Tamkang Universit
淡江大學

人工智慧文本分析 (AI for Text Analytics)

文本表達特徵工程 (Feature Engineering for Text Representation)

1091AITA05
MBA, IMTKU (M2455) (8418) (Fall 2020)
Thu 3, 4 (10:10-12:00) (B206)

Min-Yuh Day
戴敏育
Associate Professor
副教授
Institute of Information Management, National Taipei University
國立臺北大學 資訊管理研究所
https://web.ntpu.edu.tw/~myday

2020-10-22

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
1	2020/09/17	人工智慧文本分析課程介紹 (Course Orientation on Artificial Intelligence for Text Analytics)
2	2020/09/24	文本分析的基礎：自然語言處理 (Foundations of Text Analytics: Natural Language Processing; NLP)
3	2020/10/01	中秋節 (Mid-Autumn Festival) 放假一天 (Day off)
4	2020/10/08	Python自然語言處理 (Python for Natural Language Processing)
5	2020/10/15	處理和理解文本 (Processing and Understanding Text)
6	2020/10/22	文本表達特徵工程 (Feature Engineering for Text Representation)

課程大綱 (Syllabus)

週次 (Week)	日期 (Date)	內容 (Subject/Topics)
7	2020/10/29	人工智慧文本分析個案研究 I (Case Study on Artificial Intelligence for Text Analytics I)
8	2020/11/05	文本分類 (Text Classification)
9	2020/11/12	文本摘要和主題模型 (Text Summarization and Topic Models)
10	2020/11/19	期中報告 (Midterm Project Report)
11	2020/11/26	文本相似度和分群 (Text Similarity and Clustering)
12	2020/12/03	語意分析和命名實體識別 (Semantic Analysis and Named Entity Recognition; NER)

課程大綱 (Syllabus)

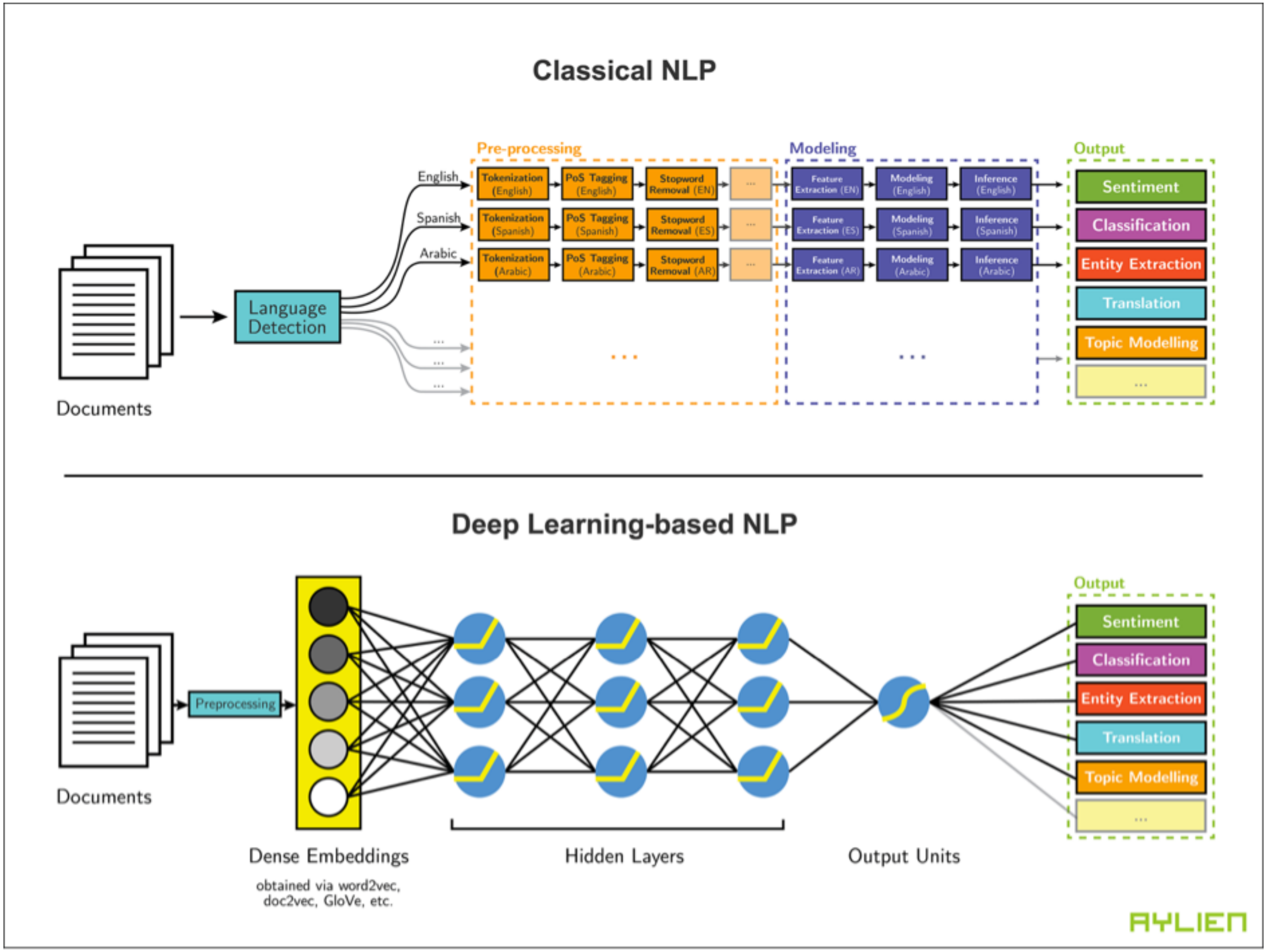
週次 (Week)	日期 (Date)	內容 (Subject/Topics)
13	2020/12/10	情感分析 (Sentiment Analysis)
14	2020/12/17	人工智慧文本分析個案研究 II (Case Study on Artificial Intelligence for Text Analytics II)
15	2020/12/24	深度學習和通用句子嵌入模型 (Deep Learning and Universal Sentence-Embedding Models)
16	2020/12/31	問答系統與對話系統 (Question Answering and Dialogue Systems)
17	2021/01/07	期末報告 I (Final Project Presentation I)
18	2021/01/14	期末報告 II (Final Project Presentation II)

Outline

- Traditional Feature Engineering for Text Data
  - Bag of Words Model
  - Bag of N-Grams Model
  - TF-IDF Model
- Advanced Word Embeddings with Deep Learning
  - Word2Vec Model
  - Robust Word2Vec Models with Gensim
  - GloVe Model
  - FastText Model

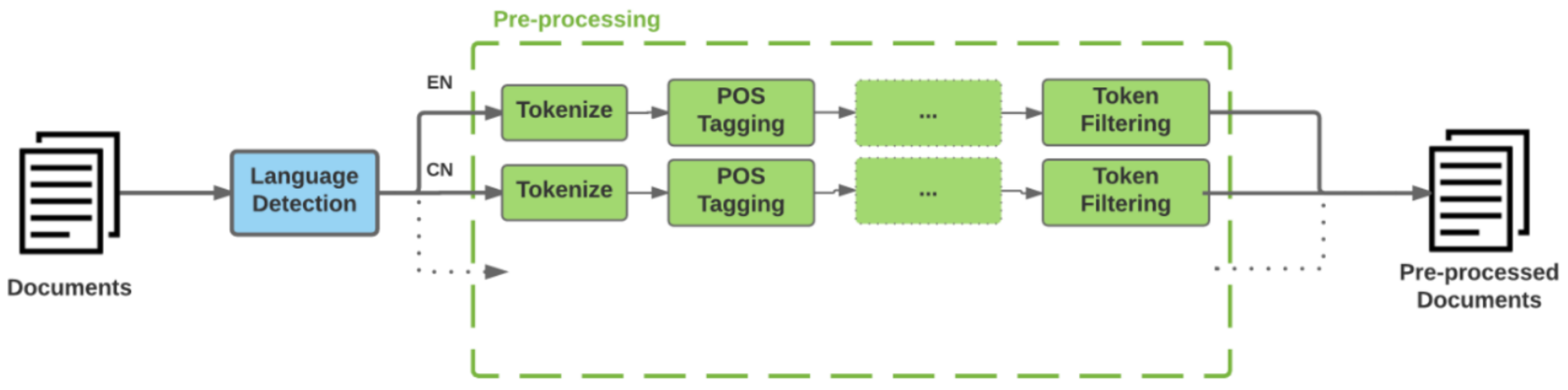
Feature Engineering for Text Representation

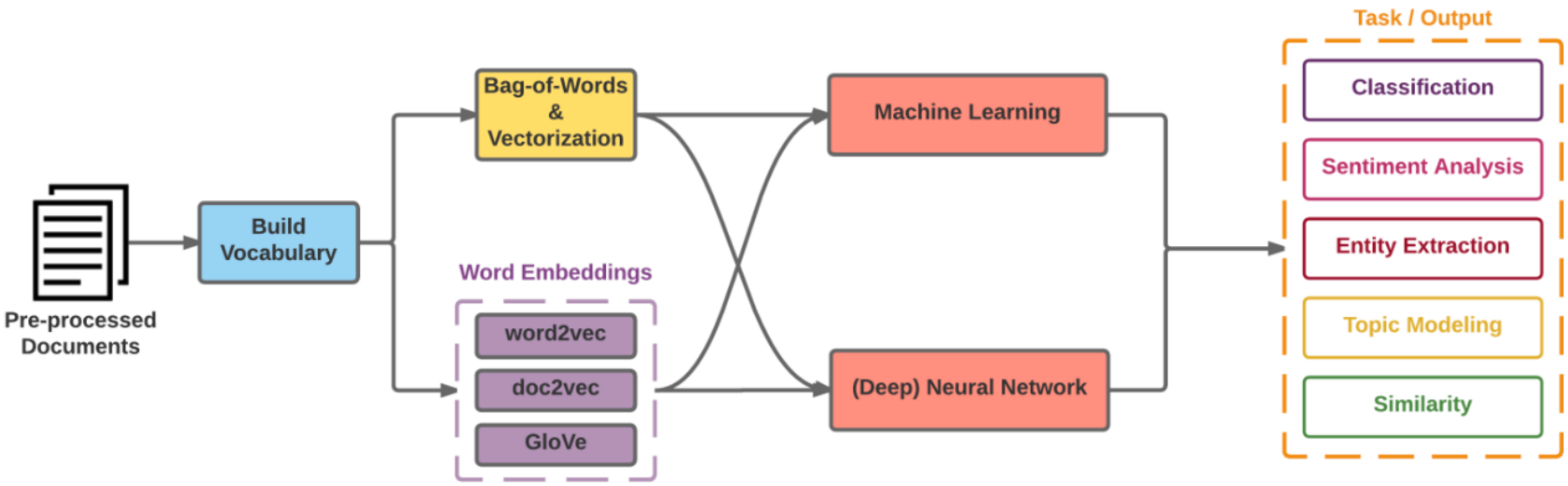
NLP



Source: http://blog.aylien.com/leveraging-deep-learning-for-multilingual/

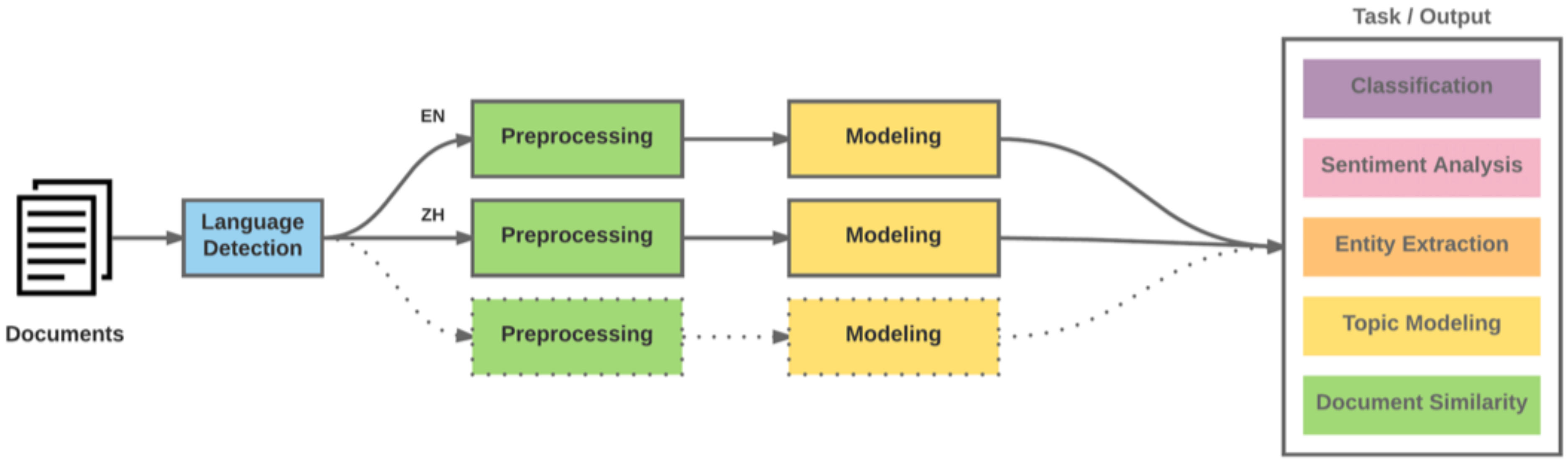
Modern NLP Pipeline





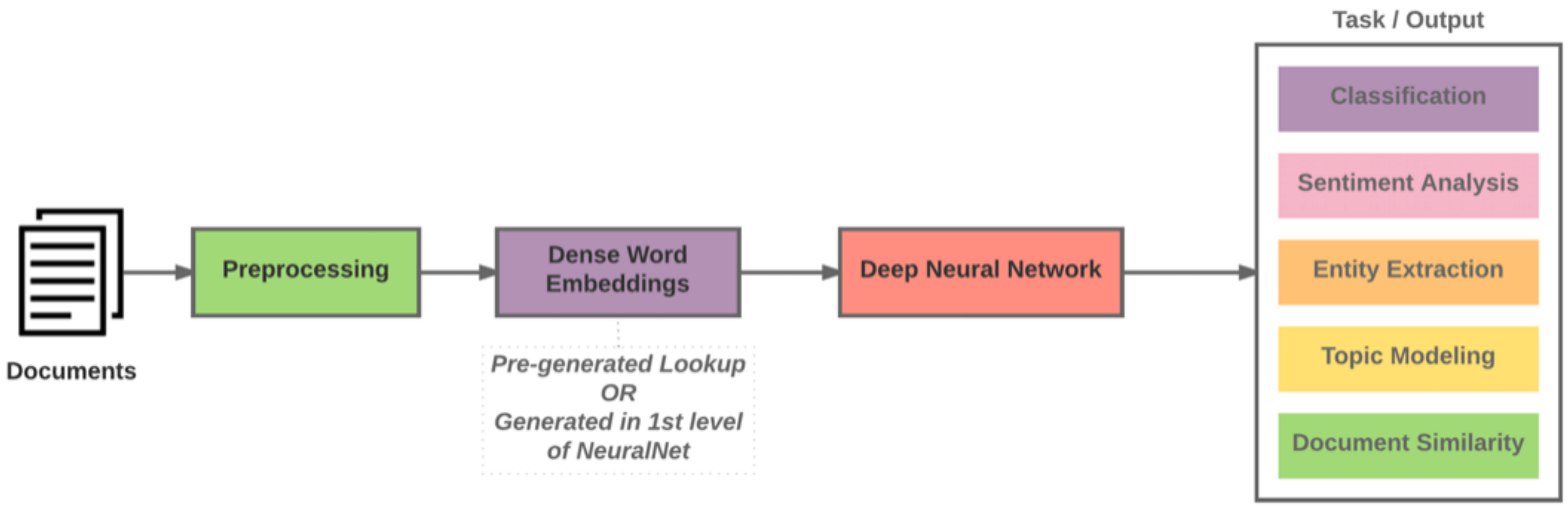
Source: https://github.com/fortiema/talks/blob/master/opendata2016sh/pragmatic-nlp-opendata2016sh.pdf

Modern NLP Pipeline



Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/

Deep Learning NLP



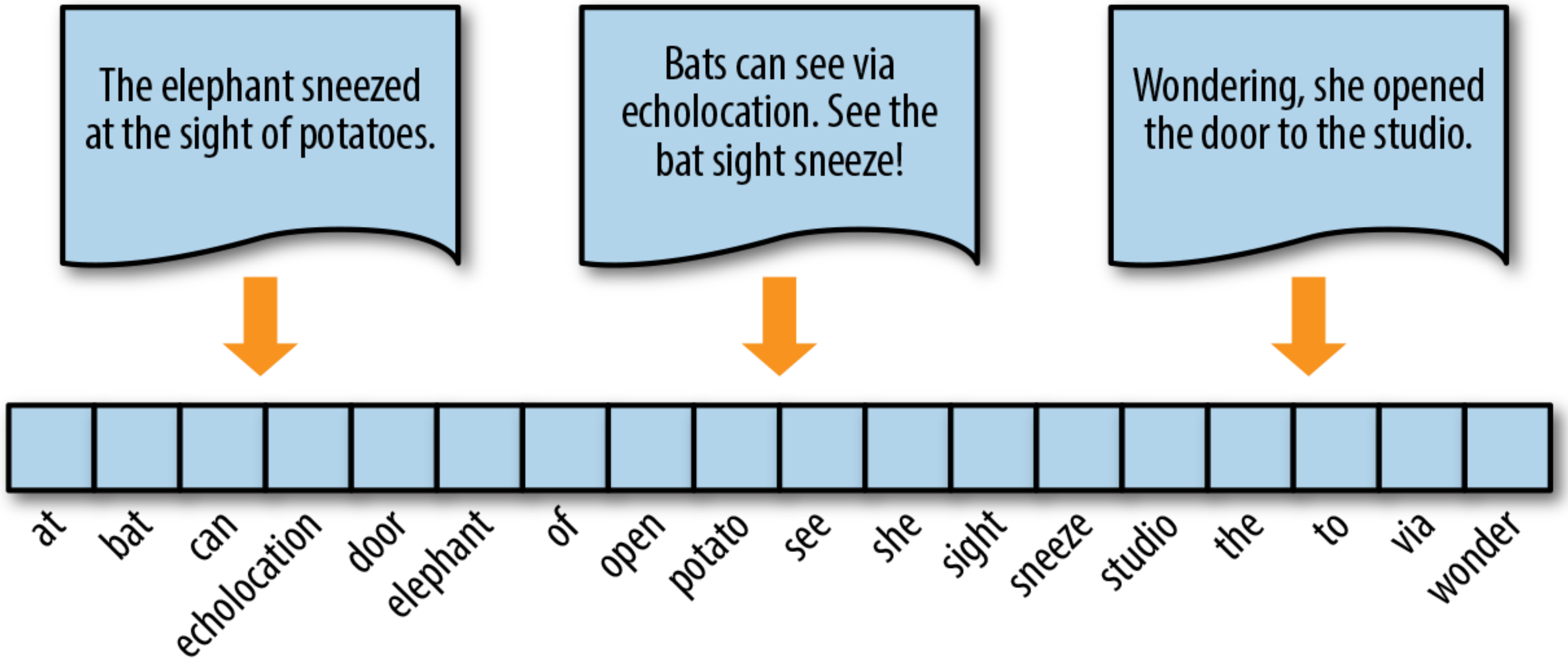
Source: http://mattfortier.me/2017/01/31/nlp-intro-pt-1-overview/

Overview of Text Vectorization Methods

Vectorization Method	Function	Good For	Considerations
Frequency	Counts term frequencies	Bayesian models	Most frequent words not always most informative
One-Hot Encoding	Binarizes term occurrence (0, 1)	Neural networks	All words equidistant, so normalization extra important
TF–IDF	Normalizes term frequencies across documents	General purpose	Moderately frequent terms may not be representative of document topics
Distributed Representations	Context-based, continuous term similarity encoding	Modeling more complex relationships	Performance intensive; difficult to scale without additional tools (e.g., Tensorflow)

