VL Deep Learning for Natural Language Processing

08. Word Embeddings II

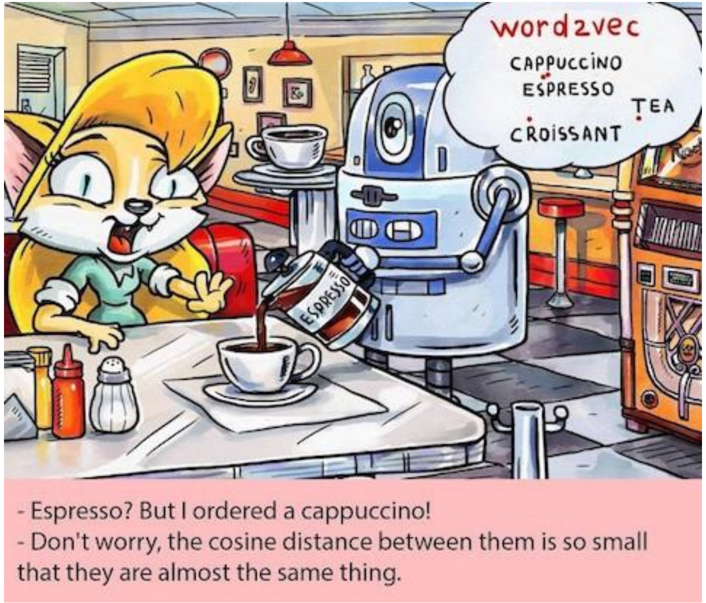
Prof. Dr. Ralf Krestel

AG Information Profiling and Retrieval

Word Embeddings

- Word embeddings represent words (discrete variables) as vectors
- Reduce the dimensionality
- Similar objects are closer to each other
  - Cosine similarity

- Neighbors of *information*
  - *info*
  - *data*
  - *documents*
  - *details*
  - *knowledge …*

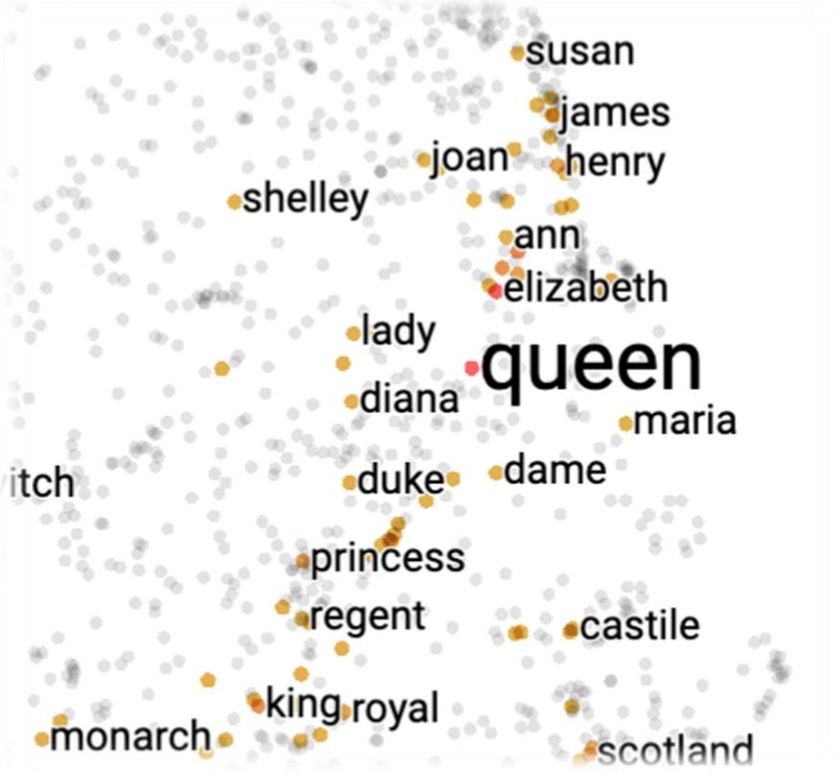


https://miro.medium.com/max/480/1*HmkxRdUcK1xZ9hQXmACg4w.jpeg

Word Embeddings

- *queen* is similar to *Elizabeth*

- Neighbors of *queen*
  - *elizabeth*
  - *anne*
  - *king*
  - *mary*
  - *princess*
  - *Catherine*
  - *Victoria*
  - *royal …*



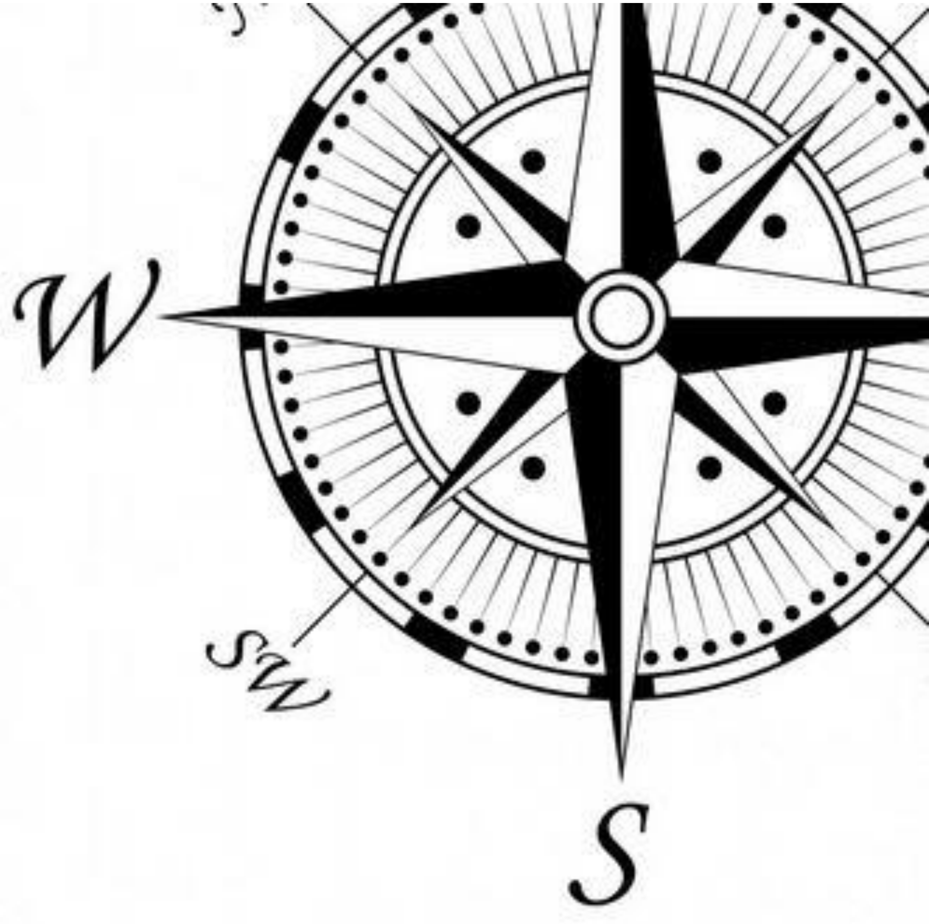
Lerning Goals for this Chapter

- Know different methods to evaluate word embeddings
  - Pros and cons of the methods
- Be able to name limitations of the evaluations

- Can implement a DNN in Keras which makes use of pretrained word vectors
- Be able to evaluate different evaluation methods for word vectors in Keras

- Relevant chapters
  - P6.1,S2

Topics Today



1. **Evaluation of Word Vectors**
2. DNN with Embedding Layer

What is a Good Word Vector?

- How to evaluate NLP systems in general? **Quantitatively**
  - **Intrinsically**
    - Based on a small, well-defined specific task
    - Useful to understand components
    - Gain in performance only useful if a connection to a real task exists
    - Fast to compute
  - **Extrinsically**
    - Based on a concrete, „real“, complex task
    - The whole system is evaluated, for NLP: the complete processing pipeline
    - Hard to tell which components of the system are performing well
    - Might take a while (at least longer than individual components)
    - Ablation test: exchanging/improving one particular component improves system
      $\longrightarrow$ The new component **is better** than the old one!

