**CSE6242 Video Lecture Transcripts**

(Originally compiled from Prof. Polo Chau’s video lecture transcripts and edited by Richard Levine)

**Table of Contents**

[**Week 1**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.gjdgxs)[**1**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.gjdgxs)

[Course Introduction](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.30j0zll)        [1](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.30j0zll)

[Hi I'm Polo](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1fob9te)        [1](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1fob9te)

[Why Data and Visual Analytics?](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3znysh7)        [5](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3znysh7)

[Course Goals & Expectations](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2et92p0)        [7](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2et92p0)

[Analytics Building Blocks](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.tyjcwt)        [8](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.tyjcwt)

[Overview](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3dy6vkm)        [9](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3dy6vkm)

[Example Project 1: Apolo Graph Exploration](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1t3h5sf)        [9](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1t3h5sf)

[Example Project 2: NetProbe Auction Fraud Detection](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4d34og8)        [12](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4d34og8)

[Data Science Buzzwords](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2s8eyo1)        [14](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2s8eyo1)

[Hype Cycle](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.17dp8vu)        [14](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.17dp8vu)

[General AI vs Narrow AI](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3rdcrjn)        [15](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3rdcrjn)

[Data Collection](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.26in1rg)        [17](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.26in1rg)

[How to Collect Data?](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.lnxbz9)        [17](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.lnxbz9)

[How to Scrape?](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.35nkun2)        [18](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.35nkun2)

[Popular Scraping Libraries](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1ksv4uv)        [20](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1ksv4uv)

[**Week 2**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.44sinio)[**20**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.44sinio)

[SQLite](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2jxsxqh)        [20](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2jxsxqh)

[Simple, Effective Storage](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.z337ya)        [20](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.z337ya)

[SQL Refresher](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3j2qqm3)        [22](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3j2qqm3)

[Beware of Missing Indexes](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1y810tw)        [24](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1y810tw)

[Data Cleaning](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4i7ojhp)        [25](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4i7ojhp)

[How Dirty is Real Data?](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2xcytpi)        [25](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2xcytpi)

[Importance of Data Cleaning](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1ci93xb)        [26](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1ci93xb)

[Data Cleaners: Open Refine & Wrangler](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3whwml4)        [27](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3whwml4)

[Class Project Overview](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2bn6wsx)        [28](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2bn6wsx)

[Forming Great Teams](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.qsh70q)        [28](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.qsh70q)

[Core Project Requirements](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3as4poj)        [29](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3as4poj)

[Project Idea Checklist: Heilmeier Questions](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1pxezwc)        [30](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1pxezwc)

[Pay Attention to Software Licenses Early On](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.49x2ik5)        [31](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.49x2ik5)

[Code Backup and Version Control](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2p2csry)        [32](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2p2csry)

[Git: Overview and Benefits](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.147n2zr)        [32](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.147n2zr)

[Warning! Keep Your Repository Private Initially](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3o7alnk)        [33](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3o7alnk)

[**Week 3**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.23ckvvd)[**34**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.23ckvvd)

[Data Integration](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.ihv636)        [34](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.ihv636)

[Knowledge Graph](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.32hioqz)        [34](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.32hioqz)

[Data De-Duplication](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1hmsyys)        [36](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1hmsyys)

[Data Analytics, Concepts and Tasks](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.41mghml)        [39](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.41mghml)

[Break Complex Problems into Simpler Ones: Part 1](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2grqrue)        [39](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2grqrue)

[Break Complex Problems into Simpler Ones: Part 2](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.vx1227)        [40](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.vx1227)

[**Week 4**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3fwokq0)[**41**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3fwokq0)

[Visualization 101](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1v1yuxt)        [41](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1v1yuxt)

[What Is Info Vis and Why It Is Important](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4f1mdlm)        [41](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4f1mdlm)

[Human Perception](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2u6wntf)        [45](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2u6wntf)

[Gestalt Psychology](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.19c6y18)        [48](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.19c6y18)

[Chart Basics](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3tbugp1)        [50](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3tbugp1)

[Colors](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.28h4qwu)        [53](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.28h4qwu)

[Fixing Common Visualization Issues](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.nmf14n)        [56](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.nmf14n)

[Fixing Bar Charts, Line Charts, Tables, and More](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.37m2jsg)        [56](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.37m2jsg)

[Applying What You've Learned](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1mrcu09)        [62](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1mrcu09)

[Crown Jewel, Self-contained Figures, More Tips](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.46r0co2)        [64](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.46r0co2)

[**Week 5**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2lwamvv)[**67**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2lwamvv)

[Data Visualization for Web (D3)](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.111kx3o)        [67](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.111kx3o)

[Why Learn D3?](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3l18frh)        [67](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3l18frh)

[Prerequisites: JavaScript and SVG](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.206ipza)        [68](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.206ipza)

[D3 Overview](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4k668n3)        [73](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4k668n3)

[Enter-Update-Exit](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2zbgiuw)        [75](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2zbgiuw)

[Attributes, Styles, Classes and Text](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1egqt2p)        [77](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1egqt2p)

[Scales and Axes](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3ygebqi)        [79](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3ygebqi)

[Dynamic Data and Interaction](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2dlolyb)        [81](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2dlolyb)

[**Week 6**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.sqyw64)[**82**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.sqyw64)

[Scalable Computing: Hadoop](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3cqmetx)        [82](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3cqmetx)

[Big Data Is Common. How to Store Them?](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1rvwp1q)        [82](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1rvwp1q)

[Why Hadoop?](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4bvk7pj)        [83](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4bvk7pj)

[MapReduce: Overview and Example](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2r0uhxc)        [84](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2r0uhxc)

[Example MapReduce Program](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1664s55)        [85](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1664s55)

[When and How to Try Hadoop?](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3q5sasy)        [87](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3q5sasy)

[Scalable Computing: Pig](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.25b2l0r)        [88](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.25b2l0r)

[Why Pig? How to Use It?](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.kgcv8k)        [88](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.kgcv8k)

[Example Pig Program](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.34g0dwd)        [89](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.34g0dwd)

[Scalable Computing: Hive](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1jlao46)        [90](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1jlao46)

[Overview, and vs Pig](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.43ky6rz)        [90](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.43ky6rz)

[**Week 7**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2iq8gzs)[**92**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2iq8gzs)

[Scalable Computing: Spark](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.xvir7l)        [92](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.xvir7l)

[Overview](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3hv69ve)        [92](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3hv69ve)

[Example Spark Programs](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1x0gk37)        [94](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1x0gk37)

[Spark SQL and other Spark Libraries](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4h042r0)        [96](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4h042r0)

[Scalable Computing: HBase](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2w5ecyt)        [97](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2w5ecyt)

[Overview](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1baon6m)        [98](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1baon6m)

[How HBase Scales Up Storage](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3vac5uf)        [99](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3vac5uf)

[How to use HBase](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2afmg28)        [100](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2afmg28)

[To Learn more about HBase](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.pkwqa1)        [101](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.pkwqa1)

[**Week 8**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.39kk8xu)[**102**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.39kk8xu)

[Classification](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1opuj5n)        [102](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1opuj5n)

[Overview](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.48pi1tg)        [103](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.48pi1tg)

[Overfitting and Cross Validation](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2nusc19)        [105](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2nusc19)

[K-NN](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1302m92)        [107](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1302m92)

[Decision Tree](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3mzq4wv)        [109](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3mzq4wv)

[Visualizing Classification](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2250f4o)        [113](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2250f4o)

[ROC, AUC, Confusion Matrix](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.haapch)        [113](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.haapch)

[**Week 9**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.319y80a)[**115**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.319y80a)

[Introducing Clustering](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1gf8i83)        [115](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1gf8i83)

[K-means, Hierarchical Clustering, DBSCAN](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.40ew0vw)        [116](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.40ew0vw)

[Visualizing Clusters](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2fk6b3p)        [120](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2fk6b3p)

[**Weeks 10 - 12**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.upglbi)[**121**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.upglbi)

[Graph Analytics](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3ep43zb)        [121](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3ep43zb)

[How to Represent and Store Graphs](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1tuee74)        [121](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1tuee74)

[Graph Power Laws](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4du1wux)        [124](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4du1wux)

[Centralities: Degree, Betweenness, Clustering Coefficient](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.184mhaj)        [128](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.184mhaj)

[PageRank and Personalized PageRank](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3s49zyc)        [132](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3s49zyc)

[Interactive Graph Exploration](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.279ka65)        [137](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.279ka65)

[Ensemble Method](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.meukdy)        [140](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.meukdy)

[Bagging and Random Forests](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.36ei31r)        [141](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.36ei31r)

[Scaling up Algorithms with Virtual Memory](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1ljsd9k)        [143](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1ljsd9k)

[Overview](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.45jfvxd)        [143](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.45jfvxd)

[**Week 13**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2koq656)[**147**](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.2koq656)

[Text Analytics](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.zu0gcz)        [147](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.zu0gcz)

[Basics: Preprocessing, Representation, Word Importance](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3jtnz0s)        [147](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.3jtnz0s)

[Latent Semantic Indexing (Singular Value Decomposition)](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1yyy98l)        [150](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.1yyy98l)

[SVD: Dimensionality Reduction, and Other Uses](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4iylrwe)        [154](https://docs.google.com/document/d/e/2PACX-1vShxYnHeVSCTLxWLfyT-huaGhaD9ItsiW1vJgYS87EIM2rI6NgEBtUGpJWAN50v1p26So-UFHmPf6_c/pub#h.4iylrwe)

Week 1

Course Introduction

Hi I'm Polo

Hi, everyone. Welcome to Data & Visual Analytics. My name is Polo Chau, I'm an Associate Director of MS Analytics Program and also an Assistant Professor in the College of Computing in the School of Computational Science and Engineering.

So today, we're going to cover some of the introductory information of this class. So we'll start with the very first question that our student would have, is how to address me. So there are two ways, one is the grammatically correct way, and the other is the grammatically incorrect way, but very popular way. So the correct way would be Professor Chau, or Dr. Chau. And the incorrect way, but very popular way, is Professor Polo and I'm Dr. Polo. So it's actually up to you, however you want to address me, that's fine. And if you want to know a lot more about my professional life, you can just Google Polo Chau, and you can see everything from my publication to my research group, so everything is open book.

So today what I would like to give you is a little bit idea about what I do for research, and how is it related to this class so that you can understand why you might want to listen to me. So this is a snapshot of some of the students who are working with me right now. So PhD student at the top, masters student on the left, and undergraduate student on the right. And you may notice that I like to name things after myself, so I named my own group the Polo Club of data science. You will see a lot of systems that I have named that include Polo in it.

So we work with really large datasets. So some of the data that you might be familiar with would be, say, the Internet, which is a 50 billion web page dataset. So the nodes here in this graph would be, say, web page, and the length will be web links or a Facebook graph, 1.2 billion users. The user in Facebook will be the nodes in the graph. And the edges will be friendship.

There are actually a lot more datasets like this such as on Twitter, who-follows-whom, or Amazon, who-buys-what? So the nodes or the items here in this graph would be the customer, and also the product. So you connect the customer to a product, if the customer has purchased the product. Cell phone network, who-calls-whom, or in biology, protein-protein interactions.

So some of the sample dataset that we analyze are shown in this table. So the largest one is the second row, which is a graph from Symantec. So it's a scrub over 30 billion edges, or 37 billion machine-file relationships. So in this graph, the nodes of either machine or files. So if a machine has reported a file, then there's an edge connecting them. So whenever I show this graph, I also like to make a small disclaimer, which is that, we not only want to work with large datasets, but we also work with small datasets. It's just very important that when you want to develop methods to work with a large dataset, you need to test it first on a smaller dataset. So, actually, I am stealing the quote from my colleague John Stasco who said, "Small data also needs love." So we work with both small dataset and also large dataset.

And having all this large amount of datasets, you must say, great, we can do a lot of exciting things. You can cure diseases by analyzing this data, you can look at the shopping trends and so on. But, I would also argue that there is a challenge of analyzing this large amount of data. I will just summarize a lot of this challenge into a single number, number 7. So number 7, actually there is a first part of what I was going to say. So there is also a second part, which is plus or minus 2. So this is a very famous Miller's law which is a cognitive psychologist. It says that the number of items an average human can hold in their working memory is actually really, really small. 7 plus or minus 2. And you know that these days, with the help of computers, all the reliance on computers, it’s actually much smaller, probably 5 plus or minus 2, if you're lucky. Right?

So what it really means is that, even though we've access to this large amount of data, a really large billion scale graph data, what we really want to help people do is to condense all of them, or distill all of that into a small number of really relevant important things that people can hold in their working memory, or things that are really important relevant to them.

So that brings us to a really popular slide, you probably have seen this many, many times in a lot of talks, which is to transform data into insights. So everyone can say about it, right? So here, I'm also saying transforming data into insights. But how do we exactly do that?

So our approach for analyzing all this amount of data or doing the distillation is to take advantage of techniques from two areas. One is data mining, and the other is human computer interaction, HCI, for short. So in data mining, its focus is on automated techniques. So as examples, we include summarization, clustering, classification, and so on. And because they're automated, often they're a lot more scalable. They can easily scale to million, billions of items. And on the other hand, HCI, Human-Computer Interaction, what the focus is to help people understand data in a very intuitive way. So, their focus will be on interaction techniques, visualization techniques and so on, and they will focus more on how to help people drive the analytics, or analysis, using iterative user-centered approaches. And because there are more manual in nature, so they would be slightly less scalable, maybe scatter thousands of items also. So the interesting thing is that both areas actually have long been developing methods to help people to analyze data. But traditionally there's less communication. So our research is to how to combine these two areas, take the best of both worlds, so they can take the scalability, the automation, and also the flexibility from Human-Computer Interaction to help people make sense of data.

So most relevant to this class is that, our approach of combining data mining and HCI is actually what this class is all about, combining computation and also the human aspect, human intuition. So how do we leverage humans’ perception in basing perception capabilities to analyze data? So computation plus human interaction.

So specifically, how do we actually do that? Right. So you have computation, very automatic, very fast, and then the interactive visualization, user-driven, very flexible. But how do we actually combine them? So here's a quick example.

So imagine you are someone working at a telecommunication company. And your boss will say, ah, so we have collected a lot of data, actually billion-scale data about customers. So each node in this graph will be say a customer, and an edge connecting two customers will say one person has called the other one, so there's an edge connecting them. And having all this data now, we want to use this data to increase the revenue of the company. So, how do we do that?

So this is actually a very common problem or issue today, is that a lot of companies have access, or they have already collected a lot of data, but then they may not necessarily know how to best use them. So in your case, you say, okay, I have this access to data, then what do I do about it? Right. Maybe you want to find something new, something novel, something I did not know about.

So, we're using traditional techniques in Human-Computer Interaction, or in visualization. Then you can first try to visualize this data. On your right, you see this very beautiful figure here. You see all the nodes, the high degree nodes, and that's like bright stars, for example. But that's probably all that you can do. So you have very pretty picture, actually we have a name for it, we call it the beautiful hair ball, as it is very pretty to look at, but not so much you can do about it. So we identify the bright spot and that's pretty much it. So in this case, you might then use techniques from the computation side, let's say from data mining or machine learning.

Particularly, you can use a class of methods we call lie detection, to pick out things that might be a little abnormal, things that are out of the ordinary, shown in red here. And maybe they can even do better. They would help you rank them. They would say, ah, so these are the five most interesting nodes, and [inaudible], for example, that you may want to look at. But often this is where they stop. So meaning they will say hey, look at these five and, trust me. So these are exactly the things we want to look at.

But in practice, we know that's not sufficient, because we want to know why. Why do you say this is the most important thing that I should spend my time on? So this is the time where the interactive visualization can be really helpful.

Example using techniques as simple as just looking at the connection between the first four items, you may see that they actually form a clique. So, a clique is a fully connected graph. So, in this case there are four nodes. So, four fully connected nodes. So, in this data, so since each node is a person and an edge will be a phone call, so, that means that these four people have called each other, right? So imagine instead of 4, now you have 20. 20 of them fully connected, 20 in a clique. Well, that may mean that probably it's the 20 people in a big family who call each other, or maybe they're up to no good. So maybe bad guys collaborating, or kind of conspiring and so on, right? So just looking at the connection you can also see so the first four may actually be interesting or suspicious. So similarly, if you provide the right interaction technique for a user to, say, expand the neighborhood of the last one, the fifth one, then you may see this is the center of a star. So, in this particular data that means this could be a telemarketer who makes a lot of phone calls to people, but those individuals do not know each other. So which is why you have a star network.

So that was a possibly over simplified example of why we need combining from data mining and interactive visualization. So, overall, what we want to do is to how to best combine these two areas so you can develop techniques and methods that are not only scalable, but also interactive and usable.

So, a key here is actually usability. So, we can have tools that are really powerful, very interactive, very scalable, but if no one knows how to use it, well, then people can't really use it. So, that's really important. So, scalable, interactive, and usable.

So, not only I say that is important, actually, Einstein also said it is important. So often this quote that you see here is attributed to him, but in reality it may or may not be actually what he said. But nonetheless, so what does it say? So it says: "Computers are incredibly fast, accurate, and stupid. Human beings are incredibly slow, inaccurate, and brilliant. Together they are powerful beyond imagination." So they key here is really how to take the best of both worlds. Take the speed of computation and then the intuition, the creativity of human beings, and then combine them. Merge it in the best way possible.

So in the Polo Club of Data Science, we have a number of projects that go in this vein, such as the Apolo System, which combines machine learning and visualization to help people explore large network. So this is not a typo, by the way. That's only one L because I put my name in it.

And then another follow-up work, we call Carina, is how to do the kind of scalable graph exploration not a test setup [?] but on any modern-day browser, such as on your mobile phone. Of course you can also run on a desktop.

So, in the same vein, we also developed a system called VISAGE, which is to help people find graph patterns in a large graph. So it won the SIGMOD Best Demo award honorable mention. So, the goal here is how to help people find interesting patterns without writing a lot of complex code. For example, in a movie dataset, you may want to say I want to find two directors who co-directed two movies, and those movies cast the same actor. So conceptually it's a pretty simple query. But if you were to write out all the code, it's actually pretty long. So it's what you see on the lower right. So VISAGE is a tool to help people more visually or more interactively find these kinds of patterns without writing that complex code.

So going back to this query here. You may wonder do we actually have such a pairing? Actually, yes. So if you're a movie fan, you will know that there's the Coen brothers and they like to cast George Clooney in a lot of movies. So there you go, there's one such pairing.

And more recently, we also apply similar techniques combining human intuition and also the power of computers to help people explore and understand deep learning models. Some of you who are catching up on the news know that deep learning is where all the buzz is. But a shortcoming of deep learning is that it's often viewed as a block box. So how do we help people open up this black box? So ActiVis is a work that we do, a collaboration that we do with Facebook that is now deployed on Facebook's machine learning platform.

So often all of these tools that we develop are in the context of my application area, which is cybersecurity. So you will know that, for a lot of faculty members, actually they will have at least one or more application areas. This is where they apply often the research techniques that they developed to solve real world problems. So in my area, on my main application area is cybersecurity. So we have done work such as working with Symantec to develop very scalable billion scale, in this case, system that can flag malware. So Polonium and AESOP are two such systems. They published in SDM 2011, data mining conference, and also the KDD conference and other data mining conferences. So both of these tools are now deployed and patented. So if you use any of the Symantec tools, then you are using part of Polonium and AESOP.

And another example could be say finding fraud on eBay, so as in people the bad guy get your money, but then they don't deliver item, so that's a serious problem as you imagine on auction.

And also, we extend our technique to detecting fake reviews, such as on Yelp, and this work is called Marco, Marco Polo as in. So it won the best paper award at SDM 2014.

So what we looked at is, what I do in my research group and how they are related to what you're going to learn in this class. You will notice that there are a lot of common themes in the computation side and also individualization side. That's also the reason that I'm very excited about teaching this class, because it's very closely related to what I do. So, I hope you will join me and to explore and learn about these topics.

Why Data and Visual Analytics?

So today, we're going to look at why we want to do data and visual analytics at all. And, I think the best way to answer is to first understand what it is.

So this is actually a little bit of a trick question because there is no formal definition. And my definition for it would be that it's an interdisciplinary science of combining computation techniques and interactive visualization to help transform data, to help make important decisions or making discoveries. So the key here is actually the end goal which is to help people make discovery and make decisions. So the techniques here is computation and interactive visualization.

And as you may expect, because of these two combinations, there's actually quite a few of things that we want to consider. So we call the ingredients of it data and visual analytics. For example, if you have data that are pretty big, then you need to worry about storage. For example, you may not be able to store on a single machine, then you need to use a cluster. In that case, I need to worry about system design. Or if an algorithm that seems to work really well on a single machine, but now because you have a computer cluster, it will scale the algorithm. So, how do you exactly do that?

And similarly, visualizing smaller dataset is probably pretty easy but now because the data also spans on multiple machines. So, now to figure out how you will scale that visualization technique to work on a large cluster as well.

And arguably, all of these ingredients were there, more or less, when the data is smaller, or not in the big data era that we are in right now. But now, because we're in the big data era, a lot of things become a lot more complex.

And actually, it's not going to stop any time soon. So, there will only be more and more data that you receive every day.

But the good news about that is there's actually a lot of demand for people who know these kinds of skills, right? Because a lot of businesses are actually dependent on access to this large amount of data. A lot of familiar examples you might see every day, such as the search engine, like Google or Yahoo or Bing, and so on. And the reason that they can rank all these webpages and predict what you're going to type in exactly because they have access to this large amount of data. And similarly, for advertisements that you see, let's say, on social networks and so on. The reason that they can look at, or they predict what you may want to look at is because they know about your friends, about your habits and so on. And similarly, to recommend you product or movie to see, let's say, on Netflix or Amazon, the reason they could do that is because they know what people's common tastes are. So actually, there are many, many more examples like that, in health, in finance, and so on.

And the good news about knowing all these skills that can analyze all these data is that there are many jobs out there. Actually, most companies out there are still having a huge demand for what are called data scientists. And, of course, many names for it, but data scientist is often the very common name that companies would post in their job posts. And the key for data scientists or to a successful data scientist, the skills that he or she should possess is actually a breadth of combination of skills. So, breadth is the key here. So meeting someone who knows how to communicate a broad set of skills work in a large team, and that's the key. So breadth of knowledge is important, and I hope that this course will help you learn a lot of these important skills.

So what we've looked at so far is a brief introduction to data and visual analytics: what it is, why is it important, and what we hope you will learn in this course

Course Goals & Expectations

Today, we're going to look at the course goals and expectations.

The course goal of this class is for you to learn visual and computation techniques. And importantly, how to use them in a complimentary way. So that means how do you take advantage of the visual aspect and also the computation, the scalability aspect. And also we want you to learn a breath of knowledge. So we're trying to do a lot of things. And more importantly, how you will use them all in practical ways. So solving real problems and using real data.

And this course schedule is roughly based on what I call the Analytics Building Blocks. So I have a lot more information about this shortly. And this course schedule, you can think of it as divided into multiple parts. The first part in green is data collection, data cleaning, integration. And then the blue part, analytics and visualization. And then finally, presentation and dissemination. So this is a pretty rough division. So that's not meant to be very rigid.

And also, I'd like to mention that these are what I call the building blocks. So they are not rigid steps. It doesn't mean that whenever you have a problem, a dataset you want to analyze, you must go through data collection, must go through data cleaning or integration. Actually, some of these building blocks we can skip. So I don't want to call them steps, because steps usually cannot really skip or go back. But here, I'm building blocks. So you can skip some. You can even go back. So you can think of these building blocks as two-way streets. For example, you can imagine that after you collected data, looking at a data type, then you can say, hah, with this type of data, let's say, time series or this is a graph data, that I may want to design visualization, so one of the blue box there, in different ways. So similarly, having collected the data, you may start to think about, oh, I will want to use a particular graph algorithm, or an algorithm with time series. So that's another blue box, analysis. Or similarly, you can go way back, for example, from visualization, starting with our blue block,s and then after looking at the data, you may seem the data is actually pretty dirty. Maybe I need to do some data cleaning. So that mean go back to the green part. And similarly, visualization may also inform algorithm design. For example, you may look at the result and then say, this doesn't really make sense, the result. So maybe I am using a wrong or maybe not a very good algorithm. So building blocks, but not rigid steps.

So a lot of students will like to ask me in the beginning of the course is, are you ready to take this class? So often, what I would say is that you'd be ready if you are willing to do a lot of programming and willing to learn a lot of new things in a very short amount of time.

So for example, in this class, we're will not only learn about how to work a large data using Hadoop and Spark, but also look at how to do visualization, let's say using JavaScript web based technology, CSS, and so on. So a lot of times, it can be overwhelming, especially if you don't know a lot of the techniques. But it's actually very, very important and very common in industry that you will need to learn a lot of things in a short amount of time. So I consider this class actually very good practice to get used to the very quick pace of learning things.

From previous classes, a lot of the projects in this class turned into publications at top tier venues. That's the KDD, a top data mining conference, or IUI, a top human computer interaction conference. And also a lot of students use their projects as portfolio pieces on the CV. So this is actually really attractive for employers to see how a student, what a student has done in a course. Not only about things that they've learned, but also actually real problems they've solved. So the project in this course allowed them to include this experience as part of their CV. And of course, because of that, they have increased job internship opportunities. And I'm really excited to often hear from students that they will send me a thank you note to say, so I learn all these skills and partly because of that I got this job. Of course, I cannot take all the full credit. But I think there's nothing more heartwarming than hearing that from the students.

So here are some example projects from previous classes. Aurigo is a project for finding interesting paths to look at or to visit tourist attractions to look at in new cities that you want to visit. And another project would be about the placement of fire station. So how do you place fire station closest to fire assistance so that you can cut down the commute time? Another project is how to find a safe path, going from point A to point B. So it's very interesting, because it's not only about looking at how do you go from those two points very quickly, but instead, how to go there in a safe way.

Here are some of the texts that I received from former students. So it will say that your class is really helpful, because it help you gain these number of skills, and then also because of that they got a job offer. So again, I cannot take full credit for it. But it's really, really helpful for me to know that they learn about these skills and, in their job, they also make use of them.

So I gave a very quick overview about the course goals and expectations. And also I used the analytics building blocks to motivate how and why we want to learn all of these. And also show you a few example projects that come from this class. So I hope we'll see a lot more example projects like this from you.

Analytics Building Blocks

Overview

So today we're going to look at Analytics Building Blocks. So you may remember that we briefly mentioned this in a previous lesson, in the course goals and expectation that this course will be based on analytics building blocks. And also very briefly we mentioned that there are multiple parts, multiple blocks in the whole pipeline.

And we also mentioned that the building blocks are not rigid steps, so you can go back to a previous building blocks this will be like two-way street, you can go between them. So in this lesson and upcoming videos, what I'm going to show you is two example projects from my research group, the Polo Club. So, see you shortly in those two videos.

Example Project 1: Apolo Graph Exploration

We're at Analytics Building Blocks. And as I mentioned previously, we're going to look at two example projects. And the first one is a tool called Apolo, a tool for exploring large graph data. And, specifically, it uses machine learning and visualization together to help with the graph exploration.

So the problem it is trying to solve is that given a large graph, you want to find out what are the interesting things that the user may want to look at. So previously, you may have seen, remember that having this kind of picture, we often call them the beautiful hairball. Actually, there are many more names for it. You can go with Death Star. You can call them Spaghetti. So the key point here is that there's not a lot of things I can do about it other than admiring its beauty.

So, what can we do about it? How do we help people explore and understand this kind of large amount of data? So, this is what Apolo was built for. So, it's to help people find more relevant things. In this particular case, more relevant nodes to look at from the large graph data.

So, this is a toy example of what Apolo can do. Imagine you are looking at a citation network. Let's say you're a student, you're trying to find related work to read about. And say your boss or adviser tell you hey, these are two interesting papers you may want to read. So one is a human computer interaction paper in orange, or in yellow, and the other one is a data mining paper, in blue. And now, off you go. So find out what other things you may want to look at. So, where would you go?

So intuitively, you may want to look at paper and say I call close to the paper that you started at, so the very left and the very right paper. So why is that? Because when you're reading paper, often these papers have a section called related work, or literature survey. And an edge here in this graph would be a citation. So that means a paper is a node, a citation is an edge. So if two papers connected, that means one paper it has cited the other, or being cited by the other one. Right, so what that means is that starting from your original paper, when you reading its related works section and you see something that's interesting, you're essentially traversing this graph going to its neighbor. And suddenly, you can repeat the same process. You look at that neighboring paper, and now again you look at a section. And then you say, this is also interesting. So that means the further and further you go, and then the relative may decrease gradually. But the closer and closer the paper is to your original one, the more relevant it is.

So what that means is we can use a technique we call guilt-by-association. So that means things that are connected in the network are probably similar. So, a vivid example would be to say, you say I'm a bad guy. So, Apolo's a bad guy. And if I know you, then well I'm sorry. Then you are also bad guy by association. So technically, the technique we use is called Belief Propagation. We won't go a lot in details into it. But that is an approach to allow you to propagate this kind of interest, or this relevance from the starting paper. In this case the two starting paper in yellow and in blue, and then go it's neighbor, and this neighbor and so on. So the closer, more the relevant, the further away, less relevant.

So now I'm going to show you a quick demo of Apolo. So specifically how Apolo helps people make sense of Sensemaking Literature. Sensemaking is a subfield in human computer interaction, and naturally interact quite a few areas, such as collaborative search, information management, or information visualization. So, in the demo that you going to see, this is actually what you going to get in the end. So, this is a graph visualization showing how these relevant areas, or how these relevant papers in this area related to what the user want to explore. So, in the video demo, we will only start with one example paper in the middle. This is actually a seminal paper, in black in the middle, called The Cost Structure of Sensemaking. So the seminal paper and I'll look at how people make sense of information and what they cost, what are the theory behind it. So in the demo you can see, look at the dataset from about 80,000 papers that we crawl from Google Scholar. So the node size here would be citation count. So the larger the node, the more citation it has received. And the edges would be citations.

So this is what we're going to get in the end. And then in the beginning of the video we only start with just one paper, as I said. This is how Apolo looks like when you first launch Apolo. There is only one paper in the middle. If you click on it, then you will see its paper information. So in this case, it's a paper called The Cost Structure of Sensemaking. At the bottom, you'll see all this information, such as a title, the author, the citation count, this pdf and so on. So this is exactly the same information that we see on Google Scholar. So in this demo, I'm going to cheat a little bit here, because I'm familiar with the paper information. So if there is only one paper in the beginning, then on the right what you will see is paper that cited it or being cited by it.

So here we are only look at the paper title, but in practice you will want to also look at the PDF as well. So just skimming the title you will see that's one paper called Information Visualization, another one is about search. So, pretty general topic. So I just drag those into the view. And in addition, what I do is I create a group for each of them. One is called search, and then the other called information visualization.

So two things happen. One is that the color is automatically assigned to each of them. And then, also the other thing is that these two papers now become, what we call the starting paper of those two groups, the blue group and the red group. And what that also means that by running the algorithm belief propagation in the background, now, I can start to propagate this relevance or interest from these two starting nodes to its neighbors, and their neighbors, and so on.

What that also means that I can actually get a ranking based on the relevance. So, I just click on the blue group and then I'll say, ah, show me the 10 most relevant papers for the blue group. And again, I can just skim the title here, and I see the first paper is indeed relevant. The second one actually also. So just looking at a paper title, I already know that it is about search as well. So I drag both of them into blue group as well, so they became additional starting paper.

So now I can go back to my original paper, and now I bring in papers that directly cited it. So by default, by showing the ten neighbors, they are ranked by citation count. So that means the size of the those nodes. And since I have already the two groups created, the blue group and the red group, I can actually change the ordering. I can say, why not we sort by how relevant they are relative to the red group. By changing the order and skimming the title, again, I can see very quickly the first few are indeed relevant to information visualization. The second and third also are. So add them also to information visualization.

So something you will see more apparent here is that not only are they added to the group, but now we also see that the color change to the remaining nodes that we haven't added. In particular, in Apolo we are using color saturation to denote how relevant a paper is for that group. So we will see that here in the vertical list of nodes, the second and the fourth nodes are actually more red, or more saturated. What this means is Apolo is trying to surface to the user the relevant information to say, hey, if you are busy, you might want to skip the first one or the third one, and we believe that the second and the fourth one are actually more relevant. So this is actually a very subtle and effective way, changing how people would interact with Apolo.

By going down the list, now I see another potentially interesting paper. It's called Personal Information Management, and it's pretty general, so I think it's probably a group by itself. So I create a group for it. And similarly, now that become a starting paper, and I can ask for recommendation, as well.

So to recap, the way that Apolo works is the user will specify exemplars like sample as you saw. And then using belief propagation in the background, it's able to find other relevant nodes.

So what did Apolo go through in terms of the building blocks? So we need to do data collection. Actually, Google Scholar doesn't provide any API. So that means we need to scrape Google Scholar, very painfully. And because of the scraping, we also need to do cleaning as well. So a lot of things are missing from the HTML page on Google Scholar. Fortunately we did not need to do integration. Everything was from Google Scholar. Our main focus there is on building the algorithm. Finding which algorithm to use. So analytics, we do quite a lot, analysis. And also visualization, the tool that you just saw, interactive visualization tool, so we spend a lot of time on that as well. And because it's a research and academic world, so we also write a paper for it, and we give talks, and now we're doing this lecture, so that's a presentation part. So we didn't get to do the dissemination, usually that means releasing Apolo to the wild, but we're actively working on it. So, we're trying to recreate Apolo as a web app, and I hope all of you will be able to use it soon.

And if you want to know a lot more about the details of Apolo, this is the paper. Published by CHI in 2011. It is on the web, so very easy to find, and you can see all the tab pages, details about what it does.

So we just look at an example project called Apolo, and also look at how it go through the different building blocks, which one has skip and which one has focus on.

Example Project 2: NetProbe Auction Fraud Detection

We're going to look at a second project called NetProbe. And we'll look at how it goes through the different analytics building block.

And NetProbe is a system that detects fraud on online auction. And this is work that was published in WWW 2007. And the problem it's trying to solve is to find the bad guys on eBay, such as people who do not deliver your item, even though you pay them. And non-delivery fraud is a very common auction fraud. It's still one of the top online crime.

The way that NetProbe works is to look at the connections among buyers and sellers on eBay. So it connect a buyer with a seller, if say the buyer has purchased something from the seller. So that means we are trying to build a graph or a network that connect these people on eBay. So in this graph, this graphic here, you will see that each node is a person and the person can be a buyer or seller and then the links would be the transactions.

So how do we find bad guys on eBay? It turns out bad guys are very smart and they will fabricate their reputation on eBay. So if you ever used eBay before, you know that there's a number next to your account which is number positive feedback you've received minus the number of negative feedback you've received. So that means, the higher the number the better. And what the bad guy does is try to make that number as high as possible by creating different kinds of account. Specifically, they create two types of account, one we call the fraudster account, in red here, and then accomplice account in yellow. The fraudster account are the ones that would trade with the victim. So that means these are the ones then would defraud the victim. And accomplice accounts are the ones that look pretty legitimate, and they will trade with honest people. But then they would also occasionally trade with the fraudsters. And these are the links between the yellow and the red nodes here. And these kinds of transactions are what I would call the heavyweight transactions. And these transaction involve expensive items like LCD TV, laptops and so on. And the reason that they want to do this heavier transaction is to make the fraudster look as good as possible. So imagine you are someone who want to buy something from say one of the red account here, the fraudster account. What would you do? You would go and look at the history of those fraudsters and then you would say, hm, has this person paid on time? And because of these heavyweight fake transactions between the accomplice and the fraudster, you will see that, actually they pay on time and that's a very good amount. And then you would trust that or you would think that the fraudster account is actually trustworthy.

So after creating these two types of accounts, then the fraudster account, the red one, will start to sell a lot of things. So I say sell in quotes, because they would never intend to deliver the item to you. So, that is how the non-delivery fraud work, which is one kind of the fraud. And you can imagine there are many, many other kinds. So, what that means is that, you want to detect action fraud, or non-delivery fraud. Here, what we wanna do is try to find this pattern between the fraudsters account and accomplice accounts. And technically we actually called it a near bipartite core. And bipartite, because there are two types of account. It's a core because it's a subgraph within the whole graph. And it's near because there's actually some missing edges. The example we look at here that there is actually one missing edge between red node and a yellow node. And the reason that this exists is because this is a side effect of how the actual fraudster would work. It just turns out that the transaction would look like this near bipartite core. So now the problem of detecting fraud becomes how do we flag this kind of pattern in the whole graph?

So what we do is we feed this matrix into the Belief Propagation algorithm. And it's able to flag actual fraud that people reported. So if you ever use eBay before, you know that there's a online website dedicated to the victims where they can report and tell other potential victims that these are the accounts that defrauded us. And the red node here are the ones that people reported. So what NetProbe is able to do is also flag other potential fraudster account or accomplice account that people didn't know yet. And this was very effective, and we run it with quite a lot of these cores. And also we publish it. And when we publish it, we got a lot of attention from the press like the New York Times, and also Wall Street Journal, and so on.

We also put a user interface to help people understand what is the patterns that have been detected? So specifically, how do these connections look like? So here we're showing one of the detected cores. So the user can mouse over the node to understand okay, why do you think that this person is flagged as a fraudster, who are the accomplices, and so on.

So what did NetProbe go through? Which building blocks did it go through? We need to do a lot of scraping from eBay, so data collection. And because of that, we also did a lot of cleaning. And all the data came from eBay, so we did not need to do data integration. But we spent a lot of time on doing the analysis. Specially on designing the detection algorithm using Belief Propagation. And you just saw we have a visualization to help people understand the detection result. And in the end is also an academic project. So we did a paper, we gave talks, and now I'm covering it in lecture. So we did not release NetProbe.

If you want to know a lot more details on NetProbe, this is the full paper that we published at WWW 2007.

So what you just saw is another example called NetProbe, and it went through different blocks in the whole pipeline of analytics building block.

Data Science Buzzwords

Hype Cycle

Today we're going to talk about Buzzwords. And the reason we want to talk about it, because, this course is about data science, and you will hear a lot about different buzzwords.

And, what are buzzwords? Buzzwords are words or phrases that become fashionable during a specific period of time. So if you do a data science project, you're likely going to use a lot of technology. And which technologies should you use? And if you ever have used some of them, you will notice that when you are trying to select which one to use, often all of them say they're the best ones. So how do you know which ones are good? And how should we really do that through our career and to figure out which one we should use?

So to give you an idea, I like to mention one specific things called the Hype Cycle, Gartner's Hype Cycle. And I would say that this is a pretty good overview about generally which buzzwords are popular, which ones may be dying down, which ones have become popular. Of course, this is debatable buzzword in itself, in that not everyone agree. So in this Hype Cycle, you will see a general trend where it says that in the beginning of some technology popularity, there will be something called a trigger. And that as something becomes more popular, it would go to the peak, we call the peak of inflated expectation. Where a lot of things are expected to be doing really great. For example, you will see some familiar names like deep learning or machine learning here. But unfortunately, not all technology would be able to deliver. So then there's a downward slope on valley where a lot of people say, actually these kinds of technology do not deliver so maybe I should not spend as much money on it. So that's a valley down there, but quite a few of the technology actually will survive in that they are actually helpful. So then there are more application, more technologies are built around them and then it will start to gain popularity and it will become integrated in our life over their course of the technology.

So hype cycle is a very good overview but I will say try not to read too much about into it. So you can look at a general trend but those specifically look at like at which point or which part of the cycle that particular technology is in, because usually not very accurate. And also many technology actually don't really go through this cycle. But instead, they may stay at the top and then they disappear, or maybe actually never even appear in this hype cycle, but then it will get adopted through our lives.

So because we need to learn a lot of things, so I actually have a model. My models learn a lot of things, but try to understand what's the reason behind a hype, what's the reason behind the popularity and also, be very cautious about it. I would say, cautiously optimistic. So you can be happy about it, you can learn about it but you should think about what's the core reasons and not just buy into it because people say it's popular or interesting.

So what we look at in this video is about hype cycle and also an overview of buzzwords. And as I mentioned in Data Science, it's important to be aware that there are many, many buzzwords out there. Some of them will survive, some of them will go away. And in our career, we want to be sure that we know when it's coming, what maybe going away, and how should think about them.

General AI vs Narrow AI

We're going to continue to talk about buzzwords today, specifically a buzzword called AI, or artificial intelligence. You've probably seen a lot about it in the news lately including self driving car, or the AlphaGo from Google, which beat the human being in the game go. But you've probably also seen some negative examples, such as a Microsoft experiment of chatbot. Where it started to say some offensive things as people feed more and more offensive comments to the chatbot. Or from the Tesla example where someone got killed, unfortunately. It's because the AI wasn't able to detect the site of a car which or the truck that looks like the sky, a brightly lit sky is all white, so someone got killed. So there's some examples of good things and some examples of bad things about AI.

And what you may not know is we're actually in the third wave of AI boom. So if you're very young, not as old as me, then you probably have never heard of something even called AI winters. Actually there are two AI winters before.

If this is the first time you hear about it, then you would say what are AI winters? AI winters a period of time in history, where the funding about AI or artificial intelligence got cut or there's a lot of less interest, because AI wasn't able to deliver.

The very first one was about 1970s, so around 1950s, that is where AI was born. And around that 20-year period after that is where a lot of excitement are generated. But unfortunately, there are actually a lot of problems in AI they are not able to solve yet because of computation power and so on. Then there was a reduced interest and eventually funding cut and it's around 1970s.

And similarly, another round happened in the early 80s and that is the second AI winter. And then after that in early 90s, that is when you start to hear a lot about machine learning. And machine learning is one way of trying to, say, sculpt the problem down instead of trying to do something which is, general AI, we called it. Meaning, having AI that can do exactly what people are able to do or even better. We say, why don't we focus on specific tasks, which is what machine learning is trying to do. Let's focus on a specific task, do not over claim and do those tasks really well.

