Deep Classifier: Automatically Categorizing Search Results into Large-Scale Hierarchies

Presented by Qiang Yang

Hong Kong Univ of Sci and Tech.

Dikan Xing, Guirong Xue, Yong Yu

Shanghai Jiao-tong University

Qiang Yang

Hong Kong University of Science and Technology

Search Result Presentation

- List form or hierarchical form
- Hierarchical form preferred by many users
  - [Chen and Dumais 2000]
  - [Hearst 2006]
  - [Etzioni et al. : Grouper; WWW ’99]
- **Question**
  - How do we automatically categorize search results from a list form into a hierarchical form?
    - Based on classification rather than clustering
    - Deep vs. Shallow

Motivation

Search Result Categorization

Help user browsing

User preferred

Large-Scale Hierarchical Categorization

Detailed Categorization

Users prefer categorization but too shallow so far

Data Mining can help, but needs to be efficient and effective apple | Search | on | Google

All Results(apple)(100)

- Arts
  - Music
    - Bands and Artists
      - A
        - **Apple, Fiona(5)**
- **Computers**
  - Software
    - Operating Systems
      - **Mac OS(3)**
  - Systems
    - **Apple(3)**
- Games
  - Card Games
    - Special Decks
      - **Apples to Apples(37)**
- Home
  - **Cooking**
    - Beverages
      - **Apple Juice and Cider(1)**
    - Fruits and Vegetables
      - Apples
        - **Apple Dumplings(1)**
- Shopping
  - Food
    - Confectionery
      - **Candy Apples(46)**

26. FIONA APPLE
Official website provided by Sony Entertainment. latest videos and album releases.
http://www.fiona-apple.com

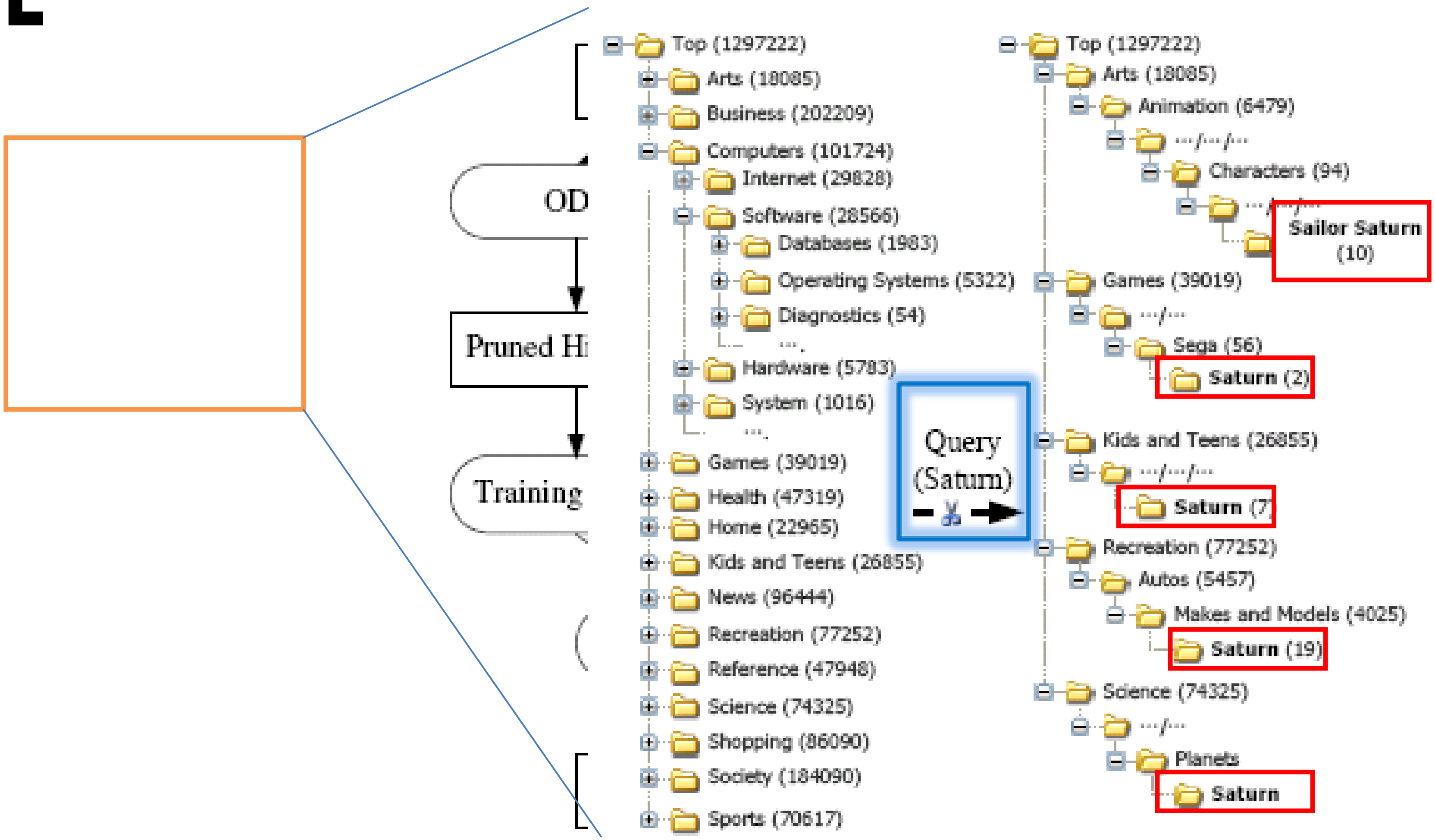
52. Apple IMC Türkiye
Apple e-Bülten'e abone olmak için tıklayınız... Ap sites around the world:. Seçin... Africa, Asia, .
http://www.apple.com.tr

