



GALLOGLY COLLEGE OF ENGINEERING
SCHOOL OF COMPUTER SCIENCE
The UNIVERSITY of OKLAHOMA

Text Analysis

Text classification



Presented by: Keerti Banweer





Would you like to be involved in research at the University of Oklahoma?

The purpose of this survey is to understand attendees' knowledge of Python and intended outcomes from attending these sessions. We are trying to gauge the demand of skills, resources, and knowledge of Python programming and associated techniques. This will, in turn, help us understand the success of the workshop and room for improvement.

SCAN ME



Overview



Text Analysis



What is Text Classification?



Machine learning approach



Types of Text Classification



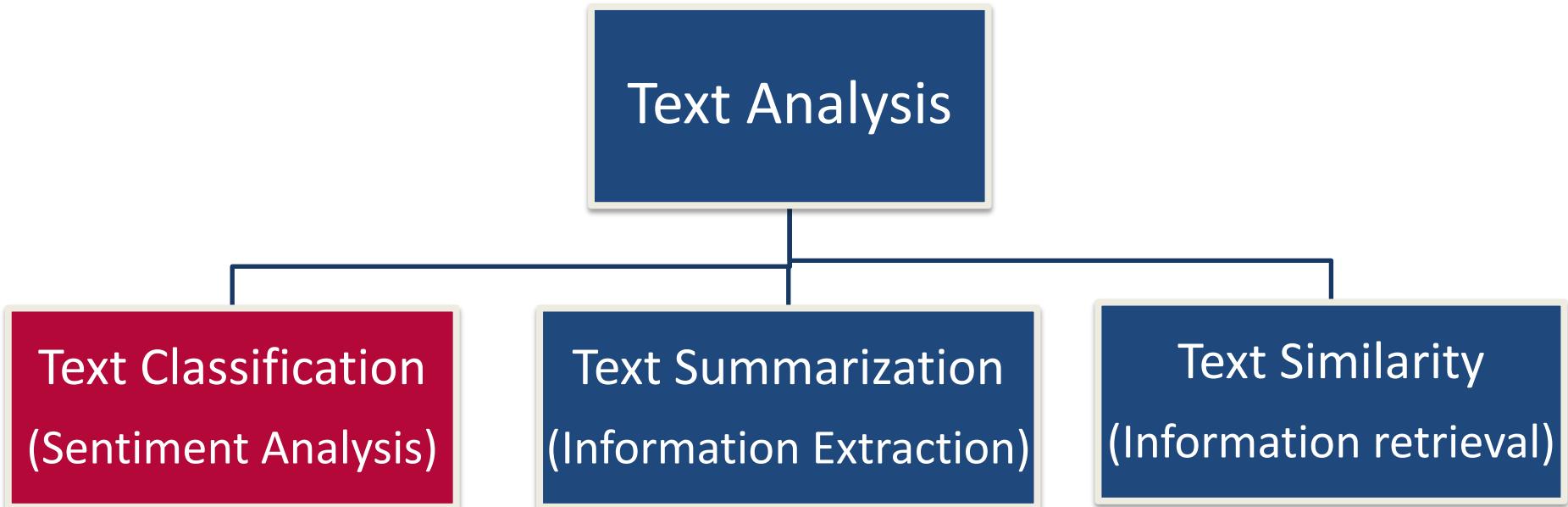
Workflow



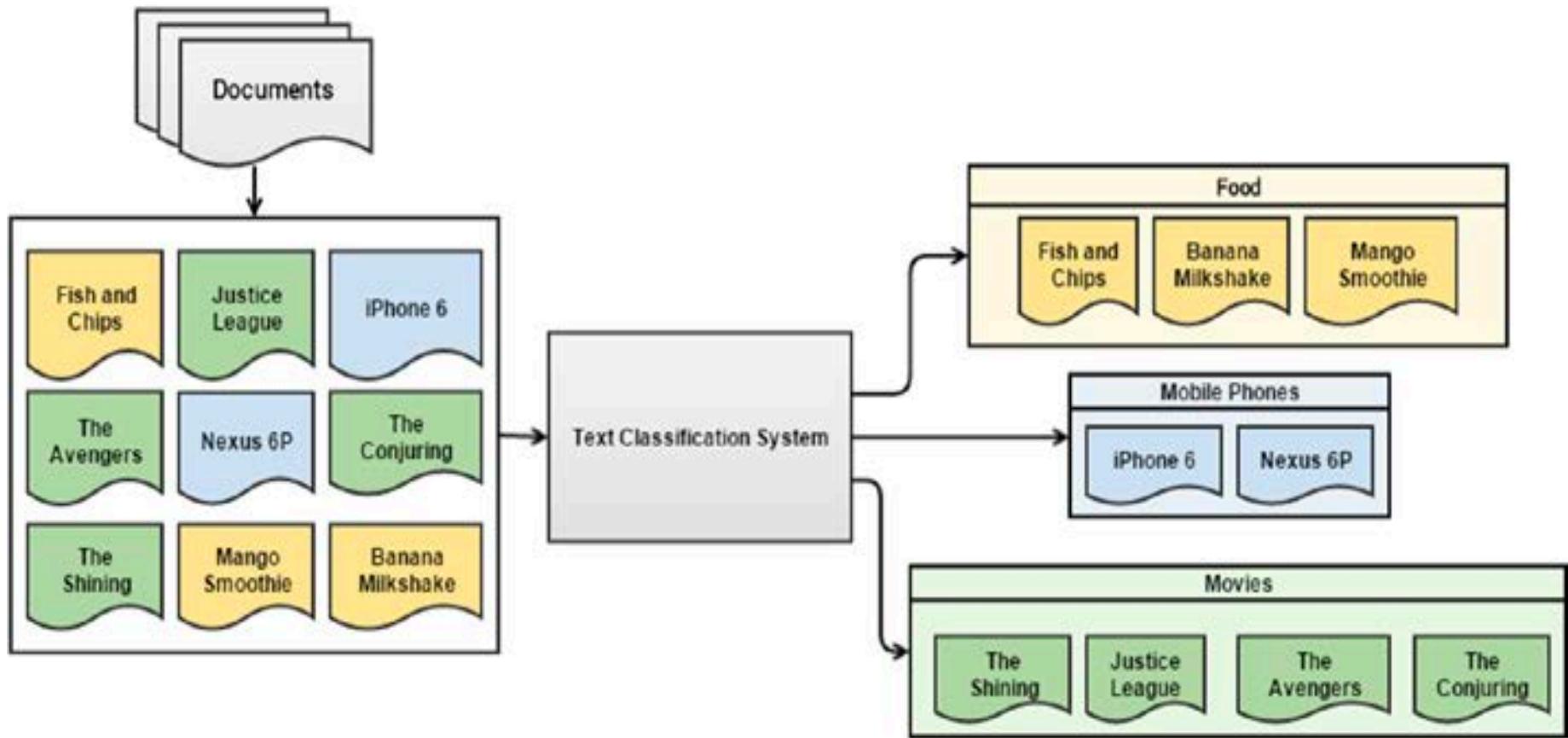
Demo

Text Analysis

Analyze complex textual data and extract meaningful patterns and insights from it.



What is Text Classification?



Source: *Text Analytics With Python : A Practical Real-World Approach To Gaining Actionable Insights From Your Data, Second Edition*, Dipanjan Sarkar, 2019, Apress L. P. (ISBN 978-1-4842-4353-4) (eBook ISBN 978-1-4842-4354-1) [Publisher Website](#)

Machine Learning Approach

Supervised Learning

- Requires labeled training data
- Each data point corresponds to a class or label
- Learns meaningful patterns for each type of class
 - Sentiment analysis
 - Spam filtering
 - Topic labeling

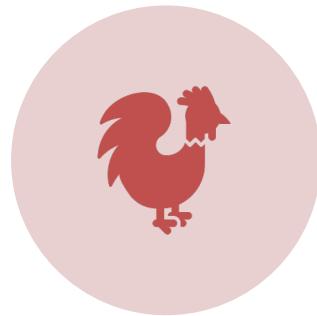
Unsupervised Learning

- Do not require any labeled training data
- Extract meaningful patterns from the data
 - Document Summarization
 - Similarity Analysis
 - Clustering (Useful in Text document categorization)

