digikalacademy

Session 4 (Lab)

Effective Word Representation by python

Azam Rabiee, PhD

September 13, 2020

Outline

Session 1: Introduction

- Applications
- Tasks
- Approaches

Session 2. Basics of Linguistics

- Components
- Challenges
- Vectorization

Session 3. Basics of ML

- word2vec
- Components
- Architectures

Session 4 (Lab). Effective Word Representation by python

System Requirements

python (>= 3.6)

Packages

- jupyter
- gensim
- numpy
- nltk

To install the packages, you may check The following page:
https://github.com/AzamRabiee/Lab-material-for-Intro-to-AI-ML

Q: Who did not work with jupyter notebook before?

Useful Python Library

Package Name: NLTK

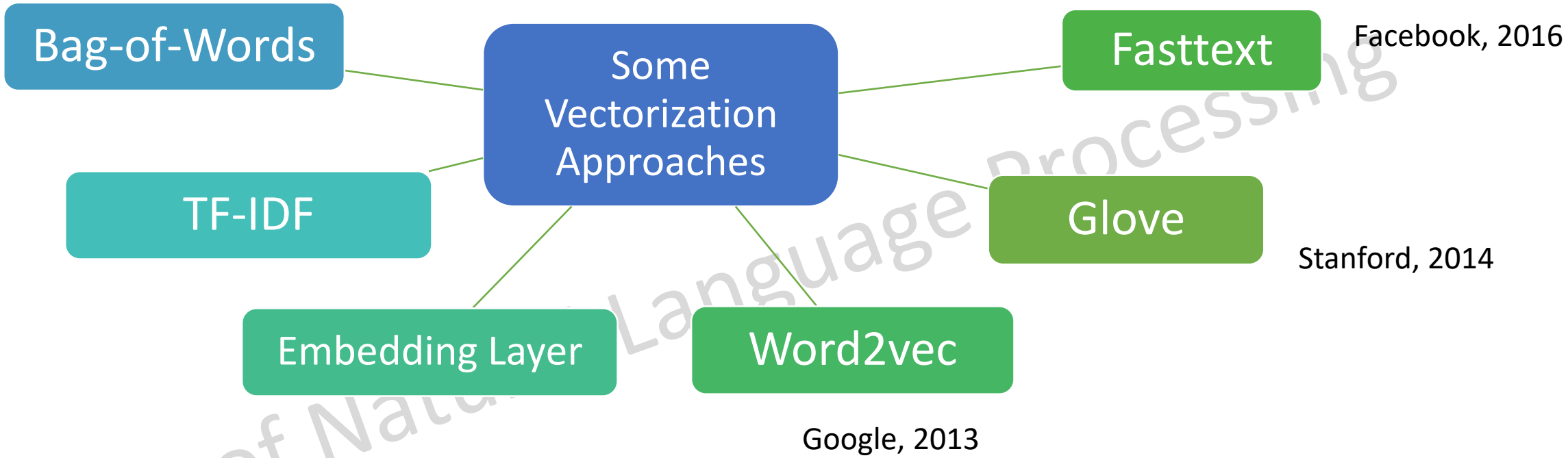
Description: Natural Language Toolkit

How to install: pip install nltk

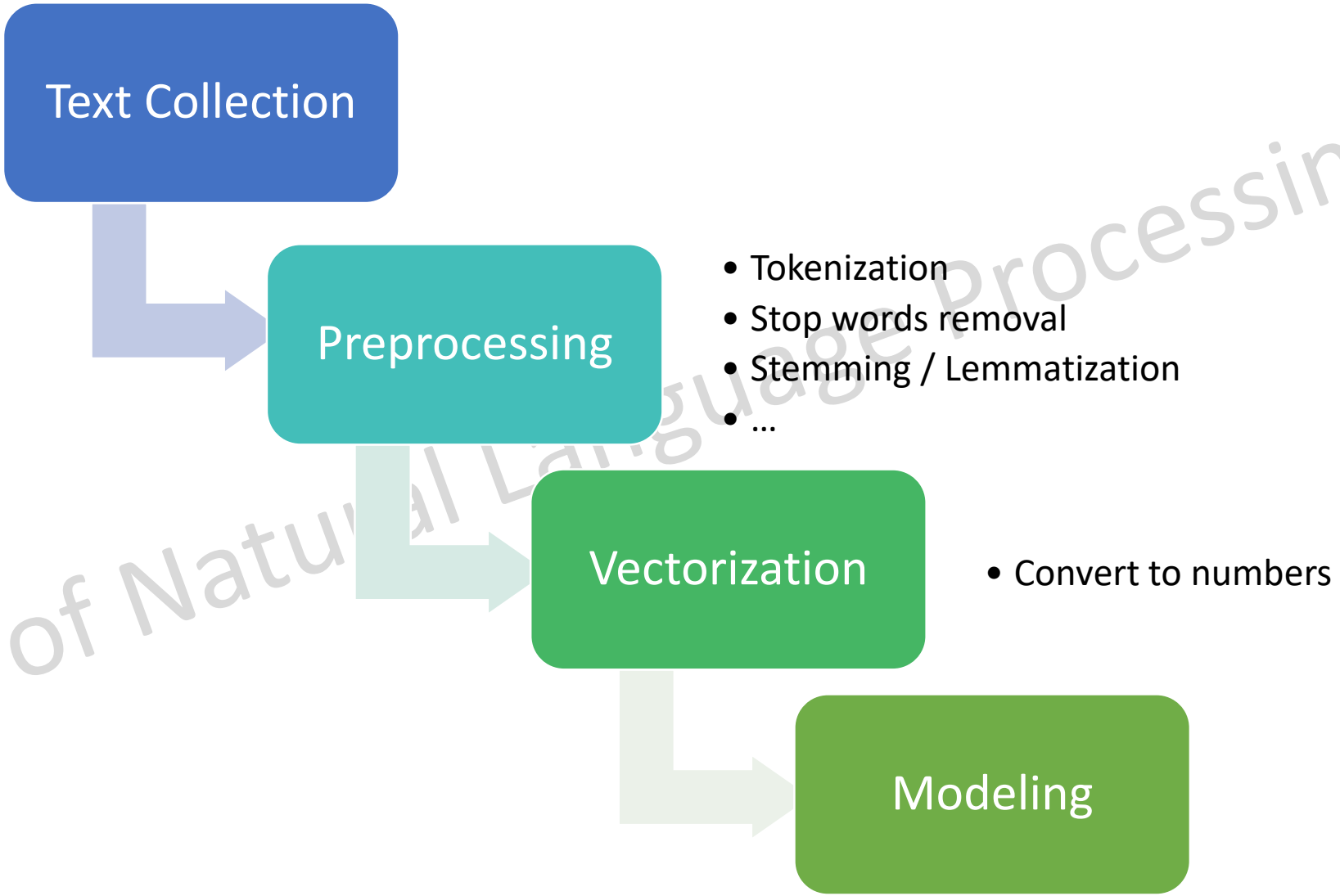

```
import nltk
```

Let's see how nltk works for stemming and lemmatization?

Vectorization



Steps of ML/DL Projects with Text Data



word2vec

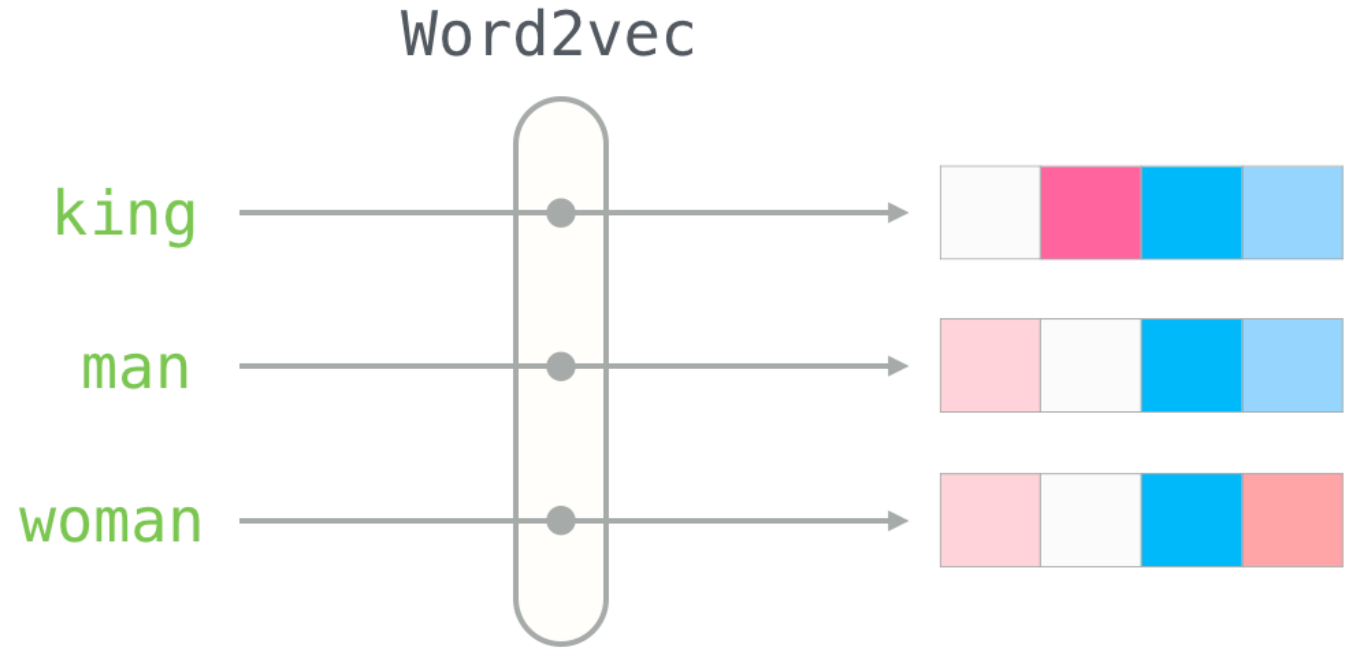
Word2vec is a model provided by Google in 2013

for the **effective word embedding**.

[Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013]

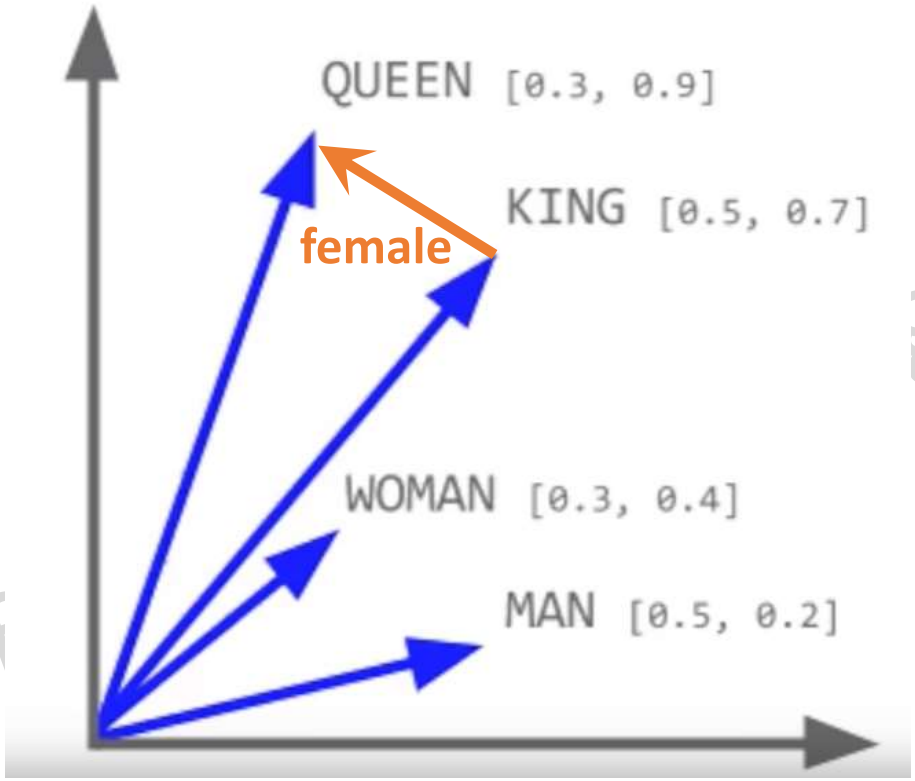
word2vec

Word2vec is a model provided by Google in 2013 for the **effective word embedding**.



[Mikolov, Tomas, et al. "Distributed representations of words and phrases and their compositionality." *Advances in neural information processing systems*. 2013]

