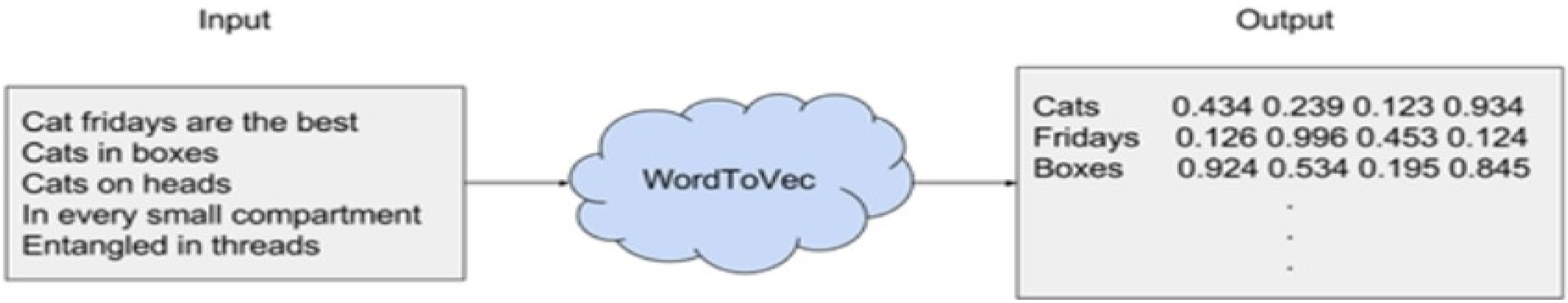
Word2vec and tSNE

https://colab.research.google.com/drive/1HJpR8BZh1NK1ugILXgLuWyohQKdx8NZz?usp=sharing

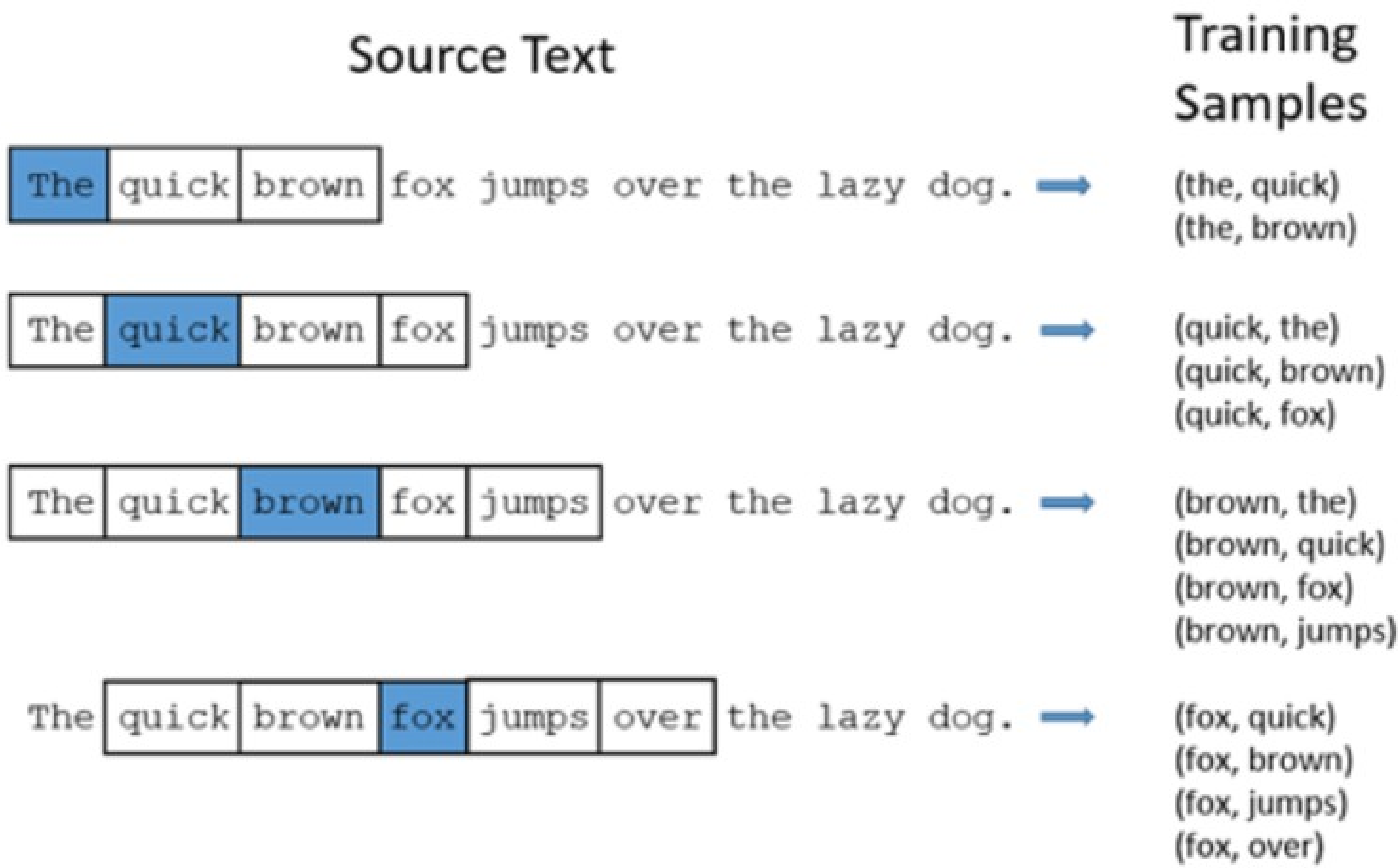
Word2vec

Word2vec can represent words by vectors. These vectors capture contextual sequential information about a word.

