierten Sentiment-Lexikon trainiert werden, und dann die Polarität für neue Wörter vorhersagen (d.h. das ursprüngliche Lexikon erweitern).
  - Das erweiterte Lexikon könnte dann zur Bestimmung des Sentiment von Texten verwendet werden (Verhältnis positiver ggü negativer Wörter).
  - Hinweis: Die Information aus den Wortvektoren kann mit Neuronalen Netzen noch effektiver verwertet werden.
- Im Beispiel: Welchen Merkmalen würde der Klassifikator **positive** Merkmalsgewichte geben, welchen **negative**, wo wäre das Gewicht **neutral** (ungefähr 0)?

Beispiel: Wort-Sentiment

Wort	**Vektor**	**Label**
absurd	[-0.4, 0.2,0.2,...]	NEG
accurate	[-0.1,-1.2,0.1,...]	POS
proper	[0.2,-0.1,0.2,...]	POS
racist	[-0.5, 0.5,0.1,...]	NEG
...		

- Im Beispiel: Welchen Merkmalen würde der Klassifikator **positive** Merkmalsgewichte geben, welchen **negative**, wo wäre das Gewicht **neutral** (ungefähr 0)?

Beispiel 2: Typ-Vorhersage

Wort/Nomen-Phrase	**Typen**
Schleswig-Holstein	location, administrative division
London Symphony Orchestra	award winner, artist, employer
Clint Eastwood	award winner, actor, producer, director, artist

...

- Gegeben eine Nomen-Phrase, sage die möglichen Typen der beschriebenen Entität voraus.
- **Anwendungsfälle?**

Beispiel 2: Typ-Vorhersage

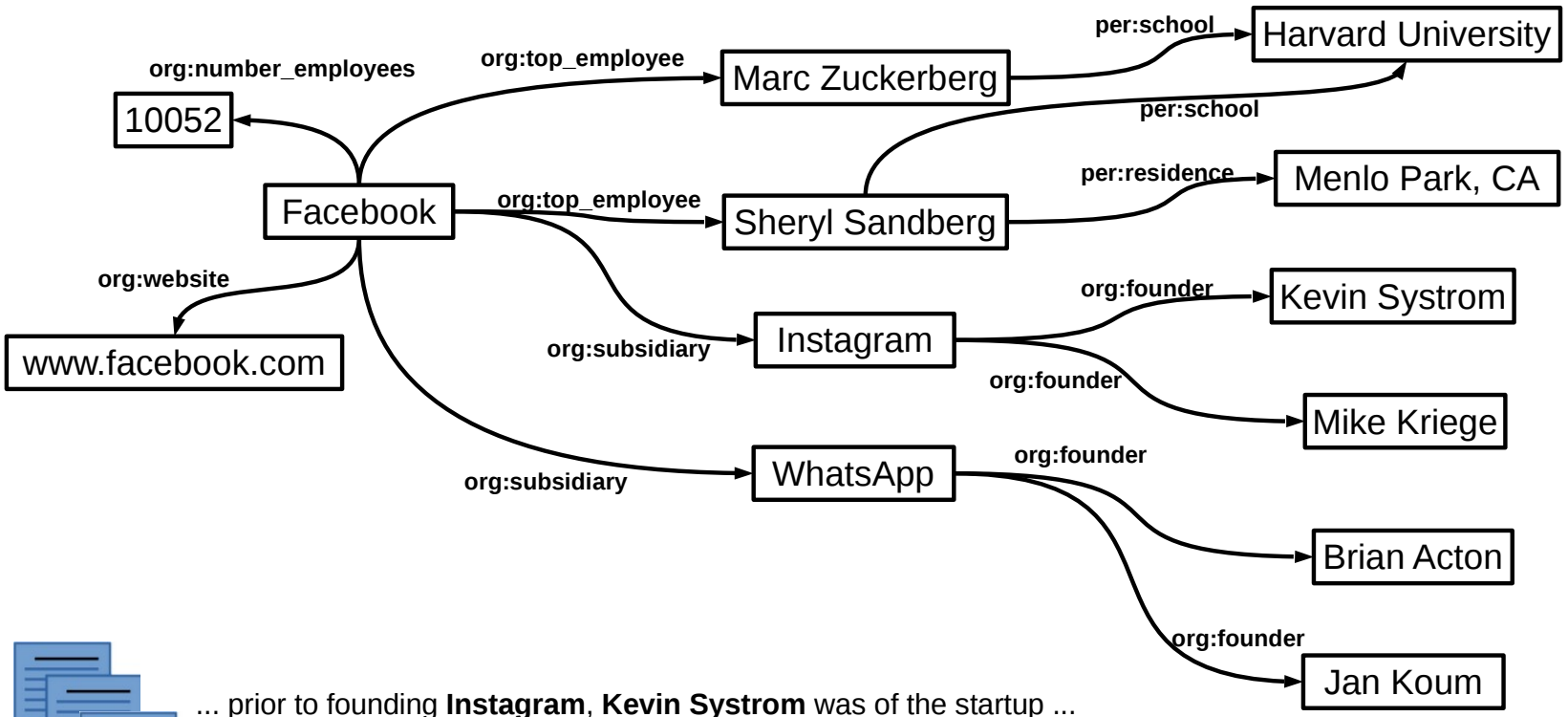
Wort/Nomen-Phrase	**Typen**
Schleswig-Holstein	location, administrative area
London Symphony Orchestra	award winner, artist, employer
Clint Eastwood	award winner, actor, producer, director, artist