54. Apple Magyarországi Képviselet
Apple IMC Hungary. Visit other Apple sites aroun Asia, Australia, Austria, Azerbaijan, Belgium, B
http://www.apple.hu

56. Applegeeks 3.0
Fission is a streamlined audio editor that allow Apple Lossless and AIFF with no re-encoding, so
http://www.applegeeks.com

67. UVM Apple Orchard
The UVM Apple Program: Extension and Research fo in Vermont and beyond...
http://orchard.uvm.edu

Query=“Saturn”



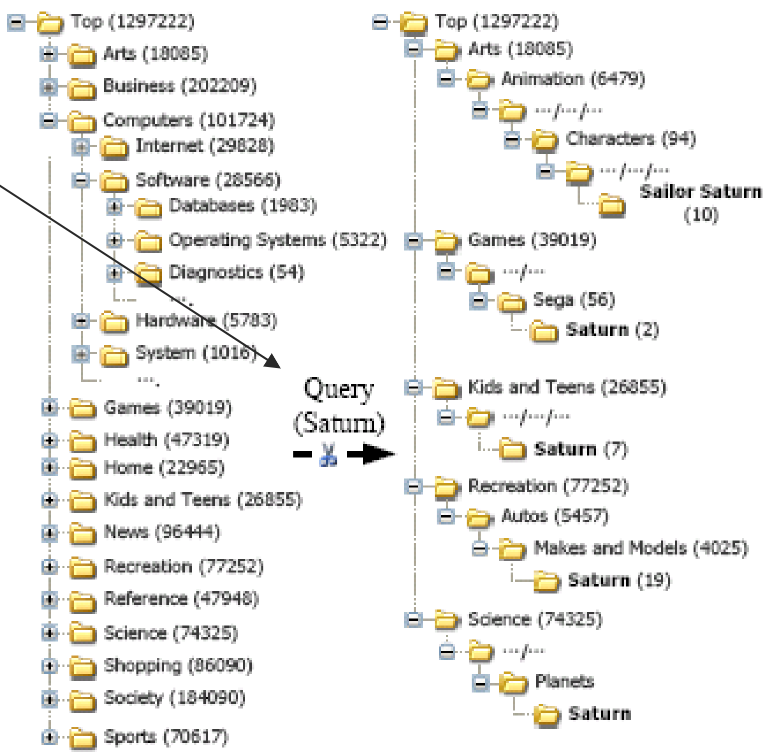
Deep Classifier: Detailed Steps

1. Identify a large online taxonomy **T** for categorization
   1. Open Directory Project, Yahoo directories, etc.
2. Given a query **Q**, obtain a set of candidate categories **C(Q)**
3. Prune **T(Q)** using **C(Q)**
   1. The result is a deep and narrow taxonomy T(Q), where all leaf nodes are candidates
4. Build a classifier into the leaf nodes in T(Q)
5. Classify each search result in S(Q) into T(Q)
6. Present T(Q) to the user

