International Conference on Digital Libraries
22-27 February, 2004 – New Delhi, IN

Organizing Digital Libraries by Automated Text Categorization

Fabrizio Sebastiani
Istituto di Scienza e Tecnologie dell'Informazione
Consiglio Nazionale delle Ricerche, Pisa, Italy
fabrizio.sebastiani@isti.cnr.it
http://www.isti.cnr.it/People/F.Sebastiani

(joint work with Henri Avancini and Andreas Rauber)

Overview of this talk

1. Introduction: Tackling Information Overload by Text Classification
2. Hierarchical Text Classification
3. Artificially Generated Hierarchies in Hierarchical Text Classification
4. Conclusion

Overview of this talk

1. **Introduction: Tackling Information Overload by Text Classification**
2. Hierarchical Text Classification
3. Artificially Generated Hierarchies in Hierarchical Text Classification
4. Conclusion

How should we tackle information overload? Two “basic philosophies”:

1. Build high-quality tools for searching an unstructured document base, such as $\boxed{\text{the Web}}$. This is the answer from $\boxed{\text{text search}}$.

2. Build high-quality tools for building structure into an unstructured document base. This is the answer from $\boxed{\text{automated text classification}}$ (ATC).

In this talk we will deal with approach (2), thus discussing how ATC can support the creation of digital libraries.

ATC is the discipline concerned with building $\boxed{\text{text classifiers}}$, i.e. tools that automatically "classify" textual documents into a finite set of m predefined classes ($\boxed{\text{classification scheme}}$). E.g.

- filing newspaper articles under thematic pages (e.g. HomeNews, Politics, Economy, Lifestyles, Sports);
- filing "classified ads" into classes (e.g. CarsForSale, RealEstate, Personals, . . .);
- filtering unsuitable content (e.g. deciding between Pornography and (**not** Pornography)) or junk mail (Spam vs. (**not** Spam));
- detecting semantic orientation (e.g. ThumbsUp vs. ThumbsDown).

- Historically, the most important application of ATC is $\boxed{\text{automated document indexing with controlled vocabularies}}$, i.e. classifying documents under classes (or “subject codes”, or “descriptors”, ...) from e.g.
  - the Library of Congress Cataloging Scheme;
  - Dewey Decimal System;
  - ACM Classification Scheme.
- Typically, this is a $\boxed{\text{multilabel}}$ task, i.e. the same document may be filed into any number of classes. This is implemented by performing m binary classification tasks, i.e. deciding between c_i and (**not** c_i) for $1 \leq i \leq m$.
- Automated document indexing with controlled vocabularies is a form of $\boxed{\text{automated metadata generation}}$.

A binary classifier for class c, i.e. one that decides between c and (**not** c)

- is built automatically, by $\boxed{\text{machine learning}}$ techniques, from a "training" set of documents preclassified under c and (**not** c);
- is tested by comparing its decisions with the human decisions encoded in a "test" set of documents preclassified under c and (**not** c).

Various ML techniques can be used in ATC; e.g. decision trees, probabilistic (Bayesian) classifiers, neural networks, boosting, support vector machines, etc.

Overview of this talk

- CompCat is an ongoing, internally funded project at ISTI-CNR for building a text classifier of scientific articles in the computer science domain.
- Main specifications:
  - The classes are the ones from the ACM Classification Scheme (version of 1998);
  - The classifier should be $\boxed{\text{interactive}}$, i.e. should suggest to the author of the document a list of classes ranked according to their estimated appropriateness for the document;
  - Main goal is enabling authors to annotate their own articles, thus avoiding the need for a specialized librarian.

Characteristics of this problem domain:

- the classification scheme has a tree structure;
- documents are to be classified into the "leaf" nodes of the tree only;
- there are thousands of classes, and many leaf classes may thus contain very few training documents.

The training and test sets are years from 1998 onwards of the ACM Digital Library.

"Trivial" solution to the problem: for each (internal of leaf) class c, train a binary classifier that decides between c and (**not** c).

Problems with this solution:

- $\boxed{\text{inefficiency}}$: since a document may belong to multiple classes, this means invoking thousands of classifiers for each document.
- $\boxed{\text{ineffectiveness}}$: the information provided by the hierarchical structure of the classification scheme is not exploited.

