word2vec

From theory to practice

Hendrik Heuer
Stockholm NLP Meetup

About me

Hendrik Heuer
hendrikheuer@gmail.com
http://hen-drik.de
@hen_drik

“You shall know a word
by the company it keeps”

–J. R. Firth 1957

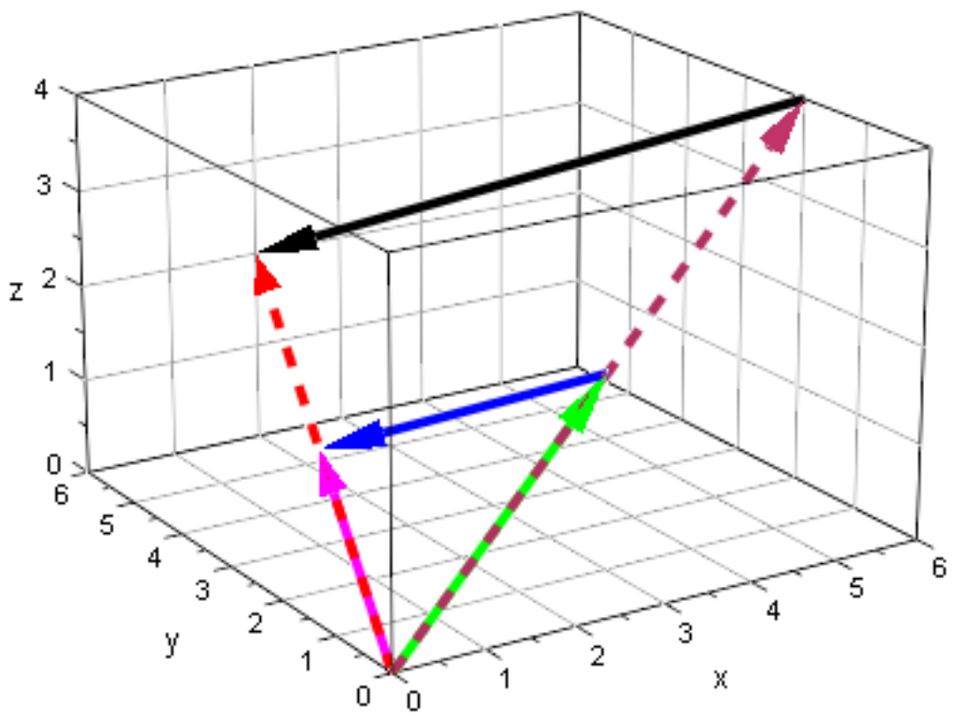
“You shall know a word by the company it keeps”

One of the most successful ideas of modern statistical NLP

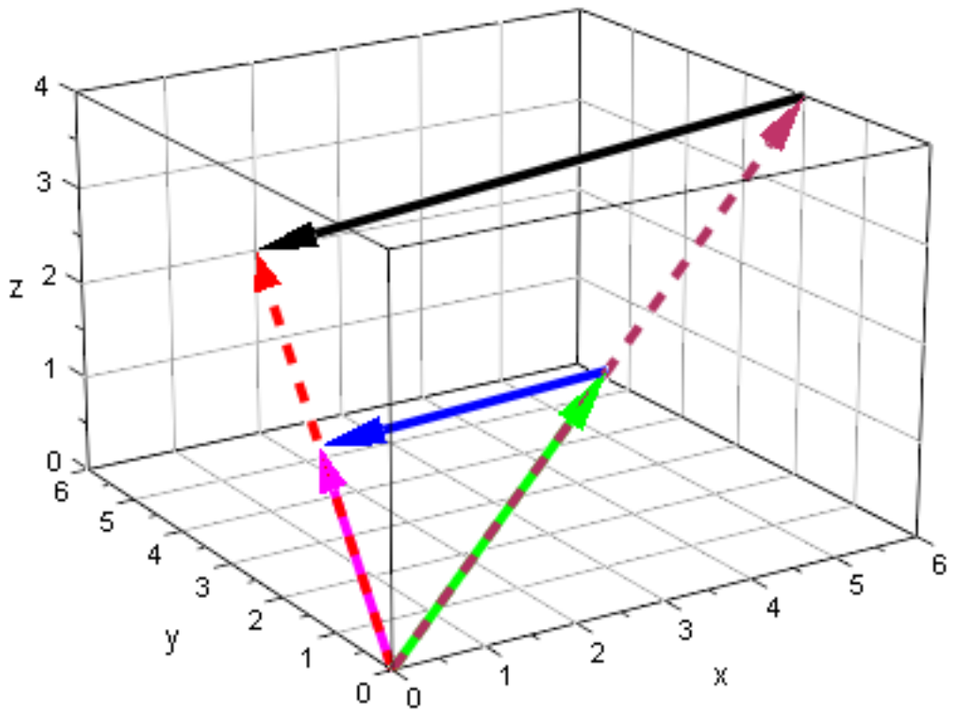
government debt problems turning into	banking	crises as has happened in
saying that Europe needs unified	banking	regulation to replace the hodgepodge