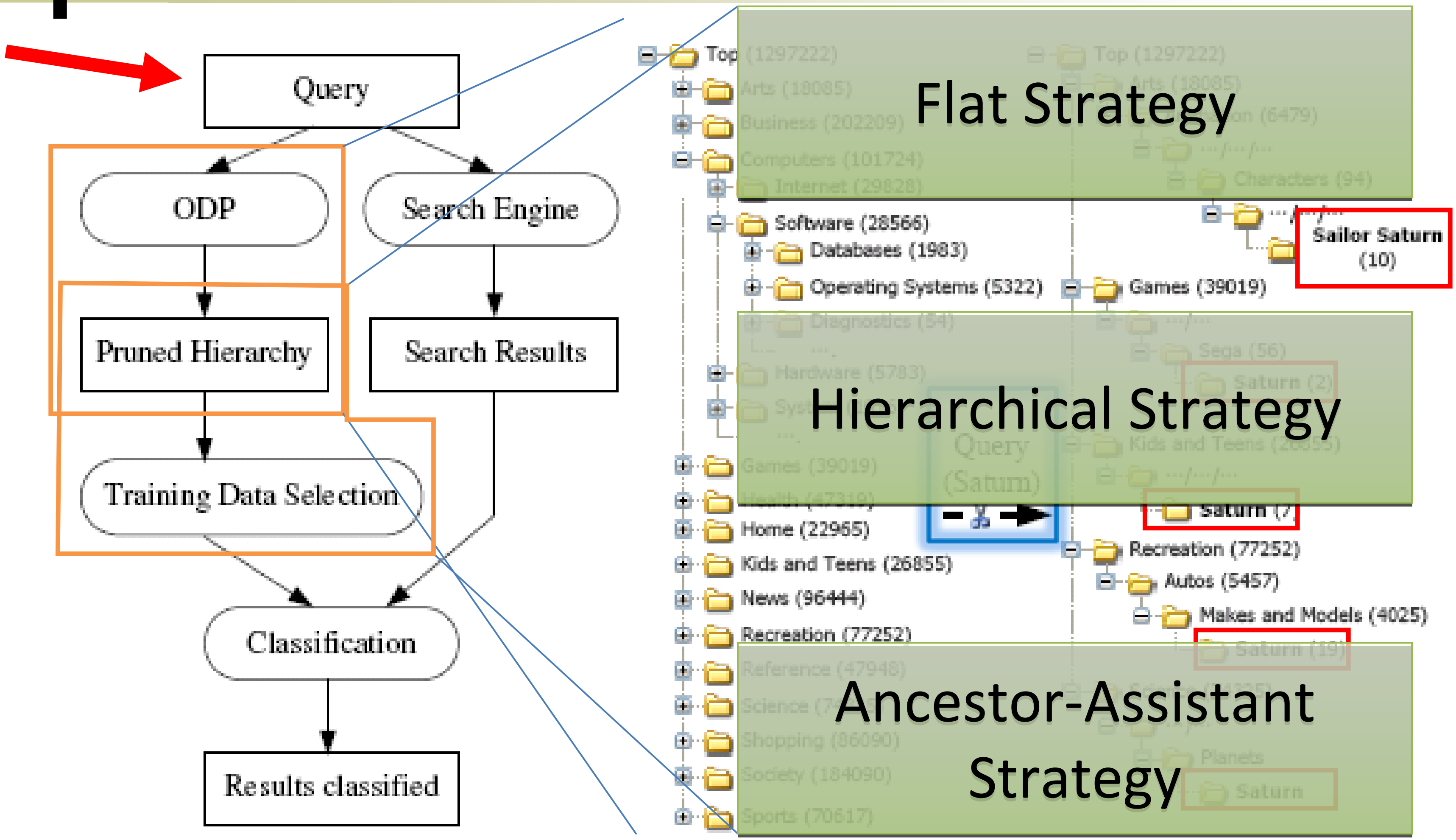
- Properties:
  - The search results S(Q) are classified into different categories C(Q) for different queries Q
  - A classifier is trained online for each incoming query
    - Is this feasible?
  - The classifier should be both trained efficiently and accurate

Research Question 1: How to build a classifier online?

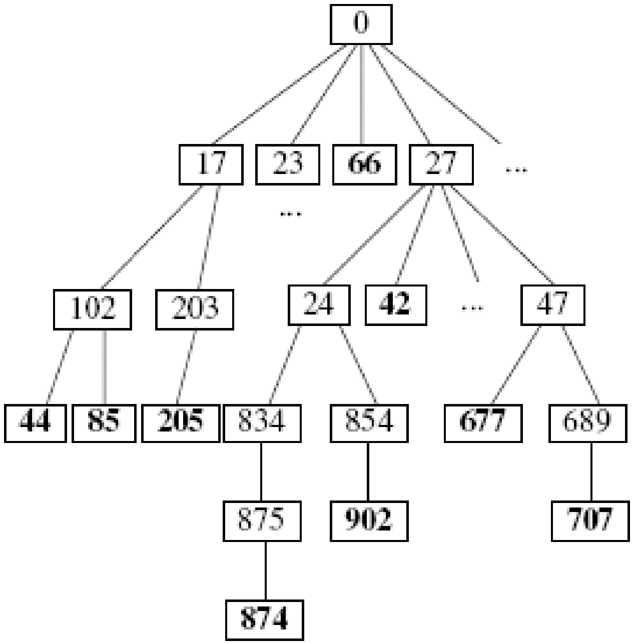
- Given a query, we can use the search functions of various online taxonomies to find the candidate categories
  - ODP already does this
- To build a classifier into these candidates, we must **collect training data** for each category
  - How?

