Text Classification

Russ M. Delos Santos

Overview

Research:
Text classification for RAX Studio

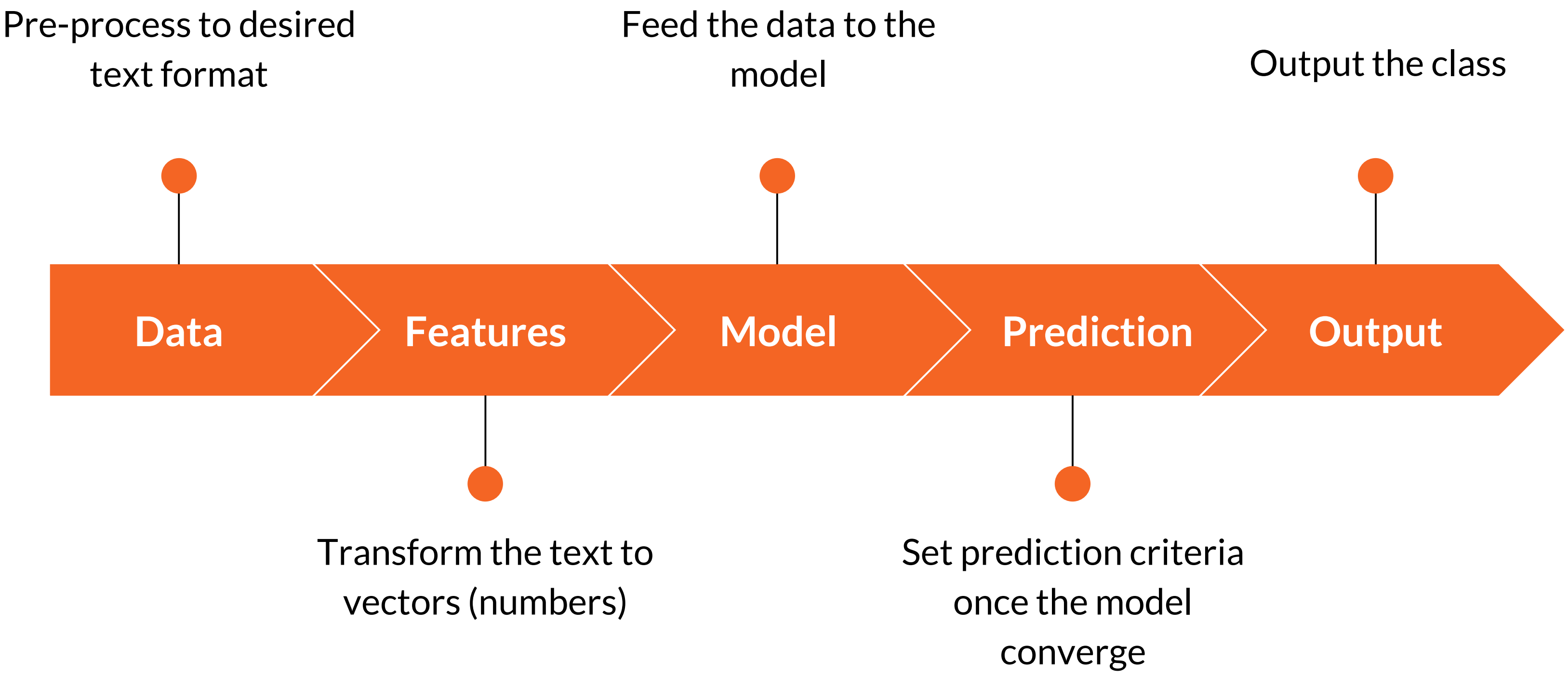
Suggested Use Case:

- Account management through email.

Natural Language Processing

1. Automatic or semi-automatic processing of human language
2. Can be used for various applications like
    a. Sentiment Analysis
    b. Intent Classification
    c. Topic Labeling

General Process



Dataset / Text Corpus

- **Dictionary or vocabulary which is used to train the model**

  - Either tagged (for supervised learning) or untagged (for unsupervised).
  - Size depends on the algorithm used. Should be pre-processed to remove unwanted characters, to convert to wanted format, etc.