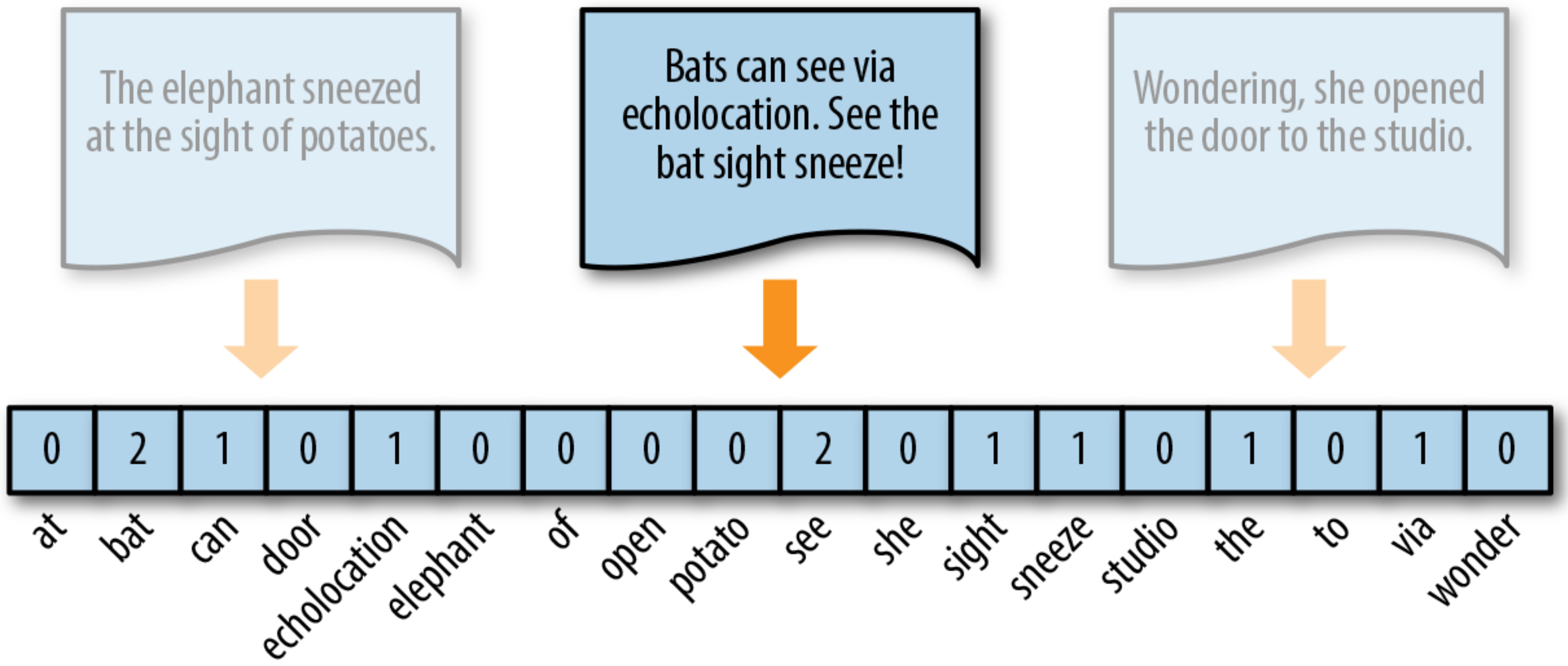
Source: Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda (2018), Applied Text Analysis with Python, O'Reilly Media.

Encoding Documents as Vectors


The elephant sneezed at the sight of potatoes.
Bats can see via echolocation. See the bat sight sneeze!
Wondering, she opened the door to the studio.
at
bat
can
echolocation
door
elephant
of
open
potato
see
she
sight
sneeze
studio
the
to
via
wonder

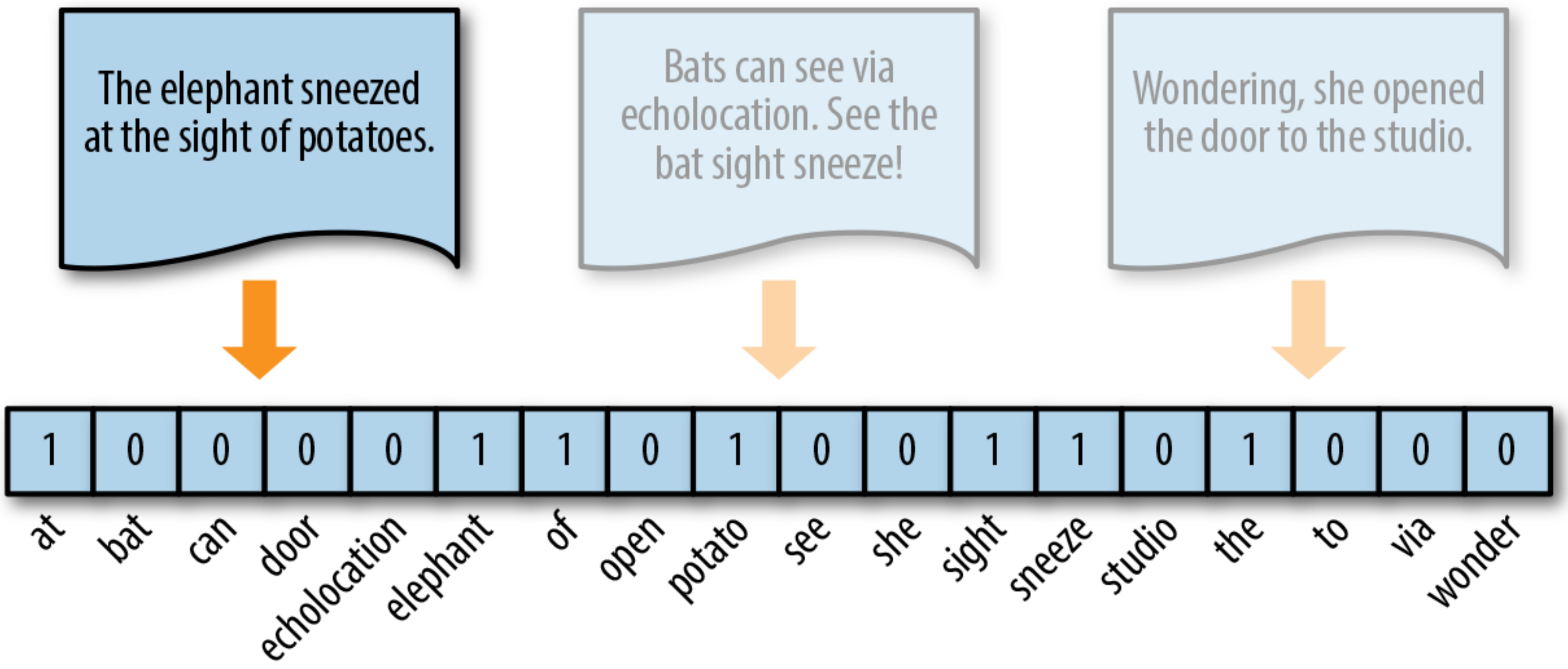
Source: Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda (2018), Applied Text Analysis with Python, O'Reilly Media.

Token Frequency as Vector Encoding

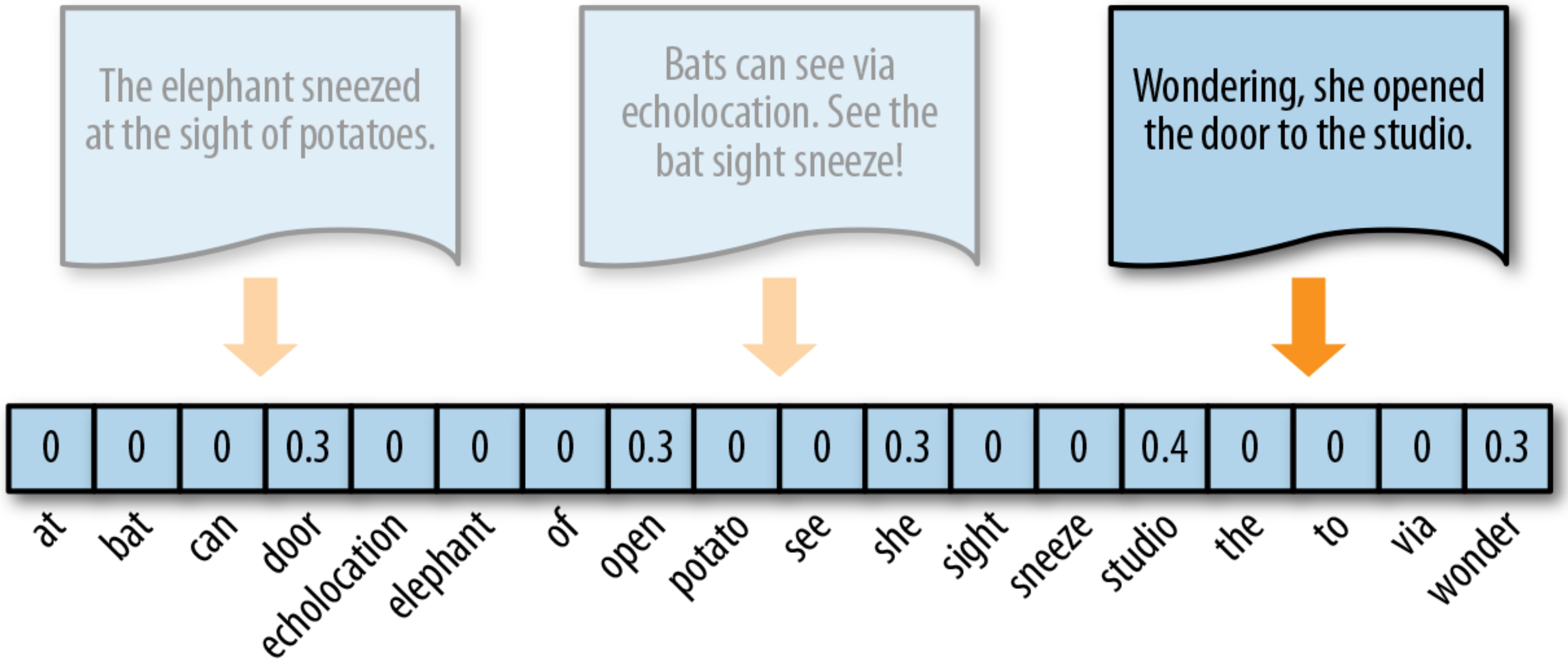


The elephant sneezed at the sight of potatoes.
Bats can see via echolocation. See the bat sight sneeze!
Wondering, she opened the door to the studio.
0
2
1
0
1
0
0
0
0
2
0
1
1
0
1
0
1
0
at
bat
can
door
echolocation
elephant
of
open
potato
see
she
sight
sneeze
studio
the
to
via
wonder

One-hot Encoding

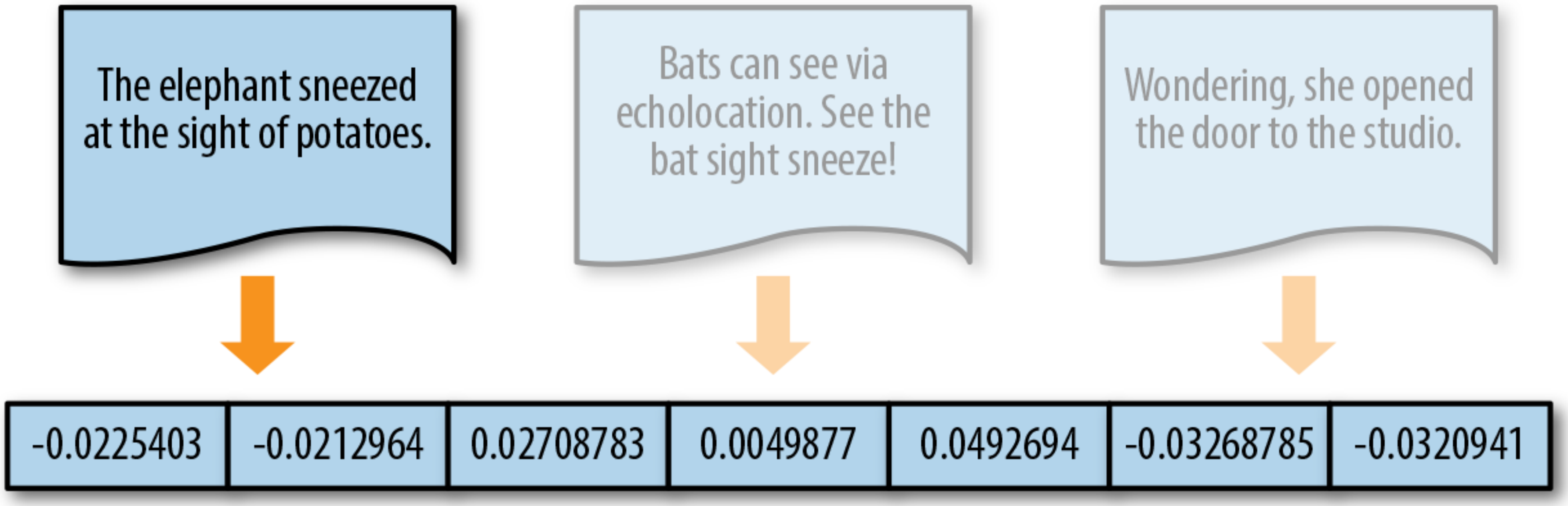

The elephant sneezed at the sight of potatoes.
Bats can see via echolocation. See the bat sight sneeze!
Wondering, she opened the door to the studio.
1 0 0 0 0 1 1 0 1 0 0 1 1 0 1 0 0 0
at
bat
can
door
echolocation
elephant
of
open
potato
see
she
sight
sneeze
studio
the
to
via
wonder

TF-IDF Encoding

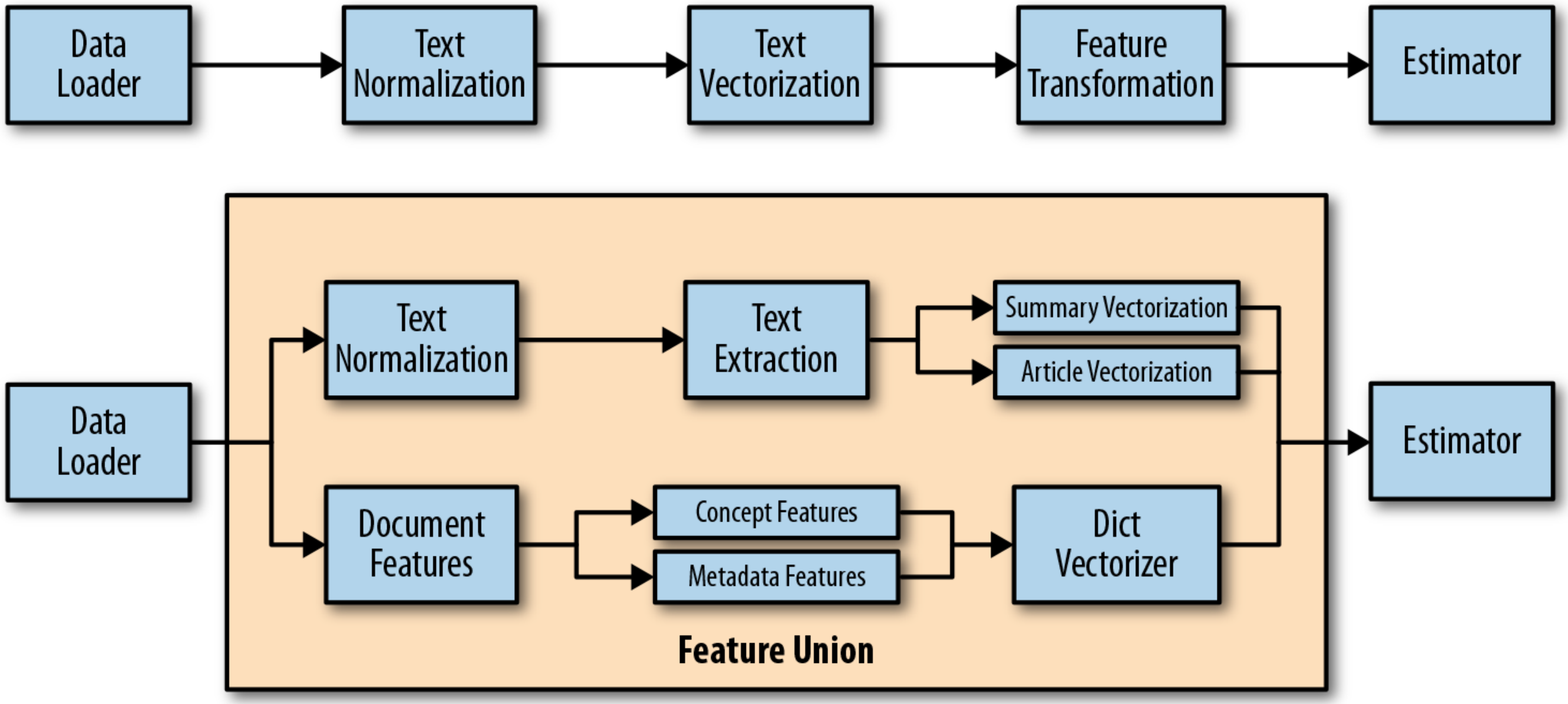

The elephant sneezed at the sight of potatoes.
Bats can see via echolocation. See the bat sight sneeze!
Wondering, she opened the door to the studio.
0
0
0
0.3
0
0
0
0.3
0
0
0.3
0
0
0.4
0
0
0
0.3
at
bat
can
door
echolocation
elephant
of
open
potato
see
she
sight
sneeze
studio
the
to
via
wonder

Source: Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda (2018), Applied Text Analysis with Python, O'Reilly Media.

