FILTERING & AGGREGATION

Miriah Meyer
University of Utah

slide acknowledgements:
Tamara Munzner, University of British Columbia
administrivia
feb 14-23 : proposal meetings

march 7 : presentation topics due

march 9 : proposals due

march 27-april 3 : project updates

april 5-24 : paper presentations

may 1 : final project presentations

may 3 : process books due
THE PANCAKE RECIPE CHALLENGE
Question: What are the most common extensions to the basic pancake recipe?

Data:
- Web Text
- Ratings
- Descriptions/Comments/Reviews
- Ingredients

Transformations:
- Web Text Normalization (e.g., 5 stars vs. scale 1-10)
- Natural Language Processing (e.g., find words like "low-calorie", "dry", "thick", "moist", "light")
- Normalization for no of servings

Derived Data:
- Ordinal table of ratings
- Categorical table of tags
- Quantitative table of people that tried it
- Quantitative table of ingredients

Machine Learning, PCA, etc. to find clusters of recipes with similar ingredient distributions

Select most popular recipe within each cluster - these are the "basic" recipes
Also select most common tags for each cluster

Categorical table indicating a recipe's "root" recipe, and its descriptive tags
Rationale: If specific ingredients are important, this visualization might help.

Interface 3:

- Symbol: colored red when clicked
- Clicking arrows or the symbol will bring up the recipe with same as web page.
- Selecting a recipe in the table will update the box chart.

Ingredients:
- Butter
- Sugar
- Salt
- ... ... ...
- Nuts

Basic recipe:
- Mousing over the title brings up the web page containing the recipe.
The data types we will work with include tabular and text data. The rows in the table will correspond to a particular type of pancake. The columns will include:

1. Name of the pancake  
2. Type of ingredient  
3. Quantity of ingredient  
4. Unit of measure  
5. URL to recipe

categorical data attribute  
categorical data attribute  
sequential quantitative, abstract, static, distinct  
categorical data attribute  
text/label data

Since ingredients for pancakes are finite, and probably sufficiently small, maintaining a column per possible ingredient will probably not be an issue. However, we can also store in a single column a compound data list type which includes all ingredients as well as the quantity of each.

In order to answer the challenge question, we must be able to ask if a pancake has a subset of the ingredients in our pantry and if for each ingredient if we have enough in our pantry to meet the ingredient quantity requirements of the recipe. The primary visualization task necessary is the filter task. We apply the first filtering technique which first checks if a pancake recipe only lists types of ingredients which we have in our pantry, and add these candidate recipes to a set $S$. The second filtering looks at each pancake in set $S$ and determines if for each required ingredient, if we have enough of that ingredient in our pantry, and if so adds the pancake to result set $R$. After obtaining $R$ we may wish to sort the pancakes in $R$ by total number of ingredients, or total quantity of ingredients, or some other user specified function.

Potential data transformations include the construction of an inverted index over ingredients. Treating the type of ingredient as a token, each token in the inverted index points to the set of pancake names which use the ingredient. In addition each pancake name could also be associated with the quantity necessary for the ingredient. To obtain candidate pancakes, we query the inverted index for each ingredient we have in our pantry. After finding candidate pancakes the problem boils down to determining if all ingredients in the candidate pancakes appear in the pantry. Since (homemade) pancakes do not typically have many ingredients this can be accomplished in a linear scan of ingredients for each pancake. Efficient string matching techniques using q-grams might also be considered to prune some pancakes from consideration. After finding pancakes with a subset of the ingredients in the pantry, the unit of measure of the required ingredient might be transformed to match that of the unit provided by the user. A simple comparison is used on the quantity of each ingredient to determine if we have enough.

The main classes of change which could be used for this analysis task include changing selection, changing highlighting, and changing view. The changing selection can be used to select and deselect pancake results. Changing highlighting is useful to show pancakes which satisfy the ingredient requirements, the pancakes which do not may be presented but not highlighted. Finally, by scrolling through the results we can change the view.
The primary mark types I have considered for my design are lines and bars, for line and bar style charts indicating quantity of ingredients. The primary visual channels are length and color highlighting for indicating quantity of ingredient as well as pancakes which contain all ingredients respectively. The primary mechanism interactions are: 1. user input forms, which allows users to change selection by inputting ingredients; 2. Selection of pancakes in results through mouse-over and clicking; 3. Scrolling through the results with mouse-over to obtain a synopsis of information before selecting a pancake; 4. Sorting results based on some user specified function, e.g. quantity of ingredients, total distinct ingredients, etc.

