Neural Networks: Architectures and Applications for NLP

Session 04

Julia Kreutzer

22. November 2016

Institut für Computerlinguistik, Heidelberg

Overview

1. Recap
2. Word Embeddings
3. Convolutional Neural Networks
4. Ausblick

Recap

NLP from Scratch I

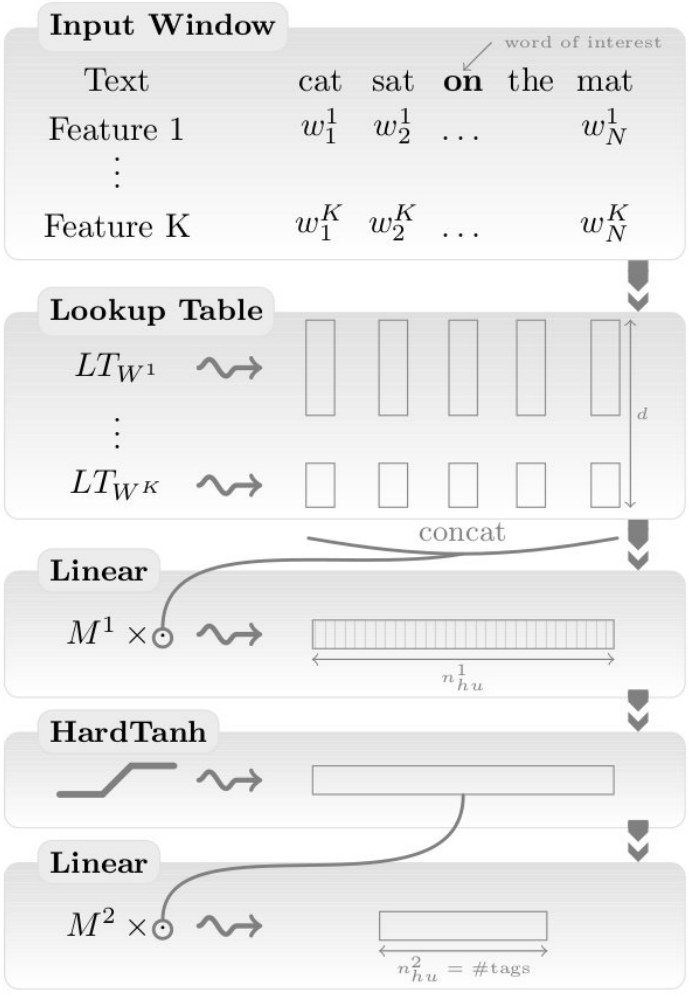


Abbildung 1: Quelle: [Collobert et al., 2011]

NLP from Scratch I

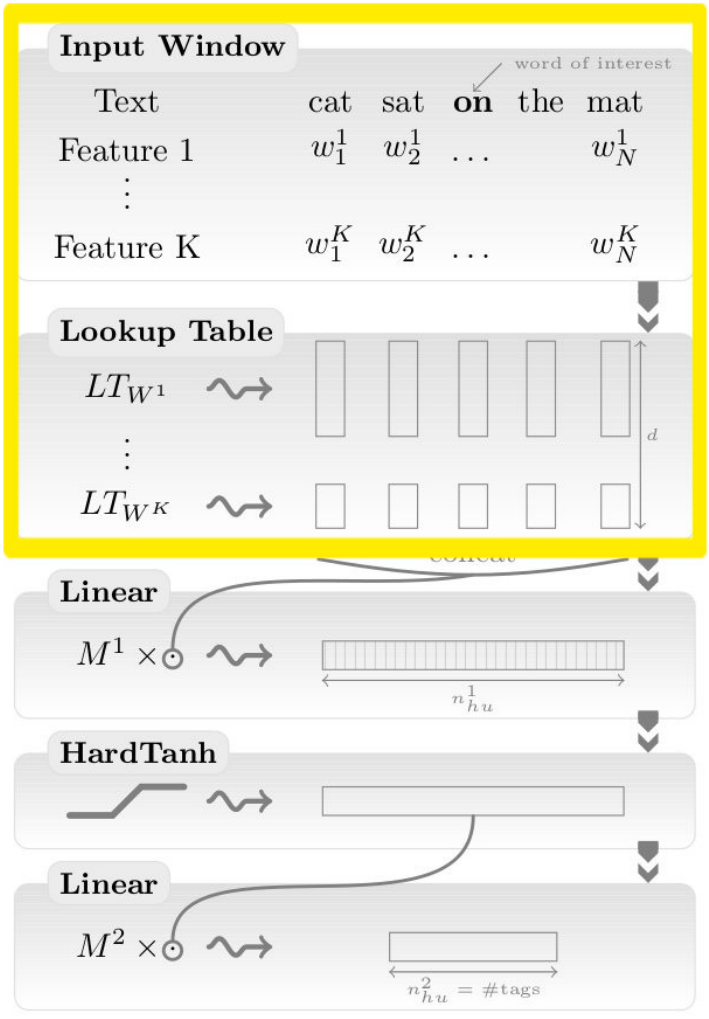


Abbildung 2: Quelle: [Collobert et al., 2011]

