John Dimm

San Diego, 2025 https://johndimm.vercel.app/ http://www.linkedin.com/in/johndimm

Full Stack Lead and Data Engineer

Highly effective in roles ranging from SQL maven to full stack developer to hands-on manager to individual contributor.

Skill Summary

- MySQL, PostgreSQL, Lovefield
- SQL query optimization
- React and nextjs for front-end development
- Serverless function apps on Azure
- Data Pipelines and ETL processing
- Amazon Alexa Skills
- Google Chrome Extensions
- Recommender systems

Highlights

Time lapse for Gardyn

Platform: Azure, python, react/nextjs, ffmpeg

The Gardyn device has cameras that capture a photo of your plants every 30 minutes. The goal of the project was to create a time lapse video of plant growth, using selected images from that stream hosted on Azure. The first version did all the work on the server in a daily job using ffmpeg.

I had an idea of doing it in a simpler and more flexible way. I made a react/nextjs proof-of-concept webapp to create a video-like experience but without actually creating a video. Instead, it shows the set of selected photos one after another in the same html img element, producing a slideshow or flipbook. It was not obvious that this approach would result in a smooth animation, but it did, and the feature allowed users to select their own time range. The same approach worked on the mobile app, with moderations.

I later added a video-generation feature to allow sharing, this time done on demand rather than in a nightly job.

Acculitx Driver Score Portal

Platform: AWS, Python, MySQL, PHP, celery, javascript, angular, react, auth0
With: Peter Ellegaard (physics)

Accuscore generates a FICO-like score for driving behavior using accelerometer data gathered by an on-board device or cellphone. The prototype analytics system I inherited was inefficient, making the common mistake of using the database only for storage, passing large amounts of data back and forth between the database and the application, and often looping through many thousands of records in python. Working alone, I replaced 90% of it, both front and back-ends. The analysis is now done in MySQL stored procedures, data transfer is a fraction of what it was, and there is a clear path for scaling up.

On the analysis side, the summarized raw physical measurements formed an unwieldy distribution, and some time had been spent trying to tame it. Playing with the data in R, I found that the data is log-normal – the log of the same readings very clearly form a normal curve. We were able to reduce six dimensions to a single score by computing a simple weighted average, and the result is again a normal curve.

Web Speech API Experiments

Platform: Google Chrome, jQuery, Systran/Google/Azure translation, WebRTC, Alchemy API, YahooBOSS, flickr API

When I was managing the team of computational linguists at Systran, the Web Speech API was released in Google Chrome with support for continuous speech recognition and I wrote several apps to make use of it. The **Translating Telephone** was based on the open source video conferencing system WebRTC. Each participant speaks in their own language and the other users see the video feed with subtitles in their own language. Translation was done by Systran, Google, or Azure. **TalkShow** is an experiment in "ambient computing" – it runs in the background while you talk, converts speech to text, analyzes the text looking for noun phrases and proper nouns, does live queries against Flickr and Bing, and displays a slideshow of possibly relevant images. **Fun With Speec**h is a set of minimal demos of several speech and language related features and was the basis for my talk at the 2014 jQuery conference.

Veoh Video Recommendations
Platform: MySQL, Oracle, Hadoop, pig
With: Alexander Sherback (SQL guru), Ted Dunning (scientist)

Users often upload videos without a good description, making them invisible to standard text search. It turns out that a system designed to give recommendations for related videos also solves the search problem.

People rarely bother to provide ratings on videos. One key insight in the new recommendations system I designed and implemented was to use passive viewing behavior instead of explicit ratings. The longer the viewing session, the more likely the user was really interested in the video. Brief viewing sessions were rejected, since they are often the result of a misleading title or thumbnail. We skimmed off the top of our mountain of data to extract information reflecting true user interest.

The second insight was to simplify the problem by using binary ratings -- you either liked it or you didn't. That made it easy to construct the group of people who liked a given video, the first step in the algorithm. The second step, finding interesting facts about that group, used log likelihood to discover items disproportionately popular within the group compared to the general population.

Finally, we realized that the method of grouping and the method of finding surprising features of a group are independent. We could group people by video viewing behavior and use that to predict search queries. Or group people by search query and then look at the videos they liked. This last technique is formally the same as a search engine: the input is a query and the output is an ordered list of videos.