Distributed Representation

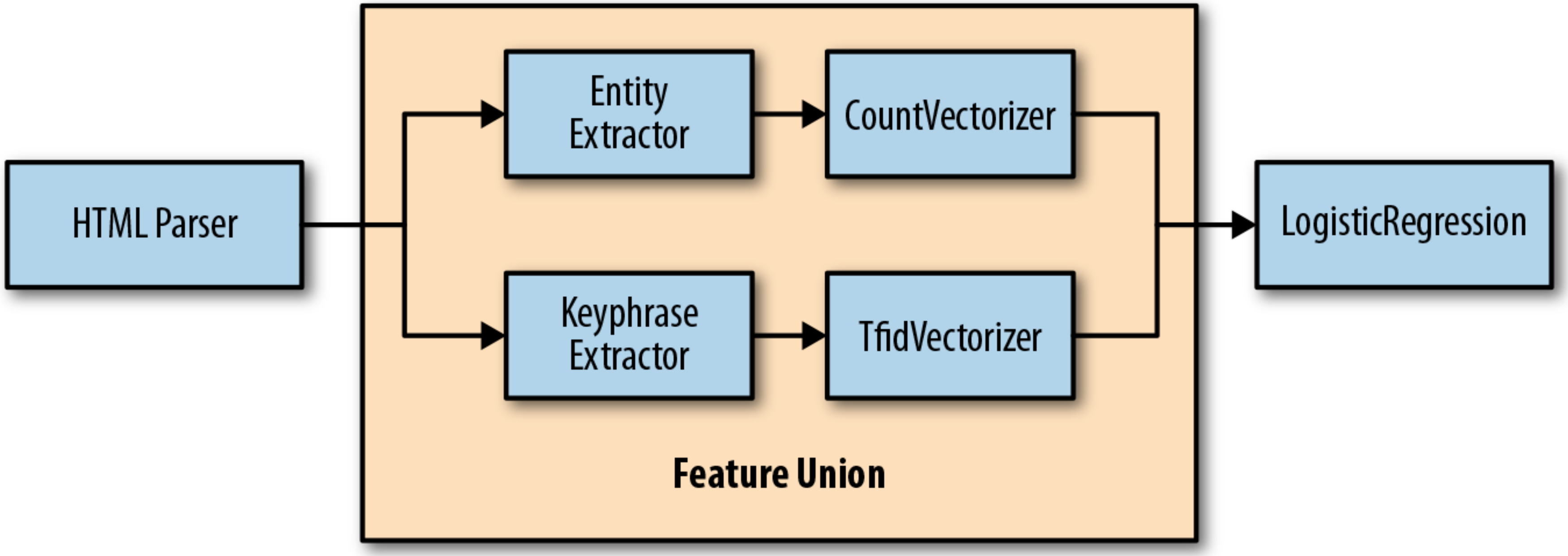

The elephant sneezed at the sight of potatoes.
Bats can see via echolocation. See the bat sight sneeze!
Wondering, she opened the door to the studio.
-0.0225403
-0.0212964
0.02708783
0.0049877
0.0492694
-0.03268785
-0.0320941

Pipelines for Text Vectorization and Feature Extraction

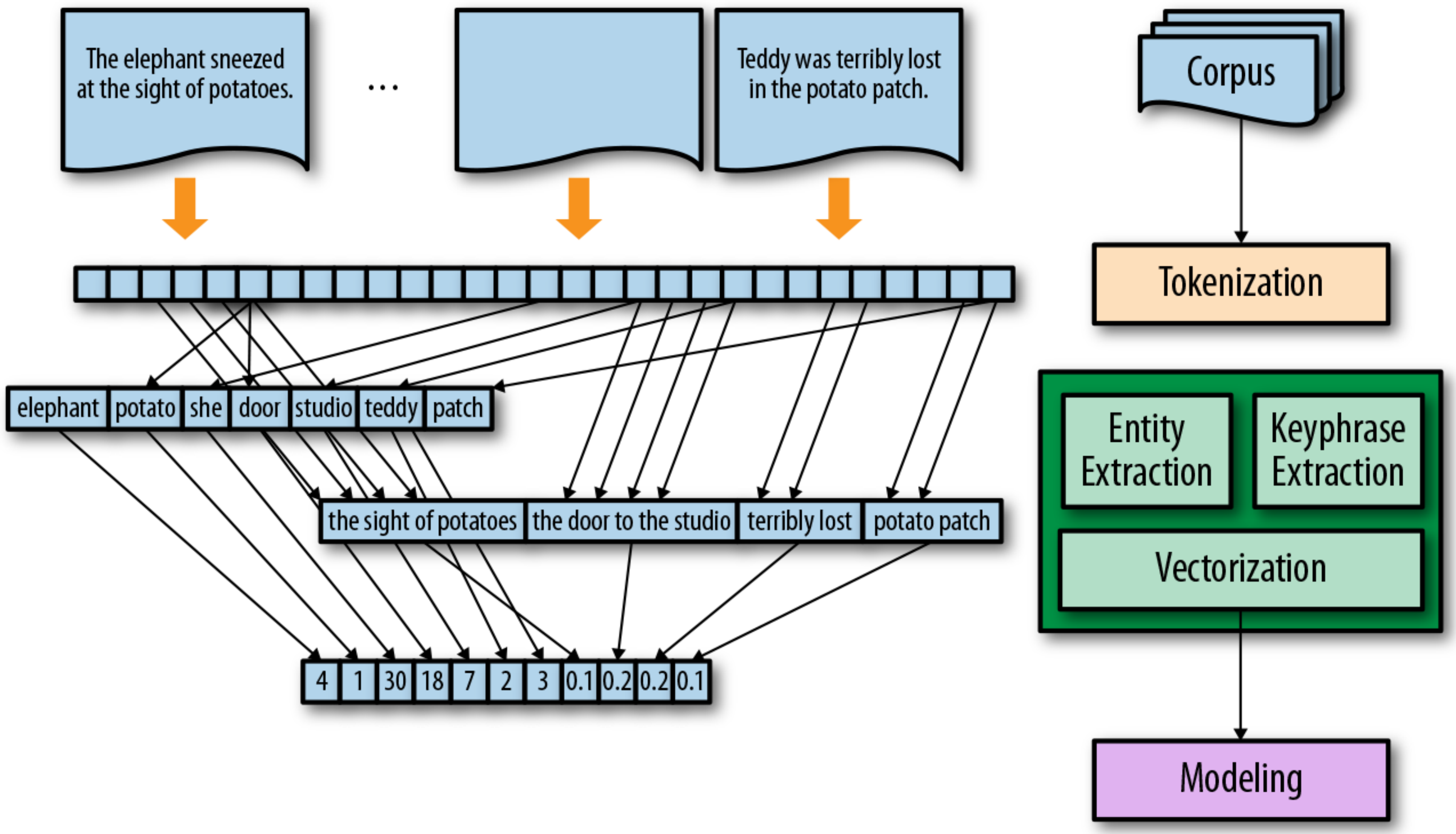


Source: Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda (2018), Applied Text Analysis with Python, O'Reilly Media.

Feature Unions for Branching Vectorization

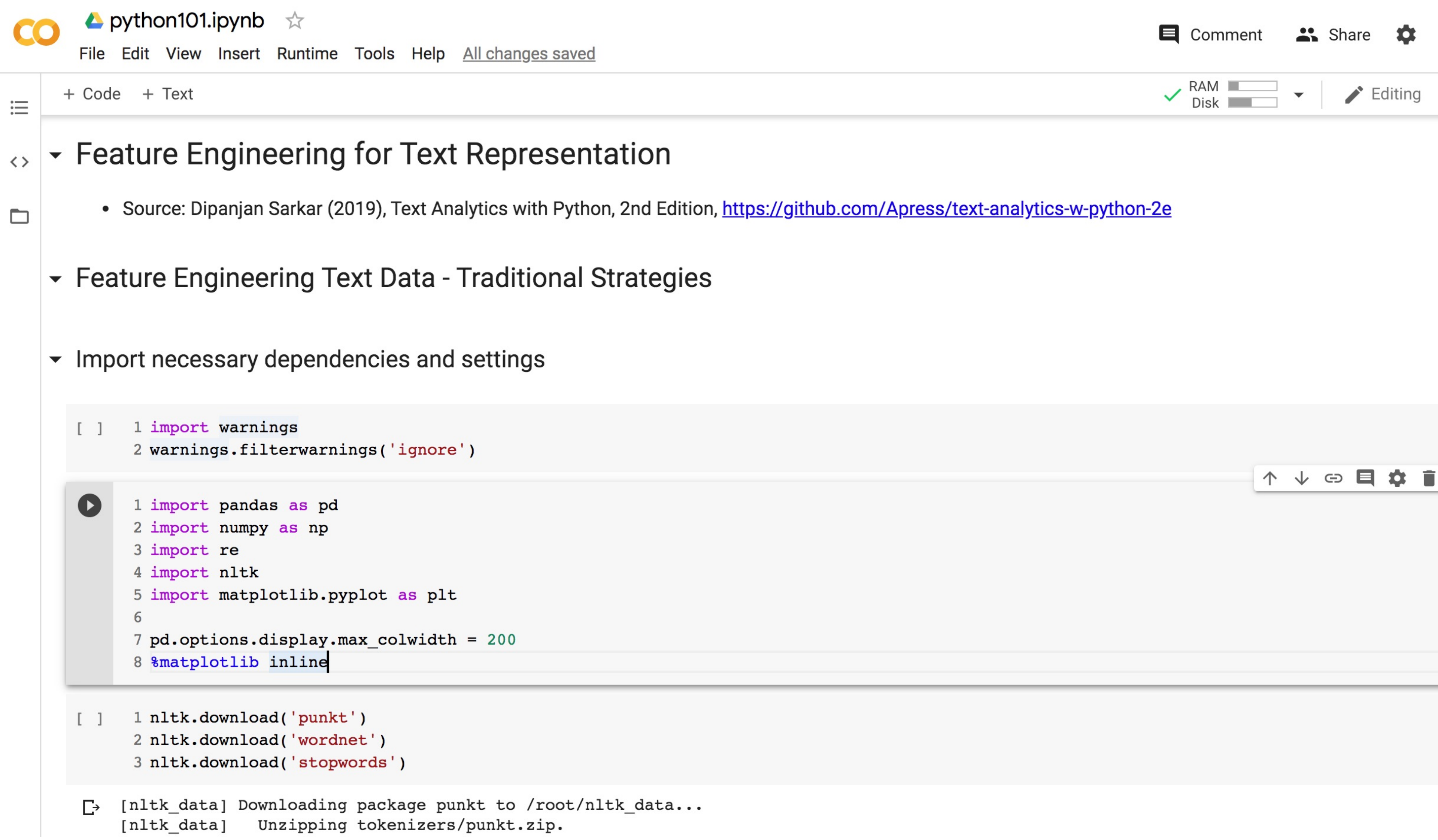


Feature Extraction and Union



Python in Google Colab (Python101)

https://colab.research.google.com/drive/1FEG6DnGvwfUbeo4zJ1zTunjMqf2RkCrT



Feature Engineering for Text Representation

- Source: Dipanjan Sarkar (2019), Text Analytics with Python, 2nd Edition, https://github.com/Apress/text-analytics-w-python-2e

Feature Engineering Text Data - Traditional Strategies

Import necessary dependencies and settings

```
[ ]
import warnings
warnings.filterwarnings('ignore')
```

```
import pandas as pd
import numpy as np
import re
import nltk
import matplotlib.pyplot as plt

pd.options.display.max_colwidth = 200
%matplotlib inline
```

```
[ ]
nltk.download('punkt')
nltk.download('wordnet')
nltk.download('stopwords')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
```

https://tinyurl.com/aintpupython101

```
corpus = ['The sky is blue and beautiful.',
'Love this blue and beautiful sky!',
'The quick brown fox jumps over the lazy dog.',
"A king's breakfast has sausages, ham, bacon, eggs, toast and
beans",
'I love green eggs, ham, sausages and bacon!',
'The brown fox is quick and the blue dog is lazy!',
'The sky is very blue and the sky is very beautiful today',
'The dog is lazy but the brown fox is quick!'
]
labels = ['weather', 'weather', 'animals', 'food', 'food',
'animals', 'weather', 'animals']

corpus = np.array(corpus)
corpus_df = pd.DataFrame({'Document': corpus,
'Category': labels})
corpus_df = corpus_df[['Document', 'Category']]
corpus_df
```

https://tinyurl.com/aintpupython101

```
corpus = np.array(corpus)
corpus_df = pd.DataFrame({'Document': corpus,
'Category': labels})
corpus_df = corpus_df[['Document', 'Category']]
corpus_df
```

	Document	Category
0	The sky is blue and beautiful.	weather
1	Love this blue and beautiful sky!	weather
2	The quick brown fox jumps over the lazy dog.	animals
3	A king's breakfast has sausages, ham, bacon, eggs, toast and beans	food
4	I love green eggs, ham, sausages and bacon!	food
5	The brown fox is quick and the blue dog is lazy!	animals
6	The sky is very blue and the sky is very beautiful today	weather
7	The dog is lazy but the brown fox is quick!	animals

https://tinyurl.com/aintpupython101

```
wpt = nltk.WordPunctTokenizer()
stop_words = nltk.corpus.stopwords.words('english')

def normalize_document(doc):
# lower case and remove special characters\whitespaces
doc = re.sub(r'[^a-zA-Z\s]', '', doc, re.I|re.A)
doc = doc.lower()
doc = doc.strip()
# tokenize document
tokens = wpt.tokenize(doc)
# filter stopwords out of document
filtered_tokens = [token for token in tokens if token not in
stop_words]
# re-create document from filtered tokens
doc = ' '.join(filtered_tokens)
return doc

normalize_corpus = np.vectorize(normalize_document)
norm_corpus = normalize_corpus(corpus)
norm_corpus
```

```python
from sklearn.feature_extraction.text import CountVectorizer
# get bag of words features in sparse format
cv = CountVectorizer(min_df=0., max_df=1.)
cv_matrix = cv.fit_transform(norm_corpus)
cv_matrix
```

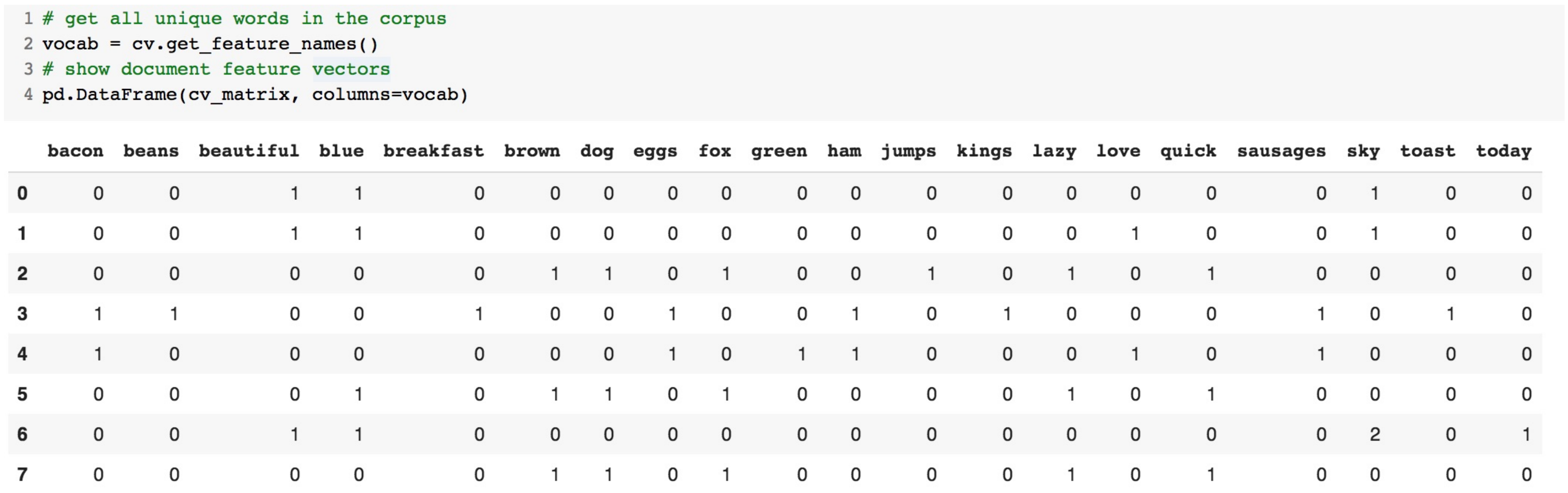
```python
# view non-zero feature positions in the sparse matrix
print(cv_matrix)
```

```python
# view dense representation
# warning might give a memory error if data is too big
cv_matrix = cv_matrix.toarray()
cv_matrix
```

```
array([
[0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0],
[0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0],
[0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0],
[1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 0, 0, 1, 0, 1, 0],
[1, 0, 0, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0],
[0, 0, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0],
[0, 0, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 2, 0, 1],
[0, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0]])
```

https://tinyurl.com/aintpupython101

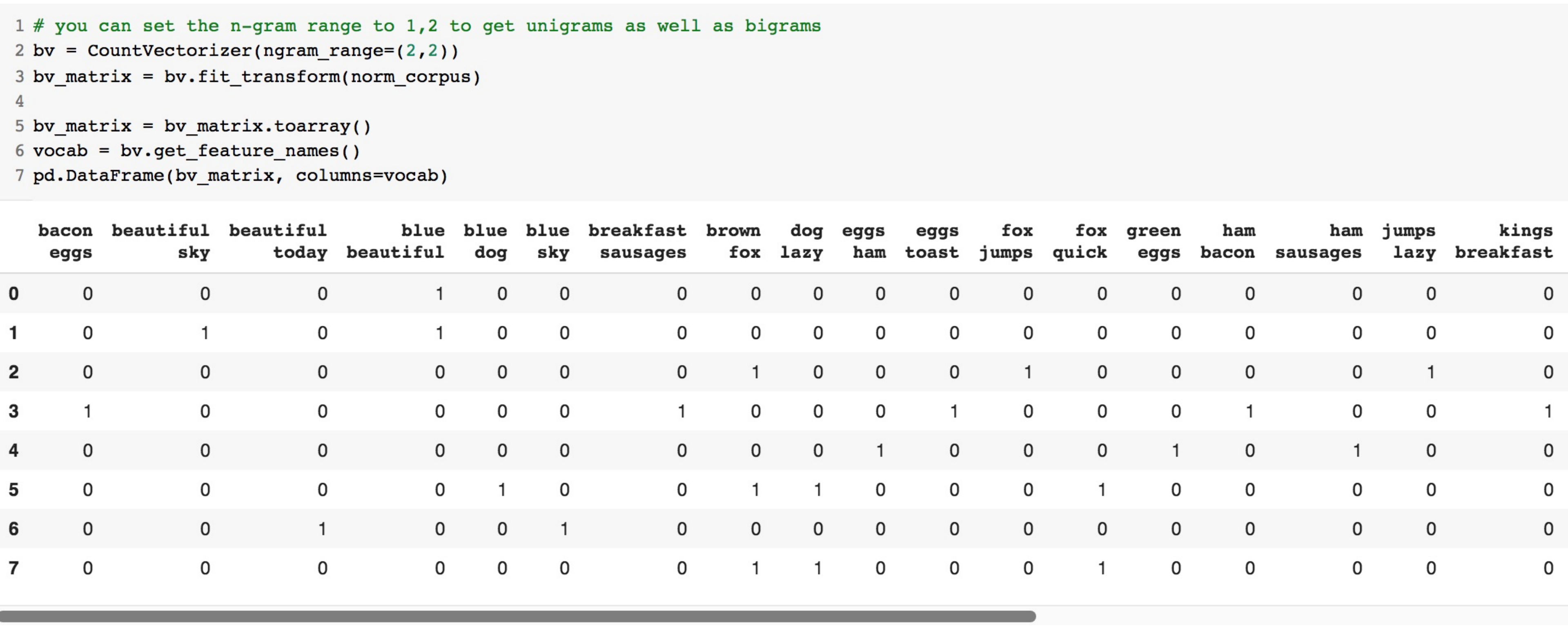
```
# get all unique words in the corpus
vocab = cv.get_feature_names()
# show document feature vectors
pd.DataFrame(cv_matrix, columns=vocab)
```



```
# get all unique words in the corpus
vocab = cv.get_feature_names()
# show document feature vectors
pd.DataFrame(cv_matrix, columns=vocab)
```

