Board Search

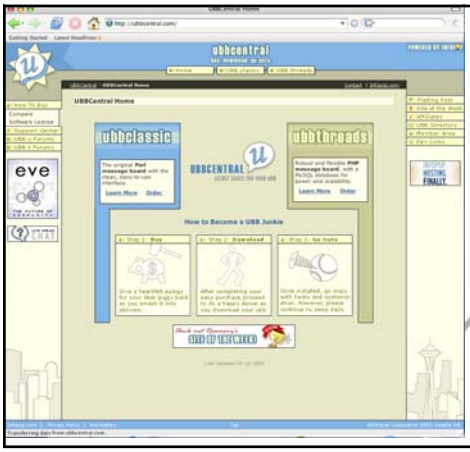
An Internet Forum Index

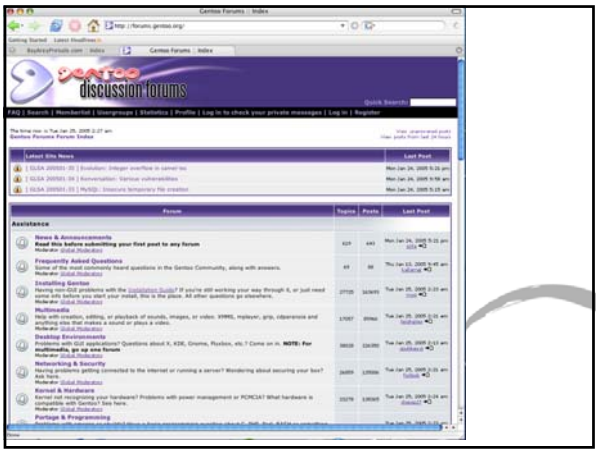
Overview

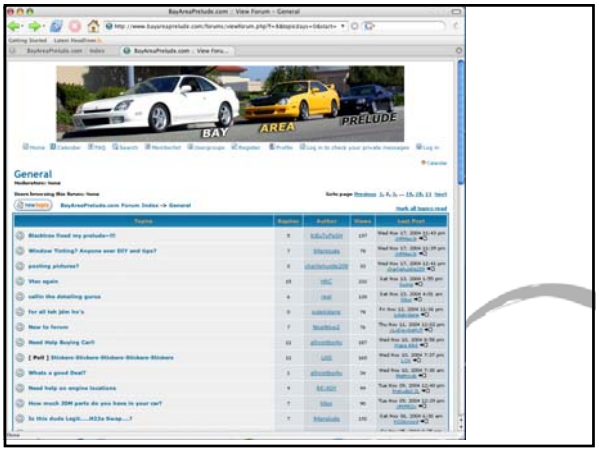
- Forums provide a wealth of information
- Semi structured data not taken advantage of by popular search software
- Despite being crawled, many information rich posts are lost in low page rank