An interesting consequence of this search method: "Paco de Lucia" yields videos on flamenco guitar even though we had no Paco de Lucia at the time. That is because users who searched for him ended up watching other flamenco videos, and behavioral search makes use of that fact. Of course, the text-based search engine finds nothing for this query. Behavioral search can find relevant results even when there is no textual evidence.

We started with a naïve SQL solution running on Oracle and ended up with an elegant Pig script running on a 20-node Hadoop cluster.

Websense Explorer
Platform: SQL Server for Windows, MySQL for Unix, perl
With: Vince Star (developer) and Don Aymar (graphic designer)

Logs of URL access by a company's employees are summarized by date, category, user, and other dimensions. Websense has had a traditional batch-oriented template-based reporting program for several years. Reports had pre-defined columns and no links. The Explorer interface lets you select a value for one dimension and drill down by another in the same click. It was a game-changing product that gave instant information, a sense of free movement through the data, and showed users immediately the issues Websense was designed to address. It was touted as the flagship product of Websense for several years and was often cited as the deciding feature in a purchase.

Employment History

2022 – 2024 Gardyn Staff engineer

Gardyn sells an end-user device that grows a wide variety of plants hydroponically in your house or apartment. I took on back-end development of a python API to manage end-user membership and expose data uploaded by the raspberry pi processors in the field, and wrote Javascript and Typescript serverless function apps hosted in Azure to change light and water schedules through phases of plant growth. I designed and implemented a method to create time lapse flipbooks rendered on the client, and later, server-generated videos using ffmpeg. To enable both features, I wrote a multi-threaded python script to select and downscale images captured on the pi.

2020 – present Lean Software Development founder

A sole-proprietorship to support my consulting work.

2020 - 2021 Galigo CTO and co-founder

Eleventh Hour Global changed its name to Galigo and my title changed, although my work continued as before on the postgres backend of the g-engage. In this iteration, the app was about managing memberships for trade associations.

2020 Eleventh Hour Global Software Engineer

Eleventh Hour is a topical search engine for green products. The standard way to aggregate product information from a group of suppliers is to get a product feed from each of them and aggregate products into a central database. I developed an alternative that requires no effort from suppliers, implemented a rating and review system for products, and created a novel recommender system that does the analysis in postgreSQL.

2018 – 2019 Zuora Senior Software and Data Engineer

Zuora manages subscriptions for companies like 24 Hour Fitness, HBO, and Carbonite. Insights is a tool that aggregates data from 20 sources to present a complete picture of customers. I implemented the utilization model to track pre-paid usage and determine when a customer was likely to go into overage, and a test harness for integration testing. Later, I created an ETL process to help the accounting department during end-of-quarter reconciliation, taking data from two sources, loading into a local postgres database, and generating a combined report. Finally, I was in a 4-person team that created a new, simplified aggregator for internal use. The new pipeline was Python-based, and the front-end was javascript with Angular. I learned more than I ever wanted to know about subscriptions, rate plans, rate plan charges, and rate plan charge tiers.

2015 – 2018 Acculitx, Inc. Senior Data Engineer

Accuscore went through a series of promising customer evaluations, but it never caught fire. A year later, the system I set up was still processing data and generating reports, with no manual intervention.

2015 – 2015 AttackIQ Developer

AttackIQ is a stealth startup with a novel approach to enterprise security testing. I have worked on trend analysis and alerting using social media coverage of security issues, built with Angular, the Twitter API, and the Alchemy API.

2014 – 2015 Google Senior Software Engineer

For Maps, Local Actions: speech recognition for user reviews, i18n for yes-no questions about places.

2011 – 2014 SYSTRAN Director of Software Engineering

A pioneer in Machine Translation, SYSTRAN was for many years the engine behind the free translation services offered by Microsoft, Google, Yahoo, and Alta Vista, where it was called Babel Fish. My group in San Diego consists of software engineers, computational linguists, classically trained linguists, and lexicographers. We use hybrid approaches and domain-specific parallel corpora to train specialized translation models and extract dictionaries.

2008 – 2011 Photometria, Inc. Director of Engineering, Software Developer

E-commerce using Amazon and linkshare. Facebook and AIR desktop widgets. Recommendations of products based on co-occurrence. Google Maps mashup showing worldwide activity. Computer vision algorithms in openCV. Fast Thin Plate Spline rendering using openGL for the iPad.