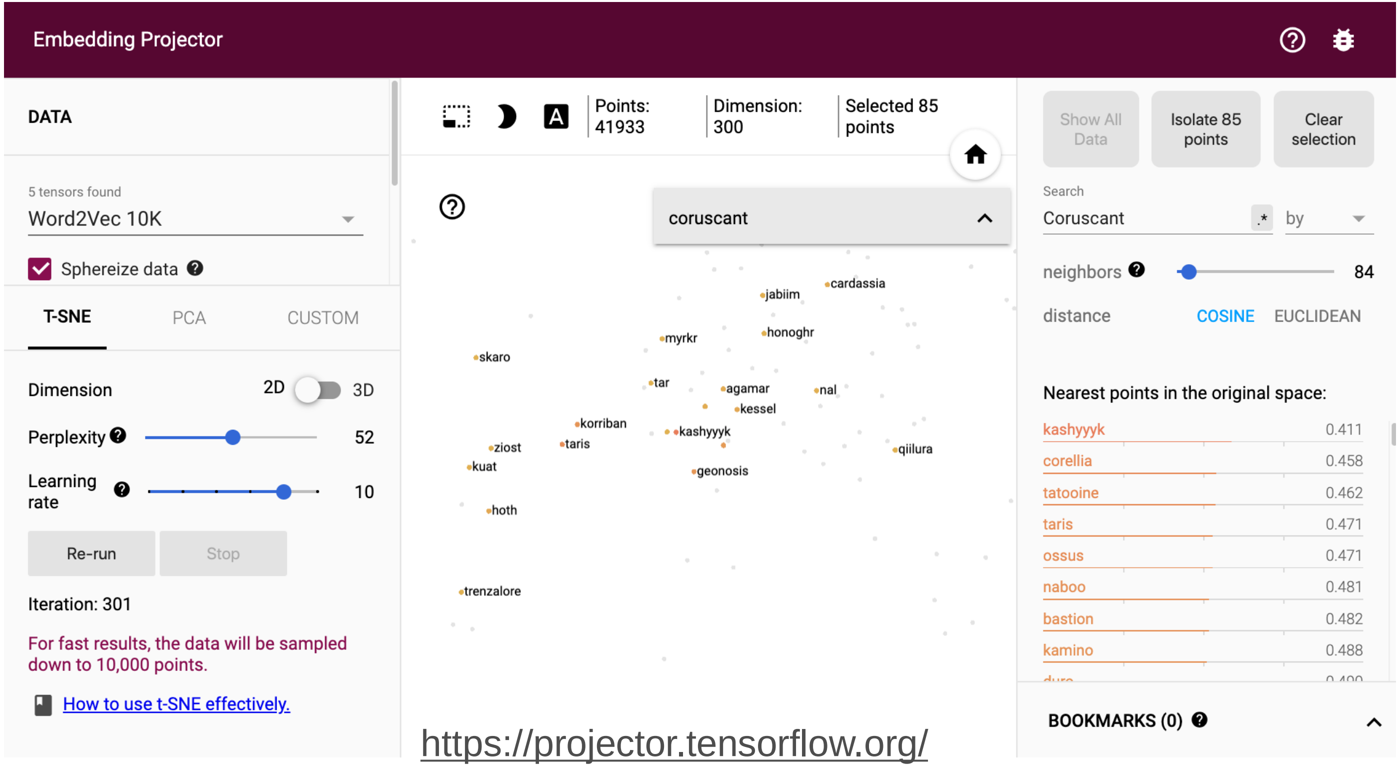
Evaluation Methods for Word Embeddings

- Intrinsically
  - Word analogy task
  - Correlation with human assessment
- Extrinsically
  - All kinds of downstream tasks
    - Classificatoin of documents
    - Classificatoin of words
    - Clustering of documents/words
    - …and many more

- Qualitatively (anecdotal evidence)
  - Nearest neighbors

In contrast: quantitatively (empirical evidence)

Interactive Demo



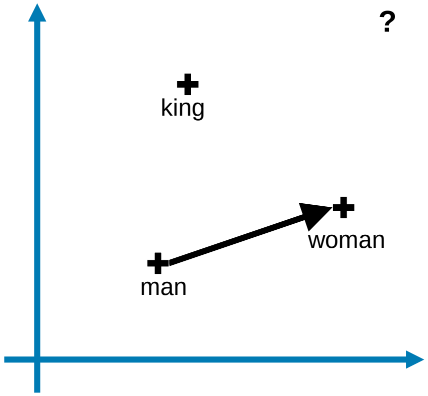
https://projector.tensorflow.org/

Links to Word Embedding Visualizations

- https://projector.tensorflow.org/?config=https://gist.githubusercontent.com/julian-risch/9d6d125b7b5e49eb9b1ffacfd6de922a/raw/22b39854e2f5acb31dac1d586871d3fee7a4d0ca/1-10k-projector_config.json
- https://projector.tensorflow.org/?config=https://gist.githubusercontent.com/julian-risch/0e4bc9ac0d5fdae61639faae8eddf23e/raw/640cf0266eee2d468b0294f8616d98da183b9881/2-projector_config.json
- https://projector.tensorflow.org/?config=https://gist.githubusercontent.com/julian-risch/15ceed73ed4beb0e84849804bd08f1d2/raw/efa486f76b49207f3111dfa318f9ea2fc7ae111e/3-projector_config.json

Word Analogy Task

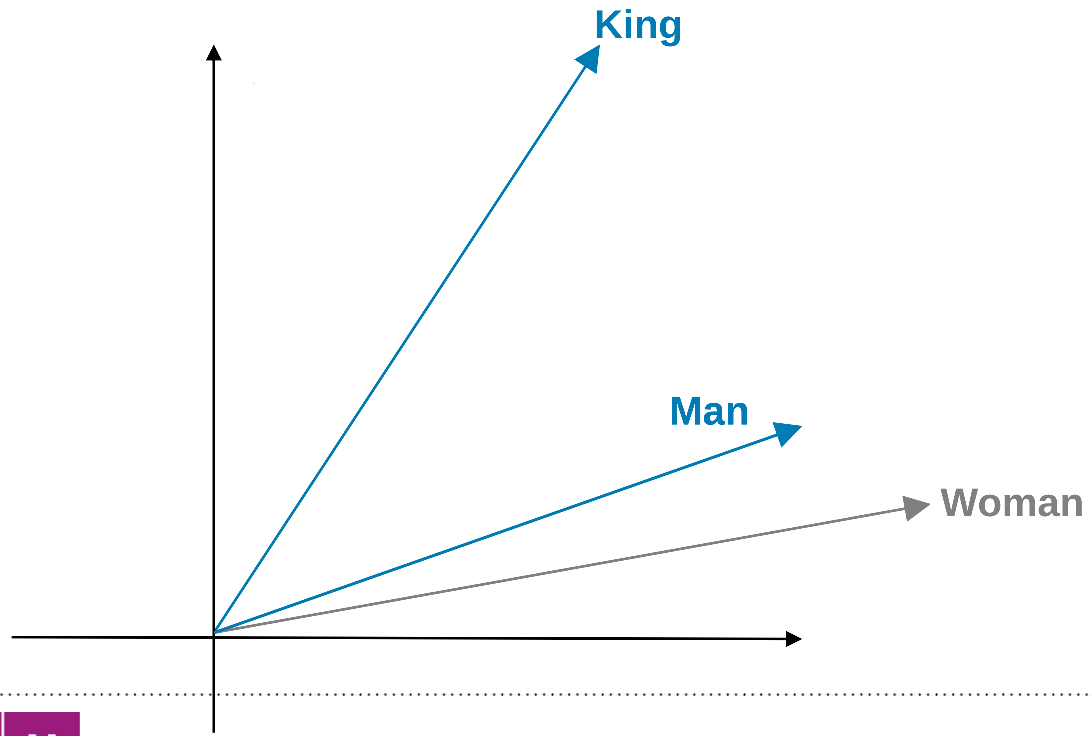
- Word vector analogies:
  - $a : b \ :: \ c : ?$
    - man:woman :: king:?
  - $d = \underset{i}{argmax} \frac{(x_b - x_a + x_c)^T x_i}{\|x_b - x_a + x_c\| \|x_i\|}$



- How well does the cosine distance describe syntactic and semantic analogies?
  - Input (query) words are not considered in the results
  - Problem: What happens with non-linear relations?