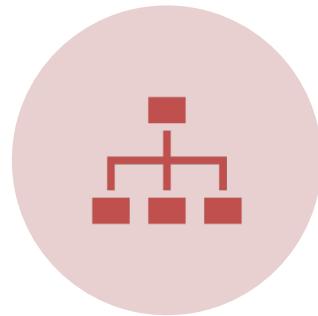
Types of Classification



BINARY
CLASSIFICATION



MULTI-CLASS
CLASSIFICATION

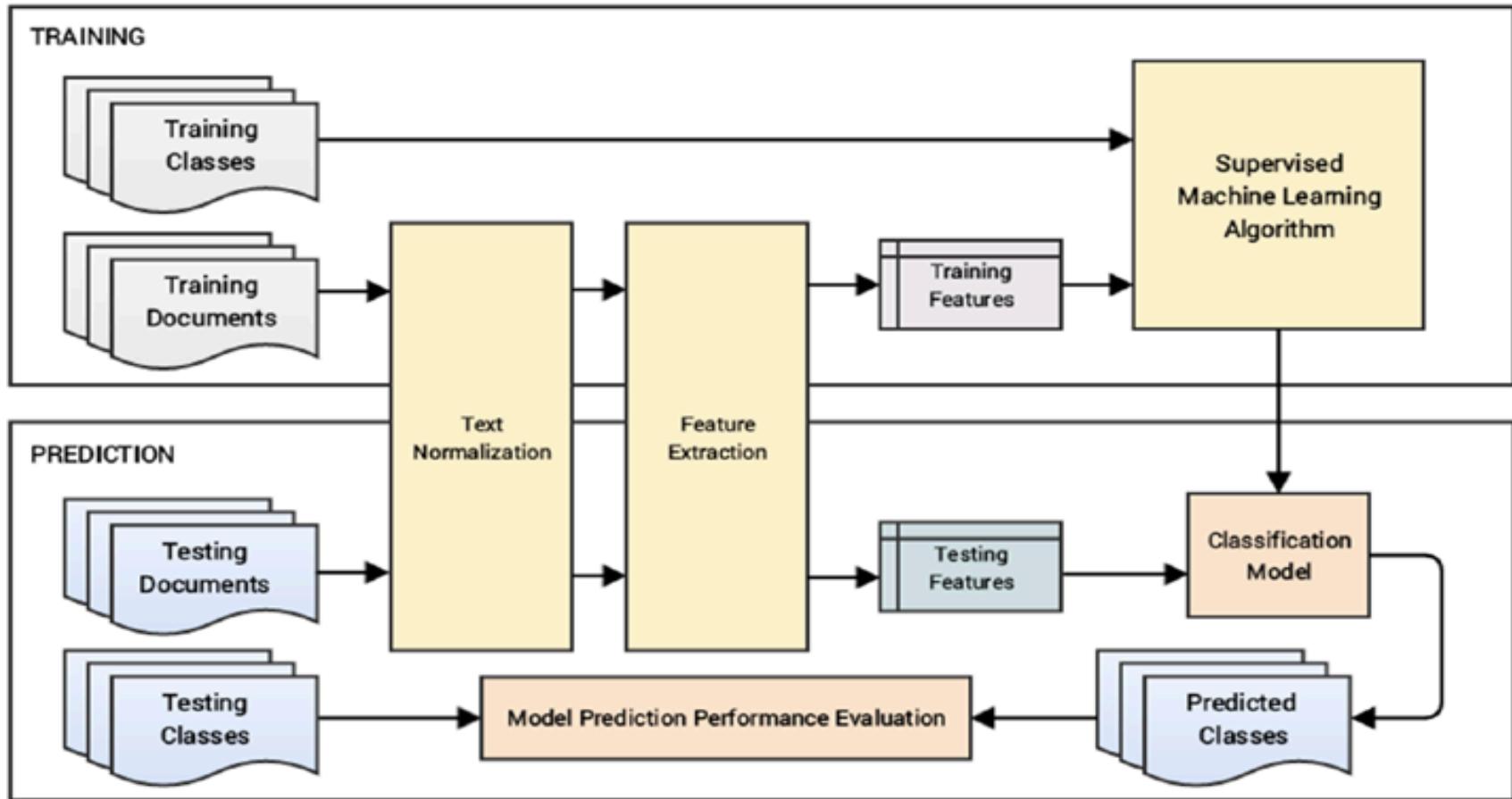


MULTI-LABEL
CLASSIFICATION

Text Classification Workflow

- A typical workflow for a text classification system is as follows:
 - Train and test datasets
 - Text Normalization
 - Feature Extraction
 - Model Training
 - Model Prediction