You may have heard some of the example here. Such as in the USPS, they do optical character recognition. So that when you write something, an address on an envelope, it's simple to detect which address you are trying to ship it to. So, that's a very good early example of machine learning. And as you know, machine learning is used almost everywhere.

So, it's important to know that we're in the third wave of AI. Because now in the popular press, it generally is the case that AI become a very broad umbrella term. Even with companies that have been doing machine learning for a long time, now they want to ride the wave of AI popularity. They would say they're also learning AI. And knowing the history about AI will help you recognize that, is it really doing AI or is it actually still doing machine learning?

And a very good way to kind of disambiguate between these points like machine learning, AI, and AI-related technologies. I would say, the best is to read the White House Report. You may actually be surprised at the actual White House Report on AI. And the title is Preparing for the Future of Artificial Intelligence. It's an excellent report, because it's very rare that you would get a report from the White House that actually talk machine learning. So, it's a very good one-to-two page on machine learning, very easy to understand and very broad. And the report we've actually generated from the expert advice and comments from experts around the world, in academia, in industry.

And specifically, I want to focus on two points in the report. One is about we call the current state of AI, where it tries to debunk some of the myths about AI and also very specific about the current status of the AI.

So specifically, they say that there are two kinds of category of AI. One is narrow AI and the other is general AI. So, actually a lot of things that we see today is what we consider narrow AI. And you'll be surprised that they actually include self-driving cars or chess playing.

You may say, why is it narrow AI? So, narrow in the sense that, whatever the technology we come up, they're able to do is self-driving or to play the game Go. They're very specific to their task, which is driving their car or playing the game. It's very hard to say I just take the technology behind self-driving car and now use it to play the game Go. You can really do it. So, you really want that kind or AI does that what we called the general AI. As in something that is really general, potentially even creative, is extensible it can do different kinds of tasks.

So, unfortunately if you are an AI fan or fan of Matrix or Terminator, then that is the kind of AI that we are seeing. But unfortunately, even according to experts that come up with this or contribute to this report, that general AI will actually not be achieved for at least some decades and probably not in my lifetime either.

So, what you heard about today was a very quick summary of AI. A little bit of our history, why we want to care about history, because we're actually in the third wave of AI. And also, what's some good resources to understand more about it? And important differentiating between narrow AI and general AI. Narrow AI is often what we see today. And general AI is often time the ideal goal that we want to achieve.

Data Collection

How to Collect Data?

Today we're going to talk about data collection. You will likely need to do data collection if you don't have the data yet, so that is why it is important.

I would say there are three primary ways of collecting data. One is the happy way, which is the low effort way, where you can just download the data. So this is the best case scenario. The second, okay way, I would say, is to download the data using API, or application program interface. What that means is you might need to write a little bit of scripting or maybe a short program to download data from a website or a data source. And it's medium effort because you do need to write the program itself. And the hardest way, I would say, is to actually have to scrape or to crawl the data, and it requires a lot of effort, and oftentimes, that means you need to look at HTML source of a webpage and identify which element of the webpage you want to extract or maybe even using regular expression. And it's very brittle, because a webpage can often change.

So there are already a lot of data that you can just download. For example, the New York taxi data, StackOverflow, which is a question answering website. If you do ever programming, actually all of you would need to do, you would need to use StackOverflow a lot. Or Wikipedia, so you can get the whole dump on Wikipedia and use it. Or you can look at a lot of crime data now different cities including Atlanta, will put that crime data up on the website as well.

So these are data that you can just download. And for data that you would need to write a little bit scripting. And get the data through API that's also a fair amount. For example, Google has their own Google data API where including Google Maps Google Books and so on. So that's also great. You can relatively very easy get the data from them. Twitter also have the API, firehose. Or in other websites like Last.fm, Flickr and so on. And one thing I want to point out is that the U.S. government has a very good effort in trying to make a lot of datasets published and accessible. So a data.gov is a very good first resource.

So in the case where data is not really available through APIs or not available for download, such as data on Amazon, ESPN, in eBay in the earlier video, or Google Scholar, again, in the early video about Apolo, then for those web sources, then, you'll need to do the scraping, or need to write program, or write regular expressions. So this was what I will call the most highest level or highest effort way of downloading data. And this usually is the last resort. So try and see if you can download a data, you can get a data through API. If not, then you might need to write a scraper.

So what we look at today is a very quick overview about the three primary ways where you can download data. Just downloading them, getting them through API, or you may need to do the scraping.

How to Scrape?

We're going to continue to look at data collection. Specifically, we're going to look at scraping. You will remember that scraping is the hardest way to get data, so I like to spend a little time on this.

So how do we scrape data? We're going to look at one example on Google Play.

So suppose you want to collect some app data from Google Play, but Google Play data is not available for download or through API. Then what do you do?

So, specifically, you want to look at or could collect all the app data that you can find. On Google Play, here we're showing an app called Shazam, which will record short period of audio and it will determine or tell you what a song name is. So suppose you're looking at Shazam and you want to say, I want to start with Shazam and find out what other related or similar apps on Google Play. And then using the same mechanism, you want to get as many app information as possible. So in other words, what you want to try and do is to collect the graph or collect a network of apps, where the nodes in the network would be an app, and the edges connecting them would be similar apps [and the edges connect similar apps]. So, how would do you do that?

So for scraping, what you will need to do most of the time is to write a program or an algorithm to iteratively go through the data. In this case, you may want to start with perpendicular node, and then go to a similar app or similar nodes, and then you want to repeat, so that when you start with one, you get ten, and after getting each of the ten, you get another ten for each of them. So using this mechanism, then you can collect the whole million-node network of Google Playground. But how to exactly do that? What are the tricky bits?

So I'd like you to spend one minute to think about this problem and then we'll look at a solution together.

So, have you figured out how?

So for most of the scraping, one of the key is to actually identify how you would uniquely label or identify a particular item in the whole dataset. And if you are able to do that, identify one thing, then that means you can identify every single thing in the whole dataset. And that's often the key because we are not able to do that thing, you don't know what you have collected so far, well other things you might need to collect next. So for Google Play, what we need to do is to figure how to identify an app and how to go from one app to the next app.

So if you look at the source code of the web page for the particular app say Shazam, you would know that there's actually a string that would be uniquely identify an app. Specifically, a URL in our case. For example, for shazam, you have a very long strength, play.google.com in the beginning. And in the end, that's something called id.com.shazam.android com.shazam.android is the unique part for this app. And similarly, for Spotify you also say it's com.spotify.music. So then, the question is how do you figure out these unique string and how do you use these unique string to construct a URL that can uniquely pinpoint a particular app. So the tool that we're going to use is actual something that you already have, which is a browser's tool called developer tool. So we look at a short video to look at how we can do that, use this tool to extract the information that you want to get.

So this is how the webpage for Shazam would look like. So I am using Chrome here. By clicking a button at the top right, you will see that there is a button for more tools developer tools. So if you click on that, it will bring up a panel at the bottom. The particular tool we are going to use is called element selection tool at the top left of the panel. So if you click on it, and then mouse over any item on the webpage and then click on those item, then you would jump through to the html source that correspond to the item. I just clicked title of Shazam at the bottom you see that Shazam the work should appear on a different element in the HTML.

So what that means is you can use the same tool here instead of looking at just the Shazam title, and I can click on it again and then go to a web link. So in this case, this is actually the Web link that will bring you to another app called SoundHound. And if you look at the HTML source, you will see that at the end of the HTML, or the URL there, you will see the unique string for that app. And of course, you can do the same thing, do a sanity check, click on another app, and you'll see another string there. What that means is now we can start with Shazam and try to figure out how to very easily extract all these IDs or all these URLs from the web page and then going from one to many of the apps in the whole dataset.

So what we have look at so far, is how to do a scraping, using a simple example on Google Play. But that is a skill that you can extend to almost any kind of scraping.

Popular Scraping Libraries

We are going to continue to look at how to scrape. In the previous video, we saw how to uniquely identify items in a web page. But how do we write a program?

So fortunately, you don't really need to write everything from scratch. So there are some really good libraries out there that you can just use. A lot of them are Python libraries, such as Selenium, Beautiful Soup, or Scrapy. So they have a high level API for it to easily extract particular element from the HTML. And there's also a popular JSoup library for Java.

So these are really popular, but there is still some consideration that you need to keep in mind when you use them. They are very usable, but there are some things that may not show up in a web page until you use a different browser for example. Or some data my not show up until after the user has done some interaction.

So Selenium is great. So it's actually an automated testing tool. And it can simulate different browsers, and also it can easily simulate user interaction. For example, an operation that would click a button for you so that you don't actually need to do it. So Selenium is great for that. And so these are two key points that you want to keep in mind, because depending on what kind of data you want to collect. For example you want to collect data that show on window browser, let's say on Chrome on Windows, and if you are running your Selenium library on a Mac machine, then you just would not be able to get that. So, just keep it in mind.

So what we looked at is a very quick overview about the tools that you may want to use. So instead of ever writing everything from scratch, you can use these high-level libraries so that you don't need to do all the difficult work.

Week 2

SQLite

Simple, Effective Storage

Today, we're going to look at how to easily store data.

In the previous video, you saw how we can download data, collect data through scraping or through API, but how do we store that data?

The easiest way to store data is using CSV, or comma-separated file. And it's a very popular way to do it because everyone can do it. You have a text editor. You can open it. Enter the item or the data you want to store, put a lot of commas between them to separate the individual items. And that's it. However, being the most popular, the easiest to store doesn't mean that it's the easiest to work with. Specifically, if we receive a CSV file it may not be the easiest for you to extract the data that is stored in CSV. So why is that?

So why is CSV not easy to work with? The main reason is because a lot of consideration a lot of special cases that we actually need to consider. So in the simplest case where everything is nice and easy. So let's say, you wanna store information about a car, that you can store as year, you can store as brand and so on, everything in one row. But there also cases where actually contains strings that you want to store, that also contain a particular commas in it. Then what do you do? Well, since the comma is also a separator. Well usually you will put a quote, a double quote around the string or a single quote if you like but that doesn't solve all the problem because your string may also contain a quote, single quote, double quote. Then what will you do? Then you do other kind of escaping where double quote needs to be replaced with two double quotes. So actually there a quite a few of these cases like that which makes the CSV not very easy to parse.

So, for that reason, often when a student asks me what you would recommend to store data, or the most easiest, the simplest way. I would actually say CSV may not be the easiest. The best way we probably using an embedded database called SQLite. And I use it a lot in my research group. And the reason you might want to use SQLite over CSV is what I call a very self-contained way of storing data. You don't need to do parsing, because everything is already stored in a table in SQLite. And also we can actually have different tables. We can imagine each table will correspond to one CSV. It's very self-contained because each table also contain schema. That means you know exactly what each row is, what's data type is and so on. And SQLite is actually one of the most popular embedded databases in the world. If you ever use any computer or any phone, you're already using multiple SQLite databases. The reason is that the whole database is actually one single file. So that means you can very easily share this file with anyone. Now say you're working on a project, you say I have a dataset, maybe one, two gigabyte. You can just put the whole thing in one single file. The self contained aspect is very attractive. And also it's cross-platform. So that means the same file that you've added to Windows, it will also show exactly the same on a Mac or Linux. And, of course, because SQLite is also a database, so you can actually to query on it. You can figure out you can find particular item in the data. So not just a gigantic CSV where a lot of searching can be hard. it can create indexes, you can do full text so on. And another nice thing about SQLite is that because it is also an embedded database, so that's zero setup we call it. So that means you don't need a server. You just run a database on your computer and then you're good to go.

So what we have looked at in this video is CSV, the simplest way to store data, and what is shortcoming, and what I would recommend, which is SQLite, to help solve those problems.

SQL Refresher

We're going to continue to talk about SQLite and we'll do a quick SQL refresher, so we can get the most out of SQLite.

So as I mentioned, SQLite is pretty easy to use, it doesn't require a server. So how you launch SQLite, you go to the command prompt, and type, let's say, sqlite3, and then give the data base a name and then create a file which will be storing the data.

So, let's look at a toy dataset.

Suppose you want to store data about course information and also students who are taking the course. So what do you do? You first go into the SQLite command prompt by typing the command I mentioned, and then we go to SQLite's command prompt, where can start to create tables and then populate the tables with data. So to create a table, you will say create table, and then give the table a name, in this case student. And also the information that you want to store. In this case, we want to store two columns of information, one a student id, which is an integer, and a student name, which is text.

So we type the command, and a table will be created for you. And pictorially, it's what you see at the bottom right, two columns, id and name.

And to check that you actually created the table correctly, you can type .schema. That's a command specific to SQLite for printing out a schema of the database tables.

After you created a table, then you want to insert data, and you would use the key word insert, and say, insert into, which table, the student table. And you will specify the specific value as a student ID, 111, and the student's name, and then insert one row into the table. And then you can repeat for another student, like Johnson, and another one, Lee. And after doing that, of course, you want to do a sanity check. So you will make sure everything is inserted correctly, and when you do, it is select \* [select star]. That means star [\*] is meaning everything you want to get: select \* [select star] from student, you get everything from the table.

And let's say, now, we want to create another table to store course information, and we want to specifically know which student is taken which course. So we create another table called takes. And in this case, we have three columns. The first one is student id, the second one is course\_id, and the third is the grade that the student has received. So again, after typing this command, we can do .schema to print out the two tables that we have so far. The takes table, pictorially, you will see that at the bottom right. Three columns.

So suppose now we want to do a little bit more sophisticated, specifically want to generate a roster for the course 6242 using the two tables that we created so far. So how do we do that? We'll use the key word called join.

So conceptually that means we want to combine information from two tables. In this case, we want to find out which are the students that are in the class 6242. And by just eyeballing, you already know that's the students Smith and Johnson, and both fall down there. The two students in 6242. But how do we pull that information from the student table on the left to the takes table on the right?

And we use a join statement. And you need to tell SQLite how you're going to join the information, and the way to do the do the joining is through the id column in both tables. So specifically, we know that the student id of Smith is 111, which also appear in the takes table. So that means we want equality, or want the student id to link the two table together. So what you would do is in the join statement, that's a where cause. This is where you specify the condition of how the two tables will joined. So here you see that we want "student.id = takes.id". So that is how we are going to check for equality. And also because we want to generate a roster only for the course 6242 we want to exclude the course 4000. You will also want to specify another condition, which is the course.id = 6242. So that means you actually have two conditions here. One is for equality to do the actual joining. And another condition there is to do the filtering.

Another very common thing you may want to do is to summarize data. So let's say you wanted to compute the GPA of every student. So GPA is a summary, right. So, what you would do? You will use aggregate function. So here, you look at the text table. And because we want to compute GPA, for each student, we want to find out all the courses that a student has taken.

So here, student 222, you know that he or she has taken two courses, 6242, and 4000, and just visually, you know the GPA would be 85, just an average. So the actual statement you would use would be to include an aggregate function. In this case, we need to use AVG, which is average. So that means I want to find both the IDs, student ID and also the average grade, which is a GPA.

And that's only part of the equation. You also need to tell SQLite how you want to group the information, so that means how do you want to aggregate information, by what information in the data. In this case, we want aggregate by id, by student id, because we want to take the average among all the courses that student has taken. That means you want to specify the id as the grouping criteria, and the specific key phrase you want to use is group by. And you will specify this at the very end of the command. So the whole segment will be, I want to select id, I want to select avg(grade) from the takes table, and I want to do this aggregation, the grade aggregation, by ID.

What if you want to do some filtering, like what we did before to a generated roster. So you use group level or aggregation level filtering. Specifically, use the keyword called having. So let's say we want only the students whose average GPA is above 90. So what you would do is to add the having clause at the very end after the group by. And now we say, ah, I only want the students whose average GPA is above 90. So having average grade greater than 90.

So what we look at so far is just all the basics that you might need to know. Actually there are quite a lot of other things that you would want to learn about. So, what we recommend is you to take a database course if you haven't taken already. SQL is such a common thing that you will do in any company that you ever join, ever work with. Sometimes, they may not even say that's it's a requirement, but they actually want you to know about it. So I highly recommend taking a database course, and also, noting that what we cover so far are some of the commands in SQLite. There are also some commands that are specific to Oracle, or MySQL, or Postgres, that you want to learn about.

So what we looked at in this video is a very quick crash course about SQL. And as I mentioned, I highly recommend that you learn a lot more about it, because this is a skill that is very essential in data science.

Beware of Missing Indexes

We're going to continue to talk about SQLite. And specifically, I want to emphasize the caveat, which is that sometimes indexes might be missing. So if you're not familiar with indexes, it's a way to speed up doing queries, or finding things from SQLite databases.

So the reason that there might be missing indexes is, it's very likely that the data you work with, or SQLite database that we work with, may not be generated by you. Actually this is extremely common, let's say you join a company, the data is already there. But now your job is actually do querying, finding thing from this database. And oftentimes, the things l want to find may not already have indexes created for it.

So, what you want to do is a sanity check. You want to check if there is already the index is created so that it can very quickly do the query.

Internally, SQLite uses the index approach called B-tree, which has the data structure which speeds up the query. And a very nice property about B-tree is, any kind of adding data, finding data, or deleting data from our SQLite database, can be done in big O(log n) speed. So what that means is, if say you have one million items, that's one million student id. And you want to find a particular id from this table. So without an index you would do likely in sequential scans, I mean you look at roughly half a million of all the one million student id, in order to find out if the student id exists or not. But over the B-tree data structure, the speed is cut down to big O(log n). So big O(log n) is roughly six times, plus or minus, of accesses that you might need to do to check if the particular ID exists. So, that's a huge difference, all right, down from half a million, now to only six.

And it's very important to find out if the index is created. If it's not, then you are doing a very slow way, which is half a million scan, but if you have the index you will speed it up to only six. So, that is a very, very huge difference.

And if you would find that an index is missing, then what do you do?

In SQLite, it's very easy to do. You use a create index statement. So let's say here, we want to create an index for the particular column student\_id, where we want to do these random query, random access and random finding on. Then we say create index on the student\_id column. And actually, the index doesn't need to be created only on one column. It can be on a combination of columns. So that means that depending on what your query want to find, you want to create different indexes to speed out your query.

So what we'll look at in this video is the importance of doing indexes, and also why you may need to do it, because oftentimes, the data you work with are actually not one that you created yourself. Instead, it's created by someone. So it's important to know, to find out if there's something missing.

Data Cleaning

How Dirty is Real Data?

We're going to look at data cleaning today.

You're likely to do data cleaning if you download data from sources that may not be very reliable, or you do a scraping, or maybe even getting through API doesn't necessarily guarantee that all the data is clean and nicely formatted.

To understand why this is important, maybe we should first look at what are the kind of data dirtiness that you may encounter?

So, we'll use a simple example, let me just say we want to store the dates in a dataset. So, there actually turn out to be many, many ways you can store a date or represent a date. It can be January 19 or we're using the short form of January, or depending on whether you're American or British, you might rearrange the month and date differently. So just here, we all ready have five formats. Actually, there are way more formats than I can include here.

So to get you to understand the extent of the problem, I'd like you to spend 60 seconds to come up with five different kinds of data dirtiness that you can think of.

So, let's see how many of these have you got. Actually, this is only a very, very tiny subset of the kind of data dirtiness that you may encounter. So, some of them you may expect would be, say, a spelling error, maybe missing data, someone forgot to enter something, or maybe it's the same quantity but somehow you use different units. Or there can be leading zeros, it can be trailing spaces. Data may be in the wrong type, we have sometimes the uppercase, lowercase, and so on.

And you may ask, so why do we have all of these problems? Actually, we human beings are to blame, because often the data is actually generated by us, ourselves. And, you know, human beings are notorious for not following the rules, which also means that we might do a lot of different errors.

So in this video, we looked at some examples of data dirtiness ,and specifically, one of the core reasons that we have to encounter all these or do all this data cleaning is because, actually, data is often generated by human beings, and we are not very good at following rules, and that's why we end up having to do a lot of data cleaning.

Importance of Data Cleaning

We're going to continue to look at data cleaning. And specifically, I want to point out why it's important to clean data.

There are actually surveys on how much time that data scientists spend on data preparation. And maybe not very surprisingly, a lot. A lot of people, data scientists would say that they spend over 80% of their time on preparing data, from cleaning, putting things into right format, and so on. And the reason that is so important to data cleaning is, a lot of other building blocks that you will need to go through, including visualization, or coming up with the algorithm, actually dependent on the quality of the data. For example, you want to do classification, and then you need to better have the label to be properly formatted in the data. And there's also a famous saying for that, garbage in, garbage out. So if your data's not clean, then any result that you derive from the data would likely be inaccurate.

So we actually have a name for doing this kind of job. We call it data janitor. And also that actually all of us are data janitors, in that if you want to analyze data, it's incredibly important for us to be able to first clean the data so that any results that we come up with will be correct.

So how to reduce the amount of data cleaning that we may need to do? So one very good place to start is to write clean code. So specifically, when we write code, we want to be careful about how we format our code. So that, if you are going to analyze the code or analyze the data we're going to generate using the code, will be properly formatted. Actually, the importance of formatting the code, or using the right amount of whitespaces versus using tabs, it actually came in popular culture. In Silicon Valley the show, some of you may have seen it, and there one of the characters would say that there's no way I'm going to be with someone that uses space over tab. So this is actually a very funny phrase that kind of look at the two kinds of people, one really wants to use spaces, the other wants to use tabs. And this is actually a very important problem. So if you're usually trying to do a coding in a company, or in a team, often there's already some coding practices in place which to unify what every team member will need to do. And if you don't do it and you try to check in code in the accompanying repository they will see that there are potentially a lot of conflicts. Or maybe there are cases that you will say, you actually are making a lot of changes to existing repository. But why is that? It turn out it's just because of different uses of space. For example, a tab could be two spaces or a tab could be three spaces. So using different definition of spaces, you will end up having a lot of these differences even though the program looks exactly the same.

So try to write clean code so they would avoid a lot of headache, a lot of mystery when you try to tracking code or generate data.

So what we look at in this video is the importance of data cleaning and also how you yourself can help reduce the data cleaning that you yourself may need to do or other people may need to do.

Data Cleaners: Open Refine & Wrangler

We're going to continue with Data Cleaning.

Good news about data cleaning is you may not need to do a lot of it yourself. And you can use tools to help yourself. One tool is called Data Wrangler and the other is called Open Refine. What we're gonna do next is to watch a video for Data Wrangler and Open Refine. And as we watch the video, you'll write down the data dirtiness that you see in the video that's shown there. And then also how the tools help address those issues. Data Wrangler was originally a project developed at Stanford, and Open Refine was originally called Google Refine, and Google open sourced it, so now it's called Open Refine. So you watch these two videos and after that, we collectively summarize the similarities and differences of these two tools.

This is a screenshot of Data Wrangler, and it's started by Professor Jeff Heer, and his student, or former student, Sean Kandel. And they now form a company called Trivecta, where DataWrangler is the main tool of the company. And here's a screenshot of Open Refine. This is a free tool that you can just download right away and use it. It's pretty powerful. So in the video, you're going to look at how these two tools are going to help simplify your life.

[OpenRefine: [https://youtu.be/B70J\_H\_zAWM](https://www.google.com/url?q=https://youtu.be/B70J_H_zAWM&sa=D&source=editors&ust=1738598889162309&usg=AOvVaw2zME1EHUI9yaui7jE3PI6G)]  
[Wrangler: [https://vimeo.com/19185801](https://www.google.com/url?q=https://vimeo.com/19185801&sa=D&source=editors&ust=1738598889162479&usg=AOvVaw3G30_K0C4AW2nGQ4WN7aB-)]

Let's see how many of the features that you have identified after watching the video. So here we're using the encoding O for Open Refine and W for Data Wrangler.

In the video, you've seen that Open Refine can deal with data clustering. And it's a very nice feature in that it can that it can highlight outliers. And for Wrangler, it also has some unique nice feature like doing previews. For example, you can do some operation. And without actually doing it, it will show you how the result might look like. And also very nicely can suggest what you may want to do, suggesting operations. Both of them provide the capability to capture the history of an operation, so they can undo. And also, other capabilities, such as offline processing, which is really helpful. So that means for larger datasets, instead of doing everything in the tool itself, you can export those as script or simple program and then you can run it on larger datasets.

In those two videos, what you saw is just a small subset of the tools' features. I recommend you to try the two tools out, so you can also learn about the other features that are not covered in those videos.

In this video, we looked at two tools, Open Refine and Wrangler, which can potentially have save you a lot of time. And we also watched the videos of these tools' similarities and their differences. So now go try them out.

Class Project Overview

Forming Great Teams

Today we're going to talk about the class project and the very first thing is about forming teams.

So you're welcome to form teams now. You don't need to wait, and the team formation deadline is shown on the course homepage. So each team should be between four to six people, and I do not dictate who forms teams with whom. So you're welcome to look for teammates right now.

So there's some challenges in forming teams. So often students ask me can we form a seven person team? So I highly recommend not to and only with my permission. So main reason is because forming larger teams are harder to run and also they are at higher costs. Higher costs as in coordination, motivation and also intellectual costs.

So if you want to run a very successful group so here's a list of tips from Professor Randy Pausch from Carnegie Mellon. So something as simple as introducing yourself to each other properly. So meeting people properly knowing each other's name, that's vital to the success of the group. And also, to make sure each other would work well together, find things in common because you're going to work with each other for a long time. And also make the meeting condition good which is why we often see lunch meeting, dinner meeting, because everyone got to eat, and just making everyone happy, very important for meetings. And let everyone talk, so everyone has great ideas, so let others share their ideas. And also, for that reason, you want to check your egos at the door. And often you might say, well, that may not be a very good idea. But there might be still something good that you want to tease out from someone's proposal. So praise each other, so that will just make the condition a lot better. Very important in working in team is that you want to put things in writing. People forget, and often not intentionally and we have a larger group that is more likely than not that someone might not be on the same page. So putting everything in writing will make sure everyone will be on the same page and also everyone will remember. When working in teams, you are going to hear different ideas. So be open to that and that same time, provide honesty feedback. Try to avoid conflict at all cost. And you can think about phrasing alternatives. For example, ask a question instead of saying, hey, I don't agree with you. So putting it as a question allows people to clarify and help avoid conflicts.

So today, what we've talked about is how to form great teams and what are list of tips that you may want to use so that you will work successfully in a large group.

Core Project Requirements

We're going to continue to talk about class project. And specifically, on the core project requirements.

So the project is actually very flexible. So there are only three main things to satisfy and you can almost do anything you want.

So the very first thing is that your project need to have a large, real dataset. You can have multiple, but at least one large, real dataset. And often students ask me, what do you consider as a large dataset? So usually my answer will be, well, when you analyzing the dataset, you're more likely to suffer a little bit. Suffer as in you don't want to have something that is really, really, easy. And, for example, you come up with something, and it just run in under, say, one second. And, that's likely going to be too small.

The second requirement is that it should involve some non-trivial analysis, algorithms, or computation. Non-trivial, as in it require quite a bit of thinking. Or something that's too simple, for example, would be compute intermediate, compute average, then that's very trivial.

The third thing is that it should have an interactive user interface that interact with the algorithm. So meaning the visualization of the user interface should not be static, should not be something that's used purely as output. But the user should be able to interact with it, to do something about it, and that it will affect the computation so that you can get some updated result.

For the grading and schedule of the project, there are three main deliverables: project proposal, progress report, and final report. For the proposal, there is a document and a brief presentation. And for the progress report, so it's mainly as a check point meaning that we want to make sure everything's running smoothly so that you need to rush at the end. And the final report we include a project writeup, documentation, and also a presentation.

So what we talked about today is the three core requirements for the project. As long as your team's project idea fits those three requirements, you can do almost anything you like.

Project Idea Checklist: Heilmeier Questions

We continue to talk about class project, specifically, how to come up with great ideas.

My recommendation is that you choose something that you're excited about. For example, for me it was the NetProbe project for eBay fraud detection. So I say choose something that's interesting. Interesting could be something about computationally, could be about design, could be about visualization. Also, choose something that's impactful. For example, it could be something that helps people save lives, fight crimes, and so on.

So work on something that's interestingly challenging, so that you will learn more. A very good suggestion is that you look at the dataset on the course homepage. And usually, looking at the data will give you ideas about what are the things that are possible.

To help you think through the project ideas, every team has to answer a list of questions called the Heilmeier Questions. It's a list of questions came up by George Heilmeier, who was a former director of DARPA. And DARPA, you can think of it as a research arm of the Department of Defense.

It's a list of questions that every project proposer needs to answer. And you can think of it as a pre-flight checklist for successful projects. And for your project, you also need to answer these. And there are nine of them. And at a first glance, you may think of them as, hmm, these are pretty easy questions. Why do I need to answer it? Exactly because they are easy question, you often will overlook them. So let's go through them briefly.

The very first question you want to answer is what are you trying to do? So you may think that is easy, but it's actually not. Because when you answer this using absolutely no jargon. So you can think of this as a version that you would probably tell your parents what you're doing. For example, when my parents ask me, what are you doing, Polo? And I was just saying, I'm doing computer, I'm doing IT. And this is probably as far as they would understand. It doesn't mean that you're trying to dumb something down. But instead, you want to make something that's really easy to understand. So using no jargon. So it should be something that everyone should be able to understand.

So second thing is that how is it done today? So as in, what's the limitation of the current practice? So why do you want to work on it? How is it better?

That is actually the third question. So you need to articulate very well what is new in your approach and why do you think it's going to be successful.

And the fourth thing is, who cares? So often when come up with ideas, we need to think about who are the user. So this is the answer to the question.

And, if you're successful, what difference are you going to make? So that means how are you going to affect people's lives? And how are you going to measure them?

And the sixth question is what are the risks and payoffs? So sometimes some projects are high risk. But at the same time the pay off my also be higher, so which is probably a very good trade-off.

And seventh, not all project may have this, is how much is it going to cost? For example, for a class project, sometimes it may cost you nothing other than your time. But for some projects, maybe it does. For example, maybe some cloud computing time.

And the eighth question is, how long is it going to take? So for this class project, probably easier to guess, because this is the length of the semester. But for some project, maybe it's harder to estimate. For example, it could be a year-long project.

The final question is that how do you define success? Or how do you measure them, how do you check for them? For example, for this class project, it's probably easier for you because we already set this as a midterm progress report and a final exam. For larger project, you will need to be able to come up with a better checkpoint throughout the project.

So today, we looked at how to come up with great project ideas, and also the checklist to help you think through them.

Pay Attention to Software Licenses Early On

We're going to continue to talk about class project. And today, I want to emphasize something that's really important for your class project, and also maybe important for projects that you may do in industry which is about the licensing term.

So when you're doing your project, likely you're going to use a lot of libraries or reusing existing code. And there is this one license called GPL or General Public License, that you want to pay attention to. So this is also an open source license, but open source does not necessarily mean that there's no restriction. So in this case, if you use a library that is GPL, and use it in your software or project, and later on you decide that you are going to sell this or distribute it, then there is a limitation that your entire source code needs to also be available to the end user. The reason is for GPL that is part of the terms, so it's still open source, it's still free to use but there's a restriction of the distribution. So, you having a project that use GPL library there's something to keep in mind. The main reason is because if you don't keep this in mind and only know about this at the end of the project that it maybe too late. For example, there is a horror story hear about which is that a project at the time of delivery, they say that I actually don't want to release only my source code, but they have to, because they use a GPL licensed software.

So today, I just wanted to point out a very important thing about the licensing terms, specifically the GPL license. This is something that's still open source license, but then, you would need to pay attention to the restriction, which is that you need open source all the code in your distribution.

Code Backup and Version Control

Git: Overview and Benefits

Today, we're going to talk about Backup & Version Control.

You want to back up your computer because computer dies, and often at the most inconvenient time. For example, right before a deadline or a at a job talk. So that's actually a horror story that I hear about which is someone's computer died right at their interview time. So, not very convenient.

Replacing a computer is easy, but recreating lost work and data is not. So, for that reason, I highly recommend Git, Git as in GitHub or Bitbucket.

Git is the most common version control system in software development which is why I highly recommend it.

So the benefit of using Git is that, of course, it backs up code. And second is it provides something we call version control. So that means you can have multiple versions of your work and you can revert back to an older version when needed. For example, you try some new feature and it doesn't work, then you can just go back. So, very convenient. Thirdly, you're likely will be collaborating with your teammates when working on a large project so Git allows you to do something we call branching. So that means you can have copies of the same code, creating branches or branching out and each person try different things. And when you're ready to combine, you can just very easily merge all of them into the main code base. And the fourth thing, which is often overlooked is that using Git is actually a very good way to encourage you to do a README file or doing documentation.

For example, if your using GitHub, then the README file would become the landing page. So that means that's what people will see when they arrive at your repository.

Also using GitHub, there is a benefit is that they provide a free Desktop Client which works on Mac and Windows. So this is great if you are new to Git or if you like GUI or you just don't like typing commands like me.

So for example, here is a screenshot of some of the GitHub pages, the most popular ones. Some you are familiar with, like, for example, the Twitter bootstrap, or the Facebook React framework. So, I would encourage you to take a look at these project websites. So you can have a really sophisticated, really nice homepage for your GitHub repository.

So, what we've looked at is an overview of Git and why you want to use Git because computers die. And also it provides additional benefits such as version control, collaboration, and documentation.

Warning! Keep Your Repository Private Initially

Hi, welcome back. We're going to continue to talk about version control. I want to point out something really important, and that's also the focus of this video, is that you want to keep your repository, at least initially, private, unless you're really, really sure that you want to make it public.

So you may ask why?

So the main reason is because you make it public then everything is on the Internet. So that means people Google for your repository or maybe just browsing, they can see your repository. And the horror story, here's one of them where some of put there a very sensitive information. In this case, the AWS key, so AWS Amazon web service, so that means this is a service that companies. For example, they can rent, they can buy some compute time, and do a lot of scalable way, a long running job on AWS. And for this person that accidentally put the key, which is a, you can think of it as a password for them to access the Cloud service on a GitHub public repository, and what happened? Well, it turned out that there's a lot of bad guys out there that are running these scripts or program that specifically scrap for these AWS keys, and what they do is they get these keys and then, they would use those AWS that's in a lot of computers to mine Bitcoin. And there are also even other horror stories where they says, well, they get a copy of all the code base the company has, and then hold ransom. So these are the bad things that can really, really happen.

So as a suggestion, when you first start off so I highly recommend that you keep things private. And once you are sure that all sensitive information is removed or not there, then you can make it public. And even better, if you want to, if you want to do a clean slate, then I would just do a clone of the repository, and it opens with that clone instead of what you're working on right now.

So, that's a warning about using a private repository initially. So make sure you're really comfortable with open sourcing it and don't just pick that public option as your default.

Week 3

Data Integration

Knowledge Graph

Today we'll look at data integration, an important topic for data analysis.

So what is data integration? At the high level, it means compiling data from multiple sources to form one unified dataset.

But why do we need to combine data in the first place, how common is that? To answer this question, take a moment and think about the websites, online services, or apps that you use on your phone or computer. For example, you probably use search engines like Google, Yahoo, or Bing every day. Do search engines need to integrate data?

Yes, very much so, and the main reason is that businesses derive value through data integration so that they can better engage their customers. If you use Google today, let's say to search for Atlanta, what you see on the search results page contains more than just web pages. You will also see new stories, things to do in Atlanta, and information about the City of Atlanta. All this information comes from multiple sources. For example, the new stories comes from these agencies, information about the city of Atlanta may come from Wikipedia.

Smart assistance like Siri also do integration behind the scenes. For example, the reason they can answer questions asking about stock prices, weather, facts and definitions, or events happening in the city is because they have access to many data sources that provide such information. Similarly for travel websites like Kayak, they also integrate multiple data sources. For example by having access to hotel prices, flight prices, and car rental prices. These websites can offer customers combo deals. Which may be appealing to the customers, helping to bring more business to the site.

There are actually many more examples. Social network websites like Facebook would combine data from users, businesses, and advertisers, to provide a richer environment for their users. For example on Facebook you can see your friends' posts, reviews of places you may want to go, and advertisements that may be relevant to you. On demand car ride apps like Uber and Lyft integrate data about drivers, potential riders, and traffic information to connect riders to drivers.

So how to do data integration?

Here we first look at the easier ways to integrate data. If the datasets are already in some kind of databases, then it can be relatively easy to integrate them. For example, let's say we have two toy datasets. One is a database of people and another is a database of salaries. Combining these two datasets would only require the use of a join statement, as long as each person has already been assigned an ID. For example, in our toy dataset, Smith has a ID 111, Johnson has an ID 222. But what if we don't have such IDs? You may be able to use tools like Open Refine to help you to figure out the IDs. Visit Open Refine's website and watch the third media there that is titled Reconcile and Match Data to learn about how Open Refine can help you figure out the unique IDs of movies. After watching the movie, come back and we'll continue to look at how that is done.

[[http://openrefine.org/](https://www.google.com/url?q=http://openrefine.org/&sa=D&source=editors&ust=1738598889170114&usg=AOvVaw01ZH--9L91qBJvlhs_BiEm)]  
[[https://youtu.be/5tsyz3ibYzk](https://www.google.com/url?q=https://youtu.be/5tsyz3ibYzk&sa=D&source=editors&ust=1738598889170242&usg=AOvVaw0x5KvhvWqwbn9XDkIqoCOD)]

Welcome back, from the Open Refine video, you saw that you can have multiple movies with the same similar names, such as Oceans 11, where the 2001 version was a remake of the original 1960 version. Being able to identify which is which is important in practice, imagine the danger of mixing up the medical records of two different patients who have the same name.

So IDs are important, but who creates them?

Freebase was a major effort that made this happen. It was a crowd-sourced knowledge base where people collectively labeled anything you can imagine and give them unique IDs. Now watch the Freebase intro video to learn about what Freebase is and what it can do.

[[https://www.youtube.com/watch?v=mmQl6VGvX-c](https://www.google.com/url?q=https://www.youtube.com/watch?v%3DmmQl6VGvX-c&sa=D&source=editors&ust=1738598889170639&usg=AOvVaw2VM_NUBR6t_NnDUss8zMtj)]

Welcome back. From the Freebase intro video you saw that Freebase not only assigns ID to things, it also links the things if they're related. In other words, you end up getting a graph of entities, which is also called a Knowledge Graph.

So what can you do with the Freebase Knowledge Graph? A little hint for you, Google acquired it in 2010. And now Google has something called Google Knowledge Graph. Visit its homepage and watch the two intro videos there to learn about what it can do.

[https://www.google.com/intl/bn/insidesearch/features/search/knowledge.html](https://www.google.com/url?q=https://www.google.com/intl/bn/insidesearch/features/search/knowledge.html&sa=D&source=editors&ust=1738598889170920&usg=AOvVaw2G7Abrlq3vp4Uim0ZUQ8rT)

Welcome back. Does the Google Knowledge Graph remind you of Freebase? It should, because Google's Knowledge Graph is powered in part by Freebase. And Google uses the graph to pull in information that you may care about into the search results page of your query. Remember the example you saw earlier on where we googled for Atlanta, and we will see tourist attractions in Atlanta. Such attractions are entities into Knowledge Graph that are linked to Atlanta. In other words, Google leverages the graph to enhance your search experience for potentially interesting information.

Google has opened up its Knowledge Graph for you to access via its API, for example you can use an API to find information about Taylor Swift.

The API returns that information in JSON format. You can see that Taylor Swift with has her own unique ID and her description is Singer-songwriter. There's a link to her picture and also a link to her homepage.

So having a Knowledge Graph of entities where everything is uniquely labeled is really helpful to Google. And in fact, not only Google, but to Facebook too.

And Facebook introduced a similar kind of knowledge base built on top of its user and their information. And it is called Graph Search. Now what should Graph Search intro video to learn about what it can do and how Facebook uses it to enhancers experience when using this platform.

[[https://youtu.be/W3k1USQbq80](https://www.google.com/url?q=https://youtu.be/W3k1USQbq80&sa=D&source=editors&ust=1738598889171525&usg=AOvVaw1ANib76akbVUOukwSrqXW5)]

In this video, we talked about data integration and learn that it is a very common thing to do because many companies define values and then enhance user experience by combining data from multiple sources. We also discussed the important role of entity IDs and that Freebase was a major effort to create such IDs. Freebase was acquired by Google and is now powering its Knowledge Graph that enhances search experience by pulling in relevant information for the user.

Data De-Duplication

In the previous video you learned that it's very important to have unique ID's assigned to things that we care about, which can simplify data integration. In this video, we'll look at what we will need to do if we don't have the luxury of having ID's.

In practice, many things in the world are not indexed by knowledge bases such as the Google Knowledge Graph. For example, in academia, it's very common for multiple researchers to have the same name. Or to be even more confusing, one person can have multiple names. For example, while most people call me Polo Chau, Polo is in fact not my legal first name. My legal first name is Duen Horng, which has two words, and very often citations will abbreviate them as DH or even just D. This means, just me alone, I already have at least four name variations. How do we resolve all these variations?

This is in fact a hard problem, commonly known as entity resolution, that is, to disambiguation variations of the same thing.

Why is entity resolution so difficult? Let us understand this through doing some shopping. Imagine we want to buy an iPhone, and we are considering whether to buy it from Apple, Amazon or eBay.

If we buy directly from Apple, then it's quite straightforward. You first visit Apple's official home page for the iPhone, then select the size that you want, then you choose the wireless carrier, then color, and finally storage capacity. And that's it. Apple already has an ID in the form of a serial number, where every combination of size, carrier, color, and capacity. In other words there's nothing for you or for Apple to disambiguate.