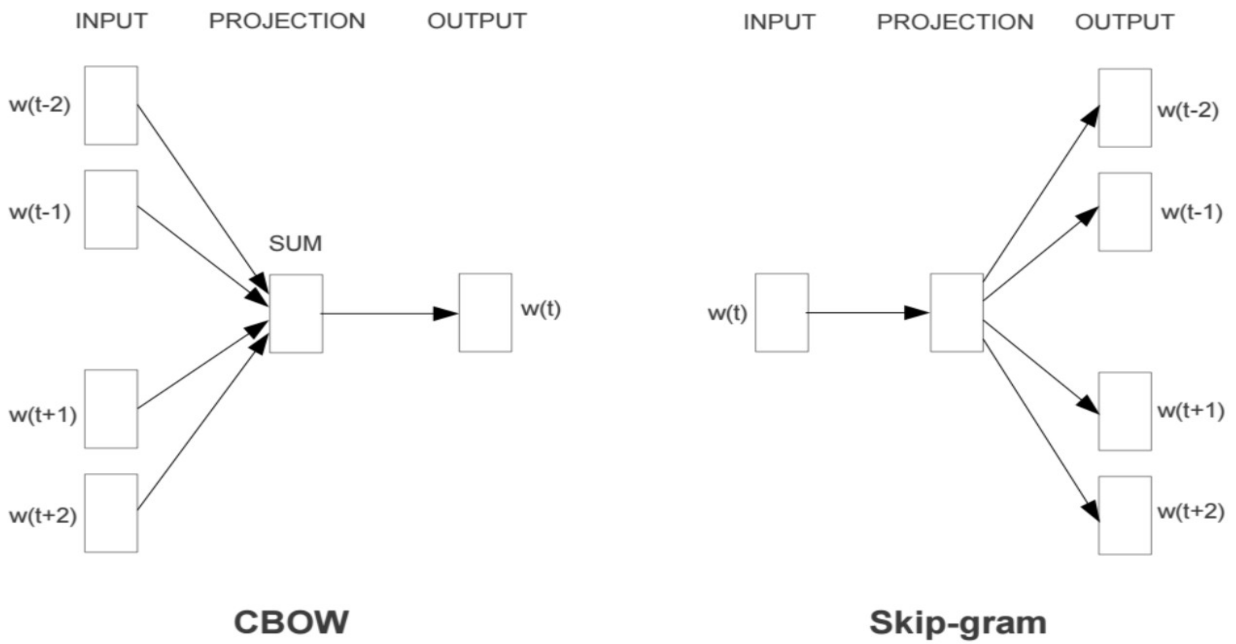
Word2vec tutorial: https://medium.com/@zafaralibagh6/a-simple-word2vec-tutorial-61e64e38a6a1



Context and windows

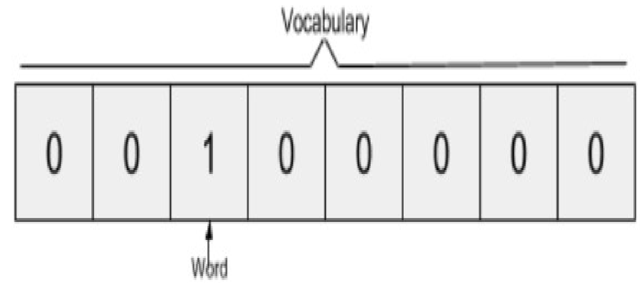


Skip-gram and CBOW models



CBOW (Continuous Bag Of Words): using context to predict current word
Skip-gram: using word to predict current context