Word Embeddings

NLP Features

Sparse Features	Dense Features (“Embeddings”)
traditionell	mit NN
1 Feature, 1 Dimension	1 Feature, “embedded” in *d* Dimensionen
viele Features	wenige Kern-Features
viele Dimensionen	wenige Dimensionen
Feature-Repräsentation ist fix	wird gelernt
Features sind unabhängig voneinander	Ähnlichkeiten der Features werden abgebildet
häufig nützlich: Feature-Kombination	Kombinationen werden erlernt (falls nützlich)
enkodieren von linguistischem Wissen	generalisiert gut
Dimensionen sind interpretierbar	Dimensionen sind latent

Embedding Layer

Embedding = Mapping: Input zu Vektor, $C : \mathbb{N} \rightarrow \mathbb{R}^d$

Embedding-Matrix ("lookup table"): $\mathbf{E} \in \mathbb{R}^{|V| \times d}$, enthält für jeden Eintrag des Vokabulars V einen d-dimensionalen Vektor. Wird über zurückpropagierte Fehler trainiert und meist zufällig initialisiert.

Lookup-Operation: $C_{\mathbf{E}}(\mathbf{f}_i) = \mathbf{f}_i\mathbf{E}$ mit one-hot Repräsentationen des Inputs $\mathbf{f}_i$ der Länge $|V|$

Überwachtes Pre-Training

- trainiere Embeddings für Aufgabe A
- nutze für Aufgabe B
- optional: trainiere noch weiter mit Aufgabe B
- nützlich wenn nur wenige Daten für B, aber viele für A
- oder trainiere für A und B gemeinsam
- Annahme, dass sich beide Aufgaben relativ ähnlich sind

Unüberwachtes Pre-Training

- für riesige unannotierte Korpora
- ähnliche Worte sollten ähnlich repräsentiert werden
- [Harris, 1954] *Distributional Hypothesis*: Worte sind einander ähnlich, wenn sie in ähnlichen Kontexten genutzt werden
- häufig wird dafür eine Hilfs-Aufgabe definiert, die überwacht trainiert wird (z.B. sage das nächste Wort vorher)
- Generalisierung hilft bei ungesehenen Worten für die eigentliche Aufgabe

Training von Word Embeddings

Prinzip:

- extrahiere Paare von Kontexten und Zielwörtern aus einem Korpus
- definiere darauf eine Lossfunktion
- repräsentiere Wörter durch d-dimensionale Vektoren
- trainiere die Word Embeddings mit der Lossfunktion

Heute:

- NLM [Bengio et al., 2003]
- CBOW [Mikolov et al., 2013a]
- SkipGram [Mikolov et al., 2013a]

Neural Language Model [Bengio et al., 2003] i

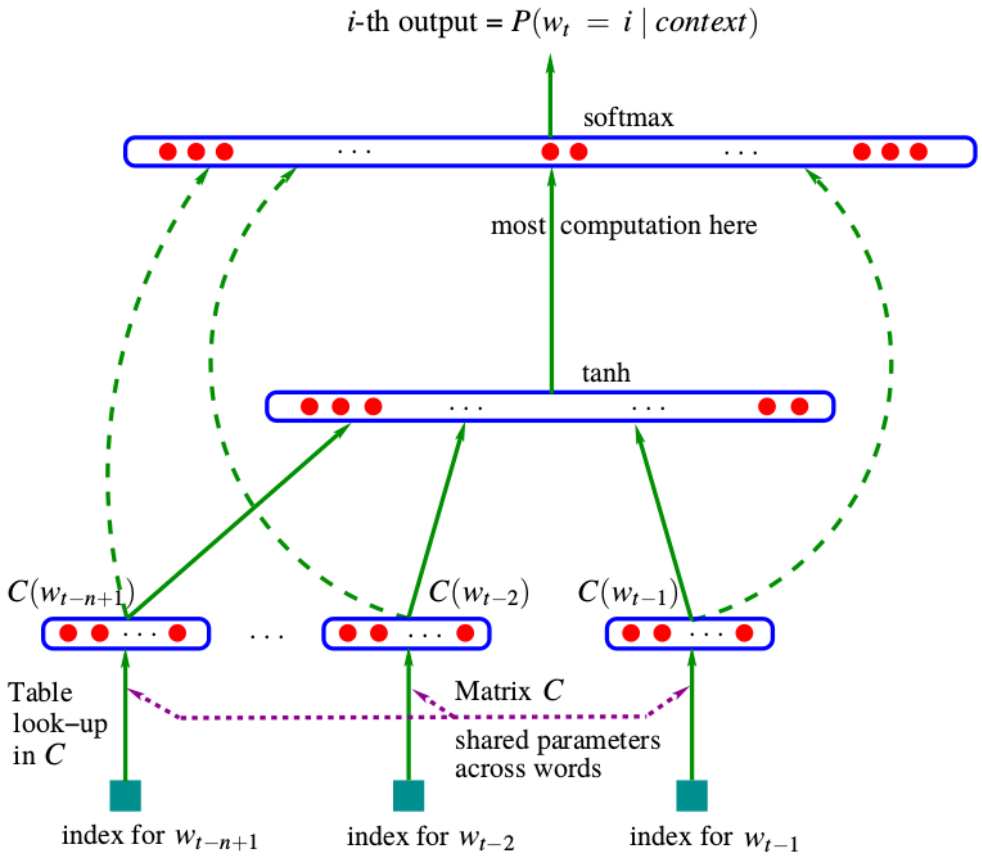


Abbildung 3: NLM Architektur, Quelle: [Bengio et al., 2003]