Word Embedding



King + female = Queen

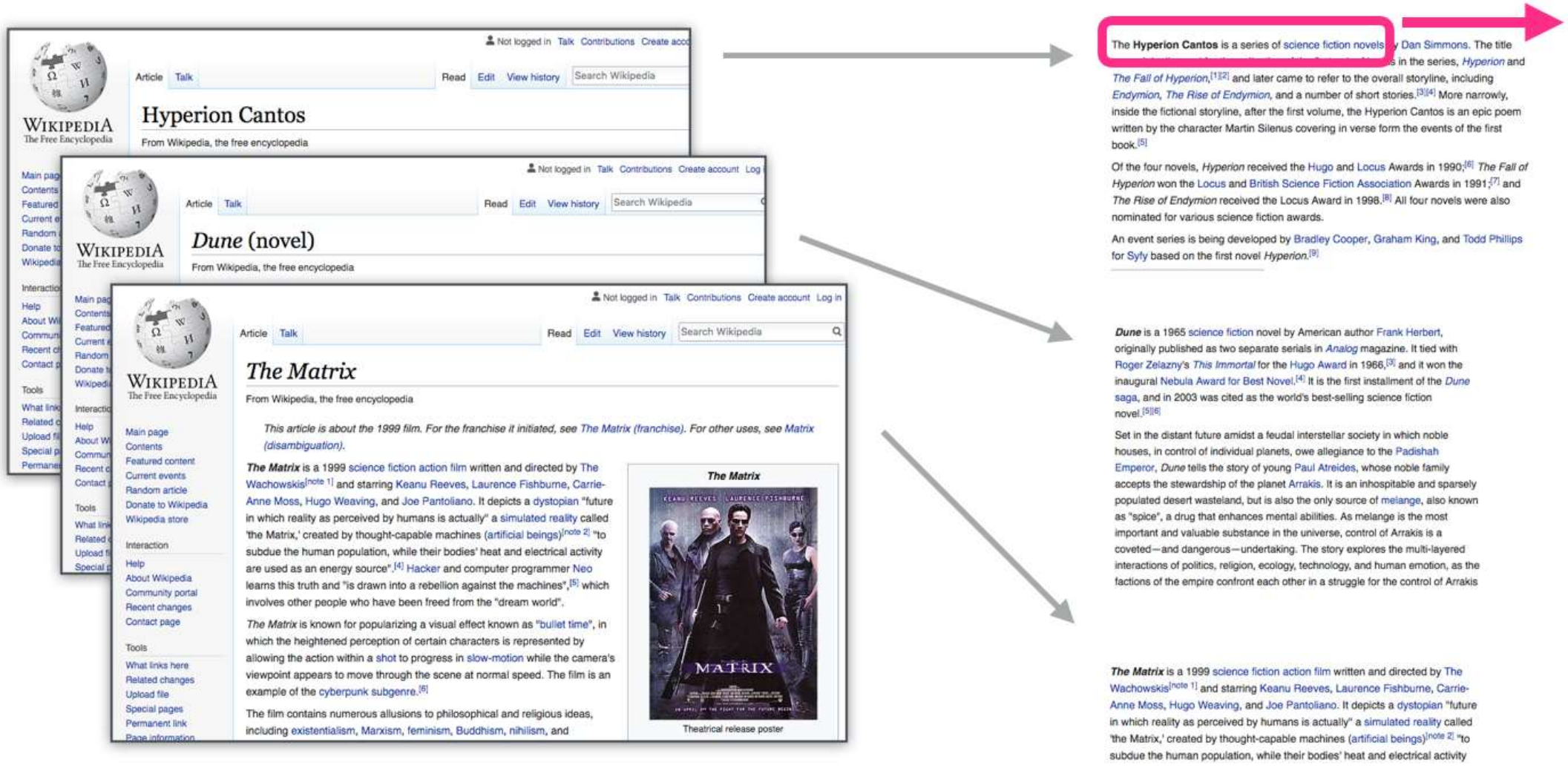
Man + female = Woman

Queen – royal = Woman

Steps of word2vec

Dataset