Dataset / Text Corpus

- Open-source dataset samples

- Amazon Reviews
- NYSK Dataset (News Articles
- Enrol Email Dataset
- Ling Spam Dataset

Feature Extraction

- **Transforms texts to numbers (vector space model)**
- **Choices:**
  - One-hot encoding
  - Bag-of-words + TF*IDF
  - Word2vec

One-hot encoding

- **Creates a binary encoding of words. 1 is encoded on the index of the word in the corpus**

Paris
Rome
word V

```
Rome   = [1, 0, 0, 0, 0, 0, ..., 0]
Paris  = [0, 1, 0, 0, 0, 0, ..., 0]
Italy  = [0, 0, 1, 0, 0, 0, ..., 0]
France = [0, 0, 0, 1, 0, 0, ..., 0]
```

Bag-of-words

- Takes the word count of the target word in the corpus as the feature

	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

TF*IDF

- **Term Frequency * Inverse Document Frequency**
    - Frequently occurring words are typically not important / has less weight (stopwords such as “is, are, the, etc.”)
    - Weights are assigned per word.

TF*IDF

- **Term Frequency * Inverse Document Frequency**

TF-IDF Score

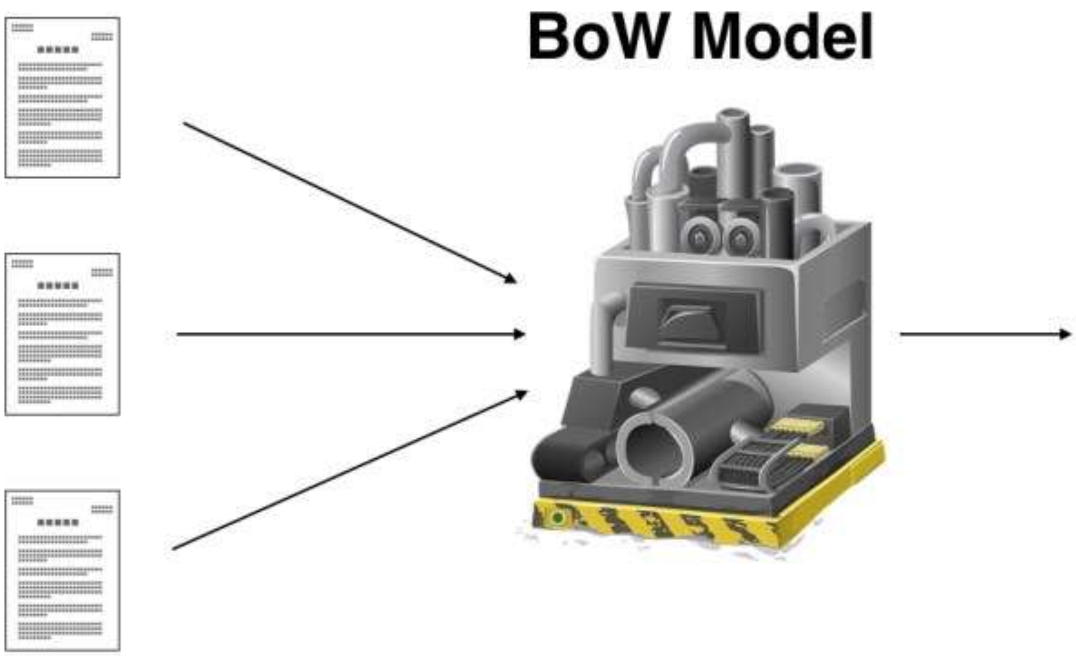
$$TF - IDF\ Score = TF_{x,y} * IDF = TF_{x,y} * \log\frac{N}{df} \ldots\ldots(1)$$

, where $TF_{x,y}$ *is the frequency of keyphrase X in the article Y,*

N is the total number of documents in the corpus.