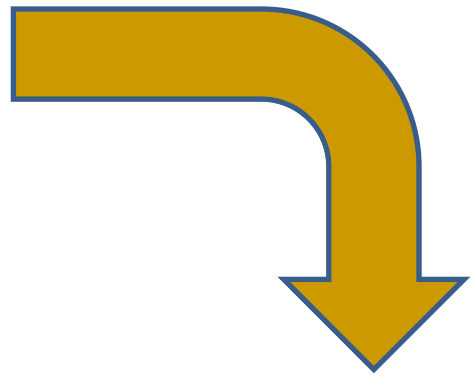


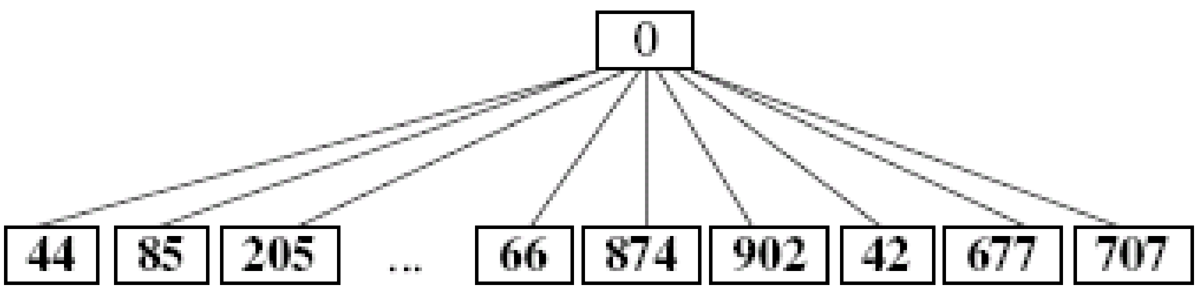
Question: How to build a classifier in real time?



Flat Strategy for Training a Classifier





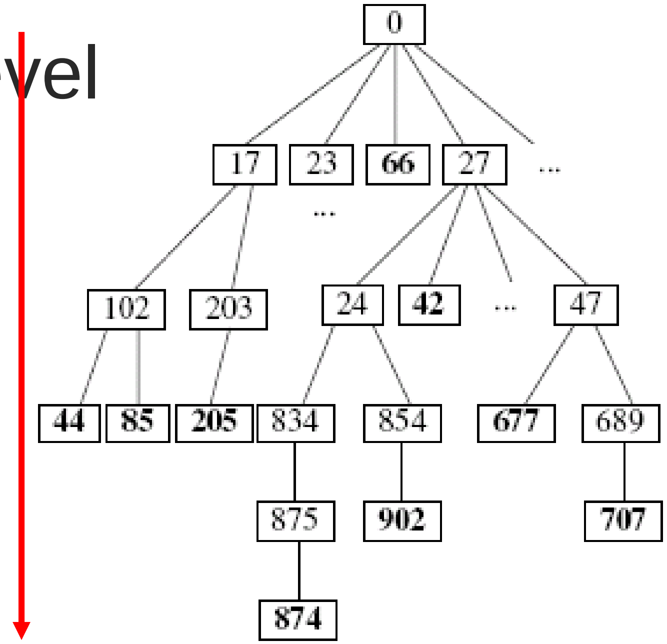


- Pros: Simple
- Cons:
  - Multi-class!
  - Data Scarcity
    - 21.6 docs per class node
    - Easy to overfit!

Hierarchical Strategy [related works…]

- Classify top-down, level-by-level

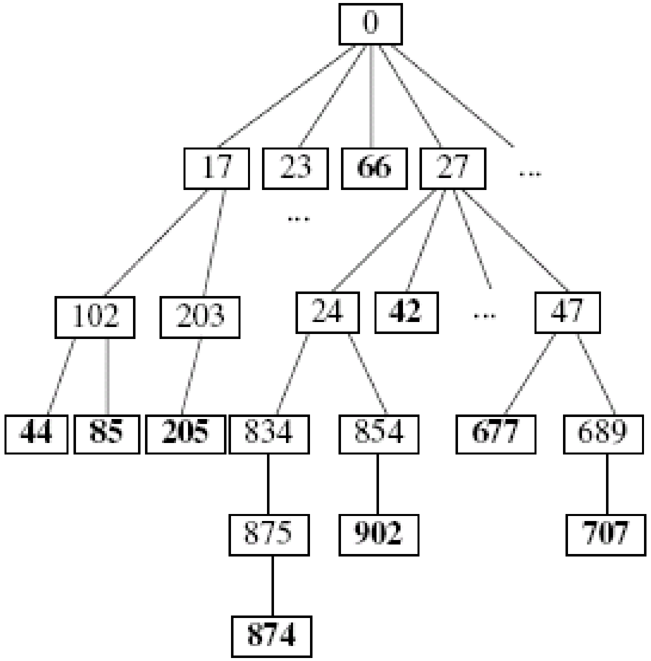
$$
\begin{aligned}
& P(c_j|x) \\
&= P\left(c_j^1, c_j^2, \cdots, c_j^{l_j} \middle| x\right) \\
&= \prod_{k=1}^{l_j} P(c_j^k | x, c_j^1, c_j^2, \cdots, c_j^{k-1})
\end{aligned}
$$



