Applications of word vectors

Benjamin Roth

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

word vectors

- *"Word vectors":*
  - Sparse:
    - from PPMI-weighted co-occurrence matrix
    - TF-IDF weighted term document matrix
  - Dense:
    - by singular value decomposition of the PPMI (or TF-IDF) matrix
    - through gradient-based machine learning (Word2Vec, GloVe, ...)
- **What are the advantages and use cases of word vectors?**

Advantage: Universal features

- Word vectors represent all words in the same feature space.
- These features can be used to predict word properties, and be weighted by a classifier depending with respect to a task.
- Examples:
  - Word types/ POS
  - Named entity types (person, location, organization, ...)
  - Fine-grained noun-typing (software, award, politician, food, ...)
  - word sentiment (*"great"* vs. *"lame"*)
  - ...

Benefits of Dense Representations: Generalization

- Dense Representations: 50-1000 dimensions (SVD, Word2Vec, Glove, ...)
- Indirect Similarity: Because the model must compress the co-occurrence information, words are similarly represented which in turn co-occur with *similar* (but not necessarily the same) words.
  $\Rightarrow$ Better generalization
- If only a few (50-1000) features are used, there is less risk of *overfitting* a classifier (compared to using sparse PPMI vectors)

Advantage: Unsupervised

- For estimating word vectors you do not need any annotations, a sufficient amount of text (for example Wikipedia) is enough.
- Classifiers can then be trained with very little annotated data, using the previously obtained word vectors.

Example: Word 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

...

- Simple use case:
  - The classifier can be trained on an annotated sentiment lexicon, and then predict the polarity for new words (i.e., the original lexicon is extended).
  - The extended lexicon could then be used to determine the sentiment of texts (ratio of positive vs. negative words).
  - Note: The information from the word vectors can be exploited even more effectively with neural networks.
- In the example: To which features would the classifier assign **positive** feature weights, to which features **negative** weights, where would the weight be **neutral** (approximately 0)?

Example: Word 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

...

- In the example: What features would the classifier assign **positive** feature weights, which **negative** ones, where would the weight be **neutral** (approximately 0)?

Example 2: Type prediction

Word / noun phrase	**Types**
Schleswig-Holstein	location, administrative division
London Symphony Orchestra	award winner, artist, employer
Clint Eastwood	award winner, actor, producer, director, artist

...

- Given a noun phrase, predict the possible types of the described entity.
- **Use cases?**

Example 2: Type prediction

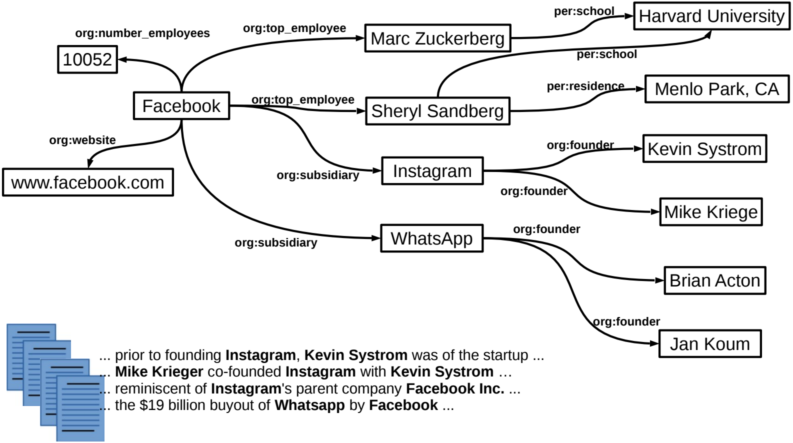
Word / noun phrase	**Typen**
Schleswig-Holstein	location, administrative area
London Symphony Orchestra	award winner, artist, employer
Clint Eastwood	award winner, actor, producer, director, artist

...