2007 – 2008 Veoh Networks Director, Search & Analytics

My international team was responsible for metrics and recommendations. Due to an exponential rise in log volume, the Veoh web metrics system stopped functioning a week before I arrived. We put together a stop-gap system using MySQL and perl in a few weeks that lasted for several months, while a consulting firm designed our data warehouse.

2004 - 2007 Websense Director, Reporting and Database Development

Teams of 8 to 45 developers

I led a team to resolve one of Websense's biggest problems: we could not assume partition support in MSDE/SQL Server and our customer's databases suffered declining performance as they filled up. We came up with a simple partitioning technique that worked even on the low-end database MSDE. It allowed users to go beyond the size limitation of MSDE, using multiple databases and a reporting database that does a UNION ALL over them. The system allowed Websense to satisfy its largest clients, among them Boeing, Rolls Royce, and British Telecom.

2000 - 2004 Websense Chief Scientist

My team developed tools for Websense's impossible core task: classify the entire internet into about 80 categories. We developed a pipeline to manage and report on computer-assisted human classification of web pages. Our Digital Fingerprint was a heat-map created from Support Vector Machine scores of membership in each category.

1999 Binary Evolution Director of Consulting Services.

My team developed the Websense AfterWork website (see below in Products). Before that, we caught the end of the internet bubble by working on a high-end furniture and interior design site. Who knew people wouldn't want to buy an $8,000 couch unless they could sit on it first?

1993 - 1999 Encyclopædia Britannica Director, Research and Development

Teams of 5-10 developers, including three PhD's.

- Information visualization tools
- One of the first searchable databases on the pre-web Internet, using WAIS and Acrobat.

The effort to create an electronic version of Britannica started before the web, in the short transition period between the CD-ROM era and the web. When the web opened up with Mosaic 0.9 beta in 1992, we quickly produced a Wide Area Information Systems cgi-bin gateway for full-text search. The first version of Britannica Online lasted five years, and we used it as a resource for experiments in search and visualization. For details

see below in Products. A record of the project was created by the editor in chief at the time, Robert McHenry.

Products

2008: Veoh Discovery

Platform: xml, json, css, ajax, Oracle, pig/hadoop
Language: perl, sql, pig
With: Ted Dunning (scientist), Alex Sherback (SQL guru)

The prototype shows four kinds of recommendations:

- Video-to-video: users who watched this video tend to watch these other videos (similar videos)
- Query-to-query: users who did this search tend to do these other searches (similar queries)
- Query-to-video: users who did this search tend to watch these videos (behavioral search)
- Video-to-query: users who watched this video tend to do these searches (related queries)

The first method of building the data needed for recommendations was written entirely in SQL. We wrote a second version using the pig framework on a 20-node hadoop cluster.

2004: Websense Explorer

Platform: Web, SQL Server for Windows, MySQL for Unix
Language: perl
With: Vince Star (developer) and Don Aymar (graphic designer)

Logs of URL access by a company's employees are summarized by date, category, user, and other dimensions. The interface lets you select a value for one dimension and drill down by another in the same click. Websense has had a traditional batch-oriented template-based reporting program for several years. Reports had pre-defined columns and no links. Explorer is a game-changing product that gives instant information, a sense of free movement through the data, and shows users immediately the issues Websense is designed to address.

2001: One Click with Digital Fingerprint

Platform: Windows
Language: Visual Studio, VB, Java
With: Mark Anderson (scientist)

A "forward-caching" web browser featuring a compact graphic display of automatic classification results in about 80 categories. Ergonomic methods for capturing web analyst judgments about sites. Websense uses human Web Analysts to categorize web sites, after automated tools have done what they can.

2000: AfterWork

Platform: Unix, World Wide Web, Oracle
Language: Perl
Team: Robert DeWitt (developer)

A straightforward Web-based bookmarking system, intended for users of Websense whose companies choose to allow access to restricted websites after working hours.

1999: Topic Miner and Context Wrapper

Platform: Unix, World Wide Web, Informix

Language: Perl

Topic Miner uses Xerox InXight's natural language tool LinguistX to analyze the sentence grammar of a Web page, file, or text from any other source. For every noun phrase, it checks for an exact match to a list of variations of topic names in the Britannica Index. Context Wrapper uses Topic Miner output, or manually created links, to show related information in the margin of a document, displayed as *gists*: article title, thumbnail image when available, and first paragraph. This was a prototype for a short-lived collaboration between Newsweek, the Washington Post, and Encyclopædia Britannica.