...

- Gegeben eine Nomen-Phrase, sage die möglichen Typen der beschriebenen Entität voraus.
- **Anwendungsfälle?**
  - **Question Answering**: *Which administrative area does Kiel belong to? What actors starred in Gran Torino?*
  - **Knowledge Graph Construction**: Finde alle Möglichen Entitäten in einer großen Textmenge, sage in einem ersten Schritt deren Typen voraus, und in einem zweiten Schritt, welche Relationen zwischen ihnen bestehen.

Knowledge Graph Construction

1. Finde alle möglichen Entitäten in einer großen Textmenge
2. **Sage die Typen voraus**
3. Finde Relationen zwischen ihnen (in Abhängigkeit der Typen)



... prior to founding **Instagram**, **Kevin Systrom** was of the startup ...
... **Mike Krieger** co-founded **Instagram** with **Kevin Systrom** ...
... reminiscent of **Instagram**'s parent company **Facebook Inc.** ...
... the $19 billion buyout of **Whatsapp** by **Facebook** ...

Beispiel 2: Typ-Vorhersage

Wort/Nomen-Phrase	**Typen**
Schleswig-Holstein	location, administrative area
London Symphony Orchestra	award winner, artist, employer
Clint Eastwood	award winner, actor, producer, director, artist

...

- Unterschiede zu Wort-Polarität:
  - Instanz betsteht möglicherweise aus mehreren Wörtern, nicht nur aus einem.
  - Instanz kann mehrere Typen haben, es gibt nicht nur ein richtiges Label.
  - Mögliche Lösungen?

Beispiel 2: Typ-Vorhersage

- Unterschiede zu Wort-Polarität:
  - Instanz betsteht möglicherweise aus mehreren Wörtern, nicht nur aus einem.
    - ★ Möglichkeit 1: Trainiere mit Einzelwörtern, und kombiniere die Vektoren nach dem Training. (Durschschnittsvektor, Neuronales Netzwerk).
    - ★ Möglichkeit 2: Füge Entitäten-Phrasen vor dem Training zu Pseudo-Wörtern zusammen (Clint_Eastwood)[1].
      Phrasen können durch einen Tagger, oder durch Co-Okkurrenz (PPMI) gefunden werden. Vorteil: Vektor genau für diese Phrase. Nachteil: Nicht kompositionell. Ich muss Phrasen vor dem Training wissen, oder es gibt ein Abdeckungsproblem.
  - Instanz kann mehrere Typen haben, es gibt nicht nur ein richtiges Label.
    $\Rightarrow$ Lösung: Vorhersage für jeden möglichen Typ (multi-label classification). Jeder Typ wird in einem Label-Vektor an einer anderen Stelle codiert.