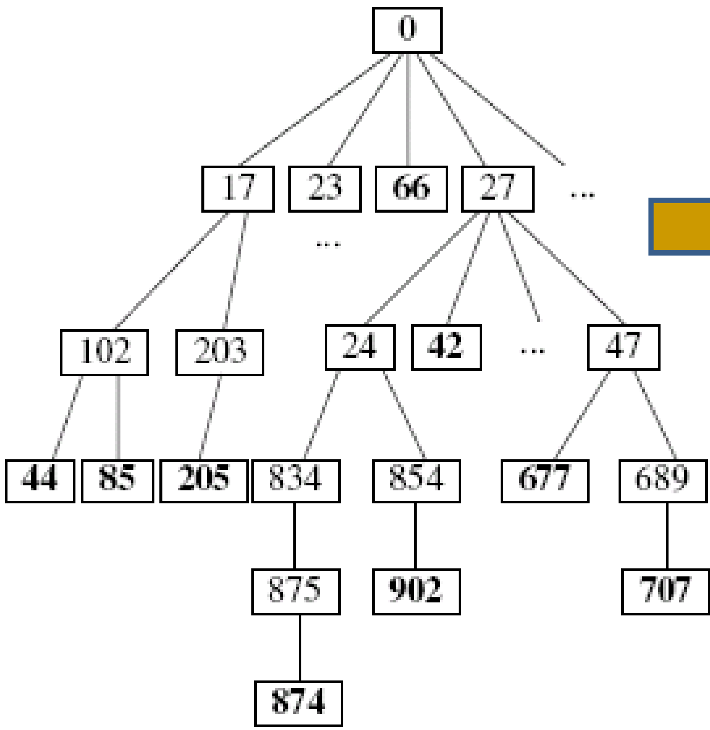
- Problem:
  - Slow
  - Few docs under each node
  - Top level docs too general

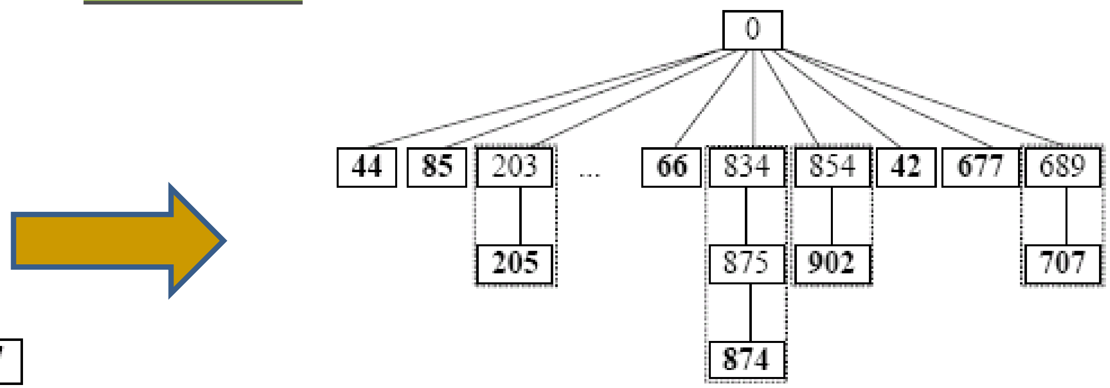
Ancestor-assistant Strategy

- To build a classifier for results of query Q, let Ci be a category of T(Q)
  - T(Q) is the pruned taxonomy tree of Q
- For each candidate Ci,
  - Collect training documents
    - from Ci,
    - Father of Ci, Cousins of Ci
    - Grandfather of Ci, Uncles of Ci
    - …
    - Until an ancestor is reached, which includes a competitor candidate Cj as descendent
- Let the union of these documents be D
- Label D by category Ci
- Build a classifier for Ci using D and using the flat strategy



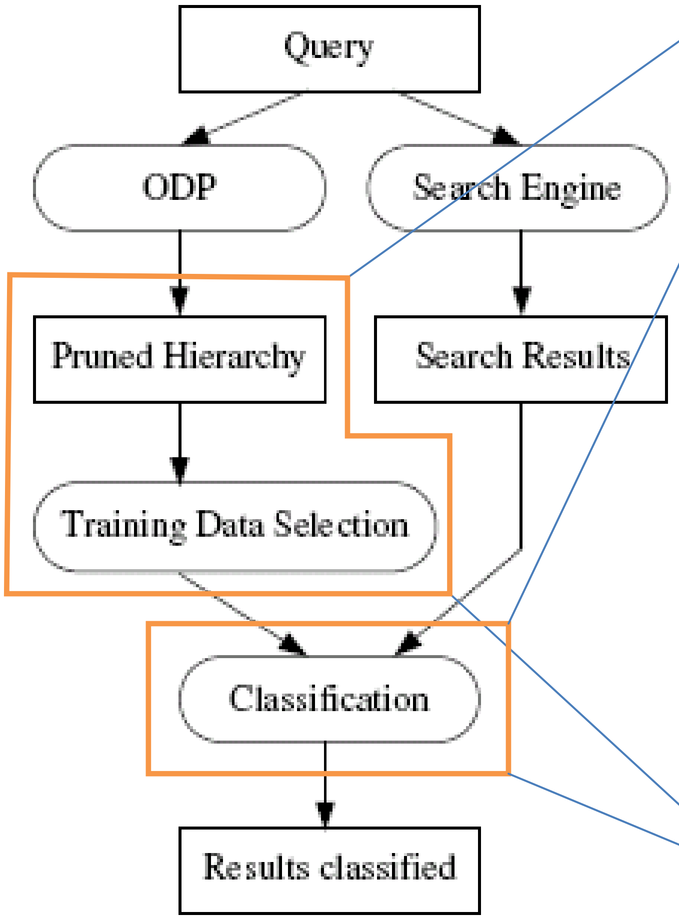
Ancestor-Assistant Strategy





- Find the highest ancestor that does not include another candidate node as descendent
- Borrow data from ancestors, and their descendents
  - Now 661.2 (vs. 21.6) per class