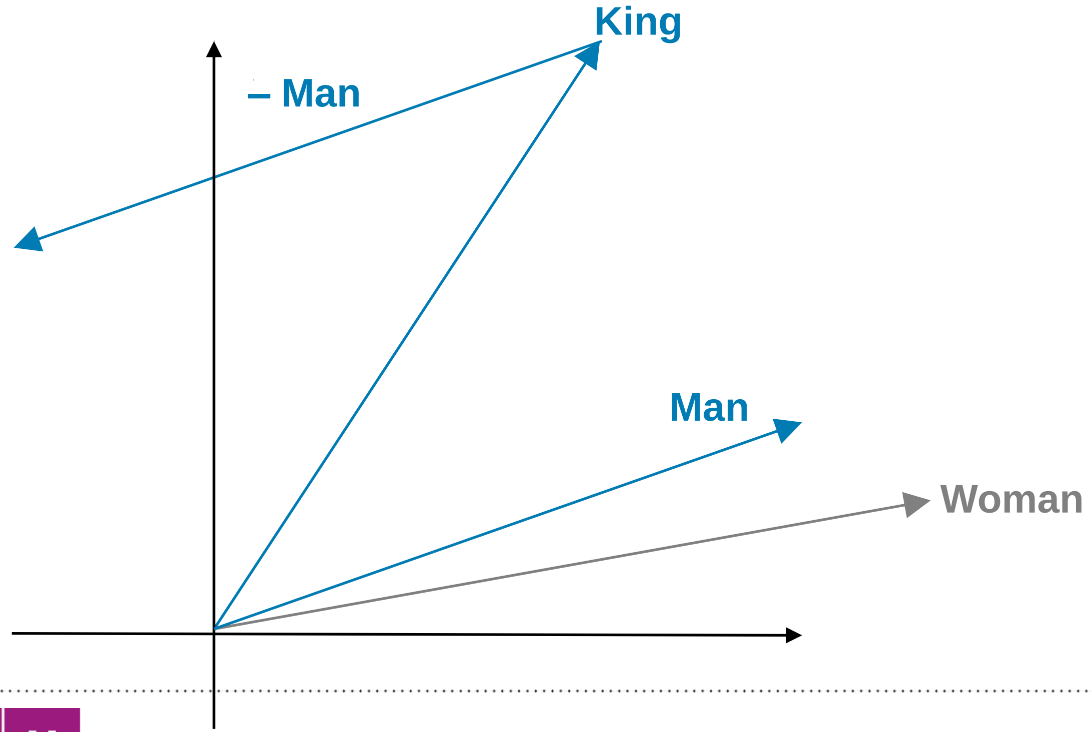
Arithmetics in Vector Space

- King – Man + Woman = ?



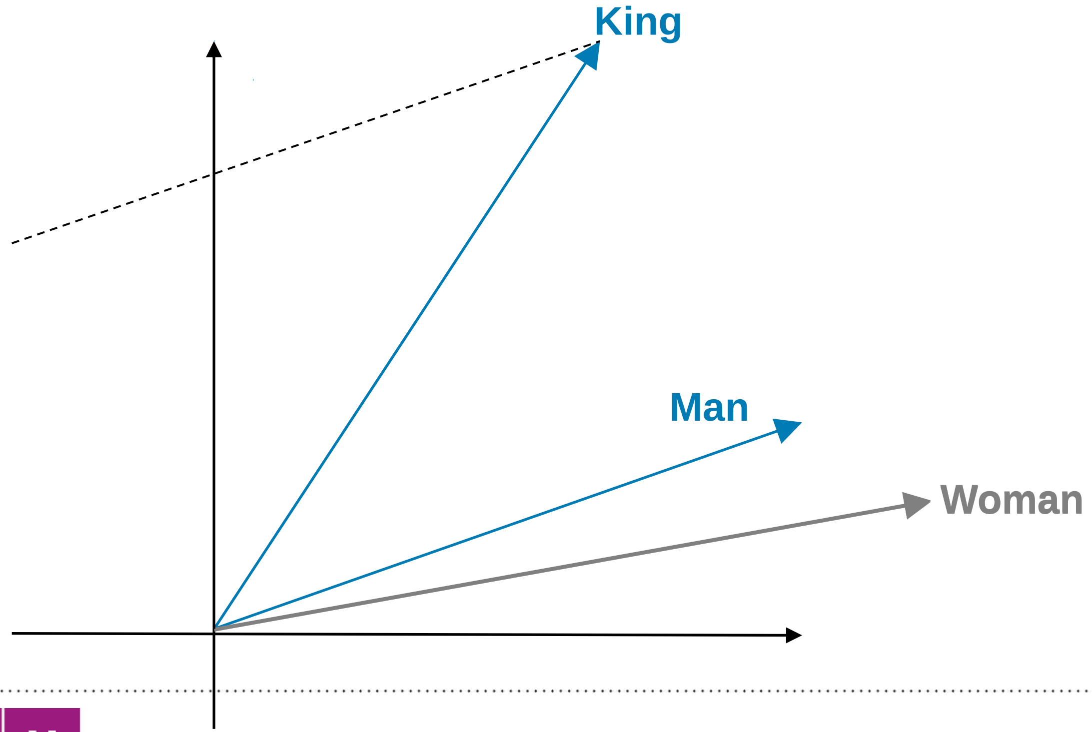
Arithmetics in Vector Space

- King – Man + Woman = ?



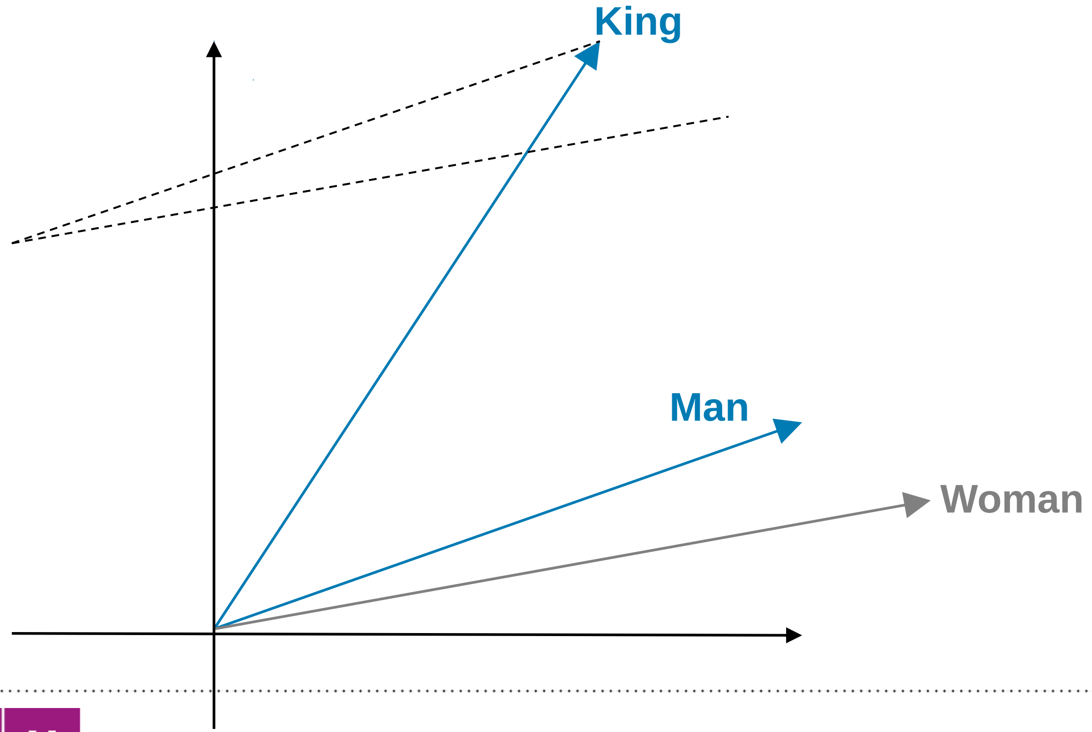
Arithmetics in Vector Space

- King – Man + Woman = ?



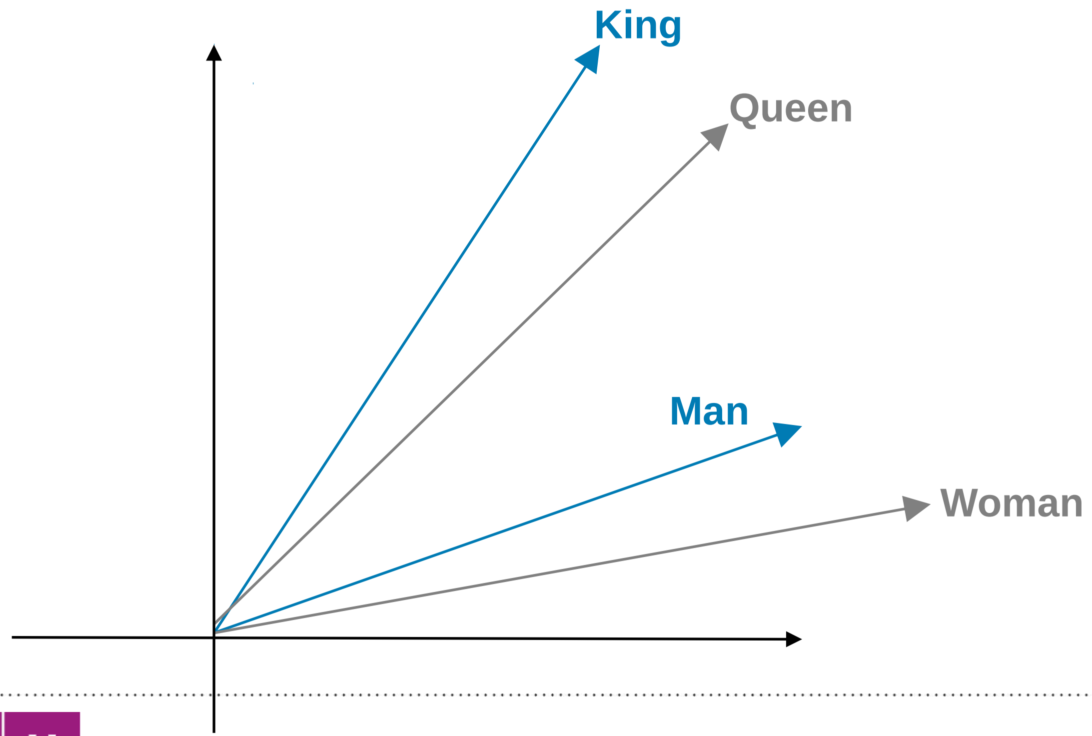
Arithmetics in Vector Space

- King – Man + Woman = ?



