



Meeting #2

William Sengstock, Kelly Jacobson, Zach
Witte



Tokenizers

- Word Tokenize
 - Gives ID's to each whole word in a sentence, counts the amount of time a word appears by its ID
 - Problems:
 - List can end up being extremely large, results in heavy models
 - Similar words get completely different ID's
 - dog and dogs
- Character Tokenize
 - Splitting a message into individual characters
 - Reduces memory and time complexity: character-based vocab > word-based vocab
 - Problems:
 - Need more context than if taking in a whole word
- Subword Tokenize
 - Frequently used words shouldn't be split into smaller subwords
 - Rare words should be broken into meaningful subwords
 - Algorithms: WordPiece, Unigram, Byte-Pair Encoding



Vectorization Strategy: Count Vectoring

- Document Term Matrix
 - Each cell represents a count of how many times that particular word appears in the message
 - Each column represents the individual word in the message
- Can be used to determine if the message is fake
 - (Ex). Count Vectoring is used to determine if news articles are fake or not
 - If a word appears multiple times in fake or real articles, can be used to determine if other articles are real or fake
- Downside: if message is lengthy, will result in huge matrix with lots of information to look through



Vectorization Strategy: N-Gram

- Count Similarities
 - Both use a Document Term Matrix
- Differences
 - Count is the combination of adjacent words in the message (of length n)
 - (Ex). "I love to sleep" has a total of 4 words and a count of $n = 4$
 - If $n = 2$, the columns would be - "I love", "love to", "to sleep"
 - If $n = 3$, the columns would be - "I love to", "love to sleep"
 - Etc.
 - Count Vectorization is where the N-Gram count is $n = 1$
- Need to be careful with N-Gram
 - Smaller N value might not be sufficient enough to display most useful information
 - Too high of an N value means huge matrix with a ton of information to go through



Vectorization Strategy: TF-IDF

- Each cell doesn't represent term frequency, but represents a weight that determines the importance of that word

$$W_{x,y} = tf_{x,y} * \log\left(\frac{N}{df_x}\right)$$

$W_{x,y}$ = Word x within document y

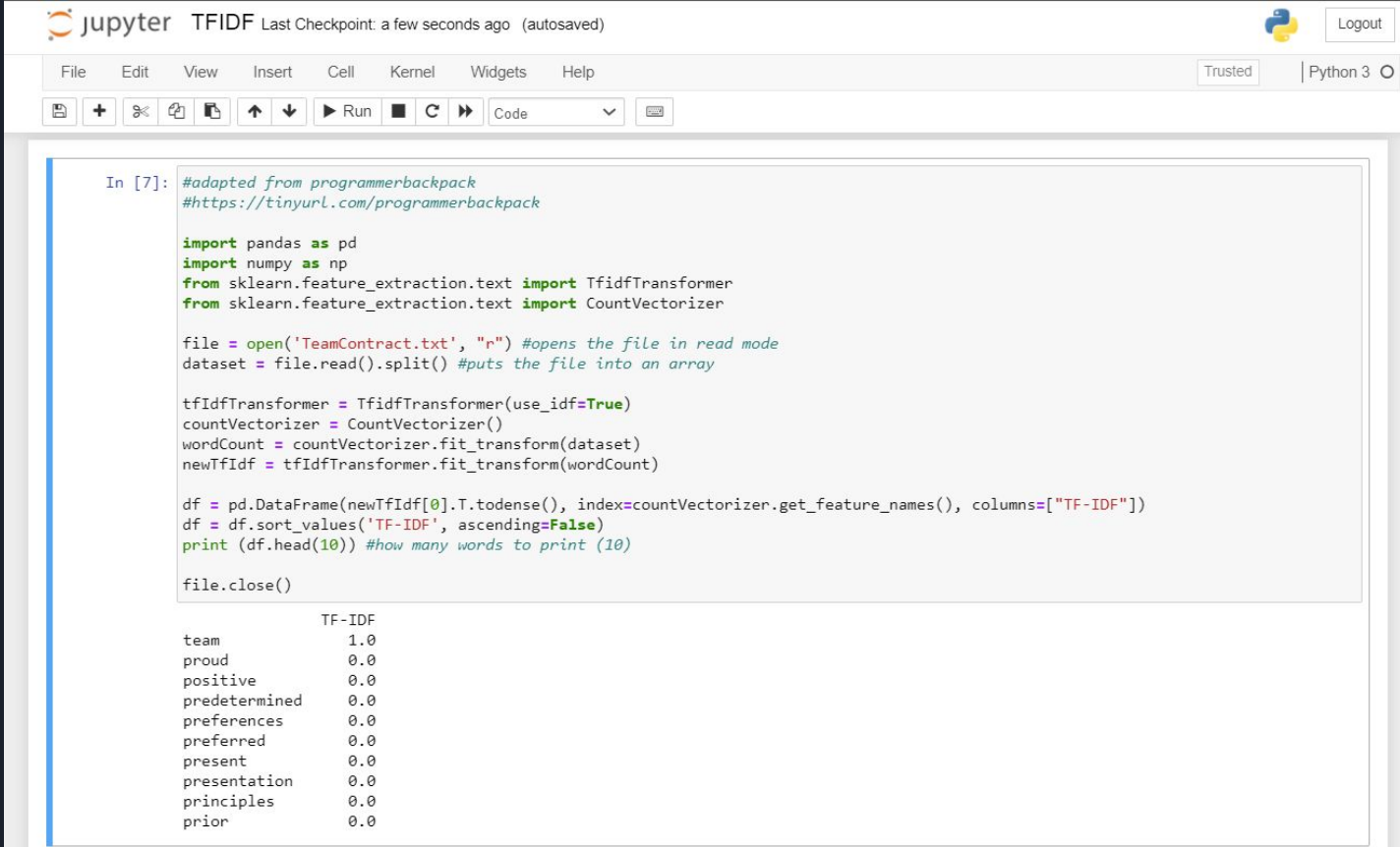
$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