Neural Language Model [Bengio et al., 2003] ii

Minimiere $L = -\frac{1}{T}\sum_t \log f(w_t, w_{t-1}, \ldots, w_{t-n+1}; \theta) + R(\theta)$, mit:

- Wort-Wahrscheinlichkeiten:
  $f(w_t, w_{t-1}, \ldots, w_{t-n+1}; \theta) = \hat{P}(w_t | w_{1,\ldots,t-1}) = \frac{\exp^{y_{w_t}}}{\sum_i \exp^{y_{w_i}}}$
- Regularisierungsterm: $R(\theta)$
- Parameter: $\theta = \{\mathbf{E}, \mathbf{b}, \mathbf{W}, \mathbf{U}, \mathbf{d}, \mathbf{H}\}$
- Word-Feature-Layer: $\mathbf{x} = [C_{\mathbf{E}}(w_{t_1}), C_{\mathbf{E}}(w_{t-2}), \ldots, C_{\mathbf{E}}(w_{t-n+1})]$
- Ausgabe: $\mathbf{y} = \mathbf{b} + \mathbf{W}\mathbf{x} + \mathbf{U}\tanh(\mathbf{d} + \mathbf{H}\mathbf{x})$
- Evaluation: Perplexity auf ungesehenen Daten
  $2^{H(w1,\ldots,w_N)} = \hat{P}(w1, \ldots, w_N)^{-\frac{1}{N}} = \sqrt[N]{\prod_{i=1}^{N} \frac{1}{\hat{P}(w_i | w_{1,\ldots,i-1})}}$

word2vec-Modelle

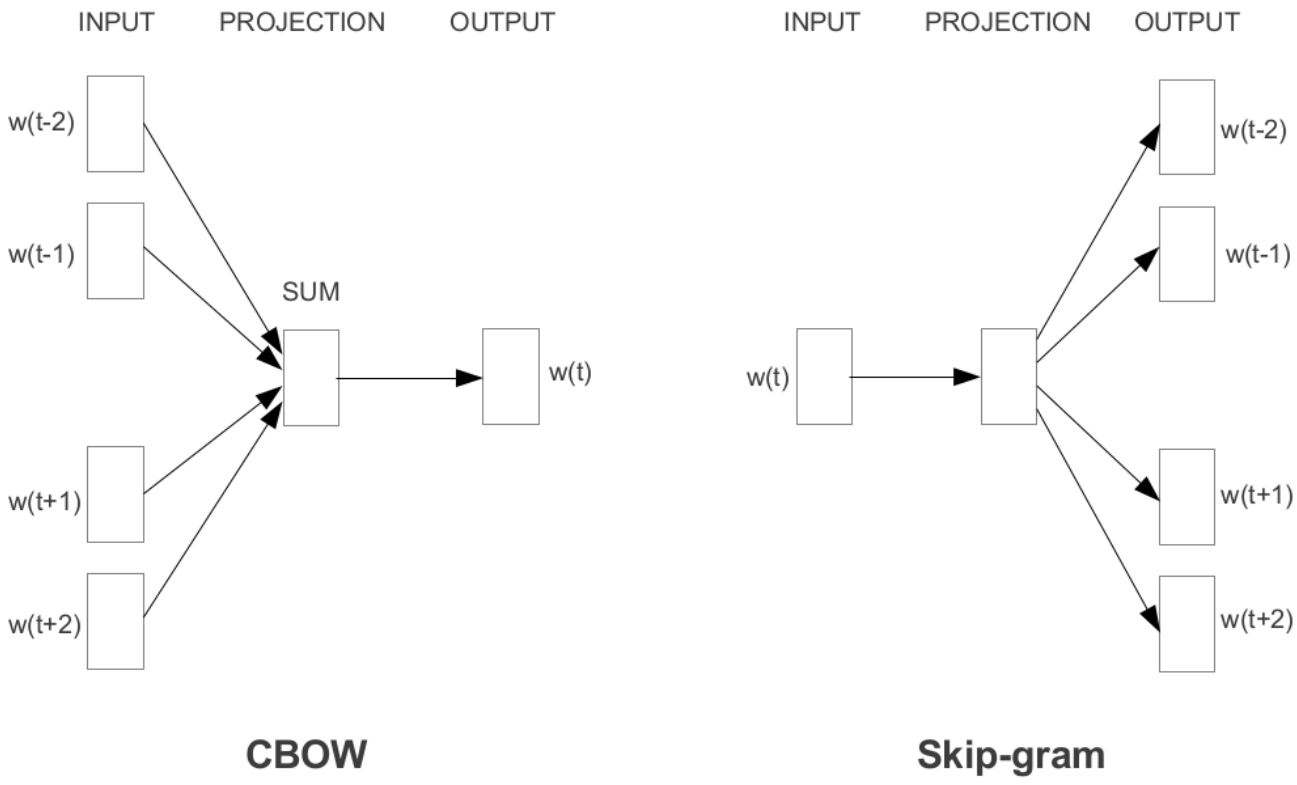


Abbildung 4: word2vec-Varianten, Quelle: [Mikolov et al., 2013a]