↖ These words will represent *banking* ↗

Quoted after Socher

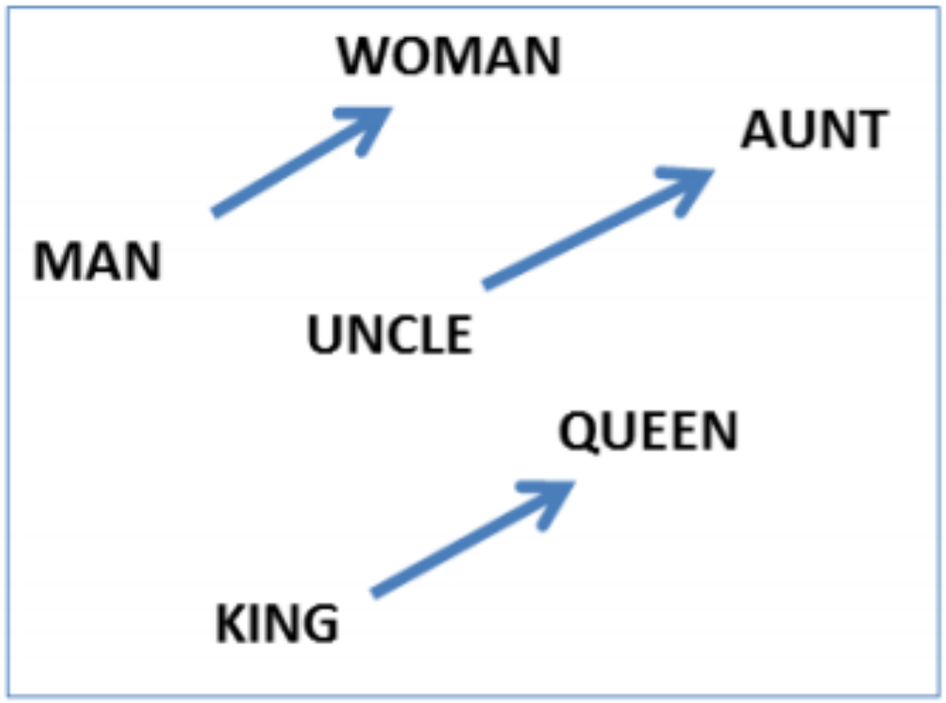


Quoted after Socher

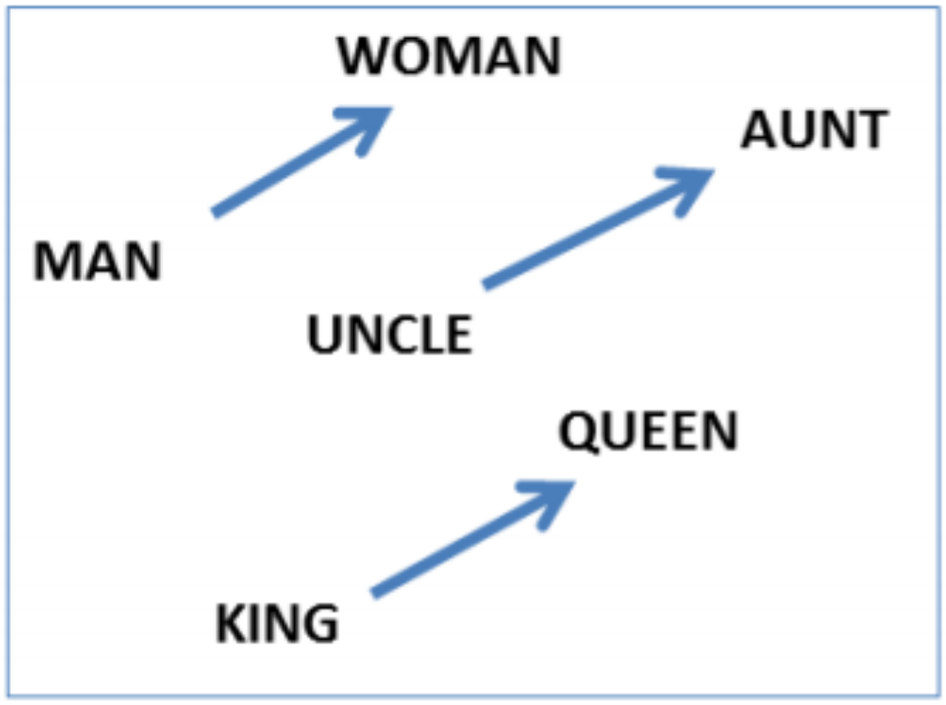


$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Quoted after Socher

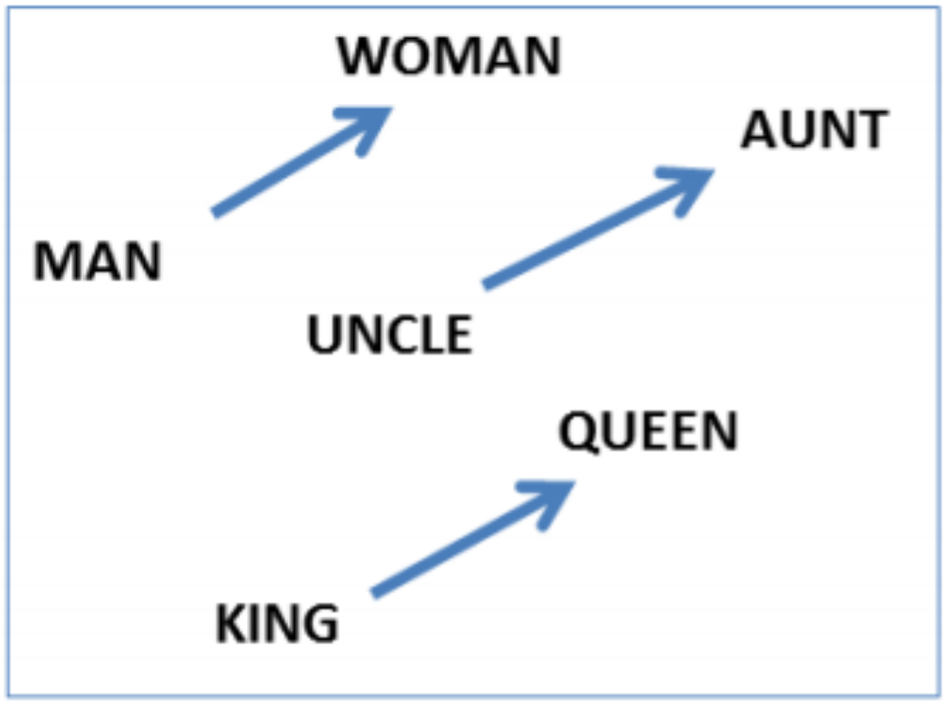


(Mikolov et al., NAACL HLT, 2013)



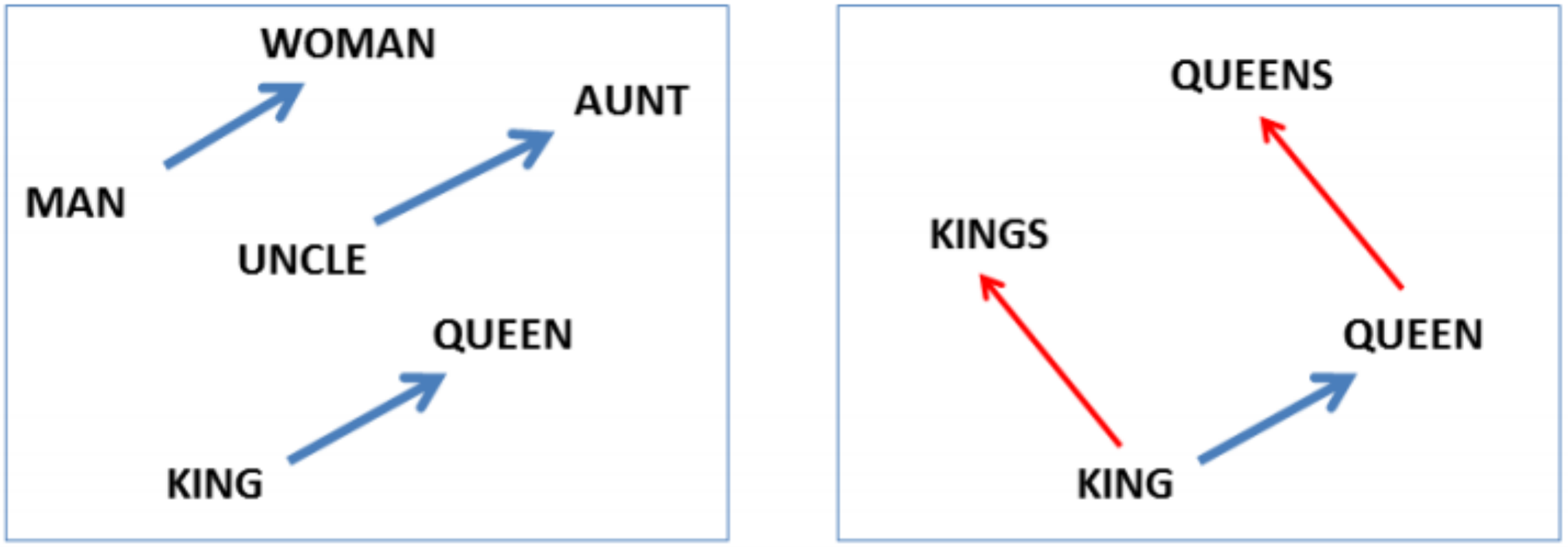
(Mikolov et al., NAACL HLT, 2013)

Vectors are directions in space
Vectors can encode relationships

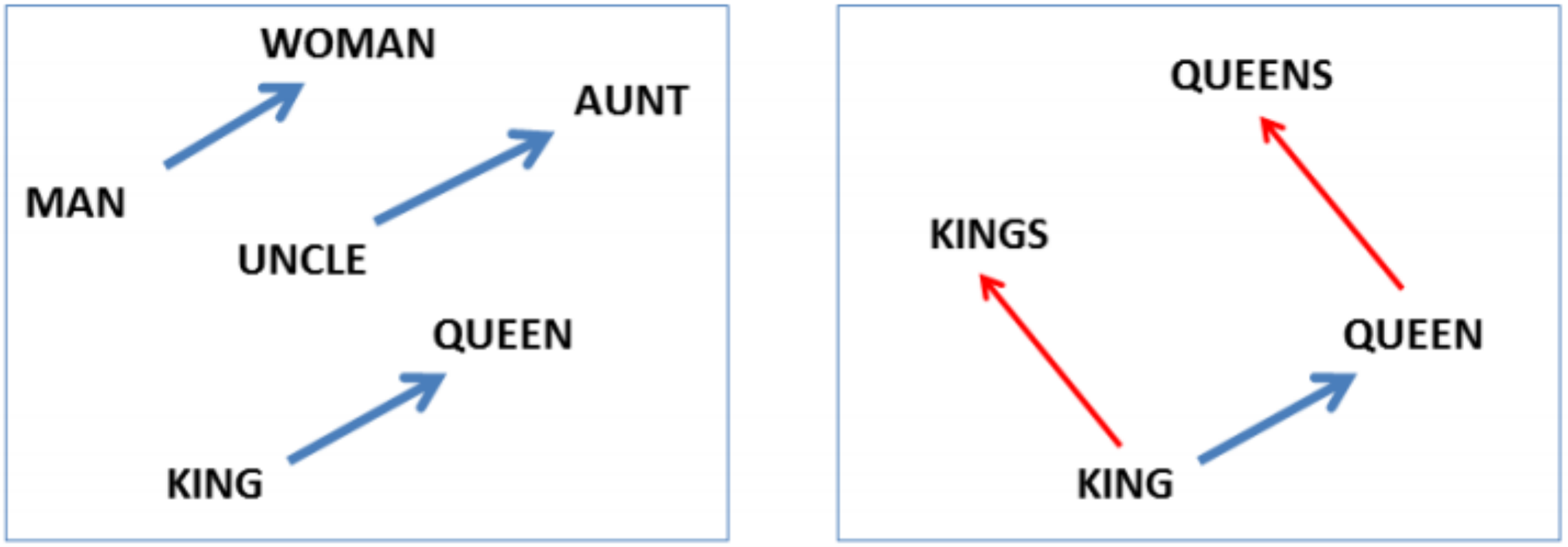


(Mikolov et al., NAACL HLT, 2013)

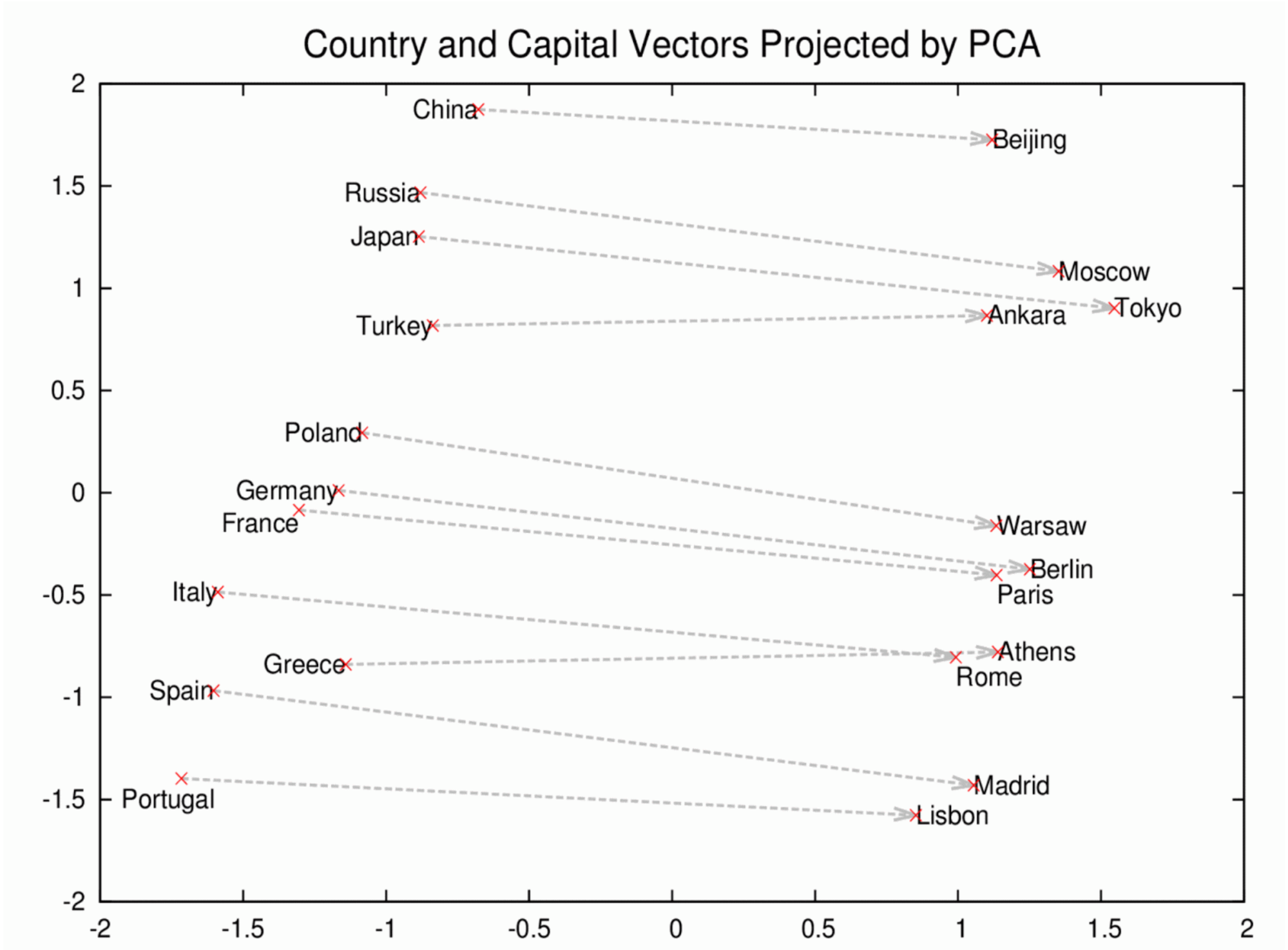
man is to **woman** as **king** is to **?**