My favorite design is C on the attached sketch paper. The primary markers are bars to indicate quantity of each distinct ingredient appearing in a pancake. There are also images of the pancake to drive user interaction and images and the ingredient bar charts are grouped. This design requires users to specify all ingredients and no quantities. This comes from reasoning that a user may what ingredients are in the pantry, but not precisely how much. After finding an appealing pancake(s) they can further refine and filter by entering quantities. Otherwise they can manually inspect the ingredient list.

In brief:
1. Bar charts allow users to quickly check quantities of ingredients
2. Grouped image groups image of pancake with name and bar chart to appeal to user
3. Interactive ingredient search specify ingredient without quantity
4. Refined search specify quantities or maximum amount of ingredient
5. Scrolling capability user can scroll through results
6. Sorting allow user to sort based on user defined function
7. Highlighting potential after refined search, highlight pancakes which match
Jeffrey Jestes
Abstract:

- Data types and their attributes:
  - Ingredients: Tabular type
    - 2 tsp sugar, salt
  - Categorical
  - Quantitative

- Recipe Instructions: Ordinal type (e.g., step 1, step 2, etc.)

- Original Rating: Ordinal
  - @@@: 3/5

- Cost: Ordinal [if scaling, from $0 to $15]
  - Quantitative [if fixed, $10, $20, etc.]

- Health: Quantitative [in kcal, e.g., 190 kcal/serving]

- Ease of Preparation: Ordinal [if scaling or classes]
  - Easy, Medium, Hard

- Time: Quantitative [e.g., 15 mins, 45 mins]

- Recommendations: Categorical
  - Facebook, Twitter

- User Tags: Categorical
  - Fast, Cheap, Vegan, etc.

Nikhil Mishrikoti
SUN: Ordinal data based exploratory tool

1. SUN - Pie-Bar graph visualization of Recipes
2. Ordinal data Selection/Filtering menu
3. Zoom level adjustment
4. View Selection - Prev/Next [Past/Latest]
5. View Selection - Up/Down [Stages in Hierarchy]
6. Overview of filtered data in SUN

Nikhil Mishrikot
All recipes available to us are arranged by their overall rating, from 0 to 5.00. e.g.: 3.9/5, 4.5/5 etc.

1. Ratings increase in clockwise direction along the circumference of the sun.

2. Indicates the total amount of recipes being viewed across all ratings.
Number indicates the number of recipes in that rating range.

- 150 recipes with rating between 0.0 and 1.0

Length of arc proportional to this number

Length $\propto$ 150/157.6

Length of the bar graph indicates the number of recipes with rating indicated on the circumference.

Does not represent accuracy, indicates shape to find patterns of interest.

Shaded pie-bar graph indicates selection during mouse hover.

Clicking on the region deepens or proceeds
1.1) Clicking on the pie-bar slice in the SUN
- Overview: the slice is rotated to align with vertical axis (animation to preserve context)
  if then proceeds to the zoom step.

- Zoom: The bar graph is then zoomed into slowly (animation) if the axes are straightened and, with a legend menu showing recipes.
Nikhil Mishrikot

**Recipes**
- Butter Pancake
- Blueberry Pancake
- Orange Pancake
- Cranberry Pancake
- etc... Pancake

Arrows used to change ratings range on y-axis if graphs are too crowded.