CBOW `word2vec` [Mikolov et al., 2013a]

Maximiere die Log-Likelihood eines Wortes w im Kontext c in Korpus D:

$$\arg\max_{\theta} \sum_{w,c \in D} \log p(w|c;\theta)$$

Kontext: “continuous bag-of-words” (CBOW)

$$\begin{aligned} p(w|c;\theta) =& p(w_t|w_{t-n},\ldots,w_{t-1},w_{t+1},\ldots,w_{t+n};\theta) \\ =& \frac{\exp^{v_c^T v_w}}{\sum_{w' \in V} \exp^{v_c^T v_{w'}}} \end{aligned}$$

Kontext und Wort werden als Vektoren v_c und v_w repräsentiert.

Continuous Bag of Words

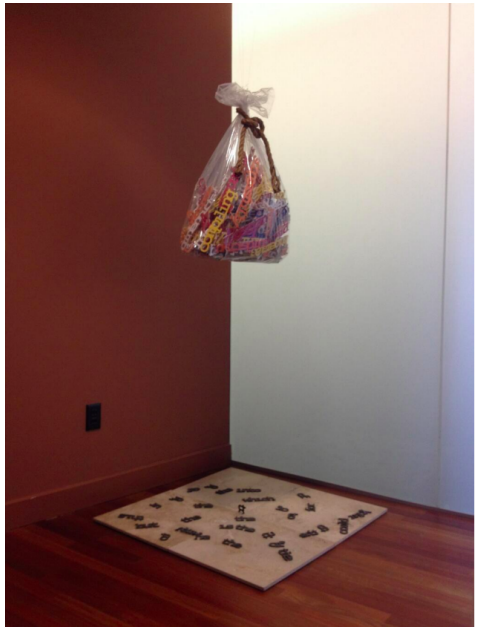


Abbildung 5: Bag of Words, Quelle: Noah Smith

Repräsentation des Kontexts als CBOW

- Wortreihenfolge spielt keine Rolle
- Summe oder Mittelwert der Wortvektoren

$$CBOW(w_1, \ldots, w_k) = \frac{1}{k} \sum_{i=1}^{k} C_{\mathbf{E}}(w_i)$$

Mit Relevanz-Gewichtung (z.B. tf-idf):

$$WCBOW(w_1, \ldots, w_k) = \frac{1}{\sum_{i=1}^{k} a_i} \sum_{i=1}^{k} a_i C_{\mathbf{E}}(w_i)$$

SkipGram `word2vec` [Mikolov et al., 2013a]

Maximiere die Log-Likelihood des Kontexts c um Wort w:[1]

$$\arg\max_{\theta} \sum_{w,c \in D} \log p(c|w;\theta)$$

Modellierung mit Wortvektoren:

$$p(c|w;\theta) = \frac{\exp^{v_c^T v_w}}{\sum_{c' \in C} \exp^{v_{c'}^T v_w}}$$

Problematisch: Summe über alle Kontexte

[1] Notation nach [Goldberg and Levy, 2014]

SkipGram & Negative Sampling

Stattdessen *Noise Contrastive Estimation* mit negativen Samples aus einem Set D':

$$\arg\max_{\theta} \prod_{w,c \in D} p(D = 1|c, w; \theta) \prod_{w',c' \in D'} p(D = 0|c', w'; \theta)$$

Wie wird D' konstruiert?
z.B. $w = w'$, Kontextwörter c' werden zufällig aus dem Trainingskorpus gewählt (uniform samples).

SkipGram & Negative Sampling

Stattdessen *Noise Contrastive Estimation* mit negativen Samples aus einem Set D':

$$\arg\max_{\theta} \prod_{w,c \in D} p(D = 1|c, w; \theta) \prod_{w',c' \in D'} p(D = 0|c', w'; \theta)$$

Wie wird D' konstruiert?
z.B. $w = w'$, Kontextwörter c' werden zufällig aus dem Trainingskorpus gewählt (uniform samples).

Ist `word2vec` "deep"?

Varianten

Zahlreiche Variationen möglich:

- Zielfunktion
- Architektur des Modells
- Definition des Kontexts
  - Wort-Fenster
  - Syntaktisch: Parse-Baum
  - Multilingual: Alignments
  - Multimodal