[1] Mikolov et al. (2013): Distributed Representations of Words and Phrases and their Compositionality

Praktische Hinweise

Praktische Hinweise

- Effiziente Implementierungen von Word2Vec, z.B.:
  `https://radimrehurek.com/gensim/models/word2vec.html`
- Vortrainierte GloVe Vektoren:
  `https://nlp.stanford.edu/projects/glove/`
- Multilabel Klassifikation mit Scikit-learn:
  - X: Trainingsdaten/Merkmale, Matrix (n_samples $\times$ n_features)
  - Y: Trainingsdaten/Labels, 0-1 Matrix (n_samples $\times$ n_classes)

```
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC

classif = OneVsRestClassifier(SVC(kernel='linear'))
classif.fit(X, Y)
```

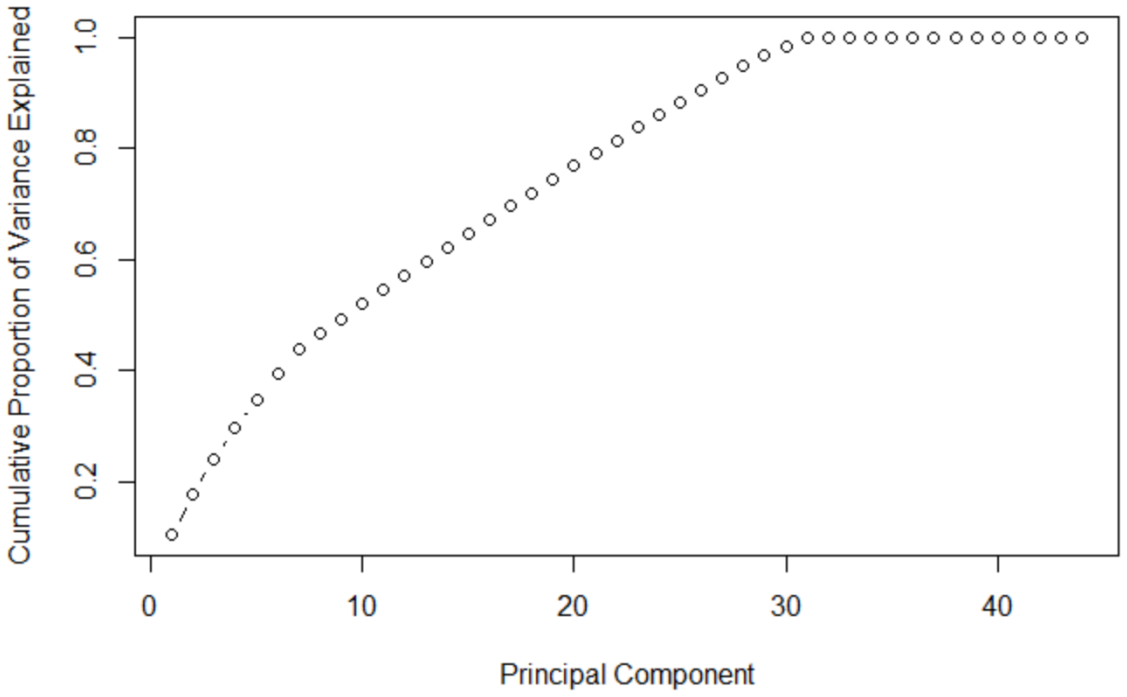
  - Statt SVC können auch andere Klassifikatoren (LogisticRegression...) gewählt werden.
  - Vorhersage ist wieder (n_samples $\times$ n_classes) 0-1 Matrix

```
classif.predict(X_test)
```