Deep Classifier: choice of classifier



Naïve Bayesian Classifier - fast

Flat Strategy

Standard NBC

Hierarchical Strategy

Discriminative NBC

Ancestor-Assistant Strategy

Next Research Question: How to choose a classifier

- A classifier is trained for each query
- Thus, efficiency is a concern!
  - Using SVM or other time-consuming classifiers would not be feasible
  - Using Naïve Bayesian Classifiers (NBC) is a good choice

$$\Pr(C_i \mid Doc) \propto \Pr(C_i) * \prod_{j=1}^{N} \Pr(word_j \mid C_i)$$

  - We can calculate the conditional probability table beforehand
  - Thus only need to multiply some factors in real time

Online Classification: choices

$$\Pr(C_i \mid Doc) \propto \Pr(C_i) * \prod_{j=1}^{N} \Pr(word_j \mid C_i)$$

- Two Problems with NBC:
  - Probability of each category in ODP is fixed
  - Probability of each category in search result varies w/ Q
    - Pr(Ci) not the same between training (ODP) and test data (top-100 search results)
      - Thus basic machine learning assumption violated, and we may need transfer learning, or…
  - Count(terms)/Count(categories) may be too small
    - when Count(categories) too large (>100),
    - The contribution of each term is tiny, thus not discriminative enough!
- We wish to make the contribution of each term much larger than in traditional NBC

Making NBC Fast and Accurate

- Two Assumptions:
  - Let Pr(Ci)=1/n, where n is # of classes, for all Ci
  - Pr(Ci|Doc) proportional to Pr(Ci|word j), which is proportional to # of categories per word
    - This is much more discriminative than Pr(word j|Ci)

$$\Pr(C_i \mid Doc) \propto \prod_{j=1}^{N} \Pr(C_i \mid word_j)$$

Time Complexity

- When testing a search result, only words occur in the snippets are considered.

- The time complexity for testing is $O(n * \log N + n * m + K)$,

  - *n* is the length of the **snippets**,
  - *m* is the number of category candidates
  - *N* is the size of the whole term vocabulary
- The first item denotes the time to convert snippets into word ID,
- the second item denotes the time to classify,
- *K* is the time for memory swapping
- However, the computational efficiency part needs to be explored much further in our future works
- Instead, in our experiments, we focused on accuracy only

Experiments

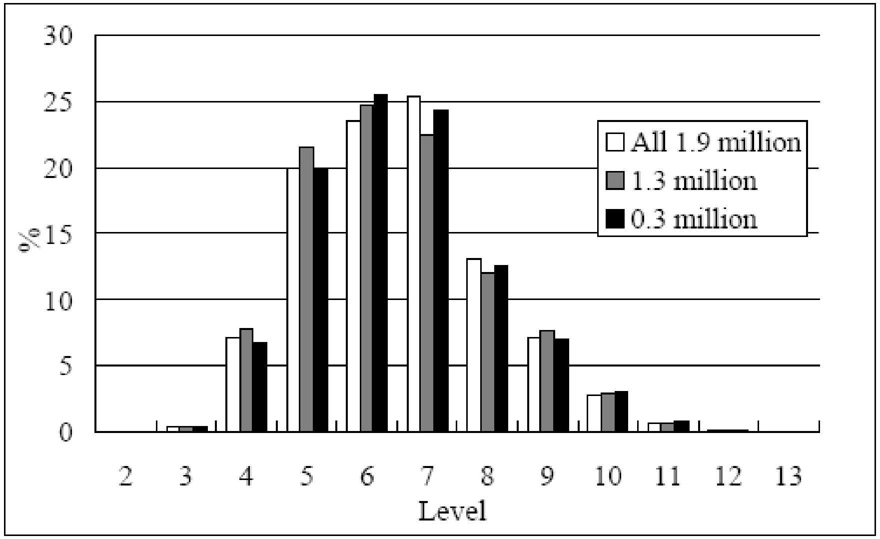
- We first collected 1000 popular queries from a search engine, and computed the distribution of their results among the top-level categories in ODP
- ~ 94.7% of the queries are distributed over less than six categories,
  - of which about 74.2% of queries are over three or less categories.
  - The two most widely distributed
    - *games* (in 14 top-level categories) and *books* (in 12 top-level categories).
- This indicates that
  - top-level categories may be too coarse for many queries
  - deep categories are necessary

Experimental Hypotheses

- The Ancestor-assistant strategy may outperform the hierarchical and the flat strategies
- The discriminative naive Bayesian classifier may outperform the traditional NBC
- The discriminative naive Bayesian classifier is much faster than SVM