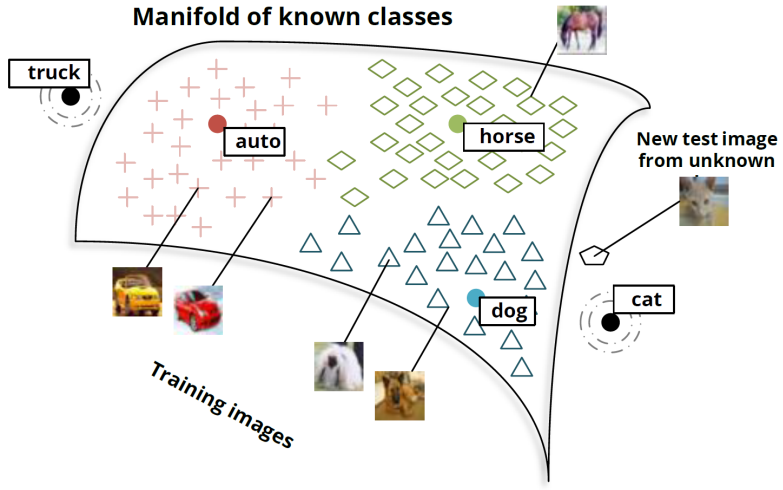


Abbildung 6: Multimodale Repräsentationen, Quelle: [Socher et al., 2013]

Sub-Word Level

Repräsentiere einzelne Buchstaben oder Wortteile als Vektoren

- Embedding eines Wortes als Komposition
- keine ungesehenen Worte (UNKs)
- nützlich für syntaktische Aufgaben
- nur kleines “Vokabular”, z.B. Alphabet
- aber: arbiträre Beziehung zwischen Funktion und Form
- als “back-off”: für UNKs oder seltene Wörter

Evaluation

Wie kann man Word Embeddings vergleichen?

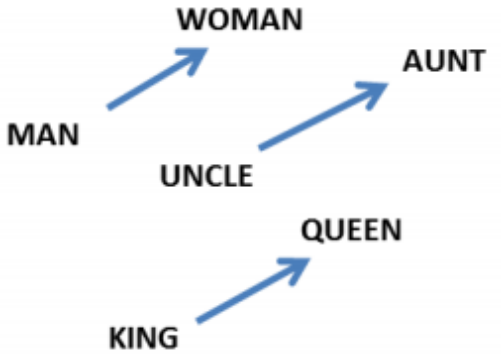
- Vektoren nicht direkt vergleichbar falls nicht gemeinsam trainiert
- i.d.R. keine Interpretation der Dimensionen

Hilfs-Evaluation:

- Clustering mit Distanzmaß
- Word Similarity & Word Relatedness Tasks
- als Features für andere NLP-Task
- Language Model Perplexity

Wortbeziehungen

Differenz zweier Vektoren entspricht der Beziehung deren Wörter [Mikolov et al., 2013b]



- Analogien: woman-man = aunt-uncle
- Komparative: big-bigger = small-larger
- Orte: Miami-Florida = Baltimore-Maryland
- Kulturen: Japan-sushi = Germany-bratwurst

Bias

Bias in Embeddings:

z.B. man-woman = computer programmer - homemaker

Gender stereotype *she-he* analogies.

sewing-carpentry	register-nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas	hairdresser-barber

Gender appropriate *she-he* analogies.

queen-king	sister-brother	mother-father
waitress-waiter	ovarian cancer-prostate cancer	convent-monastery

Abbildung 7: Quelle: [Bolukbasi et al., 2016]