(Mikolov et al., NAACL HLT, 2013)



(Mikolov et al., NAACL HLT, 2013)


Country and Capital Vectors Projected by PCA
China
Beijing
Russia
Moscow
Japan
Tokyo
Turkey
Ankara
Poland
Warsaw
Germany
Berlin
France
Paris
Italy
Rome
Greece
Athens
Spain
Madrid
Portugal
Lisbon
2
1.5
1
0.5
0
-0.5
-1
-1.5
-2
-2
-1.5
-1
-0.5
0
0.5
1
1.5
2

Efficient Estimation of Word Representations in Vector Space

Tomas Mikolov
Google Inc., Mountain View, CA
tmikolov@google.com

Kai Chen
Google Inc., Mountain View, CA
kaichen@google.com

Greg Corrado
Google Inc., Mountain View, CA
gcorrado@google.com

Jeffrey Dean
Google Inc., Mountain View, CA
jeff@google.com

Abstract

We propose two novel model architectures for computing continuous vector representations of words from very large data sets. The quality of these representations is measured in a word similarity task, and the results are compared to the previously best performing techniques based on different types of neural networks. We observe large improvements in accuracy at much lower computational cost, i.e. it takes less than a day to learn high quality word vectors from a 1.6 billion words data set. Furthermore, we show that these vectors provide state-of-the-art performance on our test set for measuring syntactic and semantic word similarities.

1 Introduction

Many current NLP systems and techniques treat words as atomic units - there is no notion of similarity between words, as these are represented as indices in a vocabulary. This choice has several good

word2vec

- by Mikolov, Sutskever, Chen, Corrado and Dean at Google

- NAACL 2013

- takes a **text corpus** as **input** and produces the **word vectors** as **output**

$$\textit{linguistics} = \begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Sweden
Similar words

Word	Cosine distance
norway	0.760124
denmark	0.715460
finland	0.620022
switzerland	0.588132
belgium	0.585835
netherlands	0.574631
iceland	0.562368
estonia	0.547621
slovenia	0.531408
floorball_federation	0.529570
luxembourg	0.529477
czech_republic	0.528778
slovakia	0.526340
romania	0.524281
kista	0.522488
helsinki_vantaa	0.519936
swedish	0.519901
balrog_ik	0.514556
portugal	0.502495
russia	0.500196
slovakia_slovenia	0.496051
ukraine	0.495712
chur	0.484225
thailand_togo	0.479172
crimea_ukraine	0.478596
panama_papua	0.477126
latvia	0.477058

Harvard
Similar words

Word	Cosine distance
yale	0.638970
cambridge	0.612665
university	0.597709
faculty	0.588422
harvey_mudd	0.578338
johns_hopkins	0.575645
graduate	0.570294
undergraduate	0.565881
professor	0.563657
mcgill	0.562168
ph_d	0.558665
california_berkeley	0.555539
yale_university	0.550480
harvard_crimson	0.549848
princeton	0.544070
college	0.542838
oxford	0.531948
barnard_college	0.530800
professors	0.529959
princeton_university	0.529763
ucl	0.527395
doctorates	0.526292
doctoral	0.523317
cambridge_massachusetts	0.519657
juris_doctor	0.518845
graduate_student	0.518815
postgraduate	0.515757

word2vec

- **word meaning** and **relationships** between words are **encoded spatially**

- two main learning algorithms in word2vec:
  continuous bag-of-words and
  continuous skip-gram

