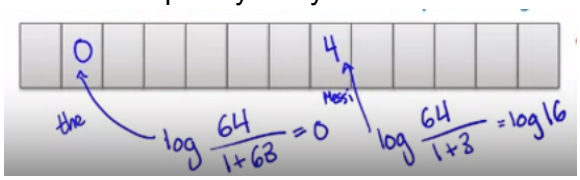
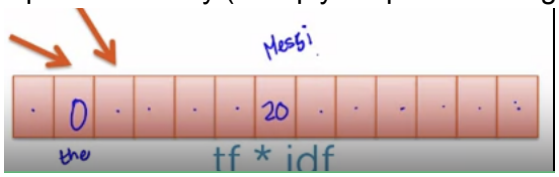
Document Retrieval	-
Overall process	In order to retrieve articles that Carlos is interested in reading.. 1. How to measure similarity between articles 2. How to search over the articles that exist out there and retrieve the next article to recommend
Measuring Similarity	Comparing the similarity between two documents where each word is an index and the vector contains the count of each word. 1 0 0 0 5 3 0 0 1 0 0 0 0 3 0 0 0 2 0 0 1 0 1 0 0 0 1*3 + 5*2 **= 13** Shortfall with this method is, if the document doubles in length, then the number of similarities increases to 52. Therefore, we take the norm of a vector for even distribution. 1 0 0 0 5 3 0 0 1 0 0 0 0 $\sqrt{(1^2 + 5^2 + 3^2 + 1^2)}$ 1/6 0 0 0 5/6 3/6 0 0 1/6 0 0 0 0
Prioritizing important words with tf-idf	**Rare Words** - words that appear infrequently but pertinent to the topic/article (futbol, messi) **Important Words** - common locally in the document but rare globally (soccer, field, goal). There can be a trade-off between local frequency and global rarity **TF-IDF** (Term Frequency - Inverse Document Frequency) - a function to weigh the rare words more than the common words - Step 1: List out the Term Frequency

<table>
<tr>
<td></td>
<td>


- Step 2: Apply the log function to create a inverse document frequency array

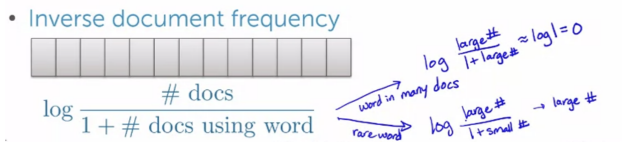


- Step 3: tf*idf array (multiply Step 1 and 2 together)



TF-IDF Logic

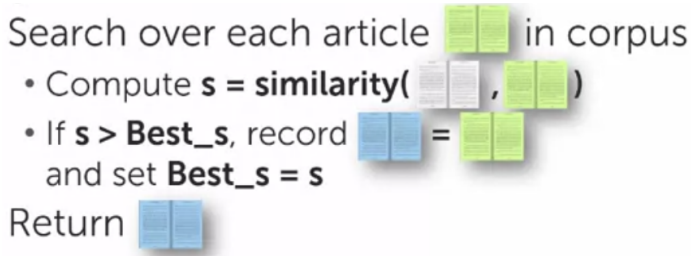
- Assumes log base 2
- Common words will be closer to 0 since # of docs the word is in will be high, causing log1 which equals 0



</td>
</tr>
<tr>
<td>Retrieving Similar Documents</td>
<td>
Present someone with another relevant article recommendation

1-Nearest Neighbor Approach

- Input is the query article, output is the most similar article
- Algorithm: We search over each article in the corpus, calculate the similarity using TF-IDF and keep tabs on the best similarity score.