Auswahl der Anzahl der Dimensionen für einen Embedding-Space

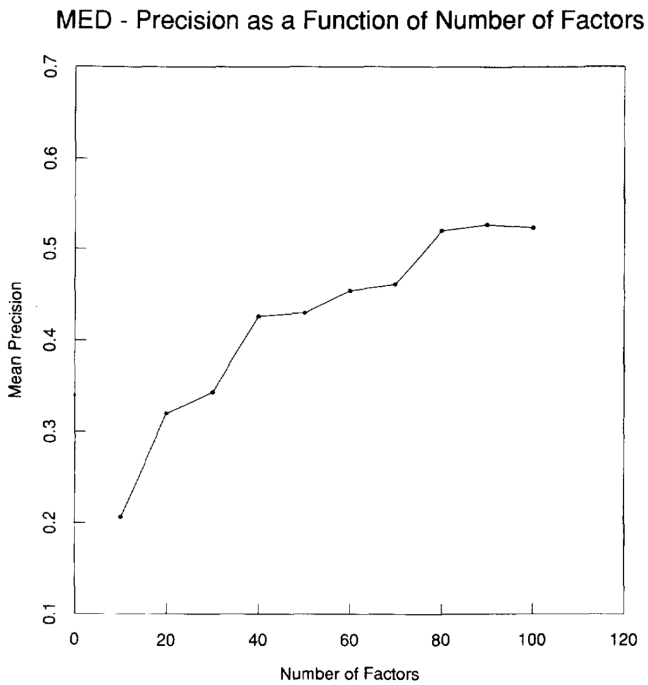
Klassische Statistik: Anteil der erklärten Varianz

- z.B. bei trunkierter SVD
- Wie nahe ist die Rekonstruktion an der originalen PPMI Matrix?
  - 0% $\Leftrightarrow$ immer Vorhersage des Mittelwertes (aller Einträge in der Matrix)
  - 100% $\Leftrightarrow$ perfekte Rekonstruktion der Matrix
- Eine Möglichkeit ist dann, zu schauen wo der zusätzliche Erklär-Nutzen abnimmt ("Knick" in der Kurve), und nur die Singulärwerte/Dimensionen bis zu diesem Punkt zu verwenden.



Auswahl in Bezug auf Task

- Wenn man annotierte Entwicklungsdaten hat, kann man auf diesen verschieden Größen des Embedding-Raums durchprobieren.
- Benötigt eine Task-spezifische Kostenfunktion.
- Wähle Anzahl mit den geringsten Kosten (mit dem größten Nutzen)
- Aus dem original LSI-Papier:



Vergleich von Verfahren für Wortvektoren

Vergleichsaspekte

- **order**: wird die Reihenfolge der Kontext-wörter im Training berücksichtigt?
- **time to train**: Ist ein effizientes Training möglich?
- $n > 1$ **lang's**: Are embeddings in multiple languages comparable?
- **syntax**: Is the syntactic information (e.g. dependency relation) between words taken into account during training?

Weitere Vergleichsaspekte

- Wir haben einige Aspekte gesehen, nach denen man Modelle für Wortvektoren unterscheiden kann.
- **compact**: Ist das Modell kompakt (dense, niedrig-dimensional) oder nicht? (z.B. SVD vs. Wordspace)
- **rare words**: Können seltene oder nicht im Korpus vorgekommene Wörter gut repräsentiert werden? (z.B. fasttext vs. word2vec)
- **units**: Was sind die Repräsentationseinheitem im Training? Wörter (w), Buchstaben (characters, c), Absätze (paragraphs, p)

Kategorisierung nach Schütze

	compact	rare words	units	order	time to train	$n > 2$ lang's	syntax
WordSpace	−	0	w	−	+	−	−
w2v skipgram	+	0	w/p	−	+	−	−
w2v CBOW	+	−	w	−	+	−	−
bengio&schwenk	+	?	w	+	−	−	−
LBL	+	?	w	+	−	−	−
CWIN	+	?	w	+	−	−	−
wang2vec	+	?	w	+	−	−	−
glove	+	?	w	−	+	+	−
fasttext	+	+	c/w/p	−	+	−	−
random	+	+	c/w/p	?	−	−	−
CCA	+	?	w	+	−	−	−
factorization	+				+	−	−
multilingual	+		w		−	+	−
dependencies	+		w			−	+

Referenzen:

- WordSpace
  - Gerard Salton. Automatic Information Organization and Retrieval. 1968. McGraw Hill.
  - Hinrich Schütze. "Dimensions of meaning". ACM/IEEE Conference on Supercomputing. 1992.
- factorization, SVD
  - Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, Richard A. Harshman. "Indexing by Latent Semantic Analysis". JASIS 41:6. 1990.
  - Omer Levy, Yoav Goldberg. "Neural Word Embedding as Implicit Matrix Factorization". Advances in Neural Information Processing Systems. 2014.
- Word2vec skipgram, CBOW
  - Tomas Mikolov, Kai Chen, Greg Corrado, Jeffrey Dean. "Efficient estimation of word representations in vector space". ICLR. 2013.
  - Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, Jeffrey Dean. "Distributed Representations of Words and Phrases and their Compositionality". NIPS. 2013.