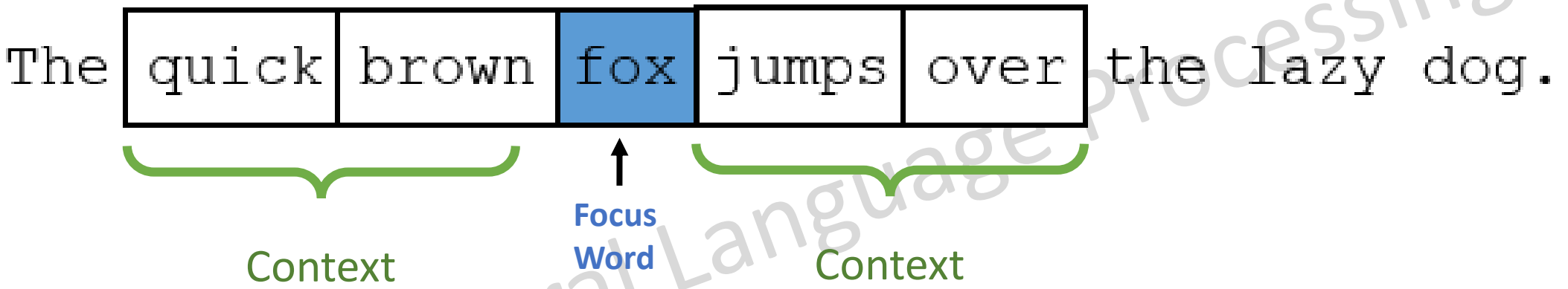
Sliding window across running text



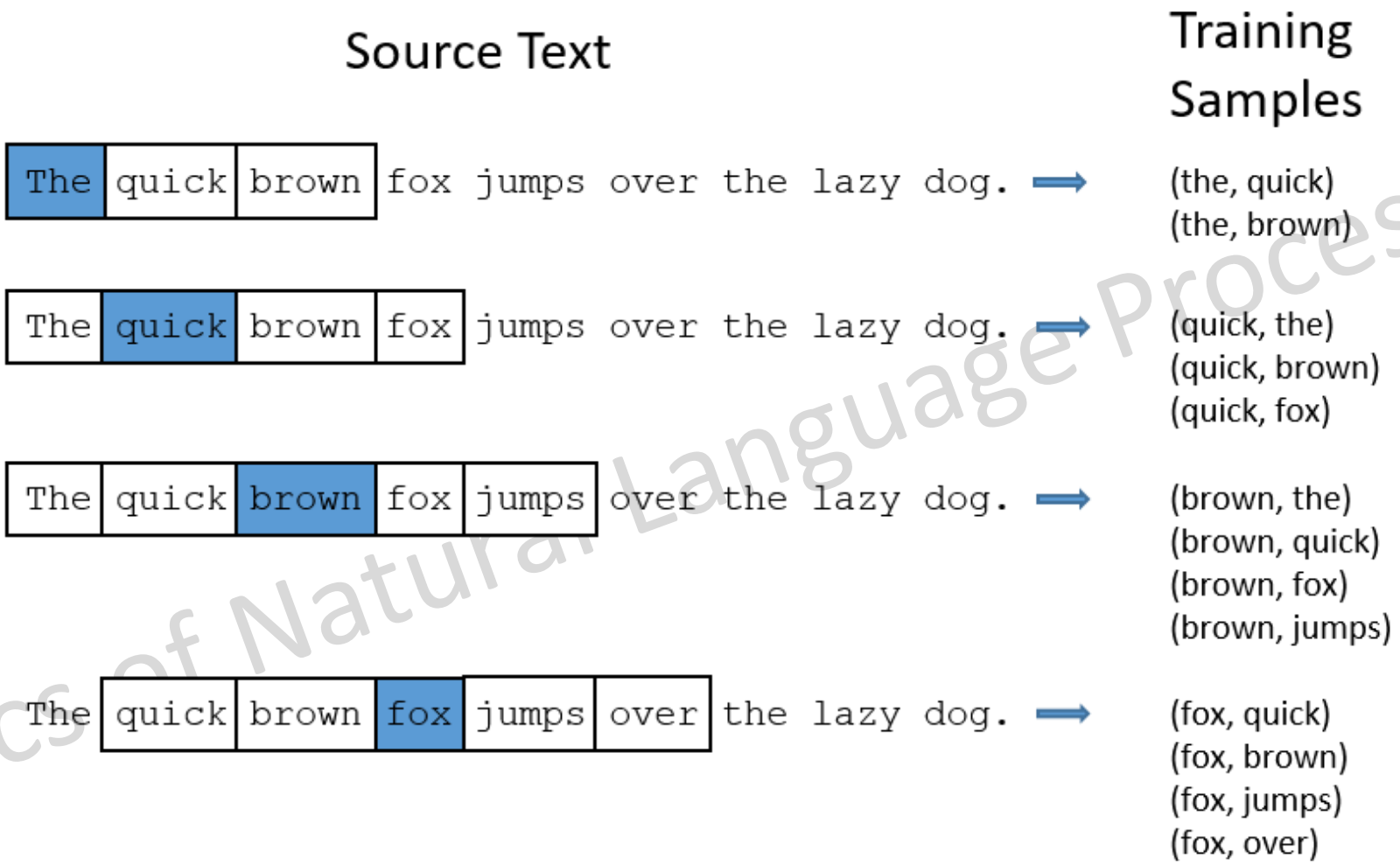
[http://jalammar.github.io/illustrated-word2vec/]