Arithmetics in Vector Space

- King – Man + Woman = Queen



Word2Vec -> Nerd2Vec

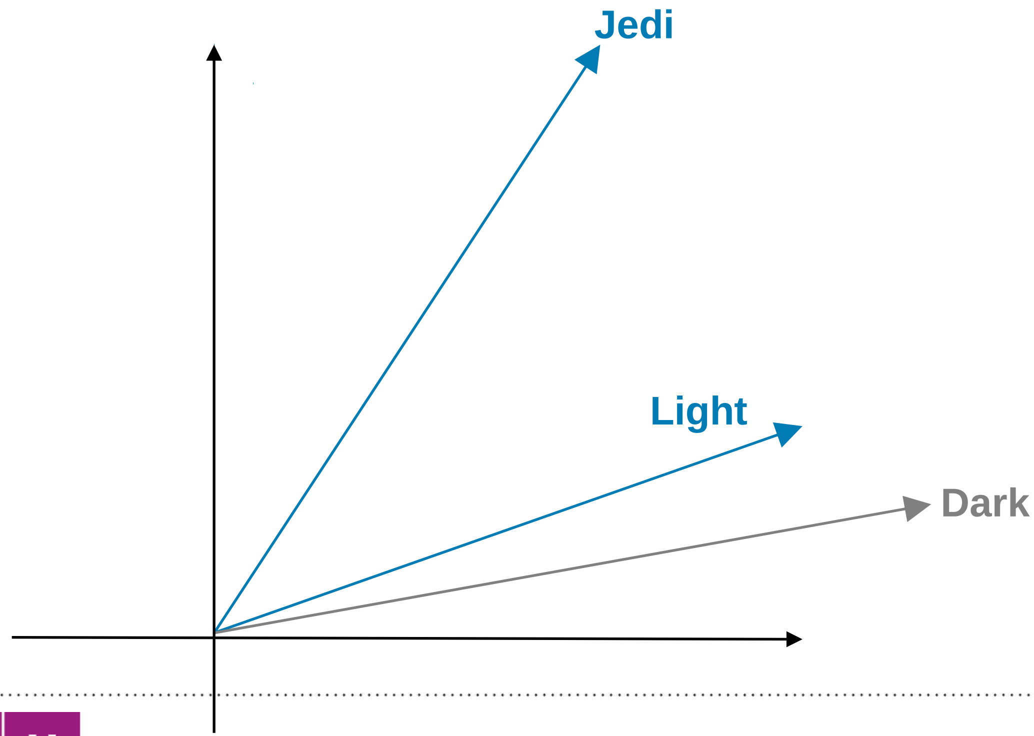
- Word2Vec
  - Assign a real-valued vector representation to each word
  - Learn the vectors on large corpora
  - Words that appear in similar context shall have similar vectors

- Nerd2Vec
  - Based on Wookieepedia, a Star Wars Wiki
  - Captures semantic similarities of fictional characters, locations, e

https://blogs.oracle.com/irml/nerd2vec:-jointly-embedding-star-trek,-star-wars-and-doctor-who-wikias

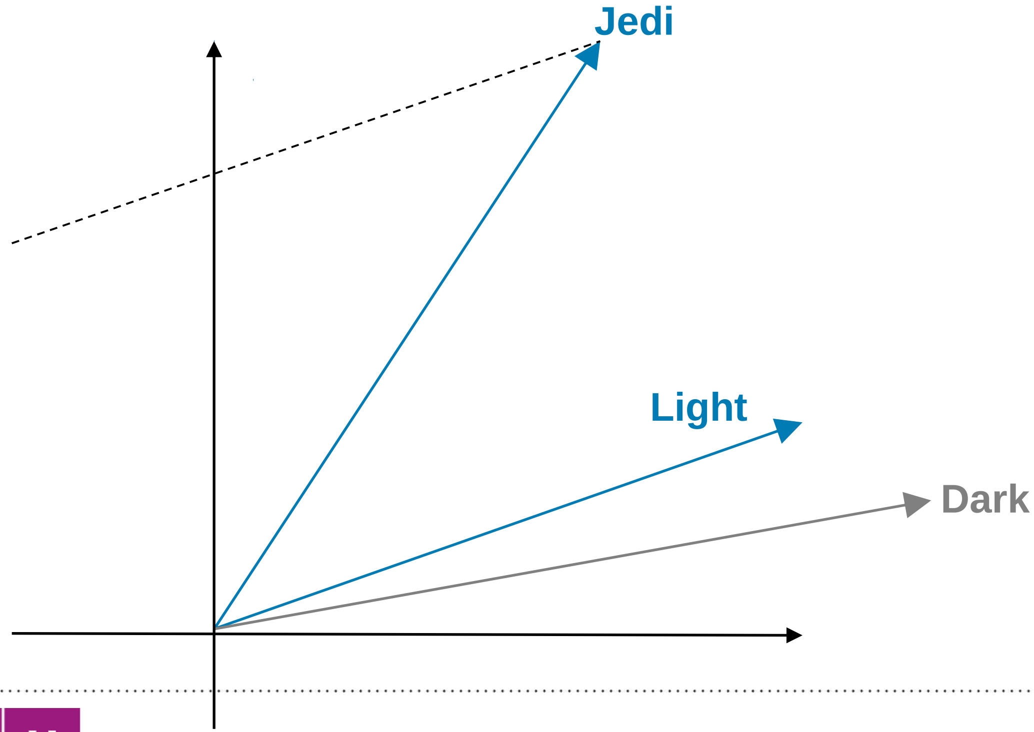
Arithmetics in Vector Space

- Jedi – Light + Dark = ?



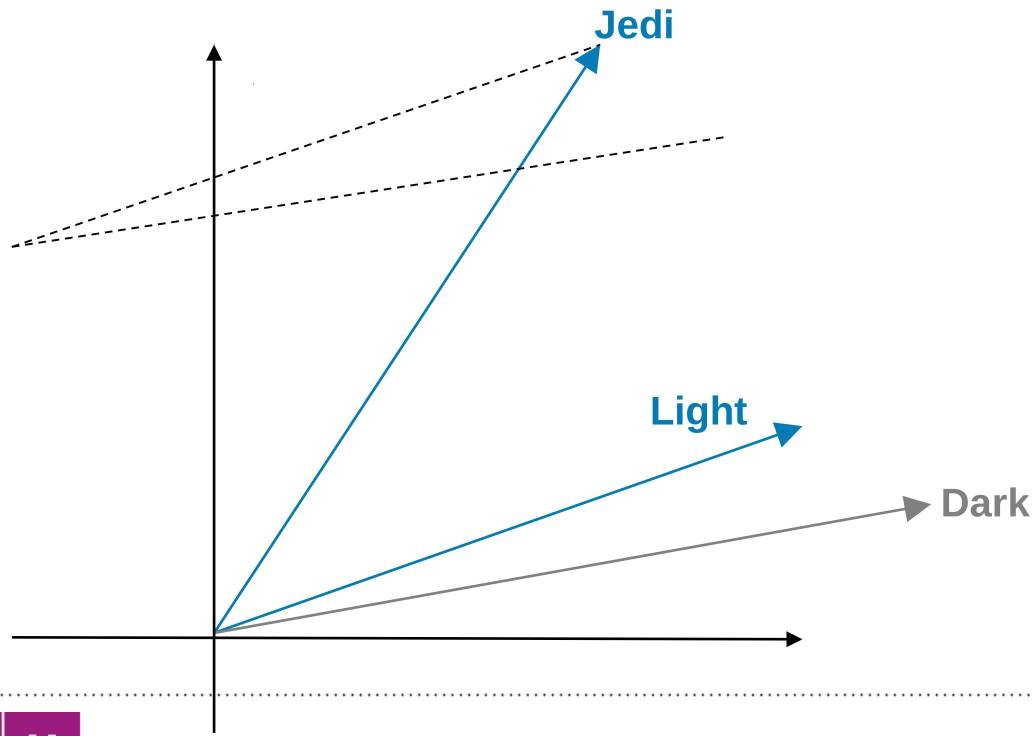
Arithmetics in Vector Space

- Jedi – Light + Dark = ?