Now, let's go to, Amazon. Maybe this is a webpage dedicated to the iPhone. So, you'll need to search for the phone that you want using a search box. In the search results, instead of seeing only one phone, you see multiple phones. And also some accessories like headphones and cases. The phones may be from the same or different sellers, most of the time from Amazon, but sometimes from other third-party sellers. The phones may vary in sizes, colors and so on. If a phone seller is Amazon, usually Amazon has already associated that phone with the correct Apple serial number, and there should not be any ambiguity. But it may not be true for the phones from third-party sellers.

Now, let's go to eBay. It's even more challenging to figure out exactly which phone you might want there. eBay has numerous sellers, selling anything you can imagine What's more is that eBay is also an online auction, meaning that you can find both auction listings and fixed priced listings of the phones that you are looking for. And if your phone is popular, you may find hundreds or even thousands of such listings, each with a different price and a different listing title. This means, while eBay may get you great deals, say through there auctions, you may need to do a lot of disambiguation yourself. You could imagine that eBay would love to see all there listings of the same products nicely grouped by the same, unique serial numbers. But that is not always possible. One reason is that sellers get to choose how they set the listing titles, and they may or may not adhere to eBay's suggestion to associate their phone listings with official Apple serial numbers. Another reason is that, some products are juts so rare that no one has ever assigned them IDs.

So what do we do then? We don't need to give up. There are tools we can use.

Besides using Open Refine, there are techniques from research you may want to consider. One such tool is D-Dupe, originally developed at University of Maryland.

D-Dupe resolves ambiguity among potential duplicates by simultaneously considering those entity's attribute similarity and relational similarity. Here in this screenshot of D-Dupe, we're looking at information about researchers in a research collaboration network. Each node in this network is a researcher, and two researchers are connected by an edge if they have co-authored a paper together. In the relational context viewer at the top right, data shows a potential duplicate researcher in region two and region four. Here, we see there are two researchers, possibly referring to the same physical person, with almost identical names. The name in region two has a middle name, while the name in region four doesn't. A name that's shared by both researchers are shown in region three, and non overlapping neighbors are shown in regions one and five. This means, if we see there's a lot of nodes in region three, then the two names in region two and four, are likely the same person, because it is not very likely for many people to have worked with two persons with the same name. To determine textural similarity between entities, D-Dupe offers a number of string based similarity measures, such as string at a distance.

To summarize, D-Dupe helps user locate potential duplicates by comparing the entity's attribute values, such as their spelling and also their relations, to see how many neighbors that they share.

In more detail, D-Dupe allows the user to adjust how to balance between the two types of similarities using a weight constant alpha. The overall similarity score for a pair of entities is then, the sum of the weighted average of the attribute similarity score and the relational similarity score.

One important point to know is that there are many similarity measures you could use, and often it's not easy to know which one to use unless you try them. The rule of thumb is that, you may want to start with the more common measures and then proceed to the more complex ones if the simpler ones do not work well. For example, for numerical values, Euclidean distance is a popular first choice, since it is easier to interpret and has nice geometrical meaning.

If you want to compare two sets of items and determine the extent of overlap as a similarity measure, then you want to try Jaccard similarity, or Jaccard index, which is defined as the number of elements shared divided by the total number of elements. In the example shown there are two sets, S and T. Set S has five elements and T has six. They share three items and there's a total of eight unique items, so the Jaccard similarity is 3 over 8. A Jaccard similarity of 1 means complete overlap and a Jaccard similarity measure of 0 is no overlap at all.

If you want to compare two pieces of string like Polo Chau and Polo Chan, you might want to try string edit distance, which measures how many textual transformations you need to do to convert one string into the other. The transformations include adding a character, deleting a character or replacing a character. The more transformations you need to do, the less similar your two strings are.

As mentioned before, there are many more measures you may want to try. For example, in MATLAB alone, there are already tens of measures. So go try them out.

To learn more about entity solution, I highly recommend excellent KDD2013 tutorial by Lise [Getoor] and Ashwin [Machanavajjhala].

[[http://www.umiacs.umd.edu/~getoor/Tutorials/ER\_KDD2013.pdf](https://www.google.com/url?q=http://www.umiacs.umd.edu/~getoor/Tutorials/ER_KDD2013.pdf&sa=D&source=editors&ust=1738598889173394&usg=AOvVaw2iHMjYvfIjnoUO7sDnwARk)]

In this video, we have looked at entity resolution and discussed why it's an important, but hard practical problem to solve. And we also mentioned the role of similarity measures that's central to entity resolution, and the many measures that we may need to try to figure out which one may be the best for our task.

Data Analytics, Concepts and Tasks

Break Complex Problems into Simpler Ones: Part 1

When you work on a data analytics problem in industry, the problem is likely very complex, requiring the combination of many techniques and the use of many tools. This video, will give you an overview of the kinds of techniques you should learn about. In later videos, we'll go into more depth about some of them.

The ability to decompose a complex problem into simpler ones is one extremely important skill that all data scientists should have. In addition, a data scientist should also be able to identify parts of the problem that already have known solutions, so that they can use and apply existing tools to solve those parts. This will allow us to focus our energy on the harder parts of the problem.

The good news is that while there may be a huge number of complex problems to solve, there's only a small number of categories of analytics tasks. What we'll do next is to go through eight of them, based on the book Data Science for Business. An important point is that these tasks are not mutually exclusive, meaning some tasks can share some underlying principles or algorithmic techniques. In fact, towards the end of this course you will see that some of these principles and techniques form the foundation of many higher level tools.

Let's look at classification first. Classification is the most common analytics task. In one sentence, what classification does is to predict which class and entity it belongs to, as shown in the green text on your screen. An alternative name for class is label. So, we can also say that classification is to figure out the label of an item. You can easily come up with examples of classification problems, from classifying emails into spam or not spam, classifying the sentiment of tweets, labeling medical images that detect cancer, and so on.

A second closely related task is regression. The one sentence summary is to predict a numerical value of something that you care about, such as predicting temperatures, stock prices, credit stores, game scores, wines scores, and so on. The reason that regression is closely related to classification is that you can always convert to results of regression, which is a range of values, into a few classes by thresholding the values. For example, you can decide that any temperature below 70 is cool, and any above is warm. And you now have a binary two class classification problem. The inverse going from classification to regression is not always possible, however, and for that reason some people say regression problems are harder to solve.

The third task is similarity matching. This means you want to find other things similar to what you are interested in, as in finding similar gene sequences, similar patients, or finding likeminded people to date. The need to find similar things, in fact, underpins other category of tasks. For example, in classification, suppose we know that the label of an item x. If you are able to locate other similar items of [to] x, you can assign the label of x to those unknown but similar items.

Our fourth task category, clustering, is based on similarity matching, too. Clustering means grouping things based on their similarity. Cluster is a fancy way to say group. For most clustering algorithms the user of the algorithm would need to provide a number of groups to find. Because for the same dataset it can be correct to use different numbers of clusters depending on the user's tasks. For example, we can say there are tens more groups, or we can merge them into say two bigger groups. We have examples of clustering in topic discovery for documents, determining land cover, and use that segmentation in advertising.

In this video, we highlighted the importance of being able to break complex problems into simpler ones. So that we can focus our energy into the harder parts of the problem. Then, we went through four categories of analytics tasks, classification, regression, similarity matching, and clustering.

Break Complex Problems into Simpler Ones: Part 2

We're continuing to look at some more analytics tasks.

The fifth category of tasks has many names. Here we call it co-occurence grouping. Other names include frequent itemset mining, association rule discovery, and market-basket analysis. A typical use which also explains why it's called market-basket analysis. Is to figure out what items that people often buy together at grocery stores. For example, bread and milk are often bought together. A modern-day example is where Target figured that a girl was pregnant before her father did. And the reason was, because based on the purchase histories of many target customers, Target was able to figure out the items that pregnant women often buy around the time of pregnancy.

The sixth task category is profiling. It too has many names, such as pattern mining, and anomaly detection. All these concepts are closely related. What we want to do here is to find a typical behavior, or trends, or patterns of entities. If we can do that then we can also find outliers and anomalies. That is, to find the extraordinary, we first need to find the ordinary. Every day exampled include intrusion detection, sign-in alerts, or Moneyball, in baseball, to find extraordinary but underrated players.

The seventh task is link prediction or, more generally, recommendation. In the context of social networks like LinkedIn or Facebook, the first name, link prediction, is more straight forward to understand, because we indeed want to link two people together. However, for Amazon, Netflix, or Pandora, they are also doing link prediction, but we'll instead call that recommendation, as in recommending a movie you may want to watch, a product you may want to buy, or a song you may want to listen to. In later videos, we will see that you can use the same numerical technique, called singular value decomposition, or SVD for short, to solve both the link prediction and the recommendation problem.

The final task category is data reduction or dimensionality reduction. We need to do this if we want to visualize data in 2D or 3D. In other words, we need to shrink the data which has many features or dimensions down to only a few. The other benefits of doing dimensionality reduction include faster computation and less storage, since there is now less data to work with. And some noise in the data may also be removed, because dimensionality reduction techniques often keep only the more general patterns in the data, dropping small noises on the way.

In this video, we went through four more categories of analytics tasks. Co-occurence grouping, profiling, link prediction, and data reduction. In future videos, we will go into more depth about some of them.

Week 4

Visualization 101

What Is Info Vis and Why It Is Important

Starting with this video, we'll talk about information visualization or Info Vis for short and discuss how it helps with data analytics. First, we'll take a quick look at what it is, understand why it's important, and how it's related to human's visual perception and the Gestalt psychology. Then finally, we'll go through the fundamentals in designing effective charts and how to use colors effectively in your visualization. Let's start and look at what Info Viz is, and why is this important.

The materials in this lecture is based in part from the materials by Professor Chad Stolper, who is now an Assistant Professor at Southwestern University. Chad were graduated from Georgia Tech, from the CS PhD Program. And he was co-advised by me and also Professor John Stasco.

So what is information visualization? The use of computer-supported, interactive, visual representation of abstract idea to amplify cognition is probably the most well known definition of information visualization. And it can help with communication and exploratory data analysis. So if EDA, or exploratory data analysis, is new to you, then you will find that visualizations are very powerful to help people explore and make sense of information.

But first, we'll start with the more common use, which is communication. As in how do you relate information to people through visualization? And a good way to understand why visualization is important for communication, I think the best way is to understand what would happen when something goes wrong.

We'll use an example from the book Visual Explanations by Edward Tufte. If you haven't heard of Edward Tufte yet, he's a very famous researcher and a pioneer in the field of data visualization. And he's also Professor of Computer Science at Yale University. You'll hear about his name a lot as we go through the visualization materials in this video.

The example we're going to look at it is the Space Shuttle Challenger incident. You may remember that it was a tragedy that happened in 1986, January 28, in a very cold morning, with only 31 [degrees] Fahrenheit. What happened was that the shuttle exploded in the air shortly after launch. After the incident, very quickly people could discover that the problem was due to a rubber O-ring, which is in the middle of the rocket, because the temperature was very cold, it become very rigid and it breaks. And then the leaked fuel goes of out the rocket, and it caused explosion. And in fact, this was highlighted in a video by Richard Feynman, who is a Physics Nobel laureate. And in front of Congress, he explained how the rubber became so rigid in cold temperature. So if you haven't watched this experiment, I highly encourage you to go to the YouTube video link there and take a look.

[[https://youtu.be/6Rwcbsn19c0](https://www.google.com/url?q=https://youtu.be/6Rwcbsn19c0&sa=D&source=editors&ust=1738598889177538&usg=AOvVaw3pida0xvxIQk3EPVUs9quT)]

So how did this happen and how is it related to visualization?

In fact, the day before the tragedy, engineers at Morton Thiokol, which is the rocket maker, presented on the day before and recommended to NASA not to launch the shuttle on the next day. In one of the slides, there's a specific recommendation that says the O-ring temperature, which is a rubber ring on the rocket, it should be higher than 53 [degrees] Fahrenheit at launch, and if it's lower, then it's probably not a good situation. And they even have a slide that shows the damage that was recorded in previous experiments. And all these rockets are shown here on a screen, it's looking at different temperatures, as you can see the numbers at the top of the rockets, and also what are the damages that are recorded. For example, on the first row you'll see that the second set of rockets, there's some damage towards the lower right of the rocket on the right. So what is showing here is that, at various temperatures, there are actually certain damages happening to rocket, here and there. And generally, it seems like that for lower temperatures, there are more damages.

However, in this chart, it's actually not very easy to see. For example, here you will see there are a lot of rockets, but as you know, a space shuttle has two rockets. So that means they're left and right and here's represent as A and B. But at a higher level at a glance, you probably see, wow, there's just so many of them. And so how should I interpret? How should I group these rockets together? And how do I really correlate the temperature and the damage that recorded?

So what Edward Tufte is arguing is that, this is probably not a good visualization, and it partly contributes to the incident because this is not very convincing and NASA decided still to launch a shuttle. And what he did was he came up with alternative design.

And first of all, he would say that probably it's more effective to show all the rockets on one line, horizontal line, instead of having two columns. Because having two columns, then you also need to think about vertically, what does it mean, what does the layout mean.

Even more, he thinks that we should probably use something that looks like an histogram, where we have a horizontal axis, which is the temperature, and then put all the experiment, each one as one set of two rockets in this histogram. So this is very familiar to what you learned about in elementary school, right? Where each pair of picture denotes an experiment that you have run. So just by putting all the data in this histogram format, you can already see that most of the experiments were run towards the high temperature scale, so around 70, while for the temperature at 30 there's only one launch. So that means we don't really have that much data to really look at what would happen during a very cold day like the day of launch. And also visually, now you can see that when the temperature is high, let's say above 70, there's generally not really that much damage. But as they go lower, starting around 70 and below, then we see a lot more highlighted area or opaque area on the rocket, indicating damage.

And Edward Tufte says, that well probably that's not even clear enough. So what about the extent of damage? As in how severe was the damage? In the histogram, we don't really see it. So what he's suggesting is that maybe we should have a damage index, which encodes the severity. Which is shown on the last column here, damage index column, the higher the column, the more severe. So by turning all those pictures or visual information into damage index, now we can see that for lower temperatures like the first row, 53 degrees, then the damage can be really high.

So that was in picture form. Edward Tufte also tried that by using a scatterplot, we can probably even see trends which may not be covered in the table format. So here we have two axes, the vertical axis is the damage index, which we explained previously, and the horizontal axis is the temperature. So each dot here is one experiment. So as you can probably already see, there's actually an upward trend going from the lower right. And then as the temperature goes lower, which is to your left, we can see that the damage tends to increase, actually it's a slope going up pretty rapidly. And if you believe in this trend, what would you expect to see at a very low temperature, which is towards the lower left of the screen. Likely it's not a very good idea to launch, because, well, if this curve is true, then the damage index going to be really high.

So the example that we just saw was for the Space Shuttle Challenger where communication could have played a very important role in the incident. So in other words, communication is extremely important, and visualization can make or break your presentation.

To convince yourself more, I highly recommend that you watch the TED Talk videos by Professor Hans Rosling where he turns everyday statistics into some of the most exciting, most compelling presentation you have ever seen.

[[https://www.ted.com/talks/hans\_rosling\_the\_best\_stats\_you\_ve\_ever\_seen](https://www.google.com/url?q=https://www.ted.com/talks/hans_rosling_the_best_stats_you_ve_ever_seen&sa=D&source=editors&ust=1738598889178673&usg=AOvVaw29rsjaSYKmtUMyms1se2Gn)]

So visualization can help with communication. Another lesser known use of visualization is that it can also help with exploratory data analysis, or EDA. As the name suggest, exploratory data analysis, or EDA, means to explore data, as in if you not really familiar with the data, then how to make sense of it. How to better understand it or get yourselves more familiar with the dataset. A lot of people would argue that exploratory data analysis is probably not really important because, well, you collected the dataset, then you probably know very well what was in it. But is that always the case? And when do you need to explore data and why do you need to do it?

So to motivate the use of EDA, I will start with a quote, which it says there are three kinds of lies, lies, damned lies, and statistics. What is it saying is that if you don't visualize the data or don't really see what's in the data and rely on statistics or numbers alone, you may not get the full picture and sometimes even get very misleading picture.

So here's the mystery dataset, and I want you to guess what it really represents. So in this dataset, we have two variables, x and y. In this table, I will tell you about some statistics in the dataset. So say the mean of x variable is 9, the variance is 11. The mean of y is 7.5 and variance is 4.122. And I will even tell you the correlation is 0.8 and even better I would do the regression for you. So I'll fit a line to the data, and I will say relation between y and x is y = 3 + 0.5x. So what would this dataset look like?

Would it look like this? So here I'm showing you a scattered plot, plotting x and y, and you see the line there which is a regression line and it cutting through the data. That looks all right, right? So I guess probably most people will agree, but now what about this one? So is a little more curvy and you may be surprised that it actually gives the exact same identical numbers that you always saw in the table. In this case, you may say, okay, maybe I would say that the blue line, maybe I should tilt up a little bit. But if you say it's the same statistics, okay, I agree with you, and that's probably okay. What about the third case, when we have outlier? One point really sticking out to the top. And while the remaining points, they are more or less lining up on the line. And just visually, you may already see that we'll probably want to ignore the data point up there, right? And so that my blue line can really cut through the middle of all the other points. But surprisingly, you will also find that it gives the same identical statistics in the table. If that's not extreme enough, there's a fourth case. This also gives you the same statistics, and in this case, we have one point far to the top right, and then all the other points lined up vertically. And unfortunately, this also gives you the same identical statistics.

So these four charts are actually from the famous Anscombe's Quartet. And this is a perfect illustration to show you that if you just look at the numbers, which is the statistics that we saw in the table, they don't necessarily tell you the full picture or the distribution of the data. By visualizing the data, we will see that you can actually have many, many possibilities that give you the same thing. This really motivates why you want to do visualization, because it really helps you explore and make sense of the data. And if someone just says here is a dataset, just trust us, you probably want to do a sanity check.

So sanity checking models is a really powerful use of visualization. And also you know that is also really good at detecting outliers. Because people can really quickly pick out things that are sticking out, such as the lower two cases, or third case, and fourth cases, where there are a data point that are really to the top right or at the very top.

In this video, we look at what Invo Viz is and why it is important. We saw that visualization can play critical roles in communication and exploratory data analysis. And not doing it correctly, can have really bad consequences.

Human Perception

In this video we'll talk about human perception. The reason we should understand human perception is because data visualization leverages human perception. So to design effective visualization it's important to understand at least the basics about it.

The human visual perception is a very powerful channel to leverage, because it can process a lot of information. So a quick quiz for you. Can you name the five senses of human beings? Well, certainly sight is one of them, and the other ones are touch, hearing, smell, and taste. Something that you may not know is that sight is a lot more sensitive and it can process a lot more information every second than all the other senses. For example, it's an order of magnitude more sensitive than touch, and too orders magnitude more so than hearing than smelling, and even a lot more so than taste.

So what does that mean? So that means the human eyes can process a lot of information every second.

So here's a very maybe even simplified model of human visual perception. When we look at something, what really happen is that there are primarily two stages. In first stage, we do something we call a parallel detection. So that means whether we want to do it or not, our eyes actually is parallelly scanning things, and where we're looking at are things that happen in the world, things that are fast moving. We're looking for basic features, and so on. And the second case is where we really focus, as in we're looking at something in detail. We want to find out what's really in the picture or we're trying to make sense of what we're looking at. And this stage two unfortunately is rather slow, and often people will say that it's more like serial processing. That means you're kind of looking at things one by one, and trying to identify what are the objects that you're seeing while looking at the layout or relative positioning of the things.

So we'll go through each of these two stages and understand why it's important to understand this parallel detection and also serial processing, and how if affects our design of visualization.

So in stage one, we often call this pre-attentive processing, so that means, whether we like it or not, we will have to do it, so we just naturally do it. It's a very rapid stage, and it happen in parallel, so that means we can do a lot of this at the same time, and it's also automatic. So automatic as in we are not really conscious about it. And it's fleeting, so meaning it lasts only for a short time. We cannot really say, ah, what happened one second, two seconds ago, so all those are gone.

In stage two, we have serial processing. So this stage would happen if stage one doesn't really tell us everything that we want to see. So in this stage two is relatively slow, so meaning we might be looking at things one by one, and sometimes we'll incorporate our memory when we try to make sense of what we see, and also it's more conscious and manual as in get to choose what to look at, what to process, what to skip.

Stage one pre-attentive processing, as we mentioned, happens really quickly. It actually happens around 200ms to 250ms. And the reason is because the eye moves every 200ms. So that means every time our eyes move or we blink, we do this pre-attentive processing together.

Now let's look at the power of pre-attentive processing. So we'll do a quick quiz to convince you of that. So here's a whole bunch of numbers here. I'll allow you to actually, maybe even a few seconds, to skim. Can you answer the question of how many 3s, number 3, how many 3s are there on the screen? Probably really hard, but what about now? Much easier. So what that means is that using opacity, as we'll see later, is a very effective way to leverage human perception for the pre-attentive processing speed, where we can just do a quick glance, not even really focus on anything, and we can quickly pull out there are actually four number 3.

There actually are a few more examples from Professor Chris Healy at NC State, which also will further convince you about the importance of pre-attentive processing. So we'll do a few exercise very quickly, and the set up is that on the screen, you have two boxes, a left side and right side, and in the next few exercise, I will put some icons or images in the left side and in the right side, and I will go through those slides really quickly, and what you will do is to try and spot, whether on the left or on the right, that you will see a red dot. Okay? So there will be a red dot, either on the right or on the left, and you need to figure out where, and I will flash you the slide very quickly. Okay, here you go. So, where was the red dot? On the left or on the right?

The answer was on the left. You probably got it already, and the main reason is because color or hue is pre-attentively processed. So in the example before you saw there's only one red dot, but there's a lot of blue dot. So now we can go for the next exercise. So same drill. So what you want to do is to find the red dot, either on the left or on the right side. Ready? So where was the red dot? On left or right?

The answer's on the right, and likely most of you also got it right. Main reason is because a shape is also pre-attentively processed. In the previous example we saw there are mostly squares, and there's only one red dot. And the final exercise. So, same thing, try to spot the red dot. Ready?

This time probably a lot harder. Maybe half of you got it right, half of you do not, and the main reason is because hue and shape together are not pre-attentively processed. While hue alone is, shape alone is, but when you combine them, it's actually exactly not, and the answer is actually on the right so for those who can really spot it quickly, or were already looking on the right, you will see that the red that was on the right-hand side.

So there are actually some very well known fact about what kind of visual features can be pre-attentively processed. So, for example, what we saw before hue, shape they are, and also very effectively would be length, width, size, curvature, and so on. So this is a quick list of all those features that can be pre-attentively processed.

[length, width, size, curvature, number, terminators, intersection, closure, hue, lightness, flicker, direction of motion, binocular luster, stereoscopic depth, 3-D depth cues, lighting direction]

And even better I would recommend that you read the book, Show Me the Numbers ,where I have a very nice chart to really show you how to group all the pre-attentively process features into a table, and also known in a very visual format. So, here we'll see the same information. We can say, if things differ by length, something stick out, we can very easily pick it up. When we're looking for size, when something is much larger than the other, very quickly it is to pick out.

[Few, S. (2012). Show Me the Numbers: Designing Tables and Graphs to Enlighten. Burlingame, Calif: Analytics Press.]

In this video we saw an overview of human perception and learned about why our visual perception is very powerful, in that it can process a lot of information at a high rate.

Gestalt Psychology

In this video we'll take a quick look at the Gestalt Psychology, and how it's related to visualization. The reason we want to understand the Gestalt Psychology is because now we're moving on to stage two about cognitive processing. So, that's more manual, so we can incorporate our experience and so on.

The Gestalt Psychology started in early 1900s, and the goal was to understand how people make sense of a seemingly chaotic world, and how do we discover patterns and so on. So, gestalt is a German word, so pardon my German, and what it means is that about seeing the whole world picture all at once. So, that means how do we perceive things in collection instead of a collection of parts.

And the psychology principle identified eight laws of grouping, and they are: grouping by proximity, similarity, closure, symmetry, common fate, continuity, good gestalt, and past experience. So what does that mean? So we'll go through each one, and we'll start with proximity.

What you will see next are a few illustrations, so let's start with first one. How many groups are there?

So there are a lot of dots, but if you are grouping things by proximity, you probably say there are four of them. Because based on the spacing between these groups of dots, you will say, there's one square and then three rectangle. So this is first principle in gestalt psychology, that people like to group things if they're close together. How many groups are there?

In this case, you'll probably will say there are two of them, there are different ways to explain this. You can say, well, there's probably six groups, one white, one black, and then white, black, and so on. Or you can say, they are probably based on the color. So all the white dots are one group, all the black dots are another. This kind of grouping is what we call grouping by similarity. We try to identify what are the common things in the dots, and then we will think about them as one group How many shapes are there?

So despite there are some broken segments in these two shapes here, one is a circle and one is a rectangle. Human beings are really good at completing all these missing parts. So what that means is that we can understanding by doing closure, so we like it close things together. If some missing small part, we are very good at completing them. How many items are there?

You probably agree there are three of them. And the reason that we can spot out there are three is because people like symmetry, so they like things that are mirror images. So first one you can say is parentheses, and then brackets, and so on. How many sets are there?

So that's one animation of some dots moving. And one set of dots are moving towards the right, and the other moving towards the left. And why would I say that? The main reason is because, people are very good at spotting continuity, so in other words common fate. So we'll see one group of dots having the fate of going to the top right, and then the other group going to the top left. How many objects are there?

You likely will say that there are two keys. But if you really analyze it more closely, you will see that the key on the left is actually overlapped, or behind, the first key to the right. And the reason that people can really say there are two keys, instead of saying there is one key and there's some key fragment, and it's because people like continuity. So we like thing that would continue, that would continue to go along, instead of saying, there some broken pieces. We like to make things more smooth, more continuous. How many objects are there?

So this is an interesting one, because if we interpret these images by actually negative space. For example, in figure one here, we don't really have a triangle here, but instead we have three dots, and each of them has like a small cut out. And together, we imagine there is a triangle. And similarly, in C, we see a mace, even though there's no sphere in the middle and we only saw some spikes. And similarly for D, we expect to see that this is probably like a snake, maybe on top of some water. Even though we don't see most of the parts of the snake body. So this is what we call a good gestalt, so that means that it just look right. People will say, so it has to be a triangle, has to be a sphere there, or there has to be some plane cutting through the snake. And what is this word?

For most of you that word's flight, how hard can it be? But if you really look closer, you might see that, well, L and U are actually really close so I could have interpreted it as a U. But then you also know from past experience, that FUGHT, that's not really a word, so you're actually using your knowledge to interpret this word. So that means when we are looking at things, we are actually using our past experience, or our memory.

In this video, we look at the eight laws of grouping in Gestalt Psychology, which can be a good way to explain how we perceive and make sense of the world.

Chart Basics

In the previous video we learned the basics of information visualization, human perception, and also the gestalt psychology. In this video, we'll learn about how to apply all these principles to design effective charts.

You may recall from previous videos that we talked about Pre-Attentive Processing and also the Gestalt Laws. And we say that Pre-Attentive Processing allow to detect things very quickly. But detecting quickly doesn't really mean that we can detect things really accurately. Which means if we can say something happened, it isn't really necessary to say that we can figure out exactly how much something has changed. And ideally you want both.

There have been research to look into this. Specifically there was a study in 2010 by a Professor Jeff Hair and his students to look at how do different kind of charts would allow people to detect things quickly and accurately. They look at different kind of charts, such as bar charts in T1, T3, and also charts that uses area, such as circular area in T7 and treemap in T9.

And what they found was that for charts that uses position, such as in T1. Like in bar chart where the changes are encoded by length, people can very quickly and accurately say what the changes are. While as you go down further, for charts that use area, like pie chart, circular area, treemap in T9, for example, then people are not really good at doing so. Actually, for circular area that's particularly problematic because if you say, you show people two circles and say which one's bigger? Which one bigger number? And often they can just tell you which one's bigger but they cannot exactly tell you by how much because there's an ambiguity in interpretation. So one way to interpret it could be actually by area which we'll do some math to compute it. And other is to say well we can compare them using their radii or their parameters. And both of them are not really easy to compare. And for that reason this chart can be really inaccurate and you see that actually circular area is the most problematic here.

In summary, what that means is for positioning or using length, those are really accurate and you can also detect them very quickly. And as you go down, where let's say we leverage the angle or slope of the items that we are looking at, then we get worse. And even worse are the cases where we use area or volume. And for the very interesting case is that color is actually the least accurate. So you might notice that when we talk about pre-attentive processing, we say color or hue. We can detect it very quickly. While we can detect them quickly, we actually are really bad at saying by exactly how much. That, let's say, two shades of red are different. So, this means that you ideally want to combine both the visual and coding, like positioning and length, and also more subtle design decision by using color more effectively, so that the design would be most accurate and can be detected quickly.

So what does all this tell us? It tells us that bar charts, scatterplots, and line charts are all really effective for visualization. And because they combine all the best, or the most accurate, and also the most easy to detect. For example, in bar chart, we use its length. In a scatterplot we use the position, and for line chart we use a little bit of slope and also position.

And, in general, for statistics distribution besides using line chart, bar chart and scatterplot, one very good tool is the Tukey box plot. You've probably seen this already, if you haven't, here's a very quick overview. You'll see some rectangular box here and also maybe some lines and dots. And what this very compact virtualization encodes is the median in the middle, and also the quartile, first quartile on the lower part of the rectangle and the third quartile at the upper part of the rectangle. And also, there are some points here which are showing the outliers, things that are not really falling into the expected range.

So what we saw previously was to say that, well, there are three kinds of charts that are really effective. But how do we design them? We can follow Tufte's chart principles. So this is probably second time and you'll see him a lot more, second time that we see Edward Tufte. So what are the principles? So the very first one is what we call do not lie! And the second is to maximize Data-Ink ratio. And the third is to minimize chart junk.

So we start with the very first one, do not lie. What does that mean? Why would we ever lie with data visualization? So, here's is a very good example. So you probably remember the oil spill incidents a few years back and there was of course, oil collection effort. And in one of the presentation we saw a chart like this where the company representative say, we are doing a very good job at collecting oil, and you see that our collection is actually ramping up. So if you look really closely to the title of the chart, you will see that this is actually a chart representing the cumulative oil collected over the dates. So what does that mean? If we're looking at cumulative, that means the sum is always going to increase. In the worst case, it just stay the same. So on the right here we're showing the oil collected daily. So that means no longer cumulative. And you will see that actually that's not really ramping up anymore. We're actually slowing down. So there's a ramp up in the beginning. But then once it reach the peak, then it can slow down. And then in the later days actually stall and then stay at the same rate. So, that is an example of what Tufte would say, do not lie. So that mean we should be truthful, we should be trying to pick the chart that would tell the best picture as possible.

And the second and the third principle, maximize Data-Ink ratio and minimize chart junk often get discussed together. The second one means we want to spend all the ink that we use, let's say when we printed our chart out, to be dedicated to the data. So that means every single drop of ink, ideally it should be representing data instead of representing some kind of decoration. So by maximizing this Data-Ink Ratio, that means we are really showing our people everything, all the essential things that they should care about. And naturally that would also help us minimize Chart Junk, because Chart Junk means declaration or things that are dormant that are not really critical in having to understand data. So by maximizing Data-Ink ratio naturally, we will also reduce Chart Junk.

So here’s a very good example to show you what we mean. On the left we see a bar chart here. So the data that we really care about actually is encoded completely in the length of the charts. And while the green background, for example, has some bricks, you can see them clearly. And also a lot of horizontal lines, for example, they're actually occluding or even making the labels, let's say the axes, on the vertical axis or the horizontal axis, really hard to read. So it has a really low Data-Ink ratio, and also there is a lot of chart junk. And similarly on the right, we have a map. And here, you can also see that the gradients behind each of the country is actually also not necessary. Because all we care about is the boundary and also the name of the country. And all those gradients, although they might be interesting to look at, they are not really informative.

And finally, I would say please, please, please no pie charts. No 2.5D charts. So pie chart is actually really popular. You have probably seen a lot of these in business presentation. But I would argue that they are really not good visualizations.

So, let's say we look at this example. And I would even give you a hint. I would say, well there are three big pieces of pie here. And two of them, actually all three of them are different. And the two biggest pieces, the dark gray and the lighter gray, they're actually slightly different. So which one is bigger? Probably it's pretty hard to figure out. So the darker one is actually bigger, is a value of 37, and the smaller one is 36, off by only 1. So here you see that pie charts are not really good at helping people to distinguish between these small differences.

And this is what we call 2.5D chart. So this is really, really bad. The main reason is because it's unclear what the length here is really encoding. We have like volumes of pillar. And if you look at the vertical axis, you'll see that it starts from 0 to 40. Then, what can we say about the first bar? How tall is it really is? Well notice that the vertical axis is not really starting at whether is the front or the middle of the column. Instead, that's like a little bit offsetting, more coming, to the front of the screen. So what that means is we don't really know where to start the measuring and where to end the measuring.

And this is even worse. So we not only have pie chart, we also have it in 2.5D. So please, please, please don't ever do this, not really good.

And if you really want people to be able to detect things accurately and very quickly, you can just use a bar chart. Even though you might say this is very boring, but it is extremely effective. And here, same data you saw previously using pie chart and 2.5D bar chart. Here you can very easily see that one is slightly longer than the other one.

What we just looked at are some basics of chart design. And we look at what to do, what not to do. And we saw the scatterplots, bar charts, and line charts are really effective, while for some things, like pie charts or 2.5D charts, we should rarely do them.

Colors

Today, we're going to look at colors and how to use color effectively in data visualization.

So in previous videos, you probably saw that I use black and white a lot or grayscale a lot. So you probably got pretty bored about that. And even using just grayscale can be pretty risky and needs a lot of thinking.

Here's a quick quiz for you. So on your screen, you see there are four rectangles, small rectangles in gray. Which one is darker? You probably may see that the left one is darker and the right ones are lighter, but actually they're the same. So the reason that we perceive the left one to be darker was because it was in front of a white color. And the one to the right, when you say it was lighter, was because it was in front of a darker color. So what that means is that how we perceive color actually affected by the context, and also by the color surrounding what we're looking at.

Color is very powerful, however, so if we can use it appropriately, we can call attention to the right amount of information that people should care about, or make it more attractive, increasing the appeal. It might help people to better remember the visualization. And of course, it add another dimension to work with, so can use different colors to encode information. So now we're going to understand a little bit more about color and how to best leverage it.

We'll first start with the ways to represent color digitally. And RGB's color model is the most common way to do it. And it is a what we call an additive color model that's based on mixing red, green, and blue light together to create different colors. So it’s very commonly used because a lot of devices like cell phones or a TV, the colors a generated based on a combination of this light.

However, even though we see in RGB, we don’t really necessarily interpret thing in RGB. The more intuitive way to interpret color is actually using the HSV color model. So this is a color picker from an illustration program called Affinity Designer.

In the HSV model, H means hue or what most people call color, and S is saturation and V is lightness. So hue is pretty easy to understand. So you can just by looking at it is this the kind of rainbow hue going around. And for lightness, you can think of it is how do we get a lighter red, so as in by adding more white. For saturation, you can think of it how to remove the color like desaturate or adding more gray to it. This is actually a more common way for people to interpret color, because often we start with mentioning the hue. Very rarely, we will say, it's maybe lighter red or lighter blue. Saturation, usually you don't even mention it. But nonetheless, that's actually a very good way to also represent color.

So when you use color you need to be careful because about 10% of males and 1% of females in the world are red-green colorblind. So that means if your visualization design depends on using red and green to distinguish between values then about 10% of population actually cannot tell the difference.

And there are even more people who are colorblind, so that means the may have a hard time of distinguishing any color. Look sample here, if you have a normal vision, then you can see 74. If you are color blind, then it might come out as 21.

And also, when picking color, you want to pick a scale that would not accidentally give you some unintentional meaning. For example here, we are using the color scale that go from white to dark blue and then transiting to red and to yellow to encode temperature. And take a moment and see what you see in the image. You probably say, this is like something on fire, right? So because you're using yellow and red to encode the high temperature, but is this really what you want to convey in your message? Probably not, you probably wouldn't want to say that more Southern States are on fire, right? So this is an intentional use of color scale.

And also, when you use color, don't go overboard. There's a downside of doing that. For example, you may think that using more color is probably better, right? So you can make it more attractive and so on, but it's actually not good for a lot of quantitative data. For example, you may have values that are in a range, let's say between zero and one. But then you decided that for the lower value you assign the purple, middle value is yellow and orange, high value's green and red, that is not really good. Actually, it's counter intuitive and really detrimental. Main reason is because we can really say which color is lower or higher than the other. For example, they just give you purple and red, which you say purple necessary smaller or something people would actually say purple is higher.

So to help you design more meaningful color schemes, so this is a very good image.

[[http://www.personal.psu.edu/faculty/c/a/cab38/ColorSch/Schemes.html](https://www.google.com/url?q=http://www.personal.psu.edu/faculty/c/a/cab38/ColorSch/Schemes.html&sa=D&source=editors&ust=1738598889187929&usg=AOvVaw05pSTjbaF1fdloOc-b8MOS)]

So this is a guide for you to look at first what your data is trying to convey and then pick an appropriate color. For example, on the top left, it's saying that you only have two kinds of information like binary, yes or no, then you can use shade like one is very dark, one is much lighter. And similarly, at the top right, so we had categorical data. So that means you don't really care about the absolute difference or relative difference, you only want to convey out there. Actually, three different categories, then those two, three colors, two or three colors should be quite different. So in this case, we're using red, blue, and green as far apart as possible. Actually, you may even want to make them color blind safe, so that everyone can see them. So towards the lower left, if your values is between the scale of let's say -1 to 1, so that means you cut across zero. And then we would want to use something we call a diverging scale. So that means we may want to have a neutral middle point, in this case, using gray or white as a middle point. And then for one end is a negative, we use a dark purple, and then towards the right, we use orange, and then anything in the middle would be a gradient. And similarly to your right, you have the sequential data. So it's very similar to binary data, but in this case, we have more separation of gradation, instead of only two, here we have three. And you can also mix all these up. For example, right in the middle, we not only have diverging information, but also within each color, we have sequential information. So I'm not getting a matrix of color. So this is a very good guide of how you might want to determine which kind of color combination you may want to use.

But if this is still too complicated, so this is a very good website you might want to use, it's called Color Brewer.

[[https://colorbrewer2.org/](https://www.google.com/url?q=https://colorbrewer2.org/&sa=D&source=editors&ust=1738598889188688&usg=AOvVaw3x-Xmq0vBVCwvDbwI4TLHO)]

So it's an excellent website, because it will show you a preview on the right, here we're showing a map, and look at how your colors are apply. And also on your left, you have the option to choose different kind of the color scale. For example, you can pick sequential and here, we pick divergent which is good for ranges and values as they go from negative to positive. So we have two colors here. Or you can pick qualatative, right? So that means categorical, you don't really care about absolute differences. You can only say there are maybe seven category, five category and so on. And also even better is on the bottom left of the screen. You get to choose to only focus on the color scales that a color blind see or that are print friendly. So that means if you print them out in black and white using black and white printer, it will still show up really nicely. So this is a color picker website that I would highly recommend. So it would really take a lot of the guesswork out.

In this video, we look at color and also look at how we may apply color effectively in data visualization. And we should take great care when using color, because a big part of the population may not be able to see color. And also we talked about a very good website ColorBrewer for you to choose color scales, so that you don't need to do the guess work.

Fixing Common Visualization Issues

Fixing Bar Charts, Line Charts, Tables, and More

In previous videos you learned about the basics of charts and also how to apply color effectively. In today's video we're looking at how to apply those principles and specifically, we'll do learning by example. So we'll look at some of the charts that can be improved and then we will look at how to improve them. So let's start.

And some of the materials that you're going to see today is based on this book, the Guide to Information Graphics, the Dos and Don'ts of Presenting Data, Facts, and Figures. And it is by Dona Wong who is the student of Edward Tufte. So this is probably second and one of the many times that you're going to hear the name Edward Tufte.

So there are also other books that I will recommend. So the author Stephen Few, and Now you See It, Information Dashboard Design and Show Me the Numbers. So those are really effective and compelling examples that you will find in those books that will guide you how to do effective design.

So we'll first look at bar charts, and here we have two series of data. And we decided to use two different colors, red and green, to encode the different series. So there's an issue with using red and green. First of all, maybe a question for you is, what does this remind you of? So very likely, it's Christmas, right, because it's red and green. And there's an issue with that. It's because a majority or a big majority of the [colorblind] population actually cannot see red and green. So red and green blindness is very common.

So what I would recommend is using a color scheme and stay away from the color scheme red and green, and instead use a color scheme such as those based on gradations, such as using different shades of blue. Or if you do want to use different colors, then yellow and green actually show up really well and most people can see them. And as we already mentioned before, it's a good idea to use a website like ColorBrewer to find good color schemes that might be colorblind safe.

The next example we are looking at another dataset. And here, we've decided to use red to encode company profits. So what's the issue with it? So my intention of using red could be to highlight this chart, so actually you will see quite a bit of charts that use red as highlight. But since our chart topic is company profits, so red is actually not a good design choice. Because red often means something that is negative, something that is not good. So if you have good things to say, such as company profits, actually using red is a bad idea. So instead, you would want to probably remove the color and then think very carefully about what are the things that you really want to highlight. Maybe, for example, only one of the bars you want to highlight. That maybe in that one, you choose very carefully, maybe use blue or green which tends to mean a good thing.

So as a joke, so probably the only time you will want to use all red is when your company is having losses. And in that case, you want to use red. But also in that case you probably don't want to be working in the company any more so if you have this many losses.