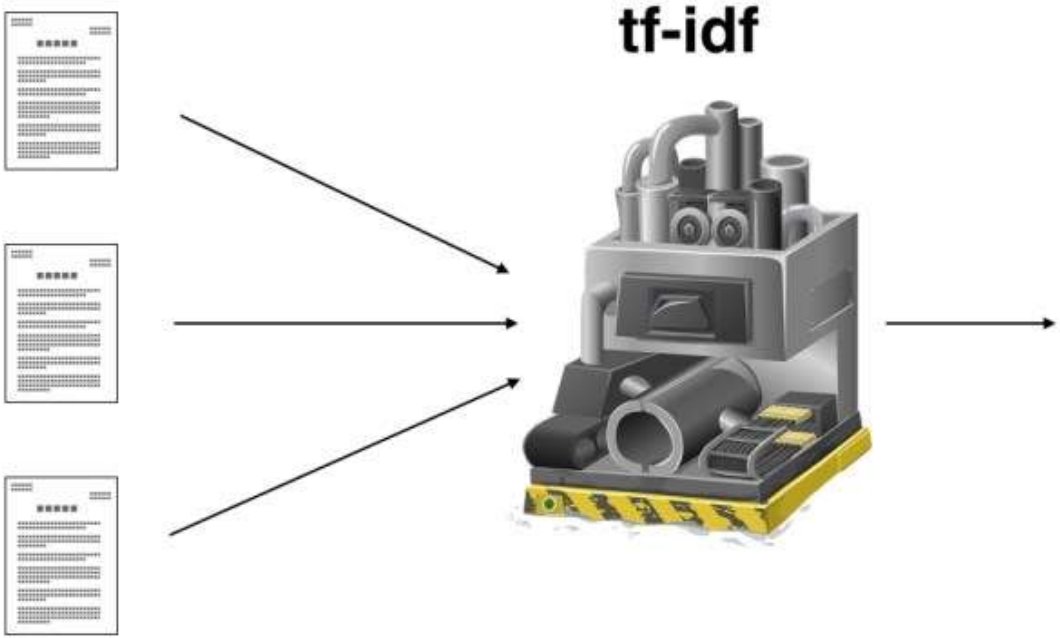
df is the number of documents containing keyphrase X

BOW + TF*IDF



	I	love	dogs	hate	and	knittin	is	my	hobby	passi
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

BOW + TF*IDF



	I	love	dogs	hate	and	knittin	is	my	hobby	passi
Doc 1	0.18	0.48	0.18							
Doc 2	0.18		0.18	0.48	0.18	0.18				
Doc 3					0.18	0.18	0.48	0.95	0.48	0.48

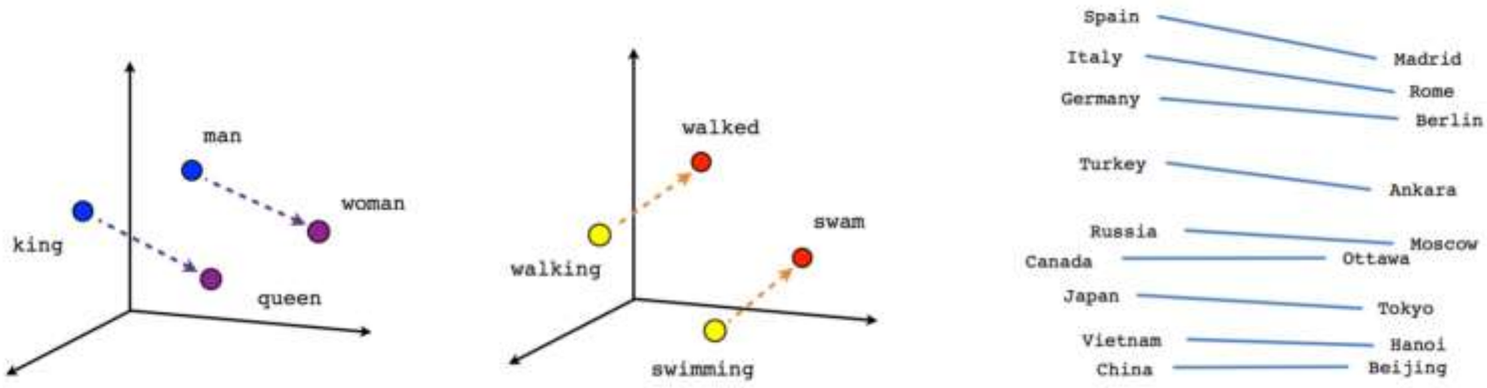
word2vec

Uses the weights of the hidden layer of a neural network as features of the words