Sliding Window

Window size=5

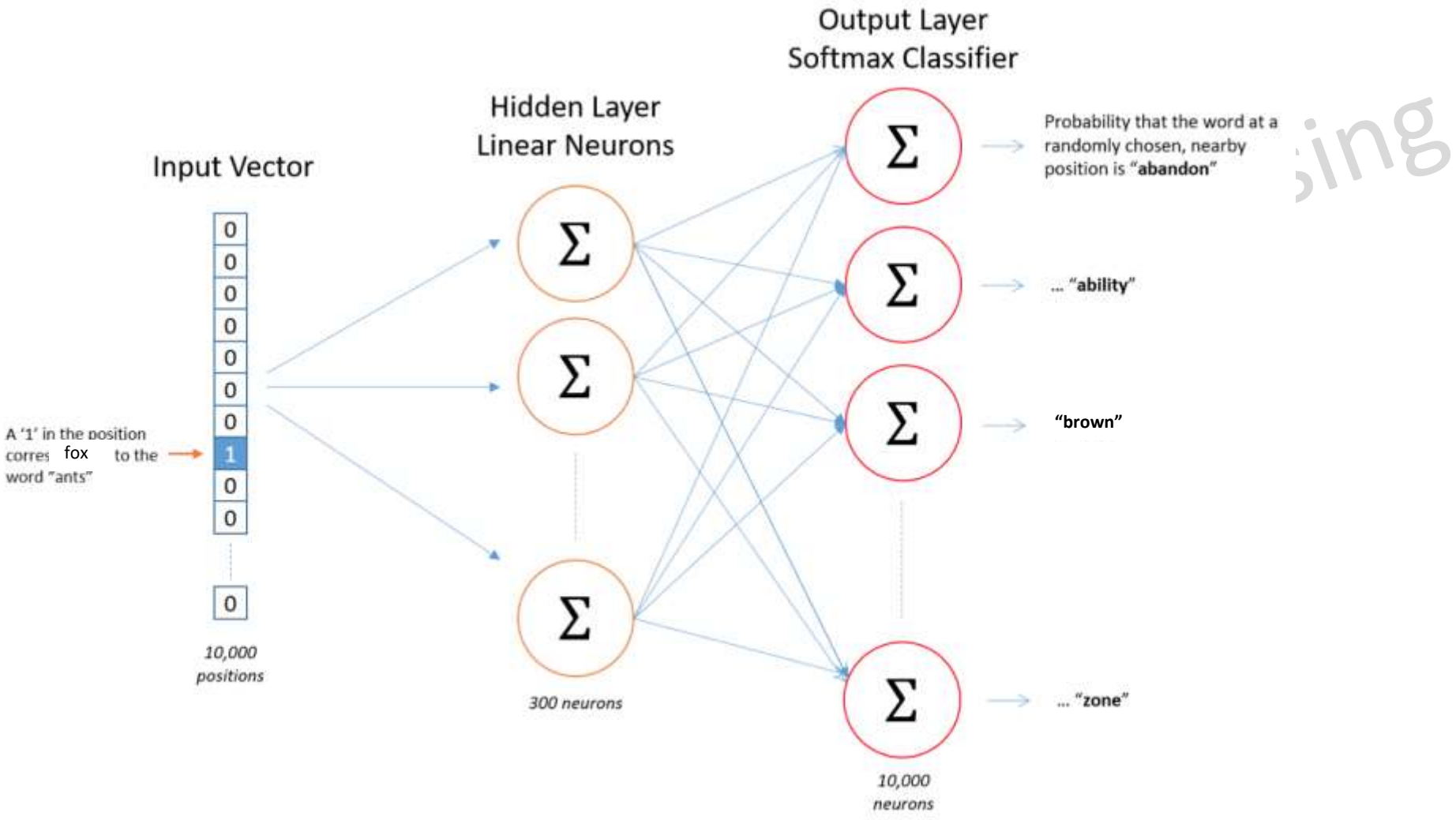


Example Dataset



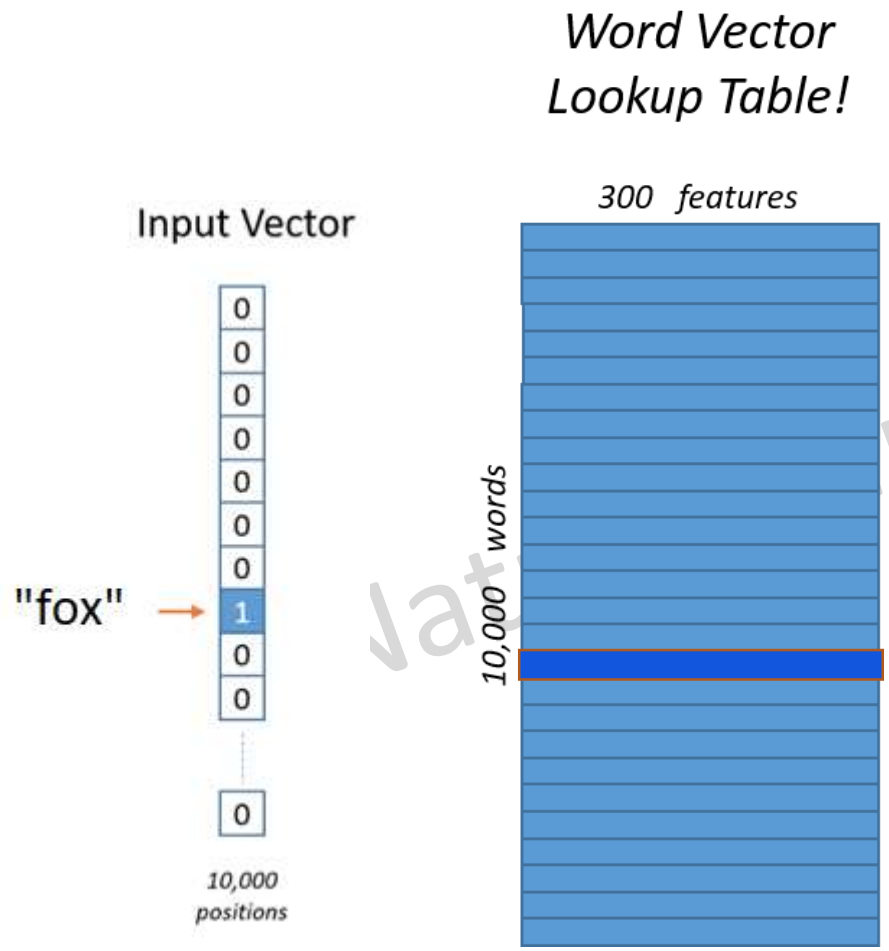
[McCormick, C.: Word2vec Tutorial - The Skip-Gram Model. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/]