Architecture



Source: *Text Analytics With Python : A Practical Real-World Approach To Gaining Actionable Insights From Your Data, Second Edition*, Dipanjan Sarkar, 2019, Apress L. P. (ISBN 978-1-4842-4353-4) (eBook ISBN 978-1-4842-4354-1) [Publisher Website](#)

Text Normalization



Tokenization – Segmenting a sentence into words



Named Entity Recognition – like persons, locations or companies



Word vectors – represent the meaning of the word in form of vectors of numbers



Similarity – Comparing words or documents

Bag of Words, TF-IDF, and Advanced word
vectorization models

FEATURE EXTRACTION

Bag of Words model

- Convert text document into vectors
- Vector represents the frequency of all the distinct words for specific document
- The frequency of occurrence is the weight of each word

Useful link:

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html

Example

```
##Corpus is a collection of text:  
corpus = ['the sky is so beautiful',  
          'sky is blue and sky is beautiful',  
          'the sky is perfectly blue',  
          'i love cream cheese'  
        ]  
  
from sklearn.feature_extraction.text import CountVectorizer  
vectorizer = CountVectorizer(min_df=1, ngram_range=(1,1))  
features = vectorizer.fit_transform(corpus)  
vectorizer.get_feature_names() ##print the features name
```

```
print('The feature names: ')
print(vectorizer.get_feature_names())
print('Number of features: ', len(vectorizer.get_feature_names()))
print('The vectorized text (Encoded): ')
print(features.toarray())
```

The feature names:

['and', 'beautiful', 'blue', 'cheese', 'cream', 'is', 'love', 'perfectly', 'sky', 'so', 'the']

Number of features: 11

The vectorized text (Encoded):

```
[[0 1 0 0 0 1 0 0 1 1 1]
 [1 1 1 0 0 2 0 0 2 0 0]
 [0 0 1 0 0 1 0 1 1 0 1]
 [0 0 0 1 1 0 1 0 0 0 0]]
```

TF-IDF Model

- TF-IDF reflects the importance of a word to document in a collection
- TF - term frequency
- IDF - inverse of document frequency for each term
- TF-IDF is a combination of the above two metrics and represents a weighting of a word

Mathematically, TF-IDF is the product of two metrics and can be represented as:

$$tfidf = tf * idf$$

Useful link:

https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

Example

```
from sklearn.feature_extraction.text import TfidfVectorizer
tale_of_cities = ['it was the best of times',
                  'it was the worst of times',
                  'it was the age of wisdom',
                  'it was the age of foolishness',
                  'it was the epoch of belief'
                 ]
tfidf_vectorizer = TfidfVectorizer()
tfidf_features = tfidf_vectorizer.fit_transform(tale_of_cities)
print(tfidf_vectorizer.get_feature_names())
print(tfidf_features.shape)
```

Feature names:

```
['age', 'belief', 'best', 'epoch', 'foolishness', 'it',
 'of', 'the', 'times', 'was', 'wisdom', 'worst']
```

Number of features:

```
(5, 12)
```

```
print('The feature names: ', tfidf_vectorizer.get_feature_names())
print('Number of features: ', len(tfidf_vectorizer.get_feature_names()))
print('The vectorized text (Encoded): ')
print(tfidf_features.toarray())|
```