And also, when you're doing bar charts, we need to be careful about how we encode data by the length of the bars. So at a first glance, looking at this chart, you might see that probably nothing is really wrong about it. We're here showing the growth of company profits, let's say, from 2004 to 2007. This is actually very similar to what you would often see in papers and reports and so on. But if you look very, very carefully at the vertical axis, you will notice that it starts at 75 and the maximum range is 100. So what that really means is that, even though here you see a pretty dramatic growth from 2004 to 2007, actually the increases may not be as impressive as it sounds, right? So 2004 is at 80, in 2007 is at 100, so it's only about a difference of 20. So what that means is that using a axis that is not really starting with 0 can give the false impression that something is a lot more impressive than it really is. So this is, according to our Tufte's principle of do not lie when you are doing chart design.

So for most bar charts I would say, almost always the vertical axis should start with 0. So that means we should be truthful about how we do our design.

When we are visualizing multiple series of data using bar charts, or often, as we mentioned, we may want to use different colors or different color gradation. So this is also an example now that you will see often and probably even the default of a lot of the charting software. They say, okay, I'm going to automatically assign a color or a color gradation for each of the series. So this is likely the default you're going to get. So what is the problem with this design? You might find it's is pretty disorienting. So as in the first series is medium gray and then lighter gray and then black and then white and then darker gray. So as you may remember from the Gestalt psychology, people like to see things a lot smoother. So things that are going at abrupt changes, high contrast, are quite disturbing or disorienting.

So a better design will actually be using gradation. So for example, same kind of data but different color choices. So here we start with the lightest and then slightly darker, darker and then black eventually. So this is much more pleasing to the eyes and also easier to pick out the times on a series.

When working with bar charts, you also need to think about labels. So labels, as in the vertical axis label, horizontal axis label. And often, when you're doing a vertical bar chart as shown here, the labels may be too long to fit them horizontally. And often the solution to resolve the problem is to rotate the label. So for example, here we tilt the labels at an angle, so that we can actually show the full length. There's an issue with this which is that by tilting the text, it makes the text a lot harder to read. You may say, well, so that is the default, so what can we do about it? There's not really enough space. So this is a very good example to illustrate that often the default may not be the best design for your data. For example, bars can actually be horizontal, so you can use horizontal bar charts. So no one is stopping you from doing that. And in this case, it's even better in that by doing the bar charts in a horizontal layout, you can fit all the text very, very nicely. So that means the labels are very easy to read. And what that means is that do not settle for the default. So think about what are the small modification you can do to make the chart much easier to read.

In fact, you can get a lot of inspiration from everyday websites or news reports and so on. For example, this is a screen shot taken from Apple, where they're also showing bar charts, right? So here they're looking at how the different interfaces speed are different and they're doing a comparison. And you'll notice that they're actually using a bar chart here. But the bar chart is very well designed in that the data's encoded by length and also the label is directly underneath closest to the data. So in general, putting data illustration or information closest to what you want to say. That's a very effective design. So take inspiration from what we see every day

So now we will switch gear a little bit to line charts. So line chart is another very popular kind of charts that you will see. So this is a chart that you might, someone may give you, and they may say, I have already plotted the data, and I even have given you the axes, so I labelled the vertical axis 0, 3, 6, 9, and so on, and we're looking at some growth over some time, for example. What's wrong with this? So, can you improve this chart?

So if you look closely you will notice that the interval is at every three units, so 0, 3, 6, and 9. So this is actually a little unnatural, so usually what people are more used to seeing is interval of five or interval of two right? So, even though the previous chart was correct, in that the data is correct, the axis label is a little unnatural. So, given we have the flexibility on doing or changing that intervals or labels pretty easily, we should choose to pick the more common interval, like 2, 5, 10 and so on.

And when designing a line chart, we need to pay special care about how we are designing the range, as in the vertical range here. So if we set the range to be too high, that means the lines now become too flat. Then it can obscure the message you want to convey. And similarly, if we set the range to be too narrow, where the lower point of your line and the higher point of the line is really touching the range completely, then we will exaggerate or overstate the trend that we want to capture in the data.

So, what that means is that we want to aim for something that's not really going for the extreme. So a rule of thumb for designing line chart is to have the data or the data line to be roughly two-thirds of the range of the charts. So, of cause, this is a rule of thumb, so that means there may be rules that we can break. But generally, this roughly two-thirds guideline is very good starting point.

Sometimes we need to visualize multiple line charts altogether so that we can compare them. For example, for stock prices it's actual quite common to see multiple stocks and then we have all of them lined up. So, how will you design that effectively? So a common default that you'll see, as shown on the screen here, is that the software would assign a pattern or like a mark for each of the line that you have. So maybe a dotted line, maybe a solid with circle, or maybe solids with hollow circles, and so on. So there's nothing really wrong or inaccurate about this representation, but again, if you may remember what people prefer to see often is things that are not as overwhelming. So how can we kind of calm people down when they look at this figure? It's overwhelming them with a lot of different patterns, a lot of different mark. So maybe you want to take a few seconds and think about what you could really do by applying some of the things that you have learned before.

So have you thought about how to do that? So here's a good suggestion, which is that you may want to use a color, you may want to use a line thickness instead of using patterns and marks. So using the marks as in circle, triangle, and so on, so those are actually pretty hard to distinguish between when you're zooming out. Sometimes at a distance circles look pretty much like rectangles or small triangles, and also patterns are difficult to distinguish also. So for example dotted line or versus slightly more spaced out dots still actually are very hard to distinguish between. So, on the other hand, you may recall that thickness and length are really good, really fast for people to pick out the difference of. So using line thickness here is very effective. And then also you notice that we're doing double encoding, so double encoding as in, not only line thickness, we're also using the brightness. So using both of them will ensure that when we do print this chart out, let's say in grayscale, everything will still show up very nicely. So this is a very good design alternative that is significantly better than the default, which is using marks and patterns.

So sometimes you have a lot of lines that you want to compare, and it's not really possible to put all of them, all at once in one chart. For example, in the previous design, it may be good for about seven plus or minus lines but if you have tens, twenties, or even a hundred lines then that's too much to fit. So what you can do then, is to use what we call a small multiple, it's a term coined by Edward Tufte. So small multiples, meaning using a series, or create a small similar graphic. So usually using similar or the same range in the axis, horizontal axis and vertical axis, and putting all these charts in a grid, so that they can be very easily compared. So in this case here you will see that we have six chats here, so all the horizontal axis can expect to have the same, let's say same time range, and then all the vertical axis you can expect to have the same vertical range as well.

So now we switch gears to table. So tables is also very common, especially if you have a lot of numbers to represent, and you want to show people exact values, then table is a great choice. So in this design, can you spot any problem with it?

So notice that we are having a lot of things in bold, where there are only four numbers that are not in bold, so we can ignore the zero, just assume there are some concrete values there, and having everything in bold is pretty problematic, or in other words, by making everything in bold or everything in italic, by doing that to emphasize is actually not a good idea. So main reason is because, when you are making everything bold, everything italic, it also means you are actually deemphasizing everything. So this is actually a very good quote from my favorite movie The Incredibles. It says, when everyone's special, no one really is special. So, I'm honing that principle to charts, that means when you're making everything bold, everything in italic, actually nothing really stands out. So think carefully before you apply these emphases like bolding or italic.

So this is a likely default that you will get when you are visualizing table of data, and meaning all the grid lines are turned on, and again there's nothing really wrong about it in terms of accuracy. So you show all the number, you show all the row and column heading, but this is the scenario where Edward Tufte would say that is is a very low data to ink ratio. So we remember data to ink ratio, meaning how much ink, if you are to print this chart out, how much ink are you going to spend on printing the data. That means the numbers, the labels, and so on, and how much data are you going to spend on the grid lines, which may not be essential to interpreting the data. So in this case we're doing a lot of printing for the grid line. So that means a lot of chart junks, so things that may not really matter.

So how to improve this? So, of course, first is to try and remove the grid lines, and then you gradually add the grid lines back, which will help you better interpret the data, such as using the grid lines to Group the data let's say every four row, so better organization. And also you can use a background color to highlight a column that you want to emphasize.

When you're representing data, we need to be careful about how we do alignment, all right so here we're showing you two examples, and the one on the left where we're doing left alignment and instead of right. And then on the right we have numbers that have decimal points, and something you might notice is that the decimal digits they are not really align and also for column three company C, it actually doesn't have a decimal point. So not using the same number of decimal points or not having any outer decimal points at all, can be misleading, because at a first glance I'm going to see that whoa, the number for Company A is much higher it has four numbers or for Company C is much smaller, all right. So to help improve this what we should do is to unify the number of decimal points that we want to use, and also to align them by decimal points, so it it much easier to glance at and to compare them.

So as we have mentioned before, pie charts we should really use them rarely. But there may be times that we will be given some pie charts, so in those cases then, what can we do about them? How do we now be truthful and when we're using pie charts?

So why do people like to use pie charts? So even Steve Jobs used it, so there's a screenshot of a much earlier presentation that he gave many years ago where he's showing the market share of Apple products and comparing to other competitors. So pie chart's a very popular choice in marketing. And particularly in this scenario where you may notice the pie chart is not only, it's not only a pie chart but it's also a tilted pie chart. So we mentioned that this is the 2.5 the chart that we exactly want to avoid. So why is that? So in this case you notice that Apple has a lower part of the chart, and tries to detour the audience. So what that means is that the Apple pie is actually appearing much larger than it would be because of perspective, so it's hitting towards their audience, all right? So for that reason in marketing a lot people like to use it because it gives false illusion that are things are bigger than they are.

So if you really do need to use pie chart, so please, please, please, so never use this kind of design. So something you notice here is that the pies, these are pies so do not even add up to 100, so which is absolute no, no, so if you're really doing a pie chart everything should add up to 100 or add up to 1.

And possibly, the only scenario I can imagine that is a perfect use of pie chart is to show people how much of the pie that you've really eaten, and how much you have not yet eaten. This is the legitimate great use of pie.

So when working with a large amount of data, you will need to think not only about the design of the chart but also the axes. And so far we have only been looking about what we call linear scale, so meaning every single point, every single number, we're faithfully mapping that specially to what people will see. But when you're working with a large amount of data you're maybe dealing with numbers in high range of orders of magnitude. So in that case, so how do we compress all these numbers and still make them all fit on the screen or on the piece of paper.

So in those case and sort of linear scale then you would want to use a log scale. So log scales will allow you to compress a wide range of data and then put them in a much more compact layout. So here in the example, showing you how we apply a scale to both vertical and horizontal exit. That means we are doing a log-log plot. So data that you are showing here is what we call the file prevalence. So meaning how many times or how many machines that a file, say Microsoftword.exe, would appear on. So for example, the point at the top left, right, so horizontal axis is file prevalence, so here is we say a very low prevalence file, that means so far only appears on a few machines. So maybe this is a file that is pretty new, so only appears on a few machines. And the vertical axis we see, there are actually close to 1 billion, to be exact is 850 million such files, so here we highlight that in red. And notice that the range, the vertical range, is actually going from 1 to 1 billion, so extremely wide range, but because we're using log-log plot, so we're able to compress and still able to show that dot on the same screen.

So the log scale is also great for compressing time as well. So here horizontally, we are showing the chart in time, and we're compressing from 1 month, 3 months, all the way to 30 years. So that means in general if we want to compress things and put in all the data in a chart then using a log scale may be a good idea.

So in this video we'll look at how to fix common issues in bar charts, line charts, tables, and also briefly discuss why we do not want to use pie charts.

Applying What You've Learned

Previously, you learned the basics of charts and also how to fix common issues in chart design. And today, we're going to look how to apply those techniques and principles and more holistically fix the issues that you may see in the chart design.

The first example that we're going to look at will be this one. So suppose you can ignore the meaning of the data points. So here we're showing a bar chart with, let's say the population of four towns, Town A, B, C, D. And also we've already given you the heading and also subheading. So can you improve this? So you may want to take a few seconds and think about what you may want to do. So keep in mind that you want to apply techniques and principles that you've learned in previous videos.

So let's see what you've come up with. So on the left is our original design and on the right is our improved design. So what are the changes that we have made and why do we make them?

So the very first thing that you might see is that we're removing the emphasis, such as in the heading, where in the previous design we are using white text on black. And that's a very, very bold and possibly overwhelming design choice. So as we mentioned, a good idea is to actually strip these emphasis out first and then very carefully add them back in. So once we remove them you will notice that well, by just making it bold is probably already sufficient. And similarly for subheading, you'll notice that we remove the bold and italics. So this is the application of the principle that we said that we should not really use over emphasize or overbold our text, so in this case, using just normal text is fine. And also you notice that we apply the same principle as in the vertical axes. We are not bolding it because actually there's no real reason that we make the bolding. And for the horizontal axis, so we apply a technique of not tilting the text so that we can make it easier to read. So you notice that just by applying these simple principles all together, you make the chart much, much easier to read.

So now let's look at another example. So this is an example taken from the website darkhosrseanalytics.

[[http://www.darkhorseanalytics.com/blog/clear-off-the-table](https://www.google.com/url?q=http://www.darkhorseanalytics.com/blog/clear-off-the-table&sa=D&source=editors&ust=1738598889194402&usg=AOvVaw0MSj1enwxcvwLdQ1ESSpEW)]

This is a really great example because they also show you the step by step way by how to improve this. So arguably what we see here, which is the before scenario. That's a very common table that we'll see in many kinds of different reports or even academic papers. So I will argue that this is probably not the best design and you can do a lot to really greatly improve this. So now take a few seconds and see what are the designs choice that you'll make to improve the design.

So, now let's see how much you've gotten right. So let's do a quick review. So something you notice after we move all the colors, all the blue and light blue colors, and also all the green lines are gone. So those are what I would say are chart junk so they're not necessarily help the user interpret the data, but more of a decoration. So also you will notice here, we are using color pretty carefully. So we have most of the text in black and then only the row, the third row, that we want to emphasize and have it in red. And also if you notice that we have very careful alignment of the numbers. So we're aligning all of them to the right and also all of them have the same number of decimal points. And also you will notice that we 're adding using spacing for effective grouping. So every five rows we have a little bit more vertical space. And also on the first column you'll notice that we're removing repetition. So meaning for the second to the fifth row, we're removing some of the text so inside there just redundant. So again you'll see that in this example we are applying a series of simple techniques but all together they will create a very effective redesign of our table.

So in this video we look at two examples of how to apply the previously learned techniques and principles more holistically to improve those designs.

Crown Jewel, Self-contained Figures, More Tips

Today we're going to look at a few more tips in designing effective charts. And also you can think of those as a practitioner's guide. So when you're designing things or using figures in your presentation or papers, what would you want to be careful about.

So we'll start with colors. So colors, as we mentioned, can be really effective, but we should be using color carefully. So my recommendation is that you often will want to start with grayscale or black and white. And after you have a very good content and layout, then you start to gradually add in color for emphasis. So starting with grayscale is a very good starting point because it forces you to focus on the content and layout, instead of the color choices. And for fonts I would suggest using sans-serif. So sans-serif meaning the font that do not have the small features at the end of the strokes. And sans-serif usually tend to be simpler and easier to look at. So if you're using a Mac like me, then the font Helvetica is a great start. For Windows I believe that would be Arial.

And for animation, similar to our use of colors, I would recommend starting with no animation. That will also force users to focus on the content and layout. And after that then you gradually add in the more meaningful animation. So animation can be really effective. Especially you want to convey the transition or the dynamics of your data. In those cases, animation may be a good idea. But you want to think very carefully about those before applying them.

So charts are actually one of the many kinds of pictures that you can use. Pictures includes tables, diagrams and charts. And I recommend using pictures and even better would be video. It's because pictures often are more succinct and memorable, and also people like to see pictures, and even more so for movies. And pictures are great, because they can attract peoples' attention, so that they don't fall asleep. And because once people fall asleep, it's really hard to wake them up. So general rule of thumb is, when you have good stuff, show them right away.

Going along the same lines, when using pictures, how do we use pictures effectively let's say in a paper. So this is what I call a crown jewel image that you might want to consider including in a paper. So crown jewel image meaning it's give you an overview of the whole paper, of the whole project. And in this example, this crown jewel image is showing the main idea of this work. So if you look at the title, it says scene completion using millions of photographs. And while in the crown jewel figure, you'll see there's a series of four images. And they actually are very, very succinctly, and very nicely done. It conveys the meaning of the scene completion. So, it says that from the left, in the beginning we saw the original image. And then the second image there, we are saying that there's a portion of the image we want to remove. And then the third one we will say we are using a lot of images, looking at a lot of images. And then we do what we call scene matches. And using those images then we fill out the part that we want to remove from our original image. So the fourth one, the one to the right is the output. So here you're just looking at this series of animation, if you would. You can already get a very good idea about what this paper is about. So this is what I call a very good and effective crown jewel image. So in general, for every project, every presentation, you should try and come up with something like this. So this is really what I call we cut to the chase.

So give an overview, to help people what they're going to get right off the bat so that they don't need to figure out or they don't need to wait. And the main reason you want to do this, is because people often skim, and they tend to look at the most interesting thing first. And if you are researcher or [inaudible] student, then this is particularly important. Because you are submitting this for review, let's say to a conference. Often reviewers, they are very busy, and they can assign 5-10 or even more papers to read. So it's really great for your paper to stand out, and having a crown jewel image is a very effective way to do that.

And in practice, how do you pick the figure? You can use your most impressive figures and put it up front. It doesn't mean that you do not want to show that figure again. You can actually show that figure again. Maybe provide more information. So that means the crown jewel image may be the most compelling or most impressive thing that we want to show. And then later on, you want to provide more detail.

Including a crown jewel figure, any figures that you design, you should have to figure to be what I call self-contained. So meaning, if someone were to skim between these figures or flip through the pages of your report or jump through different slides of presentation. They should still be able to get the main message of what you want to say in the image by just looking at the image as caption. The main reason is because people skim, right? And also even if they don't skim, if you only provide very little information in the caption of the figure then you will force people to go back and forth, jumping back and forth. And you may be able to tell from your experience reading a book, jumping back and forth between pages actually not very good idea. It will slow you down, and probably even confuses you. So, making every figure self-contained is very important. And also because figures are often the very first thing that people will be looking at. So having a very hard to read, requiring people to jump back and forth, will give a very bad first impression.

So, how do we fix that? To fix that is to think about what is the main take away message that you want to convey. So, here on the right, I'll zoom in a little bit, is an example of what I call a self-contained image. So, you can ignore what the chart is showing. That's a more technical, but even just skimming it, looking at the caption, here we see that the ROC curves of 7 iterations and true positive rate incremental improve. So the main message here is to show that sometimes numbers, here we say true positive rate incremental improves over 7 iterations. So that's our main message. So this is pretty easy to understand. And also in the figure itself, you will see that we clearly label iteration 1 to 7. And then also we point at each line at each iteration. And vertically we highlight the improvement from 0.849 to 0.871. So this is what I would call an example of self-contained image, where just looking at a figure itself, looking at the caption, I already get a main message about what I want to convey.

So to put everything together, the crown jewel and also the self-contained principle. Then often what you see in presentation or in papers like this is that we'll put a figure on the first page and often at the upper right corner or may be even extending and taking up the full first row of the upper right half of a paper.

If you're giving a presentation, a very good first rule is to use legible fonts. So the main reason is because if people cannot see it, then they would not appreciate it. So that means no matter how impressive the things that you want to show, if they cannot see it, well it doesn't really matter. So for printed material, I recommend that you actually print them out. Because on screen, you may be zooming into the text and everything seems legible. But once you print them out, they may the font size maybe just too small. So for presentation, the rule of thumb is that you might want to fit about 7 lines of text on a slide. So again, this is the rule of thumb. It's generally a rule that you cannot break. But in general, for legibility, especially when showing the presentation in a large lecture hall, for example, then the 7 line of text rule of thumb is a very good one to follow.

And when you are putting figures in presentations, then you may find that you often may need to recreate the figure. So here we're showing you a figure in the paper, where we show not only the exact values of the bars but we also show the error bars. But if we were to put this figure let's say in presentation then we likely don't want to put all these numbers there. Instead we might want to come up with something like this, where you will notice that we've removed the error bars, we've removed the values. And also we simplify the title too, because the main message that we want to convey here is that whatever tool that we're developing, in this case, Apolo, is better. So that means the main message is we are faster, we are better and the Apolo software wins. So in this case, you see that we added a big arrow to help convey the main message.

And the last tip is about presentation. So presentation is extremely important, and it is an advertisement of your work. So it's probably also the very first time that someone else see your work. So, it doesn't really matter how great your work is if they do not understand your presentation, then it doesn't really matter. Because they have never heard about what you do, and this is the first time, and you are not really selling it very well. So there's actually excellent videos also from the Comedy Center from Don McMillan, so I highly recommend that you look at. It's not only funny. But it's also very, very informative. For example, we'll look at how bad color schemes can affect readability. For example, having black text on a very dark red, extremely hard to read. And also using bad or tiny fonts, very hard to read. Too much animation can distract people from the main message or maybe even putting too much data. So this is actually often a very first thing that a lot of first time presenter may run into, is that they try to cram everything into presentation. And it's not a very good idea because that just makes everything not very easy to read. So same rule here. You were putting a lot of things and you're basically de-emphasizing everything. So think carefully about what you want to put on the screen.

So in this video we'll look at some of the practical tips of how to improve figures and presentations.

Week 5

Data Visualization for Web (D3)

Why Learn D3?

Starting with this video, we're going to look at D3. So D3 is a very popular way of creating data visualization for the web. And you may want to learn D3 because this allow you to deliver your visualization across a wide range of platforms, and reaching wide ranges of audience.

Some materials from this video is based on Dr. Chad Stolper, who is now Assistant Professor at Southwestern University. So Chad was actually a co-advisee of myself and also Professor John Stasko. So in Chad's work these materials is actually what I would consider the crash course of D3, and/or early sticking points or only the beginning. Actually there's a lot to learn about D3.

So you probably seen a lot of the D3-based visualization on the web. For example, here is a screenshot from Bloomberg, which is a very interactive storytelling kind of infographics that show you how the car uses in America looks like over time. So this is something that you can scroll through. There's a little animation and tells you the whole story and is a combination of many charts. And you have probably seen a lot of examples like this, and a lot of them are created based on D3.

So, why would you want to learn D3, and when would you want to use it?

So we recommend that you use learn D3 if your vision visualization, your system, or your tool will actually benefit from interactivity. As in when the user interact with your chart, they may be able to get more information out of it, or to explore the data better. So on the other hand, if you want to create static charts, then you should probably use just anything you want, like using Tableau, Excel, Python, ggplot, and so on. And there's actually a lot more discussion about this. So this is common question. People say, so I hear about this D3 thing. Should I really learn about it? So this is what it says here on Hacker News. So it's saying that D3's good if your visualization benefit from interactivity. And D3 may not be appropriate if all you want is a more basic, more static chart. The main reason is because D3 has a very steep learning curve. So it's actually consisting of many parts. You need to learn JavaScript. You need to learn some HTML, CSS, and so on.

And this video is about D3 version 3. Version 3 is actually not the latest version. Version 4 came out not too long ago, and this is the latest. [Version 5 was released in 2018.] And however, it has what we call breaking changes, so that means coding written in version 4 is actually not backward compatible to version 3. And the reason we are staying with version 3 for now is because most D3 examples and tutorials that you see on the web are still using version 3. So for that reason to help you better learn and be able to learn from these examples and tutorials, that's why we stay with version 3. And there's an excellent article that compare version 4 and 3, so basically telling you what are the major changes. And even better, there's another web link here that show you if you want to upgrade your code from version 3 to version 4, what you want to do.

In this video, we'll look at a quick overview of D3. And we said that it's a great choice for web-based graphics if you can benefit from the interactivity. But on the other hand, if you want to create more basic charts or static charts, then D3 may be overkill.

Prerequisites: JavaScript and SVG

We're going to continue with D3. D3 actually consists of many parts, like JavaScript, SVG, CSS, HTML, and so on. So actually quite a few more I call prerequisites. So let's look at them before I dive into D3.

Since the trials that you create in D3 will be running in browser, so that means you will need to know how to debug your D3 charts. And to do that in Chrome, for example, what you would do is open the web page that include your D3 charts. And then you can right-click on anywhere, and then you can click the inspect button. And that will allow you to look at the source code of your chart. And in other browsers like Firefox and Safari it's similar. So right-click on any element on screen, and click inspect or inspect element, then it will allow you to look at the source code. And also something you will need to do quite often is to also bring up the console, so that you can see the error messages, since you will be programming in JavaScript.

Something that may be new to most of you is that when you are running a D3 visualization, especially on your local host or local computer instead of running it through the web, instead you will need to start what we call a simple local server. The main reason is that if your chart is reading data or data file from your disk, then there's a security measure in most browser to prevent that from happening. So this is a necessary thing to do for Chrome. It's not necessary for Safari or Firefox. And to a launch a simple server, the easiest way is probably using Python. So in Python 2.x, you will type python -m SimpleHTTPServer, and you also need to provide a port. And in Python 3, the little shorter command, python -m http.server also need to apply the port 8000. And after you launch these two commands either way, then you will go to your browser, and then you will say localhost, and then :, and then you provide the port number. So this will allow you to access file that are hosted on your computer. So in other words, it's like you're turning your computer into a web server, and you're just accessing the same web server as what you're using.

So if you're new to JavaScript, you will need to be doing a lot of that when you are creating D3 charts. And if you're new to it, you will need to be prepared with a lot of confusion. So JavaScript is a scripting language, and it can do a lot of type inferring. It's pretty flexible, but also because of flexibility, it can keep a lot of hair pulling moments.

And I'm pretty serious about it because that is something that you will be running into a lot. And this is actually a great video that I encourage you to watch. You can start with 1 minute 20 second. And it will tell you about some of the funny, but probably also frustrating things that we run into and help you foresee the quirks that you need to work with when you're programming with JavaScript.

So we'll give you some JavaScript 101 or JavaScript 102, if you would, about what you might need to be careful about. So if you're new to JavaScript, all variables are global unless you add the keyword, var, V-A-R. So what that means is that in a function, if you have, just say, create or declare a variable x and then assign 300, value 300 to it, then it's actually a global variable. That means you can access it everywhere. On the other hand, if you want to restrict the scope to only the method itself, then you should add the name var.

And semicolons in JavaScript are optional. So you can include it or not. And my suggestion is that be consistent. So if you are using semicolon, just use them use them throughout. If you don't, then just do not use them throughout. So just avoid confusion.

Double quote and single quote in JavaScript is the same. And JavaScript arrays and objects are similar to what you expect in Python lists and dictionary.

And object.key is the same as kind of object, and a square bracket, and putting the key value in quote.

You will be printing error messages or text to the console a lot. So to do the printing, you will do console.log.

So JavaScript supports functional programming style. And a lot of people say that JavaScript is more like a multi-paradigm programming language because it also supports other kind of programming styles. So what that means about functional programming is that functions are themselves objects. So that means they can be stored as variables. And because of that, they can also be passed as parameters into methods. And D3 uses this ability extensively. So if this is new to you, functional programming, so here's an example to give you an idea why it does.

So in this example we are trying to take the square root of a series number. So we have number 1, 4, and 9, and we want to take a square root. So that means we would get 1, 2, and 3. So in the example here, we are using a function called Math.square. And this is a function that we're passing as a parameter to numbers.map. So the map function is a function that will apply whatever you pass it into as a parameter. And then apply the function to every element of an array. So what that means is since our function is Math.sqrt, so that is the square root function that can be applied to every single element in the array 1, 4 to 9. So naturally, we're going to get 1, 2 and 3. So here is an example you just saw, functional programming.

So we need to look at a lot of information about JavaScript in general. And the Mozilla Developer Network is the excellent, probably the best JavaScript reference. So you can, of course, go to through the website, go through the whole reference. But it's really long, so the easier way to do it is actually just Google the command. So we can say, type google, I go to Google, and then type a command and then mdn, which is short for Mozilla Developer Network. And that'll give you a full information about the JavaScript commands.

In D3 you will see a lot what we call method chaining. So method chaining is what we call syntactic sugar. So that means it doesn't really change what the program does. But this is more to potentially make the program easier to read. So what that means is that every method or most of the methods in D3, they will actually also return the object itself that it was called on. So here we're looking at two pieces of code that does the identical thing. So for example, at the top we say group.attr ("x", 5), and attr ("y", 5). So what that means is we're trying to set the x and y coordinate of whatever group is to 5 and 5. So here you notice that right after calling .attr, we called .attr again. The reason we can do it is because the .attr method return a reference to the object which is group. And so that allowed us to call .attr again because essentially, we're calling the function of the group. And of course, it's equivalent to writing out the two commands one by one. So that means same effect as in calling group.attr("x", 5), and then calling immediately, group.attr("y", 5).

In D3, to create graphics, you will need to lean about SVG. SVG stands for Scalable Vector Graphics. So you can think of it as a mathematical formula that you can use to generate very smooth looking graphics, where no matter how much you zoom in to the graphics, it's still very smooth. It's different from what we call raster graphics like JPEG, GIF, PNG, and so on, where zooming into those pictures, you will see that they are pixelated.

When working with D3 graphics, you also need to work with a very unnatural coordinate system. So in data analysis, when creating charts, often we expect the origin to be at lower left. But in SVG or in HTML, the origin actually is top left. So that means when you're creating bar charts and so on, you actually need to think how to start from the top left instead of starting from the lower left.

So SVG stands for Scalable Vector Graphics, and is actually an XML document. So that also means that you're essentially writing markup. And by processing those markup, you're able to generate these graphics. For example, you can say, I want to create a circle. So you can write out the line circle r=5, fill="green". So that means you want a circle radius of 5 and using the fill color green. And it's part of the W3C Standard and supported by all three major browsers. So for that reason, it's great to deliver your visualization in D3 because everyone can see it.

So we'll cover some of the basics in SVG. For example, we'll cover the SVG elements, circle, rectangle, g, which stands for group, and also how to show text.

So the SVG element is what we call the overarching canvas. So this is the canvas where you would draw your visualization on. And optionally, you can provide the width and height when you create it. And to create SVG the way that you do it in JavaScript is you would want to first pick out the container. So in this case, we have a div with an ID called vis. So we will first select it. So select and then #vis, so that we're selecting the div by its ID. And then you would call append. So that is when you actually add the SVG element to the div.

So after adding the canvas, then you must want to start adding elements like adding circles, rectangles, and so on. So they are very similar. For circle you will need to specify the attribute like the center of the x coordinate and the center of the y coordinate. So again, remember that the origin is at the top left. So that means all these values are relative to the top-left corner of the container. And you will also specify the radius. And optionally, you can provide the fill color, the stroke, the stroke-width. And similar to adding SVG to the canvas, you will say .append, and then provide circle as the tech that you want to add.

So rectangle is very similar. So here you will specify x and y. So again, this is relative to the top-left corner. And you will also provide the width and height of the rectangle. And similar to circle, you can optionally provide the fill color, stroke, and stroke-width.

Creating a bar chart using rectangle, it may be conceptually very simple. But again, remember that since the coordinate system, the zero start at top left, so that means you want to create a bar of a certain width and height. You will think about how it would start for this layout relative to the top left. So this can be a little unnatural because you need to actually subtract that part of the bar from the top left.

So when creating charts, you will likely need to manipulate a group of elements such as in a bar chart. You may want to apply the same color to all the bars in the chart. So instead of positioning every element individually, so how do we do that to save ourselves some trouble? So instead of placing everything one by one, can we do that all at once?

So you can use the g or the group element to do it. So g is a generic container element, so that means you can put all different things into it. Really, you can think of it as just a way of grouping things. And after grouping elements, now you can apply transformation, let's say, moving the whole group of elements all together. You can also change the color, change the stroke, and so on all at once. And very similar to autotext, you will see the way that you create a group is you just append and then append g. And once the group was created, to add things to the group then you can say group.append, let's say, add a circle, or append a rectangle, append text.

When creating visualization you will likely need to select things such as selecting all the bars and then applying some color, or selecting all the data point of a particular series and then doing something to it. So there is a very powerful way based on CSS, Cascading Style Sheet, to select things. But to select things you need to first assign things to what you want to group together. So there are different ways of doing it. So one is you assign an ID to every individual element. So ID is supposed to be unique. So that means once you use an ID, you should not use it again. So you already saw the previous example where we can assign, let's say, the ID vis to a div element. So to recall this particular element then you would use the pound key. So #vis, that would give you the element that is assigned ID, vis.

Another way is to recall an element by its tag name. For example, if you want to get all the circles, then you can just get say, I want circle. So here in this case, there's no pound key.

And the third way is by class name. So that means you can assign the class to different elements. So they can be a mix of circles, a mix of rectangles. And in this case, we can say that as long as an element has the class name, then we can recall it using .class name. So here we have the class name canary. So .canary, that would give us all the elements that have canary as their class name.

And the fourth way is by attribute. So that means I want a specific value. Let's say, I want all the circles with colors blue. Then you can use the syntax here to pull all those elements out.

Actually there are numerously many more ways to assign things or to select things. So there's a full list of them, and the URL here shown on the screen.

A very powerful way to select and assign thing is through the combination. So for example, you can actually do something called AND. So let's say I want all the circle that has a particular class name. So that means I want circle and canary, for example. Then doing circle.canary, that will be applying the AND condition.

And similarly, you can say I want circle or anything that has a class canary. Then what do you do? Then you will do circle, .canary. So here, slight difference from the example you saw previously. Here we're using the comma to separate the two kind of selection. And what we will get then is all the circle no mater what class names they are assigned to. And also we will get anything that has the class name canary. So that means you can get a rectangle whose class name is canary and so on.

So in this video we looked at some of the prerequisites of D3. So about JavaScript in general, about SVG, and also about CSS, how to assign classes, and how to recall classes.

D3 Overview

Now that we have looked at the prerequisites of D3, now we can dig a little bit deeper into D3. And look at what it is, what it's capable of doing, and also a little bit of history about D3.

So a little bit of history first. So, D3 started as actually called Protovis at Stanford. Inspired by Professor Jeff Heer and his former student Mike Bostock. And in 2011 it was renamed to D3.js, and also Mike graduated. So now, he's at New York Times, which might partly explain why on New York Times we often see a lot interactive graphics. And Professor Jeff Heer has moved to University of Washington.

So a very quick overview about the outline of this and the next videos that you are going to see. So we start with what I call the very grand reductionist statement about D3. So in a nutshell what it really is. And then we look at how we load data into D3. And then the very important paradigm in D3 called Enter-Update-Exit. And then we look at some more specific things like scales, axes, doing layouts, transitions and interaction. And also some extra information about where to go next.

So in a nutshell, D3 is what we would call a very powerful for-loop with a lot of useful helper functions. So it's not a very compelling way of describing D3, but it's really what it's doing. So it's really a for-loop, so meaning it can do a lot of things. And for you automatically. And it also provides you with a lot of fantastic helper functions, so simplify your life. And more technically, D3, you can think of it as what we call a declarative domain-specific specification language for manipulating the DOM. The DOM is document object models. So that means how do you manipulate HTML document in this case for visualizing data?

When you're creating D3 visualization, the very first thing you need to do of course, is to import the D3 Library. And to do that you will be doing that in the header section and using the script command. And something you might notice is that here, not only are we loading Java, we're also specifying that character is set, and here we use utf-8. So that means we are using unicode. And you may say, why do we need unicode, so, it's just text file, right? The main reason is, unfortunately, in the source code JavaScript for D3, we have some unicode characters, such as, pi, so, for that reason, we have to use unicode. So, that's a little bit inconvenience, but it's incredibly important, because if you do not set to utf-8, then they'll be error.

So, after importing the library then you will want to add the visualization element, right? So, you remember from previous example, what do we need is a container. So, let's say we create a div container with id vis. And then after that, then you can add the SVG element. So it can append SVG. So after calling the above command by selecting first the div with the id "vis", and then you do append.svg, then it'll be added to the div element, as shown below.

So now you're ready to load data. And as you may recall, d3 has some fantastic helper functions to help simplify your life. So for example you want to load CSV file, you just need to call .csv. Similarly for a tap separator file or a JSON file. So something you notice here that in those function call, they're also parameter called callback. So this is where you can provide a function that will be called when the file loading is done. So this may be a little unnatural for some of you who have never seen this before. So what this means here is that JavaScript is actually loading the data asynchronously. Once you call, let's say, .csv then we actually continue execution of your file. So that means if you have a large data file, which can take a few seconds to run. So in those cases then, you may want to wait a little bit until the data is done loading. And then after it's done loading, then the callback function will be invoked.

So the very first time that you use the .csv function to call or to load data, there may be some surprising thing that you run into. So let's say we have a toy dataset of people, each row is a person, and then there's one column which has age, so which has numbers. So we're gonna use a .csv function call to load this data into a let's say JSON internal format. Then you will notice that there something is special going on. And you have a good eye to spot that. You will notice that age is actually encoded as string. So it has double quotes around, for example 18 is quoted, and 22 also quoted, 30 is equal to as well. So what that means is that the type is not automatically inferred correctly. And we need to fix this because they should be integers.

And the low tech way, unfortunately, to fix it is to loop through all the numbers, and then you reassign the integer version of those age value to a cell. So that will help JavaScript to infer this is actually a number, because only numbers can have the plus sign. So this is one of the example quirks that you will need to deal with when you're programming with JavaScript. So once you do that, then now you check the internal representation again, you will see that now the numbers are correctly encoded.

So once you've loaded the data correctly, then the next thing you could is to map this data to visual elements. So essentially it is what we want to create, so visualization of the data.

So remember how we want to do it is to using D3 are very helpful series of transformation that we provide. So as we mentioned, this is a declarative language for manipulating the DOM. So that means we want to create elements as circles or bars in a visualization. What we want to do is to tell D3. So declare what we want to do, and then allow D3 to figure out how to create those. And at the very high level we're going to do is to define a template. So that means we tell D3 visually what we want each element or each number to be turned into. And then D3 would draw one element for every numbers on the screen. So we'll look into more details about how to do it exactly.

In this video, look at how to set up the environment in D3. How to load the library, how to load the data, and so on.

Enter-Update-Exit

Today we're going to look at a very important paradigm in D3 called Enter-Update-Exit. So it's important to know because it's how D3 assigns visual elements to datapoints.

So Enter-Update-Exit is probably the most critical facets of how D3 works. And probably if you remember nothing from today and only thing you may want to take away would be Enter-Update-Exit, Enter-Update-Exit, Enter-Update-Exit. And so it's really, really important.

So this is related to how D3 assigns elements to your data. So the pattern in Enter-Update-Exit is that it will select a group of elements, so let's say circles that you want to create. And you assign the data to this group of elements, and then based on when your data comes in, or to be removed, the elements actually get drawn on the screen.

So there are three phases you can think of in Enter-Update-Exit. So in the Enter phase, this is where new elements are created for data points that are not yet associated with anything. And then there's the Update phase which is where attributes are set for all elements based on data. And then the Exit phase is when elements are removed that do not have data anymore. So elements here in orange are actually visual elements. And data in green are the actual data points. So at a first glance, so this may be similar to abstract. So what do you mean by creating elements of data, they are not associated so they're abstract. So we look at the example very quickly now and then figure out what all those means.

So here, we have data that we want to load. So imagine in the beginning you actually have no data. And so every time we called a function .data, that is when we load in data or change the data that we're seeing. In this example, imagine you do not have any data yet, and we want to see in all those three phases what we're going to get. So, nothing in the beginning. So the first time we call .data, that is when we're going to change the data. So in the beginning, since we don't have data the first time we called .data(), 1, 2, 3, 4, as an array, so that is we're going to see in the interface. So highlight that in orange here. In the Update phase, remember that we're going to look at all the elements, right? So since 1, 2, 3, 4 is all we have, so that means 1,2,3,4 is all that we're going to manipulate. And then in the phase three, the Exit phase. So since we're not removing anything, so that means there's nothing there to manipulate. So that is the initial scenario. So to recap, we start with nothing. We load on the array, 1,2,3,4. And that end phase Enter and phase Update we'll be able to manipulate the whole array.

And now imagine we are going to change a data. So, instead of 1, 2, 3, 4, now we want to change the data to 1, 2, 3, 4, 5, 6. And to change a data, we will call .data again. So internal into D3 is keeping the state of the data so that it can actually track whether the new items that are added. So what means is that in the Enter phase, now the elements there is 5 and 6. While for the Update phase, you remember that we try to manipulate everything, everything that we have seen. So that means the whole array, 1, 2, 3, 4, 5, 6. And for the Exit phase, since we were not removing anything, so that means nothing to manipulate.

And finally, now suppose we want to remove some data. So data point we want to remove 4, 5, and 6. So how do we do that? We call a .data again. In this case, we're only passing 1,2, 3. So that is telling D3 that we want to remove 4, 5 and 6. So in phase Enter, so nothing to manipulate there since we are not adding new things. Update, when you think about what it is, but for Exit we know that's easy. So 4,5, and 6, that's where we going to manipulate. So what are the question mark? So Update is actually still 1, 2, 3, 4, 5, 6 because nothing has yet been moved yet until the Exit phase.

So as I recap, so the Enter phase is where you see new data point. And for Exit is where elements that will be removed can be manipulated. So, also that Enter and Exit will only exist when the data has been called. While for the Update phase, you get everything. Everything that you've seen, everything that you will remove and so on.

So, this hopefully will give you a more concrete idea about what the Enter-Update-Exit means. And that's an important thing to know as because that is how D3 allows you do transitions, how to handle dynamic data and so on. If this is still a little abstract, I highly recommend that you go to this link down there. Very, very good example to help you better understand what all these phrases mean.

And some more examples, actually, there's one interactive demo to help better explain what End-Update-Exit does. And there's even a very long tutorial about this paradigm.

In D3 whenever you call .data, what it assumes about your data is that the index of each data point will become the key. So that means the index become a way that it can pull a particular data point.

But often the index automatically assigned may not be the best where you want to refer to your data. So in that case, then you might want to set it yourself. For example, the two lines down there essentially designing your own function where you can say the first one I want to use the data's ID value as the index. And the second line saying that you want to use the name of the data value.