- Term frequency (tf) = % of number of times a word (x) appears in that document (y) / total number of words in y
 - Helps determine “rare” words
- Requires more calculations than the others
- Focuses on the frequency of words, as well as their importance

TF-IDF Implementation in Jupyter Notebook



The image shows a Jupyter Notebook interface with a dark theme. The top bar includes the Jupyter logo, the title 'TFIDF', and a status message 'Last Checkpoint: a few seconds ago (autosaved)'. On the right, there is a Python logo and a 'Logout' button. Below the top bar is a menu bar with 'File', 'Edit', 'View', 'Insert', 'Cell', 'Kernel', 'Widgets', and 'Help'. A 'Trusted' badge and 'Python 3' are also visible. The main toolbar contains icons for saving, adding, deleting, copying, pasting, undo, redo, and running cells. The code area shows a Python script for TF-IDF implementation. The output area displays a table of words and their TF-IDF values.

```
In [7]: #adapted from programmerbackpack
#https://tinyurl.com/programmerbackpack

import pandas as pd
import numpy as np
from sklearn.feature_extraction.text import TfidfTransformer
from sklearn.feature_extraction.text import CountVectorizer

file = open('TeamContract.txt', "r") #opens the file in read mode
dataset = file.read().split() #puts the file into an array

tfidfTransformer = TfidfTransformer(use_idf=True)
countVectorizer = CountVectorizer()
wordCount = countVectorizer.fit_transform(dataset)
newTfidf = tfidfTransformer.fit_transform(wordCount)

df = pd.DataFrame(newTfidf[0].T.todense(), index=countVectorizer.get_feature_names(), columns=["TF-IDF"])
df = df.sort_values('TF-IDF', ascending=False)
print (df.head(10)) #how many words to print (10)

file.close()
```

	TF-IDF
team	1.0
proud	0.0
positive	0.0
predetermined	0.0
preferences	0.0
preferred	0.0
present	0.0
presentation	0.0
principles	0.0
prior	0.0



word2vec & Fasttext vectorization

word2vec

- Skip-Gram
 - given a single word, attempts to predict the context that word might be used in
 - needs a set of training data, but can work for even rare words (tensorflow)
- CBOW
 - given a set of words in context, predicts what word will be used next
 - faster than skip-gram, and better at identifying frequently used words (tensorflow)

Fasttext

- also supports Skip-Gram and CBOW models
- treats each word as an object composed of character n-grams (thinkinfi)
- [fastText](#) has pretrained libraries for 157 human-language libraries, but no coding languages



NLP Toolkits

- NLTK - best for simple text analysis, not for massive amounts of data
- Stanford NLP - similar to NLTK but scalable to large amounts of data
- spaCy - better for complex applications, best for comparisons, API has everything included, better for market research (theappsolutions)
- SyntaxNet - Google made, neural network framework, relatively new
- OpenNLP - simple, best for detection and recognition



IEEE recommendations for Analyzing Software Documentation (IEEE)

- NLPCompare to compare performance
- experimented on Java programming from Stack Overflow, GitHub ReadMe files, Java API Documentation with random sampling
- compared Stanford NLP, SyntaxNet, spaCy, NLTK against Manual Annotation
- NLTK - best for tokenization and comparison
- spaCy and NLTK best for General POS Tagging
- spaCy and SyntaxNet best for Specific POS Tagging (NLTK the worst)



What is Unsupervised Learning

- **Unsupervised learning** involves deriving structure from data where we don't know the effect of any of the variables
 - There is nothing in particular that you are trying to predict because there are no labels tied to the data to help make that prediction
- **Clustering methods** are the most common evaluation methods used in unsupervised learning
 - “**Clustering** is the grouping of objects or data points that are similar to each other and dissimilar to objects in other clusters.”
 - A **cluster** is “a collection of data points aggregated together because of certain similarities”
 - Popular clustering methods include: K-means Clustering, Density-Based Clustering, and Hierarchical Clustering.