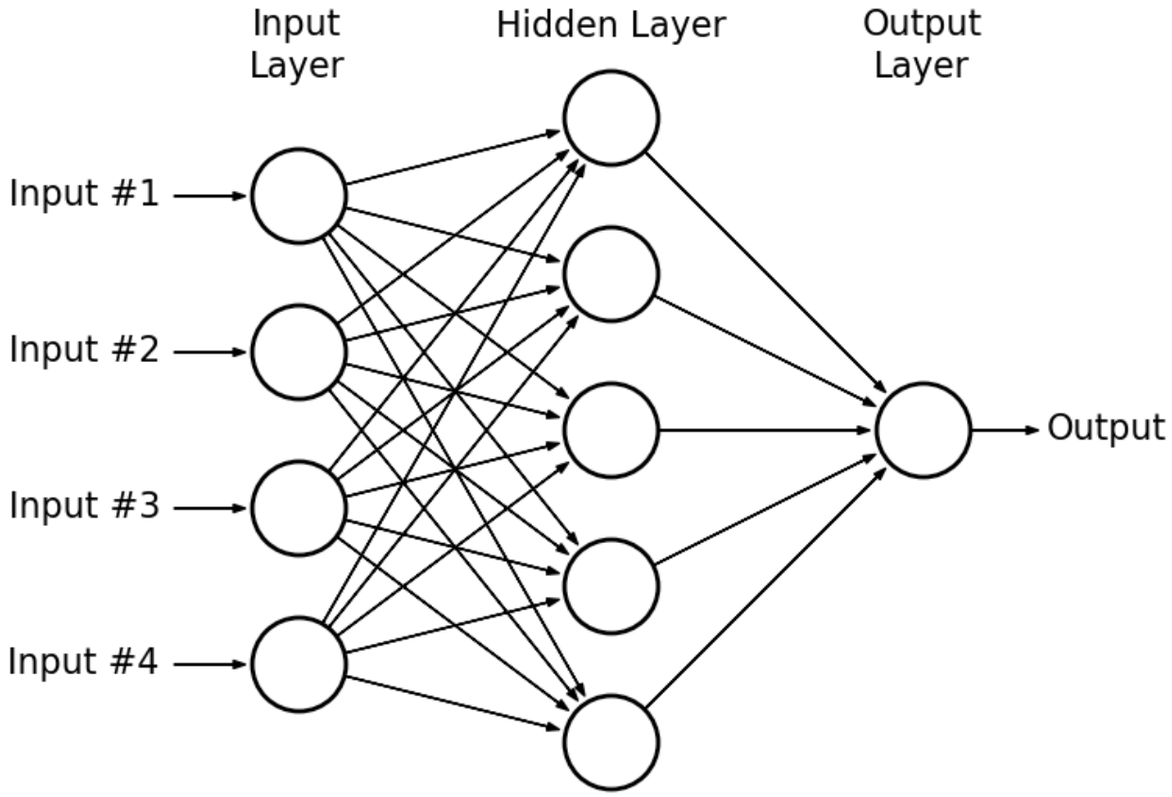
linguistics =

$$\begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$

Goal

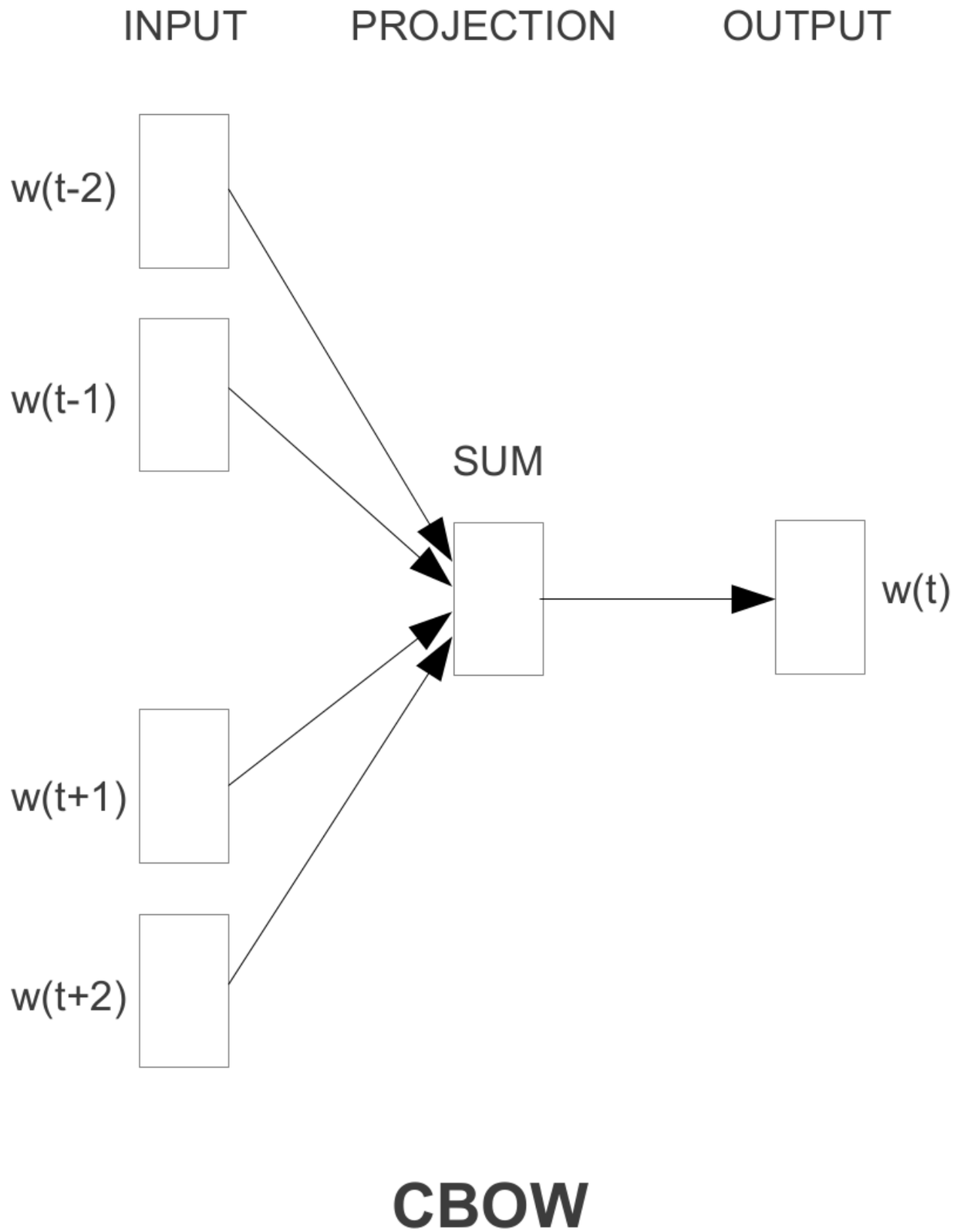
linguistics =

$$\begin{pmatrix} 0.286 \\ 0.792 \\ -0.177 \\ -0.107 \\ 0.109 \\ -0.542 \\ 0.349 \\ 0.271 \end{pmatrix}$$



continuous bag-of-words

- predicting the current word **based on the context**
- **order** of words in the history **does not influence the projection**
- faster & **more appropriate for larger corpora**

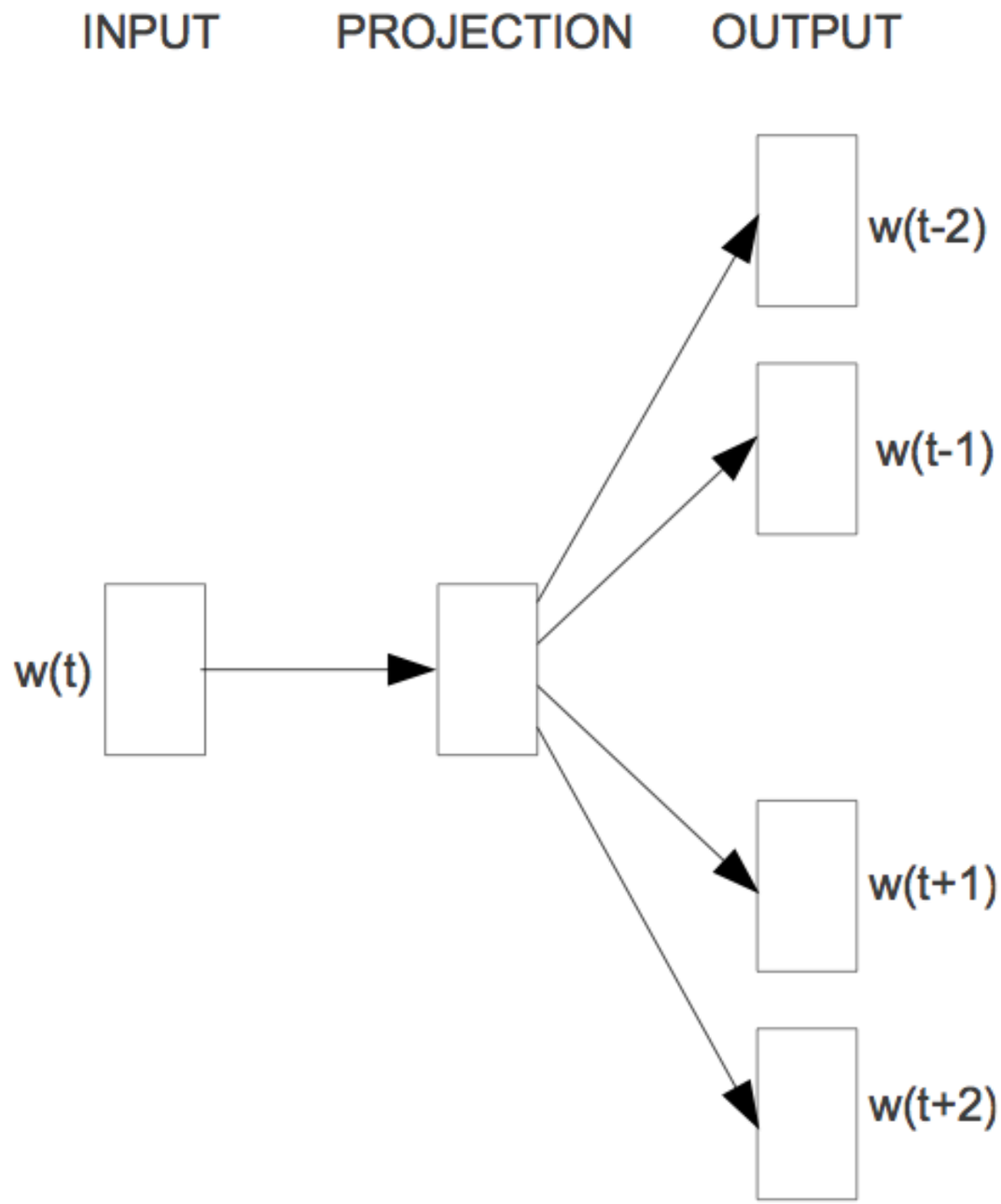


Mikolov et al.

continuous skip-gram

- maximize classification of a word **based on another word in the same sentence**

- better word vectors for **frequent words**, but slower to train



Skip-gram

Mikolov et al.

Why it is awesome

- there is a **fast open-source implementation**

- can be used as **features**
  for **natural language processing tasks** and
  machine learning algorithms

Machine Translation

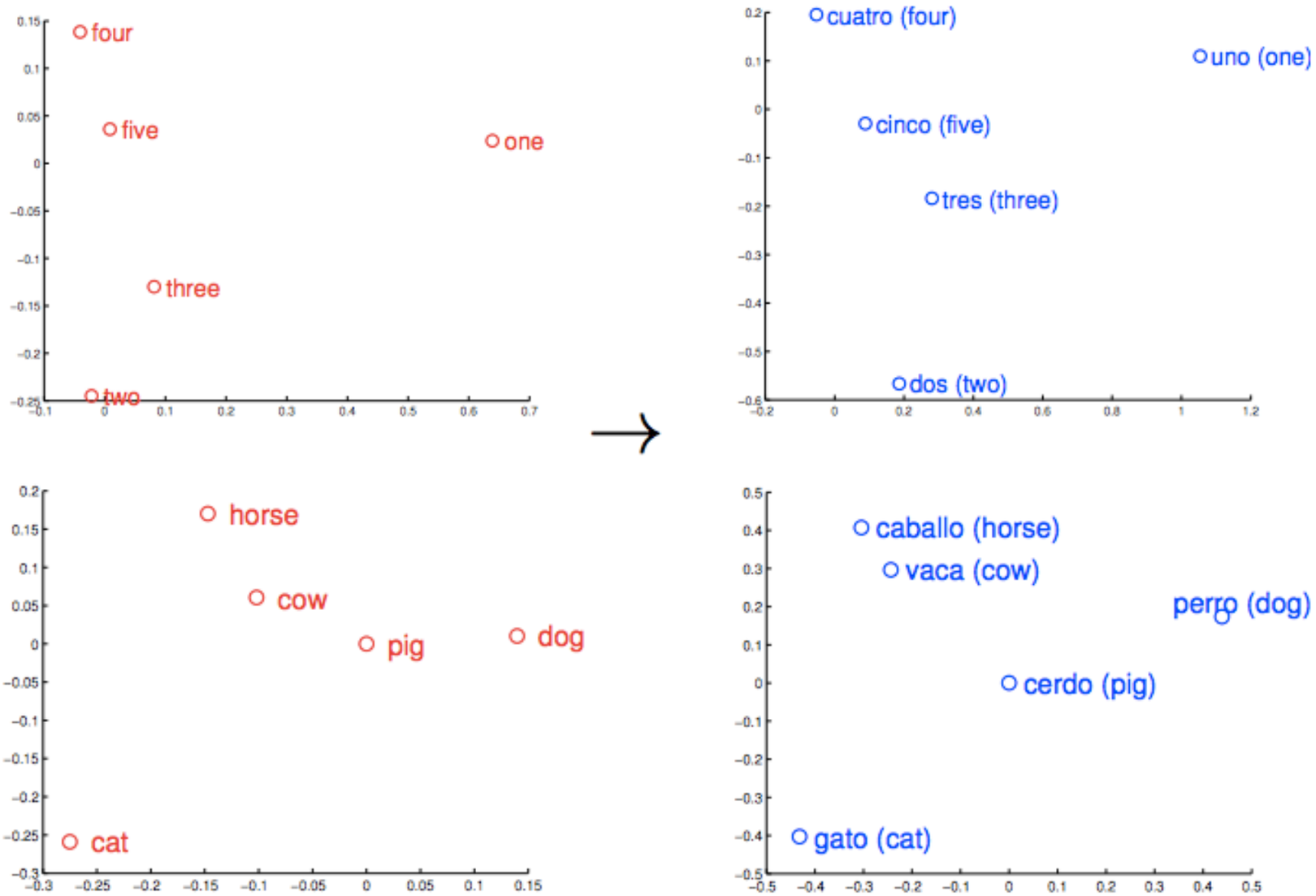
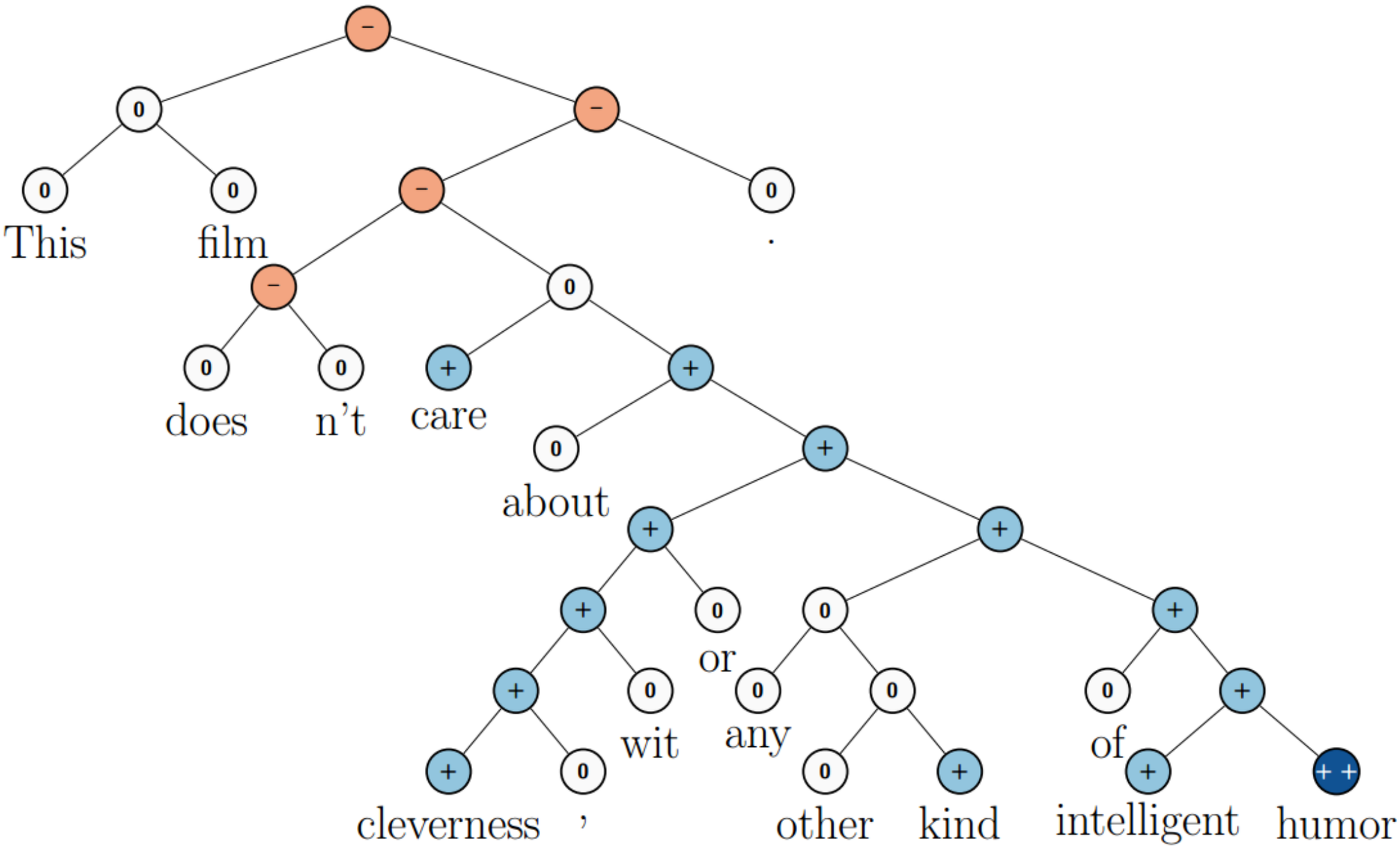


