Wranglr

Executive summary

Searching through journal articles is tedious and the current convention of filling in forms for keywords and authors requires multiple searches and refreshes. Wranglr uses a recommendation engine and an interactive web interface to actively explore literature. A prototype using a database of geophysics articles will be demonstrated.

Personnel

Ben Bougher (ben.bougher@gmail.com)
Principal engineer
Ben holds a physics degree from Dalhousie University and has spent 5 years in various research and software development roles. He is currently an MSc candidate at UBC studying machine-learning applications in seismic imaging and moonlights developing cloud-based geophysics software.

Task

Literature search engines are well adept to specific queries, where the user has a narrow search criterion. The use case that motivates Wranglr is finding relevant articles with uncertain or "fuzzy" search criteria.

Current scenario

The common workflow of finding articles with fuzzy search criteria is shown in Figure 1. First, the user searches using a best guess at keywords. Upon scanning the results, they are either happy with what they have found or they adjust their criteria. The new keywords are used to perform a refined search and this process is iterated until they are satisfied. This scenario requires page refreshes and form re-filling, which discourages exploration.

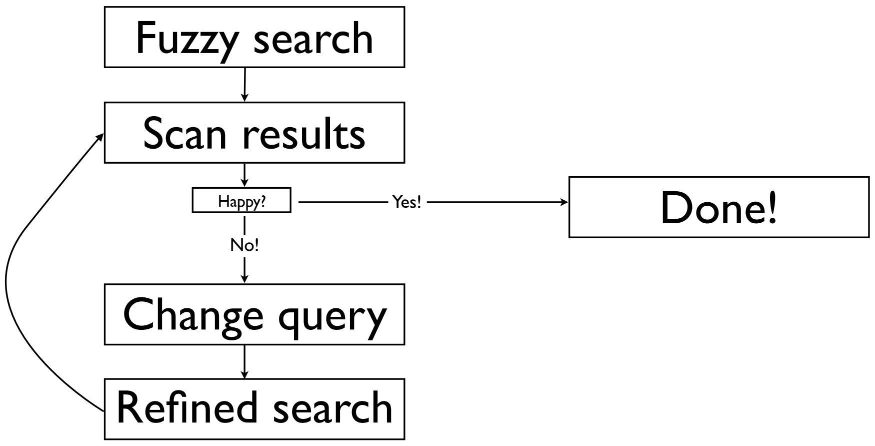
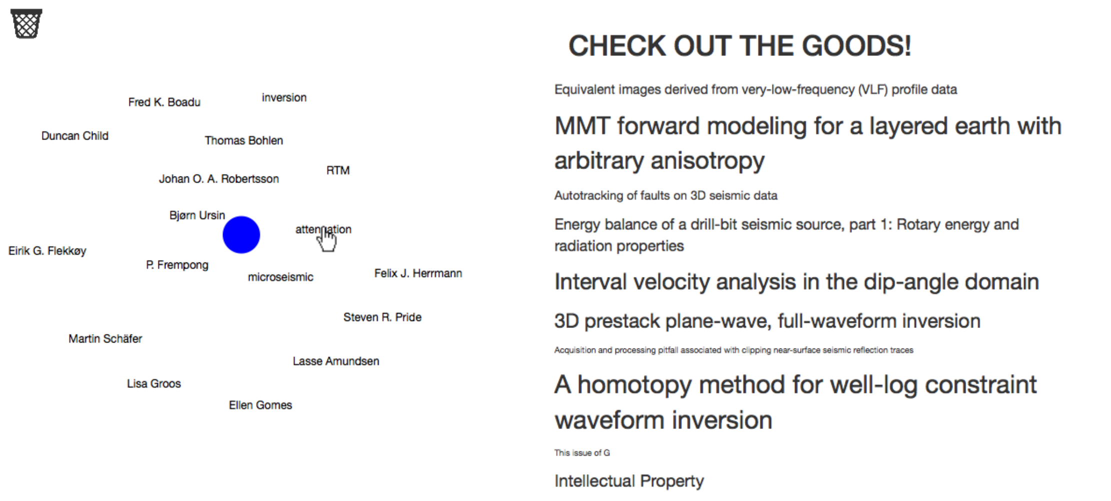


Figure 1: Fuzzy searching scenario using conventional journal searches.

