Common Models

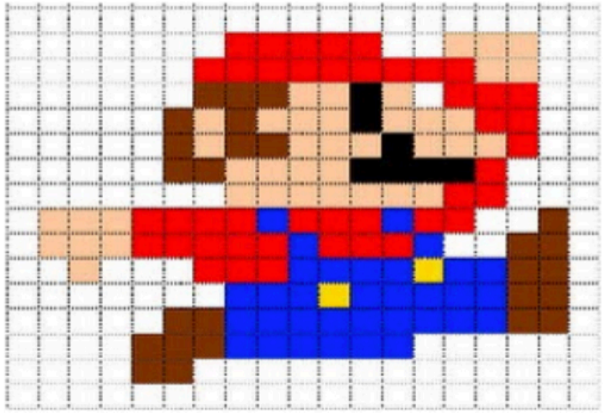
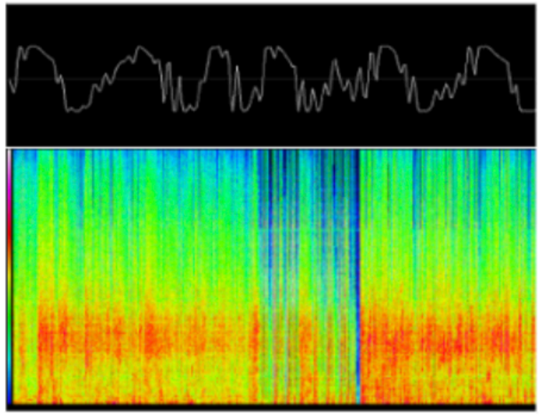


IMAGE PIXELS (DENSE)



AUDIO SPECTROGRAM (DENSE)

bacon	beans	beautiful	blue	breakfast	brown	dog	eggs	fox
0	0	1	1	0	0	0	0	0

WORD VECTORS (SPARSE)

Vectorization

A process of converting text into numbers

Vector Space

A representation of text as numeric vectors where each dimension is a feature

Count-Based Models

Bag of Words

Bag of N-Grams

TF-IDF

Similarity Features

Topic Models

- a bag of unstructured words
- context information is not preserved

Dipanjan Sarkar. 2018. Understanding Feature Engineering

Summary

Distributed Representation = Word Vector = Word Embedding

Bag-of-Words Models

- Count-based: TF, TF-IDF, N-grams

Prediction-Based Models

- Based on distributed representations (a dense representations of words in a low-dimensional vector space): Word2Vec, FastText

Word = one point in the embedding space

```
king = np.array([.2, -.5, .7, .2, -.9])
man = np.array([-.5, .2, -.2, .3, 0.])
woman = np.array([.7, -.3, .3, .6, .1])
```

n dimensions = 5: 1 x 5 vector

Word is associated with a continuous vector representation

Dimensions = latent characteristics of a word (grammatical or semantic property)

```
[0.3, 0.2, …]
[0.2, 0.1, …]
[0.9, 0.6, …]
```

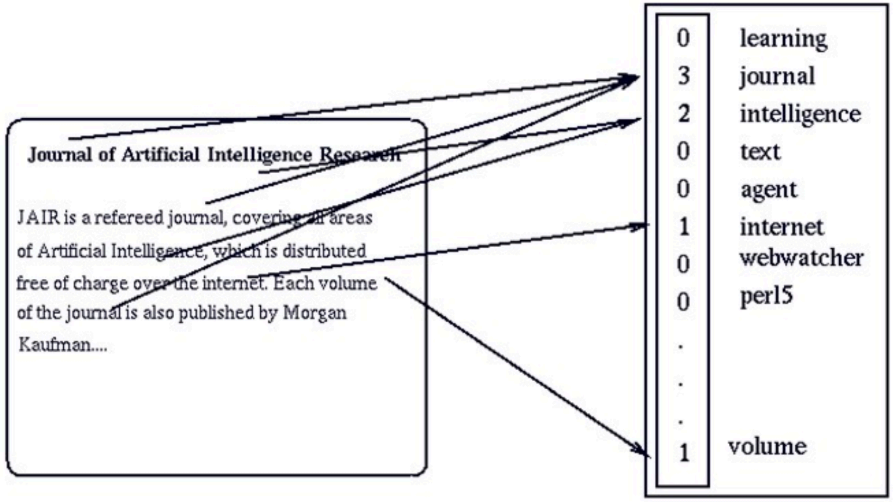
Sarkar, D. 2018. Deep Learning Methods for Text Data – Word2Vec, GloVe, FastText. Towards Data Science