Visualisierung i

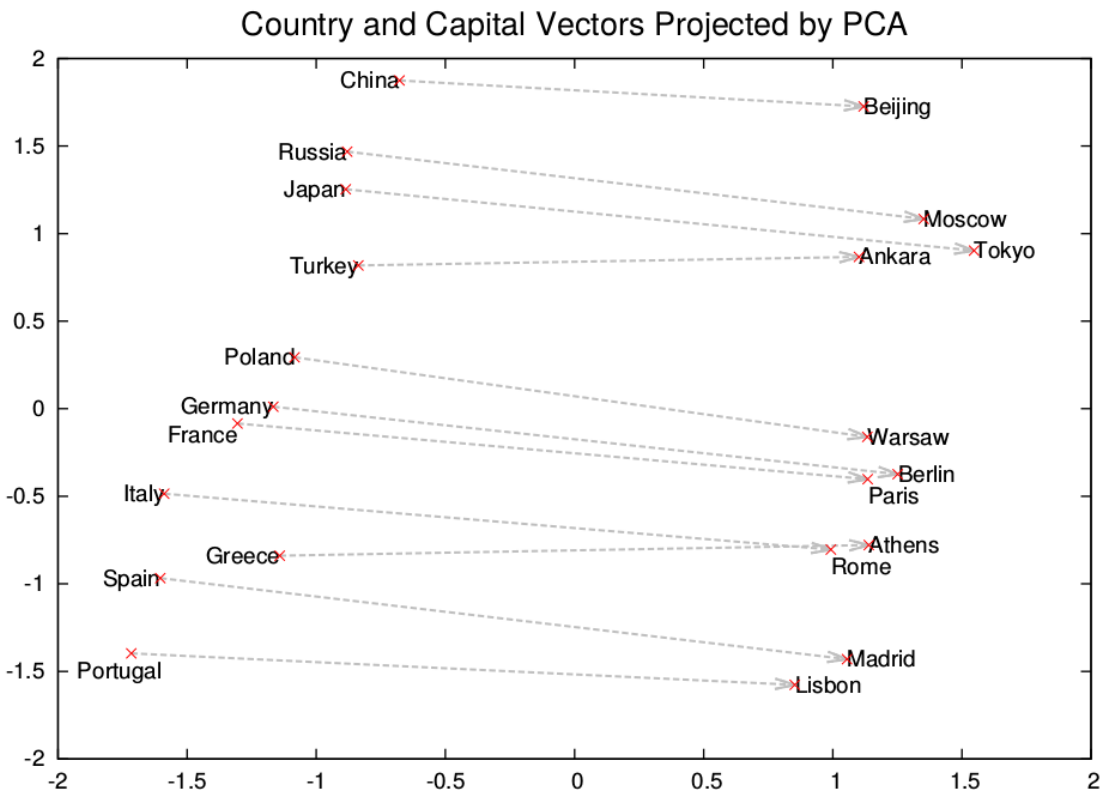


Abbildung 8: 2D PCA-Projektion von 1000-dimensionalen Wordvektoren, Quelle: [Mikolov et al., 2013a]

Visualisierung ii

Visualisierung in 2D

- Vektoren auswählen
- Dimensionen auf 2 reduzieren
  - t-SNE
  - PCA
- Plotten

Vorsicht mit der Interpretation. **Warum?**

Embeddings und NLP

Word Embeddings als **“secret sauce”** in NLP [Luong et al., 2013]

Aber: welches Rezept? Und: wie evaluieren?

ACL ’16 Workshops:

- “The First Workshop on Evaluating Vector Space Representations for NLP”
- “1st Workshop on Representation Learning for NLP”

Convolutional Neural Networks

CNN in der Bilderkennung

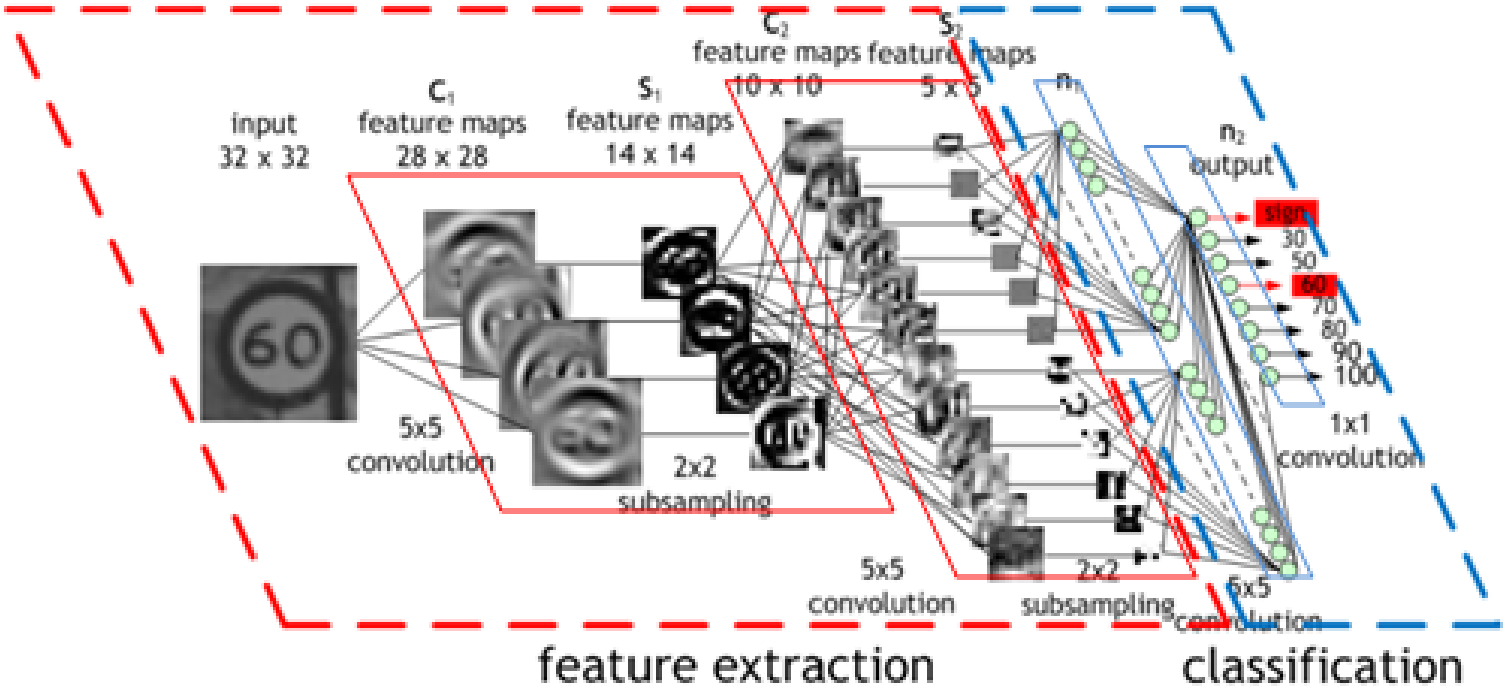
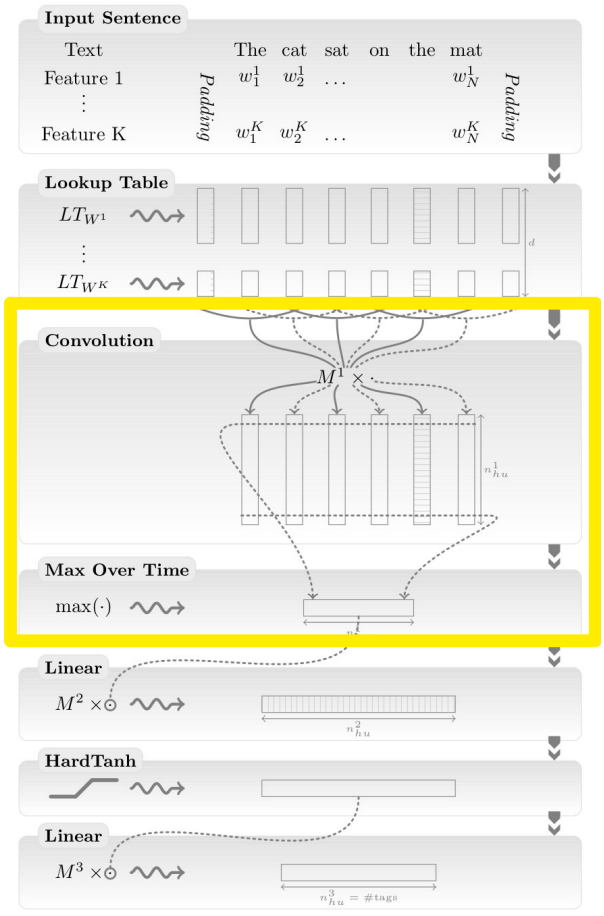