This is a typical skeleton of a D3 program that applies the Enter-Update-Exit pattern. So in the first real line, you'll see that we're loading the data using the .data command. And then after that we'll be entering the interface. And then in the middle is the Update. And then finally is the Exit phase.

So a big warning. When you are looking at a lot of examples online, often you will see that they drop the word group as in this example where right after calling .data, they immediately enter the Enter phase by calling .enter(). So we highly recommend that you don't do that because it may confuse you as in which phase you are working with. And this is particularly important when your data changes, as you will see in later videos, where knowing very precisely when data are added, when data is manipulated or removed is incredibly important. And the main reason that often examples can just drop all those and have interface immediately called after .data is because those charts are static. So relatively simple, so everything can just all come together. But I highly recommend that you do not, so just include specific commands so that you can very clearly see where the phases are.

So in this video we look at the very important Enter-Update-Exit paradigm, look at some of the caveats in using it, and also some example to explain it.

Attributes, Styles, Classes and Text

In this video we're going to look at how to manipulate the visual aspect of D3 elements, such as the attribute, the styles, and also how to assign classes to elements so that you can more easily select and change them.

The first tool you may want to use is the .attr() command. So it allows you to change attributes such as the x, y coordinates, the width and height, the fill color, and so on. And to use it you can call it on a circle element, or on a group element as in shown below group.attr ("x", 5).

So as mentioned before, you may not want to apply all the change to attribute of every element one by one. So you can do it in the functional programming style. So suppose here we have a data point 10, 8, and 12.2, and you want to map this to a series of rectangles. So which already showing you in the middle, what we're going to get. So we have three parts, the height based on our data point.

So what we'll do here is use a command or defined a function return d.size, so that is a way for us to pull out the size attribute, from our input data, and assign that to the height of the bar.

So, text in SVG is not well designed so, you will need to work with it, unfortunately. So here's some caveats that we want to point out, and often these things are things that you just need to memorize, or keep referring back to.

So to specified text in D3 you will use the .text command, and then put the text that you want to create, and you can set the x and y attribute of the text block, and something that's confusing is that depending on where you set the text-anchor and dominant-baseline, you can have very different looking text.

So for text-anchor, we have three possibilities start, middle, and end, and dominant-baseline: [nothing], hanging, and middle.

So what are they? So why are they so confusing? So here's an example, for text-anchor, if you do not change the default, the default then is on the left side. So we call this start. So that's basically our (0,0) coordinate for the text block, and similarly for the baseline, if you do not do anything that's at the lower left. But you can also change it to the middle or hanging, so let's say if you want to change to hanging then the top left becomes a (0,0) point.

So here is an example to illustrate why it is confusing to work with text, because it's dependent on what text anchor you're going to pick. For example, here we're showing three pieces of text, all of the x-coordinate are set to 50, but depending on the text-anchor, let's say the middle one, we set text-anchor to middle, then it can shift it to the left, and similarly, if we use text and character End, than it's going to be shifted more to the left. So that means that you can end up having very different looking text, and it can be confusing to a lot of people because they say, I have the same x-coordinate, but how come my text has shifted?

So next tool you might want to use is the .style() function. So a light attribute, it is for styling things, and it's great for putting in CSS styling. For example, you can use .style, and then you say, property1, change to value 1, and similar to .attribute, you can also add functions, or use function to determine what the value is, so that means you can change a group of elements. For example, based on say all the data points, and then you have the function to look through all the data points.

So the unfortunate thing about a style is that you will need to remember when to use .style and when to use .attribute. For example, here for a text .attribute you can use it to set the text blocks x and y coordinate, while you need to use the .style to change the dominant-baseline and text-anchor. So, unfortunately, you really need to remember when to use .style and .attribute. There's not really a good way to go around it.

And sometimes you need to select different visual elements or you need it to find different elements, and how do we do that? Let's say we have two kind of circles when we want to be able to distinguish between them. So this is done through what we call classing. So we already seen some example previously in the overview of D3. So here more concretely we see, through JavaScript, what you're going to do.

So let's say here we have two groups of circles, one is blue circle, the other is the red circle, and what you can do is to use the .class function, and you first select the group of elements you want to assign the class to, and then you invoke.class, and then you assign the classname "bluecircle", and say "true". So true meaning, you want the class to be assigned, if it's false, then you want the class to be removed. So, once you've assigned the name bluecircle to all those circles, then later on what you can do, as is shown in the last line here, is you're gonna say, do selectAll (".bluecircle"), because we're calling a class, selectAll(".bluecircle"), and that allow you to change all the color, and other kind of attributes.

In this video we look at some of the helpful functions such as .style and .attribute, and how to use them to change the visual appearance of your elements.

Scales and Axes

Today, we're going to continue with D3. And we're going to look at some very helpful functions in D3 such as Scales and Axes.

So Scales are something that you would want to transform a range of input values into a range of output values. For example, you can use scales to scale the size of a circle based on the data value.

And if you don't use scales, then some things can blow up pretty easily. For example, if you use the linear scale and just map all the data value directly to height, then you can imagine when your value is very large, then the height of the bar, for example, will become to big to draw.

So the good news is that D3 has a lot of types of built-in scales. And in this video, I'm going to cover only two of them, Linear Scales and Ordinal Scales.

So linear scales define the mapping between a range of input values we call domain to a range of output values we call range. Here at the top, the domain function takes in an array, specified by the lower bound min and upper bound max. And similarly, for range, which is the output, we use input values of minOut for the lower bound, and maxOut for the upper bound.

And below, you see how we're going to use that scale that we just defined. So here we're trying to specify the x coordinate based on the scaled value of the size from the data point.

But how do you figure out the min and max for the domain? So remember that D3 is a really powerful for-loop with a ton of useful helper functions. So that means we already have functions that can do those for you. Specifically, we have functions like d3.min, which takes an array of numbers and then return the minimum value. And similarly, .max, .extent that would return the maximum value and also the range, lower bound and the upper bound.

So here's an example of how to use that in practice. Here, in this example we have an array of numbers as input, which is at the top. And at the bottom there you see we use d3.max and take in that array as input. And that would give you the maximum value for that rate, which you can pass into the .domain function.

So here the input's an array of number, but if the numbers that you want to scale is actually part of an object, then what do you do?

So the good news is that d3 allows you to use something called an accessor to extract the number that you want from an object. So here in this example, data.map is the accessor. So specifically, we're pulling out the age value from an object.

So here's how to put everything together. So at the top here, maxAge is the maximum values returned by the accessor function, data.map. And then we use the maxAge below, and put it into the domain.

For ordinal scales, D3 has built-in color scales, such as category10, for example, it gives you ten colors. Category 20 will give you 20 colors and also there are variations like, 20b, 20c, and so on.

So D3, in fact, doesn't really care so much about what you pass into an ordinal scale. And the only requirement is that the domain is discrete. So, that means you can pass in data that's actually ordinal. So meaning the ordering is meaningful, as in color scale, where you have discrete values like like, neutral, dislike and so on. Or the data can also be categorical such as blood type like A, AB, B, O and so on. So what that really means is that you shouldn't just use the color just because they are provided. You should think really carefully about whether your data's ordinal or categorical. So if it's categorical, then it's fine to use all the color scale directly. But if it is ordinal, then you don't really want to use a rainbow palette, for example.

So here's an example of how to use the categorical scale in practice. So here the input is an array of three animal objects, and each has a type. So we have Bird, Rodent, and so on. And in the color scale, each color has an index. For example, we're using category10. So that means the color for blue is index number one. Color for orange is index number two.

So when you use this color as in the middle, then there you see that we're passing in the type of the object into the color scale. So that means what will you get is at the bottom, where Bird is assigned a color of blue, Rodent the color orange, and so on.

So besides scales, which transform data, D3 also has visual helper functions such as axes. And similar to scales, axes is also very easy to use.

Here we're showing you an example where you are using the axes specifying the scale first, and then will specify the orientation. Here we're saying we want the orientation to be in the left. You can also similarly specify to right, top, bottom as well. And also, you can specify something like tick marks. How many tick marks do you want? Default is ten, and here we are using six.

So in this video, we look at how to use some of the important D3 functions, such as scales and axes.

Dynamic Data and Interaction

Today we're going to continue with D3 and specifically we're going to look at how to handle dynamic data, meaning data that change and also how to support user interaction in your visualization.

So dynamic data is very common. For example, if you are building a visualization for stock prices than you expect the top stock prices to come in every second. So you have a new value every second. So how do we handle new data points like these? And, or perhaps, how do we add transition or animations so that the data point doesn't just appear abruptly?

So actually you already know part of the solution. So remember what we are showing here is the skeleton of the D3 program, where we have the ENTER, UPDATE and EXIT phase. So all we need to do to handle dynamic data is to actually wrap this skeleton as a function.

For example, we are wrapping it with the function name called redraw. And you will call this function redraw every time you have new data. So meaning whenever you have a new data point, then you will pass the data into the redraw function, and then automatically will update that visualization.

And even better, if you want to add animation to show the transition between old values and new values then you can simply add the function transition and apply it to any selections of elements that you want.

So transitions in D3 are pretty magical, so meaning it can automatically interpolate the in between values for you. So as long as you know what's the old values, what's the new values, then intermediate values will be automatically computed.

So here an example of how to use transition. So suppose you have a rectangle where its original height is 0. Let's say it's a rectangle in a bar chart. And now you want to change the height to 5 at the bottom below and also change the fill color to green. Then all you need to do is to add .transition and then can specify the delay. How long do you need to wait before the transition starts? So that means how long do you need to wait before the height start to change, the fill color start to change. And also so you can specify how long the animation is. So here we're using 200 milliseconds.

So using transitions allow a visualization to become dynamic, but that doesn't necessarily mean that it support user interaction. And to support user interaction, you use the on method.

So here's an example of what we meant. Let's say whenever a user clicks on the rectangle, we want to change its color to blue. So how do we do that? We use the function on. And also, we specify click is the event that we're interested in. So that means mouse click. Then whenever the rectangle is clicked, then what we will do is we will update the data value for color to blue, as in the second line here. So the .data changed to blue. And then once this data is changed, all we need to do is to call the redraw function, which you'll remember, wraps around the enter update exit skeleton that you saw previously to redraw or to update a visualization.

So in this example, we are interested in the mouse click event. And also we can handle mouseover, mouseenter, and mouseout, so then you can't have hovering, for example.

So there are a lot more to learn about D3. So the best resources would be to go to the homepage of D3, which is the first link. And also look at the official D3 documentation which is extremely well written. And also D3 has done a really fantastic job of collecting a lot of tutorials online. So the third link here shows you probably all the tutorials that you may want to find. And also, when you are programming D3, a very good resource is StackOverflow. So doing a lot of Googling and looking at how people actually solve problems on StackOverflow is a great resource.

In this video, we look at how to handle dynamic data, data that changes, and also, we look at how to support user interaction.

Week 6

Scalable Computing: Hadoop

Big Data Is Common. How to Store Them?

Starting with this video, we'll talk about how to scale up computation. We'll learn about collection of system and tools, such as the Hadoop and Spark ecosystems that will help us more easily work with large amount of data. But before that, let's take a moment to think about the large data. How large is large?

In general, there isn't really a hard definition for what is considered large. But most people will consider data in the petabytes or beyond to be large where one petabyte is about 1,000 terabytes.

You may ask, do we really need to work with such big data? The answer is yes.

In fact, Big Data is very common. You can find a lot of examples in the tech industry. For example, back in 2009, Google was already processing 24 petabytes per day, and Facebook adds about half a petabytes of data every day. In physics, the CERN particle collider generated over 200 PB of data in its experiment. You can also find Big Data in the movie industry. For for example, the 3D effects in the movie Avatar took one PB of storage.

So how to store such large datasets?

How about using one of our own computer? Large hard drives are readily available these days for example CK announced their 60 TB SSD drives. So yes it's possible to use a single machine but if your data is in the scale of a few terabytes.

But if it's in petabytes then you will need to use multiple computers. Once you use multiple computer there are a lot of things you will need to consider. For example, how many machines you use, what to do when a machine dies, or when a hard drive dies, you may ask do I really need to worry about that? My computer hasn't died yet. Well, first congratulation you're very lucky. But the bad news is your computer will eventually die I'm sorry. And bad things tend to happen together. So it's likely that your computer may die at the most inconvenient time, like right before deadlines. So backup is important.

And going back to our discussion about using cluster machines, how often to hard drives die? Is it really a concern at all? And the short answer is yes. Google run an experiment and they found that 3% of their 100,000 hard drives failed within the first 3 months. And 3% of 100,000 is 3,000. That's a lot of hard drives to replace.

So, in summary, you need think about big data, because they are coming. And to analyze them is as important that you use tools that provide some sort of backup and redundancy because computers and hard drives die often.

Why Hadoop?

In this video, I'll give you an overview of Hadoop.

You may recall from the previous video that when we analyze large data sets, we need to think about quite a few things. For example, we need to think about what software libraries to use, which programming languages to learn, so that our programs and algorithms can run on multiple machines. And more generally, what framework to use. Hadoop was a very nice solution for all of those problems. Some of the materials and example in this lesson come from the book Hadoop: Definitive Guide. It's a very nice book that provides a good overview of Hadoop in a number of related technologies in the Hadoop ecosystem, such as Pig and HBase, which we will cover in future videos.

Hadoop is an open-source software for scalable, distributed computing. It is written in Java and has a number of components. And it's designed to scale to thousands of machines, to offer linear scalability in the ideal case, meaning if your computer cluster has 100 machines, your program would run 100 times as fast as when you only had 1 machine. Programs that run on Hadoop would use a simple programming model called MapReduce, which we will go through in more detail in our next video.

A very nice feature of Hadoop is that its file system, called HDFS, provides automatic fault tolerance, meaning if a computer dies, you can just replace it, and you don't need to restart your computation.

You should learn Hadoop because many Fortune 500 companies use it, as well as many research groups across the world, so it has very strong community support. It's also free and open sourced. And it's relatively cheap to set up. A Hadoop cluster can be built using commodity machines. This is in contrast to super computers that each may cost millions of dollars. And since Hadoop is so popular, it has become an essential skill like SQL.

A fun fact, in case you're wondering where the name Hadoop comes from, it's named after a toy elephant of Doug Cutting's son, and that was one of the Hadoop co-creators.

So in today's video, we looked at what Hadoop is, why you want to learn about it. And in the next video, we'll look at how you can write programs for Hadoop.

MapReduce: Overview and Example

Today we're going to look at MapReduce, the programming model of Hadoop. If you want to write a program that runs on Hadoop, that program mention in a special way that follows a MapReduce model so that your program can leverage the scalability and fault tolerance that a Hadoop cluster can provide. First let us understand MapReduce at a high level, then we'll dig deeper into the details.

The high-level Hadoop scale computation using a master-worker architecture, meaning a master computer coordinates many worker computers to work together to solve a computation problem using the MapReduce model. A simplified way to think about this is that there are two main stages.

Stage one is to divide the data and its associated computation into smaller pieces so that each machine can work on one piece. After that, in stage two, the results from all the machines are combined to produce the final results.

Now let's describe this in more detail using technical terms from the MapReduce model. We now call our original divide stage the map phase. We will see shortly were the name map comes from. In this map phase, the work that is done is still conceptually the same, meaning data and computation are still divided into smaller pieces. But now we review that it's the master computer that decides on how this division is done, such as how big the pieces are. Also, we now give each worker a name. We call them mappers. Each mapper, as before, works on any pieces of data and computation that a master has assigned to it. The mappers work independently and in parallel. For each piece of data and computation, a mapper will produce a corresponding intermediate result, meaning we will get many pieces of intermediate results from the map phase.

Now recall that eventually, we want to combine all these intermediate results into final results. So we need to move intermediate results from the mapper workers to the workers that do the combination. We call this moving of intermediate results shuffling, because intermediate results actually get moved from one machine to another across the network. But the good news is that this shuffling phase is automatically done for you.

Now after this shuffling, the intermediate results arrive at the right worker machines. And we call the workers that does the combination reducers. These reducer workers also work independently and in parallel.

In this video, we learned about MapReduce at a high level. In the next video, we'll see an example MapReduce program.

Example MapReduce Program

In this video we'll see an example MapReduce program. This is a famous word count example. It's like the Hello World of Hadoop.

Let's first look at what the input is and what we will get in the end as output. As input we have a collection of text files For simplicity we use only two, and each file contains only one line of fruit names. Just eyeballing those words, you can already tell the answer, that is Apple, 4, appear four times, Grapes, 1, Mango, 2, and so on. So again, given a collection of text files we want to know every words frequency.

This is a small dataset so you can easily fit all the text files in memory and then count the words using our favorite methods and data structures, such as array, maps, dictionaries and so on. But imagine you have billions of text files and millions of words, can you still fit all of that in memory? Likely not.

How do you write a MapReduce program, so you can do this for such large dataset on Hadoop? Let's take a look.

This is a one-page summary of the MapReduce program that will do that. It may look a little scary in the beginning, but it's actually quite simple. It is like a flow chart that describes the MapReduce program. We have the input files on the far left and the output on the far right, which is exactly what you saw previously. Now we see concretely how the master computer divides up to work, where each piece is a single file. That is, suppose the first file has two lines, now the first line apple, orange, mango becomes one unit of work. And this line is processed by a mapper computer in the map phase, where data and computation are divided up into pieces, and each mapper computer handles one piece independently and in parallel.

Here, each mapper does a very simple thing, which is to turn each word that it sees into a key-value pair. For example, when it sees apple, the mapper outputs apple one, where apple is the key and one is the value. Now you know why the worker is called mapper. The reason is that it maps an input which is a single word, such as apple, into an output that is a key-value pair. Now with many such key-value pairs, if you can scan them quickly, you see that every word occurrence is associated with a 1. This is great because we can exactly get the result if somehow we can group all the four pairs of Apple, 1 together, then we get the result, which is Apple, 4. Similarly if you can group the two mango one together, we get mango 2. So how do we group them.

Fortunately, this is automatically done for you in the shuffling phase. All the key-value pairs that you have got after map phase are automatically sorted by their keys and moved to the reducer for processing. This means all the Apple, 1 will be grouped together, exactly what we want. So for example, the reducer that handles the Apple 1 key-value pairs will see four of them as input. And it can easily add the four ones, and we get our final output.

What you just saw was a simple but complete map reduce program. And this exact same program will also work for large datasets with billions of files and millions of words. In other words, if you want to write a program to handle large datasets on Hadoop the most important thing is to figure out how to express it as a series of map and reduce operations.

The main implementation for the program you just saw is also quite simple. You only need to implement two functions. The first one is the map function, which takes in one line from an input file, breaks that line into individual words, and then output the word and a 1 as a key-value pair. The second function is the reduce function, which will see a list of key-value pairs as input. And all it does is add up the values and output a key-value pair consisting of the same key and the result of the addition One thing to note here is that, in the input, instead of list we see an iterator. The reason for using iterator is so that we don't need to know the number of items beforehand. So we can enter through billions of items without having to worry about allocating enough memory for them.

In this video, we saw a simple but complete reduced program that would work for both tiny datasets and datasets with billions of files.

When and How to Try Hadoop?

In this video we'll talk about when and how to try Hadoop.

We can use Hadoop like a swiss [army] knife since it works well for many kinds of tasks. However, [for] machine learning tasks, you will learn that there may be better solutions, such as Spark. A main reason is that Hadoop's computation is disk based, meaning, for tasks like machine learning when [an] algorithm is iterated through the data many times, it is going to need to read from and write to disk after each iteration. Spark, on the other hand, can keep data in distributed memory to speed up computation.

To write Hadoop programs there are tools that you can use to simplify that. For example, you can write the more high-level Pig scripts that can automatically be converted into MapReduce programs instead of implementing many map and reduce functions from scratch. Also, if your main task is to query data stored on Hadoop, you can use tools such as Hive where you write scripts that are very similar to SQL. We will cover Pig and Hive in upcoming videos.

You can try Hadoop easily. You do not need multiple computers. You can even try it on your own laptop using it as a single machine cluster, where it simultaneously serves as the master, mappers, and reducers.

There's an excellent tutorial from Yahoo that teaches you how to set up Hadoop on your computer. And run the working example in less than 30 minutes.

[[https://developer.yahoo.com/hadoop/tutorial/](https://www.google.com/url?q=https://developer.yahoo.com/hadoop/tutorial/&sa=D&source=editors&ust=1738598889218463&usg=AOvVaw3yWTtUgkK_c1hyiGEVFAXt)]

In academia, it's common for research groups to put together older machines that they have retired to form a home-grown cluster instead of throwing them away.

Today a popular way for companies and research groups to use Hadoop is through cloud platforms like Amazon Web Services, AWS for short, and Microsoft Azure. When you use these platforms you only pay for what you use, such as storage and compute time. They are attractive solutions because the users do not need to maintain the clusters themselves or purchase the machines up front.

In this video we talked about when you may want to use Hadoop and how you can use it.

Scalable Computing: Pig

Why Pig? How to Use It?

In this video we'll talk about Pig, a simpler way to write MapReduce programs.

Pig is a high-level language developed at Yahoo that allows you to write, understand, and maintain MapReduce programs more easily. A Pig program that you have written can automatically be converted into MapReduce programs, so you don't need to implement many map and reduce functions from scratch.

Pig language is called Pig Latin. When using Pig your program becomes a data flow sequence. You can think of it as multiple data transformations, meaning you have your input data then some operation is done on that input data to generate some intermediate data. And Pig performs several such intermediate operations on the intermediate data, eventually producing the results that you want.

There are two primary benefits in using Pig. The first benefit is that you can save a lot of time and effort, because typically a Pig program is a lot shorter than a full MapReduce program. And Pig is quite easy to learn, too. A second advantage is that Pig has a very nice built-in support to perform a sample run on a large dataset. That is, you can test and debug your program much faster, instead of running it against a full dataset, which can take hours.

Since a Pig program is eventually converted into a MapReduce program, it would have the same limits as Hadoop MapReduce, meaning it is still disk-based. So Pig is still mainly suitable for batch processing, and not for real time random read or write of data. And the Pig program could be slower than a MapReduce program that you write from scratch, because Pig may need to figure out a few details, such as what MapReduce function you should convert your Pig script into. However, the trade-off is likely worth it, because you save a lot of time in writing and maintaining a program in the long run.

You can try Pig easily, because Pig is a client-side application, meaning it runs on your computer. There's nothing to install on the Hadoop cluster.

There are two ways to run Pig. One is the local mode, which is great for smaller datasets and can be run on your computer. The program will not be converted into aMapReduce program.

Another mode is MapReduce Mode where your program will be converted into MapReduce program, which can then be run on a Hadoop cluster. This you can set up a single machine cluster using your own computer. You can try both modes easily.

There are three ways to write Pig programs. The first one is as scripts. The second is through a command prompt interactively. We'll look at an example of how to do that in a later video. Pig's interactive shell is called Grunt. And like any interactive programing, it's great for debugging. Grunt has also a very nice feature of code completion. The third way is as embedded program as you may expect, as in an embedded program of a loger code base.

In this video we introduced you to Pig and talked about the benefits of Pig, when you may want to use it, and how you can try it out.

Example Pig Program

In this video we will look at a Pig example program. We will find the highest temperature by year using the grunt interactive shell.

In most Pig programs you start by loading the data using the LOAD command. You specify the data types at the same time. Here for each row of the file you are loading three data values. The first one is year, which is a character rate or string. The second is temperature, which is an integer. The third is air quality, also an integer. You will assign the name records to these loaded data records.

To check the loading is successful, use the DUMP command to print out what has been loaded. Here our dataset is small with only five rows of data. We call each row a tuple or tuple. You can use a DESCRIBE command to check that the schema is correct. Here, we see indeed all three types of data are correct.

Next, we will do some filtering. We want to remove some invalid temperature values and some invalid air quality values. And we name the records after filtering filtered\_records. Again, we use the DUMP command to print out the results. In this example, no records have been filtered.

Recall that our goal is to find the highest yearly temperature. These records you would want to pull together all the temperature for each year. To do that, we will use the GROUP-BY command, just like in SQL. Here we group filter records by year and save the results as grouped\_records.

Let's see what we get using that DUMP command. We see two lines here, the beginning of each line is year, which makes sense, since we wanted to group the records by year. After the year value, we see an ordered collection of tuples with the same year. We call this an ordered collection, a bag. As in we have a bag containing a bunch of records, and we shake the bag so that the records are not ordered in any way.

Now if we use a DESCRIBE command to schema of grouped\_records, we will see that the year at the beginning of each line is given a special name, called group, which is an alias for year that Pig has created for you.

Now we'll see how the year alias can be used. Since we want to get the highest temperature for each year, conceptually what you want is to pull the temperature values highlighted in red here out of all the records for each year, and then take the maximum of those temperature values. So we use the FOREACH command to go through each grouped record. First extracting the year, highlighted in blue here, and then taking the maximum of all temperatures in that year using the max command. Since we only have two years, 1949 and 1950, we will see two records as output when we dump the results. So 1949, temperature 111, 1950, temperature 22.

Here's the whole pig data flow that we just looked at. First we loaded the data records and we filtered the records. Next we grouped them by year. And finally, for each year we got its maximum temperature.

What you just saw was an example run on a tiny dataset. The same program would also work from millions and billions of records, too.

But how to test your program without having to wait for long time? The great news is that you can use the ILLUSTRATE command to generate a small sample dataset from the whole dataset and run your program on the small dataset to quickly check for issues.

Here's an example. We use the ILLUSTRATE command on max\_temp, which you may recall stores the result of our program. Here you see Pig has selected three records out of our full dataset. The records are transformed, step by step, first as records when they are loaded, and then into filtered records, which is the second table, and then grouped records, and finally the results.

There's a lot more to learn about Pig. What we had just looked at was a small part of what Pig has to offer, so I encourage you to try it out.

In this lesson, we looked at Pig, a simpler way to write MapReduce programs. We walked through an example program to highlight some of the key operators in Pig and we discussed how the ILLUSTRATE command can be very helpful in helping us test our Pig program on large datasets.

Scalable Computing: Hive

Overview, and vs Pig

Today we're going to look at Hive.

Hive allows you to use SQL to run a query against large datasets stored on Hadoop. Originally it was developed at Facebook. And very similar to Pig, Hive also can run on a client-side, meaning there's nothing to install on Hadoop cluster. And the language that you use is called HiveQL, so it's very similar to SQL. And same thing, as in Pig, something that you write in using HiveQL, it would get converted into a MapReduce job, which then you can run on a Hadoop cluster.

So let's look at an example of how you would use Hive. So similar to the example you saw for Pig, where you want to find the maximum temperature across all years. So what would you do here?

So first thing is you create a table, as in any SQL, right? So you CREATE TABLE, and here, this is the time where you specify how your data is separated. So here we say the data file is tab separated.

And the second thing that's unique to Hive we don't see in Pig is that you need to load the data. And you use a LOAD DATA command where you will load the data into a local directory of Hive. And after the data is loaded into the table then you can do the query.

So this is exactly the same query that you would use to In the same example that you saw in Pig, so finding the highest temperature by year. So if you look at the syntax here, it's actually very similar, I would say even identical, to what you would write in SQL. You say, SELECT year, the maximum temperature, FROM your records table, and do some filtering on the temperature, grouping by year, so then you get a year for maximum temperature for every year. So you get 1949 111, 1950 22. That's the result. So, and actually, that's it. So that's how you will use Hive. So once you have the data loaded, everything is pretty simple.

And now you might ask, so, well, I can do the same query in Hive, and previously you saw this is the same thing that's done using Pig. So they look somewhat similar. In Pig, you had the data flow sequence, and in Hive, you had the SQL, exactly same SQL you would use in a relational database. So what a difference?

So, here's a very good comparison. So Hive or roughly SQL, we'll say this is a declarative language, so meaning you tell SQL or Hive what you want, but you don't specify how to do it. While in Pig, it's what we call a procedural language. So meaning you specify every single step, so which is why you have the data flow sequence. So every data flow or transformation is one step.

The second difference is that in Pig you can do something we call checkpointing. So that means in the whole pipeline, the whole data flow sequence, after every single step you can actually check what's the output of the data. This is not something Hive or SQL can offer. You only write one long SQL and you cannot really see what's happening in between. You only get the final result in the end.

And the third thing is that when you're using Hive or SQL you need to have some faith in the SQL optimizer because it's a declarative approach. So you would say, I want this but I don't really know how its done. So Hive or SQL will need to figure out a lot of that out.

And the fourth thing very unique to Pig is it allows something we call splitting of the pipeline. So that means, in Pig, you can, let's say, you want to do two kinds of analysis, but the first part of these two analyses are the same. So that means you're going to share the same data flow up to that point and then you split the pipeline, and now each part would be performed or processed differently.

Finally, Pig developers can use their own code. Meaning they can specify different ways of loading data or writing data after each step in the Pig pipeline.

So we've look at a quick example of Hive and also we compare how it's different or similar to Pig

Week 7

Scalable Computing: Spark

Overview

Today we're going to look at Spark.

Spark is a popular distributed computing framework that many people call Hadoop 2.0. It is, in fact, not a modified version of Hadoop, but it was created to address some limitations of Hadoop. Spark is a fast MapReduce-like engine that speeds up computation by storing data in distributed memory across multiple machines. And because of that, a Spark program could get a great speed boost over Hadoop MapReduce, which is disk-based. Spark itself does not come with a backend distributed storage system, so it depends on other existing systems, such as Hadoop and HBase.

When we talk about Hadoop, usually we will use the term Hadoop ecosystem, because Hadoop is in fact a collection of many technologies, and also because many technologies have been built on top of Hadoop ,such as Pig, Hive and HBase.

For Spark we have something similar. Spark SQL is a component of Spark ported from Hive. So it's compatible with Hive data and Hive's query language.

Spark SQL enjoys great speed boost over Hadoop, thanks to Spark's ability to store data in distributed memory.

The Spark project started in 2009 at UC Berkeley and was open sourced in 2010. It became a top-level Apache open source project in 2014. This is great because that means it's here to stay. Most top-level Apache projects have extremely strong community support from industry and academia. Spark is built by over 1,000 developers and people from over 200 companies and organizations such as Amazon, eBay, IBM, NASA, and Yahoo. Spark is now used in large computing clusters with over 1,000 nodes.

A main reason behind the creation of Spark was due to Hadoop MapReduce not being able to efficiently handle iterative computation typically needed in machine learning and graph algorithms. So Spark was created to speed up data sharing, and specifically by storing data across distributed memory.

One misconception that many people have is that they think MapReduce is dead, especially after they have read articles that suggest Google has dumped MapReduce. This is not true for multiple reasons. First, MapReduce is a programming model. Many high-level technologies are actually built on top of it. For example, a Pig program or Hive program are converted into MapReduce programs.

Another reason is that it takes time to completely face our technology that has been in place for years. So while it may very well be that Google wants to eventually use some more of their newer technologies, it's going to take some time. To easily refute article that wrongly claim Google has dumped MapReduce, a Google employee said on Reddit that he was just running some MapReduce jobs a few weeks ago, before the article appeared.

Now let's take a closer look at how Spark speeds up computation. First, let's consider what happens in a case of Hadoop MapReduce. Let's say we want to run a machine learning algorithm that uses gradient descent, which is iterated computation. Since Hadoop is disk-based, in the beginning of each iteration the program would need to first load a data from the disk to do the computation, and then at the end of the iteration write the results back to disk. And this happens for every iteration. So that means there's a lot of disk IO, reading from disk at the beginning of iterations and writing to disk at the end of iterations.

Now let's consider another scenario, where I want to do some queries on a large dataset. Let's say the first query tried to find a subset of data that contains a particular word. And the second query is almost identical to the first query, except that it tries to find a different word. If these two queries are implemented as conventional Hadoop MapReduce programs, the two queries would need to read the full dataset from disk twice, the first read for the first query, and the second read for the second query. Needless to say, this is very expensive.

Now if we use Spark, we can cut down the computation time dramatically by keeping the data that needs to be iterated through many times in distributed memory. That is, for machine learning algorithms that involve many iterations or computations to the data, we can keep the data and intermediate results in memory as shown at the top of the screen instead of pushing them to disk.

Similarly, for the query scenario, we can first load our dataset into distributive memory and then run our queries against this in-memory data. This saves us from having to reload a data from disk for future queries. So not surprisingly, Spark can be quite a lot faster than Hadoop MapReduce, since memory IO is much faster than disk IO.

So how does Spark keep data in memory? Internally, Spark uses a data structure called RDD, short for resilient distributed datasets, to let users explicitly persist into immediate results in memory across multiple computer nodes and to manipulate them using parallel operators that can process these in memory data. To provide fault tolerance, these RDDs can be automatically rebuilt if a machine dies.

Spark now supports several programming languages, including Scala, Java, Python, and R. Spark can also be used interactively from the command prompt.

In this video, we saw an overview of Spark, looked at the main benefit of Spark in that it can store data in distributive memory to avoid disk IO. We also briefly described RDDs, Spark's underlying data structure that enables this.

Example Spark Programs

Today we'll look at two examples of Spark programs.

The first program does log mining, meaning you want to analyze logs, as in computer logs, to identify patterns and discover issues. Let's look at the program in more detail, which is shown on the left.

Our Spark program first loads some log files from the disk. In this case from HDFS. Recall that Spark can load and save data to a variety of data storage systems. And here our text files are stored on HDFS.

Next, in the second line of our program we filtered our lines in the files. So, they're only the ones containing error messages in this case those starting with the word error will remain.

After that in line three, we do some additional processing, splitting the line into two parts and using only the second part.

Next in line four, we called an important function called cache, which explicitly loads all the filtered data lines into distributed memory. This function is important because Spark, by default, does not load data into memory because memory is a scarce resource. And there can be a lot of data that your program may need to use, but not all data is worth keeping in memory. Spark needs your help to decide on what to load into memory.

Like Hadoop, Spark also uses a driver worker model. Meaning there is a driver program that coordinates how a user's program is divided into tasks, and how to distribute them across worker nodes.

The first three lines of code we have seen so far, have been creating and manipulating RDDs on the way. For example, in line one when we first load the data from the disk, we get the base RDD. The second line that does the filtering, creates a transformed RDD which is an intermediate result that we get after filtering. Similarly, this intermediate data is then further processed or transformed by the third line which does the splitting. The fourth line pushes a data into distributed memory. Once in memory, you can query the data quickly.

For example, we want to first count the lines that contain the word foo. Next, we count the lines that contain the word bar. These two queries would complete very quickly, because all the data that we want to sort through is already in memory. So it's possible to finish similar queries in seconds. Even for terabyte scale data.

We mentioned that RDDs can be automatically rebuilt if a machine dies. This fault tolerance is enabled by Sparks ability to keep track of all the transformation that are performed on the base RDD. We call this series of transformation a lineage. This means, if a computer dies, Spark will replay the lineage of transformations to reconstruct the necessary RDDs.

Now, let's look at another example Spark program that does machine learning. Logistic regression is a common machine learning algorithm that does projection. It's also an iterative algorithm in that it uses gradient descent internally to find optimal algorithm parameters. It needs to iterate through the data many times to incrementally search for and hone in to the optimal parameter values.

Recall that Spark has sped up such iterative computation by loading into memory the data that we need to iterate through. And that is exactly what our Spark program does here. In the program shown here, at the end of the very first line, we used to import in a cache method to load the data in memory, which only needs to be done once. Then a few lines down, we used a for loop that loops through the data many times to optimize the vector w. Which is the algorithm's parameters.

So in a nutshell, a key consideration when we write Spark program is to identify the data that we need to iterate through many times, and try and load that into memory to speed things up.

Not surprisingly, Spark can be quite a lot faster than Hadoop MapReduce, up to orders of magnitude faster. Note, however, if you want to only run one iteration of the algorithm. Spark can actually be slower, as seen in the far left on this chart. The reason is that loading data into extruded memory can take time.

In this video we look at two examples of Spark programs and discuss how, by storing data in memory, Spark can significantly speed up computation.

Spark SQL and other Spark Libraries

We'll look at Spark SQL, which is doing queries using Spark with a writing Spark code in Scala, Java, or Python. How do we do that?

For Hadoop, you may recall that instead of writing low level MapReduce programs, we would write out the queries in Hive QL, instead, which is very similar to SQL. Programs written in Hive QL are automatically converted into MapReduce program. However such programs may take time to run seeing as how MapReduce programs are disk-based.

To take advantage of Hive and to speed up computation, Spark offers Spark SQL, which is like Hive but imported to Spark.

In terms of architecture, Spark SQL replaces part of the Hive Architecture, specifically, the MapReduce programming model shown in the bottom of the screen and also the query planning optimizer and execution components.

This screen shows the architecture in Spark SQL. The components in blue are what Spark SQL has introduced or replaced. Note that the very bottom layer, HDFS, is still intact, since this is where data's stored. And neither Spark or Spark SQL comes with a storage system. So they need to rely on distributed storage like HDFS.

For small queries such as doing simple scans through a large text dataset to search for particular word, they can finish very quickly once the data is cached in memory, shown as a green bar here. If we do not keep the data in memory, then the speed of Spark SQL is quite similar to Hive.

For more complex queries Spark is also faster, though the speed of its not as dramatic. In this example it's a few times faster.

There are a number of components in a Spark ecosystem besides Spark SQL. Spark streaming is another popular component designed to handle streaming data, as in click-stream analysis and real time event detection, like spam filtering.

The main idea behind Spark streaming is that it extends Spark. To first divide a streaming data into small batches. Create RDDs for these batches. Keep them in memory. And then run a Spark program pipeline on these in-memory RDDs. Conceptually, this is a very small modification of a standard Spark program that runs on the full data. This is great because we can easily intermix batch and ad hoc queries.

With a large number of machines, Spark streaming can process streaming data at high speed, up to tens of millions of records per second, with low latency.

Another increasingly popular Spark component is the machine learning library, MLlib, for short. It now has a good collection of some of the most popular machine learning algorithms. I encourage you to check them out.

Spark can also process graph data, via its GraphX component. There are some limited algorithms at this point, however there will be more as time passes.

When you think Spark, the ideal case is that you keep all the data that you want to work with, in memory. However, sometimes you may not have enough memory. If that happens, Spark rates really degrade to spill the data to disk. Of course, this means lower speed, but the program will still run.

Spark is now version 2 and is still relatively new. One thing to note is that version 2 has introduced breaking API changes. Meaning this is not backward and compatible with Spark version one programs.

In this video, we look at some of the popular Spark components. Such as Spark SQL, Spark streaming, and graphics.

Scalable Computing: HBase

Overview

In this video, we're going to look at how to handle random read and write of large datasets using HBase. Materials in this video are based in part on the book HBase: The Definitive Guide.

HBase is part of the Hadoop ecosystem, typically run on top of HDFS. It supports random read and write and scales to very large datasets using many machines. HBase is not a relational database and it does not support SQL as a query language. HBase is considered a so-called NoSQL database, meaning it does not use the conventional table storage model of relational databases.

HBase gives us a number of features of relational database, such as data consistency to provide massive scalability. HBase is designed to support billions of rows and millions of columns. For example, it is a database that powers Facebook's messaging platform. HBase is written in Java, and it supports APIs in many languages.

HBase is a column-oriented database, meaning data's stored by columns, basically, instead of by rows. A row conceptually consists on many columns and can be up to millions. Rows form a table and you can look at a row by its unique row key. The row key's like an index. In this space, you can only have one index per table. There's no built-in support for multiple indices, but such support is possible via extensions. Rows are sorted by their row keys lexicographically or alphabetically.

Here's an example of how rows may be stored in each base. Here, we have seven rows. The first column contains the row keys. Row-1, row-10, row-11, row-2 and so on. You will see that row-10 comes before row-2. This is due to the alphabetical sorting order. To bring row-2 to right after row-1, we need to pad row-2 with a 0, so that instead of row-2, we use row-02.

Imagine that HBase is a column-oriented database where data is sorted by column. Column are further grouped into column families to help with data organization and storage optimization. In detail, columns in the same family are stored in the same file on disk called HFile. And we comply compression on all the columns in the same family.

At a first glance, column families is a very nice way to group columns. However, unfortunately HBase supports only a few families due to limitations in its implementation. Family names must be printable and column family names should be defined when a table is created and should not be changed often. To locate a column, you should include its family name. That means there are actually two parts in a column name. The first part is the family name, followed by a colon, then the column qualifier, which can be arbitrary bites.

The actual data is stored in cells. And these cell values are timestamped. You can explicitly set a time stamp. If you don't, the system automatically set it for you. This means you can store multiple versions of the data values over time. The values store in decreasing order, so the most recent value can be retrieved first.

Here's an example of how time evolving data will be stored in a row in HBase as multiple versions. Here we have only one row whose row's key is row 1. And we have three columns. The first column is in green, the data column. We're writing [the first column] to Excel at three timestamps, 3, 6, and 9. While in the second column, we write a value only at timestamp 8, and the third column we only write at timestamp 3. At the bottom of this screen, we show a table view of the same example, where we can more clearly see which timestamp cell values are written.

HBase allocates space only for values that are present. Empty cells do not take up space. This means that if a column is sparse where only a few cells have values, the column would only take up little space.

Now that you are more familiar with how HBase stores data, let’s do an exercise. How would you use HBase to store all webpages in the world? Meaning for each webpage, we would want to store its content, its web links, its images and so on. Also, we want to store all this information over time, so that it can keep a complete record. Now take a minute to think about this, and we'll discuss the answer together.

Have you figured it out? The first thing to think about is, what should a row key be? Remember, the row key must be unique, for webpages to URLs are unique, so we can set it as a row key. So each row will represent one webpage. The next question is, what should the columns and rows be? For each webpage, there is quite a lot of information to store. Such as its HTML content, JavaScript files, images, web links, and so on. So how do we store all of these?