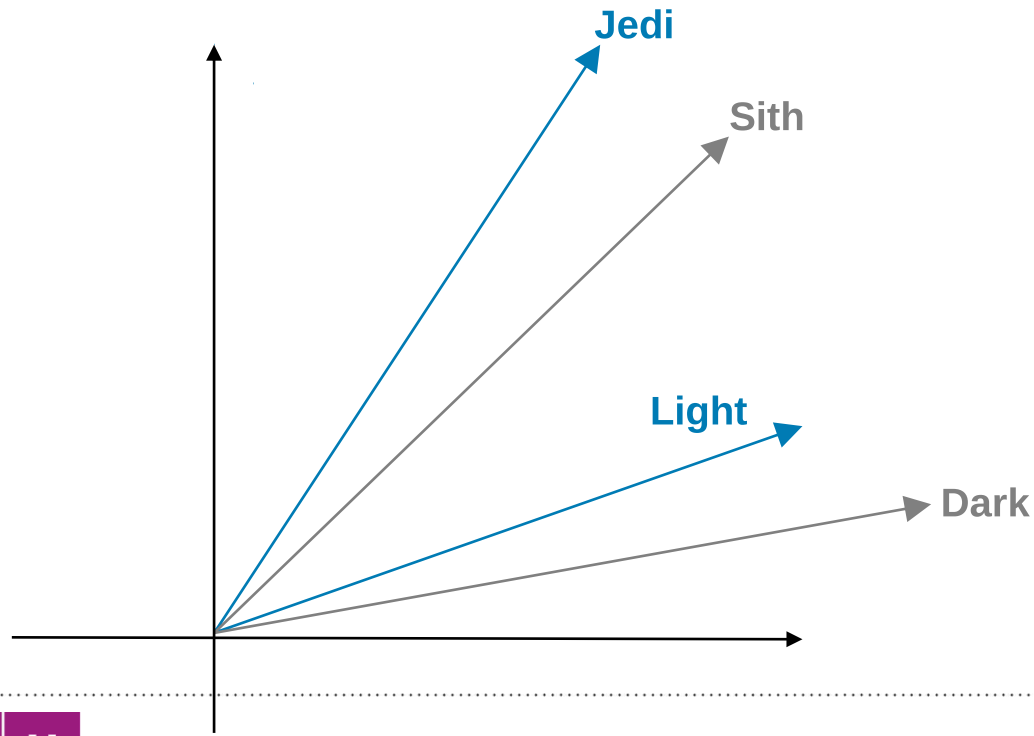
Arithmetics in Vector Space

- Jedi – Light + Dark = ?



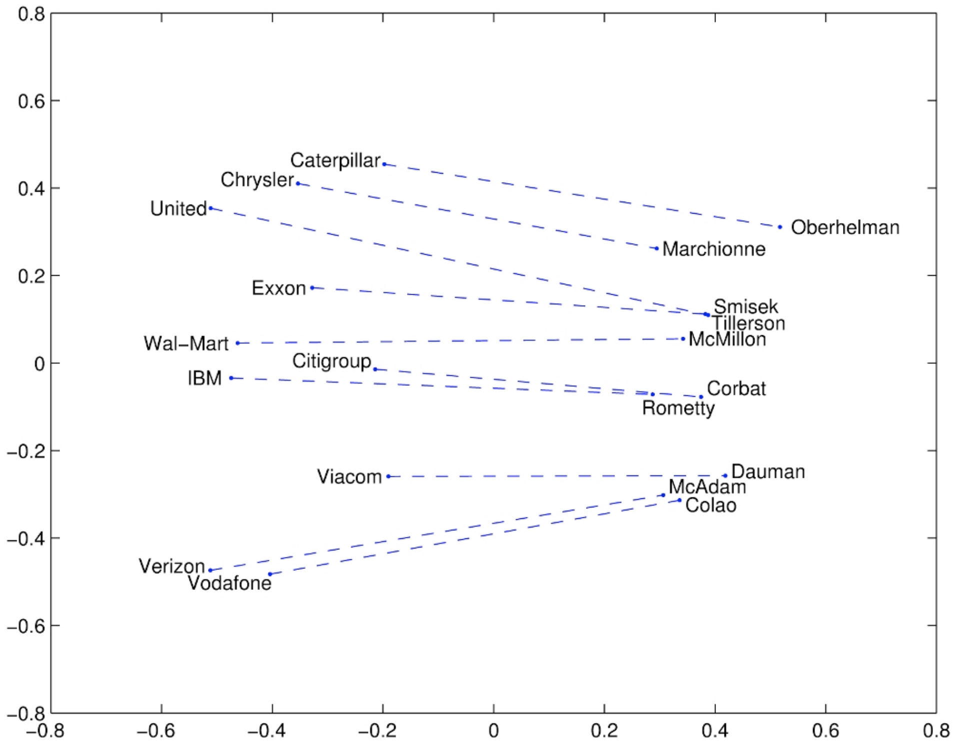
Arithmetics in Vector Space

- Jedi – Light + Dark = Sith



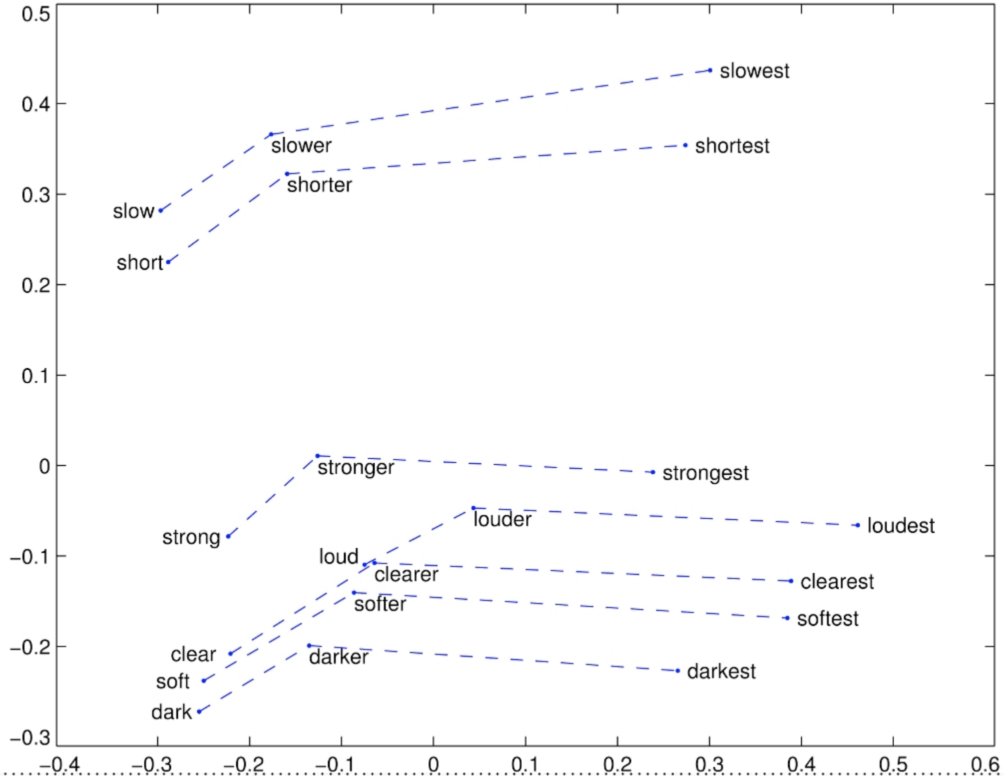
Word Analogies: Example GloVe I

- Company - CEO



Word Analogies: Example GloVe II

- Superlatives



Word Analogies: Example Word2Vec

Vector Expression					Nearest Word
Paris	-	France	+	Italy	Rome
Bigger	-	Big	+	Cold	Colder
Sushi	-	Japan	+	Germany	Bratwurst
Cu	-	Copper	+	Gold	Au
Windows	-	Microsoft	+	Google	Android
Montreal Canadiens	-	Montreal	+	Toronto	Toronto maple leafs

Word Analogies: Gold Standard Dataset I

- Semantic examples
- city-in-state
  - Chicago Illinois Houston Texas
  - Chicago Illinois Philadelphia Pennsylvania
  - Chicago Illinois Phoenix Arizona
  - Chicago Illinois Dallas Texas
  - Chicago Illinois Jacksonville Florida
  - Chicago Illinois Indianapolis Indiana
  - Chicago Illinois Austin Texas
  - Chicago Illinois Detroit Michigan
  - Chicago Illinois Memphis Tennessee
  - Chicago Illinois Boston Massachusetts

Problem: Many cities have the same name

https://code.google.com/archive/p/word2vec/source/default/source/word2vec/trunk/questions-words.txt