Abbildung 9: Quelle: NVIDIA Dev Blog

CNN für NLP: NLP from Scratch II



Semantic Role Labeling:
Klassifikation in Bezug auf Verb

- ganzer Satz als Input
- gegeben: Position der Verben
- zusätzliche Features: Distanz zum Verb, Distanz zum Fokus-Wort
- Klassifikation des ganzen Satzes für jedes Verb

Abbildung 10: Quelle: [Collobert et al., 2011]

Convolution & Pooling für NLP

Convolution erfasst informative lokale Features, Pooling kombiniert sie zu globalen Repräsentationen

Wichtig für NLP:

- **Unbegrenzter Kontext**: das Modell findet selbst die informativen Features
- **Wortreihenfolge** bleibt im Gegensatz zu CBOW erhalten

Convolution & Pooling: Idee

Convolution:

- wende eine non-lineare Funktion (*"filter"* oder *"kernel"*) auf jedes Fenster von k Worten an
- bewege das Fenster dabei über die gesamte Input-Sequenz (kann mehrere *"channels"* enthalten)
- der Filter transformiert jedes Fenster zu einem d-dimensionalen Vektor

Pooling:

- Pooling-Operation kombiniert die Vektoren der Fenster zu einem einzelnen d-dimensionalen Vektor
- Max- oder Average Pooling

Convolution & Pooling: Formal[2]

- Input: Sequenz von Wörtern: $\mathbf{x} = x_1, \ldots, x_n$, jedes mit d_{emb}-dimensionalem Embedding $C_{\mathbf{E}}(x_i)$
- Fenster mit Weite k: $\mathbf{w_i} = [C_{\mathbf{E}}(x_i), C_{\mathbf{E}}(x_{i+1}), \ldots, C_{\mathbf{E}}(x_{i+k-1})]$, $\mathbf{w_i} \in \mathbb{R}^{kd_{emb}}$
- Padding an Rändern des Inputs (wide vs. narrow convolution, optional)
- Filter $\mathbf{W} \in \mathbb{R}^{kd_{emb} \times d_{conv}}$ extrahiert m Vektoren $\mathbf{p_1}, \ldots, \mathbf{p_m}$,, $\mathbf{p_i} = \sigma(\mathbf{w_i W} + \mathbf{b})$, $\mathbf{p_i} \in \mathbb{R}^{d_{conv}}$
- Max-Pooling: extrahiert einen einzelnen d_{conv}-dimensionalen Vektor $\mathbf{c}$ mit den Einträgen $c_j = \max_{1<i\leq m} \mathbf{p_i}[j]$
- $\mathbf{c}$ ist das Input zu weiteren Verabeitungsschichten des NN

[2] Für NLP i.d.R. nur 1D-Convolution, für Bildverarbeitung ist 2D Standard.