Figure 1: Distributed word vector representations of numbers and animals in English (left) and Spanish (right). The five vectors in each language were projected down to two dimensions using PCA, and then manually rotated to accentuate their similarity. It can be seen that these concepts have similar geometric arrangements in both spaces, suggesting that it is possible to learn an accurate linear mapping from one space to another. This is the key idea behind our method of translation.

Quoted after Mikolov

Sentiment Analysis



Recursive Neural Tensor Network

Quoted after Socher

Image Descriptions

Figure 5. A selection of evaluation results, grouped by human rating.

Quoted after Vinyals

Using word2vec

- Original: http://word2vec.googlecode.com/svn/trunk/
- C++11 version: https://github.com/jdeng/word2vec
- Python: http://radimrehurek.com/gensim/models/word2vec.html
- Java: https://github.com/ansjsun/word2vec_java
- Parallel java: https://github.com/siegfang/word2vec
- CUDAversion: https://github.com/whatupbiatch/cuda-word2vec

Quoted after Wang

Using it in Python

gensim

topic modelling for humans

Download
latest version from the Python Package Index

Direct install with:
easy_install -U gensim

Home | Tutorials | Install | Support | API | About

```
>>> from gensim import corpora, models, similarities
>>>
>>> # Load corpus iterator from a Matrix Market file on disk.
>>> corpus = corpora.MmCorpus('/path/to/corpus.mm')
>>>
>>> # Initialize Latent Semantic Indexing with 200 dimensions.
>>> lsi = models.LsiModel(corpus, num_topics=200)
>>>
>>> # Convert another corpus to the latent space and index it.
>>> index = similarities.MatrixSimilarity(lsi[another_corpus])
>>>
>>> # Compute similarity of a query vs. indexed documents
>>> sims = index[query]
```

Gensim is a FREE Python library

- Scalable statistical semantics
- Analyze plain-text documents for semantic structure
- Retrieve semantically similar documents

Features

Hover your mouse over each feature for more info.

- Scalability
- Platform independent
- Robust
- Open source
- Efficient implementations
- Converters & I/O formats
- Similarity queries
- Support

Usage

```
model = word2vec.Word2Vec.load_word2vec_format( word2vec_vectors_filename, binary=True )

for word in words:
    if word in model:
        print model[ word ]
```

Training a model

```
>>> # import modules & set up logging
>>> import gensim, logging
>>> logging.basicConfig(format='%(asctime)s : %(levelname)s : %(message)s', level=log
>>>
>>> sentences = [['first', 'sentence'], ['second', 'sentence']]
>>> # train word2vec on the two sentences
>>> model = gensim.models.Word2Vec(sentences, min_count=1)
```

Quoted after Řehůřek

Training a model with iterator

```
>>> class MySentences(object):
...     def __init__(self, dirname):
...         self.dirname = dirname
...
...     def __iter__(self):
...         for fname in os.listdir(self.dirname):
...             for line in open(os.path.join(self.dirname, fname)):
...                 yield line.split()
>>>
>>> sentences = MySentences('/some/directory') # a memory-friendly iterator
>>> model = gensim.models.Word2Vec(sentences)
```

Quoted after Řehůřek

Doing it in C

- Download the code:
  git clone **https://github.com/h10r/word2vec-macosx-maverics.git**

- Run 'make' to compile word2vec tool

- Run the demo scripts:

  ./demo-word.sh

  and

  ./demo-phrases.sh

Doing it in C

- Download the code:
  git clone **https://github.com/h10r/word2vec-macosx-maverics.git**

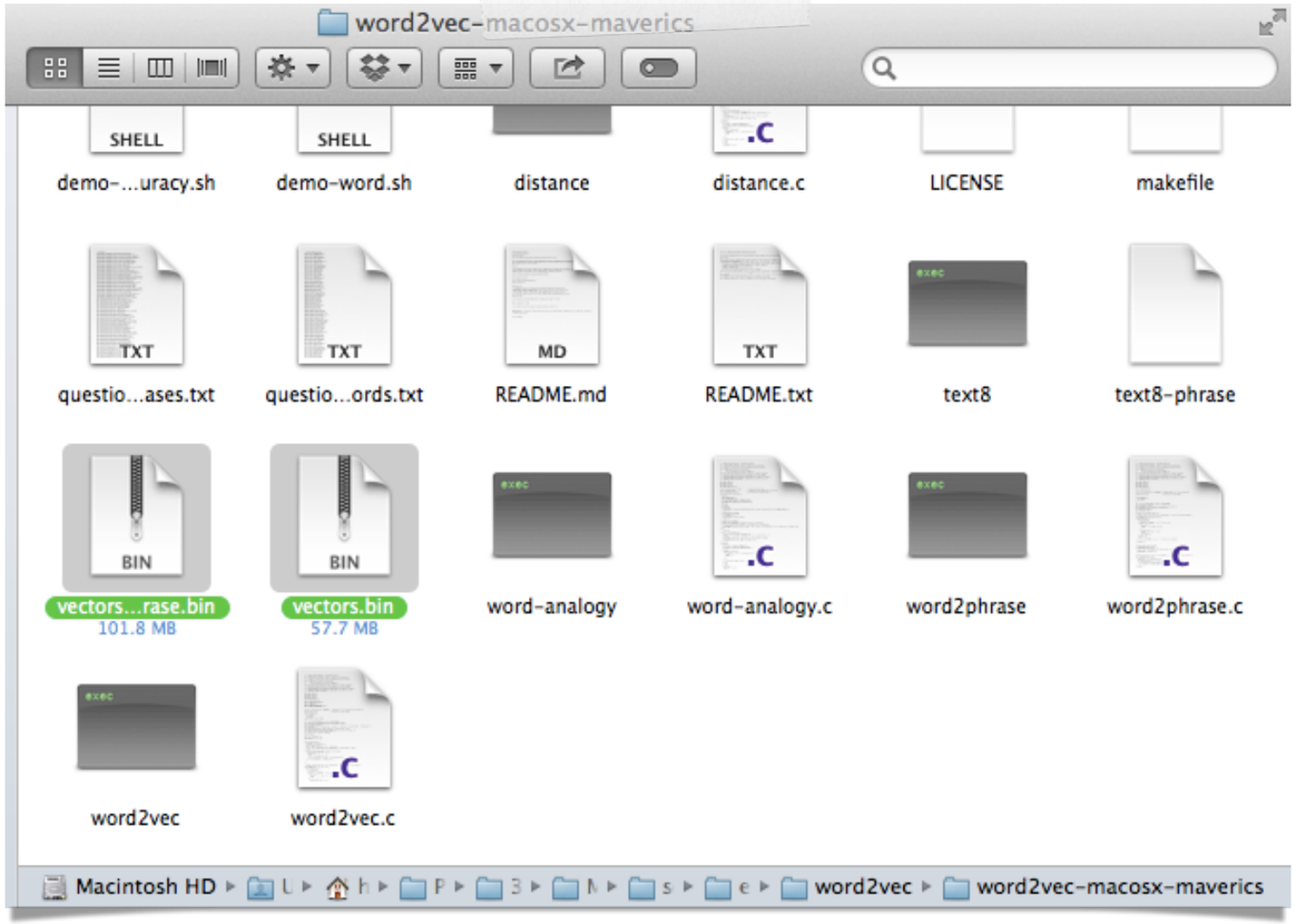
- Run 'make' to compile word2vec tool

- Run the demo scripts:

  ./demo-word.sh

  and

  ./demo-phrases.sh



Pre-trained entity vectors with Freebase naming

We are also offering more than 1.4M pre-trained entity vectors with naming from Freebase. This is especially helpful for projects related to knowledge mining.