Forum Examples

- vBulletin
- phpBB
- UBB
- Invision
- YaBB
- Phorum
- WWWBoard

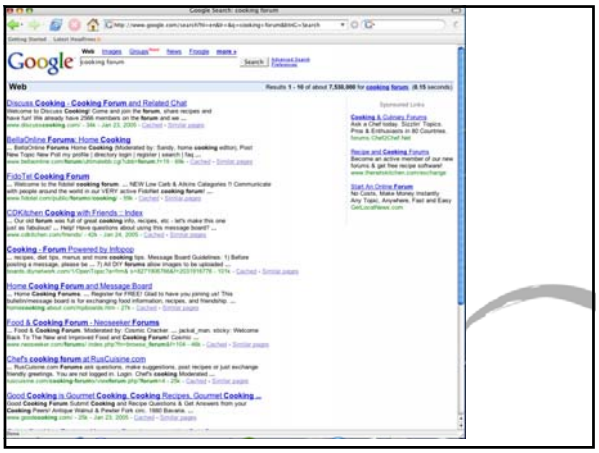


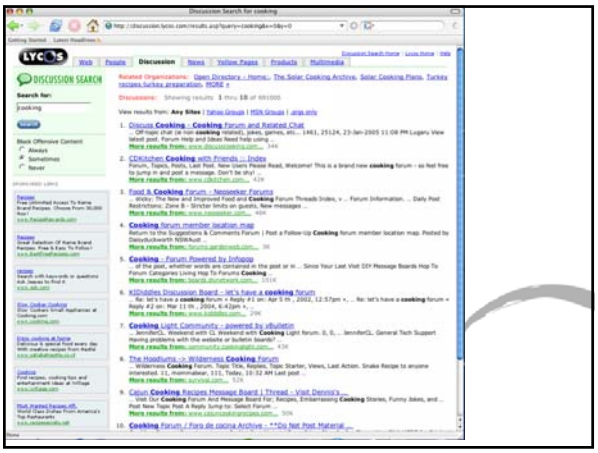


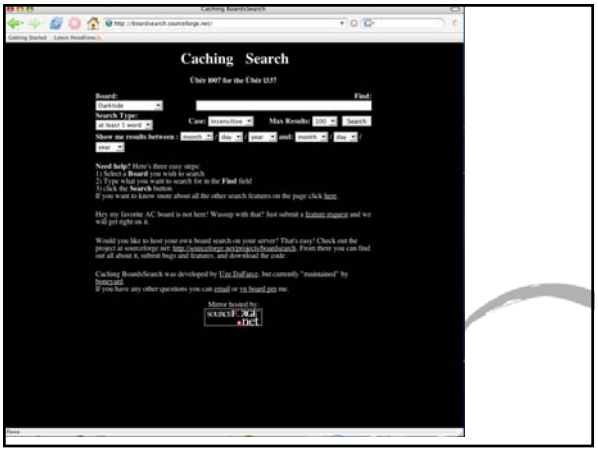


Current Solutions

- Search engines
- Forum's internal search







Evaluation Metric

Metrics: Recall - C/N, Precision C/E

Rival system:

- Rival system is the search engine / forum internal search combination
- Rival system lacks precision

Evaluations:

- How good our system is at finding forums
- How good our system is at finding relevant posts/threads

Problems:

- Relevance is in the eye of the beholder
- How many correct extractions exist?

Implementation

- Lucene
- Mysql
- Ted Grenager's Crawler Source
- Jakarta HTTPClient

Improving Software Package Search Quality

Dan Fingal
and
Jamie Nicolson

The Problem

- Search engines for softare packages typically perform poorly
- Tend to search project name an blurb only
- For example…

Sourceforge.org

Gentoo.org

Freshmeat.net

How can we improve this?

- Better keyword matching
- Better ranking of the results
- Better source of information about the package
- Pulling in nearest neighbors of top matches

Better Sources of Information

- Every package is associated with a website that contains much more detailed information about it
- Spidering these sites should give us a richer representation of the package
- Freshmeat.net has data regarding popularity, vitality, and user ratings

Building the System

- Will spider freshmeat.net and the project webpages, put into mySQL database
- Also convert gentoo package database to mySQL
- Text indexing done with Lucene
- Results generator will combine this with other available metrics

How do we measure success?

- Create a gold corpus of queries to relevant packages
- Measure precision within the first N results
- Compare results with search on packages.gentoo.org, freshmeat.net, and google.com

Any questions?

Incorporating Social Clusters in Email Classification

By

Mahesh Kumar Chhaparia

Previous Work

- Previous work on email classification focus mostly on:
  - Binary classification (spam vs. Non-spam)
  - Supervised learning techniques for grouping into multiple existing folders
    - Rule-based learning, naïve-Bayes classifier, support vector machines
    - Sender and recipient information usually discarded
- Some existing classification tools
  - POPFile : Naïve-Bayes classifier
  - RIPPER : Rule-Based learning
  - MailCat : TF-IDF weighting

Email Classification

- Emails:
  - Usually small documents
  - Keyword sharing across related emails may be small or indistinctive
  - Hence, on-the-fly training may be slow
  - Classifications change over time, and
  - Different for different users !!
- Motivation:
  - The sender-receiver link mostly has a unique role (social/professional) for a particular user
  - Hence, it may be used as one of the distinctive characteristics of classification

Incorporating Social Clusters

- Identify initial social clusters (unsupervised)
- Weights to distinguish
  - From and cc fields,
  - Number of occurrences in distinct emails
- Study effects of incorporating sender and recipient information:
  - Can it substitute part of the training required ?
  - Can it compensate for documental evidence of similarity ?
  - Quality of results vs. Training time tradeoff ?
  - How does it affect regular classification if used as terms too ?