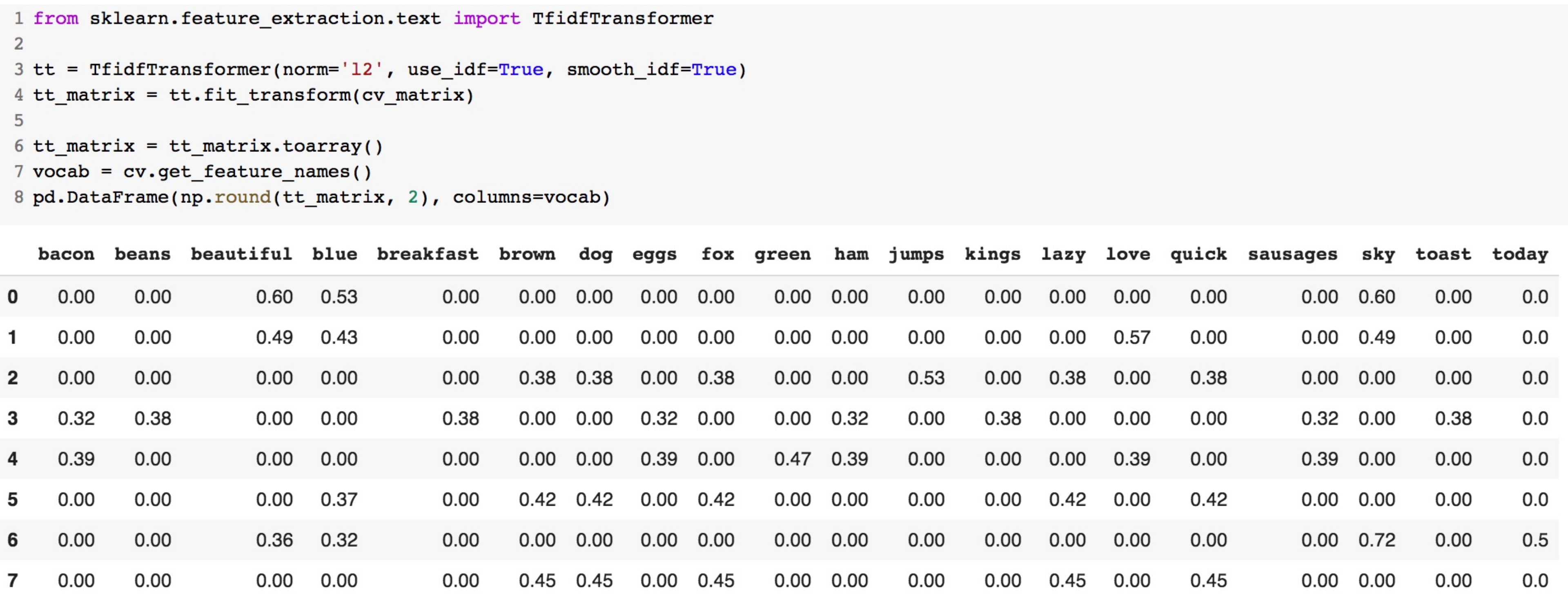
```
   bacon  beans  beautiful  blue  breakfast  brown  dog  eggs  fox  green  ham  jumps  kings  lazy  love  quick  sausages  sky  toast  today
0      0      0          1     1          0      0    0     0    0      0    0      0      0     0     0      0         0    1      0      0
1      0      0          1     1          0      0    0     0    0      0    0      0      0     0     1      0         0    1      0      0
2      0      0          0     0          0      1    1     0    1      0    0      1      0     1     0      1         0    0      0      0
3      1      1          0     0          1      0    0     1    0      0    1      0      1     0     0      0         1    0      1      0
4      1      0          0     0          0      0    0     1    0      1    1      0      0     0     1      0         1    0      0      0
5      0      0          0     1          0      1    1     0    1      0    0      0      0     1     0      1         0    0      0      0
6      0      0          1     1          0      0    0     0    0      0    0      0      0     0     0      0         0    2      0      1
7      0      0          0     0          0      1    1     0    1      0    0      0      0     1     0      1         0    0      0      0
```

https://tinyurl.com/aintpupython101



```python
# you can set the n-gram range to 1,2 to get unigrams as well as bigrams
bv = CountVectorizer(ngram_range=(2,2))
bv_matrix = bv.fit_transform(norm_corpus)

bv_matrix = bv_matrix.toarray()
vocab = bv.get_feature_names()
pd.DataFrame(bv_matrix, columns=vocab)
```

	bacon eggs	beautiful sky	beautiful today	blue beautiful	blue dog	blue sky	breakfast sausages	brown fox	dog lazy	eggs ham	eggs toast	fox jumps	fox quick	green eggs	ham bacon	ham sausages	jumps lazy	kings breakfast
0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	1	0
3	1	0	0	0	0	0	1	0	0	0	1	0	0	0	1	0	0	1
4	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0	0
5	0	0	0	0	1	0	0	1	1	0	0	0	1	0	0	0	0	0
6	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	1	1	0	0	0	1	0	0	0	0	0



```python
from sklearn.feature_extraction.text import TfidfTransformer

tt = TfidfTransformer(norm='l2', use_idf=True, smooth_idf=True)
tt_matrix = tt.fit_transform(cv_matrix)

tt_matrix = tt_matrix.toarray()
vocab = cv.get_feature_names()
pd.DataFrame(np.round(tt_matrix, 2), columns=vocab)
```

	bacon	beans	beautiful	blue	breakfast	brown	dog	eggs	fox	green	ham	jumps	kings	lazy	love	quick	sausages	sky	toast	today
0	0.00	0.00	0.60	0.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.60	0.00	0.0
1	0.00	0.00	0.49	0.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.57	0.00	0.00	0.49	0.00	0.0
2	0.00	0.00	0.00	0.00	0.00	0.38	0.38	0.00	0.38	0.00	0.00	0.53	0.00	0.38	0.00	0.38	0.00	0.00	0.00	0.0
3	0.32	0.38	0.00	0.00	0.38	0.00	0.00	0.32	0.00	0.00	0.32	0.00	0.38	0.00	0.00	0.00	0.32	0.00	0.38	0.0
4	0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.39	0.00	0.47	0.39	0.00	0.00	0.00	0.39	0.00	0.39	0.00	0.00	0.0
5	0.00	0.00	0.00	0.37	0.00	0.42	0.42	0.00	0.42	0.00	0.00	0.00	0.00	0.42	0.00	0.42	0.00	0.00	0.00	0.0
6	0.00	0.00	0.36	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.72	0.00	0.5
7	0.00	0.00	0.00	0.00	0.00	0.45	0.45	0.00	0.45	0.00	0.00	0.00	0.00	0.45	0.00	0.45	0.00	0.00	0.00	0.0

https://tinyurl.com/aintpupython101

```python
from sklearn.feature_extraction.text import TfidfVectorizer

tv = TfidfVectorizer(min_df=0., max_df=1., norm='l2',
                     use_idf=True, smooth_idf=True)
tv_matrix = tv.fit_transform(norm_corpus)
tv_matrix = tv_matrix.toarray()

vocab = tv.get_feature_names()
pd.DataFrame(np.round(tv_matrix, 2), columns=vocab)
```

	bacon	beans	beautiful	blue	breakfast	brown	dog	eggs	fox	green	ham	jumps	kings	lazy	love	quick	sausages	sky	toast	today
0	0.00	0.00	0.60	0.53	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.60	0.00	0.0
1	0.00	0.00	0.49	0.43	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.57	0.00	0.00	0.49	0.00	0.0
2	0.00	0.00	0.00	0.00	0.00	0.38	0.38	0.00	0.38	0.00	0.00	0.53	0.00	0.38	0.00	0.38	0.00	0.00	0.00	0.0
3	0.32	0.38	0.00	0.00	0.38	0.00	0.00	0.32	0.00	0.00	0.32	0.00	0.38	0.00	0.00	0.00	0.32	0.00	0.38	0.0
4	0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.39	0.00	0.47	0.39	0.00	0.00	0.00	0.39	0.00	0.39	0.00	0.00	0.0
5	0.00	0.00	0.00	0.37	0.00	0.42	0.42	0.00	0.42	0.00	0.00	0.00	0.00	0.42	0.00	0.42	0.00	0.00	0.00	0.0
6	0.00	0.00	0.36	0.32	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.72	0.00	0.5
7	0.00	0.00	0.00	0.00	0.00	0.45	0.45	0.00	0.45	0.00	0.00	0.00	0.00	0.45	0.00	0.45	0.00	0.00	0.00	0.0

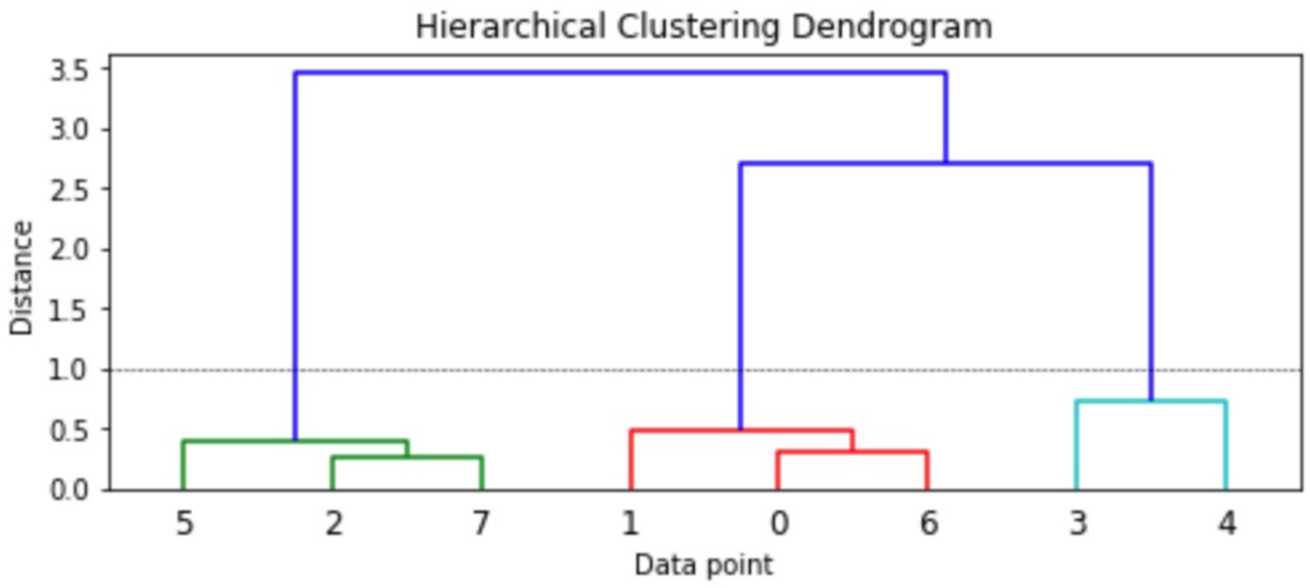
https://tinyurl.com/aintpupython101

```python
from scipy.cluster.hierarchy import dendrogram, linkage

Z = linkage(similarity_matrix, 'ward')
pd.DataFrame(Z, columns=['Document Cluster 1', 'Document Cluster 2',
                         'Distance', 'Cluster Size'], dtype='object')
```

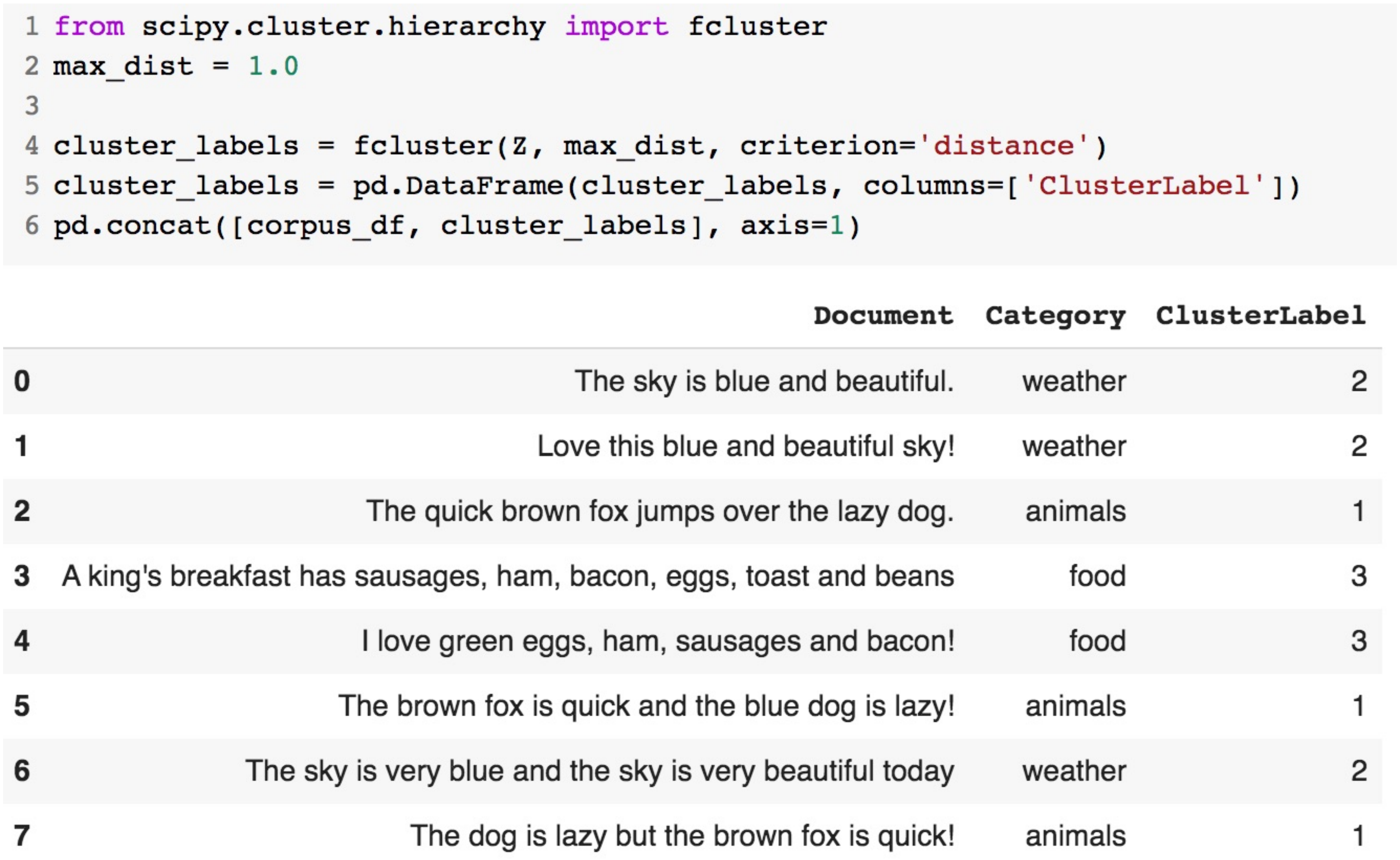
	Document Cluster 1	Document Cluster 2	Distance	Cluster Size
0	2	7	0.253098	2
1	0	6	0.308539	2
2	5	8	0.386952	3
3	1	9	0.489845	3
4	3	4	0.732945	2
5	11	12	2.69565	5
6	10	13	3.45108	8

```python
plt.figure(figsize=(8, 3))
plt.title('Hierarchical Clustering Dendrogram')
plt.xlabel('Data point')
plt.ylabel('Distance')
dendrogram(Z)
plt.axhline(y=1.0, c='k', ls='--', lw=0.5)
```

```
<matplotlib.lines.Line2D at 0x7ff7b5d793c8>
```



https://tinyurl.com/aintpupython101



```python
from scipy.cluster.hierarchy import fcluster
max_dist = 1.0

cluster_labels = fcluster(Z, max_dist, criterion='distance')
cluster_labels = pd.DataFrame(cluster_labels, columns=['ClusterLabel'])
pd.concat([corpus_df, cluster_labels], axis=1)
```

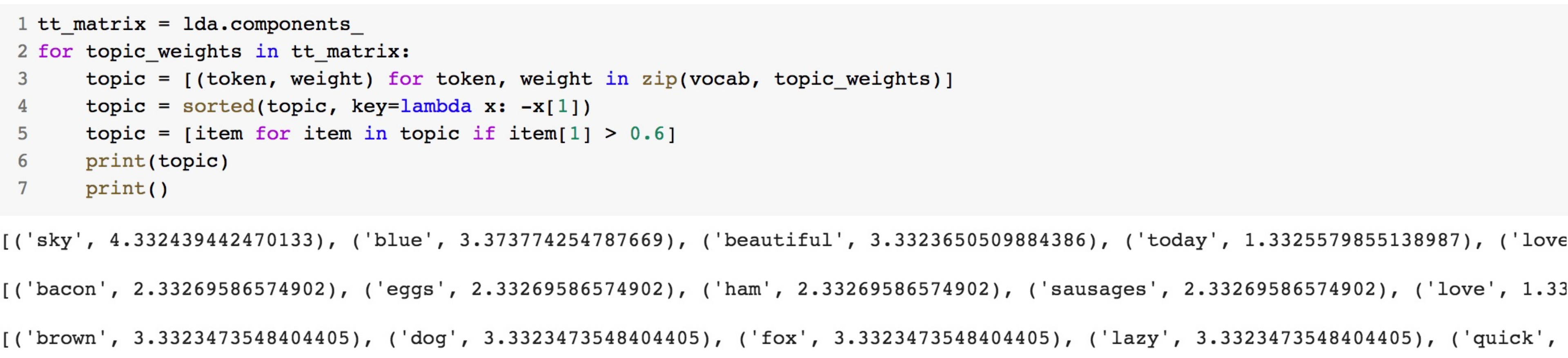
	Document	Category	ClusterLabel
0	The sky is blue and beautiful.	weather	2
1	Love this blue and beautiful sky!	weather	2
2	The quick brown fox jumps over the lazy dog.	animals	1
3	A king's breakfast has sausages, ham, bacon, eggs, toast and beans	food	3
4	I love green eggs, ham, sausages and bacon!	food	3
5	The brown fox is quick and the blue dog is lazy!	animals	1
6	The sky is very blue and the sky is very beautiful today	weather	2
7	The dog is lazy but the brown fox is quick!	animals	1

https://tinyurl.com/aintpupython101

```python
from sklearn.decomposition import LatentDirichletAllocation
lda = LatentDirichletAllocation(n_components=3, max_iter=10000, random_state=0)
#lda = LatentDirichletAllocation(n_topics=3, max_iter=10000, random_state=0)
dt_matrix = lda.fit_transform(cv_matrix)
features = pd.DataFrame(dt_matrix, columns=['T1', 'T2', 'T3'])
features
```

	T1	T2	T3
0	0.832191	0.083480	0.084329
1	0.863554	0.069100	0.067346
2	0.047794	0.047776	0.904430
3	0.037243	0.925559	0.037198
4	0.049121	0.903076	0.047802
5	0.054902	0.047778	0.897321
6	0.888287	0.055697	0.056016
7	0.055704	0.055689	0.888607