Word Analogies: Gold Standard Dataset II

- Semantic Examples
- capital-country
  - Abuja Nigeria Accra Ghana
  - Abuja Nigeria Algiers Algeria
  - Abuja Nigeria Amman Jordan
  - Abuja Nigeria Ankara Turkey
  - Abuja Nigeria Antananarivo Madagascar
  - Abuja Nigeria Apia Samoa
  - Abuja Nigeria Ashgabat Turkmenistan
  - Abuja Nigeria Asmara Eritrea
  - Abuja Nigeria Astana Kazakhstan

Problem: Facts can change

https://code.google.com/archive/p/word2vec/source/default/source/word2vec/trunk/questions-words.txt

Word Analogies: Gold Standard Dataset III

- Syntactic examples
- gram4-superlative
  - bad worst big biggest
  - bad worst bright brightest
  - bad worst cold coldest
  - bad worst cool coolest
  - bad worst dark darkest
  - bad worst easy easiest
  - bad worst fast fastest
  - bad worst good best
  - bad worst great greatest

https://code.google.com/archive/p/word2vec/source/default/source/word2vec/trunk/questions-words.txt

Correlation

- Word vector distances and their correlation to human assessment
- Dataset: WordSim353
  - http://www.cs.technion.ac.il/~gabr/resources/data/wordsim353/

Word 1	Word 2	Human Score
Tiger	Cat	7.35
Tiger	Tiger	10.00
Book	Paper	7.46
Computer	Internet	7.58
Plane	Car	5.77
Professor	Doctor	6.62
Stock	Phone	1.62

Word	Cosine Distance to „Sweden“
Norway	0.76
Denmark	0.71
Finland	0.62
Switzerland	0.59
Belgium	0.58
Netherlands	0.57
Iceland	0.56
Estonia	0.55

What about Ambigous Words?

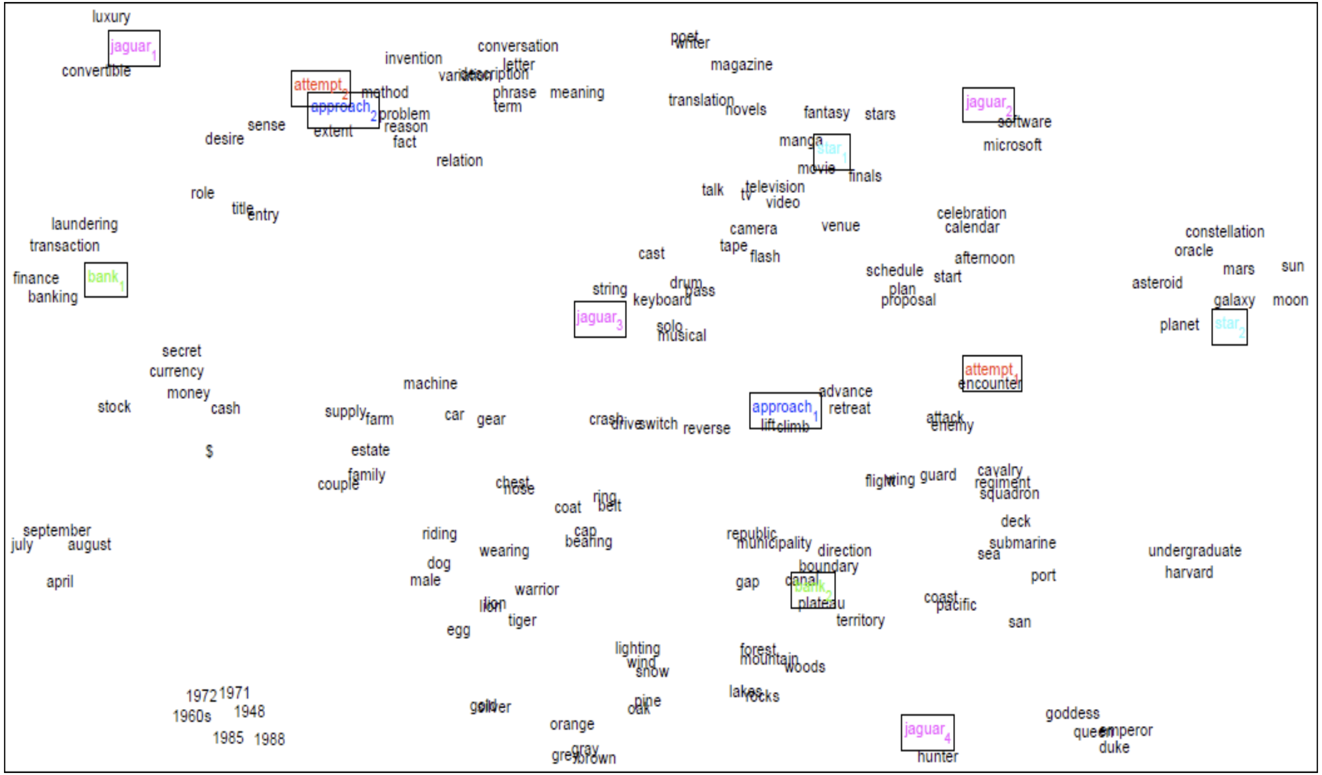
- One word = one vector
  - *to run* vs. *the run*
  - *jaguar (cat) vs. jaguar (car)*

- Idea:
  - Clustering of word windows
  - Word will be assigned to appropriate cluster
    - $jaguar_1, jaguar_2, \ldots$

Huang, E. H., Socher, R., Manning, C. D., & Ng, A. Y. (2012, July). Improving word representations via global context and multiple word prototypes. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1* (pp. 873-882). Association for Computational Linguistics.

https://encrypted-tbn0.gstatic.com/images?q=tbn%3AANd9GcQpSr-u-en0Bnsal_G9MghQP577_ukz4LADkuKPlCsDPzqVlT_1
https://encrypted-tbn0.gstatic.com/images?q=tbn%3AANd9GcTuUpkl4OEJWAH2sQdC2jEqRQn7mDbC1XS09XtRqthM0zetY_iq

Vectors Learned from Local and Global Context



Extrinsically: NER

- Named Entity Recognition (NER)
  - Word classification task
    - Denotes a word a person, organisation or location?
  - Better word vectors
    = better representation of input words
    = better features
    = higher accuracy for classification task

Model	Dev	Test	ACE	MUC7
Discrete	91.0	85.4	77.4	73.4
SVD	90.8	85.7	77.3	73.7
SVD-S	91.0	85.5	77.6	74.3
SVD-L	90.5	84.8	73.6	71.5
HPCA	92.6	**88.7**	81.7	80.7
HSMN	90.5	85.7	78.7	74.7
CW	92.2	87.4	81.7	80.2
CBOW	93.1	88.2	82.2	81.1
GloVe	**93.2**	88.3	**82.9**	**82.2**

Bias in Word Embeddings

- Gender stereotypes

Occupations as projected on to the she−he gender direction on w2vNEWS

Extreme *she*	**Extreme *he***
1. homemaker	1. maestro
2. nurse	2. skipper
3. receptionist	3. protege
4. librarian	4. philosopher
5. socialite	5. captain
6. hairdresser	6. architect
7. nanny	7. financier
8. bookkeeper	8. warrior
9. stylist	9. broadcaster
10. housekeeper	10. magician