- Entity vectors trained on 100B words from various news articles: freebase-vectors-skipgram1000.bin.gz
- Entity vectors trained on 100B words from various news articles, using the deprecated /en/ naming (more easily readable); the vectors are sorted by frequency: freebase-vectors-skipgram1000-en.bin.gz

Here is an example output of *./distance freebase-vectors-skipgram1000-en.bin*:

```
Enter word or sentence (EXIT to break): /en/geoffrey_hinton

                          Word       Cosine distance
------------------------------------------------------------
              /en/marvin_minsky                 0.457204
                /en/paul_corkum                 0.443342
 /en/william_richard_peltier                    0.432396
              /en/brenda_milner                 0.430886
    /en/john_charles_polanyi                    0.419538
             /en/leslie_valiant                 0.416399
```

Testing a model

Google have released their testing set of about 20,000 syntactic and semantic test examples, following the “A is to B as C is to D” task: http://word2vec.googlecode.com/svn/trunk/questions-words.txt.

Gensim support the same evaluation set, in exactly the same format:

```
>>> model.accuracy('/tmp/questions-words.txt')
2014-02-01 22:14:28,387 : INFO : family: 88.9% (304/342)
2014-02-01 22:29:24,006 : INFO : gram1-adjective-to-adverb: 32.4% (263/812)
2014-02-01 22:36:26,528 : INFO : gram2-opposite: 50.3% (191/380)
2014-02-01 23:00:52,406 : INFO : gram3-comparative: 91.7% (1222/1332)
2014-02-01 23:13:48,243 : INFO : gram4-superlative: 87.9% (617/702)
2014-02-01 23:29:52,268 : INFO : gram5-present-participle: 79.4% (691/870)
2014-02-01 23:57:04,965 : INFO : gram7-past-tense: 67.1% (995/1482)
2014-02-02 00:15:18,525 : INFO : gram8-plural: 89.6% (889/992)
2014-02-02 00:28:18,140 : INFO : gram9-plural-verbs: 68.7% (482/702)
2014-02-02 00:28:18,140 : INFO : total: 74.3% (5654/7614)

```

Quoted after Řehůřek

nutrition

neurological addiction

disorders obesity diet

respond disorder

pathological craving alcohol

purchase

employees marketing

profits

corporate

reporting

advertising

bulletin consumers loyalty

customer

email

laboratory

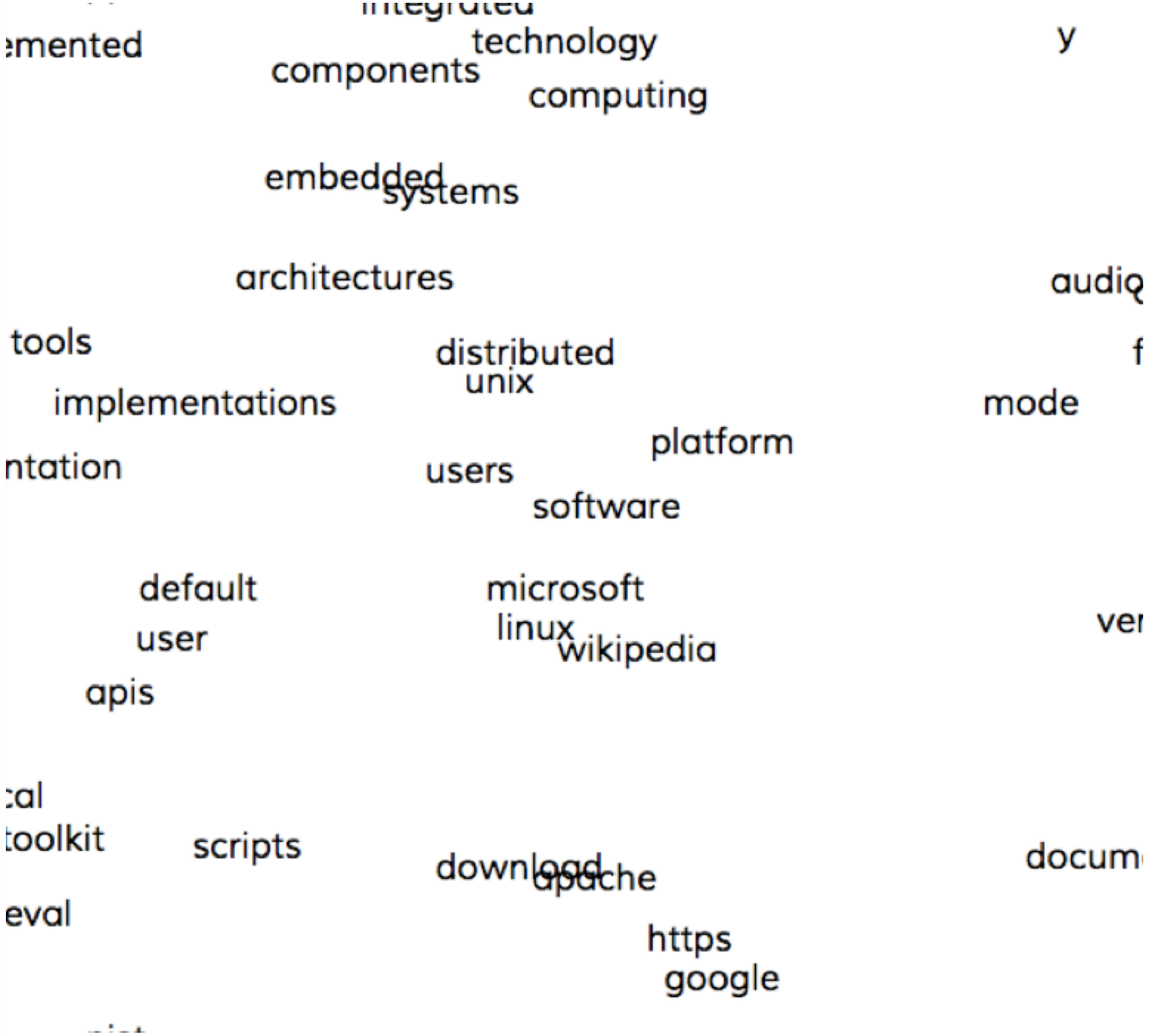
experimental

experiments

experiment

studies

researchers scientists


integrated
emented
technology
y
components
computing
embedded
systems
architectures
audio
tools
distributed
f
unix
implementations
mode
platform
ntation
users
software
default
microsoft
user
linux
ver
wikipedia
apis
cal
toolkit
scripts
docum
download
apache
eval
https
google

localhost:8000/compare.h ×

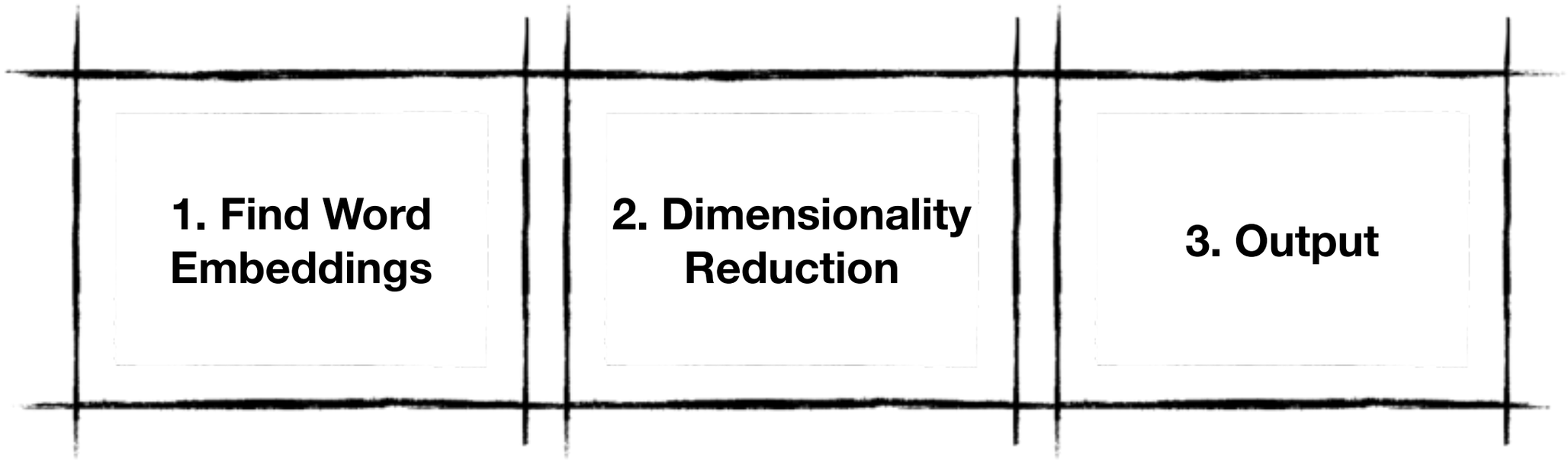
localhost:8000/compare.html

Stockholm NLP Meetup

Hendrik Heuer

word2vec & t-SNE

+ Zoom in | - Zoom out | Toggle - Topics A | Toggle - Topics B | Toggle - Topics Intersection