Referenzen:

- Fasttext
  - Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. “Enriching Word Vectors with Subword Information”. TACL. 2017.
  - Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, Jeffrey Dean. “Distributed Representations of Words and Phrases and their Compositionality”. NIPS. 2013.
  - Piotr Bojanowski, Edouard Grave, Armand Joulin, Tomas Mikolov. “Enriching Word Vectors with Subword Information”. TACL. 2017.
- Glove
  - Jeffrey Pennington, Richard Socher, Christopher D. Manning. “Glove: Global Vectors for Word Representation”. EMNLP. 2014.
- CWINDOW / Structured Skip-Ngram
  - Wang Ling, Chris Dyer, Alan W. Black, Isabel Trancoso “Two/Too Simple Adaptations of Word2Vec for Syntax Problems”. NAACL/HLT. 2015.

Referenzen:

- Embeddings based on syntactic dependencies
  - Omer Levy, Yoav Goldberg. “Dependency-Based Word Embeddings”. ACL. 2014.
- Multilingual embeddings
  - Tomas Mikolov, Quoc V. Le, Ilya Sutskever. “Exploiting Similarities among Languages for Machine Translation”. CoRR. 2013.

Rekursive Neuronale Netzwerke (RNNs)

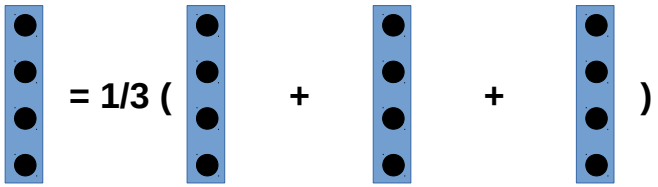
Rekursive Neuronale Netzwerke: Motivation

Wie kann man ...

- ... am besten eine Sequenz von Wörtern als Vektor repräsentieren?
- ... die gelernten Wort-Vektoren effektive kombinieren?
- ... die für eine bestimmte Aufgabe relevante Information (bestimmte Merkmale bestimmter Wörter) behalten, unwesentliches unterdrücken?

Rekursive Neuronale Netzwerke: Motivation

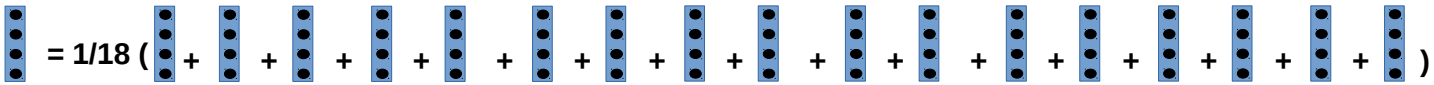
Bei kurzen Phrasen: Durchschnittsvektor evtl. Möglichkeit:



London Symphony Orchestra

$\Rightarrow$ employer?

Bei langen Phrasen problematisch.



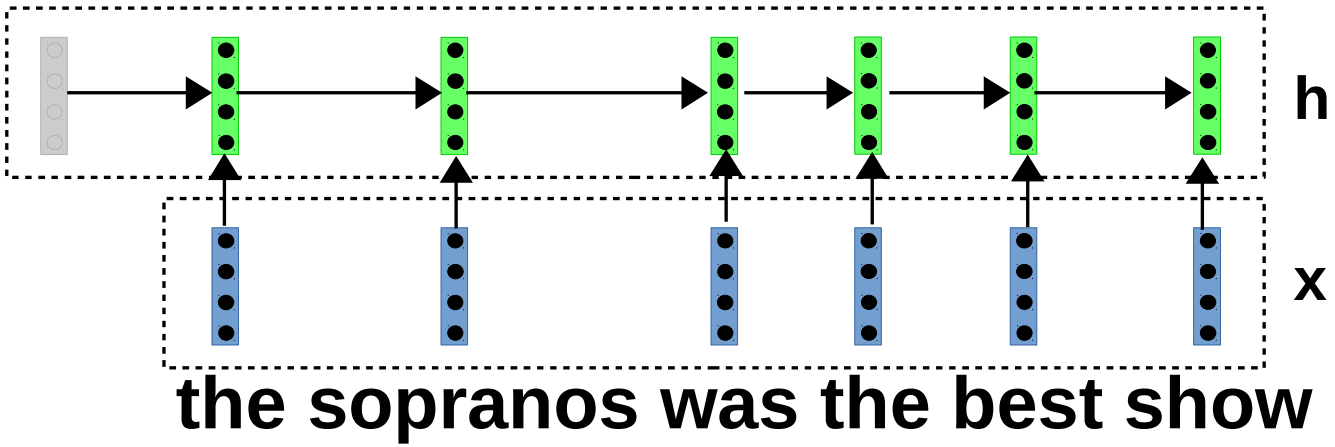
The sopranos was probably the last best show to air in the 90's. its sad that its over