Automatically generated analogies for the pair she-he

Gender stereotype *she*-*he* analogies

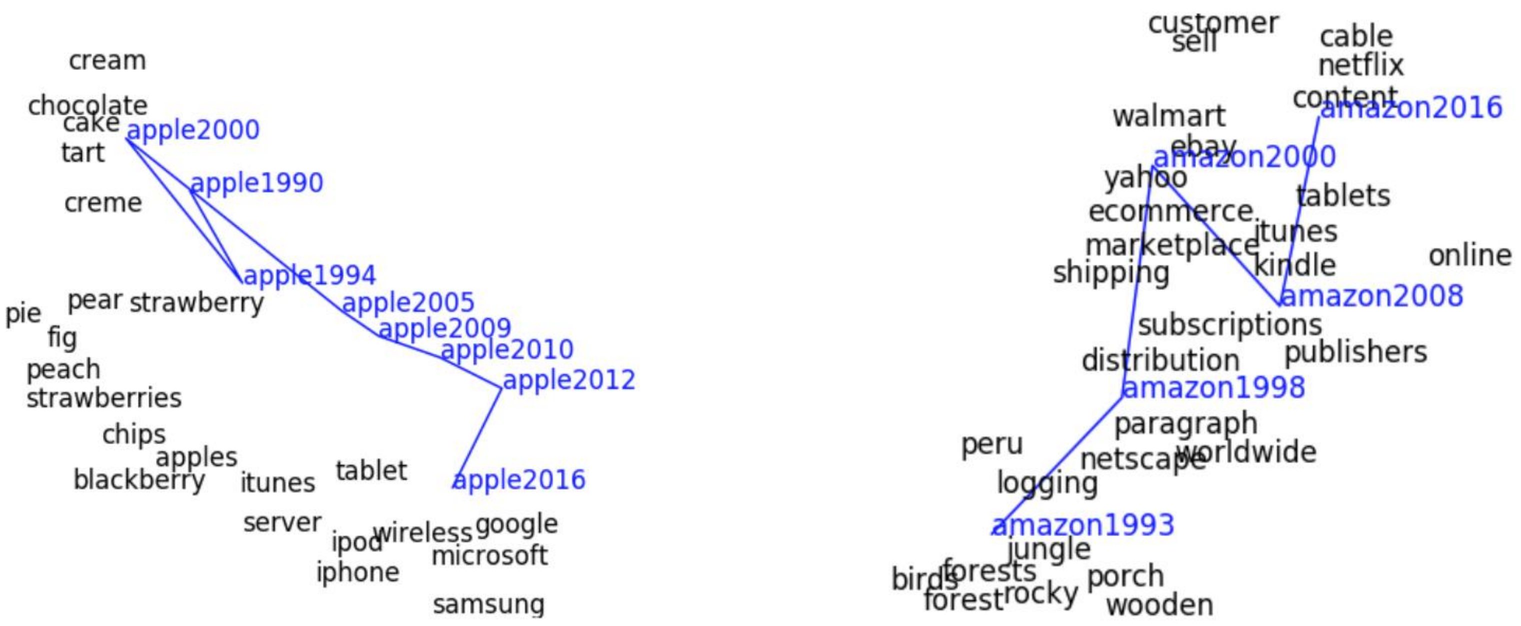
sewing-carpentry	registered nurse-physician	housewife-shopkeeper
nurse-surgeon	interior designer-architect	softball-baseball
blond-burly	feminism-conservatism	cosmetics-pharmaceuticals
giggle-chuckle	vocalist-guitarist	petite-lanky
sassy-snappy	diva-superstar	charming-affable
volleyball-football	cupcakes-pizzas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

Gender appropriate *she*-*he* analogies

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

Bolukbasi, T., Chang, K. W., Zou, J. Y., Saligrama, V., & Kalai, A. T. (2016). Man is to computer programmer as woman is to homemaker? Debiasing word embeddings. *Advances in neural information processing systems* (NIPS), 4349-4357.

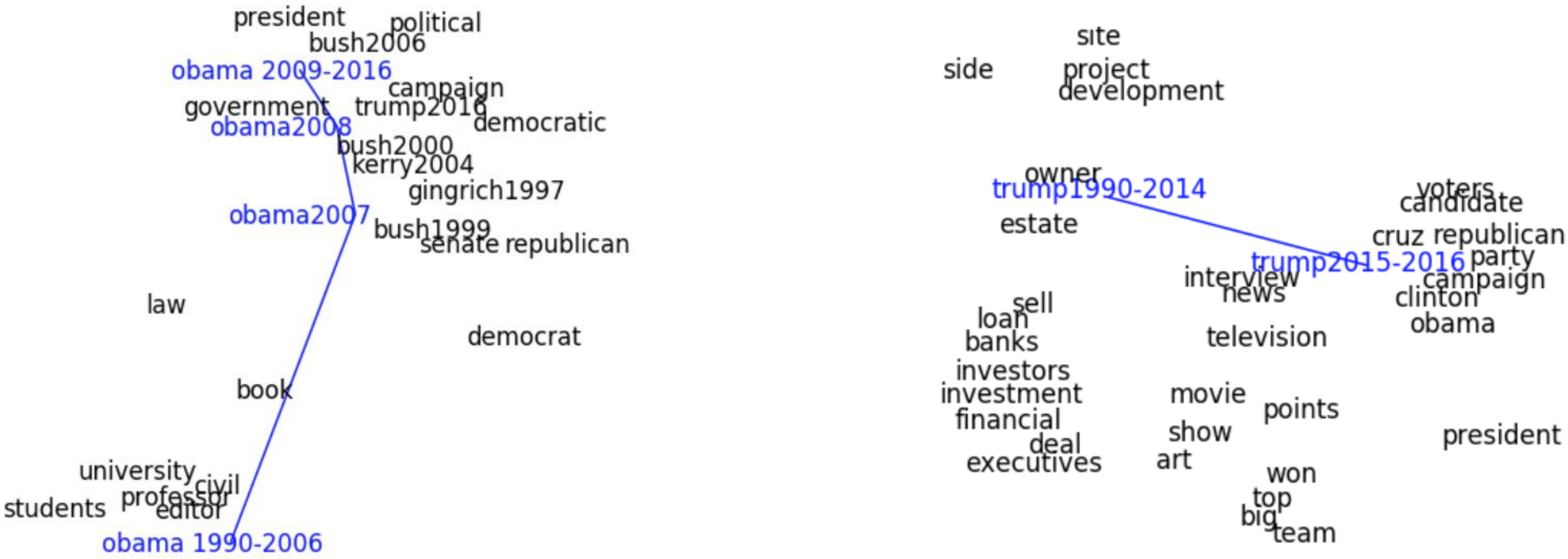
Change Analysis



(a) apple **(b) amazon**

Yao, Z., Sun, Y., Ding, W., Rao, N., & Xiong, H. (2018, February). Dynamic word embeddings for evolving semantic discovery. In *Proceedings of the eleventh international conference on web search and data mining* (WSDM) (pp. 673-681).

Change Analysis [Yao18]



(c) obama (d) trump

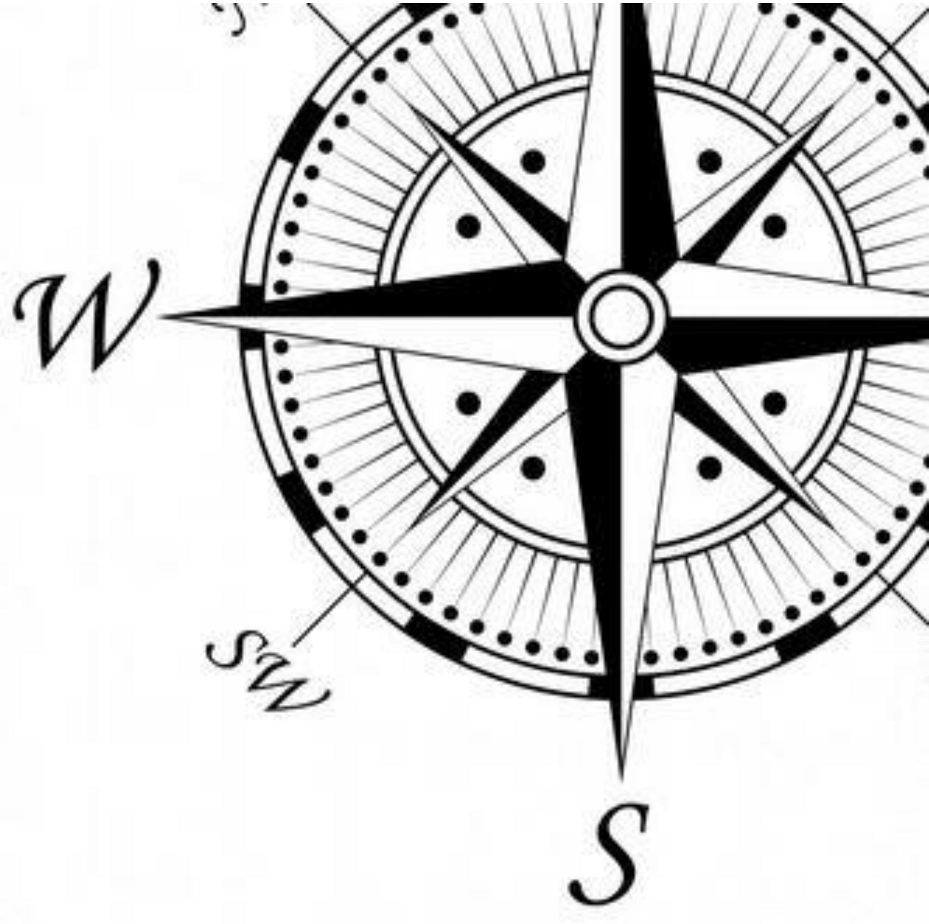
Evaluation of Word Vectors

- How could you generate a gold standard word analogy dataset automatically?