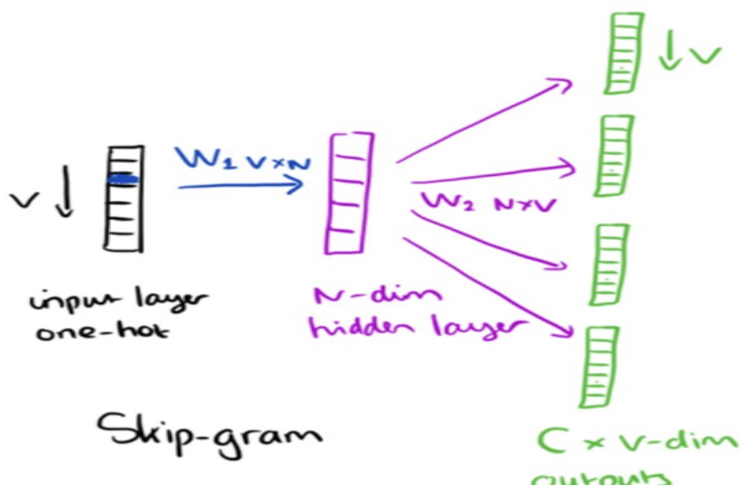
word2vec vector

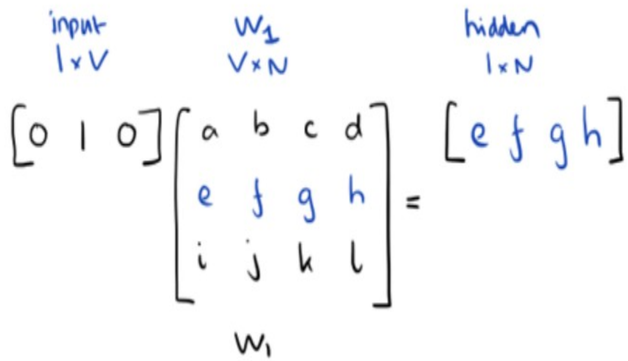


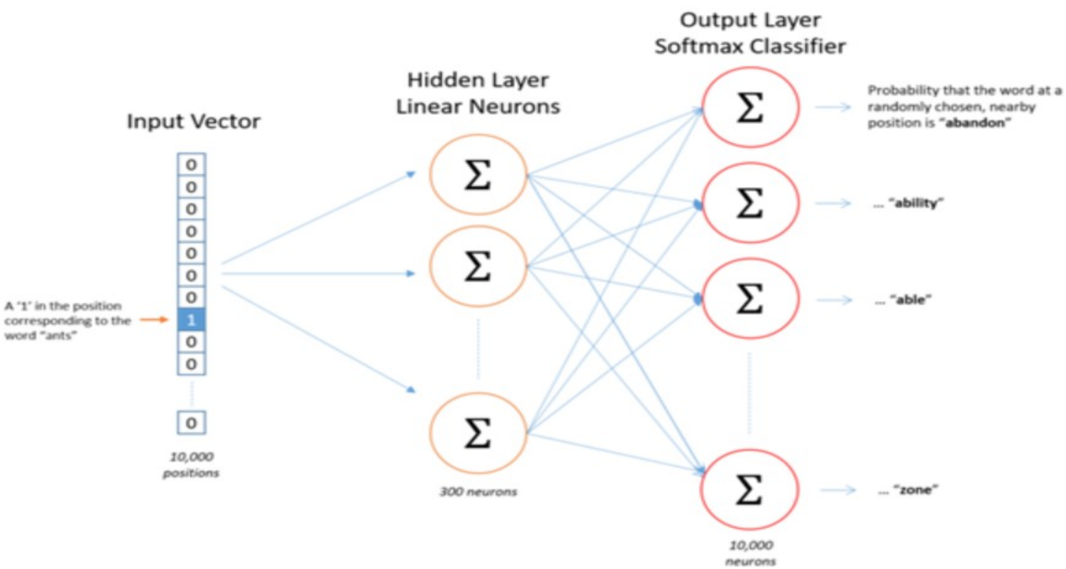
The output of word2vec is a single vector (also with 10,000 components) containing, for every word in our vocabulary, the probability that a randomly selected nearby word is that vocabulary word.

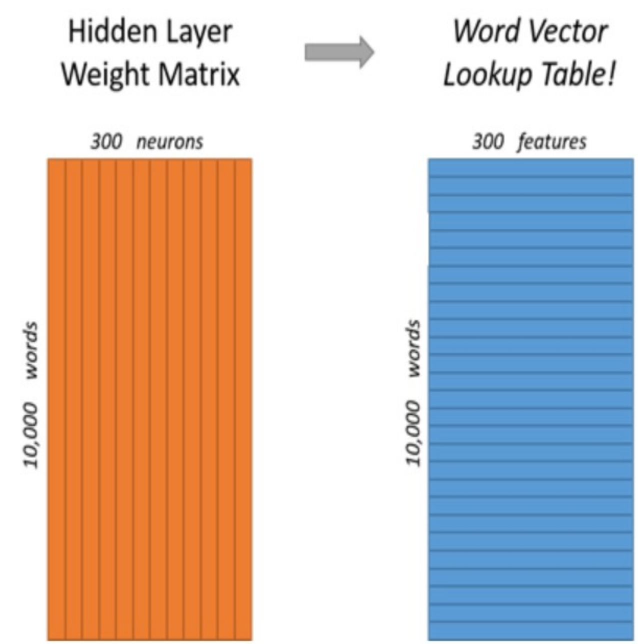
Hypothetical label – one word in blue

Skip-gram


V
W1 V×N
W2 N×V
V
input layer
one-hot
N-dim
hidden layer
Skip-gram
C × V-dim

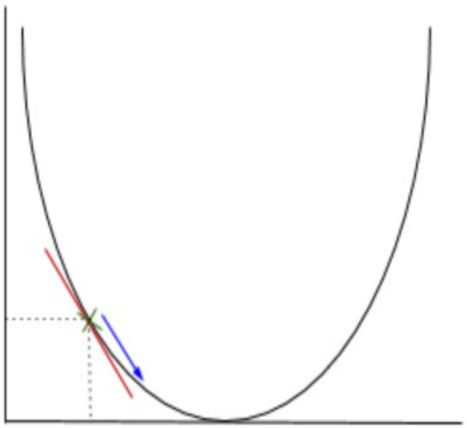

input
1×V
W1
V×N
hidden
1×N
[0 1 0]
a b c d
e f g h
i j k l
=
[e f g h]
W1


Input Vector
Hidden Layer
Linear Neurons
Output Layer
Softmax Classifier
A '1' in the position corresponding to the word "ants"
10,000 positions
300 neurons
10,000 neurons
Probability that the word at a randomly chosen, nearby position is "abandon"
... "ability"
... "able"
... "zone"


Hidden Layer
Weight Matrix
Word Vector
Lookup Table!
300 neurons
300 features
10,000 words
10,000 words