Evaluation

- Recently Enron Email Dataset made public
  - The only substantial collection of “real” email that is public
  - Fast becoming a benchmark for most of the experiments in
    - Social Network Analysis
    - Email Classification
    - Textual Analysis …
- Study/Comparison of aforementioned metrics with the already available folder classification on Enron Dataset

Extensions

- Role discovery using Author-Topic-Recipient Model to facilitate classification
- Lexicon expansion to capture similarity in small amounts of data
- Using past history of conversation to relate messages

References

- Provost, J. “Naïve-Bayes vs. Rule-Learning in Classification of Email”, The University of Texas at Austin, Artificial Intelligence Lab. Technical Report AI-TR-99-284, 1999.
- E. Crawford, J. Kay, and E. McCreath, “Automatic Induction of Rules for E-mail Classification,” in Proc. Australasian Document Computing Symposium 2001.
- Kiritchenko S. & Matwin S. “Email Classification with Co-Training”, CASCON’02 (IBM Center for Advanced Studies Conference), Toronto, 2002.
- Nicolas Turenne. “Learning Semantic Classes for improving Email Classification”, Proc. IJCAI 2003, Text-Mining and Link-Analysis Workshop, 2003.
- Manu Arey & Sharma Chakravarthy. “eMailSift: Adapting Graph Mining Techniques for Email Classification”, SIGIR 2004.

A research literature search engine with abbreviation recognition

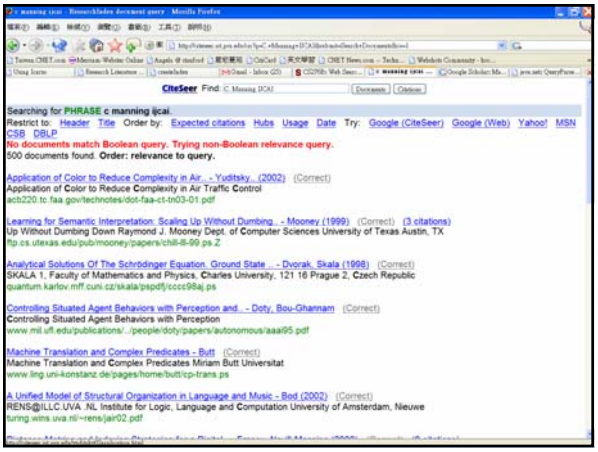
Group members
Cheng-Tao Chu
Pei-Chin Wang

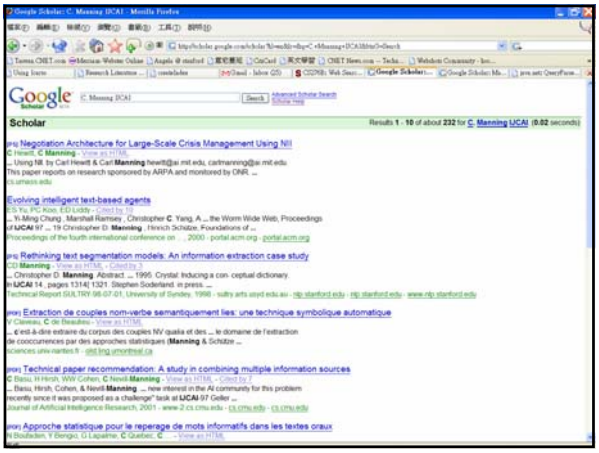
Outline

- Motivation
- Approach
  - Architecture
- Technology
- Evaluation

Motivation

- Existing research literature search engines don't perform well in author, conference, proceedings abbreviation
- Ex: search "C. Manning, IJCAI" in Citeseer, Google Scholar





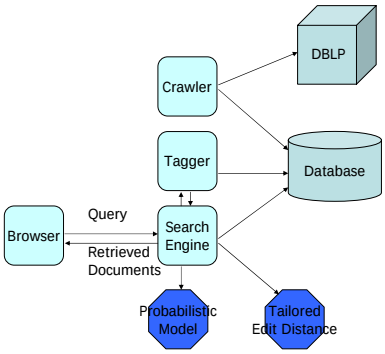
Goal

- Instead of searching by only index, identify the semantic in query
- Recognize abbreviation for author and proceedings names

Approach

- Crawl DBLP as the data source
- Index the data with fields of authors, proceedings, etc.
- Train the tagger to recognize authors and proceedings
- Use the probabilistic model to calculate the probability of each possible name
- Use the tailored edit distance function to calculate the weight of each possible proceeding
- Combine these weights to the score of each selected result

Architecture



Technology

- Crawler: UbiCrawler
- Tagger: LingPipe or YamCha
- Search Engine: Lucene
- Bayesian Network: BNJ
- Web Server: Tomcat
- Database: MySQL
- Programming Language: J2SE 1.4.2

Evaluation

- 1. We will ask for friends to participate in the evaluation (estimated: 2000 queries/200 friends).
- 2. Randomly sample 1000 data from DBLP, extract the authors and proceedings info, query with abbreviated info, check how well the retrieved documents match the result from the Google scholar

A Web-based Question Answering System

Yu-shan & Wenxiu

01.25.2005

Outline

- QA Background
- Introduction to our system
- System architecture
  - Query classification
  - Query rewriting
  - Pattern learning
- Evaluation