InfoViz approach

Wranglr aims to offer a visualization approach, where it applies principles of information visualization to augment the search capability of the user. The articles in a journal can be considered items, containing attributes such as associated keywords and authors. Previous work has modeled journal databases as a network of citations, but alternatively Wranglr forms clusters of articles based on similarity measurements of their attributes. Modeling the data as items, attributes, and clusters suggests an idiom involving tables or lists. An example of the proposed interface is shown in Figure 2.



CHECK OUT THE GOODS!

Equivalent images derived from very-low-frequency (VLF) profile data

MMT forward modeling for a layered earth with arbitrary anisotropy

Autotracking of faults on 3D seismic data

Energy balance of a drill-bit seismic source, part 1: Rotary energy and radiation properties

Interval velocity analysis in the dip-angle domain

3D prestack plane-wave, full-waveform inversion

Acquisition and processing pitfall associated with clipping near-surface seismic reflection traces

A homotopy method for well-log constraint waveform inversion

This issue of G

Intellectual Property

Figure 2: Prototype interface for Wranglr. The left part of the display shows a cluster of attributes (authors and keywords) and on the right is a list of items (articles). The user ranks the importance of the attributes by moving them closer or farther from the center, which dynamically updates the list of items. This interactive interface encourages exploration of the search space.

The new scenario of fuzzy searching becomes much more exploratory. The initial fuzzy search brings users to the interface in Figure 2. The user can move and delete attributes based on perceived importance, which will dynamically update the list of items. The user can explore the database interactively without re-filling forms or refreshing the page.

Wranglr does not encode the data itself but instead encodes metadata about the relationships between items. The dimensionality of the data is relatively small, which allows Wranglr to take full advantage of the spatial channel with minimal occlusion and clutter. The user-perceived importance of the attributes is encoded in the spatial location. The closer to the center the user places an attribute determine the attribute ranking used to update the items. Ordering of the items encodes the search query ranking.

Additional abstraction uses size to encode the correlation between attributes and items. For example, hovering over an attribute changes the text size of the items based on the relevance of the specific attribute. This is a two-way mapping, as hovering over an item changes the attribute text size.

Wranglr will use embedding to manage additional attributes such as abstracts and DOIs, but specific display idioms are still undecided.

Data

Although this idiom can be applied to any article database, recent work at a geophysics hackathon scraped the Journal of Geophysics and performed rudimentary machine learning and feature extraction. Text processing and clustering on the articles extracted a set of keywords and authors. Scoring each keyword and author in an article forms a feature vector. For example, first author gets a higher score than a reference or cited by, and a keyword appearing in a title receives a higher rank than appearing in the abstract.

The articles can be ranked based on a query of authors and keywords. Weighting the query attributes allows for biasing the search ranking.

Implementation

The Wranglr prototype will be public facing web app served on Google App Engine. The backend and machine learning aspect of the app will be written in Python and the interactive user interface will be on top of D3.

Project plan

Much of the project work has been heavily front-loaded. Basic machine learning algorithms have been developed, and the data with feature vectors is in an SQL database on Google Cloud Services. An initial frontend framework has been developed in D3. A prototype full stack application is currently serving at https://geophyzviz.appspot.com/ and the project code is available at https://github.com/ben-bougher/GeophyzViz. The current application has a place holder for the search ranking (AKA random number generator).

Search ranking integration ~ 2 days of effort, medium risk
The search-ranking algorithm needs to be integrated into the app. This is a medium risk task as the algorithm might slow down the required interactivity of the application.

Frontend design ~ 5 days of effort, low risk
This task requires final decisions on encodings and idioms, as well as an aesthetically pleasing interface.

These first two milestones need to be completed by 30 November to allow time for a short validation study before the project deadline.

Previous work

This project shares a very similar task to PaperQuest, but takes on a different approach. PaperQuest modeled the articles as a network and used symbols to encode the data. This approach does not abstract the data, and instead uses spatial encodings to abstract learned relationships and clusters between article features.