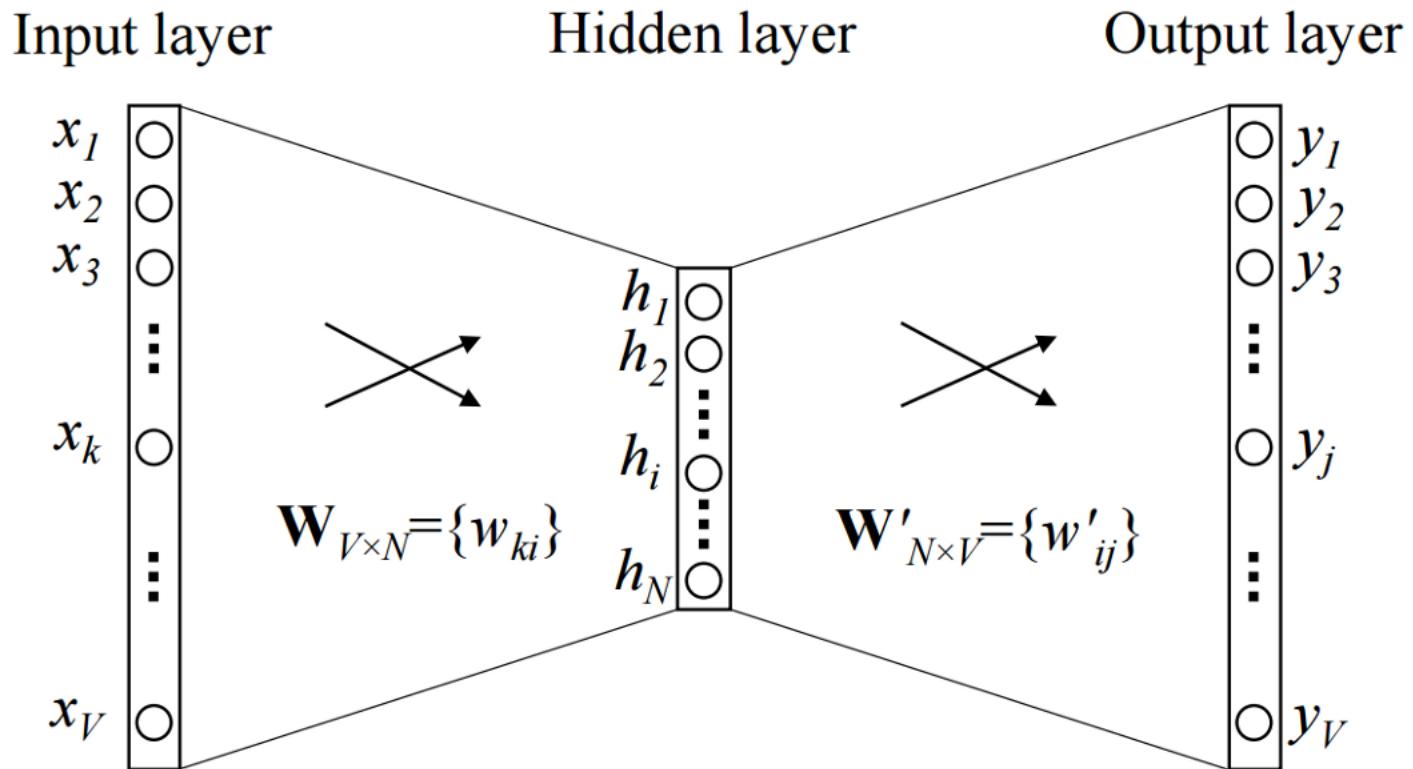
```
Number of features: 12
The vectorized text (Encoded):
[[0.          0.          0.62510433 0.          0.          0.29786556
  0.29786556 0.29786556 0.50433024 0.29786556 0.          0.          ]
 [0.          0.          0.          0.          0.          0.29786556
  0.29786556 0.29786556 0.50433024 0.29786556 0.          0.62510433]
 [0.50433024 0.          0.          0.          0.          0.29786556
  0.29786556 0.29786556 0.          0.29786556 0.62510433 0.          ]
 [0.50433024 0.          0.          0.          0.62510433 0.29786556
  0.29786556 0.29786556 0.          0.29786556 0.          0.          ]
 [0.29786556 0.29786556 0.          0.29786556 0.          0.
  0.27941741 0.27941741 0.          0.27941741 0.          0.        ]]
```