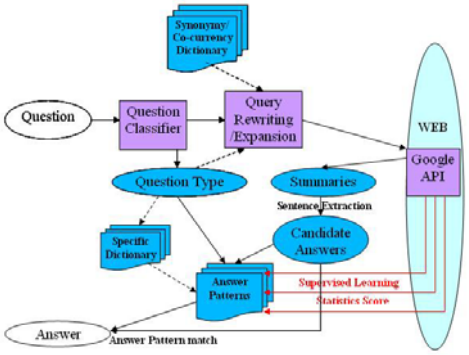
QA Background

- Traditional Search Engine
  - Google, Yahoo, MSN,…
  - Users construct keywords query
  - Users go through the HitPages to find answer
- Question Answering SE
  - Askjeeve, AskMSR, …
  - Users ask in natural language pattern
  - Return short answers
  - Maybe support by reference

Our QA System

- Open domain
- Massive web documents based
  - redundancy guarantee effective
- Question classification
  - focus on numeric, definition, human…
- Exact answer pattern

System Architecture



Question Classifier

- Given a question, map it to one of the predefined classes.
- 6 coarse classes (Abbreviation, Entity, Description, Human, Location, and Numeric Value) and 50 fine classes.
- Also show syntactic analysis result such as POS Tagging, Name Entity Tagging, and Chunking.
- http://l2r.cs.uiuc.edu/~cogcomp/demo.php?dkey=QC

Query Rewrite

- Use the syntactic analysis result to decide which part of question to be expanded with synonym.
- Use WordNet for synonyms.

Answer Pattern Learning

- Supervised machine learning approach
- Select correct answers/patterns manually
- Statistics answer pattern rule

Evaluation

- Use TREC 2003 QA set. Answers are retrieved from the Web, not from TREC corpus.
- Metrics
  - MRR(Mean Peciprocal Rank) of the first correct answer
  - NAns(Number of Questions Correctly Answered), and
  - %Ans(the proportion of Questions Correctly Answered)

Streaming XPath Engine

Oleg Slezberg
Amruta Joshi

Traditional XML Processing

- Parse whole document into a DOM tree structure
- Query engine search the in-memory tree to get the result
- Cons:
  - Extensive memory overhead
  - Unnecessary multiple traversals of the document fragment
    - E.G. /Descendent::x/ancestor::y/child::z
  - Can not return result as early as possible
    - E.G. Non-blocking query

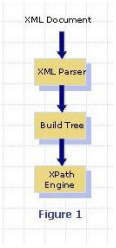
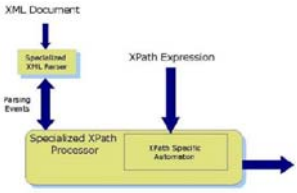


Figure 1

Streaming XML Processing

- XML parser is event-based, such as SAX
- XPath processor performs the online event-based matching
- Pros:
  - Less memory overhead
  - Only process necessary part of input document
  - Result returned on-the-fly, efficient support for non-blocking query

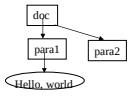


What is XPath?

- A syntax used for selecting parts of an XML document
- Describes paths to elements similar to an os describing paths to files
- Almost a small programming language; it has functions, tests, and expressions
- W3C standard
- Not itself written as XML, but is used heavily in XSLT

A Simple Example

An XML document	SAX API Event
<doc> <para1> Hello world! </para1> <para2> … </para2> </doc>	Start element: doc Start element: para1 data: Hello world! End element: para1 ... End element: doc



- XPath query Q = /doc/para1/data()
- Traditional processing:
  - Build an in-memory DOM strucuture
  - Return “Hello world” after end document
- Streaming processing
  - Match /doc in Q when start element doc
  - Match /doc/para1 in Q when start element para1
  - Return “Hello world” when end element para1

Objective

- Build an Streaming XPath Engine using TurboXPath algorithm
- Contributions:
  - comparison of FA-based (XSQ) and tree-based (TurboXPath) algorithms
  - performance comparison between TurboXPath & XSQ

XPath Challenges

- Predicates
- Backward axis
- Common subexpressions
- // + nested tags (e.g. <a> ... <a> ... ...)
- *
- Children in predicates that are not yet seen (e.g. a[b]/c and c is streamed before b)
- Simultaneous multiple XPath query processing

Algorithms

- Finite-Automata Based
  - XFilter
  - YFilter
  - XSQ
- Tree-Based
  - XAOS
  - TurboXPath

Evaluation

- Implementations will be evaluated for
  - Feature Completeness
  - Performance (QPS rate)
- XMark
  - XML Benchmarking Software