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

Rekursive Neuronale Netzwerke: Idee

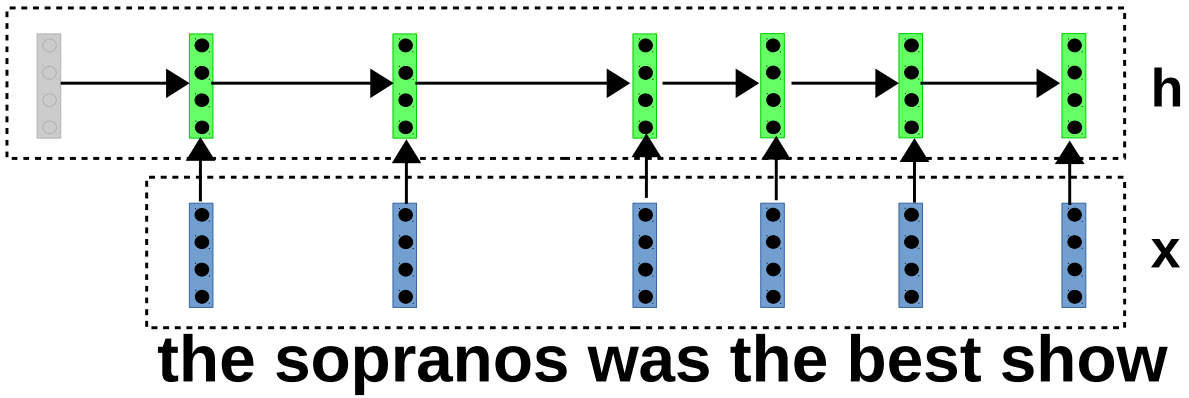
- Berechne für jede Position ("Zeitschritt", time step) im Text eine Repräsentation, die alle wesentliche Information bis zu dieser Position zusammenfasst.
- Für eine Position t ist diese Reträsentation ein Vektor $\boldsymbol{h}^{(t)}$ (hidden representation)
- $\boldsymbol{h}^{(t)}$ wird rekursiv aus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

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



Rekursive Neuronale Netzwerke

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



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

Rekursive Funktion f

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

- 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**:
    - ★ Einfachste Form einen Vektor auf einen Vektor abzubilden.
    - ★ Zunächst werden die Vektoren $\boldsymbol{h}^{(t-1)}$ (k Komponenten) und $\boldsymbol{x}^{(t)}$ (m Komponenten) aneinander gehängt (konkateniert): Ergebnis $[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}]$ hat $k + m$ Komponenten.
    - ★ Gewichtsmatrix W (Größe: $k \times (k + m)$) wird beim Trainieren des RNN optimiert.
  - und einer **nicht-linearen Funktion** (z.B. logistic Sigmoid), die auf alle Komponenten des Ergebnisvektors angewendet wird.
    - ★ Diese ist notwendig, damit durch das Netzwerk qualitativ etwas anderes als den Durchschnittsvektor berechnen kann.

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

Zusammenfassung

- Vorteile von Wortvektoren
  - Dienen als Merkmale
  - Erlauben Generalisierung
  - Können nicht-überwacht gelernt werden
- Anwendungsbeispiele
  - Typ Vorhersage
  - Klassifikation von Wort-Sentiment
- Neuronale Netzwerke
  - Rekursive Berechnung der Hidden Layer
  - Nicht-Linearität erlaubt mächtigere Darstellung als Durchschnittsvektor