1998: Fridge Door

Platform: Unix, Web
Language: Perl

A collaborative free-form communication tool for status reporting by a team of programmers. The web page was divided into sections, one for each developer. Each section could contain text, images, and links. Editing was done on the same web page, by clicking an edit button, an approach later made popular by wiki software.

1998: Britannica Event Viewer

Platform: Unix, Web
Language: Perl

Scan encyclopedia article text for *event sentences*, those containing a string that parses as a date. Create a database of them, and extract items according to date and subject, generating timeline-formatted output. An alternative presentation was as a VRML mountainscape. Categories were arranged along the x-axis, time on the y-axis, and the number of event sentences on the z-axis. The height of the mountain was the number of event sentences found in the corresponding category. At the peaks were billboards made of images found in the eventful articles.

1997: BugTracker

Platform: Unix, Web
Language: Perl
With: Robert DeWitt (developer)

A simple web-based method of reporting bugs and tracking progress, for Britannica CD-ROM and Web QA.

1996: Connections

Platform: Unix, World Wide Web, Informix
Language: Perl
With: Brent Foust (developer)

A method of browsing a topically ordered list, and a new way of displaying the Britannica Index. Scan articles as if they were books on a library shelf. The interface gives the user an ability to see the range of articles on a given subject, as that subject gives way on each end to other related topics.

1994-1998: Britannica Online enhancements

Platform: Unix, Web, Informix
Language: C++, Tcl, perl
With: six other developers

After my team created the original prototype, the MIS department of Encyclopedia Britannica in Chicago reproduced the interface directly from data in the publishing system. Several enhancements were made over the years, including

thumbnails: The original EB web design made no use of inline images. We created thumbnails of the diagrams, drawing, and photographs and included them in the HTML files and search results for a more compelling page layout.

hypertext index: The print Index contains only a point location for each index link. The first version of EB online displayed this with the inline link "[Index]". A script was written to find text similar to the index title in the running text of the article, and replace the point link with true hypertext.

Illustrated Propædia: The EB topic tree was originally displayed as a computer file system and required the user to "drill down" through folders and subfolders. This new interface fetched an image from the encyclopedia articles classified in each folder to display along with the folder title, and arranged the page to show two levels of the tree at once. A very early example of ajax, it used a frameset and innerHTML to download new nodes of the tree without refreshing the rest of the page.

1993: prototype Britannica Online web site

Platform: Unix, Web
Language: C++, perl
Team: John McInerney (scientist), Brian Bartell (scientist), Amy Steier (scientist)

The first attempt to make a Web out of EB data was the Propædia, implemented as a file system with directories named "Matter and Energy", "The Earth", and so on. This was the second, done by converting tagged EB article text into HTML. About two months after this prototype, the complete system with full-text searching was available for beta testing. This system was shown at the 1994 American Booksellers Association conference in Los Angeles, resulting in a New York Times article by John Markoff on Feb 7, 1994. This was one of the first articles in a major publication to mention the World Wide Web in connection with a traditional publishing company. Advisors on the long-term project included Brewster Kahle, Danny Hillis, Don Norman, Roger Shank, and Paul Saffo.

See The New York Times article announcing the product.

1992-1994: Propædia Space (Mortimer)

Platform: Unix, Web
Language: C++
Team: Brian Bartell (scientist)

Graphic display of automatically categorized documents using distance to indicate relevance to nodes in the category hierarchy, arranged using multidimensional scaling.

1992 : Virtual Workspace for Compton's Interactive Encyclopedia

Platform: Windows
Language: C++

The Virtual Workspace is a user interface based on a flexible modular architecture of Windows Dynamic Link Libraries. There are tool DLLs, a Message Dispatcher DLL, and User interface DLLs, like the Topic Tree, the Picture Tour, and Idea Search. The shell scans for libraries available in the system, so you can plug in a new one without modifying a line of existing code. Objects can span libraries, so a base object might be defined in one DLL, and used in many other DLLs to create derived objects. The user interface features a system based on the standard Multiple Document Interface, presenting the MDI client window (the container for all the children) as a large canvas on which the user places selected documents. You can scroll the canvas by clicking and dragging on any spot between the child windows. A map shows the relative positions of the children, and serves as a way to **manipulate** them.