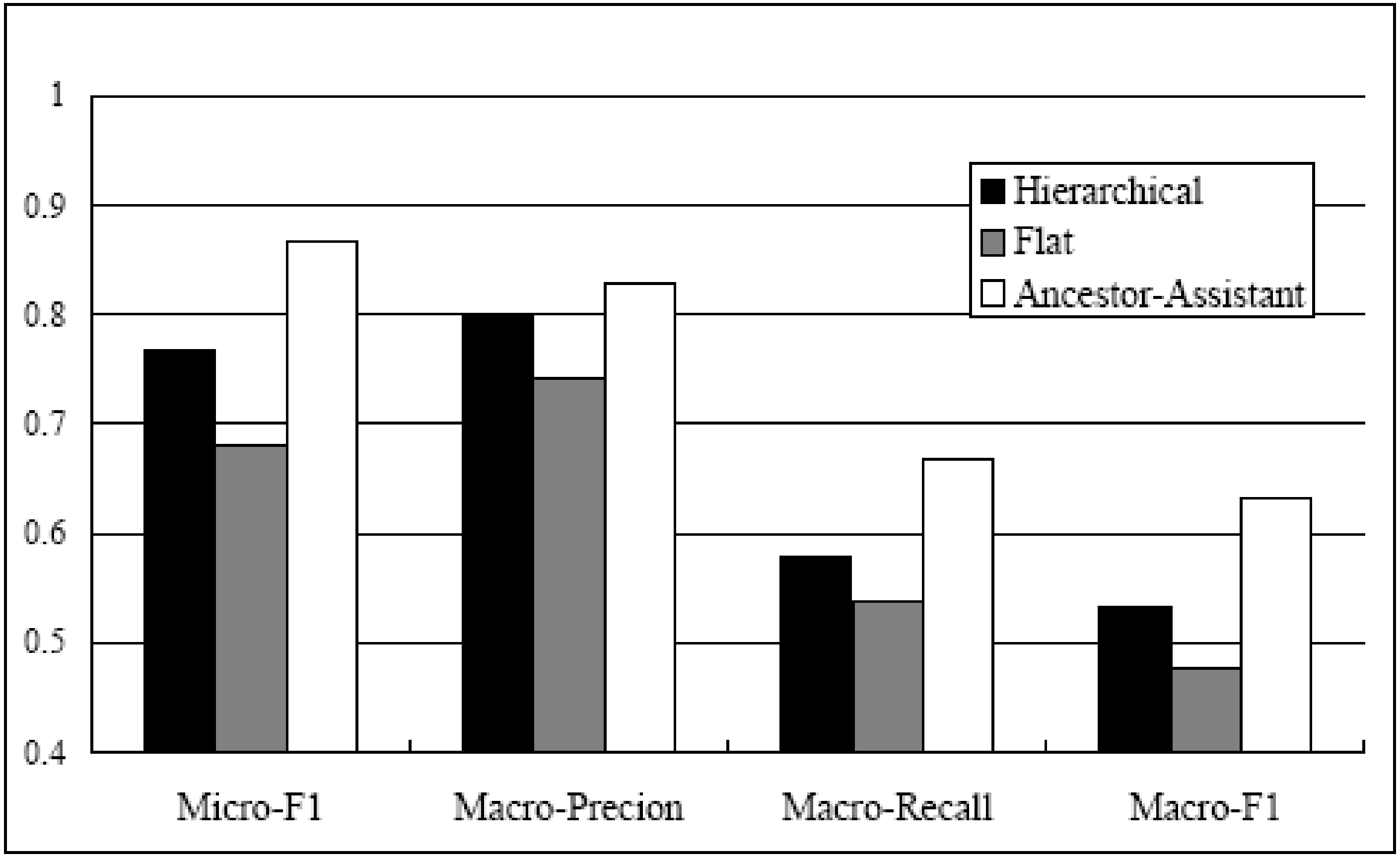
Evaluation Data

- Data Sets for Evaluation
  - Data Set I
    - Search results from simulated search engine
    - Randomly picking 100 from query log.
  - Data Set II
    - Case study: ambiguous queries.
    - Real search results from Google.