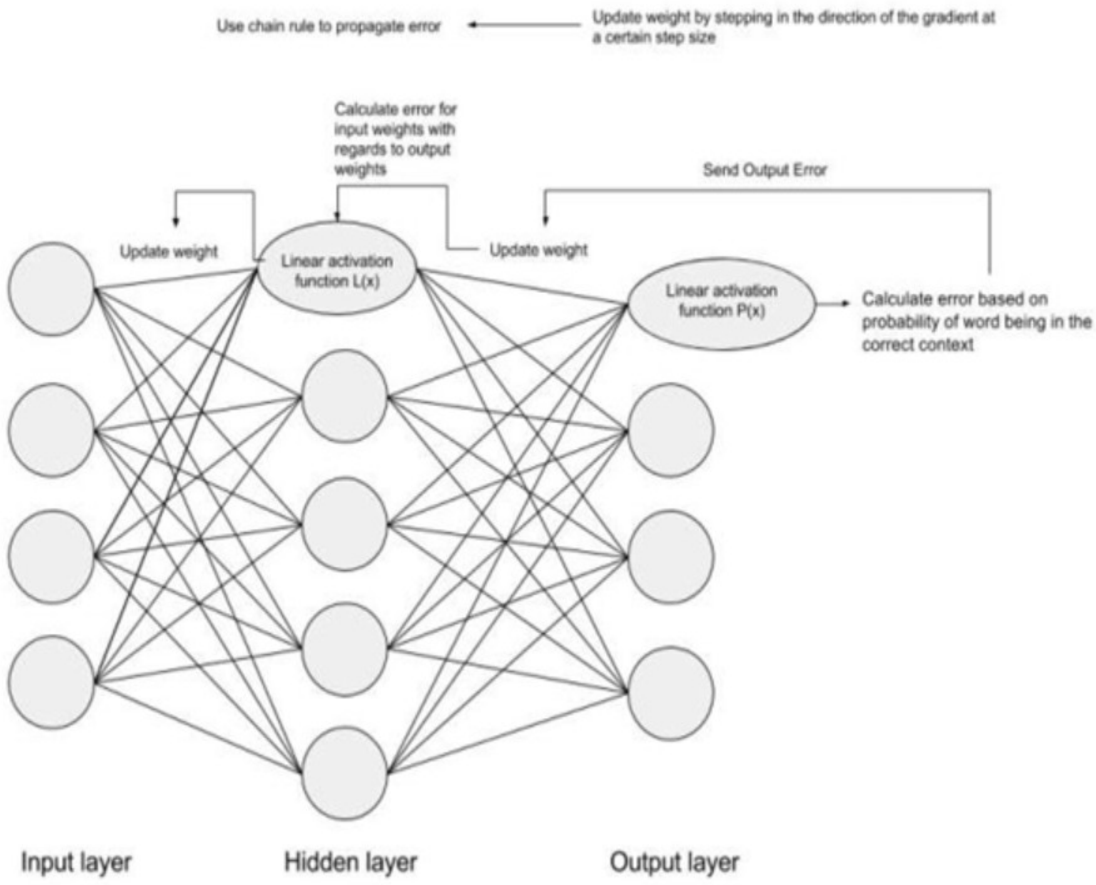
Gradient Descent

- we apply stochastic gradient descent to change the values of the weights in order to get a more desirable value for the probability calculated
- In gradient descent we need to calculate the gradient of the function being optimised at the point representing the weight that we are changing. The gradient is then used to choose the direction in which to make a step to move towards the local optimum



Backpropagation

The next step is using Backpropagation, to adjust the weights between multiple layers. The error that is computed at the end of the output layer is passed back from the output layer to the hidden layer by applying the Chain Rule. Gradient descent is used to update the weights between these two layers. The error is then adjusted at each layer and sent back further.



Download pre-trained word2vec

```
!pip install gensim
#!wget -c "http://evexdb.org/pmresources/vec-space-models/PubMed-and-PMC-ri.tar.gz" # this will take some time,
!wget -c "https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz"
import gensim
%matplotlib notebook
import numpy as np
import matplotlib.pyplot as plt
from sklearn.manifold import TSNE

# load pre-trained word2vec embeddings
#model = gensim.models.KeyedVectors.load_word2vec_format('PubMed-and-PMC-ri.tar.gz', binary=True)
model = gensim.models.KeyedVectors.load_word2vec_format('GoogleNews-vectors-negative300.bin.gz', binary=True)
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.6/dist-packages (3.6.0)
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.6/dist-packages (from gensim) (1.4.1)
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.6/dist-packages (from gensim) (4.0.1)
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.6/dist-packages (from gensim) (1.15.0)
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.6/dist-packages (from gensim) (1.19.4)
--2020-12-21 22:34:54--  https://s3.amazonaws.com/dl4j-distribution/GoogleNews-vectors-negative300.bin.gz
Resolving s3.amazonaws.com (s3.amazonaws.com)... 52.217.49.102
Connecting to s3.amazonaws.com (s3.amazonaws.com)|52.217.49.102|:443... connected.
HTTP request sent, awaiting response... 416 Requested Range Not Satisfiable

    The file is already fully retrieved; nothing to do.
```

Test word2vec

- The raw vector for a word

```
# test the loaded word2vec embeddings
print(model['headache'])
```

```
[ 1.54296875e-01  2.56347656e-02 -1.28906250e-01  3.85742188e-02
 -1.74804688e-01  1.84570312e-01 -1.37939453e-02 -7.71484375e-02
 -7.32421875e-02  3.65234375e-01  9.91210938e-02 -3.39843750e-01
  2.89306641e-02 -1.33056641e-02 -1.04980469e-02  4.53125000e-01
  4.98046875e-02  1.75781250e-02  8.15429688e-02 -4.31640625e-01
 -7.66601562e-02  1.75781250e-01  2.71484375e-01  7.81250000e-03
 -2.11181641e-02  2.22656250e-01 -1.13769531e-01 -2.96875000e-01
 -1.46484375e-01 -7.03125000e-02 -5.54199219e-02 -1.59912109e-02
  1.21093750e-01 -1.26953125e-01 -1.09863281e-01  2.64892578e-02
  1.66015625e-01 -1.57226562e-01  2.69531250e-01  7.17773438e-02
 -7.66601562e-02 -3.24218750e-01  2.32421875e-01  9.46044922e-03
 -2.24609375e-01 -7.91015625e-02  6.20117188e-02  1.13769531e-01
```

Closest words

- The words closest to a word

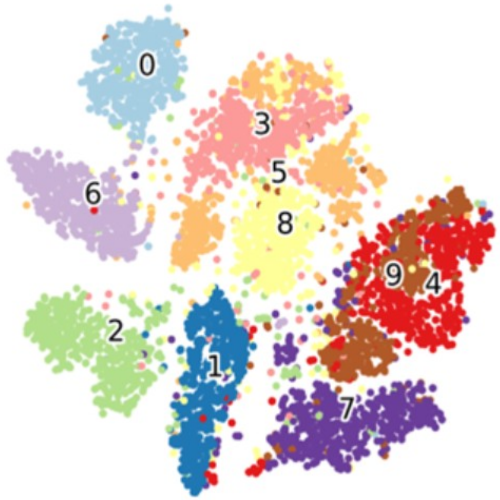
```
# the words closest to a word
model.similar_by_word('headache')
```

```
[('headaches', 0.8292464017868042),
 ('heartburn', 0.5746024847030640),
 ('indigestion', 0.5650683641433716),
 ('nightmare', 0.5287497639656067),
 ('nightmares', 0.5257063508033752),
 ('Headaches', 0.5097386837005615),
 ('stomach_ache', 0.4992206096649170),
 ('problem', 0.4964112341403961),
 ('palpitations', 0.48661914467811584),
 ('earache', 0.4859495460987091)]
```