Convolution & Pooling: Beispiel

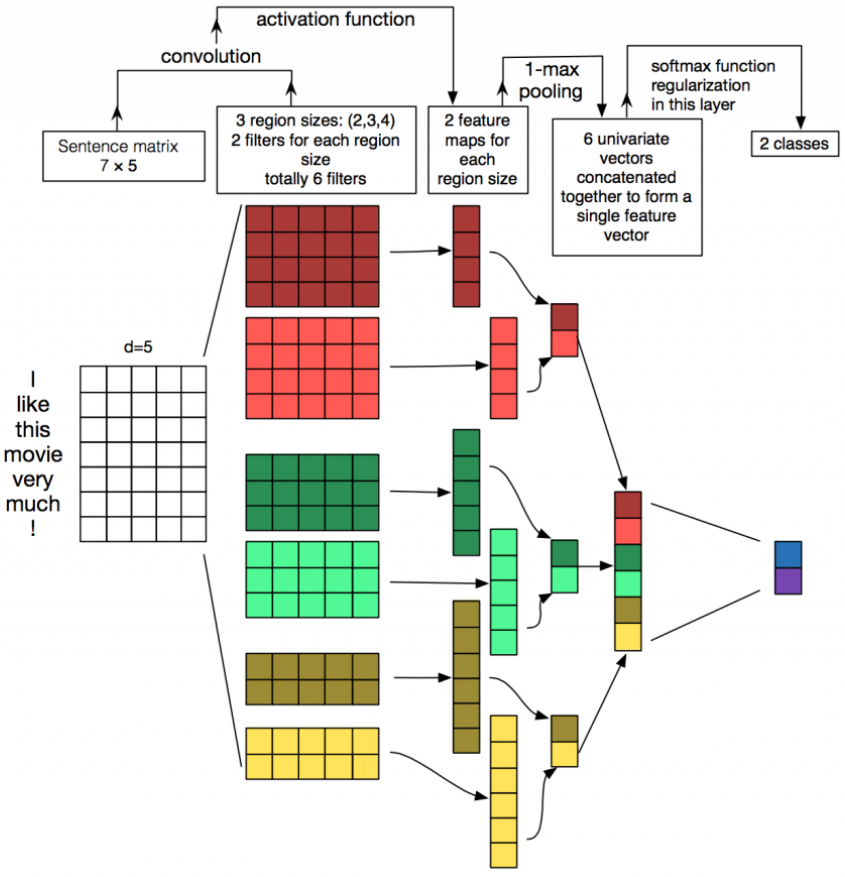


Abbildung 11: Quelle: [Zhang and Wallace, 2015][3]

[3] Achtung: hier sind die Filter als die Summe von elementweisen Produkten definiert, daher extrahieren sie Skalare aus der Satzmatrix **S**: $p_i = \sum_{j,k} S_{j,k} \cdot W_{j,k}$

Variationen

- mehrere Regionen für Pooling: $\mathbf{p}_i$ unterteilen, z.B. für Dokumentklassifikation Regionen für unterschiedliche Textabschnitte
- k-max Pooling: alle top k Werte in jeder Dimension werden erhalten
- Input muss keine lineare Sequenz sein, kann z.B. auch Syntax-Baum sein

Metaparameter

- Größe der Filter
- Anzahl der Filter
- Aktivierungsfunktionen der Filter
- Padding-Technik
- Pooling-Strategie
- Anzahl von Schichten
- Schrittgröße (“stride”)
- Kanäle: Sichten auf das Input, z.B. mehrere unterschiedliche Word Embeddings

Ausblick

Heute

Jetzt können wir

- Wörter in Vektoren verwandeln
- Sequenzen von beliebiger Länge repräsentieren
- abstrakte Repräsentationen dieser Sequenzen finden

Aber: wie können wir auch lange Abhängigkeiten repräsentieren (z.B. Verb und Partizip im Deutschen)?
Was ist, wenn die Reihenfolge eine wichtige Rolle spielt?

Nächstes Mal

- Recurrent Neural Networks (RNN)
- Long Short Term Memory (LSTM)
- Gated Recurrent Units (GRU)

Machine Learning Blogs

- Chris Olah (Google Brain)
- Andrej Karpathy (OpenAI)
- Sebastian Ruder (AYLIEN)
- Denny Britz (Google Brain): WILDML
- Shakir Mohamed (DeepMind): The Spectator

References i


Bengio, Y., Ducharme, R., Vincent, P., and Janvin, C. (2003).
A neural probabilistic language model.
The Journal of Machine Learning Research, 3:1137–1155.


Bolukbasi, T., Chang, K.-W., Zou, J., Saligrama, V., and Kalai, A. (2016).
Man is to computer programmer as woman is to homemaker? debiasing word embeddings.
arXiv preprint arXiv:1607.06520.


Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., and Kuksa, P. (2011).
Natural language processing (almost) from scratch.
The Journal of Machine Learning Research, 12:2493–2537.

References ii


Goldberg, Y. and Levy, O. (2014).
word2vec explained: deriving mikolov et al.'s negative-sampling word-embedding method.
arXiv preprint arXiv:1402.3722.


Harris, Z. S. (1954).
Distributional structure.
Word, 10(2-3):146–162.


Luong, T., Socher, R., and Manning, C. D. (2013).
Better word representations with recursive neural networks for morphology.
In *CoNLL*, pages 104–113.

References iii


Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. (2013a).
Distributed representations of words and phrases and their compositionality.
In *Advances in neural information processing systems*, pages 3111–3119.


Mikolov, T., Yih, W.-t., and Zweig, G. (2013b).
Linguistic regularities in continuous space word representations.
In *HLT-NAACL*, volume 13, pages 746–751.


Socher, R., Ganjoo, M., Manning, C. D., and Ng, A. (2013).
Zero-shot learning through cross-modal transfer.
In *Advances in neural information processing systems*, pages 935–943.

References iv


Zhang, Y. and Wallace, B. (2015).
A sensitivity analysis of (and practitioners' guide to) convolutional neural networks for sentence classification.
arXiv preprint arXiv:1510.03820.