word2vec t-SNE JSON

1. Find Word Embeddings	2. Dimensionality Reduction	3. Output
word2vec	t-SNE	JSON

```
def parse_file_with_gensim( filename ):
    model = word2vec.Word2Vec.load_word2vec_format( "data/vectors.bin", binary=True )

    frequent_words = load_list_of_frequent_words()

    keep_track_of_words = set()

    word_vectors = []
    word_labels = []

    f = codecs.open( filename, encoding='utf-8' )
    for line in f:
        line = line.lower()
        line = " ".join("".join([" " if ch in string.punctuation else ch for ch in line]).split())

        words = nltk.word_tokenize( line )

        for word in words:
            if word not in keep_track_of_words:
                if not word in frequent_words:
                    if word in model:
                        word_vectors.append( model[ word ] )
                        word_labels.append( word )

                        keep_track_of_words.add( word )

    return word_vectors, word_labels
```

```
def parse_file_with_gensim( filename ):
    model = word2vec.Word2Vec.load_word2vec_format( "data/vectors.bin", binary=True )

    frequent_words = load_list_of_frequent_words()

    keep_track_of_words = set()

    word_vectors = []
    word_labels = []

    f = codecs.open( filename, encoding='utf-8' )
    for line in f:
        line = line.lower()
        line = " ".join("".join([" " if ch in string.punctuation else ch for ch in line]).split())

        words = nltk.word_tokenize( line )

        for word in words:
            if word not in keep_track_of_words:
                if not word in frequent_words:
                    if word in model:
                        word_vectors.append( model[ word ] )
                        word_labels.append( word )

                        keep_track_of_words.add( word )

    return word_vectors, word_labels
```

1. Find Word Embeddings

2. Dimensionality Reduction

3. Output

word2vec

t-SNE

JSON

Journal of Machine Learning Research 1 (2008) 1-48 Submitted 4/00; Published 10/00

Visualizing Data using t-SNE

Laurens van der Maaten L.VANDERMAATEN@MICC.UNIMAAS.NL
MICC-IKAT
Maastricht University
P.O. Box 616, 6200 MD Maastricht, The Netherlands

Geoffrey Hinton HINTON@CS.TORONTO.EDU
Department of Computer Science
University of Toronto
6 King's College Road, M5S 3G4 Toronto, ON, Canada

Editor: Leslie Pack Kaelbling

Abstract

We present a new technique called "t-SNE" that visualizes high-dimensional data by giving each datapoint a location in a two or three-dimensional map. The technique is a variation of Stochastic Neighbor Embedding (Hinton and Roweis, 2002) that is much easier to optimize, and produces significantly better visualizations by reducing the tendency to crowd points together in the center of the map. t-SNE is better than existing techniques at creating a single map that reveals structure at many different scales. This is particularly important for high-dimensional data that lie on several different, but related, low-dimensional manifolds, such as images of objects from multiple classes seen from multiple viewpoints. For visualizing the structure of very large datasets, we show how t-SNE can use random walks on neighborhood graphs to allow the implicit structure of all of the data to influence the way in which a subset of the data is displayed. We illustrate the performance

sklearn.manifold.TSNE

class `sklearn.manifold.`**`TSNE`**(*n_components=2, perplexity=30.0, early_exaggeration=4.0, learning_rate=1000.0, n_iter=1000, metric='euclidean', init='random', verbose=0, random_state=None*)

t-distributed Stochastic Neighbor Embedding.

t-SNE [1] is a tool to visualize high-dimensional data. It converts similarities between data points to joint probabilities and tries to minimize the Kullback-Leibler divergence between the joint probabilities of the low-dimensional embedding and the high-dimensional data. t-SNE has a cost function that is not convex, i.e. with different initializations we can get different results.

It is highly recommended to use another dimensionality reduction method (e.g. PCA for dense data or TruncatedSVD for sparse data) to reduce the number of dimensions to a reasonable amount (e.g. 50) if the number of features is very high. This will suppress some noise and speed up the computation of pairwise distances between samples. For more tips see Laurens van der Maaten's FAQ [2].

Parameters:

n_components : int, optional (default: 2)

Dimension of the embedded space.

perplexity : float, optional (default: 30)

The perplexity is related to the number of nearest neighbors that is used in other manifold learning algorithms. Larger datasets usually require a larger perplexity. Consider selcting

s quite

Should be in Macports
py27-scikit-learn @0.15.2 (python, science)

```
def do_tsne_on_vectors( word_vectors ):
    vectors = np.asfarray( word_vectors, dtype='float' )

    tsne = TSNE(n_components=2, random_state=42)
    return tsne.fit_transform( vectors )
```

1. Find Word Embeddings	2. Dimensionality Reduction	**3. Output**
word2vec	t-SNE	JSON

```
def render_json( vectors, labels ):
    output_for_json = []

    vectors = np.array( vectors )

    for i in range( len( vectors ) ):
        new_hash = {}

        new_hash["title"] = str(labels[ i ])
        new_hash["x"] = int(vectors[ i ][ 0 ])
        new_hash["y"] = int(vectors[ i ][ 1 ])

        output_for_json.append( new_hash )

    print json.dumps( output_for_json )
```

python — bash — 132×34

bash | bash

```
Last login: Sun Mar 22 09:53:41 on console
n145-p213:~ hendrikheuer$ cd Desktop/word2vec_demo/
n145-p213:word2vec_demo hendrikheuer$ ls
html                            kafka_metamorphosis.txt
kafka_before_the_law.txt        python
n145-p213:word2vec_demo hendrikheuer$ cd python/
n145-p213:python hendrikheuer$ ls
data    main.py
n145-p213:python hendrikheuer$ python main.py
[{"y": 2843, "x": -2503, "title": "franz"}, {"y": -298, "x": -2276, "title": "kafka"}, {"y": 2296, "x": 3441, "title": "before"}, {"
y": 2354, "x": 2718, "title": "the"}, {"y": 1652, "x": -312, "title": "law"}, {"y": -1005, "x": -920, "title": "sits"}, {"y": -3054,
 "x": -1026, "title": "a"}, {"y": 3522, "x": 63, "title": "gatekeeper"}, {"y": -268, "x": 548, "title": "to"}, {"y": 70, "x": 2267,
"title": "this"}, {"y": 85, "x": 8, "title": "comes"}, {"y": -936, "x": 3024, "title": "man"}, {"y": 1742, "x": -594, "title": "from
"}, {"y": -2284, "x": -1748, "title": "country"}, {"y": 2200, "x": 1794, "title": "who"}, {"y": -1851, "x": 1055, "title": "asks"},
{"y": -3701, "x": 831, "title": "gain"}, {"y": 1007, "x": 1876, "title": "entry"}, {"y": -2424, "x": 1749, "title": "into"}, {"y": -
2808, "x": 794, "title": "but"}, {"y": 2327, "x": -4408, "title": "says"}, {"y": 1049, "x": -121, "title": "that"}, {"y": 1504, "x":
 790, "title": "he"}, {"y": 1122, "x": 47, "title": "can"}, {"y": -1315, "x": 2109, "title": "not"}, {"y": 644, "x": 1454, "title":
"grant"}, {"y": 626, "x": 1307, "title": "him"}, {"y": -744, "x": 1846, "title": "at"}, {"y": -1200, "x": -1103, "title": "moment"},
 {"y": 164, "x": -3534, "title": "thinks"}, {"y": -2132, "x": 1702, "title": "about"}, {"y": -1773, "x": -811, "title": "it"}, {"y":
 -650, "x": 2471, "title": "and"}, {"y": 2759, "x": 414, "title": "then"}, {"y": -325, "x": 2099, "title": "if"}, {"y": 3451, "x": 2
26, "title": "will"}, {"y": -1399, "x": 344, "title": "be"}, {"y": -2626, "x": -1122, "title": "allowed"}, {"y": -2158, "x": 925, "t
itle": "come"}, {"y": -1321, "x": 2718, "title": "in"}, {"y": -776, "x": 1422, "title": "sometime"}, {"y": -1411, "x": 1603, "title"
: "later"}, {"y": -552, "x": 1620, "title": "on"}, {"y": -1827, "x": 2222, "title": "is"}, {"y": 3054, "x": 2766, "title": "possible
"}, {"y": -2603, "x": -329, "title": "now"}, {"y": -659, "x": 280, "title": "gate"}, {"y": -147, "x": -898, "title": "stands"}, {"y"
: -2500, "x": 1308, "title": "open"}, {"y": 340, "x": 679, "title": "as"}, {"y": 896, "x": 2276, "title": "always"}, {"y": 1451, "x"
: -646, "title": "walks"}, {"y": -2868, "x": -369, "title": "side"}, {"y": 3440, "x": 1367, "title": "so"}, {"y": 794, "x": 606, "ti
tle": "bends"}, {"y": -2368, "x": -90, "title": "over"}, {"y": -270, "x": 1807, "title": "order"}, {"y": 2914, "x": -33, "title": "s
ee"}, {"y": -1854, "x": 1427, "title": "through"}, {"y": -2013, "x": 2158, "title": "inside"}, {"y": -1247, "x": -640, "title": "whe
n"}, {"y": 196, "x": -2492, "title": "notices"}, {"y": 80, "x": -1434, "title": "laughs"}, {"y": 243, "x": -1867, "title": "you"}, {
"y": 2672, "x": 161, "title": "much"}, {"y": -466, "x": 331, "title": "try"}, {"y": 56, "x": 711, "title": "going"}, {"y": 1846, "x"
: 2343, "title": "spite"}, {"y": 1654, "x": 876, "title": "of"}, {"y": -665, "x": -1047, "title": "my"}, {"y": 359, "x": 1741, "titl
e": "prohibition"}, {"y": -2933, "x": 1245, "title": "take"}, {"y": 2003, "x": -173, "title": "note"}, {"y": -2517, "x": 666, "title
": "i"}, {"y": 2847, "x": 434, "title": "am"}, {"y": -62, "x": -2787, "title": "powerful"}, {"y": -2179, "x": 635, "title": "only"},
 {"y": 3185, "x": 1373, "title": "most"}, {"y": 914, "x": -2616, "title": "lowly"}, {"y": -901, "x": -2994, "title": "room"}, {"y":
```