K-Means Clustering

- Groups similar data points together and looks for underlying patterns by looking for k clusters in a dataset
- Identify k centroids (center of a cluster) and place every data point in the nearest cluster while keeping the centroids as small as possible
- Steps:
 - Starts with group of randomly selected centroids
 - Iterative calculations to optimize positions of centroids
 - Stops when there is no change in centroid values (successful clustering) OR the specified number of iterations has completed
- Easy to understand and grants fast results but might not be as competitive as other clustering algorithms.



Density-Based Clustering

- Detects areas of concentrated data points that are separated by empty or sparse areas. Data points not in a cluster are called “noise”
- Detect patterns based on spatial location and distance to a specified number of neighbors
- Time can also be included to create space-time clusters
- Three options for Density-Based Clustering
 - **Defined distance** (DBSCAN)-> Specified distance to separate dense clusters from noise. Fastest but needs very clear search distance that works well for all clusters. (All clusters have similar densities)
 - **Self-adjusting** (HDBSCAN)-> Range of distances to separate clusters from noise. Varying densities is fine. Most data-driven = least user input
 - **Multi-scale** (OPTICS) -> Orders input points based on smallest distance to next point. Constructs reachability plot. Clusters formed based on fewest points to be considered cluster, search distance, and characteristics of reachability plot. (ex. slope) Most computationally intensive but can use time and search time interval fields to find clusters in space and time

<https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-statistics/how-density-based-clustering-works.htm>



Hierarchical Clustering

- Two types
 - Agglomerative
 - Divisive
- Agglomerative
 - Treats each data point as a separate cluster
 - Repeatedly identifies pairs of two clusters that are closest together and then merges the most similar cluster pair
 - Once all clusters have been merged, a dendrogram is created which shows the hierarchical relationship between clusters
- Divisive
 - All data points start as a single cluster
 - Separate data points from the cluster that aren't similar until there are n clusters
 - Not used much in the real world
- Similarity between clusters can be calculated through things such as min, max, average, distance between centroids etc.
- Disadvantages: high space and time complexity, and all approaches to finding similarities have their own pros and cons



Advantages to Unsupervised Learning

Advantages

- Labeling data requires a lot of manual work and expenses
 - Faster to label after data has been classified
- Can work with real-time data to identify patterns
- Dimensionality reduction can be easily accomplished
 - Reducing number of features to make model less complex
- Find up to what degree the data are similar using probabilistic methods



Disadvantages to Unsupervised Learning

Disadvantages

- Uncertainty about the accuracy of the unsupervised learning outputs
- Result may be less accurate since there is no input data to train from
- Process as a whole is time consuming
- Lack of full insight into how or why the system reached its results
- Complexity increases fast the more features there are



Additional Datasets

<https://www.kaggle.com/prondeau/superbowlads>

<https://www.kaggle.com/agajorte/zoo-animals-extended-dataset?select=zoo3.csv>

<https://www.kaggle.com/ieffgallini/college-football-attendance-2000-to-2018>



Sources

<https://towardsdatascience.com/nlp-in-python-vectorizing-a2b4fc1a339e>

https://huggingface.co/transformers/tokenizer_summary.html

<https://www.linkedin.com/learning/nlp-with-python-for-machine-learning-essential-training/what-is-machine-learning?u=42520596>

<https://searchenterpriseai.techtarget.com/definition/unsupervised-learning>

<https://techvidvan.com/tutorials/unsupervised-learning/>

<https://towardsdatascience.com/dimensionality-reduction-for-machine-learning-80a46c2ebb7e>

<https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1>

<https://pro.arcgis.com/en/pro-app/latest/tool-reference/spatial-statistics/how-density-based-clustering-works.htm>

<https://towardsdatascience.com/understanding-the-concept-of-hierarchical-clustering-technique-c6e8243758ec>



Sources cont'd

[TF-IDF Explained And Python Sklearn Implementation \(programmerbackpack.com\)](#)

<https://www.tensorflow.org/tutorials/text/word2vec>

<https://thinkinfi.com/fasttext-word-embeddings-python-implementation/>

<https://fasttext.cc/>

<https://theappsolutions.com/blog/development/nlp-tools/>

<https://ieeexplore.ieee.org/document/7962368>