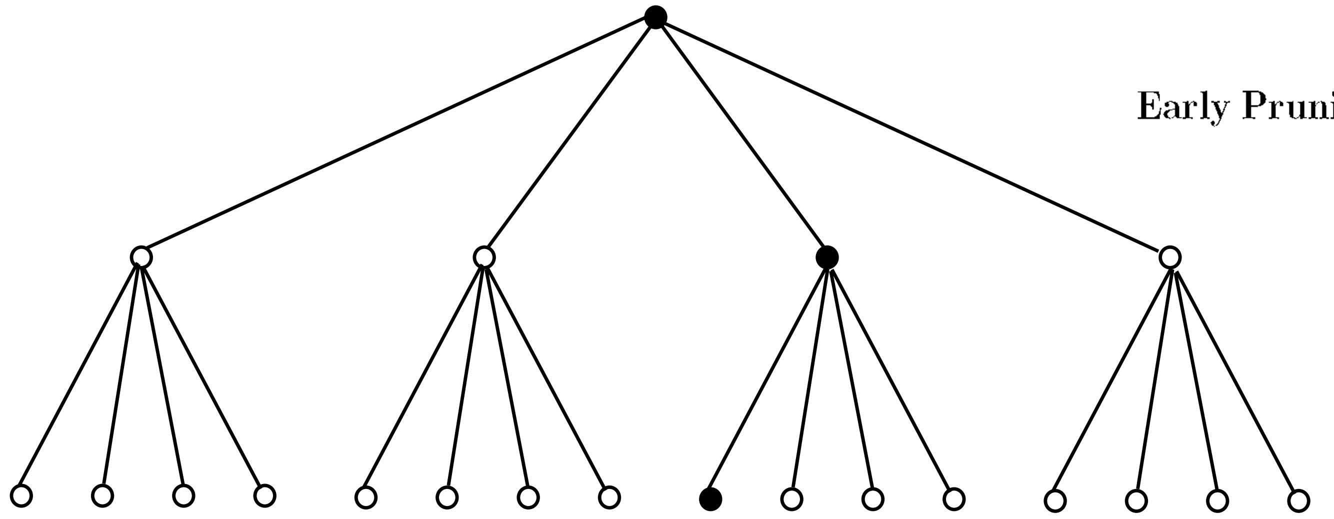
- “Standard” text classification techniques assume that no relation of dependency is defined among the classes belonging to the classification scheme.
- However, many classification schemes have a hierarchical structure defined on them. Exploiting the relationships of dependence (e.g. “more general than”, “more specific than”, “sibling of”, etc.) between the classes in the classification task may bring about higher effectiveness and higher efficiency.
- The goal of the CompCat project is to boost the effectiveness of classification by leveraging on the hierarchical structure of classes.

- There are three main intuitions that have been used in tackling the hierarchical nature of the set of classes:
  - Early pruning
  - Local selection of negative examples
  - Ancestor-based improvement of parameter estimation
- These intuitions are not mutually exclusive, and can be combined.