In HBase, we're allowed to have millions of columns. So we can actually have one column for each thing we want to store. For example, we can have one column for HTML, one for JavaScript, one for images and so on. And whenever webpage is updated with new information, we just write to the same row with a new timestamp. So we get to keep all versions of the webpage content, so there you have it.

In this video, we saw an overview of HBase, looked at how it stored data, and also an example of how to use it to store billions of webpages over time.

How HBase Scales Up Storage

In this video we'll look at how HBase scales up storage using a cluster of computers.

Scalability come from HBase ability to automatically divide contiguous range of rows into regions. Each region served by one computer called a region server. When an HBase table is empty in the beginning we'll start with one region. When that region becomes full, meaning we have a lot more rows added to it, we split that region into two regions. When the two regions become too large, we split into more regions and so on.

Here is a picture of how rows are divided up and served by multiple computers. Here the green column on the left is the row key. We have three computers, or region servers, shown in red. Server 1 handled two regions, rows T-Z, and rows A-C. Similarly, Server 2 handles rows I-M, and M-T. So the row key is really a way to figure out the servers that contain the rows that a user is looking for. In other words, the row key is like the key in a hash table. The regions are like the values, and the region servers are like the buckets.  
  
[[https://blog.cloudera.com/how-scaling-really-works-in-apache-hbase/](https://www.google.com/url?q=https://blog.cloudera.com/how-scaling-really-works-in-apache-hbase/&sa=D&source=editors&ust=1738598889233527&usg=AOvVaw0eel0tfRVLSNtz_KqhJSUx)]

In fact, the HBase data table is a sparse distributed per system multiple dimensional map. It's sparse because there are many empty cells. It's distributed because it's spread across multiple computers. It is persistent because it's stored on disk. And it's multidimensional because multiple pieces of information is needed to look up a specific cell value, which includes the row key, column family name, column qualifier, and timestamp.

In this video we looked at how HBase uses multiple region servers to scale up data storage. In a nutshell, HBase is like a gigantic distributed hash table, which spreads data across multiple machines and uses the row key to look up the machine that contains the data needed by the user.

How to use HBase

In this video, we'll look at how to use HBase.

The easiest way to use HBase is to use interactive shell, which can run on your computer without the need of HDFS. We'll show you an example shortly. You can of course also access HBase pragmatically.

To use the interactive shell, you'll first start HBase, and then you launch the shell. You can use the status command to check that HBase is running.

Next, we create an HBase table called testtable with one column family. Recall that column family should be defined when a table is created. Then we start adding data to the table using the put command. Here we add three cell values, the first one added to row one column q1. Look at, we are creating the column q1 as we are adding this new piece of data. Then we add a second and a third value in a similar fashion. Look at all these commands complete very quickly in a few milliseconds.

Using the scan command, we can print out the three values we have just added. Note that the rows are sorted by row keys.

To do a random look up for a row, we use the get command. Here, we'll retrieve one complete row by specifying the table name and the row key. The retrieval also completes very quickly. You can, of course, also retrieve cell, or a particular version of a cell, adding column names, timestamps, and so on, in the get command.

To delete a record, you would use the delete command. Here, we delete a particular cell value. This also completes quickly.

In this video, we will look at how to try out HBase quickly, using this interactive shell.

To Learn more about HBase

In the previous videos we looked at the basics of HBase. In this video, I want to give you some pointers about where to go next if you want to learn more.

HBase was developed based on Google's Bigtable, and was designed for random access. Many people view HBase as the open source version of Google's Bigtable. It's called Bigtable because you're really putting everything in one gigantic table with billions of rows and millions of columns.

For Hadoop and HDFS, they are also based on Google's published work. For example, HDFS was based on Google File System or GFS. And Hadoop MapReduce was based on Google's MapReduce paper published in 2004. If you want to know more about these techniques, I highly recommend that you read those original papers, the link are shown here on your screen.

A lot of people want to know how HBase is different from Relational Database Management Systems. The rule of thumb is that you want to use a technology that best meet your needs. At the very high level HBase may be good for you if you don't know the structure or schema beforehand. HBase is flexible in that it allows you to add millions of columns on the fly. And it's designed to handle sparse data, where many cells may be empty. This is very different from relational databases which are more suited for storing and retrieving whole rows of data.

HBase explicitly keeps multiple versions of the data while relational database do not? A very nice thing about relational data base is that it supports multiple indexes, allowing you to write queries in much more flexible ways.

In general, if you have very large datasets, like in petabyte scale, it would be a lot cheaper to deploy HBase.

There are a lot more to learn about Hbase. For example, there are commands that we have not covered which would allow you to add and retrieve data in batches at high speed, instead of doing that one by one. Also we did not touch on applying cluster management and administration, which can have significant impact on performance. For example HBase region server should typically have more RAM than Hadoop MapReduce node to speed up look up.

If you plan to use HBase to store time [event?, series?] data, you'll want to be careful when designing your row key. For example, you would not want to use timestamp, which are monotonically increasing at the row keys, because they can cause hot spotting, meaning even though you may have hundreds of machines in your cluster, only one of them is used at a time. The reason is that new rows are added to a single region instead of randomly distributed across machines. The links shown on this screen offer multiple solution to have you avoid running into such problems.

[[http://hbase.apache.org/book/rowkey.design.html](https://www.google.com/url?q=http://hbase.apache.org/book/rowkey.design.html&sa=D&source=editors&ust=1738598889236479&usg=AOvVaw1QyXcktcQPb9svk2UwXDJa)]  
[[https://db-engines.com/en/system/Cassandra%3BHBase](https://www.google.com/url?q=https://db-engines.com/en/system/Cassandra%253BHBase&sa=D&source=editors&ust=1738598889236674&usg=AOvVaw1_LnwP7tewySbG1Sq4SDKC)]

In this video we discuss some topics relates to HBase that you may want to study further, such as row key design and a Google Bigtable paper the HBase was based on.

Week 8

Classification

Overview

Today we're going to look at classification, a very common data analytics task. Specifically, we're going to look at the important ingredients of classification such as the data, the model and how to penalize mistakes.

So we're going to start with the common set up of a classification problem. So suppose we want to look at whether someone would like a song or not. So, as in this table where the first column is a song name and the right-hand column is the label. So whether I would like it, not like it or maybe neutral.

So this is a very simple but also a very standard way how classification problem is set up. So on the left is what we call the feature or characteristics about a data. And then on the right-hand side, so that's a label, that means that's what you care about.

So the question is, how do we determine the label for something we have never seen such as the very last song here, Chopin number five. So, will we like it or not based on what we know about a particular user.

So, in any classification problem there are three important ingredients. So the very first one is data. So data often we get the notation of S, which consist of two parts. The first part is x and the other is y, so x is we call it usually the attributes of the data. And usually, we use in the notation of d to signify how many attributes there are. And y is the label that you care about. So in previous example that is the sentiment label of whether I will like or not like our sum. And often we use a subscript ito specify which data instance we are looking at and also a small n to specify the number of data points.

So in the classification problem we also need something called a model, so we'll go into more detail about what we mean by model. But loosely, you can think of it as some way of mapping the data that you have, which is x, into the label that we care about, which is y. And there are many different ways to figure out this mapping and, often, this kind of mapping you can think of it as having some kind of knobs that you can turn or you can change and this is what we call parameters. And, here we say we have parameters a, b, c and so on. And you can have very few parameters, which often is actually a good sign. So that means they're not that many of these knobs that I need to tune or to figure out exact values. Or sometimes you may have a lot of these parameters that you might need to tune, and it can take a long time to do that.

And the third important ingredient of a classification problem is the loss function. So often this may not be discussed as much, but it is actually very important. So the loss function is the way for you to determine how to penalize mistakes, because when you create a model, the model is never perfect, so that means the mapping is not necessarily perfect or may be able to help you determine labels of unseen data perfectly. So you need to know how to penalize this mistakes so you can know how to tune the knobs towards a better number or values that can give you a better model.

So more concretely, here we can explain what data is and also what model and laws function. So start with data here. So using an extension of the example that we've seen before. Here at the bottom we have extended the data to also include the artist name and the length of the song. So now more clearly you can see that the attribute could be song name, it could be the artist name, it could be the song length. And the label here is the same, so that means whether I like it, I don't like or neutral.

So I need to note here about the terminology. So in machine learning, often we use different names to mean the same thing. For example, data example and other name we call it would be data instance. And similarly, for attribute, other names will include feature, dimension, characteristics. And for label sometimes people also call them target attribute or class.

So what is a model? So we mentioned that in machine learning or for classification, it's important that we can model, or a mapping. So at a very high level, a model, you can think of it as a simplified representation of reality. So this is actually a definition of Data Science for Business. So everyday example include maps, which we probably use everyday. So map is also a model because it's an abstraction of our physical world. And as you may guess, there are actually many kinds of models. So imagine when you are driving, you might want a map that benefits you to help give you direction. But sometimes if we want to know the terrain of a city, then you may want to use a satellite view. So, that means there are many models and which one works better depends on your task.

So, similarly a classification problem, you can also have different models. Sometimes one model works better and sometimes another model works better. And more concretely you could think of a model as formula to estimate what you care about. And this formula can come in a variety of different formats. It can be mathematical as just say one formula or it could be rule based, so that means you got a lot of rules or maybe even combination, some formula, some rules and so on.

So the model need to be trained or built, and this is a terminology that you'll hear a lot. So training a classifier what people means is to build a model. And when we say training a classifier what that means is that we need to determine the or the parameters in the model. So again, you may remember there could be very few perimeters which may be a good thing or may be require a lot of perimeters and you may need to spend a little more time to figure out all these value. So analogy to training a model is to determine which map in the real world is a good abstraction.

So previously we looked at data, and also model so the last piece of a classification problem is to the loss function. As you may remember, the loss function is a way to penalize mistakes so that you know how to evaluate a model. The most common loss function is the 0-1 loss function. So the full mathematical formula is here, as shown on the screen. It may look a little scary, but all it does is actually pretty simple. So specifically, we're using identity function. Here I, and it tried to look at the label, y, and whether it matches the predictor label f(x), x is our data, f is our model. So essentially it's just checking if the predicted value or predicted model is the same as a true label that will give a 0. If they do not match, then we get a 1.

So there's a drawback of 0-1 loss function, which is that it look at all different kinds of mistakes as the same. But as you can imagine even in real life, for example, that not all mistakes are the same, some are more costly. For example, if we're not able to catch cancer, then that's a huge cost, so we want to assign a bigger loss value to that.

In this video, we look at classification, specifically the important ingredients, such as data, model, and also the loss function.

Overfitting and Cross Validation

Today we're going to look at how to evaluate models, specifically how to tell if the model is going to generalize the data that we have not seen yet. We're going to look at a phenomenon called over fitting, and how to use the technique called cross-validation to prevent that from happening.

In any classification problem, what we want to do is for the model to generalize the data that we have not seen. So that means when we build a model using known or seen data, then we want the same model to work well for unknown or unseen data. In other words, when we train or build a model, we want to know how to turn the knobs or the parameters in the model. So that it can work really well for the training data, which is another word for seen or known data. Where we train the model or to build the model using those seen unknown data. And we want the same model to work really well for the test data, which is another word for unseen or unknown data. And training data and test data are just a word, train and test in general, is a very common terminology that you're going to hear a lot.

So you might ask, so why do we really need to distinguish between training and test data? It seems like that if I can get a very good model that works well for the training data, then it should ideally also work really well for the test or unknown data. So the main reason is that it's actually very easy to have perfect classification on the training data. So that means you can actually build a perfect classification model that can have 100% accuracy. Why can't it work so well and what's the problem with that?

So the reason we can do it is because you can imagine, you can write millions of rules, and that would allow you to zero in to this very specific data point in your training dataset. So that means ideally you can just remember all the rules and then also remember the label, so next time you send an other data point, that's exactly like your training data point. Then you can just recall, it's essentially like looking up the labels. However, these millions of rules that you've written, they may actually not generalize to the test data. So meaning, you may have so many rules, actually you may not even find any data points that you have not seen that, that would fit those rules.

So when your model works really well for training data but very poorly for test data, then what we call is that your model is over fitting. So this is the problem obviously because you ideally, what you want is your model to work well for things that you don't know yet. So this is a prediction that all we care about. So how do we avoid over fitting?

Cross-validation is a common technique to prevent over fitting. So here we're looking at an example of how to do five-fold cross-validation. And specifically, we're looking at only one iteration of cross-validation. So in cross-validation, we will divide the data into folds or parts. For example, in the first row here you'll see that we divided the data into five parts or five folds. And we'll use four folds or four parts as the training data. Which means we're going to use these four parts to train or to build our model to figure out its parameters. And once the model is built, then we evaluate it on that test set. Which is a remaining part, or remaining fold. And we'll get the accuracy for this fold. And we'll do the same thing for the second fold. So meaning we'll still use four parts as training dataset. But here you'll noticed that we shift the test set to the second part. And again, we get accuracy for this fold. And similarly, we move to a third fold and so on. And after doing all five folds, then we have five accuracies. And what we do is we then take the average, then that becomes our cross-validation error.

And of course, as we mentioned this is only one iterations. So the accuracy that you get will depend on how you divide the data. So that means ideally, you want to do at least a few runs, and the more runs, the better, and to compute the average.

So in other words, the exact steps that you need to do is to divide your data into n parts. You use one part as a test set or hold out set. And then you would train the data on the remaining n-1 parts, what we call the training set. And after that, we compute a test error on the test set. And we do step one to four n times, one for each of the n parts. And after that, we compute average test error over all folds.

So when doing cross-validation, we need to determine how many folds are we want to use, and the test set size of course depends on the number as well. So test sets of size equals the number item n divided by the number of folds, K. So K = 10 is most common. So in other words, ten-fold cross-validation is most common.

There's a variation of cross-validation called leave-one-out cross-validation. So the very fancy way to say essentially setting the test set size to one. And this is a great way essentially using every single data point to evaluate your model once. But as we mentioned, this is pretty expensive because then exactly you need to divide the data into n parts or to the training n times. So often, the recommended number to use is 10 because it's the most common. And using other numbers generally, we don't really recommend. For example, if we use K = 9, then people will say, why not use K = 10, so for that reason, it's good to stick with common numbers.

In this video, we looked at how to evaluate models, why over fitting will occur, and how to use cross-validation to prevent that from happening?

K-NN

Today we're going to look at a simple classifier called a k-NN, which stands for k-Nearest-Neighbor.

So k-NN, or k-Nearest-Neighbor, is one of the most popular classifiers because it is very simple to understand, and in practice, can also be very effective. It's very easy to understand, so you can use just one image to explain it.

So what's shown here is we want to do a classification of whether someone would like whisky or not. And for each data point, here is a person and we are plotting every single person on a scatterplot. And for each person, we look at two attributes or the variables, the person's age, and also let's say the bank balance, so whether the person's wealthy or not. And suppose we have people who don't like whiskey as a dot and the ones who like whiskey as white crosses. And now we have a new person coming in and we don't know that person whether he or she likes whisky. So that's the one in the red dotted circle here.

So in k-NN, what we do is we would look at the nearest neighbor of that unknown. So that means we will go around and we'll find who are the closest to this unknown. In this case, we say we look at only three neighbors, so k=3, so our three nearest neighbors. So here the neighbors are highlighted in blue. And we look at the label, so one person doesn't like whiskey and the other two like whiskey. And what we do is we take the majority vote. So that means out of the three, two of them like and one does not like, so the like people wins. And so we have to label the unknown is also like whisky, so and actually that's it.

So you might say, well, k-NN is so simple, so why should we use it? So that actually not very well. Actually, it can work really well, and if you like to listen to music, the Pandora service uses it or has used k-NN.

And so we mentioned before, actually simple models are good. So it's simple, meaning it has very few parameters to tune and in practice that also means likely very fast to train and very fast to run. So simple and effective is great. And then if it's complex but it's still effective, then we might say it's probably still okay. So it will take more time to train, more parameters to figure out, but as long as it's good or effective, then we will still accept it. So the case that we want to avoid is where the model is very complex, so having many parameters. And if it's still not very effective, then those are the case that we want to avoid. So simple, effective like k-NN, if it really works well for your problem, you should actually try it.

So even though k-NN is pretty simple, there's still a number of important points to note. Specifically, there are two main parameters you need to figure out. So first is the number of neighbors, k. So in the previous example, we say k=3. But you may say, so why not 4, why not 5, and so on. And similarly in k-NN, we need to figure out the distance function to use to determine which neighbor is the nearest. So again, there are many different variations of distance functions that we could have used.

So let’s look at a few different scenarios. So suppose k and distance function d are both fixed. Then what do we need to figure out, or what do we need to learn, and also how do we learn them? So that's scenario number one.

In the second scenario, what if distance function is fixed but you're allowed to change k? So that means you can play with the number of neighbors that you want, like three, four, and so on. So in this case, what are the things to learn and how to learn them?

So for the first scenario, actually there's nothing to learn because everything is figured out. So that means for any classifiers, whenever all these parameters are given to you then there's not really anything you need to do.

On the second scenario, if distance function is fixed but you can change k, then you will need to figure out what k is. Now the question is how do we do that?

So actually you already know how, using cross-validation to find the best k in k-NN. So the value of k really matters. For example, on the left, you see we're using 15 neighbors or 15-NN. You'll see the decision boundary is pretty smooth, dividing the two classes, orange and blue. So which also means likely it's going to generalize well to unknown data.

On the other hand, if we use one nearest neighbor, or 1-NN, then likely we're overfitting to the data. So you'll see a lot of islands, blue islands to be specific, in the orange area. So this is overfitting in that it might work really well for your training data, data that you've seen so far as shown on the screen, but it may not really generalize well to unseen data.

So now to our third scenario, suppose k is fixed, but you can change the distance function, then what do we need to figure out? So naturally, it's a distance function. As we mentioned before, there are actually a lot of distance functions you may try. And you may want to start with some more common ones like Euclidean distance, Manhattan distance, and go beyond to other techniques. So often, the general rule of thumb, as we mentioned before, is start with one that are more common and see how well it works. If it doesn't work well, then you try the more sophisticated or more complex ones.

The advantage of k-NN is that it's relatively easy to figure out, so very little learning, and it's only k and the distance function. And in practice it's quite powerful, so it also has theoretical guarantee as well. So it doesn't require a lot of research going into k-NN to figure out why it works so well.

So there's one caveat, which is that it's actually computationally quite expensive. So the reason is that you need to determine which neighbors are the closest. And often it depends on where the data point that you are querying relative to the data points that you have. So that means you need to do a lot of querying during test time.

If you want to learn a lot more about k-NN, I recommend you to read the book chapter in the book, Elements of Statistical Learning.

In this video, we looked at a simple classifier called k-NN, and we looked at that even for this simple classifier there are still a few things to figure out. Such as parameters like k, how many neighbors to look at, and also distance function to figure out.

Decision Tree

Today we're going to continue with Classification. And we're going to look at Decision Tree. So that means, how to build a tree that can help you make decision?

Decision Tree is actually a category of methods. And in general, when you're building a tree, that means you want to build a tree of rules. And the tree determines the sequence of rules that you want to check, in order to make decisions of the labels for data instance.

For example, let's say you are a little kid, and you want to know whether you want to play outside or not. So, you may use a decision tree to figure that out. For example, you either want to play or not play, and the first rule you may want to check is the temperature or the weather outside, so, such as whether it's sunny, overcast, or rainy. If it's sunny, then the next rule you may want to check is, the humidity. So, is it dry enough? Let's say, under 70. If it's dry enough, then you would decide to play. If it's not dry enough, over 70, then you do not play. So that means we'll pass the data instance through the tree, starting from the top. And then, you go through a set of rules, and the final outcome or final label is the majority class at the bottom.

So, every single decision point in a tree is called, a tree node. And the final decision tree, what that means, the points at the bottom, are the leaf nodes.

Before we go into the details of the Decision Tree, I highly recommend that you take a few minutes, to go through the interactive demo. To look at how a tree is built from data, to distinguish homes in New York from homes in San Francisco. And after that, we'll come back and look at the details of what you need to pay attention to, when building a tree.

[[http://www.r2d3.us/visual-intro-to-machine-learning-part-1/](https://www.google.com/url?q=http://www.r2d3.us/visual-intro-to-machine-learning-part-1/&sa=D&source=editors&ust=1738598889246005&usg=AOvVaw1sMzN6cUPcsMLS2J4-zkGl)]

So, like the case of k-NN when you are building decision tree, you have to figure out what are the things to learn, how to learn them? And also, to look at how the cross validation would play a role in the learning process.

So, for tree, the things to learn is what we call, the tree structure. So that means, how many branches there are? What's the fan out, and so on? And the way to learn it is to greedily minimize the overall classification loss. So, that means, how they optimize for accuracy.

By greedy, we mean, we want to make the best decision that we can at every single point. So that means, we want to determine the best next steps, when we are building the tree from the beginning, starting from the root node. So, in the beginning only have a root node. So, there's no branches. And at that point we want to know how, many branches we want to have?

For example, in this example on the right we say, I will look at outside weather. So is it sunny overcast and rainy? So in this case it’s pretty easy, in that you already know the three possibilities. But let's say, for humidity, you will need to determine what's the sweet point, so meaning should we split at 70, or should we split at 60, and so on?

So, every single time we add a tree node or a decision point, we keep it as is, that means we do not keep track, back track, and we will keep the process built already as is. And the decision that we make is to just figure out what's the best next steps. So in other words, when we're optimizing something greedily, we're only looking at what's the best local decision. So that means, we don't globally consider all possibilities. And the reason for doing greedy learning is, so that we can cut down the training type. And also, in practice this strategy, actually really work pretty well, giving a very good tree with very good accuracies.

And the role for cross-validation here is to help us to figure out the best size tree, and also what's the best split point, and so on.

When building a tree, there's a few important considerations. The first one is, how to determine the best split point of a chosen attribute. Such as for humidity, should we split of 70 or should we split on 60? And second consideration is, which attribute to split on? So that means, for a decision node, which attribute should we be using? And the third one is, when do we stop? So, as you may know, we can actually keep splitting until we have enough rules to really classify every single point in the training dataset, with 100% accuracy. But that's not really what we want, because that would be over fitting the training data, so that the tree would not generalize. And the fourth piece is, how do we do cross-validation during the decision tree training?

So, we first look at the first consideration, which is choosing the split point. So, to choose a split point we need to figure out first, what is the type of the attribute? So, there are three primary types of attributes, categorical, ordinal, and continuous. A very interesting thing about decision tree is actually, no matter which kind of attributes, we would create branches that are very similar.

For example for a categorical attribute, such as looking at the genre of songs, rock, classical, and pop. So, this is the simplest case. Because we can just have one branch for each of these categories.

And for ordinal attributes, such as achievements, platinum, gold, and silver. So, here we have order, but in the decision tree, actually we don't really care about the ordering. So that means, in this case we just have one branch for each of the achievement types.

For continuous attribute, what we do, would actually determining the split point. But once we determine a split point, then everything looks the same, as in the case categorical attribute and ordinal attribute. For example, here we can say, we look at the song duration. So, we can say, well, that's split at 500 seconds. So, if it's shorter than 500 seconds we say, it's a short song, if it's longer than that we say, it's a long song.

So, at the bottom you will see that, no matter which type of attribute you are looking at, that you will still have all the different branches, and they all look pretty similar. This is actually a nice thing about this tree, because that means it can handle all types of attributes in the same way.

To evaluate the quality of a split point, we will use a loss function. Remember, a loss function is a way to measure the penalized mistakes.

So let's say, we only have two branches at a particular decision point left and right. Here what we do is to say, if we are splitting a particular value for attribute d, then we would measure the mistakes on the left branch and the mistake on the right branch. And ideally, we want to pick a split point, so that these total number of error or the total loss is smallest as possible.

And there's different ways I can measure mistakes. For example, we can use misclassification rate, can use expected loss, or cross-entropy.

So previously, we talked about how to evaluate an attribute's split point, but we haven't talked about how to choose that attribute. So, one common way to choose an attribute is to pick the attribute that gives you the maximum improvement in training loss. Another way is to pick the attribute that gives you the highest information gain. So, intuition is that, an attribute with the highest information gain would help you the most rapidly describe an instance. So, in other words, you pick an attribute, it can help you reduce the uncertainty in your decision as much as possible. That's probably a very good attribute. Which also means that, by picking the ones with the highest information gain, likely your tree is going to be much shallower.

So for an excellent refresher on information gain, I highly recommend you to look at the lecture slides from a Carnegie Mellon class. The link is shown here.

[Excellent refresher on information gain. PDF page 7 to 21  
[http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-s06/www/DTs.pdf](https://www.google.com/url?q=http://www.cs.cmu.edu/afs/cs.cmu.edu/academic/class/15381-s06/www/DTs.pdf&sa=D&source=editors&ust=1738598889248687&usg=AOvVaw0CA5DgWn-SpNFSjkGV5Zbs)]

Remember that we can keep splitting or creating more nodes in a tree, until we can get 100% training accuracy. But that is not really what we want, because then we have to tree that overfits our training data. So, that means we want to stop early on, so that we have a more shallow tree that would generalize well to unseen data.

So, there are a few strategies to determine when to stop. So one is to look at the purity of the leaf nodes. For example, you can stop obviously, when all the points in a leaf node belongs to the same class. Or in another scenario, where you have two classes, but those points in those two classes completely overlap. So that means you don't necessarily know how to split anymore, and then you just output the majority class at this leaf as the label.

You can also stop the splitting based on the number of points. So, let's say, if a node contains points fewer than the threshold, then you can also stop. Also, using the notion of purity, you can say that as long as the node purity is higher than a certain threshold, you will stop. Another way is to look at the overall training loss in the tree that you have built so far. So you can say, if the overall training loss is still above a certain threshold, or it doesn't really improve, then you stop the splitting.

So, in a decision tree, you learn that there are quite a few things you need to figure out. And actually, there are different ways to differentiate between these different kinds of parameters or different things I need to figure out. So, this notion of model parameters and also hyper-parameters. So, for hyper-parameters, in the example of k-NN, k would be the hyper parameter, or the similarity function, too.

So, these are the things that you may not really know what they are until you try them. So, for decision tree, it will be the number of nodes in a tree.

In decision tree there are two types of parameters, one we call model parameters or just parameters, and the other one we call hyper-parameters. Hyper-parameters are probably more familiar to you. For k-NN, k and also similarity function, hyper-parameters. So, those are parameters that you would need to try to figure out. And for decision tree, the number of nodes in the tree would be a parameter. For example, you can say, I want to stop the splitting once the tree has certain number of nodes. So, for this, you don't really know, unless you really tried it.

And to determine these hyper-parameters, you can use cross-validation. Another way to do it is, to do something we call grid search. So, grid search is a very fancy was to say essentially trying all combinations, so you're walking through the grid of all combinations. And, of course, that's very expensive. So, which is why there's also a variation of it, we call random search. So that's also random. Very fancy way to say that is actually, you're only trying our subset of the combinations.

[[http://scikit-learn.org/stable/modules/grid\_search.html](https://www.google.com/url?q=http://scikit-learn.org/stable/modules/grid_search.html&sa=D&source=editors&ust=1738598889250167&usg=AOvVaw1xbllK7aqLbIaIfzcA1y3S)]

Another parameter is, we call, model parameter. So that means these are things that you can learn, estimate it or compute it directly from data. For the case of decision tree, this could be the attribute you want to split on, or the split point for an attribute.

The reason we say that these can be directly computed is because, let's say we use the entropy based method, then we can just compute those without having to try all possibilities.

Overall, for decision tree, they are quite easy to implement. Very interpretable, because you can see all the rules in the tree. And also, it's very fast during test times. So meaning, once a tree is built, then you can pass data instances through all sets of rules very quickly. And also a very nice thing is that, it can work with different kinds of attributes, ordinal, categorical, continuous, and so on. And in practice, it also works pretty well.

Some caveats though, is sometimes people say, it's too basic, because it's a set of rules, but in practice, it's actually okay as long as they work really well.

Building trees can be expensive, because there are a lot of things you need consider. Say to split point, which attribute to pick, and so on. And cross-validation can be tricky, because you may do something we call node-level cross-validation. So, at every stage of building the tree, you need to figure out different parameters, and also you need to do cross-validation at those different node levels.

In this video, we looked at a very popular classification model, decision tree, and also looked at some of the important considerations, such as which attribute to split on, and also split point, and so on.

Visualizing Classification

ROC, AUC, Confusion Matrix

Today we're going to look at multiple ways to visualize classification results such ROC, AUC, and Confusion Matrix.

A common way to visualize classification result is to use a confusion matrix. So the name is confusing, but what it does is actually very simple. So specifically it's a tabulation of all the instances that your prediction algorithm has got it right and got it wrong.

So in this example we are looking at a classification problem with three classes, Cat, Dog and Rabbit. And the rows here are the actual class, so that means it's a true label. And for the rows here are the predicted class, so same number of classes cat, dog, and rabbit. So the cells in the matrix show you exactly how many of the data instances are correctly classified, and how many of them are wrongly classified.

For example, the first row so the true label is cat, so there are eight of them. And in the cell we can see that five of them are correctly classified and three are misclassified. So similarly for the dog class, two are misclassified as cat and one misclassified as rabbit. So that means all these cell values they're exactly showing us how many are correct and how many are wrong.

So when you have a large number of classes, it's really beneficial to visualize the matrix. For example, at the top we have a lot of number, so we just look at it at a glance, it's very hard to spot any patterns. But on the other hand if you visualize this matrix mapping color value to the numerical value, let's say darker means higher value. That's really easy to see that the diagonal value is pretty dark. So that means that is what you want, because that means the predicted class is actually matching the true class. Also at the same time, you'll see row number 5 is also not that light, so that means there's quite a bit of misclassification going on there. And similarly, you will see that row number 8, column number 3, that's very dark. So that means the digit 8 is highly misclassified and very often as the digit number 3. And you would think about it, actually they look pretty similar, so which might explain the misclassification.

When you're analyzing any kind of classification result, it's very important to understand or to tell people what you mean by positive. So positive often what means the things that you care about in your classification. For example, you are doing cancer detection, positive often means someone having cancer. But often, it's not really that clear. And the drawback of not cleanly stating what positive is is that you can actually get completely opposite interpretation.

For example, if someone says I have a very high true positive rate and what you are interpreting is actually the opposite, or the class what they mean is actually not the class that you want to mean, then you can have different interpretations. So instead of saying you have very high accuracy, you actually interpret it as very low accuracy.

And also, similarly, a lot of terminology actually defined based on positive. For example, there is true positive, false positive. And also there are other measures, like specificity, recall, and so on. So all those are dependent on the definition of positive. So that's extremely important to declare that very clearly.

Another way to visualize classification result is to using the ROC curve, which stands for Receiver Operating Characteristic. So what does that mean? So we look at an example.

Here's the ROC curve of the classification result of a malware detection problem. Here positive class means malware, and negative class is benign file, such as Microsoftword.exe. And in an ROC curve, the horizontal axis is false positive rate, or false alarm. So that means good things get misclassified as bad. And vertical axis is the true positive rate. So that means bad file correctly classified as as bad. So in this example, the curve is cutting through an orange point, which says that there's 85% true positive rate at 1% false alarm rate. So I did a scenario for any classification problem In terms of ROC curve would be to hit the top left corner. So that means there's no false alarm and all the positive things get correctly classified.

So let me state the curve the more of it look like a right angle curve the better. So in this example, we see that it's quite close to a right angle, but not quite yet. So ROC curve actually can give you a series of numbers. So at any point of the curve you can look at a trade up between the false alarm and true positive rate.

Sometimes maybe you are running out of space and you only want one number to summarize the whole ROC curve, then what do you do? A very common way to measure that is using AUC or Area Under the Curve. So very literally it means taking the area under the ROC curve, which is the green area here.

So there's some important point to know about when using AUC. So for example, if someone tell you that there's a fantastic machine learning algorithm, and it achieves a 0.9 AUC score out of the maximum 1.0, then what do you think? Would you say that this is a great algorithm because it's very close to 1.0? Or maybe it's not.

So you should be very careful when you use AUC. For example, here we're showing two curves which have the same number, and you see 0.9. While on the left here, we see that to achieve a 1 True Positive Rate, we need to give up a lot of False Positive Rate. You can only get it when you have a 0.1 False Positive Rate. While on the right we have 0 false positive, but you can only get up to 0.9 true positive. So which one is better?

It actually depends on your application. For example, if you are doing cancer detection then probably 0 false positive would be the best. Because that means you are not misdiagnosing anyone who actually don't have cancer and you mistakenly call them to having cancer.

So in this video, we look at multiple important ways of visualizing classification results. ROC, AUC, and Confusion Matrix.

Week 9

Introducing Clustering

K-means, Hierarchical Clustering, DBSCAN

Hi everyone, in previous videos you learned about classification, and today we're going to learn about clustering another very popular way to make sense of data. We start with an overview of clustering, and then we'll dive deeper into three very popular techniques. K-means, hierarchical clustering and DBSCAN.

So clustering is what we call the most common type of unsupervised learning. And high-level idea is to group similar things together. So it´s called unsupervised, because the model that we learned is through examples that do not have any labels unlike classification where we have some known data with labels. And our task area was to figure out the label for the unknown data. So if a clustering we do not have any label data at all.

So we can find a lot of application of clustering. For example, in health care we want to group patients together so then we can apply similar treatment to them, or in medical imaging we may want to identify similar types of tissues. Or for a text document, we may want to do something called topic modeling, so figuring out what are the topics of documents, and also maybe, some common words in each document cluster. Or in market segmenting, we may want to group consumers, or people together, so that we know how to apply similar marketing strategies on them.

So in this video you're going to be learning three clustering techniques. So these are what I would consider the most common, and the techniques that you would need to know. So the first one is K-Means, the second is hierarchical Clustering, and the third is DBSCAN. Of course there's a lot more that you would want to learn, but these are the most fundamental ones that we would want to learn about it.

So we start with K-means and a lot of people say it's the simplest techniques, cuz it's simplest to understand, and also very easy to implement and is scalable. And instead of going through the steps of the algorithm one by one, we'll first look at a demo. And this is a demo in D3 actually, so we're using techniques that you learned about in the D3 videos.

[[http://tech.nitoyon.com/en/blog/2013/11/07/k-means/](https://www.google.com/url?q=http://tech.nitoyon.com/en/blog/2013/11/07/k-means/&sa=D&source=editors&ust=1738598889255511&usg=AOvVaw3PD4ozHJAGCk3BP270bgc0)]

So in this demo we have the data points in white, and 5 clusters automatically initialized randomly by K means. And they're assigned the different colors. And what K means would do is that it would assign the data points to the center that is closest to. So these are centers or means which is why we have name K-means. And once we assigned that the data points to the clusters, then we will recompute the means. So that means we are shifting the mean to its new location.

And now after the centers have been moved, then we can do the reassignment. So now we can shift the data point to the closest mean that it is. And then we compute the new means again, and do the new assignment. New means, new assignments. And until everything settles, so that means all the assignments to do not change anymore, and then we stop the algorithm. As you can see, the assignment is pretty easy to understand and also the algorithm itself is very conceptually simple.

So, to summarize, what we saw there is that we will first tell k-means the number of customers that we want, and then we will randomly initialize the cluster centers or means. And then it will assign an item to the cluster where the mean is closest to. And after that we'll recompute or update the new means over all the clusters. And it will keep doing the above steps until the assignments do not change.

So k-means is very simple, so what's the catch? So why do people like to use it? And what are the things that we need to be careful about?

So one thing that we need to be really careful about is, how to choose k, and in fact, it is a pretty hard problem to solve. And the best way to determine k is to evaluate your algorithm with real data, because depending on the different data, the number of clusters or k can vary.

And then another thing is that K-means is what we call locally optimal. So meaning, if you initialize the algorithm with different clusters, then you may have different results. So the way to fix it is to run the algorithm different times, or few times, and then you will get different clusters. And the more times that the cluster assignment agreed, then that's probably the better assignments that you can get.

Something to note, though, is that the algorithm can depend on the cluster assignment. So that means if you have a bad starting point, that may require a lot more iteration to run.

A good thing about K-means is it can work relatively well for large datasets. The time complexity is big O (d n log n). So n is number of items and d is dimension. So usually dimension is not very big, and the number of data points n is a lot bigger than the number of clusters k.

So what you saw previously is K-means, so one of the simplest technique. Another very popular way to learn clustering is through hierarchical clustering. So high-level idea is very simple is to build a tree or hierarchy of clusters.

For example on the left, there you see how we form a tree, starting from just the data points, A, B, C, D, E and F in black.

So this is iterative process. So in the beginning, we'll first determine which data points are grouped, for example, data point A, and C, we will first group them because they are closest together. And similarly for B and E, we'll also group them. And next, we'll want to determine where to group D and F. So for D it's now closest to the cluster form by B and E, so we'll group them into cluster three. So you can follow this process iteratively to create a whole hierarchy of clusters, which usually is visualized as a dendrogram, which is shown on the right.

So you'll notice that horizontally we have all the data point laid out, and then vertically is the clustering distance. I mean, how far the data points are apart or our data points between a data point and a cluster. And also horizontally you will see the cut point, so that means if we cut at a certain distance, how many clusters would you get? So you will see that we can almost get any number of clusters that we want. All the way down to the number of clusters equaling the number of data points or having one gigantic cluster. So which is why clustering often people will say is, you can get any kind of number of clusters. You can all the way get from just one cluster everything put together, or you can have two clusters, three clusters and so on. Which also means that the number of clusters really depend on the problem that you solve. And arguably, any number of clusters you would essentially say is correct, if it gives you the right results.

Since we need to merge data points, or data points with clusters we need to determine what distance function to use. There are a few ways, one is single linkage, the other is complete linkage, and also in between, average linkage.

In the single linkage we merge two clusters who closest members have the smallest distance. For example in the example shown here, we have cluster 1, 2, cluster 3, 4, cluster 5, 6 and so on. And we merged cluster 1, 2 with cluster 3, 4, because the distance between data point 2 and 3 is smaller than the distance between data point 2 and 6. In other words, the similarity determined here in single linkage scenario is equal to the similarity of the cluster's most similar members.

In a complete linkage scenario we merge two clusters such that the resulting cluster has the smallest diameter. So an example shown here, again we have cluster 1, 2, cluster 3, 4, cluster 5, 6. And we merged the cluster 1, 2 with cluster 5, 6 because the distance between data point 1 and 6 is smaller than the distance between data point 1 and 4. So in other words, in the complete linkage scenario the similarity of two clusters is determined by the similarity of the clusters' most dissimilar members.

And for average linkage, we will be going for in-between scenario. And people like to use average linkage, because it's computationally more efficient for then instead of looking at a lot of pairwise distances, we only need to look at the distance between the center of clusters.

A very popular way to visualize hierarchical clustering result is using cluster dendrogram. For example n the left here we show the whole end Dendrogram as a tree. So you will notice that this is not the most space efficient method, which is why we also have radial clustering Dendrogram shown on the right, where here you can see the spacial look [?] can pack a lot more points onto the screen, while we can still see the hierarchy.

So hierarchical clustering is very easy to visualize, and very easy to understand. Unfortunately, it's not very scalable. So the time complexity is between O(n^2) to O(n^3). So that means if we have millions of data points, then it's not very scalable.

However, it's great for understanding concepts because we can see every single step in the clustering process. We can see how data points get merged and then points get merged with clusters and so on.

The final method we're going to look at is called DBSCAN, which stands for density-based spatial clustering with noise. It's a great technique to learn because it received the test-of-time award at the top data mining conference KDD 2014, which an extremely prestigious award. Meaning, the technique stands the test of time, and it still have impact today as it was designed over ten years ago.

A great thing about DBSCAN is that it only needs two parameters. One is radius, and the other is the minimum number of points needed to form a cluster.

So how does DBSCAN work? We will look at simple example.

So suppose every point here is all the point here shown on screen is one data point, and initially we do not have any cluster. So what DBSCAN would do is it would start with some random data point. And then for each data point let's say it's starting from data point A. Then we draw a circle which is why we need the radius parameter, and within that radius we look at how many points there are. If the number of points exceeds the minimum number of points considered in the algorithm, in this case let's say 4. Then we'll order data points in the circle to your initial cluster. So that means if, say I assign the color red to the data point A, then now all the four points in that radius would also be colored red. And once you have all these four points added to the red cluster, then you can repeat for every single point that you haven't colored yet. For example, you can move to the point to the left of point A, and you do a similar procedure. So that means now you again draw a circle with radius epsilon. And again, you count how many points there are, in this case, there are five points in total, and you will color all those points red. So you follow this same procedure until you exhaust all the points, or if you say I cannot reach any more points anymore, then what do you do? You start a new point that you haven't considered. For example, point N, then we start a new cluster, and you color it blue, let's say.

So the benefit about DBSCAN is it can find clusters that are of irregular shape. For example, on the right, there you can see there is one roughly circular or spherical blue cluster, and then there's an elongated red cluster.

Also, it's able to detect outlier points too. For example, you see there are datapoints that are pretty far away from those two main clusters, and you can think of those as clusters by themself or outliers.

So to understand DBSCAN, I highly recommend that you look at an interactive demo. The link is shown here.

[[https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/](https://www.google.com/url?q=https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/&sa=D&source=editors&ust=1738598889259020&usg=AOvVaw2lJgLA6G5P835BWTBdRfvd)]

In the demo you can see how DBSCAN iteratively at points to cluster. It starts with the mouth region for example. And once it's exhausted all the points, then we start a new cluster in blue. And you see, it gradually colors more, and more datapoints.

In this video we learned about three very popular clustering techniques. K-means, hierarchical clustering, and DBSCAN. They are all very easy to understand and to apply, so they can be very effective first techniques to try, before trying the other more complex techniques.

Visualizing Clusters

In the previous video you looked at a few clustering algorithms. Today we're going to look at how to visualize the results of clustering.