K-Nearest Neighbor Approach

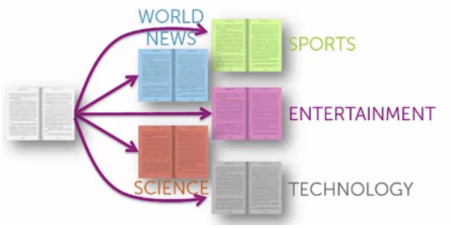
- Input is the query article, output is list of k similar article
- Algorithm is nearly the same. We search over each article i the corpus, calculate the similarity using TF-IDF, and return the top n articles with the highest similarity score
</td>
</tr>
</table>

<table>
<tr><td></td><td>  </td></tr>
<tr><td>Clustering Documents Task Overview</td><td>

How to quickly discover groups (clusters) of related articles instead of sorting through them one by one.

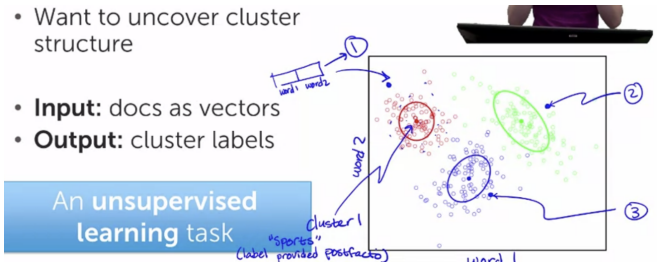
Supervised Learning: Multiclass classification problem

- Query article is not labeled, but labels are provided (corpus is already grouped)

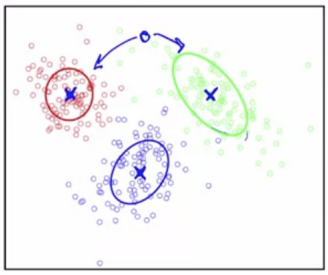


Unsupervised Learning:

- No labels are provided and want to uncover the cluster structure

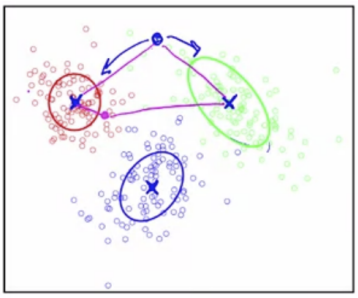
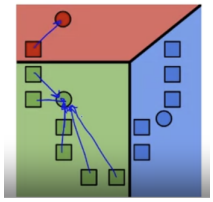
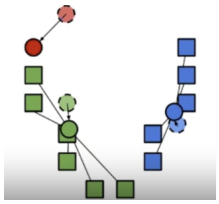


- The outputs shows the shapes of the clusters and a label is provided post-facto
- Each cluster has a center. To measure new observations, we can do two things
    1. Based on shape: observation is scored against the cluster center and the shape of the cluster is considered. In this case, it would be assigned to the oblong green shape



    2. Based on distance: observation is grouped to the cluster whose center is the shortest distance from the observation.

</td></tr>
</table>

	
Clustering Algorithm: K-Means	Clustering that uses distance from cluster center Step 1: Initialize cluster centers - specify the number of clusters ahead of time. So we are looking at k number of clusters (3) Step 2: Assign observation to the closest cluster center - Use Voronoi Tessellation to find clusters where the items are connected to their closest centers  Step 3: Revise cluster centers - Revise as new observations come in, as I update my definition of the cluster center - Redraw the Voronoi Tessellation and reassign my observations to the nearest cluster center.  Step 4: Iterate 1+2 until convergence. 
Other examples of clustering application	1. Image search 2. Grouping patients by medical conditions 3. Discover product categories from purchase histories 4. Structuring web search results 5. Discovering Similar Neighborhoods based on price of living

ML Blocking Diagram	1. Training Data (doc id, document text table) then extract the features with tf-idf. 2. The x (inputs) get put into a ML model that outputs (y-hat) cluster label 3. To assess the accuracy of the cluster though we don't have a true cluster label to compare against, the y doesn't exist. Therefore we look at how coherent the clustering is by seeing if the distance from inputs (x) to cluster centers (w-hat) are minimized. 4. ML algorithm iteratively updates the cluster center to minimize that distance