- Can predict a context or a word based on the nearby words in the corpus
- Uses continuous bag-of-words or skip-gram model + 1-1-1 neural network.

word2vec

- Gives better semantic/syntactic relationships of words through vectors



Male-Female Verb tense Country-Capital


Figure 1
was
infant
and
prisoners
them
as
daughter
who
kingdom
madam
enemy
lived
think
his
near
i
tower
corpus_raw = 'Once upon a time a wicked King invaded a Queens land, seized the \
Queen and her infant daughter as prisoners and shut them both into \
the highest room of a tall tower . The room was very tiny and empty \
with only one table and a very hard bed on the floor . Then the enemy \
King sent for a fairy who lived near his kingdom . He nearly pushed \
the Fairy up the stairs to see the Queen . The Fairy was so touched \
by the sight of the Queens miserable condition that when she kissed \
the Queens hand the Fairy whispered to her Courage madam I think I \
see a way to help you .'
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
x=0.361073 y=0.516982
ENG
11:10 AM

Machine Learning Model

- **A classifier algorithm that transforms an input to the desired class**

- Naive Bayes
- K-nearest neighbors
- Multilayer Perceptron
- Recurrent Neural Network + Long short-term memory

Naive Bayes

- **Probabilistic model that relies on word count**

  - Uses bag of words as features
  - Assumes that the position of words doesn't matter and words are independent of each other

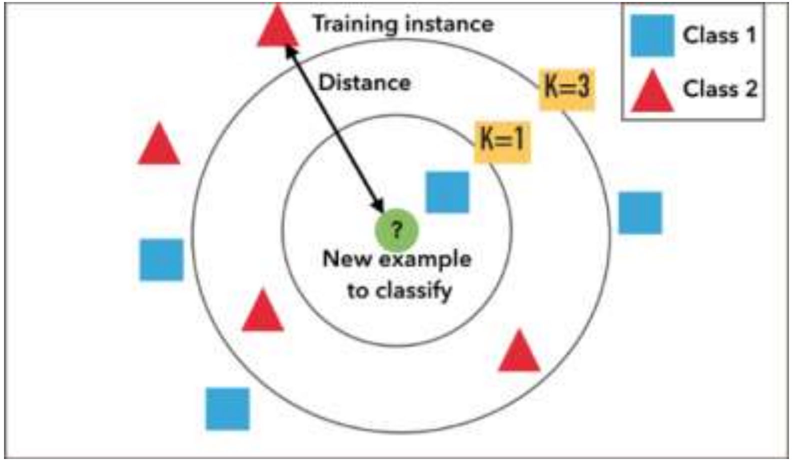
Naive Bayes

- **Probabilistic model that relies on word count**

$$P(spam|nigerian, prince) = \frac{P(nigerian|spam) * P(prince|spam) * P(spam)}{P(nigerian) * P(prince)}$$