If a particular recipe chosen from menu, then data to note that recipe name.
- Clicking on a bar, again follows the overview of zoom & filter procedure.
- Overview of zoom: The colored blocks slowly move away (animation to preserve context) & form a new bar graph.
- Filter: The bars are arranged by recipe names/ colors shown in menu.
LAST TIME
LINKED VIEWS
multiple views that are simultaneously visible and linked together such that actions in one view affect the others
- linking choices
  - linked highlighting
  - linked navigation
view choices
- encoding: same or multiform
- dataset: same or small multiple
- data: all or subset (overview/detail)
- conditioning
today
comments on readings?
DATA REDUCTION

- **how to reduce the amount of stuff to draw?**
  - crosscuts view composition and interaction considerations

- **item reduction**
  - today
  - rows of table

- **attribute reduction**
  - Thursday
  - columns of table
ITEM REDUCTION METHODS

- filtering and navigation
  - leave some things out

- aggregation
  - merge things together

- overviews
  - temporal through navigation
  - separate dedicated view
  - focus + context
    - selective filtering
    - geometric distortion
    - distortion costs/benefits
FILTERING
NONSPATIAL FILTERING

Ahlberg 1994
SPATIAL FILTERING: NAVIGATION

- **filter based on spatial position**
  - unconstrained: camera can move anywhere
  - constrained: limit on possible motion

- **panning | translating**

- **rotating (in 3D)**

- **zooming**
  - geometric: analogous to real-world
    - appearance fixed, viewpoint changes sizes of objects
  - semantic: representation adopts to screen space
AGGREGATION
-combine items (vs. eliminate them with filtering)

-derived attributes
  - min | max | avg | sum
  - many many many other ways!

-challenge: avoid averaging out signal
OVERVIEWS
- **strategies: both filter and aggregate**
  - simple: geometric zoomout
  - complex: aggregation

- **methods**
  - temporal through navigation
  - separate dedicated view
  - embedded/integrated focus + context
- **taxonomy**
  - overview + detail: spatial separation
  - zooming: temporal separation
  - focus + context: integrated/embedded

- cue-based: selectively highlight/suppress
  - *crosscuts other techniques*
OVERVIEWS
overview + detail
OVERVIEW + DETAIL
OVERVIEW + DETAIL

http://www.historyshots.com/rockmusic/
OVERVIEW + DETAIL ISSUES

- **linked navigation**
  
  - shortcut navigation: thumbnail to detail
    - *leaping to far-away regions*
  
  - explore overview without changing detail (or vice versa)
    - *full synchronize does not support*
  
  - detail changes immediately shown in overview

- **their definition: lens at O+D**
  
  - O and D separated in z-depth (layered)
  
  - Munzner’s usage: focus + context
LENSES

Cockburn 2008

Lambert 2010
OVERVIEWS
zooming
GEOMETRIC ZOOMING
ZOOMABLE USER INTERFACES
SEMANTIC ZOOMING
ZOOMING ISSUES

- **geometric zooming**
  - hard to make intuitive zoomout control
  - zoomable user interfaces (ZUIs)

- **semantic zooming**
  - different representations at different scales

- **challenge: stability**
OVERVIEWS
focus + context
FOCUS + CONTEXT

-integrate focus and context in a single view

http://www.cs.umd.edu/class/fall2002/cmsc838s/tichi/fisheye.html
F+C FORMALISM

\[ \text{DOI}_{\text{fisheye}}(a | . = b) = \text{API}(a) - D(a, b), \]

where

1. \( \text{DOI}_{\text{fisheye}}(a | . = b) \) is the degree of interest in \( a \), given that the current point of focus is \( b \).
2. \( \text{API}(a) \) is a static global value called \textit{a priori importance} at point \( a \); API values are preassigned to each point in the structure under consideration, and
3. \( D(a, b) \) is the distance between point \( a \) and the point of focus \( b \).

