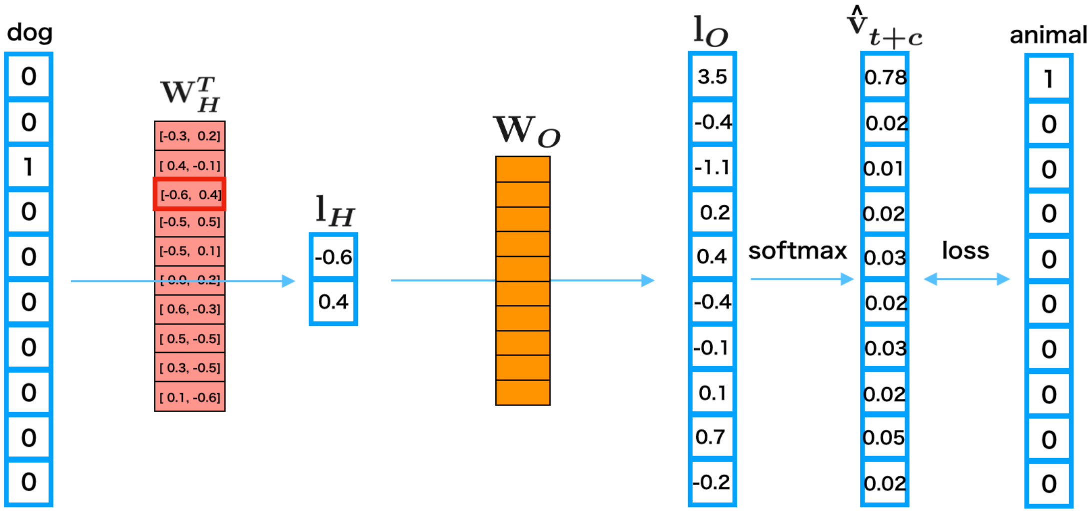
word2vec
Vectorizing Language – for Information Extraction

UNIVERSITÄT BERN



Program

b
UNIVERSITÄT
BERN

- Word to Vector (ideas and approaches)
- From w2v to character language models
- Architecture of Models
- Frameworks: SpaCy & flairNLP

Follow-up

- Hands-on Sessions: w2v

Goals

UNIVERSITÄT
BERN

- Understand why approach is interesting/important
- Be aware of context of word embeddings
- try out for one specific task
- understand (very roughly) how text generators how GPT 3.5 (one element of ChatGPT) works

Why vectorizing text? Motivation

New possibilities:

- Calculations
- Similarities
- Clustering
- …

But: Needs to be contextualized and validated (prerogative of domain experts)

Vectorizing & Distant Reading

Vectors one (of many) possibility to analyze text corpora.

Relevant for:

- Literary sciences (plot development, sentiment)
- History (text re-use, topic/keyword appearances)
- Linguistics (language developments: vanishing of genitives)

- But also for commercial players:
  Search Engines, Retailer, Social Networks..

Task

UNIVERSITÄT BERN

allow for vector representation (of text/ text parts): «words» (token)

with necessary simplification (feasibility!)

without getting rid of important traits of

languages, culture, and society

(plus *in best case* remains explainable)

((((yoav' ()ل)()ل)))
@yoavgo

Start-of-semester thoughts:
Teaching NLP is quite depressing, and I don't know how to do it well. I am torn between the two perspectives:

1) Teach the interesting problems. Why language is interesting. Why language is hard. How is language structured. What should we look at.

4:59 PM · Oct 20, 2020 · Twitter Web App

108 Retweets **23** Quote Tweets **630** Likes

((((yoav' ()ل)()ل))) @yoavgo · Oct 20
Replying to @yoavgo
The problem here is that this will be mostly presenting a set of open questions, without good solutions to any of them. We suck at everything. And things we do quite well on, like say tagging and parsing, we don't really know what to do with these structures once we have them.

3 5 113

((((yoav' ()ل)()ل))) @yoavgo · Oct 20
And for things which we do sort-of-ok on, like coreference, we don't really have a good solution, and also no real handle on how to improve things further. This feels stuck.
And for things like pragmatics, dialog... we really don't have anything. Not even proper training data.

2 3 68

((((yoav' ()ل)()ل)))
@yoavgo

Replying to @yoavgo

On the other hand, we can take the deep learning perspective, and:
2) teach embeddings, BERT, fine-tuning, squad, etc.
This results in a set of techniques that work, but:
(a) we don't really know why
(b) they also fail a lot.
(c) and again we don't know how to improve.