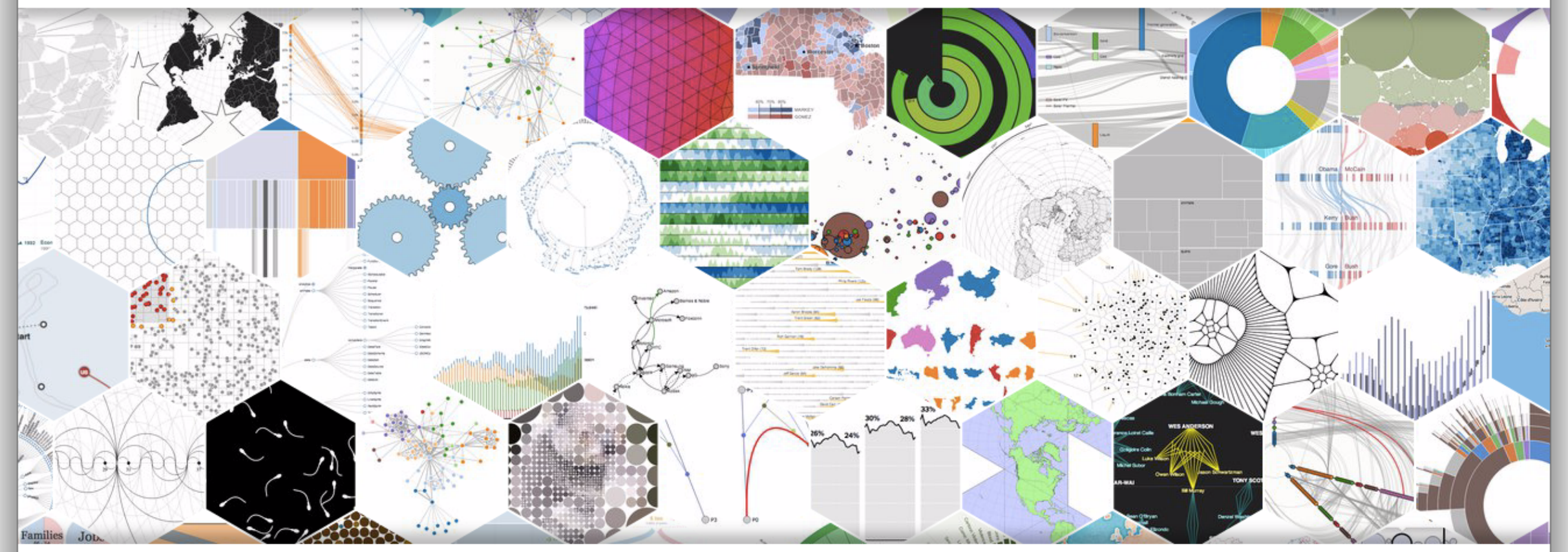
D3.js – Data-Driven Docu

d3js.org

Overview Examples Documentation Source

Fork me on GitHub

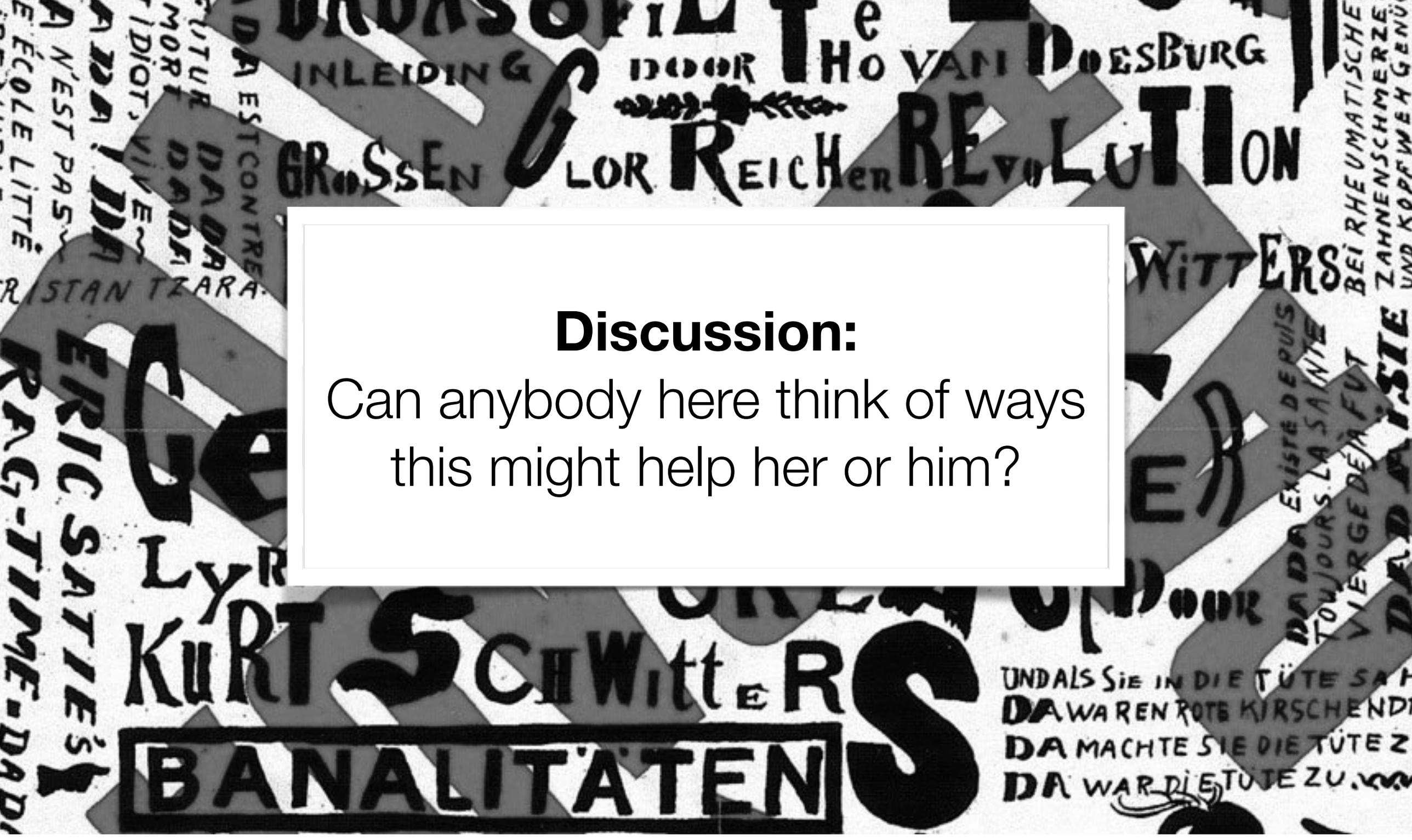
Data-Driven Documents



D3.js is a JavaScript library for manipulating documents based on data. **D3** helps you bring data to life using HTML, SVG, and CSS. D3's emphasis on web standards gives you the full capabilities of modern browsers without tying yourself to a proprietary framework, combining powerful visualization components and a data-driven approach to DOM manipulation.

See more examples.

Target Word	BoW5	BoW2	Deps
batman	nightwing	superman	superman
	aquaman	superboy	superboy
	catwoman	aquaman	supergirl
	superman	catwoman	catwoman
	manhunter	batgirl	aquaman
hogwarts	dumbledore	evernight	sunnydale
	hallows	sunnydale	collinwood
	half-blood	garderobe	calarts
	malfoy	blandings	greendale
	snape	collinwood	millfield
turing	nondeterministic	non-deterministic	pauling
	non-deterministic	finite-state	hotelling
	computability	nondeterministic	heting
	deterministic	buchi	lessing
	finite-state	primality	hamming
florida	gainesville	fla	texas
	fla	alabama	louisiana
	jacksonville	gainesville	georgia
	tampa	tallahassee	california
	lauderdale	texas	carolina
object-oriented	aspect-oriented	aspect-oriented	event-driven
	smalltalk	event-driven	domain-specific
	event-driven	objective-c	rule-based
	prolog	dataflow	data-driven
	domain-specific	4gl	human-centered
dancing	singing	singing	singing
	dance	dance	rapping
	dances	dances	breakdancing
	dancers	breakdancing	miming
	tap-dancing	clowning	busking



word2vec

From theory to practice

Hendrik Heuer
Stockholm NLP Meetup

Further Reading

- Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient Estimation of Word Representations in Vector Space. In Proceedings of Workshop at ICLR, 2013.

- Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed Representations of Words and Phrases and their Compositionality. In Proceedings of NIPS, 2013.

- Tomas Mikolov, Wen-tau Yih,and Geoffrey Zweig. Linguistic Regularities in Continuous Space Word Representations. In Proceedings of NAACL HLT, 2013.

Further Reading

- Richard Socher - Deep Learning for NLP (without Magic) http://lxmls.it.pt/2014/?page_id=5

- Wang - Introduction to Word2vec and its application to find predominant word senses http://compling.hss.ntu.edu.sg/courses/hg7017/pdf/word2vec%20and%20its%20application%20to%20wsd.pdf

- Exploiting Similarities among Languages for Machine Translation, Tomas Mikolov, Quoc V. Le, Ilya Sutskever, http://arxiv.org/abs/1309.4168

- Title Image by Hans Arp

Further Coding

- word2vec
  https://code.google.com/p/word2vec/

- word2vec for MacOSX Maverics
  https://github.com/h10r/word2vec-macosx-maverics

- Gensim Python Library
  https://radimrehurek.com/gensim/index.html

- Gensim Tutorials
  https://radimrehurek.com/gensim/tutorial.html

- Scikit-Learn TSNE
  http://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

word2vec
From theory to practice

Hendrik Heuer
Stockholm NLP Meetup