- DOI for selective presentation vs distortion
- infer DOI through interaction vs explicit selection
- single vs multiple foci
F+C ELISION: SpaceTree

selective filtering with elision

Grosjean 2002
F+C DISTORTION: 3D

Cockburn 2008
F+C DISTORTION: FISHEYE

Furnas 1995
DISTORTION CHALLENGES

- unsuitable for relative spatial judgements
- overhead of tracking distortion
- visual communication of distortion
  - gridlines, shading
- target acquisition problem
  - lens displacing items away from screen location
- mixed results compared to separate views and temporal navigation
- fisheye follow-up: concern with enthusiasm over distortion
  - what is being shown: selective filtering
  - how it is being shown: distortion as one possibility
F+C WITHOUT DISTORTION

magnification

highlight | suppress

Lambert 2010
L11: Dimensionality Reduction

REQUIRED READING
Attribute Reduction Methods

The previous chapter approached data reduction in terms of reducing the number of data items to show. This chapter covers attribute reduction, where the goal is to reduce the number of attributes that are displayed by eliminating or aggregating them. The distinction is easiest to understand when we consider tabular data. Item reduction corresponds to eliminating rows, whereas attribute reduction means eliminating columns. Of course, these two approaches are not mutually exclusive: both item and attribute reduction could be used in conjunction. Some attribute reduction methods also implicitly eliminate items, as discussed below.

There are many ways to reduce the number of displayed attributes. Camera metaphors naturally lead to slicing, cutting, and projection. Unwanted attributes can simply be filtered away so that only a subset of the attributes are shown, or aggregated so that one representative attribute is a stand-in for all those in its group. A large body of techniques attempt to preserve most of the information in the dataset originally specified through many attributes with a more compact representation using far fewer attributes. These are usually called dimensionality reduction techniques; while this book has used the term attribute thus far, in this chapter the term dimension will also be used in keeping with this terminology. Both linear and the more complex nonlinear mappings can be used to create new synthetic dimensions that capture the important content of the dataset more concisely. Common visual encodings for dimensionally reduced data include scatterplots, scatterplot matrices, and spatializations.

8.1 Camera Metaphors

One set of methods for reducing the number of attributes is based on geometric intuitions about manipulating a virtual camera. Given this foundation, the terminology is with the attribute name in this section.
Glimmer: Multilevel MDS on the GPU

Stephen Ingram, Tamara Munzner, Member, IEEE, and Marc Olano, Member, IEEE

Abstract—We present Glimmer, a new multilevel algorithm for multidimensional scaling designed to exploit modern graphics processing unit (GPU) hardware. We also present GPU-SF, a parallel, force-based subsystem used by Glimmer. Glimmer organizes input into a hierarchy of levels and recursively applies GPU-SF to combine and refine the levels. The multilevel nature of the algorithm makes local minima less likely while the GPU parallelism improves speed of computation. We propose a robust termination condition for GPU-SF based on a filtered approximation of the normalized stress function. We demonstrate the benefits of Glimmer in terms of speed, normalized stress, and visual quality against several previous algorithms for a range of synthetic and real benchmark datasets. We also show that the performance of Glimmer on GPUs is substantially faster than a CPU implementation of the same algorithm.

Index Terms—Multidimensional scaling, multilevel algorithms, optimization, GPGPU.

I. INTRODUCTION

MULTIDIMENSIONAL scaling, or MDS, is a technique for dimensionality reduction, where data in a measured high-dimensional space is mapped into some lower-dimensional target space while minimizing spatial distortion. MDS is used when the dimensionality of the dataset is conjectured to be smaller than dimensionality of the measurements. When dimensionality reduction is used for information visualization applications, the low-dimensional visualizations should reflect the structure of the data.

MDS algorithms vary in precisely what form the stress function takes and in how they minimize the stress function. Some are approximate while others are exact, some are iterative while others are completely analytical. Such diversity in algorithms leads to diversity in the quality of the results and the speed at which they are computed. Section II gives a brief overview of various relevant classes of existing MDS algorithms and their underlying characteristics.

One class of MDS algorithms that has had significant influence in information visualization is the class of iterative, force directed algorithms. In such algorithms, data points are modeled as particles in space attached to other particles with springs with an ideal length proportional to the original distance \(\delta\). The algorithm computes a simulation by integrating forces until the physical system settles down into a state of minimal energy. At this point computation halts and the final positions of the particles are assigned the resulting coordinates of the data. Naïve implementations of such algorithms can be computationally expensive and prone to converge to local minima.

We present three substantial improvements to the iterative class of MDS algorithms based on simulated forces. First, we improve algorithm speed by exploiting the modern PC graphics processing unit (GPU) as a computational engine. Second, we introduce a