Advanced word vectorization models

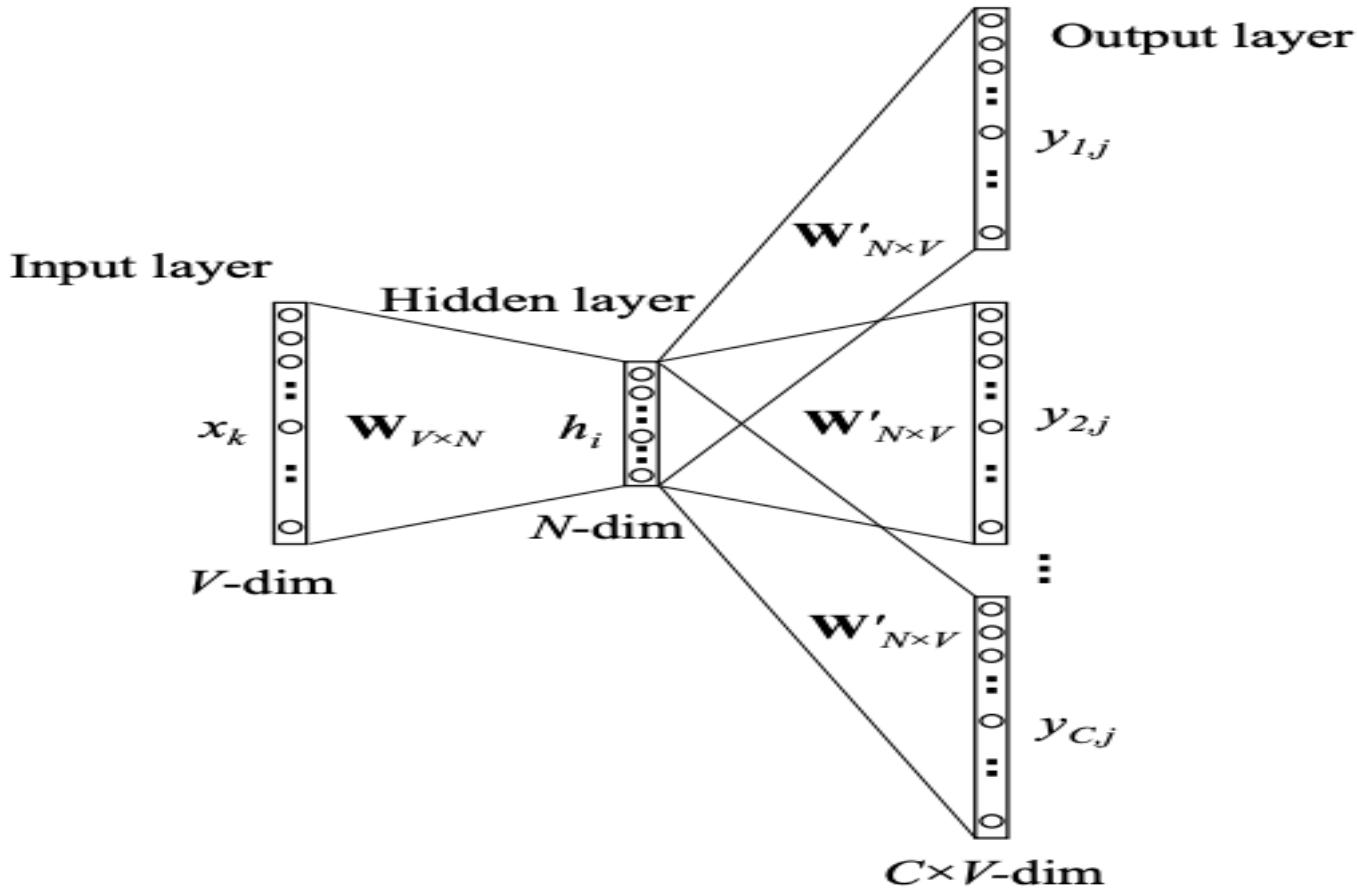
- Word2Vec - learn word embeddings
- It is a neural network-based implementation that learns distributed vector representations of words
- Gensim is a python library used for document feature extraction
- For more details on gensim:
<https://radimrehurek.com/gensim/index.html>

Source: Rong, X. (2014). word2vec parameter learning explained. *arXiv preprint arXiv:1411.2738*.