Network Architecture



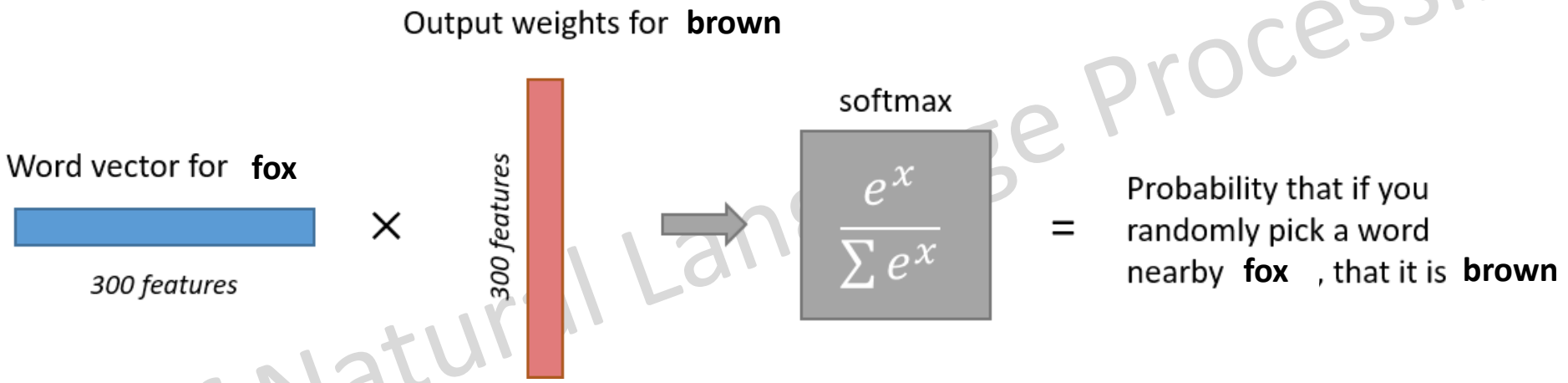
[McCormick, C.: Word2vec Tutorial - The Skip-Gram Model. http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/]

Hidden Layer



Output of the hidden layer:
embedding vector for “fox”

Output Layer



NLTK Corpora

http://www.nltk.org/nltk_data/

NLTK Corpora

NLTK has built-in support for dozens of corpora and trained models, as listed below. To use these within NLTK we recommend that you use the NLTK corpus downloader, >>> `nltk.download()`

Please consult the README file included with each corpus for further information.

1. *perluniprops: Index of Unicode Version 7.0.0 character properties in Perl* [download | source]
   id: `perluniprops`; size: 100266; author: ; copyright: ; license: ;

2. *The monolingual word aligner (Sultan et al. 2015) subset of the Paraphrase Database.* [download | source]
   id: `mwa_ppdb`; size: 1594711; author: ; copyright: ; license: Creative Commons Attribution 3.0 Unporte (CC-BY);

3. *Punkt Tokenizer Models* [download | source]
   id: `punkt`; size: 13707633; author: Jan Strunk; copyright: ; license: ;

---

36. *Crubadan Corpus* [download | source]
    id: `crubadan`; size: 5288655; author: Kevin Scannell; copyright: Copyright (C) 2010 Kevin Scannell; license: GPLv3;

37. *Project Gutenberg Selections* [download | source]
    id: `gutenberg`; size: 4251829; author: ; copyright: public domain; license: public domain;

38. *Proposition Bank Corpus 1.0* [download | source]
    id: `propbank`; size: 5323498; author: ; copyright: ; license: Distributed with permission;

---

107. *Help on Tagsets* [download | source]
     id: `tagsets`; size: 34531; author: UCREL, Lancaster University; copyright: ; license: ;

Natural Language Toolkit

Download Gutenberg Corpora

```
import nltk
from nltk.corpus import gutenberg

# download the 'gutenburg' corpora
# you can see the files at %APPDATA%/nltk_data
nltk.download('gutenberg')

# This tokenizer divides a text into words
nltk.download('punkt')

# import the corpus and convert into a list
sentences = list(gutenberg.sents('shakespeare-hamlet.txt'))
```

Data Preprocessing

```
['[', 'The', 'Tragedie', 'of', 'Hamlet', 'by', 'William', 'Shakespeare', '1599', ']']
```

Q: Any word preprocessing is needed?