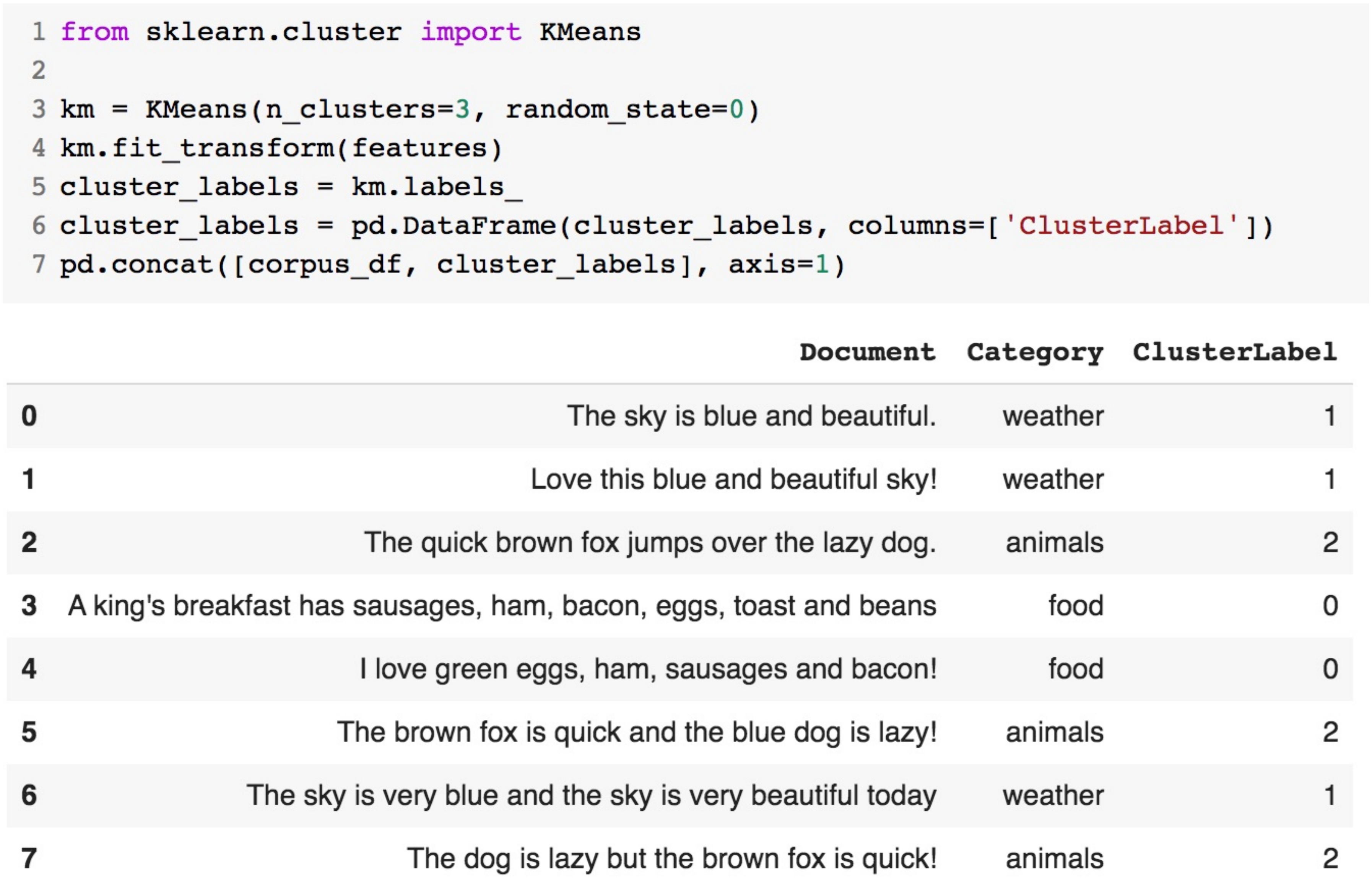
https://tinyurl.com/aintpupython101



```python
tt_matrix = lda.components_
for topic_weights in tt_matrix:
    topic = [(token, weight) for token, weight in zip(vocab, topic_weights)]
    topic = sorted(topic, key=lambda x: -x[1])
    topic = [item for item in topic if item[1] > 0.6]
    print(topic)
    print()
```

```
[('sky', 4.332439442470133), ('blue', 3.373774254787669), ('beautiful', 3.3323650509884386), ('today', 1.3325579855138987), ('love

[('bacon', 2.33269586574902), ('eggs', 2.33269586574902), ('ham', 2.33269586574902), ('sausages', 2.33269586574902), ('love', 1.33

[('brown', 3.3323473548404405), ('dog', 3.3323473548404405), ('fox', 3.3323473548404405), ('lazy', 3.3323473548404405), ('quick',
```

https://tinyurl.com/aintpupython101



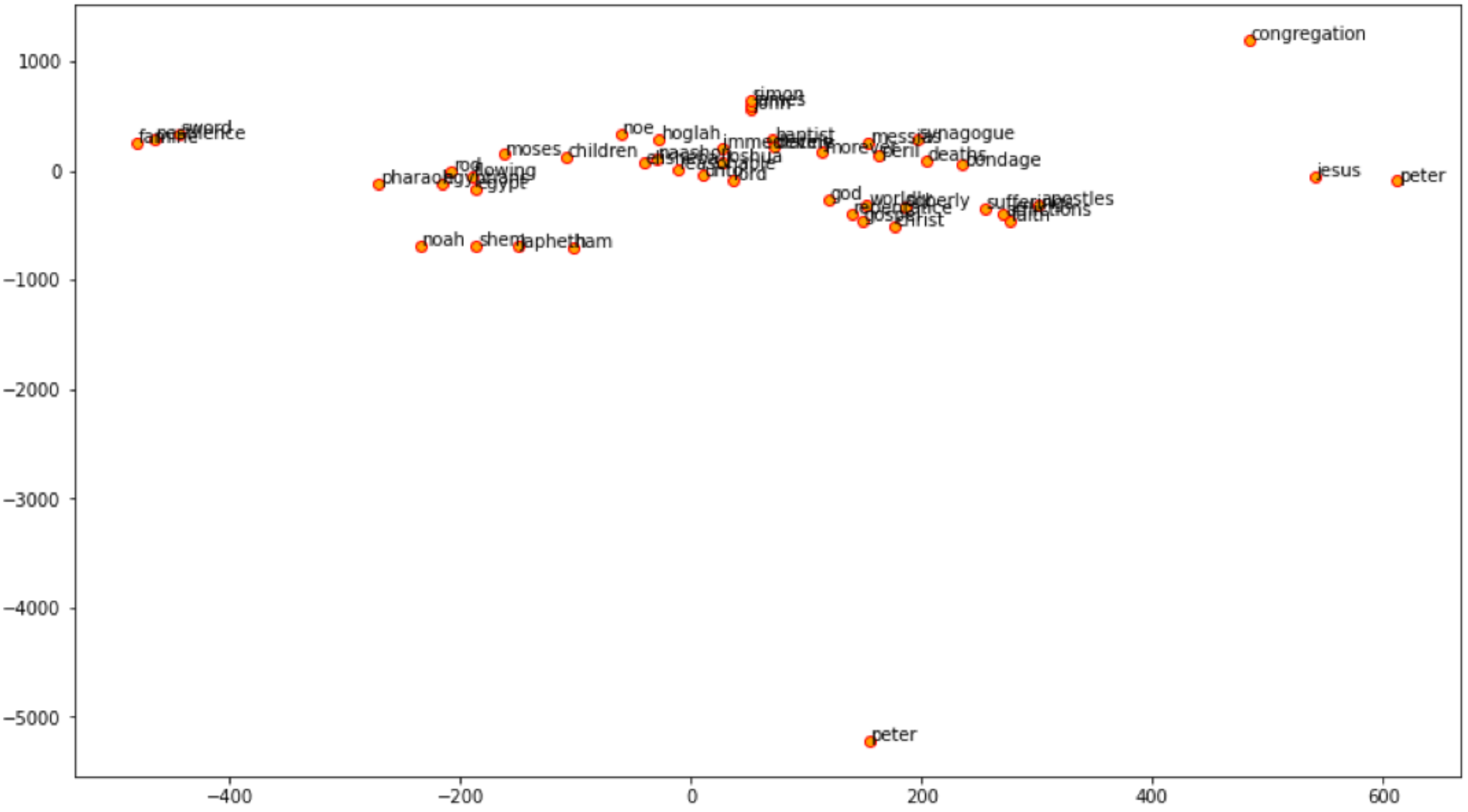
```python
from sklearn.cluster import KMeans

km = KMeans(n_clusters=3, random_state=0)
km.fit_transform(features)
cluster_labels = km.labels_
cluster_labels = pd.DataFrame(cluster_labels, columns=['ClusterLabel'])
pd.concat([corpus_df, cluster_labels], axis=1)
```

	Document	Category	ClusterLabel
0	The sky is blue and beautiful.	weather	1
1	Love this blue and beautiful sky!	weather	1
2	The quick brown fox jumps over the lazy dog.	animals	2
3	A king's breakfast has sausages, ham, bacon, eggs, toast and beans	food	0
4	I love green eggs, ham, sausages and bacon!	food	0
5	The brown fox is quick and the blue dog is lazy!	animals	2
6	The sky is very blue and the sky is very beautiful today	weather	1
7	The dog is lazy but the brown fox is quick!	animals	2

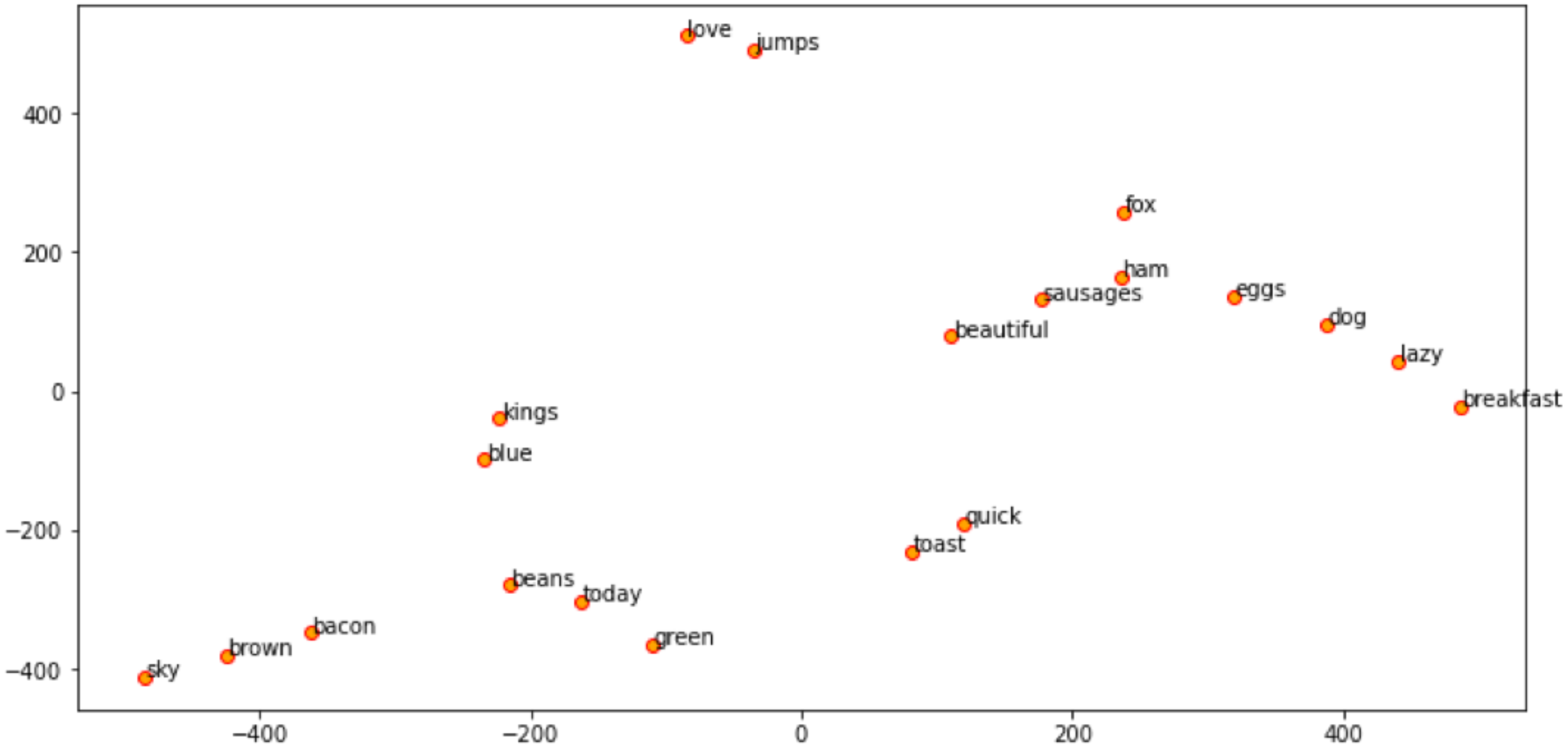
https://tinyurl.com/aintpupython101

```
from gensim.models import word2vec

# tokenize sentences in corpus
wpt = nltk.WordPunctTokenizer()
tokenized_corpus = [wpt.tokenize(document) for document in norm_bible]

# Set values for various parameters
feature_size = 100 # Word vector dimensionality
window_context = 30 # Context window size
min_word_count = 1 # Minimum word count
sample = 1e-3 # Downsample setting for frequent words

w2v_model = word2vec.Word2Vec(tokenized_corpus, size=feature_size,
window=window_context, min_count=min_word_count,
sample=sample, iter=50)

# view similar words based on gensim's model
similar_words = {search_term: [item[0] for item in
w2v_model.wv.most_similar([search_term], topn=5)]
for search_term in ['god', 'jesus', 'noah', 'egypt', 'john', 'gospel',
'moses','famine']}
similar_words
```

https://tinyurl.com/aintpupython101



https://tinyurl.com/aintpupython101

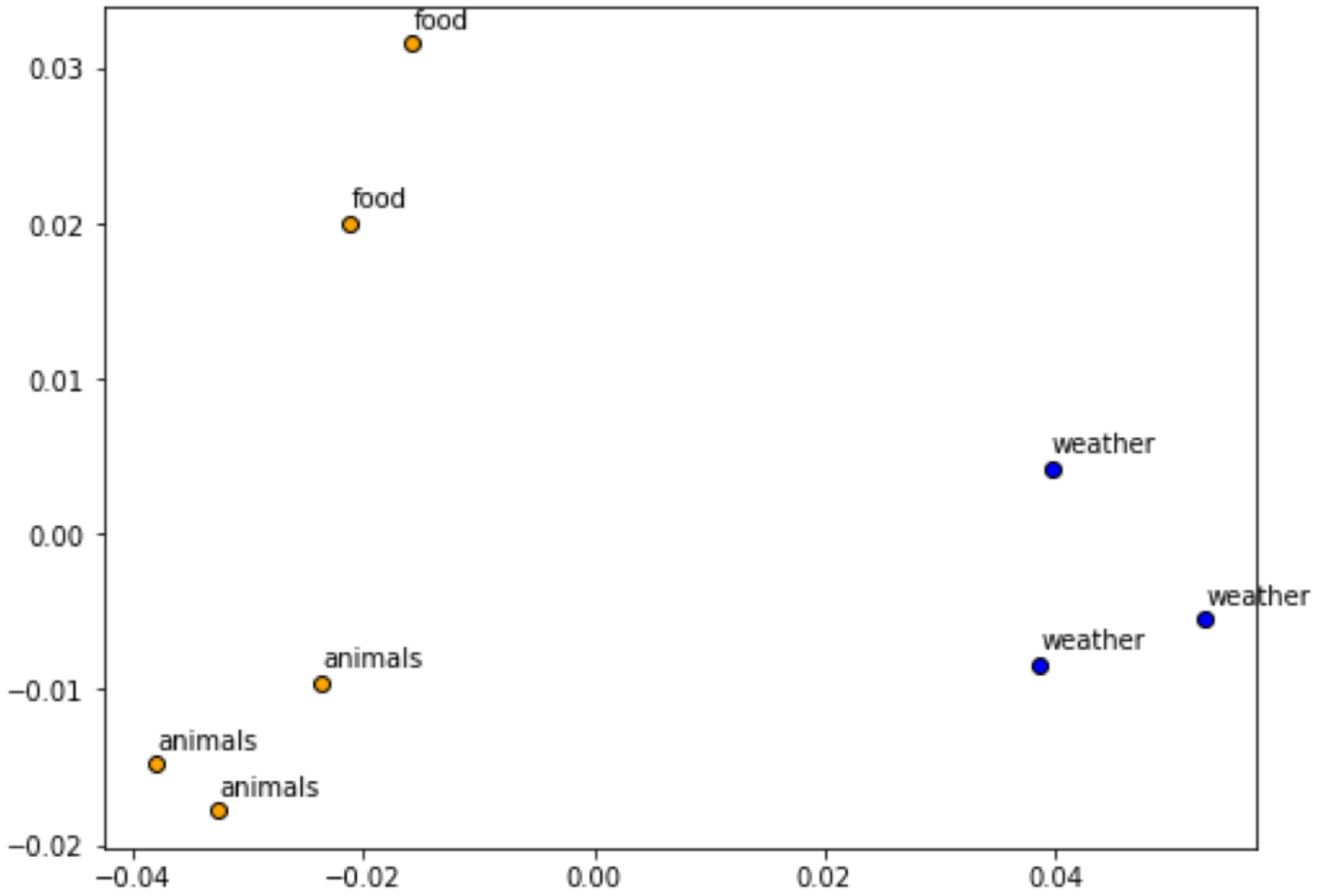


https://tinyurl.com/aintpupython101

```
w2v_model.wv.most_similar([search_term], topn=5)]
```

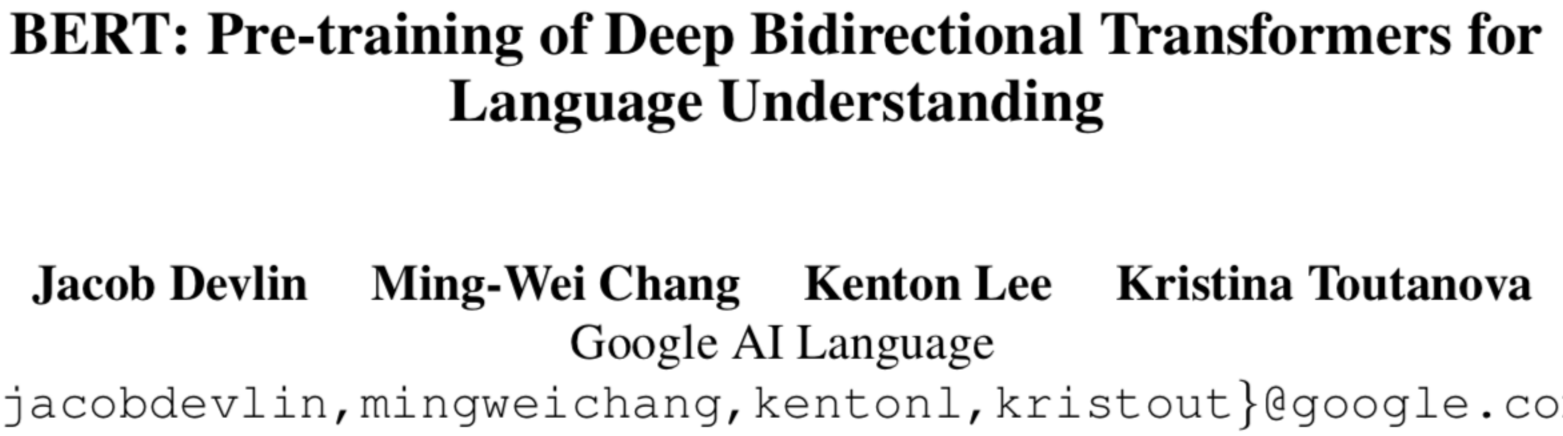
```
{'egypt': ['egyptians', 'pharaoh', 'bondage', 'flowing',
'rod'], 'famine': ['pestilence', 'peril', 'deaths',
'morever', 'sword'], 'god': ['lord', 'worldly', 'soberly',
'reasonable', 'unto'], 'gospel': ['christ', 'faith',
'repentance', 'sufferings', 'afflictions'], 'jesus':
['peter', 'messias', 'immediately', 'apostles',
'synagogue'], 'john': ['james', 'baptist', 'devine',
'peter', 'simon'], 'moses': ['congregation', 'elisheba',
'naashon', 'joshua', 'children'], 'noah': ['shem',
'japheth', 'ham', 'noe', 'hoglah']}
```

```
from sklearn.decomposition import PCA

pca = PCA(n_components=2, random_state=0)
pcs = pca.fit_transform(w2v_feature_array)
labels = ap.labels_
categories = list(corpus_df['Category'])
plt.figure(figsize=(8, 6))

for i in range(len(labels)):
label = labels[i]
color = 'orange' if label == 0 else 'blue' if label == 1
else 'green'
annotation_label = categories[i]
x, y = pcs[i]
plt.scatter(x, y, c=color, edgecolors='k')
plt.annotate(annotation_label, xy=(x+1e-4, y+1e-3),
xytext=(0, 0), textcoords='offset points')
```



https://tinyurl.com/aintpupython101

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

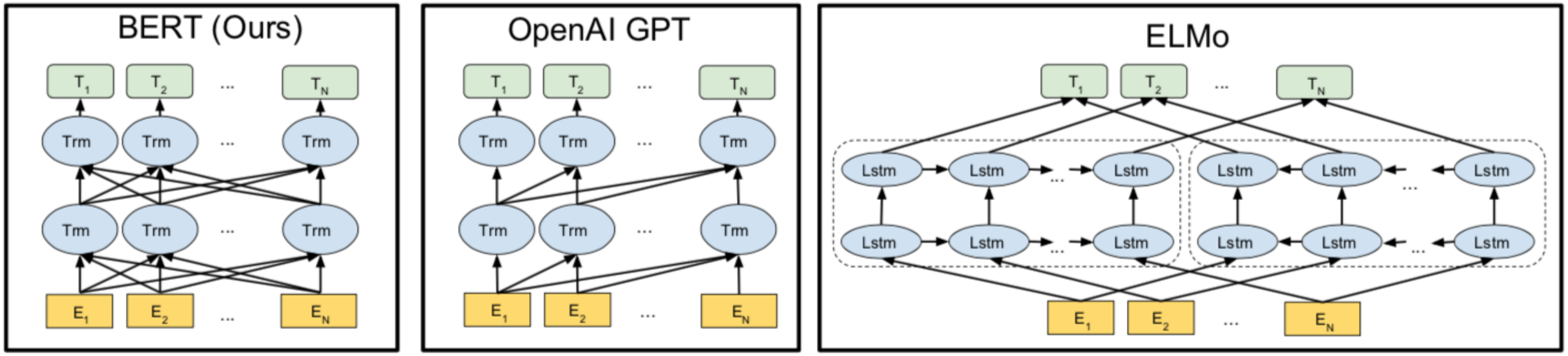


BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

Jacob Devlin Ming-Wei Chang Kenton Lee Kristina Toutanova
Google AI Language
`{jacobdevlin,mingweichang,kentonl,kristout}@google.com`