1989 : World Atlas for Compton's Encyclopedia

Platform: IBM PC under DOS and Windows
Language: Turbo Pascal, 8086 Assembly Language, C, C++

It is easy to create a linked library of map images. We decided to draw them on the fly, using vector coastline and country boundary latitude-longitude data, to provide an unlimited number of views. We projected points orthographically onto the screen and connected them with lines to produce something like a virtual a 3D globe. To color the water, we used flood-fill seed points created by editors. Labels had to be displayed at their zoom level and as close as possible to their real position without overlapping other labels. Each label was linked to the corresponding article in the encyclopedia. Part of Compton's Encyclopedia and converted to a wide variety of platforms, from DOS to Windows to set-top boxes. Much later, Google Earth did the same thing with more data and processing power.

Personal Projects

2025: Ebay Comis Sold
Platform: React/Next
Link: https://ebay-comics-sold.vercel.app

Companies that calculate the fair market value of a comic book have a tough job. This tool shows the most valuable raw data for valuation: the last 3 months of sales on ebay. It does a live query to ebay and screen-scrapes the response, extracting sold prices, sell date, grades, and whether the comic is "slabbed" or in its "raw" state. The graph shows all relevant data that determine a value, but does not attempt to interpolate a value for a given grade. It leaves that up to you.

2025: Long Tail and Collaborations
Platform: React/Next and PostgreSQL
Link: https://longtailhair.vercel.app

Adds a “Wall of Movies” UI to the **Collaboration Graph Browser.** Filter by combinations of genres and dates, click a poster to browse collaborations.

2024: Wall of Comics
Platform: React/Next
Link: https://silverage.vercel.app

I collected Marvel comics as a kid for a few years. I still have them, and they are worth something now. I wrote this tool to display them all, using the Filterpanel to let users select subsets.

2021: Collaboration Graph Browser
Platform: React/Next and PostgreSQL
Link: http://www.johndimm.com/collaboration.html

A research tool for students and collectors. I used musicbrainz's public database to extract the collaboration graph, and developed this front-end interface. It answers the question: What were the musicians on this album doing before and after? The same concept was also applied to the IMDb database for movies and tv shows.

2021: Filterpanel
Platform: React/Next
Link: https://filterpanel-csv.vercel.app/

A software tool that generates a filter panel UI, as seen on Amazon and many other e-commerce sites. Filtering is a powerful adjunct to natural language search. This version downloads a large set of search results and reads them into local memory. All analysis is done in javascript. In spite of that, or maybe because of it, response is super fast.

2019: TalkShow for Alexa
Platform: Amazon Alexa Skill
Link: https://www.amazon.com/John-Dimm-TalkShow/dp/B0816KVF14/ref=sr_1_1?keywords=TalkShow&qid=1579326798&s=digital-skills&sr=1-1

A port of my Chrome web app TalkShow, this one uses the entire user utterance as query to Flickr and Bing through Azure Cognitive Services. Results are displayed in a slideshow done using the Amazon Presentation Language and the AutoPage command.

2019: Breakdown for Mint and Personal Capital
Platform: Google Chrome Extension, React, SQL using Lovefield
Link: https://chrome.google.com/webstore/detail/breakdown-for-mint-and-pe/npkcbabhgmplkmaaljpcbombbgkggilm?h1=en

Breakdown slices and dices personal financial data. The interface works on all kinds of multidimensional data. For personal finance, it loads your downloaded Mint or Personal Capital data into Lovefield, the browser-based local database written in javascript by a team at Google.

2018: What Looks Good?
Platform: Yelp Dataset Challenge, React, SQL
Link: http://www.johndimm.com/yelp_db_caption/app/

I wanted to create an interface that lets you start by thinking about *what* you want to eat and decide later *where* you want to eat. The Yelp Dataset Challenge makes available a good chunk of anonymized data. Using only the restaurant table, the photographs, and photo captions, it was possible to extract the latent concept of a Dish. The key insight was that when people take pictures of food, the captions often have a useful feature: they just say what it is. “Sushi”, “Steak”, and “Grilled calamari” are common captions. We can identify dishes from the co-occurrence of phrases in caption text, then searches in all caption text for them, creating a graph of the relation between restaurants and their dishes. The UI relies on two recommender systems built on that relation.