Start 5 4 3 2 1 End

Topics Today

1. Evaluation of Word Vectors
2. **DNN with Embedding Layer**



Input Data = Text

- Deep neural networks need tensors as input
  - One-hot-encoding
    - Bag-of-words
    - Sparsely populated
  - Embedding
    - Dimensionality reduction
    - Densly populated

- Embedding Layer can learn representation **task-specific**
  - First layer in a DNN learns dimensionality reduction / representation
- **Pretrained** Embeddings can be used
  - First layer maps input words to pretrained word vectors

IMDB-Datensatz

- Movie reviews http://mng.bz/0tIo
  - Input: reviews as String, Output: labels (pos/neg)

```
import os
imdb_dir = '/users/krestel/Downloads/aclImdb`
train_dir = os.path.join(imdb_dir, 'train')
labels = []
texts = []
for label_type in ['neg', 'pos']:
    dir_name = os.path.join(train_dir, label_type)
    for fname in os.listdir(dir_name):
        if fname[-4:] == '.txt':
            f = open(os.path.join(dir_name, fname))
            texts.append(f.read())
            f.close()
            if label_type == 'neg':
                labels.append(0)
            else:
                labels.append(1)
```

Tokenization

- If you have a large amount of data:
  – Learn embeddings yourself

```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing.sequence import pad_sequences
import numpy as np
maxlen = 100
training_samples = 200
validation_samples = 10000
max_words = 10000
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(texts)
sequences = tokenizer.texts_to_sequences(texts)
word_index = tokenizer.word_index
print('Found %s unique tokens.' % len(word_index))
```

Only first 100 words per review

Only 200! Training samples

Training and Validation Data

- Since data is sorted, shuffling very important

```
data = pad_sequences(sequences, maxlen=maxlen)
labels = np.asarray(labels)
print('Shape of data tensor:', data.shape)
print('Shape of label tensor:', labels.shape)
indices = np.arange(data.shape[0])
np.random.shuffle(indices)
data = data[indices]
labels = labels[indices]
x_train = data[:training_samples]
y_train = labels[:training_samples]
x_val = data[training_samples: training_samples + validation_samples]
y_val = labels[training_samples: training_samples + validation_samples]
```

Parsing of Pretrained Word Embeddings

- Download of pretrained GloVe word vectors
  https://nlp.stanford.edu/projects/glove

```
glove_dir = '/users/krestel/Downloads/glove.6B'
embeddings_index = {}
f = open(os.path.join(glove_dir, 'glove.6B.100d.txt'))
for line in f:
        values = line.split()
        word = values[0]
        coefs = np.asarray(values[1:], dtype='float32')
        embeddings_index[word] = coefs
        f.close()
print('Found %s word vectors.' % len(embeddings_index))
embedding_dim = 100
embedding_matrix = np.zeros((max_words, embedding_dim))
for word, i in word_index.items():
        if i < max_words:
                embedding_vector = embeddings_index.get(word)
                if embedding_vector is not None:
                        embedding_matrix[i] = embedding_vector
```

Trained on a corpus with 6 billion tokens;
100-dimensional embeddings

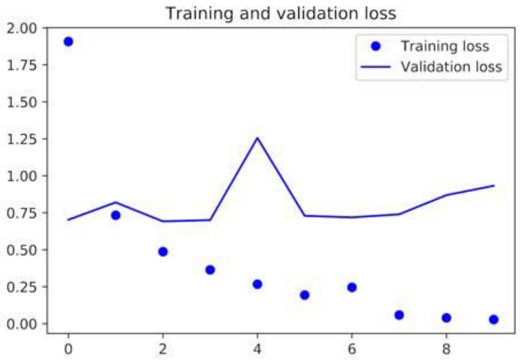
Definition and Training of the Model

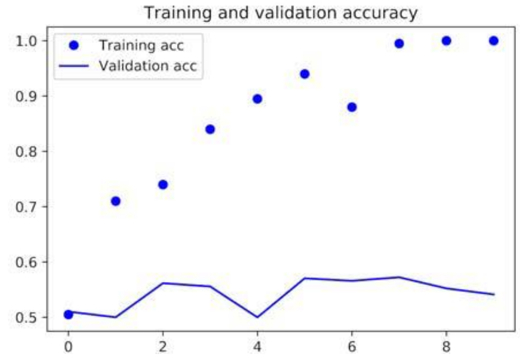
```
from keras.models import Sequential
from keras.layers import Embedding, Flatten, Dense
model = Sequential()
model.add(Embedding(max_words, embedding_dim, input_length=maxlen))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
model.layers[0].set_weights([embedding_matrix])
model.layers[0].trainable = False
model.compile(optimizer='rmsprop',
             loss='binary_crossentropy',
             metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=32,
                    validation_data=(x_val, y_val))
model.save_weights('pre_trained_glove_model.h5')
```

Loading of pretrained word vectors

Plotting the Learning Progress

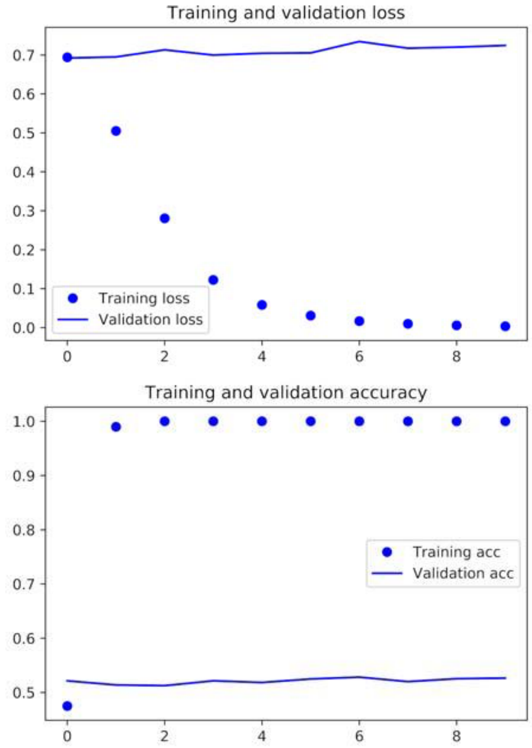
```
import matplotlib.pyplot as plt
acc = history.history['acc']
val_acc = history.history['val_acc']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(acc) + 1)
plt.plot(epochs, acc, 'bo', label='Training acc')
plt.plot(epochs, val_acc, 'b', label='Validation acc')
plt.title('Training and validation accuracy')
plt.legend()
plt.figure()
plt.plot(epochs, loss, 'bo', label='Training loss')
plt.plot(epochs, val_loss, 'b', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```





The Modell without Pretrained Vectors

```
from keras.models import Sequential
from keras.layers import Embedding, Flatten, Dense
model = Sequential()
model.add(Embedding(max_words, embedding_dim,
                    input_length=maxlen))
model.add(Flatten())
model.add(Dense(32, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
model.compile(optimizer='rmsprop',
              loss='binary_crossentropy',
              metrics=['acc'])
history = model.fit(x_train, y_train,
                    epochs=10,
                    batch_size=32,
                    validation_data=(x_val, y_val))
```

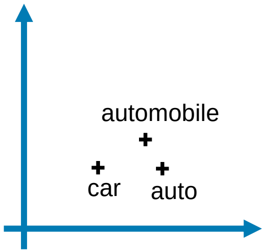


Evaluation on Test Data

```
test_dir = os.path.join(imdb_dir, 'test')
labels = []
texts = []
for label_type in ['neg', 'pos']:
        dir_name = os.path.join(test_dir, label_type)
        for fname in sorted(os.listdir(dir_name)):
                if fname[-4:] == '.txt':
                        f = open(os.path.join(dir_name, fname))
                        texts.append(f.read())
                        f.close()
                        if label_type == 'neg':
                                labels.append(0)
                        else:
                                labels.append(1)
sequences = tokenizer.texts_to_sequences(texts)
x_test = pad_sequences(sequences, maxlen=maxlen)
y_test = np.asarray(labels)
model.load_weights('pre_trained_glove_model.h5')
model.evaluate(x_test, y_test)
```

Pretrained Word Vectors vs. Newly Learned

- Classic machine learning
  - Learning feature weights ($\mathbb{R}^{Cd}$)
- Deep Learning
  - Learning feature weights and word vectors ($\mathbb{R}^{Cd+Vd}$)
  - Vd is very large
    - Danger of overfitting



Fine-Tuning

- How about a compromise?
  - Load pretrained word vectors and then continue training with current data?
- Problem:
  - Words that occur in the training set move around in the embedding space; words that do not occur in the training set but maybe in the test set stay where they are.

Embedding Layers

- Implement the previous example:
  1. With no embedding layer
  2. With pretrained embeddings
  3. With self-trained embeddings

- How many training samples are necessary to beat the performance of the pretrained word vector model?
- How can the performance be increased further?

Start 10 9 8 7 6 5 4 3 2 1 End

Lerning Goals for this Chapter

- Know different methods to evaluate word embeddings
  - Pros and cons of the methods
- Be able to name limitations of the evaluations

- Can implement a DNN in Keras which makes use of pretrained word vectors
- Be able to evaluate different evaluation methods for word vectors in Keras

- Relevant chapters
  - P6.1,S2

Literature

- Evaluation methods for unsupervised word embeddings
- Linear Algebraic Structure of Word Senses, with Applications to Polysemy
- On the Dimensionality of Word Embedding
- Debiasing Word Embeddings
- Dynamic Word Embeddings