CBOW model



Skip gram model



Example

```
from gensim.test.utils import common_texts, get_tmpfile
from gensim.models import Word2Vec

# common texts data from gensim library.
common_texts
```

```
[['human', 'interface', 'computer'],
 ['survey', 'user', 'computer', 'system', 'response', 'time'],
 ['eps', 'user', 'interface', 'system'],
 ['system', 'human', 'system', 'eps'],
 ['user', 'response', 'time'],
 ['trees'],
 ['graph', 'trees'],
 ['graph', 'minors', 'trees'],
 ['graph', 'minors', 'survey']]
```

```
# Word2Vec model
model = Word2Vec(common_texts, size=100, window=5, min_count=1)

# Vector for a specific word, for ex: human
model.wv["human"]

# Calculate the most similar words to human
model.wv.most_similar("human")
```

```
[('response', 0.08395902812480927),
 ('graph', 0.07479733973741531),
 ('time', 0.06893133372068405),
 ('survey', 0.04686649888753891),
 ('system', 0.031810905784368515),
 ('computer', 0.020856063812971115),
 ('minors', -0.02121073007583618),
 ('eps', -0.06385811418294907),
 ('user', -0.07140786200761795),
 ('interface', -0.07448561489582062)]
```

Multinomial Naïve Bayes and Support Vector Machines
(Supervised machine learning algorithm to classify
data)

CLASSIFICATION ALGORITHMS

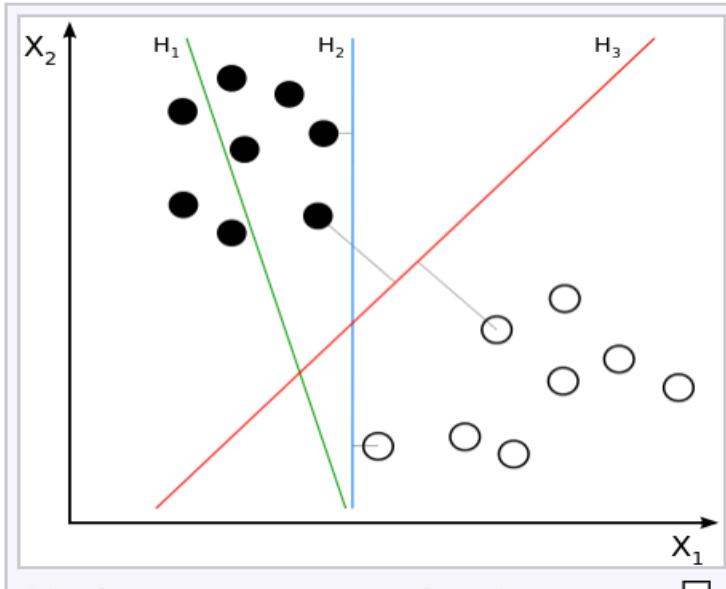
Multinomial Naïve Bayes

- This algorithm is quite effective in classifying text data.
- Special case of naïve Bayes algorithm and allows classification for problems with more than two classes

$$P(A | B) = \frac{P(B | A) P(A)}{P(B)}$$

- Assumes each feature is independent of each other
- The feature vectors are from Bag of Words model or TF-IDF
- Scikit-learn library provides an implementation of Multinomial naïve bayes (MultinomialNB)

Support Vector Machines



H_1 does not separate the classes. \square
 H_2 does, but only with a small margin.
 H_3 separates them with the maximal margin.

Supervised learning algorithm

SVM constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification, regression, or other tasks like outliers detection

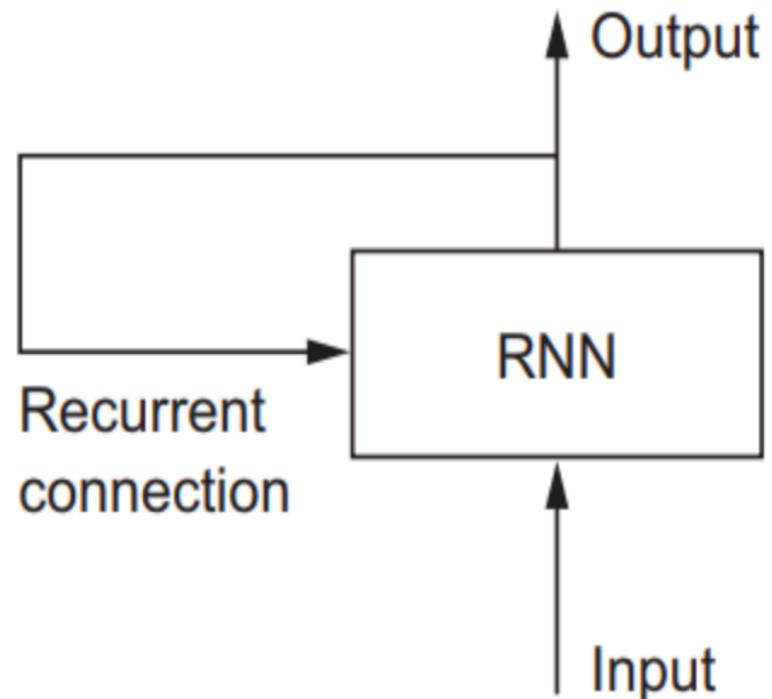
Source: https://en.wikipedia.org/wiki/Support_vector_machine