Since clustering is very popular, so there are already a lot of tools that you can use. For example, in D3 there are few built-in techniques such as the sunburst chart, bubble chart and hierarchical graph layout, and also tree map. So, if you have text clusters or topics, where a topic is determined by the strength of participation in that topic, then you can visualize this data, this clustering information, as a matrix, as in the termite work by Jeff Heer and his students.

So here, more participation is visualized by a larger and darker circle. So for example, in topic 17, you can see that the words communities, social, network, and so on, is highly associated with that topic.

So whenever we're visualizing data as a matrix, often the ordering is very important. For example, on the left, if the participation of working topics are not clusters or order, then you may not see any pattern. But on the other hand, if you use a technique called seriation, that means your reordering the word and topic by their co-location, then you can group the participation and word together so that the pattern becomes much clearer. For example, now for the topic number 17, in blue you'll see all the work grouped together.

So if you are doing clustering for graph data, then you can use color which is a very common to distinguish between the different cluster of nodes in the graph. And also, similarly, you can extend this idea to also use convex hulls. That means you're using the boundary and also the spatial location to visualize not only community, but also subgroups within it.

And similar to topics of text, you can also visualize graph data as a matrix. And, in this case, we are visualizing the adjacency matrix of the graph. And, similar to the termite example, ordering is also very important. So by appropriate ordering, you can see the graph community surface in the visualization as in this example below.

In this video, we looked at a few examples of how to visualize clustering result, such as for text data, for graph data, and so on.

Weeks 10 - 12

Graph Analytics

How to Represent and Store Graphs

Hi everyone, welcome back. Today we're going to look at graphs, or network data.

So graphs are everywhere. For example, for the Internet is a 50 billion web page network, where each web page is a node in the network, and an edge is a web link connecting to web pages. Another example is the Facebook social network, where it has over 2 billion users where each node in the graph is a user and each edge is a friendship link.

There are actually a lot more example like these. For example, Twitter is a who-follows-whom graph, Amazon is a who-buys-what graph. So for Amazon we normally don't think of it as a graph but if you can think about how to connect a customer with the product that has purchased, then we'll have a who-buys-what graph. In other domains, like telecommunication, we can also have a graph. For example, we can have a cellphone network, who-calls-whom, and in biology we'll have protein-protein interaction networks.

To analyze graphs, we first need to know how to represent them, conceptually, visually and programmatically. The most natural way to understand a graph is as a node link diagram, as shown at the top left corner. Here we have four nodes, the edges connecting them may be weighted. And we usually use edge thickness to encode edge weight. A thicker edge means heavier weight, and the edges may also be directed. And we usually add an arrow to the edge to indicate the edge direction.

An adjacency list is another way to represent a graph, where we have the nodes as the columns and also the rows. For example, here we see that there's an edge going from 1 to 3, so then for row number 1 we have the number 3 at the cell 1, 3. So meaning there's an edge of weight three connecting node one to a target node, node 3. So you notice that this is not a very space efficient way to store a graph because you will also need to store a lot of zeros, so zeros here means there's no edge.

So a more compact way would be like an adjacency list, where for each node we only keep the target node that is connected to. For example, at the top right here we see for node 1 is only connected to node 2 and 3 so only store node ID 2 and 3.

So the adjacency list is compact but it's not the most common distribution format, the most common is actually an edge list. Where each edge is represented at a row in the file, and we have the source node, target node, and optionally the edge weight. So it's most common because this is also what we call a common separated file or a CSV file. And however, being most common doesn't necessarily mean it's the most easy to work with. Most of time it is, but sometimes it can be painful. For example, if the node or edges have many attributes or columns, then it can be painful to parse them. For example, if you have, say, a lot of node attributes that have text or the node name included, then you may need to parse double quotes or single quotes, and that can be painful.

So you notice that for the nodes often that they are identified by their numerical IDs. So you may ask, why? The main reason is because nodes often appear many, many times in edges. For example, in a billion-edge graph a node can easily appear like tens of thousands of times. So if you don't represent each node by it's node ID, then you might need to repeat that node name, for example, many, many times. And imagine that is a very long node name, then it will be not very space efficient. So that is why we want to represent each node as a node ID.

So then the question is, how do we assign IDs to the nodes? So for small graphs, it can be pretty simple. But imagine this is a billion graph, then it can be pretty computationally heavy. And also you may even run out of RAM, right? For example, you would need to store the mapping of node name to node ID. So in Java, you might use a map, in Python you might use dictionary. And imagine having a billion entries, so you can run out of RAM. So how do we do this easily?

So of course, as we mentioned, we can do it programmatically. Or my favorite way is actually using SQLite, so we'll look at an example of how to do that.

So all of these techniques are doing the same thing, so essentially applying the same concept, which is that for every node or every entity in the graph, such as a person that we have never seen before, we want to identify or assign a number to it.

So here is an example of how to use SQLite to assign node IDs. So this is my favorite way to assign node IDs because everything can be done using a simple SQL, without writing complex code.

So the first step is that we create an empty table, say, with a column called name to sort node names. Second step is to create an index on that column so we can quickly check for duplicate node names. The third step then is to just keep inserting node names into that column, and duplicate node names will be automatically ignored.

And each node's ID can be retrieved by just getting the hidden rowid column associated with the row for the node. And rowid is a hidden column that SQLite uses internally to identify rows. And here we serve [unintelligible] or use it to identify nodes instead.

And lastly, once you have the node name to node ID mapping, then you can easily convert an attribute list that consist of node names into an attribute list of node IDs by using a simple joint statement.

So now that we know how to represent node IDs or a graph, so how do we store the graphs? So there are different ways.

So if the graph is not that large, then you can store it on your own computer. For example, using SQLite, or you can use a graph database, say Neo4j. And on a server, if you a relational databases, again, you can use the more scalable ones like MySQL or Postgres, and Neo4j, for example, now also support server deployment.

And if your graph exceeds the space of a single machine, then you will need to use a cluster of machines. And there are also multiple ways. For example, if you need real-time read and write of graph nodes and edges, for example, then you may want to consider Titan, which is built on top of HBase, or the alternative, S2Graph.

And if batch processing is fine, meaning you don't need real-time read and write, then you can use traditional Hadoop. You can say, for example, the graph has adjacency list or an edge list.

There are also other libraries like Hama, Giraph, and they are both inspired by Google's Pregel project. And also from other commercial companies like Twitter, they also have their own database, for example, FlockDB from Twitter.

So in my research group we like to use SQLite. And the reason that we use it is because it's great for our use cases. It can easily handle gigabytes of data. And usually that translates into about tens of millions of nodes and edges. And usually, that's already a pretty good size. Of course, if you want to handle billion edges, then you will need to use a cluster-based deployment.

Another benefit is that SQLite is very easy to maintain, so this is basically all the benefits that you would get for SQLite you would also get it here. So very easy to maintain and you have just one cross-platform file that can be used in all platforms, like Windows, Mac, or even iPad. And also have a lot of language support, so you can write programs using different languages, using APIs.

And doing queries on the graph is also easy. For example, we'll look at an example very shortly, which is to just use one SQL statement and you can find all node degrees. Another bonus is that SQLite supports full-text search. So if we want to, say, find particular nodes with a node name, you can easily do that using full-text search, which is built into SQLite.

So if you do choose to use SQLite to store graph. So here is probably the simplest schema that you may be able to use. Which is just having a table called edges and you store the source node ID in one column and the target node ID in the other column. However, you may sometimes want to also store node information, and also for edges you may want to store, let's say, the edge weight, timestamp, and so on. Then you may need to use more tables. For example, you may want to have one called nodes. And has three columns, one is the node ID, and the other is, let's say, node type, you can store an integer or anything you want actually, and also the node name.

And similarly, for edges, you may also extend a previous design to also sort the edge type, edge weight and timestamp.

Here notice that, for the nodes table, we may want to make the node ID what we call INTEGER PRIMARY KEY. So INTEGER PRIMARY KEY meaning a unique, it'll make it as a unique index, and also that's how you identify each node.

In this video we looked at how to store graphs and how to represent graphs.

Graph Power Laws

Hi everyone, we are going to continue with graphs, and we're going to learn about what to do when we first analyze graphs. Specifically, we'll also look at power laws, which is a functional relationship, and we'll try to understand why that's important in our analysis.

So after you've gotten access to a graph dataset, you should try to visualize it if it's small, and also to analyze it doing what we call graph mining, meaning doing data mining on graph data. And data mining is an interdisciplinary sub-field of computer science, aiming to discover patterns in data. So for graph data, we may want to figure out if we can identify patterns that we expect to see, and also perhaps there are patterns that we expect to see but we actually don't see them.

And the reason that it's important to do so is say if we are able to identify patterns, that means we can also build models on the data. And with the models, as we learned from the classification clustering, we can do protection. We can also do recommendation. And we can do something like, let's say you're able to figure out the patterns in social network, let's say on Facebook. Then you will be able to predict whether a person, say Alice, is going to link to or make friends with Bob.

Finding outliers is also important, because outliers are things that we don't know yet, and often that lead to new insights.

So what patterns do we expect to see? When people first started studying graphs, they proposed a random graph model, meaning the edges are independent and edges have equal chance in forming. So, for this kind of random graph, there is no obvious patterns. But are real graphs like Facebook, social media, really random?

So as there's no, actually real graphs are really not random and we expect to see a lot of patterns. For example there are patterns in diameter, where diameter is the greatest distance between any pair of nodes in a graph. There are also patterns in the in-degree and out-degree distributions. So that means for a node, how many nodes that is pointing to or how many nodes is pointing at that particular node. So that distribution will also see patterns. And also actually quite a few more. So we're going to look at the data and see what are the patterns that we're expected to see.

So the Faloutsos brothers were one of the first researchers to study the Internet topology as a scale. And I highly recommend you read this paper because it is a seminal work that has fundamentally shaped the field of graph mining. Also, a fun fact, the oldest brother, Christos, was my PhD advisor.

Okay, so what have they studied? They studied the degree rank distribution of the internet domains, such as att.com, ibm.com, etc.

So here on the screen what you see is a chart where the vertical axis is the node degree. So popular domains like att and ibm, you will be able to find them towards the top of the chart, and for horizontal axis that is the domains rank. So, higher rank domains, again say att and ibm, they would be towards the left.

So what you're seeing here is that if we plot all the domains on this scalar plot, so each dot is one domain and take the log scale on both axes then we will expect to see a straight line. So this was a major discovery about Internet topology because no one really knew that this was the case. And it's called a Power Law because here we're taking all of scale. And once we take all of scale, then the slope is essentially exponent in the relationship.

We also see Power Law in the eigenvalues of the adjacency matrix of the graph. For example here, we are doing something similar where the vertical axis is the eigenvalues and the horizontal axis is the rank of the eigenvalues. So again, we'll see a straight line in the log-log plot.

And the power laws are very common, and they can be found in numerous real world domains, such as the number of sexual contacts, income, where we now know the wealth of the world is concentrated in a small number of people, and the majority of the world they are not as wealthy. So, this is one of the instantiations of the 80-20 distribution. And also on computers you might find that the number of downloads, or the duration of downloads, or the number of jobs, for example, on a computer would also follow this Power Law.

For example, here on the left we see another example of website traffic. So, the horizontal axis is the number of visits that a website receives. So the more popular websites, let's say ebay, so those would be more towards the right. And the vertical axis here is the number of websites that have that particular [number of] web visits. So that means more towards the top would be the more websites and more towards the lower part of the chart would be fewer websites. So again, ebay, we don't expect to see many, many popular sites. So really high website count, we have very low numbers so that would be more towards the lower right.

So besides power laws in degree related distributions, are there other laws? And yes. Actually there's a lot.

And we will also see patterns in graph diameters. So specifically, graph diameters tend to be pretty small, around six or so. And a popular example of this small world phenomenon is the Kevin Bacon number, which measures how many steps away an actor or actress is from a Kevin Bacon, which is a very prolific actor. For example, if I appear in the same movie as Kevin Bacon, then I would have a Kevin Bacon number of one. And if you haven't worked with Kevin, but you have worked with me, then your number is two, because you're two steps away from Kevin Bacon.

So we know graph diameters tend to pretty small, around six or so. But how does it evolve over time?

Prior work on Power Law graphs hint at slowly growing diameters, which seems to make sense. Because if you consider adding more nodes, and so more nodes, more edges, and it seems like the nodes might actually be further away. So, diameter will increase. But is that really the case?

So it turns out, no. Diameter actually shrinks over time instead of increasing.

So here's an example where we look at real-world data, a patent citation network, where each node is a patent and an edge connects two patents if one's like the other.

So in the chart here, horizontal axis is time, so we have the data from 1975 to year 2000. And vertical axis is effective diameter of the of the graph. So we explain shortly what we mean by effective, so now you can just loosely think of it as a diameter off the graph.

So what observed is that in the earlier years, the diameter is actually pretty high, all the way up to 35. But as we add more nodes and more edges, the diameter actually shrinks.

So after the fact, after these analyses, so we know now the actual diameter shrinks. And it makes sense because as you add more nodes and more edges, the edges actually help hold the nodes closer to each other, which effectively is shrinking the diameter of the graph.

So, on the previous screen we said we would use effective diameter, so why is that? The main reason is that if we use the maximum diameter, so that means that official definition of diameter, then such measure is very susceptible to outliers. So here we're showing a toy example. Let's say you have a majority of nodes and it's concentrated in the middle but then you have, let's say, two nodes that are really, really far away, and let's say they're like 100 steps away. Then the diameter of the graph will become 100, because diameter is the longest shortest path in the graph. So to connect these two nodes that are really far away, you need 100 steps and that becomes the diameter of the graph. So for that reason we use effective diameter instead which is defined as the minimum number of hops in which 90% of the connected node pairs, that means any two pairs you want, as long as they're connected in a graph, can reach each other. So essentially, that means we're throwing away the outliers, and don't have those node pairs diameter influence our measure of majority of the nodes in the graph.

A related problem that the researchers studied was the evolution of number of nodes and number of edges in the graph. So let's say the nodes at time t is N(t) and the number of edges at time t is E(t). And suppose now I tell you that at time t+1, so that is N(t+1), we have the number of node that's twice as many as what we have at time t. Then what's your guess for the number of edges. Would it be twice as many, fewer than twice as many or more than twice as many?

So the answer is actually overly doubled. So then it's more than twice. And also it interestingly obeys what we call the Densification Power Law.

So, let's look at an example. So again, this is the patents citation network. Here on the horizontal axis, there's the number of nodes on the graph, and vertically there's the number of edges on the graph. And each dot here in the graph is one year. For example, over on the left, we see the number of nodes and number of edges for 1975, and then we have 1976, and so on. So here again, we'll do the plot in log-log scale, then we will see a straight line, so that means it's also following power law.

Actually, there are a lot more laws. And so far today, there are 11 or so and there was already results from a few years back, so we expect to see a lot more to come. For example, we have also power laws in the number triangles, we have power laws in the principle eigenvalue, and so on.

So what does this mean? So it means that whenever you have access to graph data, you should try to compare as many distributions as possible, and to see if your graph fits them. So if it doesn't, we should find out the reason why, because often that might mean there's some errors or problems in the data. And sometimes it might signify some new patterns.

So here's an example. This is a real distribution from one of my earlier works called Polonium, which studies malware at a large scale. And here's a plot that we're showing. X-axis is the number of files that we've collected, and the vertical axis is the number of machines that have a particular number of file submissions. [Note: The following is a description of a different plot shown at the 19:50 mark of the Fixing Bar Charts, Line Charts, Tables, and More lecture from Week 4 (Fixing Common Visualization Issues.] So that means you are here showing the file prevalence distribution. So for example, for some really popular files, let's say Microsoftword.exe, then those files would appear at the lower right hand corner. So they are really popular so they are more towards the right. And there are not that many of them so that's why they are more towards the bottom. And for the files that are not very popular, let's say they only appear on one machine, so they are more towards the top left corner. So very interestingly here, when we do a log-log plot, we don't really see a straight line, but instead, we actually see two little hills. And the reason we see these two hills is that actually the data is collected over two major releases of a software. So when we combine these two, so instead of a power law, now we get two little hills. So being able to plot this chart they help us to do a sanity check so that we know about the data distribution that we're dealing with. And it's important, because all the algorithms, machine learning, data mining algorithms that we want to run, often they're dependent on the distribution of the data. So if it's not really fitting our assumption, then they may not really be able to compute effectively.

In this video we looked at a few power laws in Graph. And also we understand why it's important to know them. Because then you can find patterns. And also you can find outliers, which might help us learn new insights.

Centralities: Degree, Betweenness, Clustering Coefficient

In this video, we'll learn about the common ways to figure out what are important in the graph using the so-called centrality measures. So that we can focus on looking at a few relevant notes inside of all the millions or even billions of items in a large graph.

So, we're going to look at centrality. Another more easy way to understand is the notion of importance. So, why would we want to figure out what is important or what is central in a graph? Is that so we can rank all the nodes in the graph. So, imagine you have a million- or billion-node graph, then we can't really look at all of them. So, having a ranking would really help. For example, on Twitter, it may be helpful to identify who are the influential or the celebrities so they can follow them. And also, they learn about what's happening in the network.

And similarly, there's a notion of gatekeeper, which is pretty common and very important on professional social network like LinkedIn. And often these are what we call the people who are connecting communities. So, for that reason, often headhunters likely contact these gatekeepers because they are very well versed in different domains and they can help connect people to other people.

So, having a ranking on nodes would help us with graph analysis, help focus our attention, and it will also help us with visualization understanding. So, for example, instead of visualizing all the one million of node, we can say just focus on the subgraph formed by the top 100 nodes out of the millions of in the whole graph. So, this is actually a notion that has been used a lot in search engine.

For example, when you use Google, often, you'll only focus on the top 10 or 20 pages. And the reason is that Google know that there's no way you can look at all the millions of webpages. So, they're providing a ranking that can help you focus on the ones that might be more relevant. So, we'll see examples of how we compute this kind of relevant ranking and use it to help with our analysis.

The nice thing about node centrality is that most graph analysis packages already have them built in. So you can just use them, you don't really need to implement anything.

So I'd also like point out that there's also edge centrality and here we focus mainly on node centrality and a lot of the concepts are pretty similar across both measures.

So, degree centrality is probably the simplest and most common centrality people will look at. So, degree just means a number of neighbors of a node. So, for a directed graph you will have in degree and out degree. In degree is the number of incoming edges pointing at the particular node. And out degree is the number outgoing edges. And for undirected graphs and then only degree is defined because there's no direction.

And to compute node degree we can simply do a sequential scan through edge list. For example, with this toy data on the right. And what you need to do is you create array to store the node degree of every node in the graph. And as you step through each edge in the edge list, you just increment the node degree. Let's say the degree for one, node one by one as you go through the edge 1,2 and then you increment that number by 1 again as it goes to the edge 1,3. So, you'll need to do one sequential scan. And if your graph already stored in SQLite then it's even better, you don't even need to write any code. And all you need to do is write a simple SQLite SQL statement.

So, suppose your graph is stored in a table, let's say edges with two columns, source node id and target node id. And what you need to do is first create an index for each column. And this will allow you fast access, random access to the edges or the edges associated with a particular source node id or particular target node id. And after that, then you just need to do one statement, a group by statement. For example, here we say select count(\*) from edges group by source\_id. So, this is how you would be able to find the out degrees of all the nodes. So, just one statement and you can compute the out degree distribution, so really really easy. So, this is another benefit of why using SQLite. So similarly, you can just make a small change, let's say instead of grouping by source node id, you can say group by target node id and they help you compute the in degree distribution.

And betweenness centrality is another very popular centrality measure and high betweenness score that means that is more likely to be at gatekeeper in the graph. So here on the right, we use color to indicate the betweenness score, so more towards blue that means higher betweenness, more towards red that means lower betweenness. So, what does it really mean? So intuitively, gatekeeper means you're trying to connect someone in the network. You're likely to pass through or go through this gatekeeper. Real world example is let's say two people in different communities and they want to be introduced, then often they may need to go through some middleman. And that middleman would be a gatekeeper.

So, how do we define this measure, how do we compute that? So, to compute that for a node, the betweenness score for a node. We need to find shortest path in the graph. Essentially, we need to find all shortest paths between all pairs of nodes in the graph. For example, let's say we want to measure the betweenness score of the node v, then we need to look at all the shortest paths that would go through node v for every pair of nodes in the graph. So, that would become our numerator in this formula. And then for the denominator, we would need to look at all the shortest paths connecting a node pair s and t. So, we don't really care whether these paths will go through v or not. So, that means this fraction here is a number shortest path that goes through v over the number shortest path, regardless of whether it goes through v or not.

So, another way to think about it, that means you want to measure how often this node v serves as the bridge that connects to other nodes.

Another popular centrality measure is clustering coefficient. And it's a measure for a node of how close a node's neighbors are from forming a clique. For example, if all your neighbors know each other very well, then you have a really high clustering coefficient because all your friends know each other. They're forming a a lot of triangle, so as a very close net community.

So a value of 1, which is the maximum value, that means all your neighbors form a clique, so everyone know each other. A value of 0, so that means none of your neighbors know each other. So, that means essentially, having a star, you're a center of a star subgraph where you connected to all your neighbor by definition, but none of your neighbors know each other.

So here, the three figures on the right is very good illustration. So, in the first graph here, at the top, the node that we're considering is the blue node and it has three neighbors. And in this scenario, the clustering coefficient is one because all the possible edges among the circle indicated by the black follow up line there, they are already present. So, that means the three neighbors there, excluding the blue node, they are forming a clique, so everyone know each other. So, now if you look at the example in the bottom, the first scenario, where the clustering coefficient is zero, so here none of the edges have exits, so that means they're all the dotted lines there, the red dotted lines. So, none of the edges are present. So, that means the blue nodes now is the center of a star, a three-node star. And while for the center case we have a clustering coefficient of one over three, so that means out of the thee possible edges only one is present.

So, this is an example showing you what we mean by having a minimum clustering coefficient of zero, where none of the neighbors are connected. And the maximum value of one, where everyone is connected. So, to compute clustering coefficients, we actually require us to do triangle counting. And at the first glance it may not be obvious why we need to do triangle counting. But if you look at the graphic on the right and now instead of considering as counting edges. But think of it as counting how many possible triangles that exist. So, in the first case, at the top, you will see that all three triangles are present between the blue node and two of its neighbors. So for example, the first triangle we can say is a triangle formed by the blue node and the node at the top and the node at the right. And second triangle will be the blue node, the triangle on the right, the triangle at the bottom. And the third node is the blue node, triangle at the top and triangle at the bottom. So, here we see all three triangles. So, that means I counted triangles another way of thinking, considering whether we are seeing an edge or not.

So, counting triangles are expensive. In general, that means when you do something called neighborhood intersection and fortunately there are a number of approximate algorithm to compute that. And one of them is really clever and we can compute triangles really, really quickly.

So, this leverages the formula whereas as the number of triangles in the graph is exactly the product of 1 over 6 times the sum of all the eigenvalues cubed. So this what gives you the exact triangle counts. Actually, this formula has been known for a long time. So, that means as long as you're able to compute eigenvalues of the graph adjacency matrix, you take the cube of them, you add them all up and take one over six. And then you get the exact count.

But remember that from the previous video, we tend to see power distribution in the eigenvalue distribution. And also that means, we will have a very highly skewed distribution. And that means, we only need to compute the first few eigenvalue instead of computing all of them. So, this is a reminder looking at the power law. So here, vertical is the eigenvalues and the horizontal is the rank of eigenvalue. So, that means more to the left, those are the greater eigenvalue. And very quickly, you will see that once you stop up to 10, 20, or even 100, the eigenvalues drop really, really quickly. So, remember that in the formula, we need to take the cube of the eigenvalue. So, that means for smaller eigenvalues, taking the cube of it, it becomes even smaller. So, that means we can just ignore them.

So, that means in practice, what we can do is just keep the top three or maybe ten top eigenvalues, and we can get a significant boost. For example, in this chart we are showing the accuracy versus speedup plot. So, vertical is the accuracy. So, that means how close approximation of number of triangles to the real number of triangles. And horizontal is the speed-up. So here, unsurprisingly, we will expect to see actually like thousands of time faster and while we're still able to maintain a very high accuracy. For example, we can get about 97.4 accuracy and at a 1000 times speed-up.

There are actually a lot more centrality measures. For example, what we looked at so far is degree centrality, betweenness centrality. There are also proximity based centrality, closest measure and also Eigenvector centrality.

In this video, we learned about a few centrality measures, such as degree, betweenness and clustering coefficient. So, that we can focus on a few relevant nodes to look at instead of going through millions or even billions of items in a large graph.

PageRank and Personalized PageRank

In this video, we're going to look at PageRank, another very popular centrality measure, and also its variation called personalized PageRank, which can be used to personalize recommendations for graphs.

So PageRank was originally invented by the founders of Google, Sergey Brin and Lawrence Page, and it’s published back in 1998. And this was also one of the first algorithms that they used to rank web pages.

The problem that PageRank is trying to solve is to try to rank all the web pages on the web. So that means that given the Internet, let's say, as a directed graph. So we have directions, because one web page can link to the other, but may not be vice versa. And given this graph, we want to find its most interesting or most central node. And the intuition that it's using is that we say a node is important or interesting if it's connected with other important nodes. So here, we notice that it's a recursive definition, but that's okay, because intuitively it makes sense. And the more important thing is how do we leverage intuition actually come to a score numerically. So the proposed solution is to use the random walk model. So in this random walk scenario, it means that the most popular nodes are the ones that have the highest steady-state probability or SSP. So what does that really mean how? Is there a more easy to understand way of figuring this out?

So we can consider, let's say we have a random walker. Like a person whose whole job is to click on web pages on the web. So now we give this person, this random walker the whole Internet. And we allow this random walker to start with any nodes in the graph and allow them to do what we call a random walk. So that means this person, the only thing that he or she would do is to visit a web page, any of them and then flip a coin to decide to go to any of its neighbors. So in this example, let's say I start with node 1. So I'm the random person, and I'll flip a coin and determine which is the next connected web page to visit. In this example, there's only one way, because 1 is pointed to 2, so there's only one way to go. So that means I'm surely going to node 2. So then at node 2, I will flip a coin and decide whether to go to node 3 or node 5. So equal chance here. I could go to 3, or go to 5. And I will repeat this process for an infinite time. So that means if I keep walking again and again for millions of years, then in the end, what you will better compute is the probability on specifically what we call a steady-state probability of finding myself on node number 1, probability finding myself on number 2, number 3, and so on. And these probabilities would also become the PageRank score.

So in this example, state means web pages.

So how do we compute this steady-state probability? Remember from linear algebra class, one way is to first construct the transition matrix. In this example, we have five states or five web pages. So the next thing we construct is a 5 by 5 transition matrix where the number of nodes equals the number of rows and number of columns. And also, you'll notice that this is what we call a transpose, column-normalized matrix. It's transposed because the columns are where the nodes are from, and the rows are where the nodes are going to. For example, for node 2, it has two possibilities. So it either goes to node 3 or node 5. That means for column 2, we'll have a value in cell number 3 and cell number 5. It's column normalized so that these two numbers add up to 1. So that means there's an equal chance they're going to node 3 and going to node 5. And remember from linear algebra class, to compute a steady-state probability, we use the formula Bp = p. It can be anything. Actually, it doesn't really matter. You can solve p exactly by finding an inverse.

So another way to interpret that formula you just saw is that, in such a way, computing p, which is the eigenvector that corresponds to the largest eigenvalue of our adjacency matrix. The highest eigenvalue is 1 because the matrix is column normalized. So, in other words, given the graph, we were able to construct this transition matrix, which only needs the nodes and edges of the graph. We would be able to compute the steady-state probability, which is all stored in the vector p. So a question that you might have is, why would such a p exist?

So there’s actually a theorem that shows it is guaranteed to exist. So it exists as long as the matrix is n by n, which it is for our case, number of nodes, and number of columns, and rows are the same, as long as that matrix in nonnegative, which is also satisfied, right? Because that's the probability of going from one node to the other, which is guaranteed to be nonnegative. And the third one is irreducible. So we haven't really discussed this yet, but we are going to very shortly. So, what the theorem says is that as long as all these situations are satisfied, then we are guaranteed to be able to find p, which is great.

So now, we can do a quick recap before we show you the full algorithm of PageRank. So the model that we're using or the intuition that we're using is that we're trying to imagine a person randomly moving along the nodes and the edges in the graph. And a node's PageRank score is the steady-state probability of finding that random walker at particular nodes. So we're able to compute our scores for all nodes in the graph.

So previously, we mentioned that p is guaranteed to exist if it's an n by n matrix, nonnegative, and irreducible. And we haven't talked about irreducible. So which also means we need to figure out how to make that matrix satisfy that condition. So in practice, that's actually a very intuitive way which also would help us understand a scenario or to handle scenarios where the random walker may get stuck in a particular part of the network. So imagine the graph has two connected components. So that means there's one subgraph where there's no edges connecting the other subgraph in the network. So imagine, you are a random walker starting in one of such subgraph, then there's no way for you to go to the other subgraph, right? Because all that you can do or allow to do is to follow links. So since there's no link connected to the subgraph, there's no way for you to jump. So intuitively, that means we need to solve this problem. And to solve this, we want to introduce occasional random jumps to any node. So that means, instead of only following links, now a random walker also has the option of going to any nodes like teleporting to any nodes in the graph. So it turns out that by implementing this occasional random jump, it also makes the matrix irreducible.

So now we can look at the formal definition of irreducible. So it means that from any state, in our case, any node there is a non-zero probability of going to any other state. Or any other node, so which essentially means that's the random jump on teleportation that we allowed the random walker to do.

So now we can look at the full algorithm. So here the formula, what you see, it's very similar to what you previously saw which is p = Bp which we know we can solve it exactly by using inverse operation. So the only new thing we added is the constant called c, more specifically 1- c. So 1- c, we call it the fly-out probability, or the teleportation probability for the random walker to go to any random node in the graph. So let me select part of the algorithm, which is constant c times B p. That is what we have been doing when we're doing link-following. So let me say, c probability of just doing link-following. And now we also have at the alternative which is 1- c over n, and n is the number of nodes and then an all-one vector. So this probably looks a little scary, but this is actually just a vector with all the same value. So specifically once you move n inside that vector or 1 vector essentially that means all values are 1 over n. So essentially that means there's a 1- c fly-out probability of using this 1 over n vector. And that means you have 1 over n probability going to any of the nodes with equal probability.

So that formula is actually also a very scalable way to compute a PageRank for large matrices or large graphs. So it can scale to billions of nodes and edges. And to compute this, we're actually using a power iteration method which you probably already learned in a linear algebra class. So what that means is we can literally apply the algorithm again and again and again.

For example in iteration zero, what we do is we initialize the vector p which is on the right hand side to anything you want. So it can be a random vector as long as it's not zero. So a very popular way is to initialize it to all ones. Another way to initialize the values is rather simply proportional to the node degree. And with that p vector initialized then you just apply the formula to the right-hand side, which will give you a new vector, p prime, to the left. And what you do then is to move the vector p prime to where the vector p is. So that's your second iteration. And this second iteration, you do the same thing. You apply or to multiply p prime with the matrix b and then you also add the 1 minus c over n vector. And then you'll get a new vector which is p prime prime, and that's your third iteration. So you just keep doing this, and eventually you'll notice that the vector no longer changes. The p prime prime prime for example, they no longer change. So this is the case where we say the algorithm has stabilized. So it's guaranteed to stabilize, so which means you can just run it for many iterations. And eventually you have the PageRank score for all the nodes in the graph.

And in practice for large graphs, which actually don't really need to run that many iterations. For example you just run 20 or 30, and you'll notice that the PageRank values don't really change that much anymore.

So if you really need to compute the PageRank algorithm, it's good to be able to do a sanity check. So that means to check if your algorithm's running correctly, is it giving the right values. And to do that I highly recommend going to this interactive PageRank demo where you can create arbitrary graphs. You can create a toy graph for example, putting in the nodes, putting in the edges connecting them. And you can immediately see the PageRank values, here it’s the white text on the black background. And also we can adjust the fly-out probability too, here they call the dumping factor.

So a very nice thing about PageRank is you can run it on any graphs you want. So all you need are the graph edges. So for that reason, you want to keep this as part of your algorithm toolbox. And generally it's better than degree centrality, because degree only considers neighbors that are directly connected to the node that you're looking at, while for PageRank, it actually also connects longer range relationships. For example, you can look at it maybe two steps away, three steps away, and aggregate all that information. And in practice it's also really great for  large graphs, because the run time is linear in the number of edges in the graph. So that means we can scale to billion-scale graph.

But it does have a shortcoming, which is also why that Google these days does not only use PageRank, but actually a combination of many algorithms. So specifically, we can do something called Google Bomb. So that means how do we mislead the PageRank algorithm?

So one way to raise a PageRank score of a node is to create a lot of fake nodes in the graph and have all of them point to the node that you want to raise a PageRank score for. And if you look at the algorithm again, so having more votes go into a node can help increase a PageRank score.

So you may notice that for PageRank, no matter who runs the algorithm, we get the same PageRank score values. But intuitively that doesn't quite make sense. Because let's say you want to do ranking of web pages, then maybe I am more interested in, say, sports websites, while you may be more interested in, say, news websites. So that means not all web pages are equally relevant to me or to you. So how do we adjust the PageRank algorithm so that we can rank pages such that the more relative pages to you may be ranked higher?

So fortunately we only need to make one small change to the PageRank algorithm. And specifically, all we need to change is the all-one vector. So this is a reminder of what we saw previously, where the red vector is the all-one vector. And that is also the reason why we get the same PageRank scores, no matter who runs the algorithm.

So now let’s suppose we want to compute this relative PageRank score or Personalized PageRank score for you where you are only interested or mostly interested in sports websites. So what do you do is then instead of using the all-1 vector, we change that vector to 0-1 vector or actually zeroes plus some other values that's non-zero. So what that means is that you would put a one or nonzero value for the nodes that you are more interested in. You don't need to put a value for every web page that you know about. So you only need to put that for a small number of sites. For example, let's say you are interested in a news website. So I may put a one, for let's say New York Times and another one for Wall Street Journal. And what that means or how that affects the algorithm is that when algorithm is about to do the random jump, now instead of jumping to any node in the graph with equal probability, now it has a preference. So it would more likely jump to the news website. And also here, it uses the notion that news websites tend to be more likely to be connected. So that means by putting a one on let's say, Wall Street Journal or New York Times, that also means that there are more chances that the random walker not only had a preference to jump to those websites, but also more likely that the random walker would go to the other related news websites that I haven't put a one at yet.

And the reason we want to learn personalized PageRank is that you can use it for recommendation. So imagine the case where you apply personalized PageRank to rank products. Let's say you're Amazon and you have a customer product graph. And you're a new customer to Amazon and you have purchased one or two products. So by constructing a bipartite graph connecting a product with a customer that has purchased it. Then you can put a one, essentially, in this bipartite graph on the product that you have just purchased. And now, you run the algorithm, the personalized PageRank algorithm. And after you run it, then you will be able to find a score for the related products, meaning the products that many other customers probably have also purchased, like, besides the ones that you have bought.

So, this is a very generalize-able technique because we notice that we only need to graph structure. We don't really care how you construct a graph, as long as the graph has edges, you can compute the personalized PageRank score.

So this is also a great way for helping to visualize graphs, too. Meaning, instead of visualizing every single node, every single edge in the graph. Now, suppose you are able to get some feedback from the user as to what is interesting. So that means, you can put a 1 on those nodes, and then you can run the personal PageRank algorithm and you can get a ranking very personalized to the particular user. And you only visualize the nodes and edges where the nodes are the ones that have the highest scores.

So, in fact, personalized PageRank is one of the big classes of algorithms. Generally we call them diffusion-based or guilt-by-association-based. So guilt-by-association that means things that connect are more likely to be related.

So Personalized PageRank, another name for it is called Random Walk with Restart. And in other domains, let's say Human-Computer Interaction, or HCI, they also call it spreading-activation or degree-of-interest. You'll see that powerful techniques like PageRank or personalized PageRank actually have been used in many domains and often under different names.

And also, in the next video, we're actually looking at an example, we're using another algorithm called Belief Propagation, which tries to do something very similar to personalized PageRank. Belief propagation is another powerful algorithm, where it is not only able to do recommendations. Also, it has been used in a lot of different applications, say, fraud detection to image segmentation, or error-correcting codes, etc.

And all these algorithms are really popular because they tend to be very intuitive to interpret. They use network effect and homophily. They're also very easy to implement and the math is relatively simple. For example, PageRank is only one formula and implementation is essentially just doing matrix-vector multiplication, a few of them. And in practice, they're also very fast to run. They're linear to the number of edges, so that means they can scale to a very large graph. And also, very nicely, is that they often have probabilistic meaning. That means not only empirical ranking, but also you can interpret the scores probabilistically.

In this video, we learned about PageRank and its personalized variations. And both are very popular because they are easy to understand, easy to use, and also scale to very large graphs.

Interactive Graph Exploration

Today, we're going to look at how to apply an algorithm that you've learned to help at graph exploration and visualization.

So we'll use an example that you might recall seeing from quite a few videos back where we talked about the system called Apolo, which uses machine learning and visualization to help people with graph exploration and understanding.

And in the example you may recall that our task is to help people find more relevant nodes in the graph. So we look at a toy citation network where each node is a paper and edge is a citation, one paper citing the other. And in the example we have some starting papers. For example, we may know, ah, there's one HCI paper in orange, which we might want to read. And then on data mining, paper in blue, which we may also want to read. So now that you've learned personalized PageRank, so these two papers you can think of as the ones that the random walker in the graph is a lot more likely to jump back to whenever they want to do a teleportation.

So what means is that by using the notion of say, random walk or, and also guilt by association, we'll be able to compute a personalized PageRank score for all the nodes in the graph. So guilt-by-association, as you may recall, meaning nodes that are connected in a graph are more likely to be related. So for Apolo we use an algorithm called Belief Propagation, which can do something very similar to personalized PageRank but it can do a little bit more. So specifically, instead of only able to compute one kind of score, as in, personalized PageRank, which either tell you something is relevant or not relevant, which is more one dimensional. Belief Propagation able to have you compute several simultaneous relevant score. For example, one relevant score is for the orange node, and then the other is for the blue node. So actually it can compute arbitrary number of relevant score. But the algorithm belief propagation and patron are very similar, they're also based on diffusion. So that means you can pick it up as more or less doing the random walk style of navigation and computation.

So we may recall that from previous video what we do in Apolo is we start with only one note. Let's say in this screenshot here, we start with a node in the middle in black. And then through the exploration the user would gradually bring in more nodes and edges into the visualization. So that is how we use an algorithm in the back end. So as the user is adding more nodes to the visualization, that means the user also giving feedback to the algorithm to say as in which node are relevant, which nodes they are interested in, and which ones they are not interested in. So in other words, any time a user express interest or like tags a node as being interesting, so those nodes become additional nodes that you will assign a one in your PageRank vector. So the more nodes you're able to get feedback for, the better the algorithm results.

So that means another way to think about it is, in Apolo, we're interacting with application like Apolo, which is using the guilt-by-association diffusion-based algorithm in the back end. The user will essentially be specifying the examples, and then the algorithm will find other relevant things for the user to look at. And notice that, while the graph may have hundreds of nodes or hundreds of thousands of nodes and edges, what we are showing here is only a small subset to help avoid overwhelming the user.

And also leveraging the personalization of the algorithm. Apolo is able to give a very personalized landscape for the user and also the user would feel natural when he or she is interacting with Apolo. So it's like having a partnership with the machine.

So when designing interactive application like Apolo, there's quite a few considerations. And, also, I hope to show through the example Apolo what you might want to pay attention to when you're designing such application that uses algorithm in the back end. For example, Apolo first started in 2009, we actually don't really have a graph visualization. And instead, we have more or less like list and tech space interface, where we have each group of papers represented in a text box. And then the papers that the user has expressed interest for, we highlight them in bold. And immediately we see a problem because people couldn't really find out what's the relationship among these papers.

So that prompted us to add the graph visualization. So meaning, helping people to see more clearly how things are related. And we still keep the list on the left. But interestingly, we see that people don't really use that list that much anymore. And the only time that they use it is when they want to rank things.

So that prompt us to remove that list view completely so that we have screen real estate for the graph view. And instead we do the rank in place, and the rank in place feature, where we allow people to rank nodes on demand.

What would you want to do if you want to build an application like Apolo? So I highly recommend that you build a lot of prototypes before building the final products. Prototypes will include paper prototype, lo-fi prototype, and high-fi prototype. So paper prototype more or less means sketches. So for example, for every screen data users going to see, you can do one sketch. So that is already pretty helpful because you can have your check lot of the errors, on doing sanity check right in the beginning, before you even implement anything. So never under-estimate paper prototype, you may say doing sketches that people many not really care too much about it and they may not see the full picture. And actually that's all right, because for tasks like navigation, let's say, you can add using your sketches, you can ask user, which icon to click on? Like, does this label makes sense? So those are things that people can really easy to catch. And also using sketches actually forces a user to focus on a content, instead of like decorations or color uses of your application.

So after that, you may want to move to what we call lo-fi prototype. So that means something that you very quickly put together, may not be as polished but then you might have the linking or the navigation capability already built. So again, this allows you to very quickly catch issues on doing sanity check. And after that, then, you would move on to build high-fi prototype. This means prototype is very close to the final product you want to build.