Visualization: tSNE

t-distributed stochastic neighbor embedding

T-distributed Stochastic Neighbor Embedding (t-SNE) is a machine learning algorithm for visualization developed by Laurens van der Maaten and Geoffrey Hinton. It is a nonlinear dimensionality reduction technique well-suited for embedding high-dimensional data for visualization in a low-dimensional space of two or three dimensions. Specifically, it models each high-dimensional object by a two- or three-dimensional point in such a way that similar objects are modeled by nearby points and dissimilar objects are modeled by distant points with high probability.



https://towardsdatascience.com/t-distributed-stochastic-neighbor-embedding-t-sne-bb60ff109561

PCA vs. tSNE

PCA

•**An unsupervised, deterministic algorithm used for feature extraction as well as visualization**

•Applies a **linear dimensionality reduction technique** where the **focus is on keeping the dissimilar points far apart in a lower-dimensional space**.

•**Transforms the original data to a new data by preserving the variance in the data using eigenvalues**.

•**Outliers impact PCA.**

t-SNE

•An **unsupervised, randomized algorithm, used only for visualization**

•Applies a **non-linear dimensionality reduction techniqu**e where the f**ocus is on keeping the very similar data points close together in lower-dimensional space**.

•**Preserves the local structure of the data using student t-distribution** to compute the similarity between two points in lower-dimensional space.

•t-SNE uses a **heavy-tailed Student-t distribution to compute the similarity between two points in the low-dimensional space** rather than a Gaussian distribution, which helps to a**ddress the crowding and optimization problems**.

•**Outliers do not impact t-SNE**

Headache in Spacy using word2vec and tSNE

```
import pandas as pd
pd.options.mode.chained_assignment = None
import numpy as np
import re

from gensim.models import word2vec

from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
%matplotlib inline
```

```
[2] import spacy
    nlp = spacy.load('en')
```

```
[3] # load headache notes
    from google.colab import files
    uploaded = files.upload()
```

Browse... notes_headache.txt

notes_headache.txt(text/plain) - 9433 bytes, last modified: n/a - 100% done

```
Saving notes_headache.txt to notes_headache.txt
```

Build notes

```
[4] notes = []
    with open('notes_headache.txt', 'r') as fin:
      lines = fin.readlines()
      for line in lines:
        notes.append(line)
    print(notes)
    print(len(notes))
```

```
['50 year old female presents after having fallen in t
11
```

```
[5] df=notes
```

Build corpus

```
# Build corpus of all the entities extracted from the notes using spaCy model.
# The corpus is an array of arrays or list of lists where each of the nested lists corresponds to a note.
corpus=[]
for row in range(0, len(df)):
  str_tokens=[]
  tokens= nlp(df[row]).ents
  for i in range(0, len(tokens)):
    str_tokens.append(tokens[i].text)
  corpus.append(list(str_tokens))



print(corpus)
```

```
[['50 year old', 'the bathtub 4 days ago', 'Tylenol', 'the day', 'night', 'morning'], ['23', 'PD', 'RUE', 'I
```

Get word2vec

```
!pip install gensim
```

```
Requirement already satisfied: gensim in /usr/local/lib/python3.6/dist-packages (3.6.0)
Requirement already satisfied: scipy>=0.18.1 in /usr/local/lib/python3.6/dist-packages (from gensim) (1.4.1)
Requirement already satisfied: smart-open>=1.2.1 in /usr/local/lib/python3.6/dist-packages (from gensim) (4.0.1)
Requirement already satisfied: six>=1.5.0 in /usr/local/lib/python3.6/dist-packages (from gensim) (1.15.0)
Requirement already satisfied: numpy>=1.11.3 in /usr/local/lib/python3.6/dist-packages (from gensim) (1.19.4)
```

```
[10] import gensim
```

```
[13] model = word2vec.Word2Vec(corpus, min_count=1)
```

word2vec

```
model.wv['Tylenol']
```

```
array([ 3.5755248e-03, -7.5043686e-04, -4.0175230e-03, -5.8088248e-04,
       -1.4506313e-03, -1.5477943e-03,  8.4892643e-04, -1.1564353e-03,
        5.8339350e-04, -6.3213066e-04,  4.9648127e-03,  4.2244259e-04,
       -1.5812013e-03,  6.0994085e-04, -3.7646745e-03,  8.8548280e-05,
        2.9612291e-03,  2.5223016e-03,  1.2566439e-03,  3.0257232e-03,
        4.2598490e-03, -3.7888133e-03,  3.4991202e-03,  3.4320420e-03,
        4.6869563e-03, -1.6790380e-03, -4.3789954e-03,  4.1352160e-04,
        2.0200359e-03,  4.0350412e-03, -4.1124565e-03,  2.0835653e-03,
        2.9961050e-03,  2.4687620e-03,  4.6605510e-03,  6.0498028e-04,
        2.3130388e-03,  3.9473213e-03, -3.8562799e-03,  2.5067695e-03,
       -3.9585214e-03, -3.8969840e-03, -1.2115871e-03, -1.3817309e-03,
       -4.6661417e-03,  1.0953289e-03,  4.2884829e-03,  1.8828238e-03,
        2.0013545e-03, -3.4035782e-03,  1.8964872e-03, -2.7806829e-03,
        4.3920926e-03, -2.2318871e-03, -3.1314581e-03,  4.4012973e-03,
        4.0750531e-03, -3.3024163e-03, -1.6775471e-03, -4.1697090e-03,
        4.7896975e-03,  3.7045595e-03, -3.4035568e-06,  1.1239357e-04,
        9.3653216e-05, -4.9044727e-03, -1.8739082e-03,  7.7197549e-04,
       -3.4613409e-03, -1.1089609e-03, -9.2508399e-04,  2.2311143e-03,
       -7.7253534e-04,  5.5203785e-04, -4.1779936e-03, -2.6194218e-03,
```

tSNE

Parameters:

n_components : *int, default=2*

Dimension of the embedded space.