4:59 PM · Oct 20, 2020 · Twitter Web App

Basics

Word Embedding:
Words are each asigned to a vector $v \in \mathbb{R}^n$.

Simplest form of word embeddings:
Indexing Vocabulary (0: Hello, 1: World, …)

→ Representation of the word via the index; context and similarities ('horse', 'horses') cannot be taken into account here

Basics

Some definitions/examples:

Token: a treated string (of characters), usually referred to as a «word»

Lemma: a normalized form of a token

Part-of-Speech: Verb/Noun…

Named Entity: Place name, person, organization, date (definition rather sketchy)

Based on Mikolov et al. 2013

UNIVERSITÄT BERN

2013: ***Efficient* Estimation of Word Representations in Vector Space**
(https://arxiv.org/pdf/1301.3781)

- To that point NLP systems and techniques considered **words as single entities**, without considering their similarities
- **Neural networks** enabled significantly better results compared to simple models, among others consideration of similarity
- Challenge: **High complexity of existing models**, correspondingly limited capacities → 2013 no more than a few hundred million words with a word vector dimensionality 50 to 100

Basic idea of word2vec

UNIVERSITÄT BERN

Firth (1957): You shall know a word by the company it keeps

- Similarity of word meanings can be learned with simple information: **Assumption that the meaning of words is influenced by the surrounding words**.

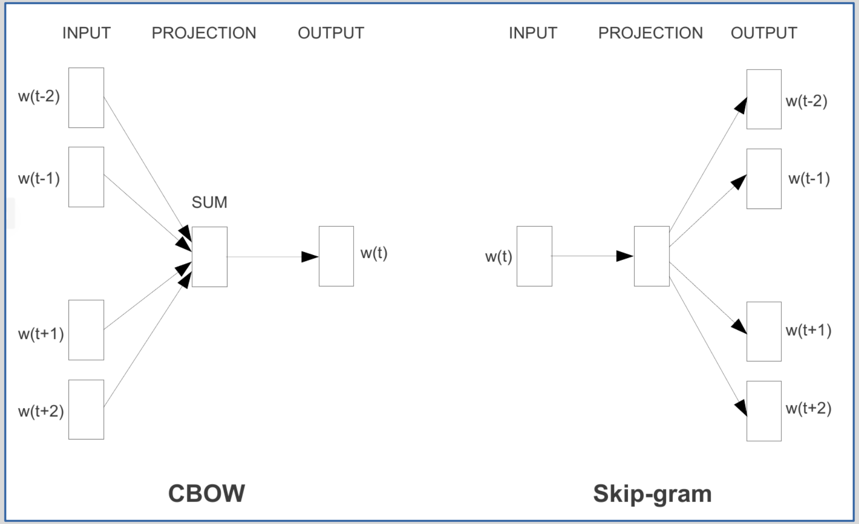
- N-grams in sentences form the context of words

c=0	The cute cat jumps over the lazy dog.
c=1	The cute cat jumps over the lazy dog.
c=2	The cute cat jumps over the lazy dog.

→ Training a neural network whose hidden layer (N-dimensional vector) represents a word embedding.