So at every stage during the prototyping it is very important that you involve real users as early as possible. So it could be a simple as recruiting friends to try using your tools. Or maybe you want to do it more vigorously, let's say, using a lab study, a very controlled study, having done conditions, everything scripted out and a very rigorous experiment. Or maybe you do a longitudinal study, so that means maybe you release it in the wild, deployed it to allow the world to use it. So all these are great options, and important thing is to involve the real users. So it's very tempting when designing thing to say, I am the developer, I know the application really well, so why do I bother to test it with real people? Actually that's exactly the problem because you know the application so well that a lot of issues that may be very obvious to you but they're not obvious to first-time user at all. So for that reason, it's very important to involve real users as early as possible so they can catch all those issues. And also I highly recommend that you take courses in Human Computer Interaction HCI or human factors or user interface design. So these courses will teach you a lot more about what you might want to pay attention to, when you are working with real users.

And, if your application uses algorithms, let's say, as in Apolo, so it's important to identify the algorithms early on. Because there may be cases that there are just no scalable algorithms available. If that's the case, then that would affect your design. So if you wait until the very end, and say, I'm going to figure out the algorithm much later, then it will affect your implementation, you might even scrap everything away. And also, when you're doing a design for interface as in like Apolo, very popular way they say it's a use iterative design. Actually, this is kind of a default in industry. And the main reason we do it is because it's very hard to get things correct the very first time. So I mean remember from in the older days in software engineering the waterfall model is a proposed way to do it. Meaning you would collect what they call user requirements, and then you will move forward to do a system design and the implementation. And every step, I don't really go back. And this says we know that's not a good model to follow, because it's very unlikely that we're able to identify all the requirements, all the issues in the first place. So for that reason we want to use a more iterative design and which means we want to create prototype, we do evaluation right of the bat. For example you can use a paper prototype and evaluation to other iteration to paper prototyping, evaluation and so on, modified paper prototype based on your user feedback and so on. So you keep doing it until your prototype or your product becomes very polished and very easy to use. So in the nutshell, so quick evaluation is very important, because it helps you identify important fixes early on and it'll save you a lot of time.

In this video we learned about how you may want to use scalable algorithm like personalized PageRank in a interactive graph exploration system. And also we touched on some of the important points you may want to note when you're designing the user interface for such application. And the very important thing is that you want to do iterative design and iterative testing as early as possible so you can catch the expensive areas of problems early on.

Ensemble Method

Bagging and Random Forests

Hi, everyone. Today we're going to look at how to combine Machine Learning models. Specifically we'll look at the Bagging technique and also how to apply it to decision trees, which we call Random Forests.

So as we mentioned before, there are many possible classifiers that we could use. For example, you learned about k-nearest neighbor, decision tree, and there are also many, many more. So which classifier or model to choose? So there are many different possible strategies. For example, you can go from the simplest model to the complex one. Or you may adapt existing models so they can discover a new model based on existing one. Another popular approach is to Combine Models.

And one strategy as specifically we call Bagging. So Bagging stand for Bootstrap Aggregating. So what it means it says you create multiple models on the same dataset using sampling. So how does that work?

So suppose we have a dataset S. And what we do is we do sampling repeatedly. And every time we sample we would call that S\* [S star]. And this sampling is done with replacement of size n. So that means, typically, the same as your original data size. And after doing this sample S\*, what we call bootstrap sample, then we will train a classifier on S\* to get a classifier f\*. Then you'll just repeat those two steps again. So meaning you will do sampling repeatedly and for every sample that you get, you'll train one classifier f1, f2 to fB, so B number of classifiers. Then after that, all you need to do is to do a majority vote. So that means you ask all these B classifiers to classify things and then get the final vote by taking the majority.

So, to apply this bagging technique on decision tree, so that means how to create multiple trees from the same dataset using the bagging technique. So very similar, so the first step is to get a sample S\* with replacement, so identical to previous screen. And then you grow a decision tree. So here we have a B number of them. So we'll repeat the process B time, so you would have a tree number one, T1 all the way to TB. And similarly, we will get a majority vote across these B number of trees to get the final verdict.

So there is an issue with this approach though. As we are growing these many trees, those trees can actually be very similar. So imagine you are using information gained to pick attribute. So you can imagine that no matter which kind of sample that we get from the original dataset, we are using information gained then very likely the first attribute that you get to pick and to split on in the tree would be very similar. So that means we're essentially getting B identical tree. So getting B identical tree, so that also means the majority of those is essentially the same as just having one tree. So how do we solve this problem?

The solution is to introduce randomness. So specifically that means instead of making all the attributes that the tree can choose from we only make a subset of attributes available. And, specifically, at every time we want to do a split in the tree, we randomly pick m of the d available attributes. So d here is old available attributes, and m is a small subset.

So by using this technique then what we have instead of a bagged decision tree, we have a bagged random decision tree. And actually that's a significant improvement and also in practice we call this random forest. Which is a very popular technique and a very affective technique to use for classification.

So, also a very nice thing about doing bagging is that cross-validation is not really necessary. So imagine in the case of decision tree. So anytime you use a decision tree, so use a subset of a data, so that means also there are data points that are you not using. So that also means that you can evaluate your tree immediately on this what we call Out Of Bag data points, so OOB error estimate. So this is a very nice thing because that means you can do training and evaluation at the same time. So of course you can still do cross validation explicitly, but that's not really necessary. Because research showed that OOB error estimate is almost as accurate as doing cross validation.

So some important points about random forests. So there are some algorithm parameters that you might need to tune. For example, we may need to determine, m, m is a number of attributes that you want to select among all of d available attributes. So some typical value that people start with would be the square root of d, or maybe you can reduce it a very small number to only just one or ten. So these are the numbers that you might need to try. And in terms of the number of trees that you may wanted to use which is B here, the good news is that you can actually keep increasing B. So that means you can just say I want to use 10 and how are the set works, use 100 trees and how are the set works. And because you are doing OOB error estimate, computing the other along the way of testing you can actually keep increasing B until a certain slot. So the chart on the right shows what happening there. So you see that when we have 10, 20 tree, the air is still pretty high, and once you go to about 100 or so, it drops significantly. And based on this curve, we can see that as we increase the number of trees, the error doesn't really improve any more, we can actually stop. And that would be the number of trees that you want to use.

So since the random forests is a collection of trees, so that means when we go pick a tree we also need to figure out the parameters of each tree, such as, the number of nodes or the size of the tree. Recent research shows that parameter tunings may actually not be necessary. For example, there as completely random approach, such as, call it the Perfect Random Tree Ensembles, where, instead of choosing a fixed number of attributes to use, you randomly pick an attribute, or even randomly pick the split point. So what that means is that almost everything is random. And that significantly simplifies implementation as you imagine, and it can also increase the training speed dramatically.

[PERT, [https://www.interfacesymposia.org/I01/I2001Proceedings/ACutler/ACutler.pdf](https://www.google.com/url?q=https://www.interfacesymposia.org/I01/I2001Proceedings/ACutler/ACutler.pdf&sa=D&source=editors&ust=1738598889304750&usg=AOvVaw2SRu7WO_3jdj50vfMp7hdx)]  
[Extremely randomized trees, [http://orbi.ulg.be/bitstream/2268/9357/1/geurts-mlj-advance.pdf](https://www.google.com/url?q=http://orbi.ulg.be/bitstream/2268/9357/1/geurts-mlj-advance.pdf&sa=D&source=editors&ust=1738598889304967&usg=AOvVaw1DSBhOW3iUI63nFBdy-uTn)]

So to conclude, random forests is pretty efficient and also very easy to train. And it's actually one of the best-performing classifiers used in practice. And the other one that people often use is called gradient-boosted tree, which we are not covering here.

[[http://fastml.com/what-is-better-gradient-boosted-trees-or-random-forest/](https://www.google.com/url?q=http://fastml.com/what-is-better-gradient-boosted-trees-or-random-forest/&sa=D&source=editors&ust=1738598889305354&usg=AOvVaw0jqhD_aqaKXYHkm49pNv77)]

So in this video we looked at bagging, a way of combining multiple simple models. And also random forest, which is an instantiation on bagging on trees. And also how, by introducing randomness, we can have a very effective classifier.

Scaling up Algorithms with Virtual Memory

Overview

Today we're going to look at a novel way to scale up computation. So previously you were look at how to scale up computation using a cluster machine, such as using Hadoop and Spark. And we'll look at today that maybe that's not necessarily all the time, so we will look at a novel way to do that using your own computer.

So previously you look at how to Spark a computation using Hadoop, and Spark, or using cloud service like Amazon Web Service or Azure. So one question often people have is when should they use it? When should they not use it? So, in this video I'm going to tell you a little bit about a project that was originally developed at Georgia Tech. And it's using Memory Mapping, or loosely, you can think of it as virtual memory to scale up computation. And it was a project that was select by Jerry Lin who was a computer science undergraduate student at Georgia Tech and now he just started his PhD at Stanford.

The problem that MMap look at is, to figure out, is it always necessary to use computer cluster to do graph computation? So the reason to look at this is because using a computer cluster it often involves a steep learning curve, so it's not as something as easy as just switching a language. You also need to learn about the hardware characteristics, how to allocate enough RAM, and so on. And also there is a cost in using a computer cluster. So there's a hardware cost, potentially, and also the cost of acquiring people who know those skills. And more importantly is that for smaller graphs, such as graphs at maybe the million scale, is actually overkill. Because for such graphs it actually may only be a couple hundred megabytes.

So it can easily fit in everyday computer these days. So this observation got a lot of researchers thinking and said well, if we don't use a cluster, what can we do? So there was originally 2012 where GraphChi is one of the first projects to look into this. And they say, well, maybe we can use a single computer. So a single machine, as in something like your laptop or desktop computer. And try to figure out how much computation power they can really squeeze it out from the machine. So GraphChi is the very first project, and also there's further work like TurboGraph, KDD 2013, that also look at similar problem and try to speed it Speed up computation even further. And what they have in common is that both of them would use sophisticated data structures to help squeeze the whole graph into memory, or part of them. Because, often, when you have a larger graph, let's say, 10 GB and your computer only has 4 through 8 GB, then there's a question of how do you fit the right portion of your data into memory. And also because of that then, these approaches need to do explicit memory management. So that means that you need to figure out when to load data into memory and when to offload data from memory that are no longer needed.

But what that got us thinking is that, well, so it seems like there's a cost to pay. You need to design data structure. You need to do the memory management. So can we do less, so meaning can we do less memory management? Can we use any standard data structure so that we can get the same or maybe even better performance? So that means can we use automatic memory management techniques, for example?

So that got us thinking. So, we're saying that if a graph is large, so usually that means there a lot of graph edges. So, for billion edge graph they are actually for example can be up to tens or hundreds of gigabytes. So it will not fit into the main memory of laptop or desktop computer. So can we say using automatic memory management technique to automatically determine which part of this edge list is needed during computation, and have that part automatically be mapped or be uploaded to the physical memory, which could be about 8 gigabyte for a commodity machine. And also, similarly, can be automatically unmap or unload data that are no longer needed? So this is actually very similar to what other approaches like GraphChi or TurboGraph are doing. But instead they are doing this manually. So that means in their memory management technique they determine when to map and when to unmap data for the memory.

The main idea we can leverage is that, well, it seems like this is something that modern day operating system is already able to do. So specifically, this is exactly what happen in the virtual memory system. So virtual memory has been around for a long time as long as there was operating system. Because there has always never enough RAM. So that means when you have an application that requires a lot of memory, and you run multiple applications like that, the operating system needs to determine when something is really needed. And have that, things as really needed, and put it in the main memory, and for the application that will not really need it, then they will also list the memory. So this is really what a virtual memory system is doing. So that means what if we directly use this virtual memory system and instead use it to scan computations, specifically to ultimately determine which of a dataset is needed and which part is not, then how much of scalability can we get?

And actually that is all the main idea that the MMap project is leveraging. So that also means that it's actually not only conceptually really simple implementation wise, it's also extremely simple.

So we're going to look at an example of how to apply this simple idea to scale up page rank, for example. So, from the graph and the latex video, you saw how we do page rank using the very scalable power iteration method where we compute a page rank vector using a series of matrix vector multiplication. So, that means in this graph the data structure or the part of the data that is taking really a lot of space is the graph edges. Which is a matrix b here. And it's often very sparse for scale free [?] network.

So an alternative view for the PageRank algorithm is shown here. So this is actually a figure that we take from the MMap publication. And here in this alternative formula, we are showing the same thing, it's just using different symbols. So that means in previous screen we see the matrix B and here we are using the notation E, so it's the same thing. But on this screen we are explicitly spelling out the data structure that we are using to store the data. So for the graph E here, so instead of showing in a matrix, which is not really efficient, space efficient, we store it in an edge list. So which also means that edge list is what is really taking a lot of space. And which also means the edge list file is what we can do map into the main memory using the memory mapping technique. So that means we're fitting this list into the virtual memory of the operating system. So that is all we're doing and that's all we need to do.

So how well does MMap work? So we did a series of experiments to compare MMap with other techniques like GraphChi and TurboGraph. And we used different datasets, and also run different algorithms on these graph datasets. For example, we are looking at a chart where we are looking at a runtime, that's for algorithm run on the LiveJournal graph, which has 17,000,000 edges. So 17,000,000 may sound pretty high, but in the fact that edges is only about 500 megabytes on disk. So at the top we are looking at the speed of running ten integrations of PageRank on this graph. So for MMap, it takes only 3.3 seconds So while for other techniques like GraphChi it takes about 54 seconds, for TurboGraph, 6.3 seconds. So it's a geometric increase over GraphChi. And also notice that GraphChi is actually a pretty big library, has about 8000 lines of code, while for MMap because the implementation is really simple, so we only need to do one function call to fit the action list file into main memory. So the actual number of lines of code is only about low hundreds.

And similarly, we look at the connected component algorithm. The result is shown at below. Very similarly, MMap is also the fastest, only taking 2 seconds to find order of connected components in the graph.

So for this LiveJournal graph, 70,000,000-edge graph, actually the whole graph data fit in memory.

So, for the case where the graph does not fit in memory as in the yahoo web graph which has 6.6 billion edges, then we also see very good results. For example, for MMap it takes about 579 seconds to run three iterations of PageRank, while it takes much longer for GraphChi.

So MMap is the timing is pretty similar to TurboGraph. So you might ask, well so it's not big increase or a speed-up. But notice that, for this other library, you will need to do explicit memory management and also implementation wise MMap is much simpler. So that means you can essentially implement different kinds of algorithms that may not be provided by GraphChi or TurboGraph. And also for other queries, such as finding the 1-step neighbor of a node in a graph, we also see that m is significantly faster than another approach like TurboGraph, where if MMap takes only 3.3 milliseconds for TurboGraph takes 154 milliseconds.

So may I ask, why memory mapping works so well? The main reason is actually leveraging the characteristics of virtual memory systems. So for example, virtual memory system has something called, Read-ahead paging. So that means we preemptively load. Part of a file that the system thinks would be needed and to memory preemptively. It works really great for our case where we have the graph edges and if we are starting to go from the beginning then the operating system must say! Probably you may need to use the rest of the edges too, so let me preemptively load those into memory as well. And also, virtual memory system would like to often catch things that I use a lot into main memory for as long as possible. So that's the least reasons use memory management technique. And that translates into keeping the high degree information in the memory system. So that's also a very natural and desirable thing. Because often high degree nodes in a graph is also where the algorithm often access. So that means the information is already cached, so that's as if they're already in memory all the time. And lastly, all these memory management operations are actually highly optimized because virtual memory has been around for a very long time. And also we don't need to explicitly manage memory. That means it saves us a lot of up keeping.

Since virtual memory is such a ubiquitous technique, it's available on almost all modern systems and also hardware as well. So that means if you want to do memory mapping on mobile devices like an iPad, you can also do that. And the speed is also very impressive. So here, what you're looking at here, is that we compared the speed of running an algorithm on iPad mini versus a MacBook Pro. So you'll see that we can handle graph as large as 272M edges. And the speed is roughly about five times slower. So not a lot, and also we notice that iPad or any mobile devices these days have become increasingly powerful. So that means the gap will be even closer as time passes.

So we also extend our investigation to look at how memory mapping might apply, how it works for machine learning algorithm. For example, on the left here, we are looking at how memory mapping can scale with data size, so horizontal axis is the data size on disk And vertical axis is runtime. So we're looking at the runtime of the logistic regression algorithm, specifically we're using L-BFGS as the optimization routine. And here the blue line showing you how well M3 scales. So it's roughly linear, so that means even after the memory exceed the memory size, which is 32 G, it can still work. All the way up to about 200 GB, and the scale is almost linear, which is fantastic. And similarly you look at how M3 compared with Spark, 4x Spark and 8x Spark. So in orange is 8x Spark and in light orange is 4x Spark. So we know that M3 is pretty comparable to 8x Spark and significantly faster for 4x Spark. If you want to learn a lot more about the memory project, I highly encourage you to go to project website to take a look at the publication and also download the dataset to try it out yourself.

In this video we look at another to scale up computation. You can just use your computer and it's based on virtual memory or area mapping.

Week 13

Text Analytics

Basics: Preprocessing, Representation, Word Importance

Hi everyone, welcome back. Starting with this video, we're going to talk about text analytics. So we first talk about the basics, like preprocessing, representation of documents and computing word importance. And then after that, we'll talk about very generalizable and a powerful technique called latent semantic indexing, and also its underlying mathematical tool called SVD, Singular Value Decomposition. So we'll start with the basics first.

So text is everywhere, for example, on the web you can see a lot of webpages. On social media like on Twitter, Facebook, we have a lot of these tweets and posts which are also text. And for academic and academia, you also have academic articles, let's say from Google Books, from journals and conferences like ACM, IEEE. Or in entertainment you can have song lyrics or closed captioning for videos. Or in healthcare you can have like medical records or EHR, Electronic Health Records. So all of these are texts.

And there are many important questions that you would want to ask and to study while analyzing large amounts of text. For example, you might want to do a classification. So you want to know what are the genres or the topics of books and articles. You may also want to do something we call sentiment analysis or tone classification, as in for online reviews, like, let's say for tweets. So is it saying something good, something negative? For example. Or you may want to do something in forensics, for example, establishing the authorship of articles or authenticity or doing plagiarism detection.

So before diving deeper into text analytics, I want to first mention that there are a number of popular tools that you can use today, right away. So these are libraries like Stanford NLP, OpenNLP, and NLTK Python library and so on. So all these tools are for essential features like tokenization, sentence segmentation, finding out the part of speech for actual word or doing something we call entity extraction. So you can see some example at the bottom there. For example, the first sentence you can extract like which person is say the president of China? So that can figure out China is a location. Also it can do something like at the example below, when you see in the second part of the sentence is showed off "his familiarity," and we know that "his" actually refers to president. So these are the things that this library can already do for you. So in this video we learned about the basics but once you learn about the basics you can immediately use these tools to help you.

So in the videos of this text analytics module we will cover several topics which include basics such as preprocessing using stemming, representing documents using something we call bag-of-words model. How to compute importance of words in a document, and also how to rank words and rank documents. And lastly we will talk about a very powerful generalizable technique called Latent Semantic Indexing that can help you find concepts or topics among a large collection of documents which will help us in retrieval, recommendation and much more.

So let's start with stemming. So stemming means reducing words to their stems or their base forms. So an example would be say, if you're giving the words like compute, computer, computing. So all of them we know about computing, in general. And the stem of all these words will be c-o-m-p-u-t-e, so it removes the e, the ing, the and so on. So this is what we call stemming. And there are a number of algorithms that can do this already such as stripping the suffixes, doing something we call a table lookup it's like having a gigantic table so the entries would be compute, computing, computer, and the value could be c-o-m-p-u-t. So that means that whenever you see a word like compute, computing you can automatically look up the stem.

A document consists of many sentences which in turns consists of words. So how do we represent a document such that it can be easily processed a computer or an algorithm? So that's a common question that we have. And a typical approach is to use what we call the bag-of-words model. What that means is that imagine that we have a bag, let's say a big plastic bag. And let's say every word, we're allowed to print it out. And we put every word into this bag and you just shake it all up. So that means you are ignoring any of the ordering among the words. So, essentially, what you end up getting is a bag-of-words, very literally a bag-of-words.

And the reason that we want to do it is for simplicity because now we only need to do some basic counting instead of really worrying about the ordering. And what that also means is now our text documents, or each document, becomes a vector of numbers.

So, for example here, let's say I have two documents, both are really short. The first one is I like visualization and the second document is I like data. So just looking at these two phrases you already know that they have four unique words, I, like, data, visualization. And you can assign a number to each of them. So we can say I is 1, like is 2, data is 3, visualization is 4. So once we have this kind of word to number or ID assignment, then you can turn the two documents or the two phrases into a vector of numbers. For example, I like visualization, now becomes 1,1,0,1, all right? Because it has a word I, it has the word like, no data, it has visualization, so 1,0,1. And for I like data, now you would turn it into 1,1,1,0.

So there's a problem with this simple bag-of-words model, however. We said we assign the same weight which is 1 to every single word. So that means as soon as we see a word, we give it a 1. And you can imagine there is a problem with this which is that now we consider all words as equally important. But in practice, we can imagine some words might be more important, more meaningful, more characteristic of the document than how do we express that importance?

So we can use a technique called TF-IDF, which stands for term frequency–inverse document frequency. And it's a very fancy name, and is also a very popular approach. And because it is very effective in practice, and also very easy to understand.

So the definition of TF-IDF is that you want to compute a word's importance score in a document among N documents. So at just even this a short phrase or description there's already a few important things. So, one is that the scores are computed relative to a document, and also, when we compute a score, we need to consider the whole document collection of N documents.

So when do we want to use TF-IDF score? So if you remember in the previous screen so actually anytime we use the bag-of-word model and start using a zero and one, where one is to indicate existence of the word. Now we can replace those ones with the words TF-IDF score. So that means any time you use word count, you can essentially use TF- IDF.

So how do we compute it? All right, so the final TF-IDF score of a word is actually the product of two parts. The first part is TF, term frequency, and the second part is IDF, inverse document frequency.

So TF, term frequency is easy to understand, it just means how many times that a word appears in a document. So naturally, this number will be high if the term appears many times in a document.

And for the IDF, the second part, and it stands for inverse document frequency. So that means we want to find out how many documents that contain the term that we're considering. But instead of putting it in the numerator, we actually put it in the denominator. So the numerator is actually N the number of documents. So the reason we do this, and we want to penalize the common words, right? So common words could be like stop words like "the," "a," "and," and so on. So these words appear in virtually all documents. So that means there will be a number very close to N. And since it's a denominator, so when N divided by this number which is very close to N, then it is very close to 1. When it's close to 1, then when we take a log of that number it is close to zero. So that means IDF score would be zero or close to zero for the very common word.

And the final score is the two parts combined, so TF times IDF. So for example, if you're considering let's say stop word, a common word like "the," it would have very high term frequency scores, so TF is very high, but then for IDF it's very low, right? Because it appears in all documents. So that means a product of TF-IDF score for the is very low. And for another term, let's say that only appear mostly in one document, it appears many times. So term frequency score would be very high, but the inverse document frequency will be low because it only appears in one or two documents. So when you multiply these two terms together, then the final score for this particular word will have a high score. And it would call this word is pretty characteristic of this document. So it's important, relevant, in that document.

So at the high level, our bag of words approach and the related TF-IDF approach of computing important essentially turn all documents into vectors of numbers. And also, all queries that you might want to make into vectors. And the main reason for using this so-called vector space model is I guess simplicity. So vector space model, again, this is a pretty fancy words. All right, so what that really means is now you're expressing everything as a vector of a number of dimensions. So in our case of using bag-of-words or using TFI [TF-IDF] to score, to encode every document. So that's pretty easy to understand, so each document, one vector. So besides simplicity by turning everything to vectors that also means that we can use all the machine learning and data mining techniques that we have learnt so far such as like classification, clustering and so on to operate on this vectors. So what that means is that now lets say you wanted to search for a document, essentially we're just trying to find similar vectors, right? And also if you want to say, I want to group documents together during clustering, essentially that means we are clustering similar vectors.

So in this video we'll talk about the basics and tech analytics such as during the processing, during the stemming, during the TF-IDF to compute what's important score.

Latent Semantic Indexing (Singular Value Decomposition)

Hi everyone, welcome back, we're going to continue with text analytics. Specifically, we're going to talk about latent semantic indexing and also is underlying mathematical tool called singular value decomposition or SVD for short. And we'll learn about how it is a tool that can help us discover topics in a large document collection.

So given a large document collection, we often want to find the common themes in the document so that we can more easily understand what's in them. And finding such themes or concept is what latent semantic indexing, or LSI for short, can do for us. And the main idea is that we map each document into some sort of concepts and also for each term or each word, we also map into the concepts.

And here, concept is loosely defined as a set of terms and with their corresponding weights. So for example, when I say we have a concept called DBMS, so database management system, and the concept here would be characterized by the words that it contains. So let's say, for the word "data" we have say a weight of 0.8, the word "system" we have a weight of 0.5, "retrieval" 0.6, and so on. So something you should note is that LSI actually doesn't give us a name, DBMS. But from the term weight like 0.8, 0.5 and so on, we can actually figure out quite easily what that concept is.

So here, I say "data" is probably about the analysis system, so that means data analysis and systems likely is probably about DBMS. So what LSI can do for us is to decompose our original matrix into smaller matrices, the first one is term concept matrix and the other is document concept matrix. So the notion of concept is what LSI introduce to us and here we have two concepts, the first is database management concept the other one is medical concept. And also, LSI will tell us to weight, for example, in the first term-concept matrix, so those are what we call the term-weight for each concept. And similarly in the document-concept matrix we also have the weight for the document relative to the concept.

So the benefit of having these two matrices is that our LSI give us a sophisticated way to do document retrieval that goes beyond exact word matching. So for example, let's say we want to find the documents related to the word system. So in our retrieval process we first make use of the first matrix, the term concept matrix to figure out which concept the word system is most strongly associated with. And then we use the second matrix, the document concept matrix. In this case we will see that the document 1 and document 2 has a really strong weight associated with the database concept. So in other words, you can think of it as some sort of lookup where we start with our query, which is a word "system." And then we figure out which concept is the most strongly associated with the word "system," in this case, database concept. And then we jump to the second matrix in using our intermediate result [unintelligible]. So database concept is what we're interested in, and using this idea to retrieve the documents are most associated with the concept.

And what that means is that LSI works more or less like an automatically constructed thesaurus. And it is very powerful in that we can retrieve documents that don't even include the word "system" but contains almost everything else. Because that's an intermediate stack you just saw where we first map our query, which is the word system into some sort of concept association. And using that notion of concept association, then we jump to the documents that are most associated with it. So that means that using this middle step, we actually can go beyond the traditional kind of exact match word, exact match approach.

So a short recap is that LSI is a great idea because it can help us derive these concepts among documents, so we haven't talked about that yet but we will very shortly. And then what we saw also is the application of LSI, which is that it actually builds some sort of thesaurus automatically. And also by using it we can do retrieval that go beyond exact match. And also later on, we'll learn that through LSI we can actually also get a ranking of the discovered concept, which would allow us to throw away the less important concept, so doing essentially what we call dimensionality reduction.

So we look at all of these, so the very first question is, LSI's great but how does it work? So it uses a mathematical tool called singular value decomposition or SVD for short.

SVD is a very powerful and generalizable technique that is going to help us solve important problems such as in finding concepts, as in the example on the right. And also, similar to what we discussed, so in this example here we are looking at transactions of people at a grocery store. So each row is a customer and each column is a product. Also again, I've only used the first group of customer is probably vegetarian so many buy vegetables. And then another group would be meat-eater so they buy chicken and beef, right, so defining these kinda concepts. And then also, the second problem that LSI or SVD can solve is through finding these concepts. We can actually rank the concepts and potentially throw away the less important ones so that we can store our data more efficiently. And also we will see that there's a strong relationship between damage and error reduction or throwing away information to compression and also recommendation.

So one thing to note is, I keep saying that SVD is a very powerful and generalizable technique. The reason is all it takes is a matrix of numbers, so it doesn't really care how you generate that matrix. For example, what we saw previously was, each row was a customer and then the columns will be like grocery store item. But that's not stopping you using auto-matrix, let's say if you are like Pandora, for example. Then your data would be customers and songs, so the music columns now can be songs. Similarly, let's say if you're a Netflix fan then your columns can be movies, so that means there is really anything you can use. And no matter what matrix you put in, it's able to help rediscover these topics or groups of concepts, so for that reason it's really, really powerful.

So now we can dig deeper into what SVD can do. We'll start with a pictorial description of SVD and then we'll look into it in a more technical sense.

So at the high level, SVD, what it does is decompose our data matrix A, so in the red box up there. However, A matrix, which is n by m, so let's say that we're looking at text documents. So n is the number of documents and m is number of terms or words among these documents. So that's our input matrix, A. And what SVD can do is it can decompose this matrix into three smaller matrices, U, lambda, and V. So the first matrix, U, is n by r, so n, still the same number of documents, r is what LSI would be able to give you, so it's r number of concepts. And also similarly, the V matrix is r by m, so r again is the number of concepts and m is the same as our original data matrix, so the number of terms. Also interestingly, what we haven't talked about so far is the middle matrix which is r by r. So r is the number of concepts, and the very interesting thing about this matrix is it's a diagonal matrix. So that means only the diagonal entries are nonzero. And also, not only that, the values along the diagonal actually encode the concepts' strength. So, also they are ordered, so the concept with the greatest or strongest strength will be at the top left and the one that has the smallest strength would be at the lower right. So that means you have a ranked list of concept strength going along the diagonal.

So putting in words, that means that we have our A matrix, n by m, and say n documents, m terms. And then we decompose that into three small matrices U matrix, n by r, and say n documents, r concepts. A very special lambda matrix, r by r, so the number r is actually also the rank of our original matrix A. So actually one pretty good use of SVD is we try to compute a rank or the effective rank of our original matrix. And also we notice that these r entries are ranked in their order of strength. And thirdly, we have our V matrix, which is m by r, m terms, r concepts.

There's actually a theorem that says, it's always possible to decompose our data matrix, A into three smaller matrices, and they are unique most of the time. And not only that, U and V, the columns in those matrices, are also normal, so meaning the columns are unique vectors and they are also orthogonal to each other. So the natural outcome of it you do transpose U, it'll five you an identity matrix, and similarly for when V-transpose V, it will also give you an identity matrix. And we said before, lambda would be a diagonal matrix with non-negative diagonal entries and also interestingly sorted in decreasing order. You will see why this order is important very shortly.

So let's look at a numerical example of what SVD can do, here the rows are documents and columns are words in a document. So cell values are word counts, but as you may recall it can be anything, It can be like 0 and 1 to express existence of words. Or they can be TF-IDF scores. So here we're just using word count for simplicity.

So let's look at the matrices and their cell values and understand how to interpret them. So let's check their dimensions, the first matrix is the U matrix, has a number of rows that's the same as the number of documents. And also, number of columns here is the concepts. And there are two concepts because this is actually also the rank of our original matrix, which if you are just eyeballing it, you know the rank is 2. And then we have the middle matrix, a lambda matrix, so its r by r, so since r is 2 its also 2x2. And the third matrix is the U matrix, it's the same number of words as in our original matrix, so 5, and then number of rows equals the number of concepts, which is 2. So those are the dimensions, so they make sense, all of them, they all add up.

And next we can look at the values. So in the U matrix, the first concept or the first column. So if you look at the values, the values seem to scale somewhat proportionally with the word counts in our original matrix, so we say 0.18, 0.36, and so on. And this seems to make sense because, for example, let's say for document 4 which is row 4 in our original matrix, the word "data" appears five times. And "information" appears five times, "retrieval" five times, and so on. And in our U matrix, you will see that we have a 0.9 value, which is the biggest value in that column. And it seems to make sense, it correlates with the word count in our original data matrix. So that's a very possible thing that we observe, meaning that particular document, document number 4 is strongly associated with the CS concept.

So similarly, in our context, we can interpret the matrix U, the document concept similarity matrix in a similar way. So here we have a higher number, that means it's more similar, or more strongly associated with that concept.

And for the second matrix, the lambda matrix, here we can see that that is a diagonal matrix, and also the diagonal entries are nonzero. And also we see that they're sorted, so in 9.64 that's the strength of the CS concept. And 5.29, that's a second one which is a smaller concept for the medical documents.

So there's another way to interpret these matrices, the first one, the U matrix, we can call it a document concept similarity matrix. So that means the values here encode similarity, so higher the value the more similar a document is to a concept. And similarly, for the V matrix, we can also call it a term concept similarity matrix, so the bigger the value, the more similar a term is to a concept.

In this video we learned about LSI and also its underlying mathematical tool SVD, singular value decomposition. And we mentioned that it's a very generalized software technique because they only requires the input to be a matrix. And it can automatically discover concepts or themes in your data.

SVD: Dimensionality Reduction, and Other Uses

We are going to continue with text analytics and focusing on SVD. So previously we learned about SVD being a very powerful generalizable technique. So today we're going to look at what are the other uses of SVD, such as doing dimensionally reduction and we'll also look at a case study of how we can use SVD.

So let's first do a quick recap of what we have learned so far about SVD. So we decompose our original matrix, A, into smaller matrices U, V, and lambda. And in the context of text analytics, we have the U matrix, which is document-concept similarity matrix. V is a term-concept similarity matrix, and also lambda, which is that diagonal matrix with a concept strength as the values. So the matrix U and V actually have some interesting relationships to our original document term matrix A.

So specifically, if A is the document-to-term matrix then A-transpose A is the term-to-term similarity matrix, and similarly, A A-transpose is the document-to-document similarity matrix.

So by computing SVD on our original matrix A, we get the matrix B, and very interestingly that actually contains eigenvectors of A-transpose A. So similarly, U contains the eigenvectors of A A-transpose.

So what does all this mean? It means that SVD is closely related to PCA, or Principal Component Analysis, a very popular dimension reduction method. Because in PCA, we need to compare the eigenvectors of A-transpose A. And multiplying A-transpose A is actually not a very good thing to do because it can lead to loss of precision. In other words, SVD allows us to compute eigenvectors needed by PCA without having to actually materialize, explicitly multiply A-transpose A. If you want more information about SVD, PCA, and also their relationship, I highly recommend you check out the link below and also the excellent book.

[[https://math.stackexchange.cx)m/questions/3869/what-is-the-intuitive-relationship-between-svd-and-pca](https://www.google.com/url?q=https://math.stackexchange.com/questions/3869/what-is-the-intuitive-relationship-between-svd-and-pca&sa=D&source=editors&ust=1738598889318273&usg=AOvVaw1NTIplDZieVhTvD6Eue7vS)]

[lanT. Jolliffe. Principal Component Analysis (2nd ed), Springer, 2002. Gilbert Strang, Linear Algebra and Its Applications (4th ed), Brooks Cole, 2005.]

So another way to interpret SVD is that SVD can help you find the best axes to project on. So best here is characterized by how well it can minimize the sum of square of projection error. So in other words, SVD can help us find, let's say, the first singular vector, v1, and v1 will be pointing along the direction of data that has the greatest variance.

And the variance will be captured by the lambda matrix. So specifically, let's say, for v1, which is the first singular vector, the variance or the spread would be captured by its concept string, which is 9.64 in the lambda matrix.

So in other words, if we multiply U and lambda, those two matrices together, it will actually give us the coordinates of the points on their projection axes, in this case v1 and v2.

So, by understanding this interpretation that SVD is actually helping us find the best axes to project our data on, now we can actually figure out why we can use SVD to do dimensionality reduction. Right? For example, instead of, say, two concepts, in the same way you're seeing on this screen, we can actually use the values in lambda matrix to guide us. For example, if say we have a gigantic matrix with millions of rows and billions of columns. Right? So how do we compress this? Or how do we extract the most important, most relevant concepts of themes in this data so that we don’t need to store the whole matrix?

So the way that we can use the lambda matrix is we can look at the strength. So remember that these values in the lambda matrix, they are sorted. So the one with the greatest strength is at the top left. The that has the smallest strength is at the lower right.

So what that also means is we can actually start setting the values that are the smallest to zero. Like going from the bottom right, and then gradually going to the top left. Right? So that means however many concepts that we want to keep, so those are the diagonal entries that we want to still keep that original values. And all the other diagonal entries, we just set them to zero. And by setting those to zero, what that also means is we're essentially zeroing out the corresponding vectors. So if you do the multiplication, you will know that we don't care about those vectors any more, because as long as the values, the concept values in the lambda matrix are set to zero, the other ones, we don't really care what they are.

So essentially, that means we are cutting out all those columns, those rows from our view and the matrix.

If you multiply those three truncated matrices together, you will end up getting a matrix that has the same dimensions as our original data matrix. But you will know that here on the lower right, so all those non-zeros in our original matrix now become zeroes. So that is how dimensionality reduction is done. So by setting the concepts to zero, that we don't really care about, ensures you're removing information in original matrix that are not as important, and essentially compressing our data, throwing away things that we may not care about so much.

So the time complexity to compute SVD exactly is O(n\*m\*m), or O(n\*n\*m), so whichever is smaller. But in practice, it's very rare that you want to solve this exactly. And in practice, often instead of asking for finding all concepts, which is equal to rank of the matrix, often we'll say, for the million, times billion matrix, maybe we only want the top 50 or top 100. And in practice, that already probably more than enough, right? So imagine, let's say you're Netflix and you want to find out how many groups of users there are. Maybe a couple hundred's already more than sufficient. So what they also mean is that instead of computing all the singular vectors, you may only want the first case singular vectors. So these would be much faster to compute. So, and also if you matrix in practice most likely it is, that means a lot of zero, that is even faster to compute.

A very nice thing about SVD is because it's such a powerful and generalizable technique, you don't really need to write the algorithm yourself. So it's available in almost all linear algebra packages. So that means you can just use them.

So now that we know how SVD works, and also numerically we looked at a few examples, so let's look at some case study. For example, how we can do queries using LSI, or using SVD.

All right, for example, let's say I bring back our documents and terms context. So how do we find documents with the word, say, "data"?

And you may recall from very early on, our main idea is to first try to map our data query, right, our query consisting of one word, "data," to map this into the concept space. Right, so how do we do that?

So first we would want to represent our query, the work data, as a vector, so essentially putting into the vector space model. So that means now we have five words, so we would set the first entry to 1 for data, and then all the other entries to 0. So that means I have a four [?] dimension vector, and to map this vector or our query in to concept space all we need to do is to do the inner product. So our cosine similarity between our query vector q and also each of the concept vectors like v1 and v2, so that means q dot v1 we have one value, and also q dot v2 another value.

Right, so in mathematical notation that means q times V, and then we have the q concept vector. Right, and below on this screen you'll see that we now have the query map to concept space. So 0.58 and 0, so what that means is that now the query data or query vector, and it is strongly associated with the CS concept.

So really the question, so how would we handle the document and say it contains the word information and retrieval?

So actually using the same approach. So we'll also first convert our document into vector. So again, five words, so dimension equals five. And we set the entry for information and retrieval to 1, and then all the other entry to 0. And we do the same thing, we do the dot product, so that means our document times our V matrix. And we get the document concept vector, so again here we will see that we have the strength of 1.16 for CS concept, and 0 for the other medical concept.

So the important observation here is that we started off with the query which contains only one word, data. However, by expressing everything into vectors, including the documents or the documents and also mapping all of them into the concept space. So now we can actually query or find all these documents that may not even include the word data, as in the example shown below. So our query includes only data, it doesn't contain information and retrieval. But by mapping that query vector into concept space, now we have the vector 0.58, 0, that is association in terms of the CS concept. And we can use this notion, now we can pull out all the documents that are strongly associated with the CS concept. So as an example below, even though our document did not contain the word data we can still get it.

So in this video we look at SVD as add application, such as doing dimensional reduction by simply zeroing out the concept LS important. And also how it can help us with retrieval, such as finding documents that may not essentially contain some word that we're interested in.

**Text Visualization**

We are going to continue with Text Analytics, and today we are going to look at a few popular ways of visualizing text and text documents.

So a common way to visualize text is using word or tag cloud. It is very popular because it's a very visually interesting way to look at the data, and although it may not be very informative. A typical way is you scale the word by its word count or popularity in your document. So the larger it is, the more times its appeared in. So, this is probably the most that you can get out of word cloud. So, you don't really get any kind of structural information like word ordering and so on, so the only thing that you get is mostly word count.

A variation of it is the bubble chart, so instead of showing the work, now you scale the bubbles instead of the word count. So small variation, also very popular, because you can look at it, as in this example pretty interesting to look at.

So there are some are some other visualization techniques that can review more of the structure in your documents. For example, another popular way is Word Tree. Word can show you the hierarchy of the sentences in your document. For example here we're showing all the sentences that start with the what we, and then you can see it gradually. As you move to the right, you can see how many of those sentences, the second word is must. How many of those sentences among that we contain the word act, and so on. So essentially, you are looking at a hierarchy that are built from your document collections.

Another way to visualize text documents is using something we call our Phrase Net. So, instead of hierarchy or tree, now we built a graph out of our document. So specifically we look at our relationships among words. So let's say in this example, we're interested in the trigram X and Y, so that means word number one separated by word n and then word number two. So that means that words in our graph are the words that you will see in a document. And then exists in this graph, let's say the trigram father and mother. And then we'll see it has a really thick arch, connecting father and mother in our example, that means there are any such diagrams that we cad define in our document. And of course you can define the arch anyway you want, so here in our example we're saying X and Y but you can also say X of the Y or X the Y and Y and so on, or even X base Y. So that means that defining different meaning for the arches, you can have gathering different phrase nets of your documents.

So you've seen this example before, which is a termite system and this is a canonical way of visualizing documents and topics. So here each word would be a row and a topic would be column. So I can imagine these columns are what you would get let's say, from SVD. And the cell values could be the values that SVD would give you. So essentially saying, let's say for topic 17, which are the words that are most strongly associated with it.

And you may recall that whenever you visualize any kind of matrices, the ordering of the rows and columns are important. On the left is without any sort of ordering, and on the right is with ordering. Something called seriation, so the columns and rows are grouped based on column location.

In this video, we looked at a few popular ways how visualizing text such as using matrix, using trees and graphs.