perplexity : *float, default=30.0*

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 selecting a value between 5 and 50. Different values can result in significantly different results.

early_exaggeration : *float, default=12.0*

Controls how tight natural clusters in the original space are in the embedded space and how much space will be between them. For larger values, the space between natural clusters will be larger in the embedded space. Again, the choice of this parameter is not very critical. If the cost function increases during initial optimization, the early exaggeration factor or the learning rate might be too high.

learning_rate : *float, default=200.0*

The learning rate for t-SNE is usually in the range [10.0, 1000.0]. If the learning rate is too high, the data may look like a 'ball' with any point approximately equidistant from its nearest neighbours. If the learning rate is too low, most points may look compressed in a dense cloud with few outliers. If the cost function gets stuck in a bad local minimum increasing the learning rate may help.

n_iter : *int, default=1000*

https://scikit-learn.org/stable/modules/generated/sklearn.manifold.TSNE.html

tSNE

init : *{'random', 'pca'} or ndarray of shape (n_samples, n_components), default='random'*

Initialization of embedding. Possible options are 'random', 'pca', and a numpy array of shape (n_samples, n_components). PCA initialization cannot be used with precomputed distances and is usually more globally stable than random initialization.

random_state : *int, RandomState instance or None, default=None*

Determines the random number generator. Pass an int for reproducible results across multiple function calls. Note that different initializations might result in different local minima of the cost function. See :term: `Glossary <random_state>`.

n_iter : *int, default=1000*

Maximum number of iterations for the optimization. Should be at least 250.

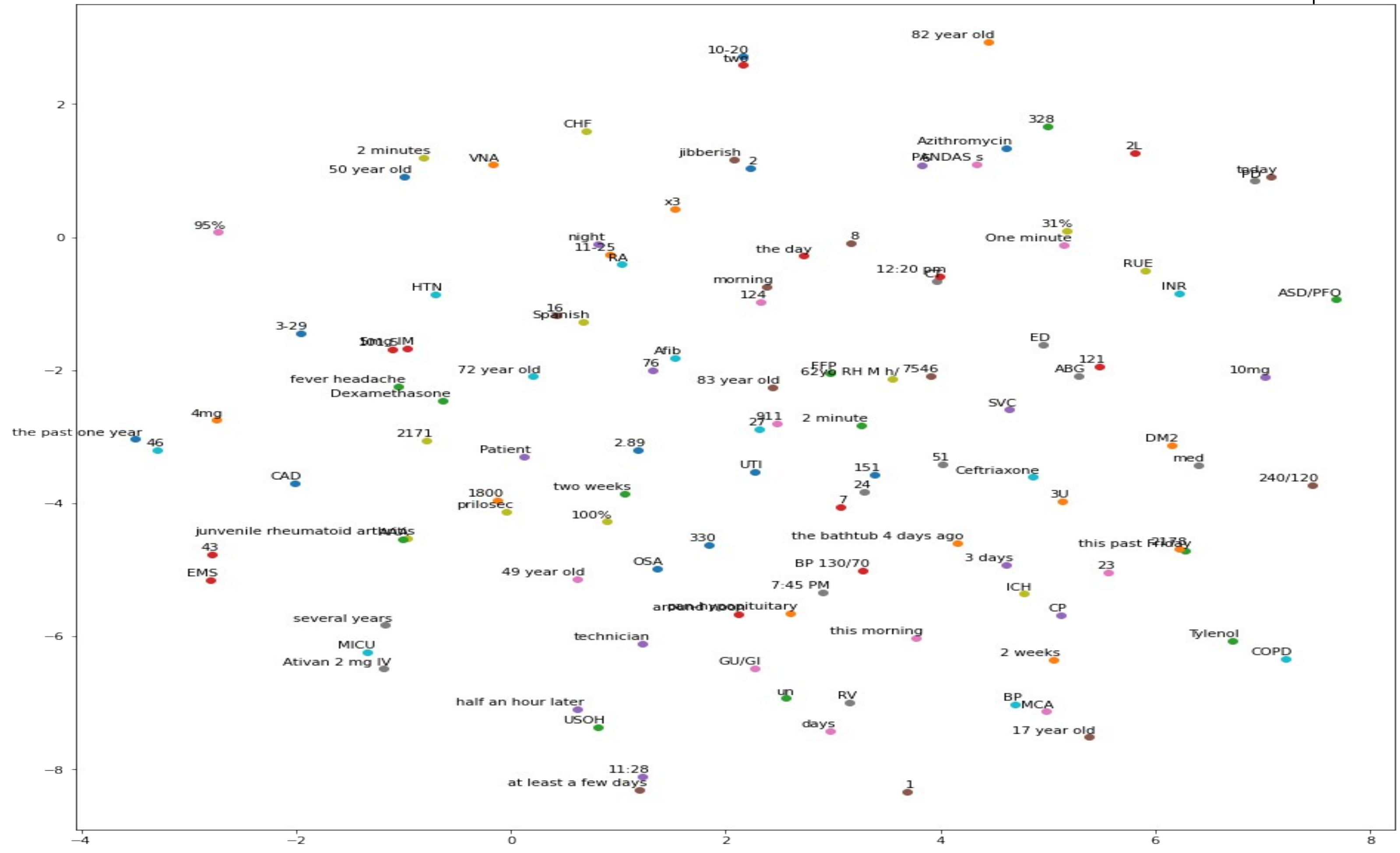
Using tSNE to plot

```
def tsne_plot(model):
    "Creates and TSNE model and plots it"
    labels = []
    tokens = []

    for word in model.wv.vocab:
        tokens.append(model[word])
        labels.append(word)

    tsne_model = TSNE(perplexity=30, early_exaggeration=12, n_components=2, init='pca', n_iter=1000, random_state=23)
    new_values = tsne_model.fit_transform(tokens)

    x = []
    y = []
    for value in new_values:
        x.append(value[0])
        y.append(value[1])
```

```
plt.figure(figsize=(16, 16))
for i in range(len(x)):
    plt.scatter(x[i],y[i])
    plt.annotate(labels[i],
                 xy=(x[i], y[i]),
                 xytext=(5, 2),
                 textcoords='offset points',
                 ha='right',
                 va='bottom')
plt.show()
```

```
tsne_plot(model)
```