BERT

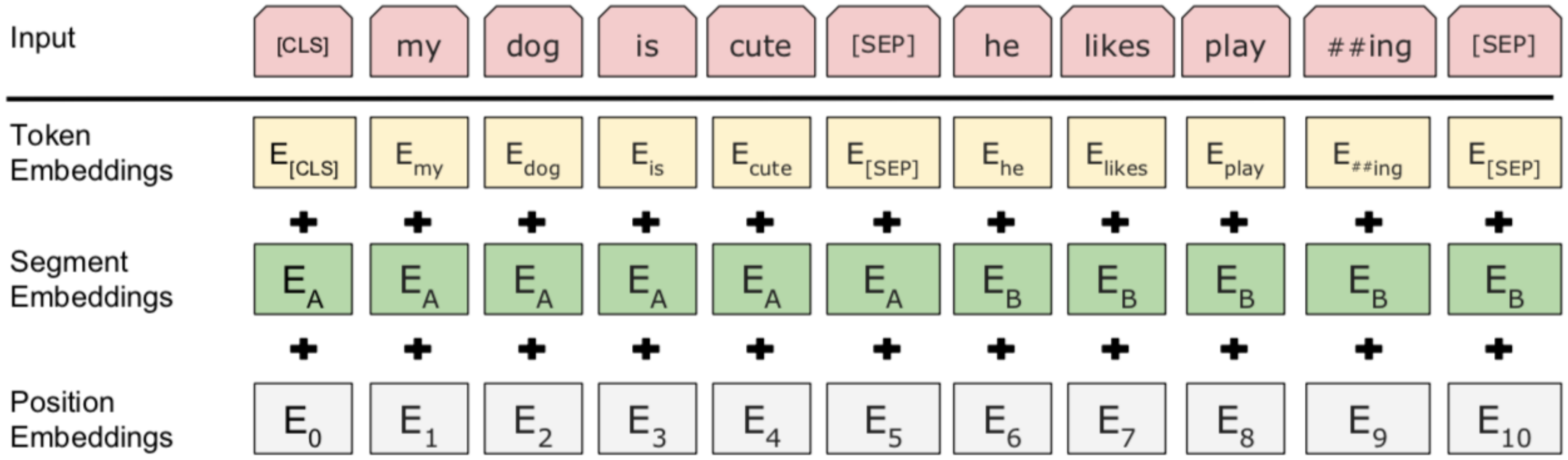
Bidirectional Encoder Representations from Transformers



Pre-training model architectures

BERT uses a bidirectional Transformer.
OpenAI GPT uses a left-to-right Transformer.
ELMo uses the concatenation of independently trained left-to-right and right- to-left LSTM to generate features for downstream tasks.
Among three, only BERT representations are jointly conditioned on both left and right context in all layers.

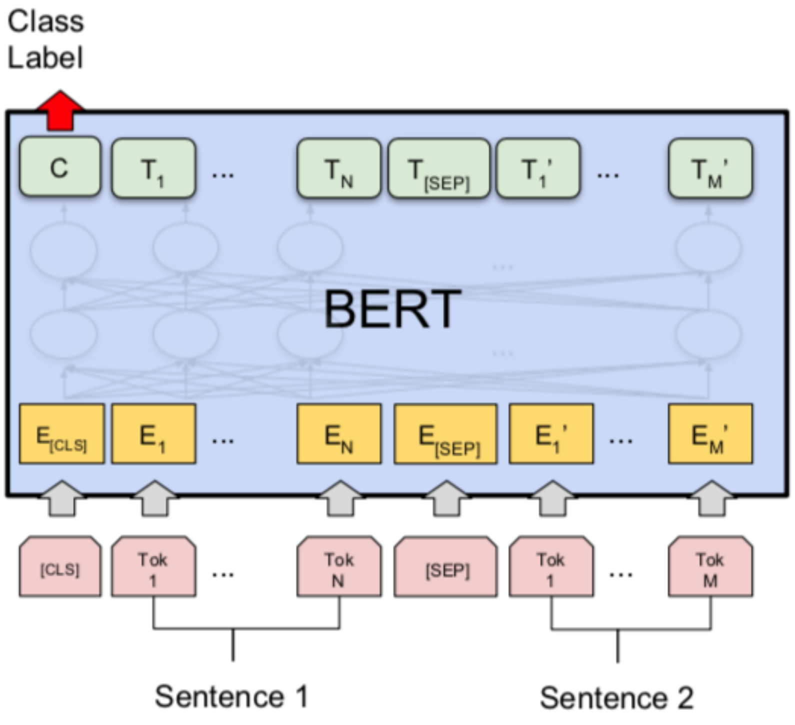
BERT input representation


Input
[CLS]
my
dog
is
cute
[SEP]
he
likes
play
##ing
[SEP]
Token Embeddings
$E_{[CLS]}$ E_{my} E_{dog} E_{is} E_{cute} $E_{[SEP]}$ E_{he} E_{likes} E_{play} $E_{\#\#ing}$ $E_{[SEP]}$
Segment Embeddings
E_A E_A E_A E_A E_A E_A E_B E_B E_B E_B E_B
Position Embeddings
E_0 E_1 E_2 E_3 E_4 E_5 E_6 E_7 E_8 E_9 E_{10}

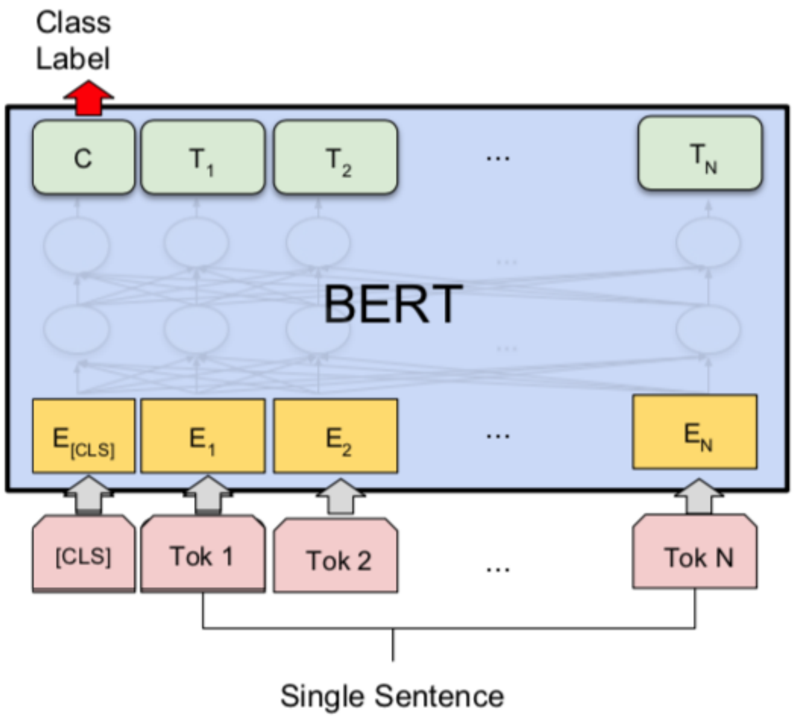
The input embeddings is the sum of the token embeddings, the segmentation embeddings and the position embeddings.

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

BERT Sequence-level tasks



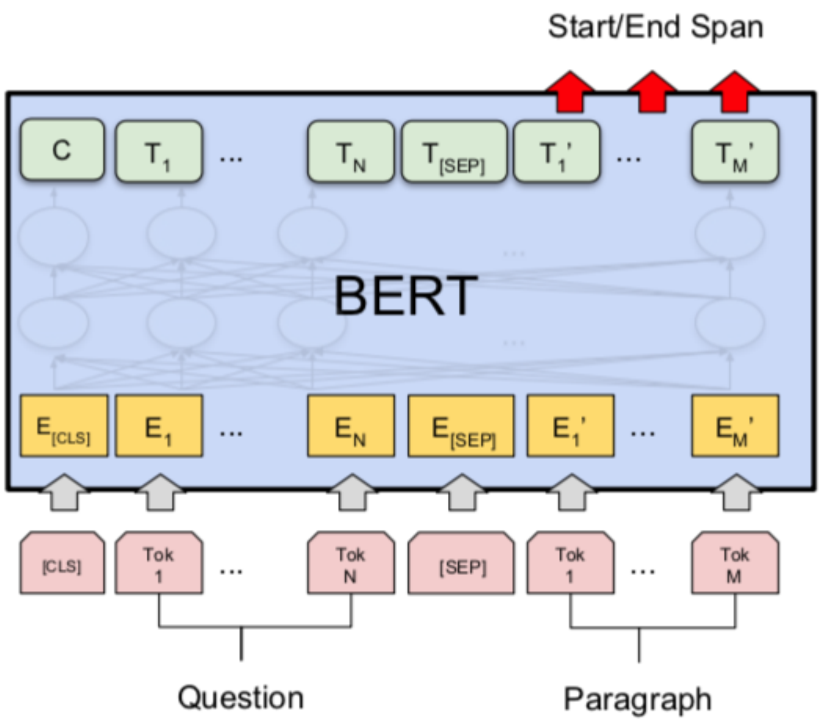
(a) Sentence Pair Classification Tasks: MNLI, QQP, QNLI, STS-B, MRPC, RTE, SWAG



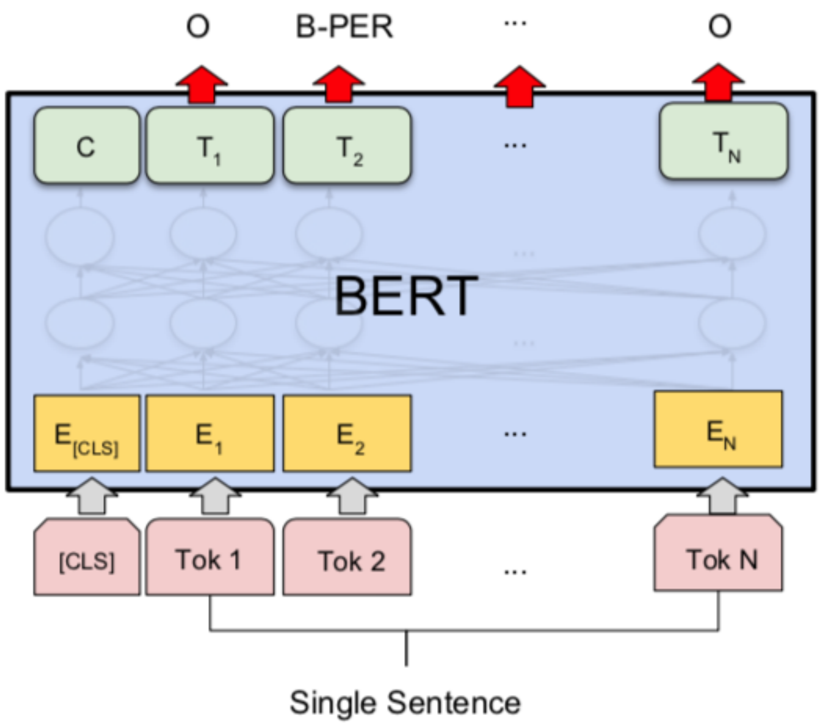
(b) Single Sentence Classification Tasks: SST-2, CoLA

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

BERT Token-level tasks



(c) Question Answering Tasks:
SQuAD v1.1

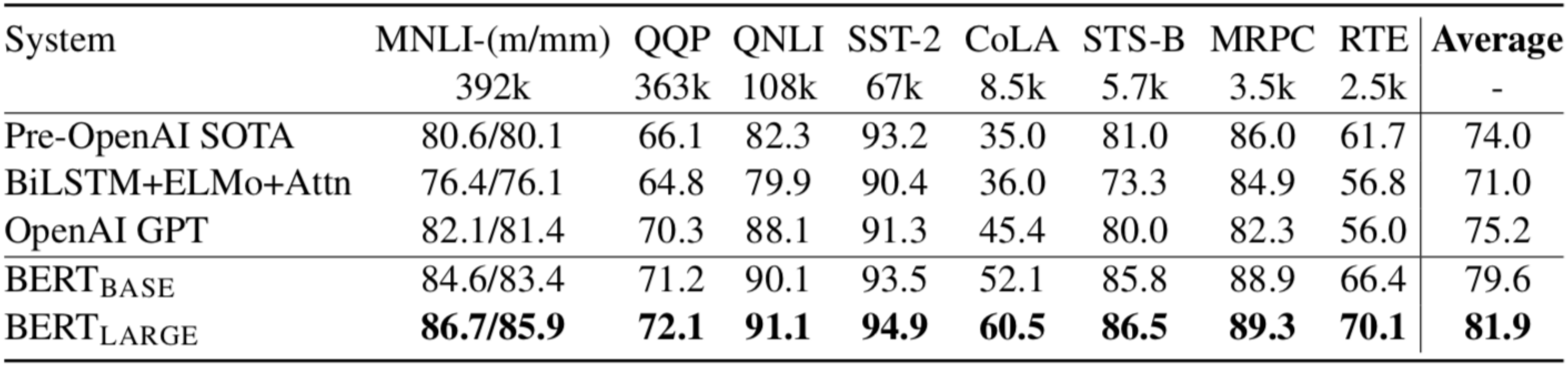


(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

General Language Understanding Evaluation (GLUE) benchmark

GLUE Test results



System	MNLI-(m/mm)	QQP	QNLI	SST-2	CoLA	STS-B	MRPC	RTE	**Average**
	392k	363k	108k	67k	8.5k	5.7k	3.5k	2.5k	-
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.9	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	88.1	91.3	45.4	80.0	82.3	56.0	75.2
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.1	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**91.1**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**81.9**

MNLI: Multi-Genre Natural Language Inference
QQP: Quora Question Pairs
QNLI: Question Natural Language Inference
SST-2: The Stanford Sentiment Treebank
CoLA: The Corpus of Linguistic Acceptability
STS-B:The Semantic Textual Similarity Benchmark
MRPC: Microsoft Research Paraphrase Corpus
RTE: Recognizing Textual Entailment

Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." arXiv preprint arXiv:1810.04805

Facebook Research FastText

Pre-trained word vectors
Word2Vec
wiki.zh.vec (861MB)
332647 word
300 vec

Pre-trained word vectors for 90 languages, trained on Wikipedia using fastText.

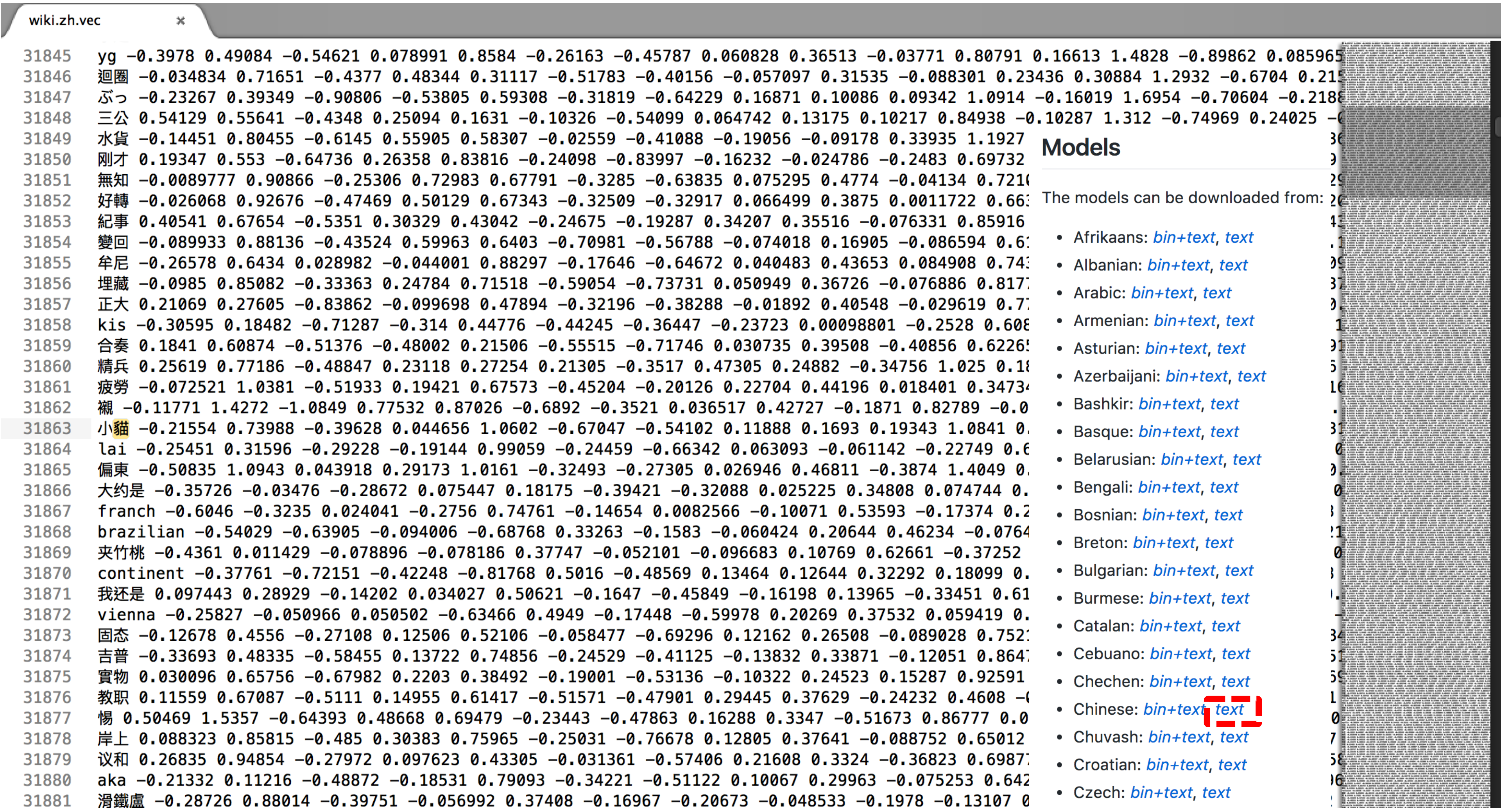
These vectors in dimension 300 were obtained using the skip-gram model with default parameters.

https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

Source: Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. "Enriching word vectors with subword information." *arXiv preprint arXiv:1607.04606* (2016).