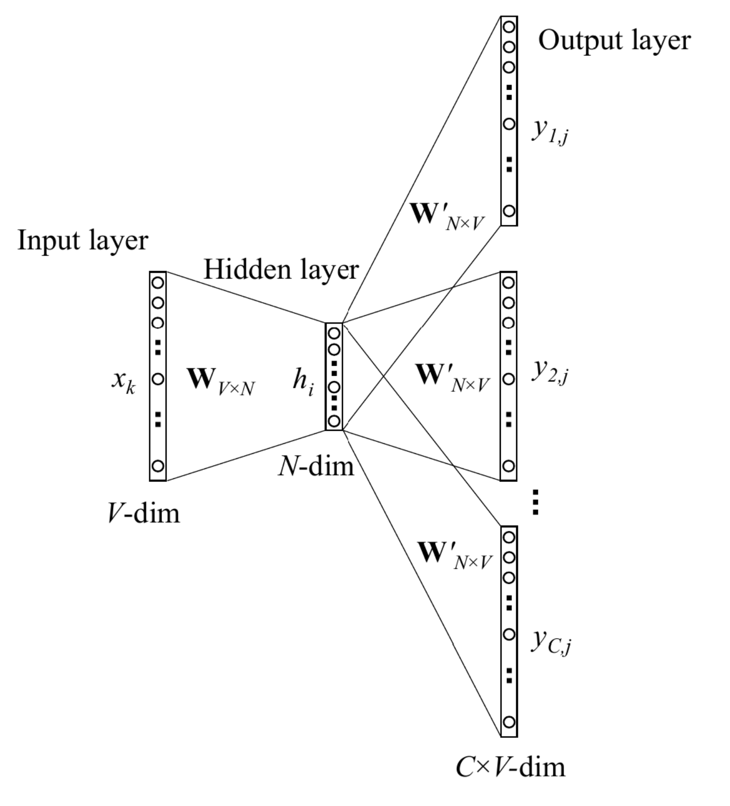
Algorithms for word2vec

- **Continuous Bag-of-Words (CBOW)**
  We learn a central word based on the context (surrounding words)
- **Continuous Skip-gram**
  We learn surrounding words with a central (given) word
- Consideration of context words is enough (instead of the whole document) and optimizes the required training time
- We focus on skip-gram: Used more often due to better accuracy



Skip-Gram

- **V = Lenght of vocabulary**
- **N = Dimensions of word embeddings (e.g. 1000)**
- **Input x_k:** One-Hot-Vektor of the k^{th} word
- **W_{VxN}:** Weight-Matrix I→H
  Line w_j = Embedding of the j^{th} word
- **h_i:** N-dimensional embedding (=w_i)
- **W'_{NxV}:** Weight-Matrix H→ O
- **Output-Layer** CxV-dimensionale Matrix
- **C** = Anzahl Kontextwörter

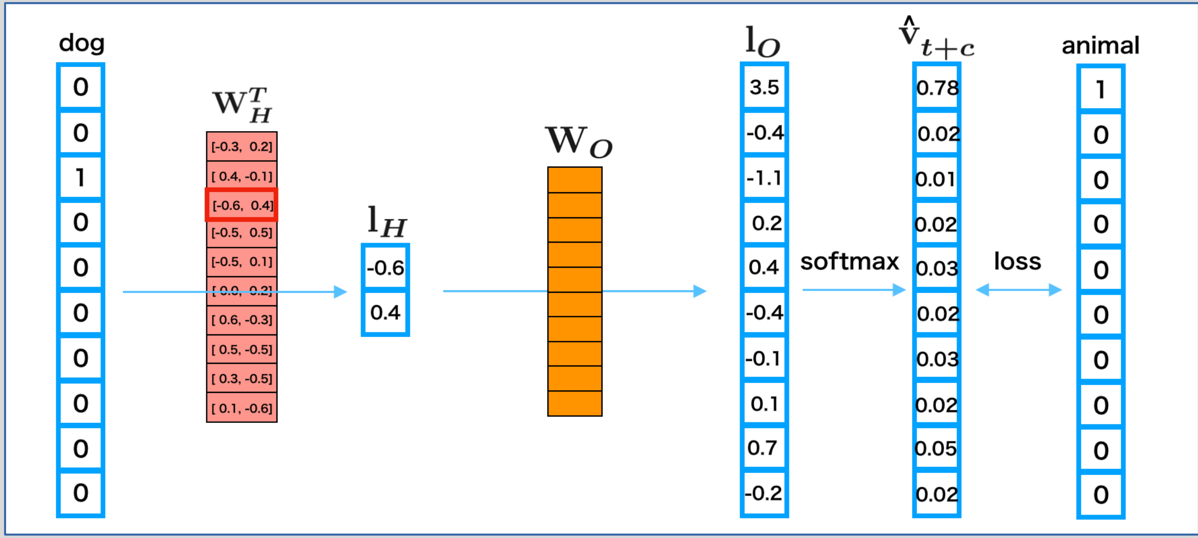


Skip-Gram

- **Example**
  Sentence: «~~This~~ animal ~~is a~~ little dog sitting ~~in~~ front ~~of the~~ huge house ~~and it~~ unfortunately fears most cats.»