2014: Lingo Hero
Platform: Web Speech API, Google text-to-speech, Microsoft Bing Image Search, Tatoeba Corpus
Language: Javascript

Guitar Hero gives you the impression you are playing guitar without ever studying it. Lingo Hero gives you the impression you speak a foreign language without ever studying it. It uses speech synthesis to read sentences to you. Your goal is to say each word just well enough for speech recognition in that language to think you said it. It turns out speech recognition is very forgiving, especially if it hears sequences of words, which lets the language model correct errors in your speech. I'm using short sentences from the multilingual tatoeba corpus of colloquial speech. Hear and speak the whole sentence or each word independently. You get a great sense of accomplishment when you fool it into thinking you just spoke Russian or Arabic, when all you are really doing is imitating what you just heard.

2014: Fun with Speech
Platform: Web Speech API, Google text-to-speech, Microsoft Bing Image Search , Flickr image search, Bing Translator
Language: Javascript, PHP

A presentation, series of demos, and github repository for the 2014 jQuery Conference in San Diego.

2013: DrillDown.js
Platform: jQuery, MySQL
Language: Javascript, PHP

Summarize a table of dimensions and measures and browse it using SQL group by queries.

2013: Translating Telephone
Platform: WebRTC, Web Speech API, Google text-to-speech, Microsoft Bing Translator
Language: Javascript

Multilingual video conferencing with subtitles.

2013: TalkShow Story Illustrator
Platform: Web Speech API, jQuery, Alchemy API, YahooBOSS, flickr API
Language: Javascript

Eavesdrops your conversation and shows pictures of the things you're talking about.

2013: Interpreter
Platform: Web Speech API, jQuery, Microsoft Bing Translator API, Google text-to-speech
Language: Javascript

Hands-free continuous voice-to-voice translation on the web, with back-translation.

2001: Animated Listboxes

Platform: World Wide Web
Language: Perl, Javascript

There's a big problem with slide shows. In return for simplicity of use, you give up all control. You don't know where you are. Sometimes you don't care, when the slide show tells a story. But if the sequence of images is more or less random, it would be helpful to have an overview of the collection of images that make up the slide show. Slide shows hide this information. It doesn't have to be that way.

1990 : w, an object-oriented graphic shell

Platform: IBM PC under DOS
Language: Turbo Pascal with Objects

A DOS graphic windowing shell. The original version was a visually accurate and detailed Mac clone (even the thumb in the scrollbar followed the special Mac xor pattern). Windows received messages, like Paint, from an event manager. New objects could be added with one minor change to existing code. Used for a series of classes at Josten's Learning Corp on object-oriented programming using Borland Turbo Pascal. It was released as part of the World Atlas for Compton's Family Encyclopedia for DOS.

1987 : The Icon Shell

Platform: IBM PC
Language: Turbo Pascal, 8086 Assembly Language

A text-mode DOS shell program, using ASCII graphics animation. An arbitrary number of icons, arranged in a hierarchy, scroll vertically through rows of three. Levels above the first are shown in popup windows. Besides brief mention in *BYTE* and *Personal Computing*, there was a long review by Stan Kelly-Bootle in *Computer Language*, July 1988

> ICON SHELL shows great creativity, gives your application instant user appeal, and has success written all over it ... When I demonstrated it to Steve Bourne, the eponymous UNIX shell progenitor, he was quite impressed, exclaiming, "Ah, well! Back to the drawing board."

and this from Hal Nieburg in *Computer Shopper*, September 1988

> In a world that is increasingly weary of shell programs, here is a lively contribution that gives computing on a DOS machine an extra kick, a brilliant set of pictures and a bright highlight image for pointing and activation. ... Here is a new offering, that presents an attractive interface to the user, and is easy for the neophyte, not only to use, but to install as a gateway system for maintaining and operating a large capacity hard disk.

1986 : Automatic Sonata Composer

Platform: Yamaha CM5 personal computer
Language: Basic

Written for a Z-80 computer with eight Yamaha DX-9 FM synthesis sound chips and inspired by the simple rules in Paul Hindemith's *Traditional Harmony*. Random four part harmony with contrary motion, no parallel octaves or fifths, often moving down a fifth. Sonatas were created by following an AABBA structure, where B is similar to A but modulated up a fifth. The soprano voice was played by a synthesized clarinet, the other three by a piano, as accompaniment. Between chords in the Hindemith stuff, the soprano voice embellished with a few nearby notes also chosen at random. It is surprising how repeating something, anything, can make it sound intelligent and intended, just because it happens twice.