Facebook Research FastText Word2Vec: wiki.zh.vec

(861MB) (332647 word 300 vec)



wiki.zh.vec

```
yg -0.3978 0.49084 -0.54621 0.078991 0.8584 -0.26163 -0.45787 0.060828 0.36513 -0.03771 0.80791 0.16613 1.4828 -0.89862 0.085965
迴圈 -0.034834 0.71651 -0.4377 0.48344 0.31117 -0.51783 -0.40156 -0.057097 0.31535 -0.088301 0.23436 0.30884 1.2932 -0.6704 0.215
ぶっ -0.23267 0.39349 -0.90806 -0.53805 0.59308 -0.31819 -0.64229 0.16871 0.10086 0.09342 1.0914 -0.16019 1.6954 -0.70604 -0.2188
三公 0.54129 0.55641 -0.4348 0.25094 0.1631 -0.10326 -0.54099 0.064742 0.13175 0.10217 0.84938 -0.10287 1.312 -0.74969 0.24025 -
水貨 -0.14451 0.80455 -0.6145 0.55905 0.58307 -0.02559 -0.41088 -0.19056 -0.09178 0.33935 1.1927
刚才 0.19347 0.553 -0.64736 0.26358 0.83816 -0.24098 -0.83997 -0.16232 -0.024786 -0.2483 0.69732
無知 -0.0089777 0.90866 -0.25306 0.72983 0.67791 -0.3285 -0.63835 0.075295 0.4774 -0.04134 0.721
好轉 -0.026068 0.92676 -0.47469 0.50129 0.67343 -0.32509 -0.32917 0.066499 0.3875 0.0011722 0.66
紀事 0.40541 0.67654 -0.5351 0.30329 0.43042 -0.24675 -0.19287 0.34207 0.35516 -0.076331 0.85916
變回 -0.089933 0.88136 -0.43524 0.59963 0.6403 -0.70981 -0.56788 -0.074018 0.16905 -0.086594 0.6
牟尼 -0.26578 0.6434 0.028982 -0.044001 0.88297 -0.17646 -0.64672 0.040483 0.43653 0.084908 0.74
埋藏 -0.0985 0.85082 -0.33363 0.24784 0.71518 -0.59054 -0.73731 0.050949 0.36726 -0.076886 0.817
正大 0.21069 0.27605 -0.83862 -0.099698 0.47894 -0.32196 -0.38288 -0.01892 0.40548 -0.029619 0.7
kis -0.30595 0.18482 -0.71287 -0.314 0.44776 -0.44245 -0.36447 -0.23723 0.00098801 -0.2528 0.608
合奏 0.1841 0.60874 -0.51376 -0.48002 0.21506 -0.55515 -0.71746 0.030735 0.39508 -0.40856 0.6226
精兵 0.25619 0.77186 -0.48847 0.23118 0.27254 0.21305 -0.3517 0.47305 0.24882 -0.34756 1.025 0.1
疲勞 -0.072521 1.0381 -0.51933 0.19421 0.67573 -0.45204 -0.20126 0.22704 0.44196 0.018401 0.3473
襯 -0.11771 1.4272 -1.0849 0.77532 0.87026 -0.6892 -0.3521 0.036517 0.42727 -0.1871 0.82789 -0.0
小貓 -0.21554 0.73988 -0.39628 0.044656 1.0602 -0.67047 -0.54102 0.11888 0.1693 0.19343 1.0841 0.
lai -0.25451 0.31596 -0.29228 -0.19144 0.99059 -0.24459 -0.66342 0.063093 -0.061142 -0.22749 0.6
偏東 -0.50835 1.0943 0.043918 0.29173 1.0161 -0.32493 -0.27305 0.026946 0.46811 -0.3874 1.4049 0.
大约是 -0.35726 -0.03476 -0.28672 0.075447 0.18175 -0.39421 -0.32088 0.025225 0.34808 0.074744 0.
franch -0.6046 -0.3235 0.024041 -0.2756 0.74761 -0.14654 0.0082566 -0.10071 0.53593 -0.17374 0.2
brazilian -0.54029 -0.63905 -0.094006 -0.68768 0.33263 -0.1583 -0.060424 0.20644 0.46234 -0.0764
夹竹桃 -0.4361 0.011429 -0.078896 -0.078186 0.37747 -0.052101 -0.096683 0.10769 0.62661 -0.37252
continent -0.37761 -0.72151 -0.42248 -0.81768 0.5016 -0.48569 0.13464 0.12644 0.32292 0.18099 0.
我还是 0.097443 0.28929 -0.14202 0.034027 0.50621 -0.1647 -0.45849 -0.16198 0.13965 -0.33451 0.61
vienna -0.25827 -0.050966 0.050502 -0.63466 0.4949 -0.17448 -0.59978 0.20269 0.37532 0.059419 0.
固态 -0.12678 0.4556 -0.27108 0.12506 0.52106 -0.058477 -0.69296 0.12162 0.26508 -0.089028 0.752
吉普 -0.33693 0.48335 -0.58455 0.13722 0.74856 -0.24529 -0.41125 -0.13832 0.33871 -0.12051 0.864
實物 0.030096 0.65756 -0.67982 0.2203 0.38492 -0.19001 -0.53136 -0.10322 0.24523 0.15287 0.92591
教职 0.11559 0.67087 -0.5111 0.14955 0.61417 -0.51571 -0.47901 0.29445 0.37629 -0.24232 0.4608 -
惕 0.50469 1.5357 -0.64393 0.48668 0.69479 -0.23443 -0.47863 0.16288 0.3347 -0.51673 0.86777 0.0
岸上 0.088323 0.85815 -0.485 0.30383 0.75965 -0.25031 -0.76678 0.12805 0.37641 -0.088752 0.65012
议和 0.26835 0.94854 -0.27972 0.097623 0.43305 -0.031361 -0.57406 0.21608 0.3324 -0.36823 0.6987
aka -0.21332 0.11216 -0.48872 -0.18531 0.79093 -0.34221 -0.51122 0.10067 0.29963 -0.075253 0.642
滑鐵盧 -0.28726 0.88014 -0.39751 -0.056992 0.37408 -0.16967 -0.20673 -0.048533 -0.1978 -0.13107 0
```

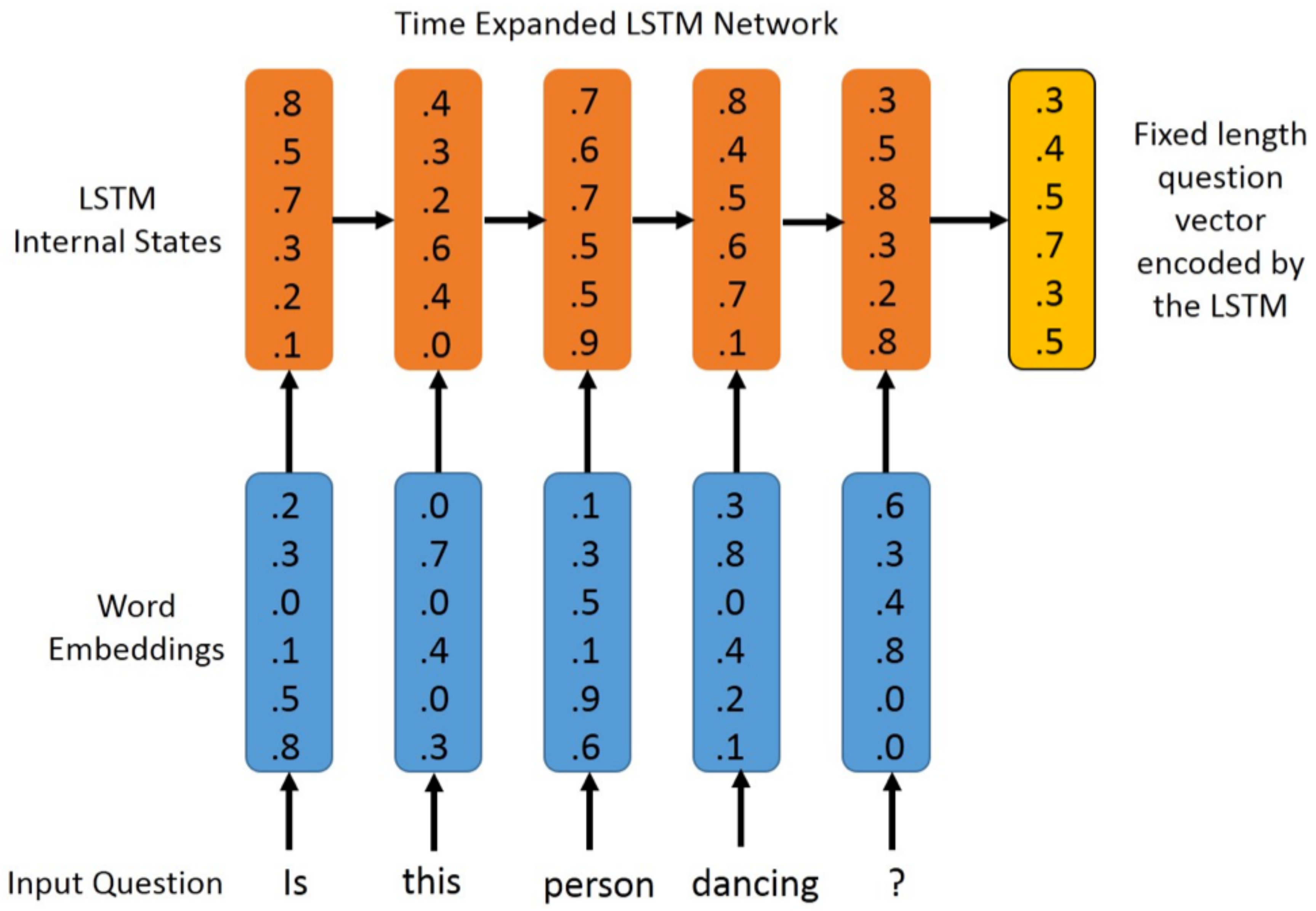
Models

The models can be downloaded from:

- Afrikaans: *bin+text*, *text*
- Albanian: *bin+text*, *text*
- Arabic: *bin+text*, *text*
- Armenian: *bin+text*, *text*
- Asturian: *bin+text*, *text*
- Azerbaijani: *bin+text*, *text*
- Bashkir: *bin+text*, *text*
- Basque: *bin+text*, *text*
- Belarusian: *bin+text*, *text*
- Bengali: *bin+text*, *text*
- Bosnian: *bin+text*, *text*
- Breton: *bin+text*, *text*
- Bulgarian: *bin+text*, *text*
- Burmese: *bin+text*, *text*
- Catalan: *bin+text*, *text*
- Cebuano: *bin+text*, *text*
- Chechen: *bin+text*, *text*
- Chinese: *bin+text*, *text*
- Chuvash: *bin+text*, *text*
- Croatian: *bin+text*, *text*
- Czech: *bin+text*, *text*

https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

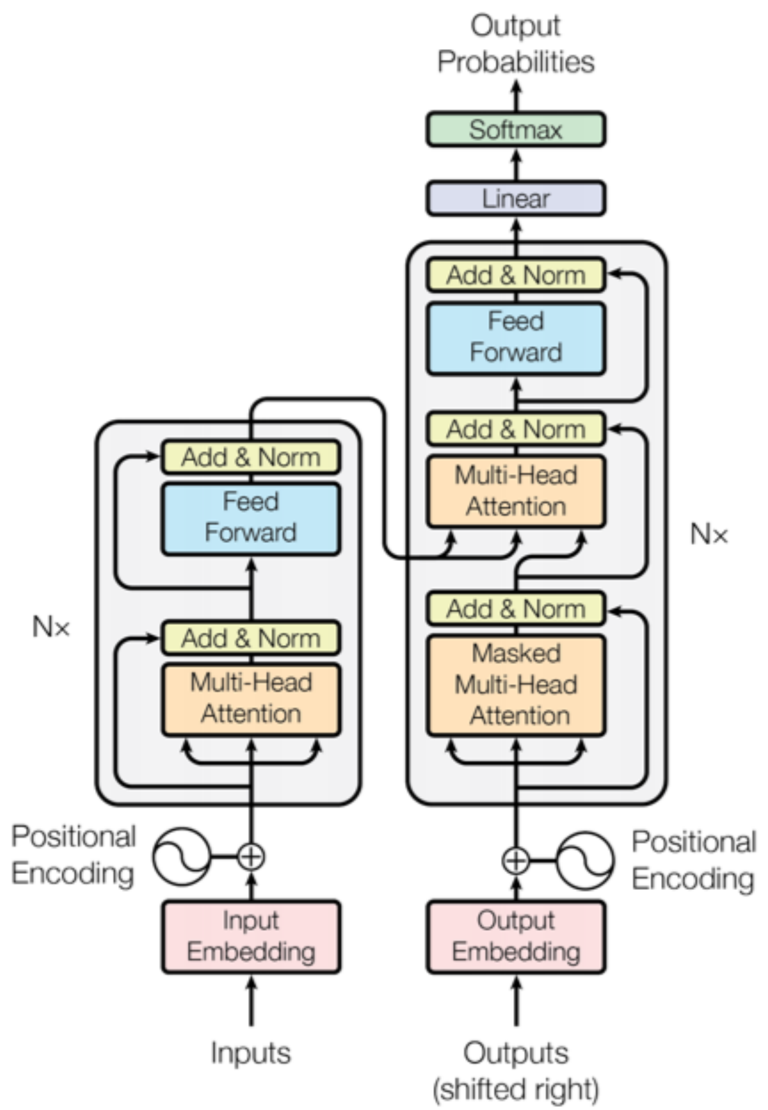
Word Embeddings in LSTM RNN



Source: https://avisingh599.github.io/deeplearning/visual-qa/

Transformer (Attention is All You Need)

(Vaswani et al., 2017)

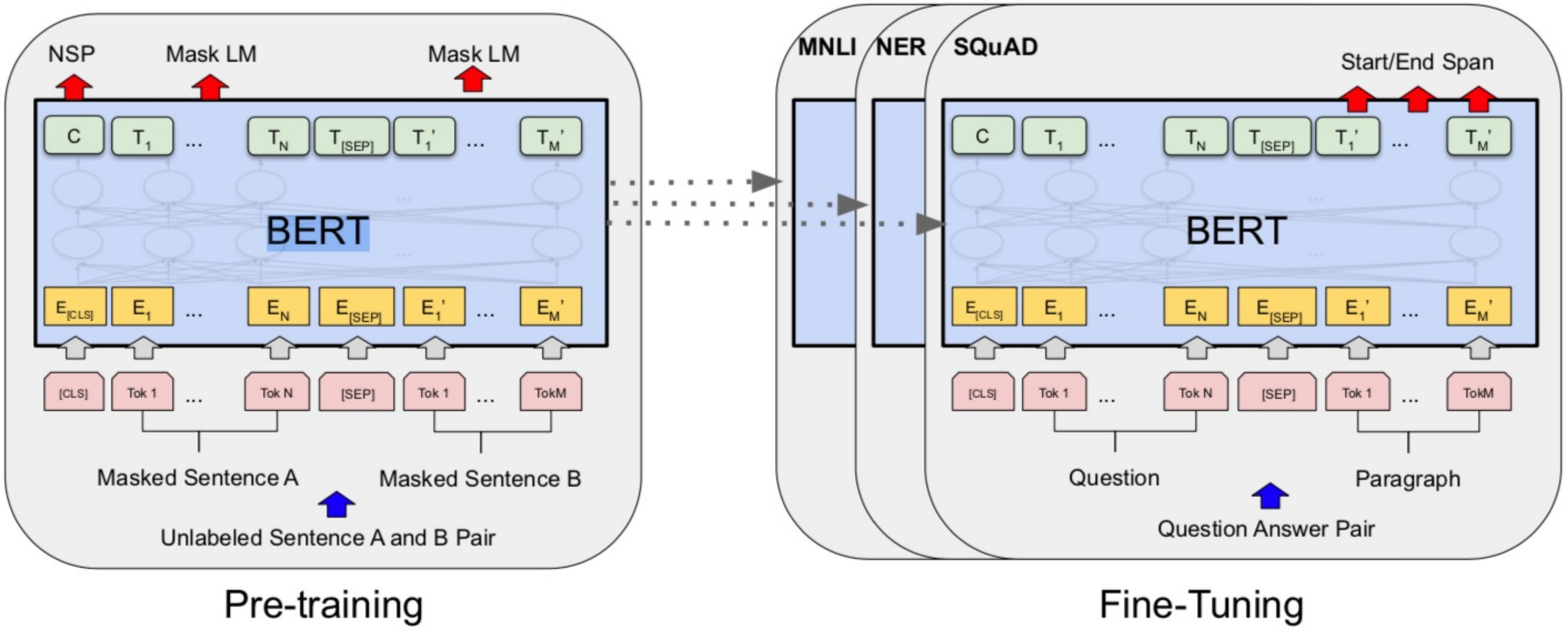


Source: Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." In *Advances in neural information processing systems*, pp. 5998-6008. 2017.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

BERT (Bidirectional Encoder Representations from Transformers)

Overall pre-training and fine-tuning procedures for BERT

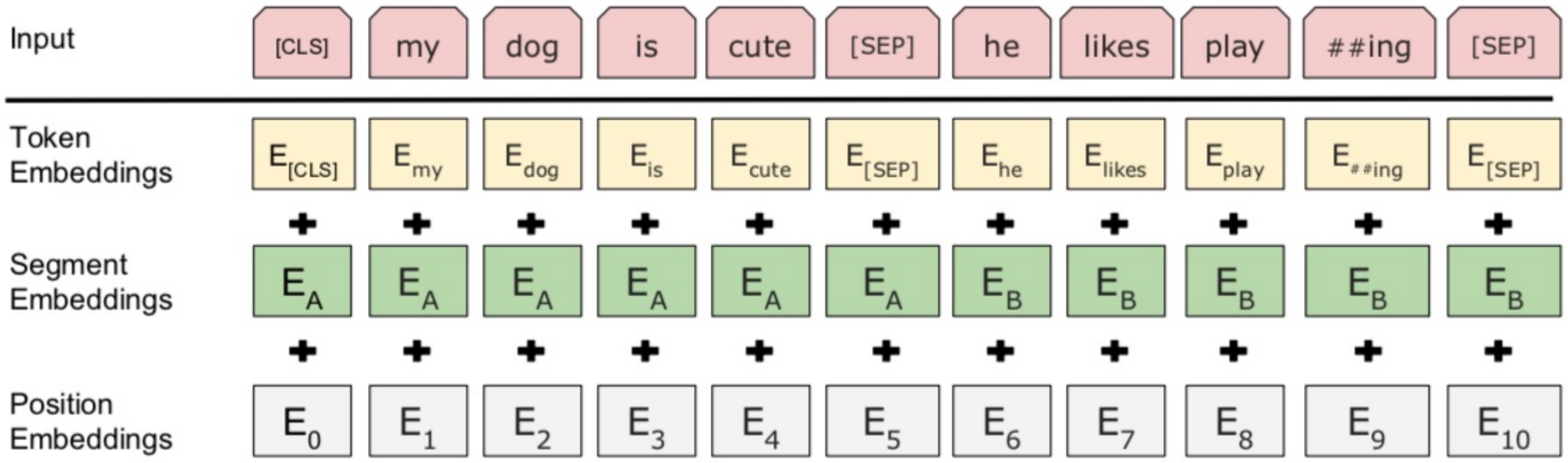


Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018). "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding

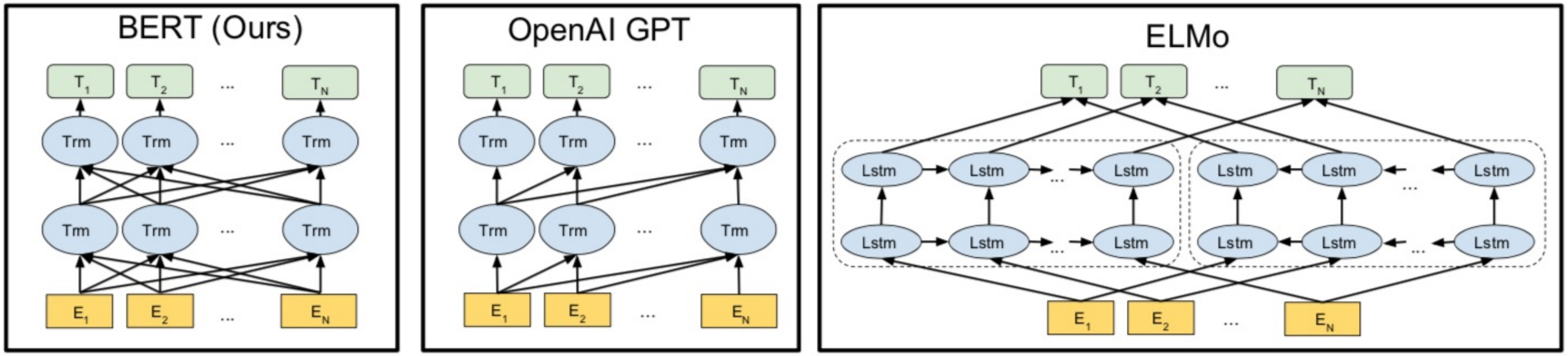
BERT (Bidirectional Encoder Representations from Transformers)