K-Nearest Neighbors

- Classifies the class based on the nearest distance from a known class

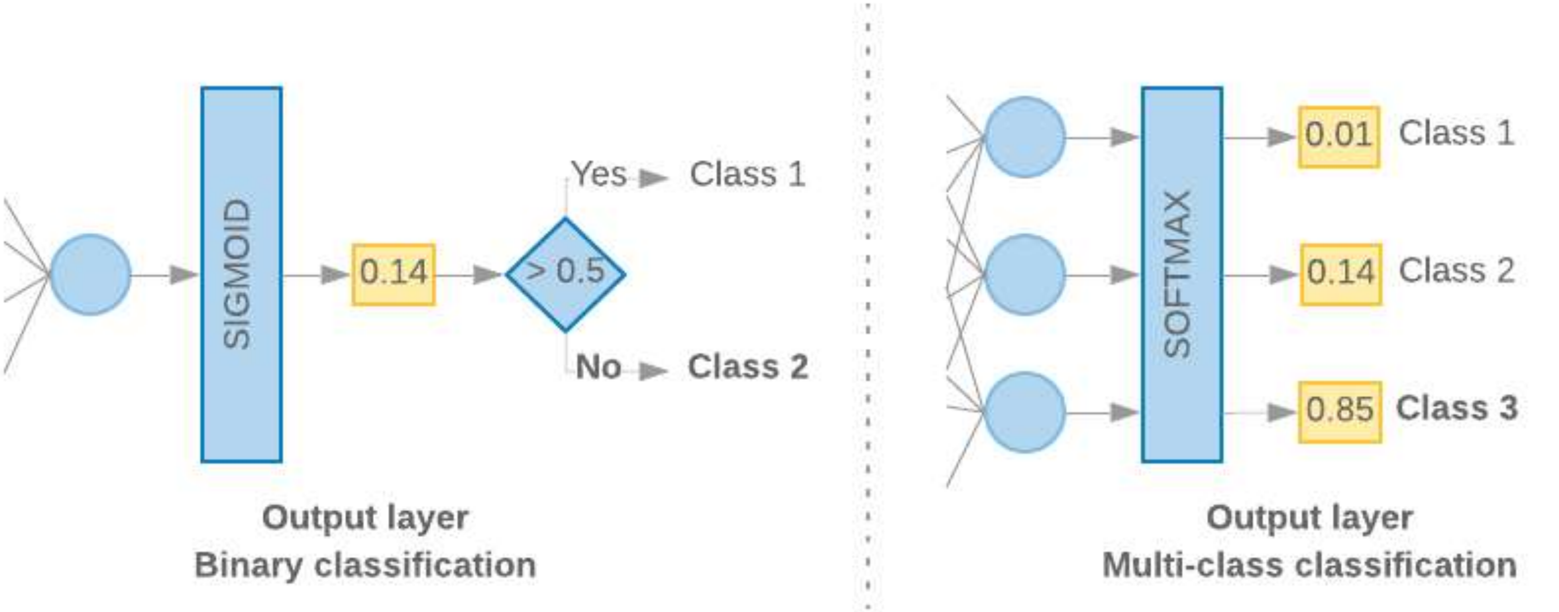


Multilayer Perceptron

- **A feed-forward neural network**

  - Has at least 2 hidden layers
  - Sigmoid function - binary classification
  - Softmax function - multiclass classification

Multilayer Perceptron



Assessment

Option 1

- Features: BOW + TF*IDF
- ML Algorithm: Naive Bayes
- Pros: Easier to implement
- Cons: Word count instead of word sequence.
- Ex. ‘Live to eat’ and ‘Eat to live’ may mean the same’

Option 2

- Features: word2vec word embeddings
- ML Algorithm: Multilayer Perceptron
- Pros: Produces better results, semantically and syntactically
- Cons: Needs a big labeled dataset to perform well

Main Blocks

ML.NET Learning Curve

- Still studying the framework.
- Not as well documented compared to Python frameworks/libraries
- Ex. Has a method called TextCatalog.FeaturizeText() but there's no indication of the kind of feature extraction.

Supervised Learning Needs Big Data

- We can use open-source datasets for benchmark.
- But we need datasets with specific labels for the algorithm to work.

Main Blocks

Model Update Criteria

- Retraining the model for every unknown word is impractical.
- Suggestion:
    - Set a minimum number of occurence of new words before a model is to be retrained
    - Ignore the rare, new words since it may not affect the entire intent, sentiment, meaning, of the text.

Implementation Plan

- Email Cleaner
  - Clean special characters, HTML tags, header and footer of the email, etc.
  - Set a standard file format (tsv, csv, txt, etc. or transform to bin)
  - Use spam dataset for the mean time as benchmark (binary classification)
- Sentence Tokenizer + Feature Extraction
  - Divide emails per sentence + word2vec
- Create Neural Network
  - 1 input, 2 hidden, 1 output.
  - Activation function - sigmoid

References

[1]D. Jurafsky and J. Martin, Speech and language processing. Upper Saddle River, N.J.: Pearson Prentice Hall, 2009.

[2]https://developers.google.com/machine-learning/

[3]bunch of stackoverflow / stackexchange / Kaggle threads

[4]bunch of Medium posts