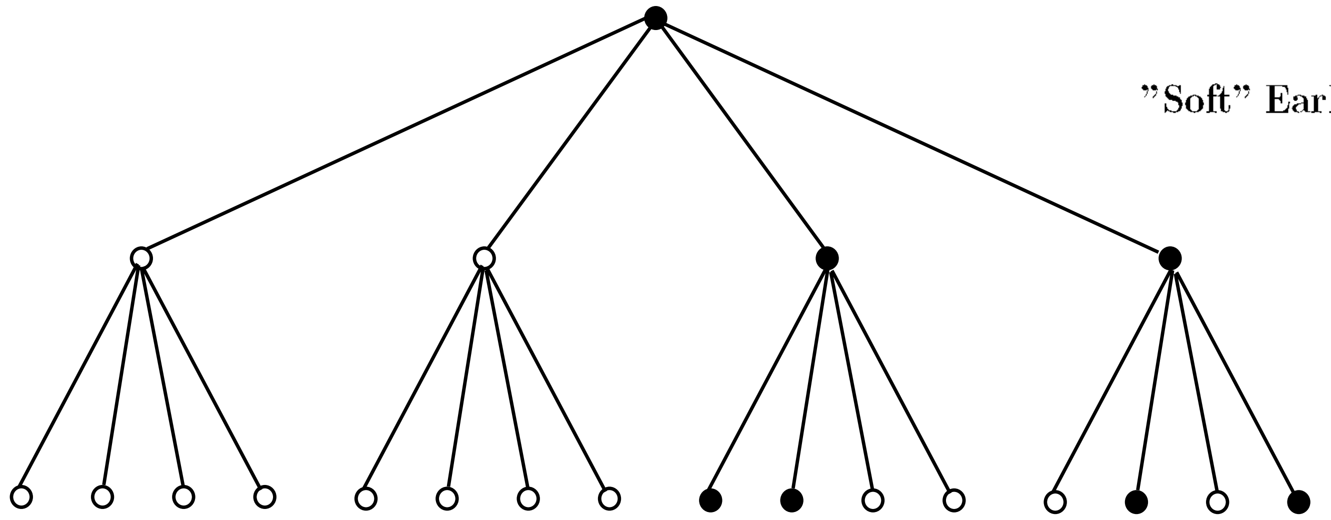
2.1. Early pruning

One may “greedily” select one or more children $c_1, \ldots, c_k$ of an internal node c_i, “prune” all the subtrees rooted in the other children, and recursively process the subtrees rooted in $c_1, \ldots, c_k$.

- Pros: we get an exponentially smaller set of classification problems
- Cons: a misclassification early up in the tree cannot be recovered. To make up for this, “soft” pruning should be enforced.



Early Pruning



"Soft" Early Pruning

2.2. Local selection of negative examples

- The hierarchical structure may be helpful in selecting better negative examples for each class c_i, i.e. choosing the negatives of c_i that belong to classes "close" to c_i ($\boxed{\text{quasi-positives}}$). These are much more helpful in learning an effective classifier than "generic" negatives.

- Advantages:
  - higher effectiveness, since the classifier can be more focused on capturing the fine-grained distinctions among closely related classes;
  - higher efficiency, since (a) the number of training documents is much smaller and (b) only the terms that appear either in the positives or in the quasi-positives of c_i need to be considered.

2.3. Ancestor-Based Improvement of Parameter Estimation

- The principle of $\boxed{\text{shrinkage}}$ is to obtain, in a probabilistic classification context, better estimates $\theta_{ik} = \Pr(t_k|c_i)$ for a leaf class c_i than the ones obtained by ML, since ML may yield unreliable estimates if c_i is an infrequent class.
- Shrinkage obtains a better estimate of $\Pr(t_k|c_i)$ by interpolating the ML estimate of $\Pr(t_k|c_i)$ with the ML estimates for t_k obtained for its ancestors. This trades specificity (which is typical of nodes deep in the hierarchy) for robustness (which is typical of nodes close to the root).
- [McCallum+98] shows that this can bring about significant improvements in effectiveness.