- *animal, little, **dog**, sitting, front*
  2-dimensional embedding
  Hidden Layer with no activation

- **Calculate the probability of the output per softmax function**
  C mal (here 4), each with back propagation



Reduce complexity

UNIVERSITÄT BERN

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-c\leq j\leq c, j\neq 0}\log p(w_{t+j}|w_t)$$

Goal: Optimize output probabilities

Probability that at input w_I the output is w_O evaluated by softmax :

$$p(w_O|w_I) = \frac{\exp\left({v'_{w_O}}^\top v_{w_I}\right)}{\sum_{w=1}^{W}\exp\left({v'_w}^\top v_{w_I}\right)}$$

Result: Vector with sum of vector elements = 1.0

Problem: Vector of dimension V, computational cost is O(V).
Even if the vocabulary contains "only" 500,000 words, the training takes too long.

The most commonly used solution according to Mikolov et al. (cf. https://arxiv.org/pdf/1310.4546.pdf) is **hierarchical Softmax**: Calculation of probability with Huffman Tree (binary) → $\mathbf{O(\log_2(V))}$

Reduce complexity

Goal: Reduce training time

Convert frequency word pairs or phrases into a «word» («San Francisco»)

- **Subsampling of high-frequency words** (the, a, ...)
- **«Negative sampling»: training samples adjust only a few percent of the weights**

Advantages & disadvantages of word2vec

UNIVERSITÄT BERN

Arithmetic operations can be performed

v('king') – v('man') + v('woman') = v('queen')
v('france') – v('paris') + v('berlin') = v('germany')

Measurement with similarity & distance measures, e.g. cosine similarity (sim_{cos}= 1 for identical vectors)

$$sim_{\cos}(\vec{x},\vec{y}) = \cos\alpha(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{|\vec{x}|\cdot|\vec{y}|} = \frac{\sum_{i=1}^{n}(x_i\cdot y_i)}{\sqrt{\sum_{i=1}^{n}(x_i)^2}\cdot\sqrt{\sum_{i=1}^{n}(y_i)^2}}$$

Euclidean distance (with d(p,p) = 0)

$$d(p,q) = \|q-p\|_2 = \sqrt{(q_1-p_1)^2+\cdots+(q_n-p_n)^2} = \sqrt{\sum_{i=1}^{n}(q_i-p_i)^2}$$

Word at polysemy in the middle of the context

E.g. *v('bat')* between *v('baseball')* und *v('cave')*

Bias possible due to training data

E.g. *v('doctor') – v('man') + v('woman') = v('nurse')*

Worth discussing

- What corpora were used for training
  - Identify problems/biases due to the corpus
- What's not covered by this approach?

- Google News Data set:
  100 Bio. words
  [exact sources, quality of OCR
  etc. unknown!]
- Glove:
  Wikipedia 2014
  Gigaword 5 (newswire text data)

Out of Vocabulary Words!?

Fixed vectors become a problem/can’t deal with

- Language shifts (new words emerging)
- High variability in language (not standardized like historical languages and dialects)
- Plus some smaller stuff
  - failed tokenization
  - …

Character Language Models & Tokenization not on Word-level

Word embeddings based on characters

- Trained similarly (w2v) but takes not only words (tokenized strings) into account
- Similar character strings get similar embeddings

But… what about polysemic words (homonyms)?

Context is key: Sub-Word-Token-Context-Models

Word embeddings based on context

- BERT (sub-word level embeddings)
- ELMO
- GPT

But… what about non-normalized languages

1-1 2

2-1 Räth und Bürger dieser lobigen stadt Bern, uff

2-2 ihrer gethanen Eydt zu recht erkent und

2-3 gesprochen.

2-4 daß mahn, inne dem nachrichter benehlen

2-5 der inne obenuß uff gewarlich kichstatt füren,

2-6 Inne allda uß gnad und Barmhartsigkeit wegen

2-7 das haupt abschlachen, und inne also mit dem

2-8 schwärt vom läben zum Todt nach keyßer-

2-9 lichem Rechten Richten soile.

2-10 digere urtheill ist an ime erstatet worden.

2-11 Thomas Adlis

2-12 Examen.

2-13 Allsie zu Martsili uß benelch gnädigen Herren u

2-14

2-15 den 22. Augusti 1629 durch die ehrenwetsten e

2-16

2-17 herrn simon würstenberg des raths und ihren

2-18 Ulrich Küntzi der Burgeren gegen Thomas

2-19 Adelis synen angaben nach uß Franckrych

2-20 bürtig, verrichtet.

Context + Character Language models!

Word embeddings based on context + character language modles

- FlairEmbeddings
- CharacterBERT

(and if you want to stack it all): flairNLP as framework:

https://github.com/flairNLP/flair