Vector Space Model

D - a document
W – a word
VS - a document vector space
n dimension = n of distinct words
W_{Dn} - a weight for word n in document D

$$VS = \{W_1, W_2, \ldots, W_n\}$$

$$D = \{w_{D1}, w_{D2}, \ldots, w_{Dn}\}$$



Text document – a numeric vector

Dimension – a specific word with its frequency or occurrence (0 and 1) or weighted values.

Marco Grobelnik et al. Text Mining Tutorial. 2015.

Vectorization

Common methods for extracting numerical features from text content:

- **tokenizing** strings and giving an integer id for each possible token
- **counting** the occurrences of tokens in each document
- **normalizing** and weighting tokens

Features and samples

- **feature:** each individual token occurrence frequency (normalized or not)
- **sample:** the vector of all the token frequencies for a given document

Corpus

- a matrix with one row per document and one column per token
- a sparse matrix

sparse matrix

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

Common Vectorizer Usage: CountVectorizer

Useful Parameters:
lowercase
token pattern
tokenizer

from sklearn.feature_extraction.text import CountVectorizer

```
CountVectorizer(analyzer='word', binary=False, decode_error='strict',
        dtype=<class 'numpy.int64'>, encoding='utf-8', input='content',
        lowercase=True, max_df=1.0, max_features=None, min_df=1,
        ngram_range=(1, 1), preprocessor=None, stop_words=None,
        strip_accents=None, token_pattern='(?u)\\b\\w\\w+\\b',
        tokenizer=None, vocabulary=None)
```

1

```
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!']
```

2

```
vectorizer = CountVectorizer(stop_words='english',min_df=0.
X = vectorizer.fit_transform(corpus)
X_matrix = X.toarray()
```

```
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]],
      dtype=int64)
```

3

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

	bacon	beans	beautiful	blue	breakfast	brown	dog	eggs	fox	green	ham	jumps	king	l
0	0	0	1	1	0	0	0	0	0	0	0	0	0	
1	0	0	1	1	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	1	1	0	1	0	0	1	0	
3	1	1	0	0	1	0	0	1	0	0	1	0	1	
4	1	0	0	0	0	0	0	1	0	1	1	0	0	
5	0	0	0	1	0	1	1	0	1	0	0	0	0	
6	0	0	1	1	0	0	0	0	0	0	0	0	0	
7	0	0	0	0	0	1	1	0	1	0	0	0	0	

https://scikit-learn.org/stable/modules/feature_extraction.html

Bag of N-Grams

Dimension = Number of unique words

	bacon	beans	beautiful	blue	breakfast	brown	dog	e
0	0	0	1	1	0	0	0	
1	0	0	1	1	0	0	0	
2	0	0	0	0	0	1	1	
3	1	1	0	0	1	0	0	
4	1	0	0	0	0	0	0	
5	0	0	0	1	0	1	1	
6	0	0	1	1	0	0	0	
7	0	0	0	0	0	1	1	

Bag of Words

	bacon eggs	beautiful sky	beautiful today	blue beautiful	blue dog	blue sky	breakfast sausages
0	0	0	0	1	0	0	0
1	0	1	0	1	0	0	0
2	0	0	0	0	0	0	0
3	1	0	0	0	0	0	1
4	0	0	0	0	0	0	0
5	0	0	0	0	1	0	0
6	0	0	1	0	0	1	0
7	0	0	0	0	0	0	0

Bag of N-Grams

CountVectorizer(ngram_range=(1,1))