Patents

2008: Search System using Patterns of Usage

From the filing: “The fact that recommendation technology can be used to implement a search function appears to be completely novel.”

We used collaborative filtering to do a search-like thing: map queries to documents (in this case, videos). Normally, you use collaborative filtering to group people by a given behavior (products purchased, web pages viewed), and then use that to predict future behavior of the same sort. You might want to buy this product because it was bought by people who bought some of the things you bought in the past.

The twist here is to switch in the middle, from one kind of behavior to another. For the query-to-document mapping, it looks at the users who have done a given query, and then instead of looking at the other queries they have done (which produces similar queries), it looks at all the videos they have watched. It doesn't matter how they got to them: from the given query or from some other query or by browsing or whatever. People who did this query disproportionately watched these videos. For document-to-query, it looks at the users who have watched a given video, and then looks at the queries they have done.

As far as we can tell, no one is doing this yet. It would be a great feature for any searchable site. People tend to ask the same things, even though there are a lot of things to ask. -- about 850,000 queries at veoh have been done by at least two people.

2007: System and method of monitoring and controlling application files, Patent number: 7185015, with Harold Kester, Ron Hegli, and Mark Andersen

Websense wanted to create a categorized database of Windows applications, in order to filter and control them. This was seen as a logical extension of their core business, filtering websites by category. The patent dealt with

a practical problem: we couldn't possibly buy or somehow acquire every piece of software in the world. Our solution was to examine and generate hashes for the application files our client software finds on our user's computers and send that information back to Websense for classification. I was part of the design team and wrote the Windows Portable Executable hashing routines and a C# GUI to display the captured version info, link to websites, and facilitate the manual classification process.

1992: The infamous Compton's Multimedia Patent: "Multimedia search system using a plurality of entry path means which indicate inter relatedness of information.", Patent number: 6978277

Compton's Encyclopedia was a huge effort and the company thought it should be rewarded for creating something brand new. The company executives announced the royalty payment schedule at Comdex that year. All hell broke loose. A few years later, all 41 claims were overturned due to prior art (Hypercard). See The New York Times.

I was not listed by name in this patent, but my World Atlas was one of the entry paths. The patent was granted again in 2005 with a special focus on the World Atlas. Two years later Britannica filed a lawsuit against Garmin, Magellan, and Tom Tom.

Publications

2001: Presenting content one slide at a time

Platform: Unix or Windows, World Wide Web
Language: Perl, Javascript
Link: https://people.apache.org/~jim/NewArchitect/webtech/2001/04/dimm/index.html

The lowly slide show is an underrated user interface paradigm. This article examines four implementations in detail.

1991 : A tiny Windows Class Library

Platform: Windows
Language: C++
Link: https://www.unz.com/print/ProgrammersJournal-1991nov-00016/

Soon after the release of the first Windows C++ compiler by Borland, this was the feature article in the November/December 1991 issue of *Programmer's Journal.* Before OWL or MFC, it showed how a simple class library in Borland C++ lets developers avoid the usual Windows boilerplate code. Create a generic window in two statements.

Education

- Princeton University: full fellowship in History and Philosophy of Science. Linguistics, Metaphysics (the problem of identity over time), Decision Theory, Logic, Foundations of Mathematics.
- University of Washington: Ethnomusicology, Philosophy, Linguistics, Mathematics. B.S. Mathematics, B.A. Philosophy, cum laude, Phi Beta Kappa.
- Favorite class: Point-Set Topology by the Moore Method, with Martin Bendersky. The first day of class the teacher passed out the course materials -- about 10 Xeroxed pages containing definitions, axioms, lemmas, and theorems. There was no narration, no diagrams, and above all no proofs. Moore said "That student is taught best who is told the least". The professor has it easy. After the first class, where

he explains how things will work, the students are invited to go to the board to do proofs. They get progressively harder until each one takes up several boards. It was incredibly satisfying to get a proof on your own, but for the hard ones we worked in groups, in grad students' offices or empty classrooms.

- Reed College: Physics major

Foreign Languages

I spent four years working in Paris and my extended family is French, so I can speak and read it.