- Given a noun phrase, predict the possible types of the described entity.
- **Use cases?**
  - **Question Answering**: *Which administrative area does Kiel belong to? What actors starred in Gran Torino?*
  - **Knowledge Graph Construction**: Find all possible entities in a large amount of text, in a first step predict their types, and in a second step, what relations exist between them.

Knowledge Graph Construction

1. Find all possible entities in a large amount of text
2. **predict their types**
3. Find relations between them (depending on the types)



Example 2: Type prediction

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

...

- Differences to word polarity:
  - Instance may consist of several words, not just one.
  - Instance can have multiple types, not just a real label.
  - Possible solutions?

Example 2: Type prediction

- Differences to word polarity:
  - Instance may consist of several words, not just one.
    - ⋆ Option 1: Train with single words and combine the vectors after training. (Average vector, Neural network).
    - ⋆ Option 2: Combine entity phrases into pseudo-words before training (Clint_Eastwood)[1].
      Phrases can be found through a tagger, or through co-occurrence (PPMI). Advantage: Vector exactly for this phrase. Disadvantage: Not compositional. One needs to know phrases before training or there is a coverage problem.
  - Instance can have multiple types, not just a real label.
    ⇒ Solution: Prediction for each type (multi-label classification). Each type is encoded in a label vector elsewhere.