Data Preprocessing

```
['[', 'The', 'Tragedie', 'of', 'Hamlet', 'by', 'William', 'Shakespeare', '1599', ']']
```

```
[`the', `tragedie', 'of', `hamlet', 'by', `william', `shakespeare']
```

- Change to lowercase
- Remove numbers, punctuations and any thing rather than ‘a’ to ‘z’

Useful Python Library

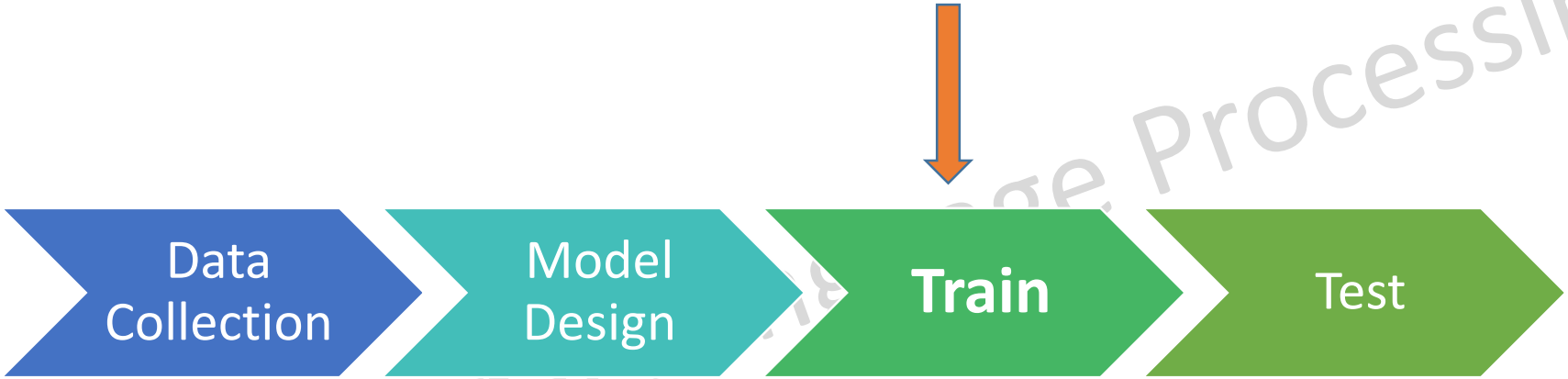
Package Name: re

Description: Regular expression

How to install: It is already installed by python

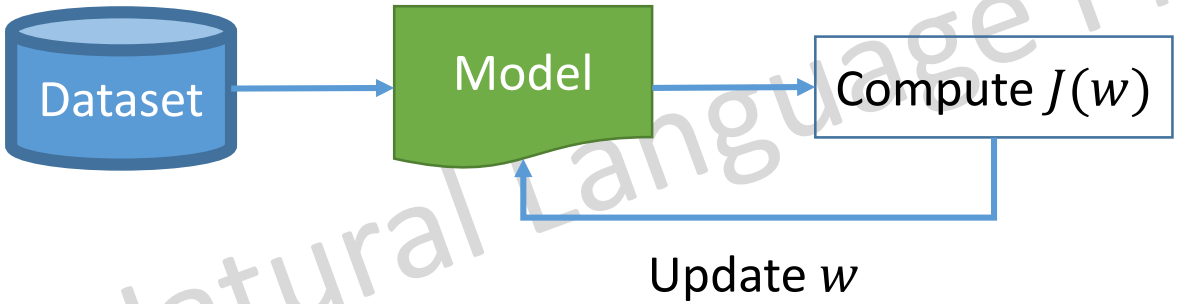
```
import re

# The following code returns true if the
word contains only 'a-z' and 'A-Z'

re.match('[a-zA-Z]+', word)
```

Steps of word2vec



Training Process

Finding optimum parameters that minimize a desired cost function $J(w)$.



Useful Python Library

Package Name: gensim

Description: Generate Similar; useful package for text-mining and NLP

How to install: pip install gensim

```
import gensim
```


Note: we are going to use the Word2Vec class of this package.

Word2Vec Class

```
from gensim.models import word2Vec
```


[https://radimrehurek.com/gensim/models/word2vec.html]

Word2Vec Class

```
model = Word2Vec(sentences=sentences, size=20, window=3, iter=100)
```

The above line makes an instance of the Word2Vec class, named model, initiates it, and trains the network.

sentences: training data (has to be a list with tokenized sentences)

size: dimension of embedding space

window: number of words accounted for each context (if the window size is 3, 3 word in the left neighorhood and 3 word in the right neighborhood are considered)

iter: number of training iterations

[https://radimrehurek.com/gensim/models/word2vec.html]

Let’s see every thing in Google Colab