CountVectorizer(ngram_range=(2,2))

one Word = unigram (unique token)

two Words = bi-grams (unique tokens)

https://scikit-learn.org/stable/modules/feature_extraction.html

ngram_range : tuple (min_n, max_n)

Bag of Words versus TF-IDF Model

Absolute term frequencies

king	lazy	love	quick	sausages	sky
0	0	0	0	0	1
0	0	1	0	0	1
0	1	0	1	0	0
1	0	0	0	1	0
0	0	1	0	1	0
0	1	0	1	0	0
0	0	0	0	0	2
0	1	0	1	0	0

king	1
lazy	3
love	2
quick	3
sausages	2
sky	4

King is important entity and more meaningful than quick!

Normalized term frequencies

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

N - total number of documents

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

term frequency–inverse document frequency

TF-IDF score

term frequency of i word per j document

df – number of documents containing i word

TF-IDF

Document 1 : The car is driven on the road.

Document 2: The truck is driven on the highway.

Document 1 Document 2

total documents =2
documents with
“the” word = 2

Word	TF		IDF	TF*IDF	
	A	B		A	B
The	1/7	1/7	log(2/2) = 0	0	0
Car	1/7	0	log(2/1) = 0.3	0.043	0
Truck	0	1/7	log(2/1) = 0.3	0	0.043
Is	1/7	1/7	log(2/2) = 0	0	0
Driven	1/7	1/7	log(2/2) = 0	0	0
On	1/7	1/7	log(2/2) = 0	0	0
The	1/7	1/7	log(2/2) = 0	0	0
Road	1/7	0	log(2/1) = 0.3	0.043	0
Highway	0	1/7	log(2/1) = 0.3	0	0.043

Common word

Meaningful word

(Mayank Tripathi. 2018)

TF-IDF in Python

from sklearn.feature_extraction.text import TfidfVectorizer

```
class sklearn.feature_extraction.text. TfidfVectorizer (input='content', encoding='utf-8',
decode_error='strict', strip_accents=None, lowercase=True, preprocessor=None, tokenizer=None,
analyzer='word', stop_words=None, token_pattern='(?u)\b\w\w+\b', ngram_range=(1, 1), max_df=1.0, min_df=1,
max_features=None, vocabulary=None, binary=False, dtype=<class 'numpy.float64'>, norm='l2', use_idf=True,
smooth_idf=True, sublinear_tf=False) ¶                                                        [source]
```

```
vectorizer = TfidfVectorizer()
model = vectorizer.fit_transform(norm_corpus)
model_matrix = model.toarray()
vocab = vectorizer.get_feature_names()
pd.DataFrame(model_matrix.round(2), columns=vocab)
```

columns = 20 unique features/words

rows = 8 documents

```
array(['sky blue beautiful', 'love blue beautiful sky',
       'quick brown fox jumps lazy dog',
       'kings breakfast sausages ham bacon eggs toast beans',
       'love green eggs ham sausages bacon',
       'brown fox quick blue dog lazy', 'sky blue sky beautiful today
       'dog lazy brown fox quick'], dtype='<U51')
```

	bacon	beans	beautiful	blue	breakfast	brown	dog	eggs	fox	green
0	0.00	0.00	0.60	0.53	0.00	0.00	0.00	0.00	0.00	0.00
1	0.00	0.00	0.49	0.43	0.00	0.00	0.00	0.00	0.00	0.00
2	0.00	0.00	0.00	0.00	0.00	0.38	0.38	0.00	0.38	0.00
3	0.32	0.38	0.00	0.00	0.38	0.00	0.00	0.32	0.00	0.00
4	0.39	0.00	0.00	0.00	0.00	0.00	0.00	0.39	0.00	0.47
5	0.00	0.00	0.00	0.37	0.00	0.42	0.42	0.00	0.42	0.00
6	0.00	0.00	0.36	0.32	0.00	0.00	0.00	0.00	0.00	0.00
7	0.00	0.00	0.00	0.00	0.00	0.45	0.45	0.00	0.45	0.00