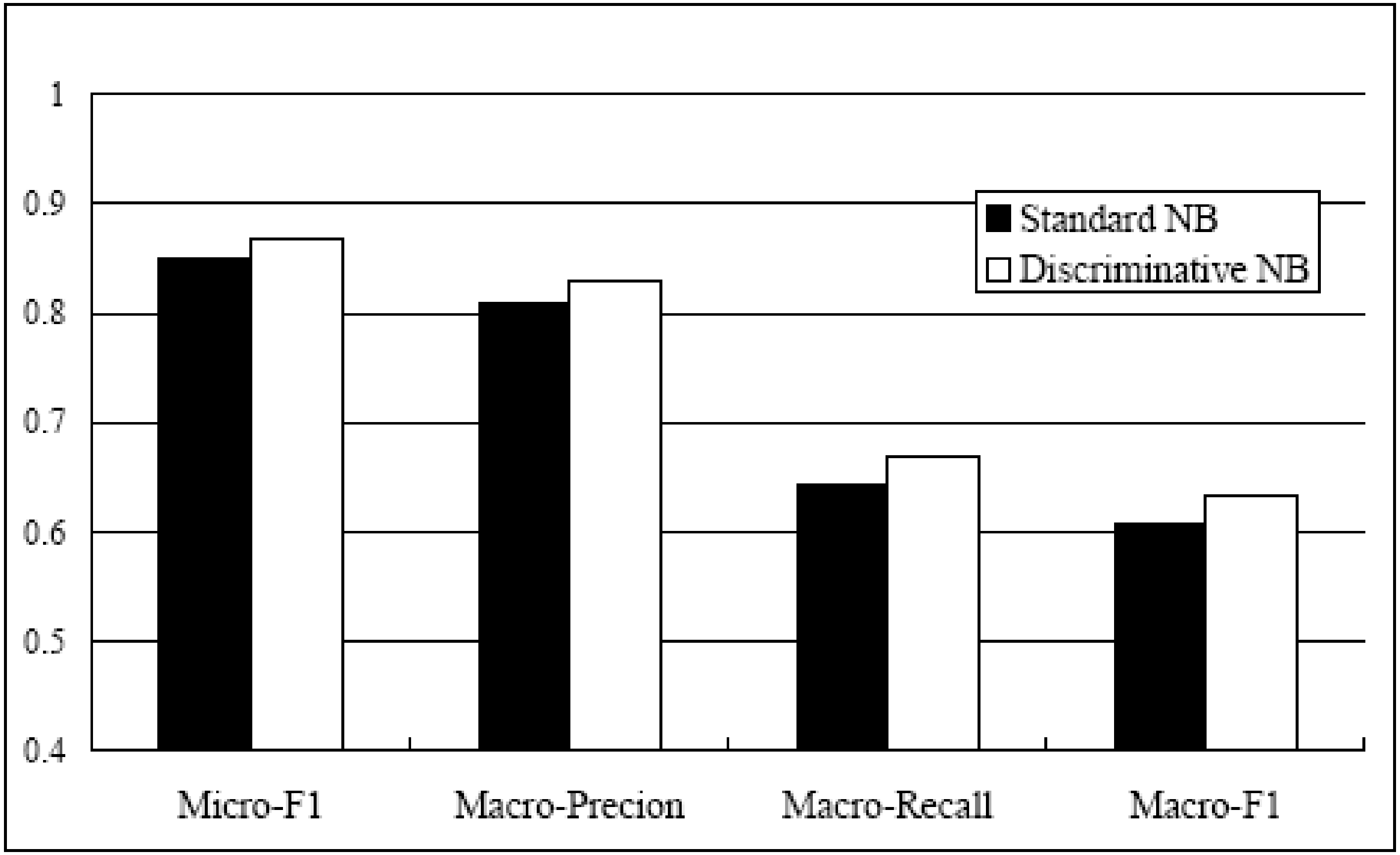
Pages	Categories
1, 297, 222	157, 927

Different Training Data Selection Strategies



Different Classifiers

(Each is averaged over all queries in the data set.)



Results on Queries
as function of training data selection strategy

Query	Micro-F1			Macro-Precision			Macro-Recall			Macro-F1		
	flat	hie	aa	flat	hie	aa	flat	hie	aa	flat	hie	aa
ajax	0.99	0.86	0.99	**0.92**	0.52	0.85	0.67	0.62	0.67	0.64	0.37	0.64
apple	**0.77**	0.21	0.72	**0.74**	0.28	0.68	0.33	0.26	**0.52**	0.29	0.19	**0.50**
dell	**0.67**	0.62	0.64	0.20	0.35	**0.39**	0.23	0.30	**0.54**	0.18	0.30	**0.33**
jaguar	0.61	0.41	**0.94**	0.59	0.51	**0.83**	0.30	0.53	**0.85**	0.26	0.45	**0.83**
java	**0.93**	0.29	0.83	0.48	0.34	**0.62**	0.37	0.20	**0.58**	0.32	0.20	**0.49**
saturn	0.71	0.41	**0.98**	0.60	0.69	**0.76**	0.43	0.63	**0.98**	0.29	0.50	**0.79**
subway	0.91	0.86	**0.94**	0.70	0.44	**0.71**	0.69	0.54	**0.88**	0.67	0.47	**0.78**
trec	0.60	0.52	**0.80**	0.34	0.34	**0.61**	0.54	0.54	0.46	0.38	0.38	**0.44**
ups	0.72	0.81	0.81	0.32	0.31	**0.33**	0.36	0.72	0.72	0.24	0.41	**0.43**
(average)	0.78	0.55	**0.85**	0.56	0.42	**0.64**	0.41	0.48	**0.69**	0.35	0.36	**0.58**

Conclusions

- Objective
  - Applying Hierarchical Classification to Search Result Categorization

Problem	Solution
Large Hierarchies	**Pruned** for each query
Few Training Data	**Ancestor-assistant** Strategy
Efficiency for Online Application	Faster and more effective **Discriminative** Naïve Bayesian classifier