tSNE plot



82 year old
10-20
two
CHF
328
Azithromycin
2L
2 minutes
VNA
jibberish
2
PANDAS s
today
PD
50 year old
x3
95%
night
11-25
RA
8
the day
31%
One minute
12:20 pm
RUE
HTN
morning
124
INR
ASD/PFO
16
Spanish
3-29
SIM
ED
Afib
121
76
ABG
10mg
72 year old
fever headache
83 year old
7546
Dexamethasone
SVC
4mg
911
27
2 minute
the past one year
2171
46
Patient
2.89
DM2
UTI
51
med
151
Ceftriaxone
CAD
240/120
24
1800
prilosec
two weeks
7
3U
100%
junvenile rheumatoid arthritis
330
the bathtub 4 days ago
43
this past Friday
3 days
OSA
BP 130/70
EMS
49 year old
23
7:45 PM
ICH
CP
several years
this morning
Tylenol
technician
MICU
2 weeks
COPD
Ativan 2 mg IV
GU/GI
up
RV
BP
MCA
half an hour later
USOH
days
17 year old
11:28
at least a few days
1

Headache in SciSpacy using word2vec and tSNE

```
[ ]  !pip install scispacy
```

```
        |                                | 51kB 2.7MB/s
Collecting nmslib>=1.7.3.6
  Downloading https://files.pythonhosted.org/packages/d5/
        |                                | 13.0MB 266kB/s
Collecting pysbd
  Downloading https://files.pythonhosted.org/packages/26/
        |                                | 71kB 6.6MB/s
Requirement already satisfied: joblib in /usr/local/lib/p
```

```
import scispacy
import spacy

import en_ner_bc5cdr_md

nlp = en_ner_bc5cdr_md.load()
```

```
/usr/local/lib/python3.6/dist-packages/spac
  warnings.warn(warn_msg)
```

Load notes

```
# load headache notes
from google.colab import files
uploaded = files.upload()
```

Browse... notes_headache.txt

notes_headache.txt(text/plain) - 9433 bytes, last modified: n/a - 100% done

```
Saving notes_headache.txt to notes_headache (1).txt
```

```
notes = []
with open('notes_headache.txt', 'r') as fin:
  lines = fin.readlines()
  for line in lines:
    notes.append(line)
print(notes)
print(len(notes))
```

```
['50 year old female presents after having fallen in the bat
11
```

```
df=notes
```

Build corpus using SciSpacy

```
# Build corpus of all the entities extracted from the notes using
# The corpus is an array of arrays or list of lists where each o
corpus=[]
for row in range(0, len(df)):
  str_tokens=[]
  tokens= nlp(df[row]).ents
  for i in range(0, len(tokens)):
    str_tokens.append(tokens[i].text)
  corpus.append(list(str_tokens))


print(corpus)
```

```
[['headache', 'Tylenol', 'pain', 'dizziness', 'nausea', 'vomiting
```

Get word2vec

```
!pip install gensim
```

```
Requirement already satisfied: gensim in /usr/lo
Requirement already satisfied: six>=1.5.0 in /us
Requirement already satisfied: smart-open>=1.2.1
Requirement already satisfied: scipy>=0.18.1 in
Requirement already satisfied: numpy>=1.11.3 in
```

```
[8] import gensim
```

```
[10] import pandas as pd
     pd.options.mode.chained_assignment = None
     import numpy as np
     import re

     from gensim.models import word2vec

     from sklearn.manifold import TSNE
     import matplotlib.pyplot as plt
     %matplotlib inline
```

```
[11] model = word2vec.Word2Vec(corpus, min_count=1)
```