BERT input representation

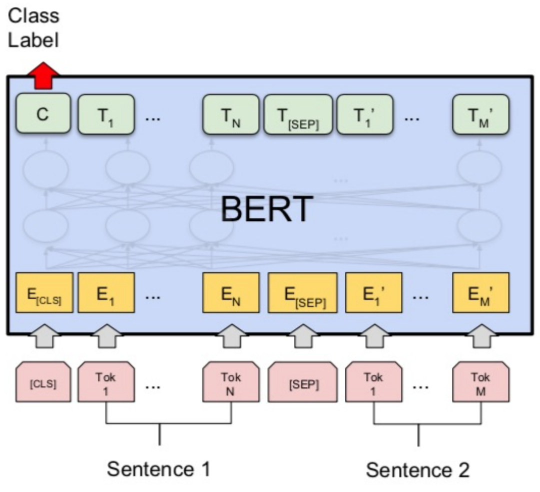


Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

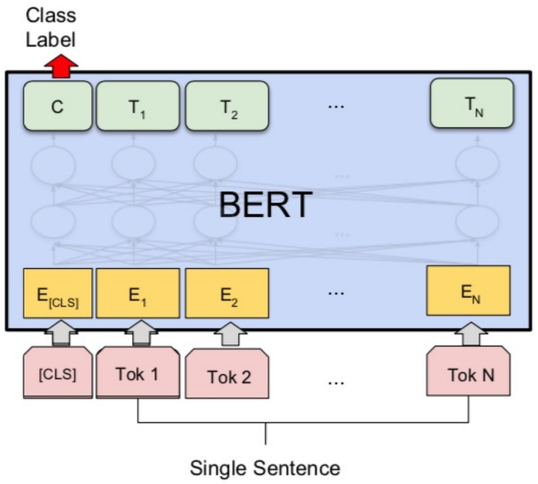
BERT, OpenAI GPT, ELMo



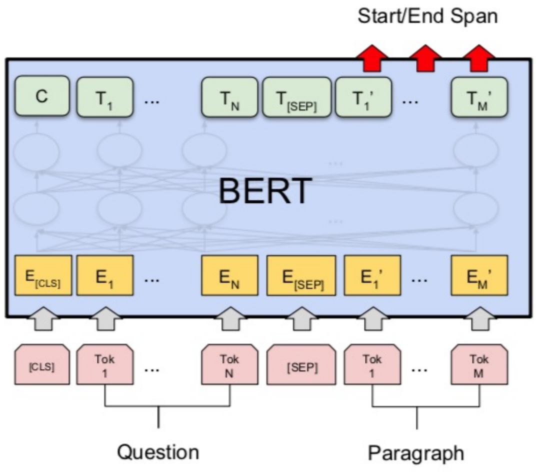
Fine-tuning BERT on Different Tasks



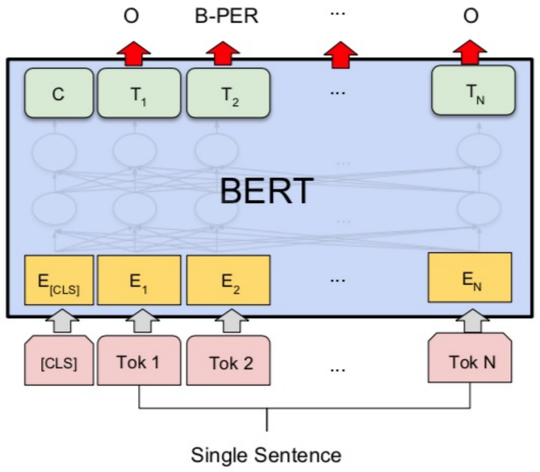
(a) Sentence Pair Classification Tasks:
MNLI, QQP, QNLI, STS-B, MRPC,
RTE, SWAG



(b) Single Sentence Classification Tasks:
SST-2, CoLA



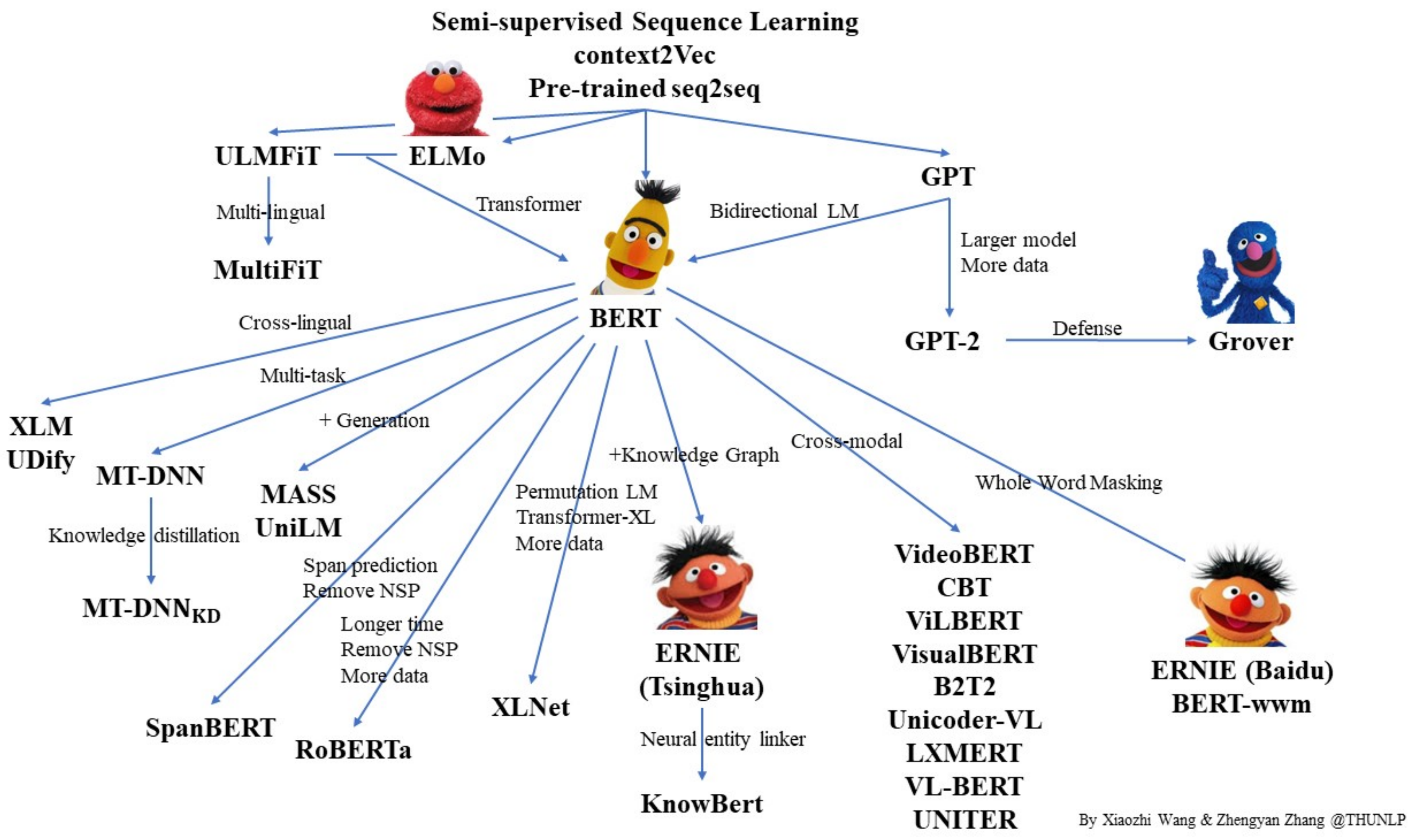
(c) Question Answering Tasks:
SQuAD v1.1



(d) Single Sentence Tagging Tasks:
CoNLL-2003 NER

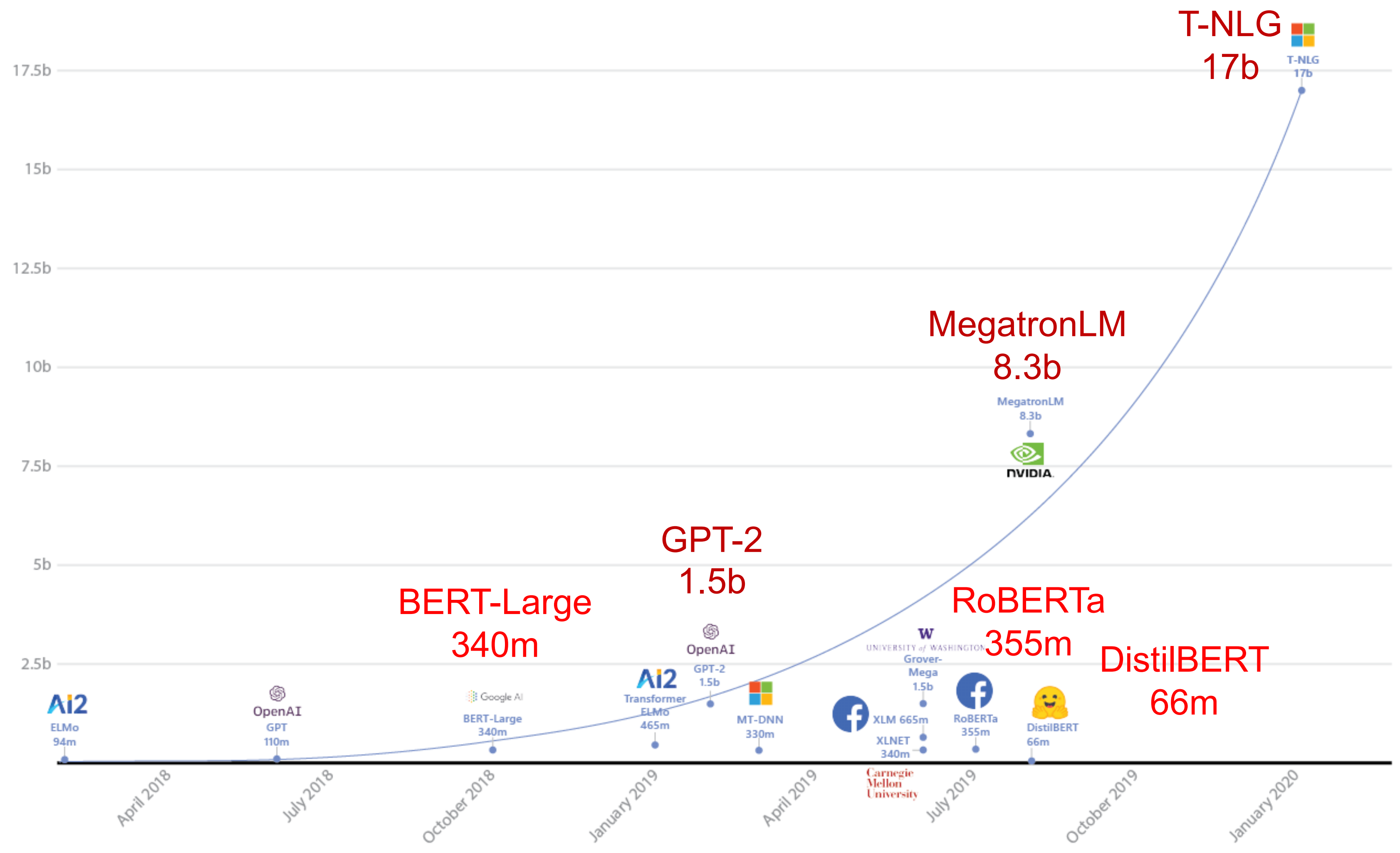
Source: Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova (2018).
"Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805.

Pre-trained Language Model (PLM)



Source: https://github.com/thunlp/PLMpapers

Turing Natural Language Generation (T-NLG)



2018 2019 2020

Source: https://www.microsoft.com/en-us/research/blog/turing-nlg-a-17-billion-parameter-language-model-by-microsoft/

🤗 Transformers Transformers

State-of-the-art Natural Language Processing for TensorFlow 2.0 and PyTorch

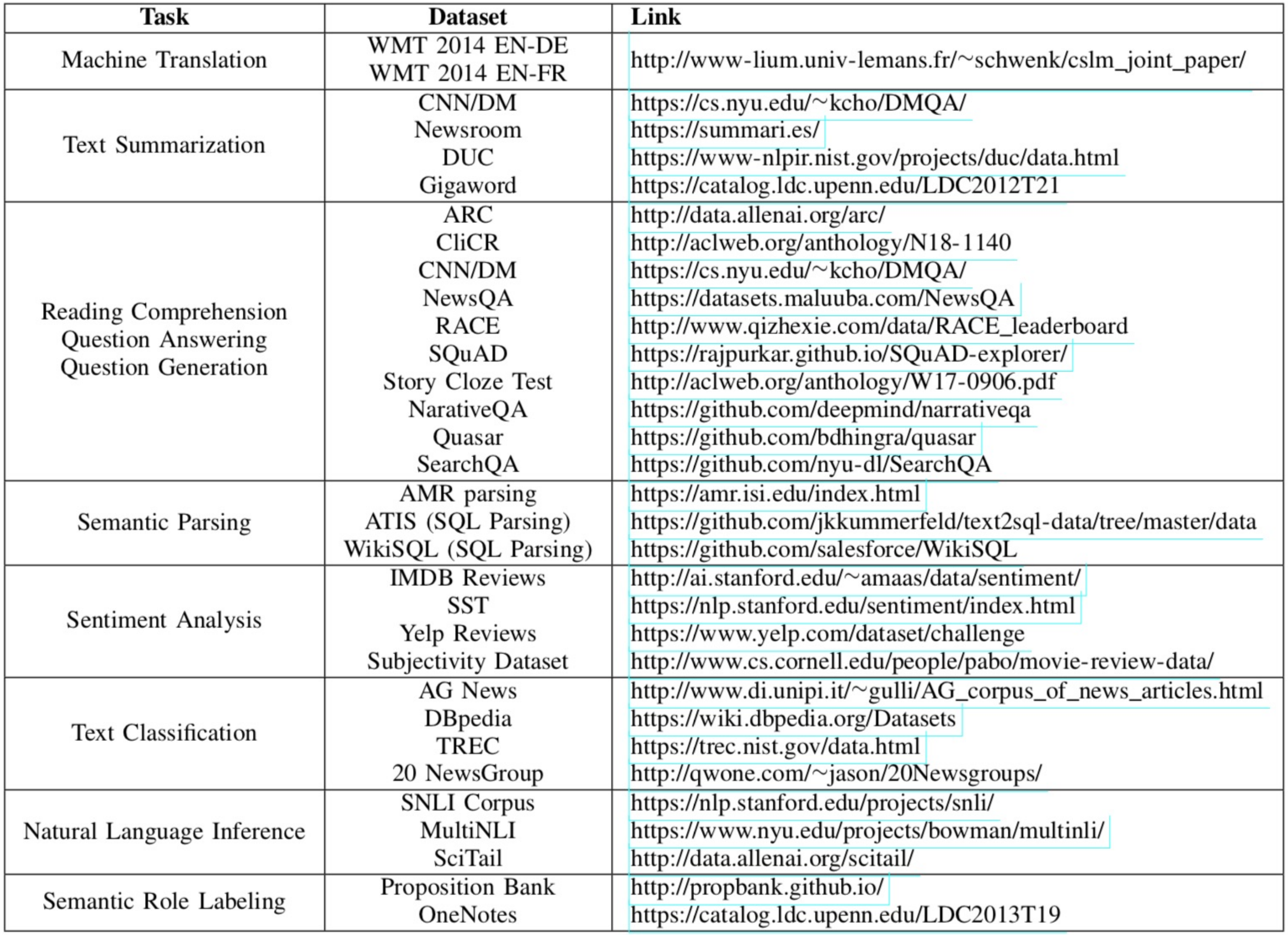
- Transformers
  - pytorch-transformers
  - pytorch-pretrained-bert
- provides state-of-the-art general-purpose architectures
  - (BERT, GPT-2, RoBERTa, XLM, DistilBert, XLNet, CTRL...)
  - for Natural Language Understanding (NLU) and
    Natural Language Generation (NLG)
    with over 32+ pretrained models
    in 100+ languages
    and deep interoperability between
    TensorFlow 2.0 and
    PyTorch.

Source: https://github.com/huggingface/transformers

Transfer Learning in Natural Language Processing

Source: Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf (2019), "Transfer learning in natural language processing." In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials, pp. 15-18.

NLP Benchmark Datasets



Task	Dataset	Link
Machine Translation	WMT 2014 EN-DE WMT 2014 EN-FR	http://www-lium.univ-lemans.fr/~schwenk/cslm_joint_paper/
Text Summarization	CNN/DM	https://cs.nyu.edu/~kcho/DMQA/
	Newsroom	https://summari.es/
	DUC	https://www-nlpir.nist.gov/projects/duc/data.html
	Gigaword	https://catalog.ldc.upenn.edu/LDC2012T21
Reading Comprehension Question Answering Question Generation	ARC	http://data.allenai.org/arc/
	CliCR	http://aclweb.org/anthology/N18-1140
	CNN/DM	https://cs.nyu.edu/~kcho/DMQA/
	NewsQA	https://datasets.maluuba.com/NewsQA
	RACE	http://www.qizhexie.com/data/RACE_leaderboard
	SQuAD	https://rajpurkar.github.io/SQuAD-explorer/
	Story Cloze Test	http://aclweb.org/anthology/W17-0906.pdf
	NarativeQA	https://github.com/deepmind/narrativeqa
	Quasar	https://github.com/bdhingra/quasar
	SearchQA	https://github.com/nyu-dl/SearchQA
Semantic Parsing	AMR parsing	https://amr.isi.edu/index.html
	ATIS (SQL Parsing)	https://github.com/jkkummerfeld/text2sql-data/tree/master/data
	WikiSQL (SQL Parsing)	https://github.com/salesforce/WikiSQL
Sentiment Analysis	IMDB Reviews	http://ai.stanford.edu/~amaas/data/sentiment/
	SST	https://nlp.stanford.edu/sentiment/index.html
	Yelp Reviews	https://www.yelp.com/dataset/challenge
	Subjectivity Dataset	http://www.cs.cornell.edu/people/pabo/movie-review-data/
Text Classification	AG News	http://www.di.unipi.it/~gulli/AG_corpus_of_news_articles.html
	DBpedia	https://wiki.dbpedia.org/Datasets
	TREC	https://trec.nist.gov/data.html
	20 NewsGroup	http://qwone.com/~jason/20Newsgroups/
Natural Language Inference	SNLI Corpus	https://nlp.stanford.edu/projects/snli/
	MultiNLI	https://www.nyu.edu/projects/bowman/multinli/
	SciTail	http://data.allenai.org/scitail/
Semantic Role Labeling	Proposition Bank	http://propbank.github.io/
	OneNotes	https://catalog.ldc.upenn.edu/LDC2013T19

Source: Amirsina Torfi, Rouzbeh A. Shirvani, Yaser Keneshloo, Nader Tavvaf, and Edward A. Fox (2020). "Natural Language Processing Advancements By Deep Learning: A Survey." arXiv preprint arXiv:2003.01200.

Summary

- Traditional Feature Engineering for Text Data
  - Bag of Words Model
  - Bag of N-Grams Model
  - TF-IDF Model
- Advanced Word Embeddings with Deep Learning
  - Word2Vec Model
  - Robust Word2Vec Models with Gensim
  - GloVe Model
  - FastText Model

References

- Dipanjan Sarkar (2019),
  Text Analytics with Python: A Practitioner’s Guide to Natural Language Processing, Second Edition. APress.
  https://github.com/Apress/text-analytics-w-python-2e
- Benjamin Bengfort, Rebecca Bilbro, and Tony Ojeda (2018), Applied Text Analysis with Python,
  O'Reilly Media.
  https://www.oreilly.com/library/view/applied-text-analysis/9781491963036/