Long Short-Term Memory: Allows access to previous information to used at later time

LSTM

RNNs

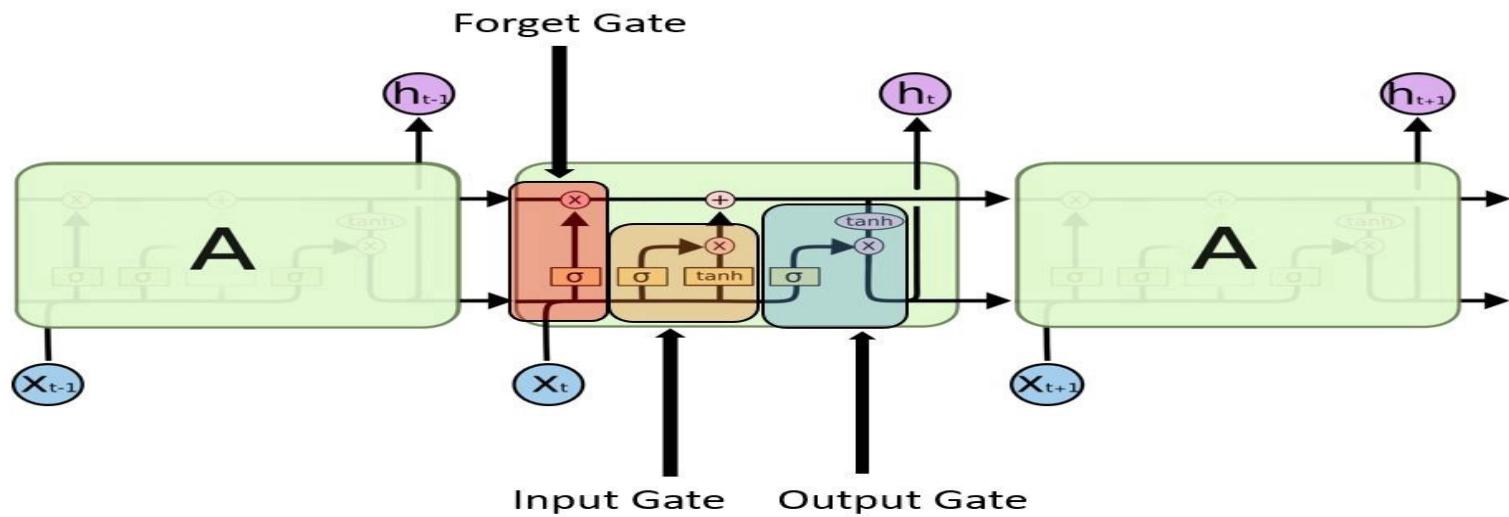
- Recurrent Neural Networks
- Is a generalization of feedforward neural network that has an internal memory
- RNNs can use their internal state (memory) to process sequences of inputs.



Source: <https://www.manning.com/books/deep-learning-with-python>

LSTMs

- It remembers the previous data



Source: <https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e>

Demo

- Implement Feature Extraction using TF-IDF
- Text classification using existing packages in Scikit learn
- Sentiment analysis of imdb dataset using Tensorflow Keras
- Text categorization using Clustering



GALLOGLY COLLEGE OF ENGINEERING
SCHOOL OF COMPUTER SCIENCE
The UNIVERSITY *of* OKLAHOMA

Questions?

Thank you!



Would you like to be involved in research at the University of Oklahoma?

The purpose of this survey is to understand attendees' knowledge of Python and intended outcomes from attending these sessions. We are trying to gauge the demand of skills, resources, and knowledge of Python programming and associated techniques. This will, in turn, help us understand the success of the workshop and room for improvement.



SCAN ME