Show one word2vec example

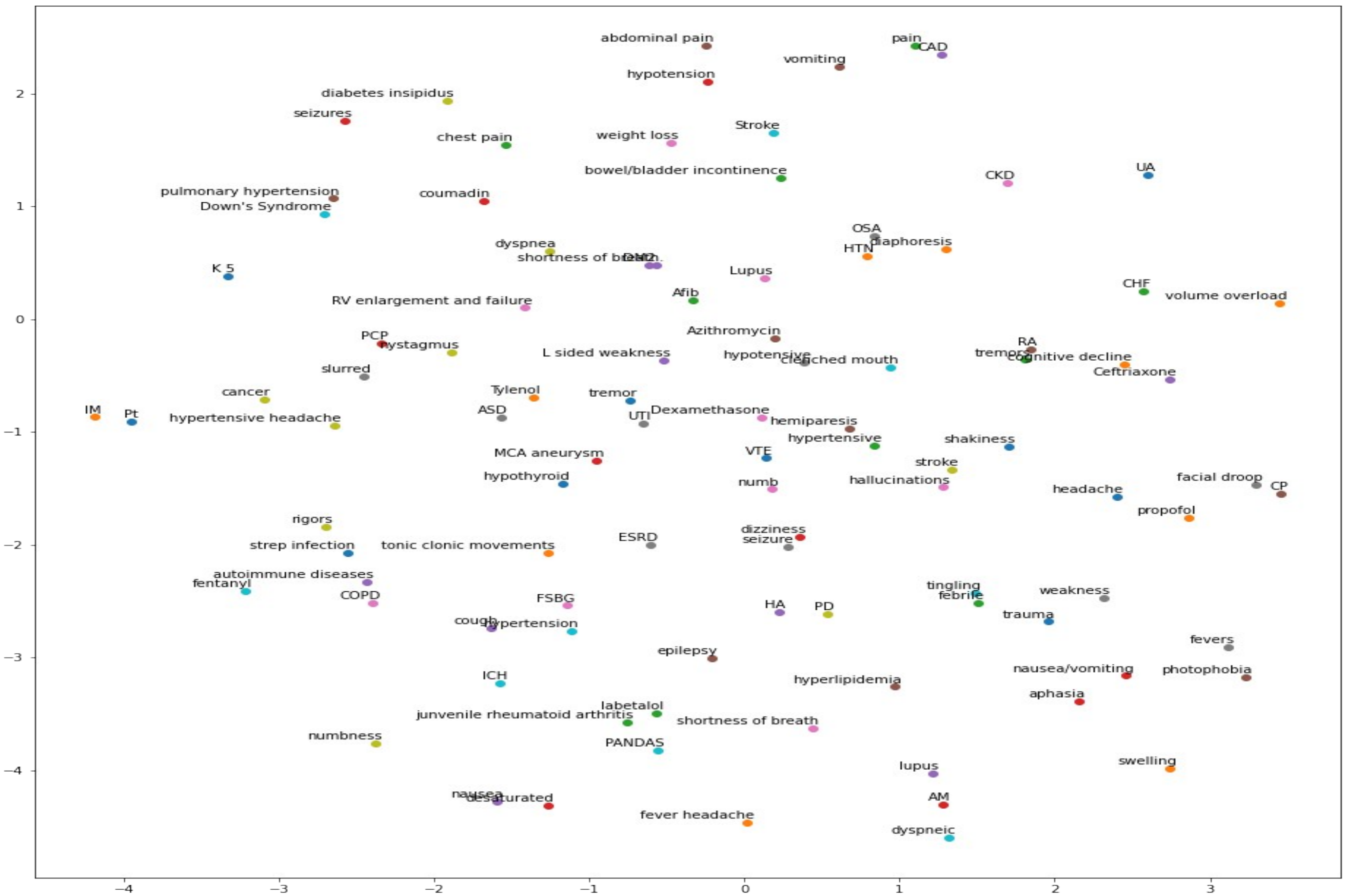
```
[12] model.wv['Tylenol']

array([ 0.00421088,  0.00021863, -0.00035665,  0.00322251,  0.00073423,
       -0.00270951, -0.00204165, -0.00214327,  0.0003273 , -0.00207853,
       -0.00156233, -0.00453952,  0.00329102,  0.00021155,  0.00089793,
        0.0006254 ,  0.0025706 , -0.0007755 , -0.00275109, -0.0030539 ,
       -0.00065839,  0.00411886, -0.0025017 , -0.00214221, -0.0040169 ,
        0.00014435, -0.00144451, -0.00358639,  0.00494652,  0.00434491,
       -0.0049477 , -0.00241251,  0.00287517,  0.00384575, -0.00263347,
        0.00472971,  0.00027301,  0.002738  , -0.00365123,  0.00489407,
        0.00369412, -0.00314624, -0.00289283,  0.00131375, -0.00289287,
        0.00205069, -0.00407033,  0.00303441, -0.00062189,  0.00429522,
        0.00373649,  0.0030437 , -0.00100484, -0.00059962, -0.00244993,
       -0.00459783,  0.00123144, -0.0017654 ,  0.00143283, -0.0041041 ,
       -0.00480347, -0.00153869, -0.00160133,  0.00375598, -0.0031069 ,
        0.0004454 ,  0.00474415, -0.00210162,  0.00059001,  0.00245465,
        0.00175198, -0.00054866,  0.00081636,  0.00372542, -0.00212218,
       -0.00335156,  0.00112638, -0.00133766,  0.00219538,  0.00339135,
        0.00437942,  0.00143798, -0.00139074, -0.0019345 , -0.00314154,
       -0.00173181, -0.00307571, -0.00368569, -0.00472091,  0.00260101,
       -0.00292352,  0.00355645, -0.001855  , -0.00047691,  0.00441105,
        0.00378569,  0.0006488 ,  0.00290273, -0.0008887 ,  0.00294835],
```

tSNE plot

```python
def tsne_plot(model):
    "Creates and TSNE model and plots it"
    labels = []
    tokens = []

    for word in model.wv.vocab:
        tokens.append(model[word])
        labels.append(word)

    tsne_model = TSNE(perplexity=30, early_exaggeration=12, n_components=2, init='pca', n_iter=1000, random_state=23)
    new_values = tsne_model.fit_transform(tokens)

    x = []
    y = []
    for value in new_values:
        x.append(value[0])
        y.append(value[1])
```

```python
plt.figure(figsize=(16, 16))
for i in range(len(x)):
    plt.scatter(x[i],y[i])
    plt.annotate(labels[i],
                 xy=(x[i], y[i]),
                 xytext=(5, 2),
                 textcoords='offset points',
                 ha='right',
                 va='bottom')
plt.show()
```

tSNE plot using SciSpacy




```
tsne_plot(model)
```

Exercise

- Getting headache discharge notes from MIMIC III (around 700 notes)
- Using SciSpacy entity extraction
- Connecting with word2vec
- Plotting them using tSNE

Headache notes from MIMIC

```
select diagnoses_icd.subject_id, diagnoses_icd.hadm_id, text
from noteevents inner join diagnoses_icd
on noteevents.hadm_id=diagnoses_icd.hadm_id
and noteevents.subject_id=diagnoses_icd.subject_id
where diagnoses_icd.ICD9_CODE='430'
```

https://querybuilder-lcp.mit.edu/

```
select diagnoses_icd.subject_id, diagnoses_icd.hadm_id, text
from noteevents inner join diagnoses_icd
on noteevents.hadm_id=diagnoses_icd.hadm_id
where diagnoses_icd.ICD9_CODE='430' and noteevents.CATEGORY = 'Discharge summary'
```

```
select diagnoses_icd.subject_id, diagnoses_icd.hadm_id, ICD9_CODE
from noteevents inner join diagnoses_icd
on noteevents.hadm_id=diagnoses_icd.hadm_id
where diagnoses_icd.ICD9_CODE='430' and noteevents.CATEGORY = 'Discharge summary'
```

Thank you!