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

Practical information

Practical information

- Efficient implementations of Word2Vec, for example:
  `https://radimrehurek.com/gensim/models/word2vec.html`
- Pretrained GloVe vectors:
  `https://nlp.stanford.edu/projects/glove/`
- Multilabel classification with Scikit-learn:
  - X: training data/features, Matrix (n_samples $\times$ n_features)
  - Y: traning data/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)
```

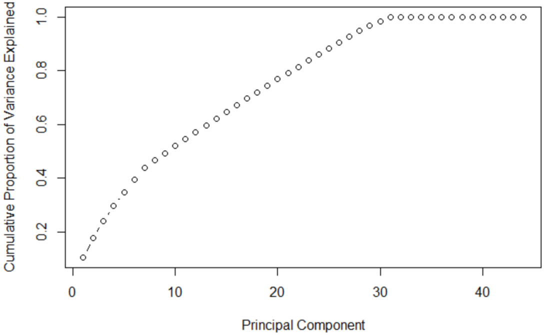
  - Instead of SVC, other classifiers (LogisticRegression ...) can also be selected.
  - Prediction is again (n_samples $\times$ n_classes) 0-1 matrix

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

Selection of the number of dimensions for an embedding space

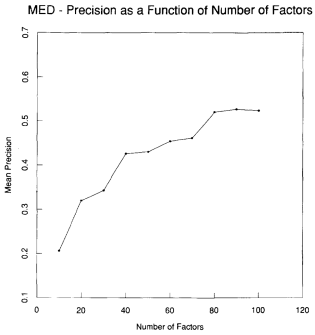
Classical Statistics: Proportion of explained variance

- e.g. with truncated SVD
- How close is the reconstruction to the original PPMI matrix?
  - 0% ⇔ always prediction of the mean value (of all entries in the matrix)
  - 100% ⇔ perfect reconstruction of the matrix
- One possibility is then to look where the extra explanatory utility decreases (*"kink"* in the graph, *"elbow"*), and only use the singular values / dimensions up to that point.



Selection in relation to task

- If you have annotated development data, you can try different sizes of the embedding space and evaluate the performance.
- Requires a task-specific cost function.
- Choose number with the lowest cost (with the largest utility)
- From the original LSI paper:



Comparison of methods for word vectors

Comparative aspects

- **order**: is the order of context words used in training?
- **time to train**: Is an efficient training possible?
- $n > 1$ **lang's**: Are embeddings in multiple languages comparable?
- **syntax**: Is the syntactic information (e.g. dependency relation) between words taken into account during training?

Further comparative aspects

- We have seen some aspects that distinguish models of word vectors.
- **compact**: Is the model compact (dense, low-dimensional) or not? (e.g., SVD vs. Wordspace)
- **rare words**: Can rare or out-of-vocabulary (OOV) words be represented well? (e.g., fasttext vs. word2vec)
- **units**: What are the representative units in training? Words (w), letters (characters, c), paragraphs (paragraphs, p)

Categorization according to 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			−	+

References:

- 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.

Recursive Neural Networks (RNNs)

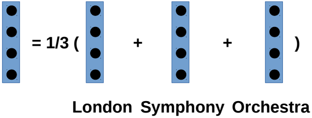
Recursive Neural Networks: Motivation

How do you ...

- ... best represent a sequence of words as a vector?
- ... combine the learned word vectors effectively?
- ... retain the information relevant to a particular task (certain features of particular words), suppress unessential aspects?

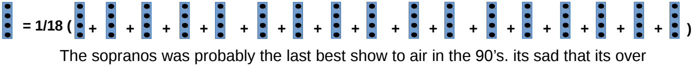
Recursive Neural Networks: Motivation

For short phrases: average vector could be one possibility



$\Rightarrow$ employer?

For long phrases problematic.

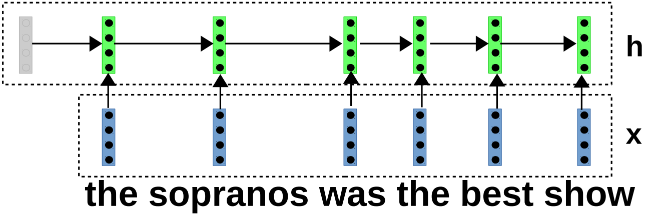


- Any information about the order of words is lost.
- There are no parameters that can already during combination distinguish between important and unimportant information. (Only the classifier can try this).

Recursive Neural Networks: Idea

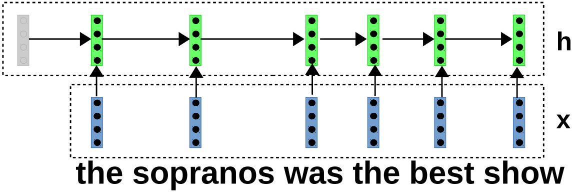
- Calculate for each position (" time step ") in the text a representation that summarizes all essential information up to this position.
- For a position t this representation is a vector $\boldsymbol{h}^{(t)}$ (hidden representation)
- $\boldsymbol{h}^{(t)}$ is calculated recursively from the word vector $\boldsymbol{x}^{(t)}$ and the hidden vector of the previous position:

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



Recursive Neural Networks

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



- The hidden vector in the last time step $\boldsymbol{h}^{(n)}$ can then be used for classification (*"Sentiment of the sentence?"*)
- The predecessor representation of the first time step uses the $\mathbf{0}$ vector (containing only zeros).

Recursive function f

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

- The f function takes two vectors as input and outputs a vector.
- f is in most cases a combination of:
  - **Vector matrix multiplication**:
    - Simplest form of mapping a vector onto a vector.
    - First, the vectors $\boldsymbol{h}^{(t-1)}$ (k components) and $\boldsymbol{x}^{(t)}$ (m components) are concatenated (appended):
      Result $[\boldsymbol{h}^{(t-1)}; \boldsymbol{x}^{(t)}]$ has $k + m$ components.
    - Weight matrix W (size: $k \times (k + m)$) is optimized when training the RNN.
  - and a **non-linear function** (e.g., logistic sigmoid) applied to all components of the resulting vector.
    - This is necessary so that the network can compute interesting, non-linear interactions, such as the effect of negation.

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

Summary

- Advantages of word vectors
  - Serve as features
  - Allow generalization
  - Can be learned non-supervised
- Use cases
  - type prediction
  - classification of word sentiment
- Neural networks
  - Recursive calculation of the hidden layer
  - Non-linearity allows more powerful representation than average vector