Overview of this talk

1. Introduction: Tackling Information Overload by Text Classification

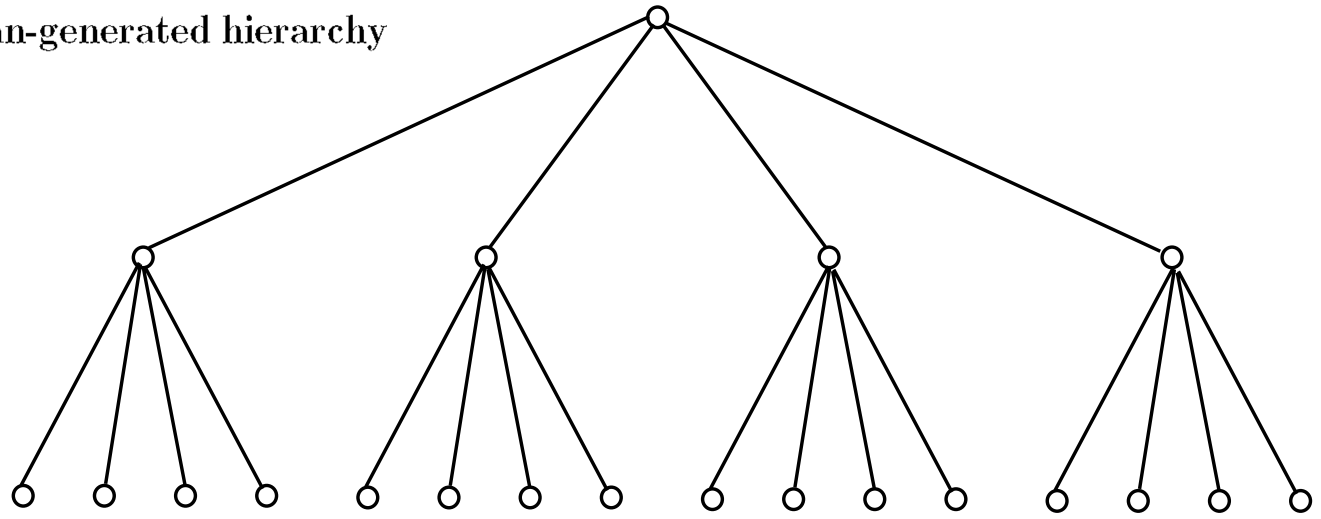
2. Hierarchical Text Classification

3. **Artificially Generated Hierarchies in Hierarchical Text Classification**

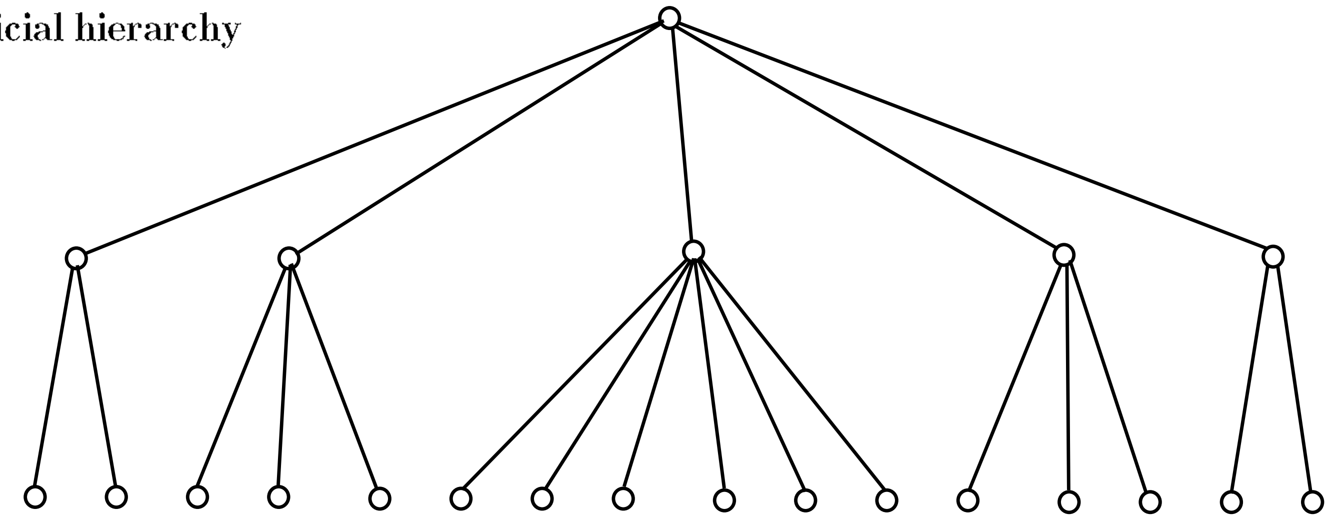
4. Conclusion

- In CompCat we apply these three insights in a novel way, i.e. we
  1. Discard the naturally occurring hierarchy (ACM Classification Scheme);
  2. Generate an artificial hierarchy by means of a hierarchical agglomerative clustering technique applied to leaf classes;
  3. Use this latter hierarchy instead of the naturally occurring one for learning the classifiers.
- This apparently counterintuitive idea relies on the hypothesis that, while naturally occurring hierarchical structures are advantageous for human understanding, a statistical classification algorithm can profit more from those generated by a statistical clustering algorithm.

Human-generated hierarchy



Artificial hierarchy



- The “user experience” will not be affected by this, since the artificially obtained hierarchy will only be used by the classifier; the user will continue to “see” the original hierarchy only.
- For this to happen, the clustering engine should “respect” the integrity of the leaf classes; i.e. the leaf classes of the original hierarchy must be all and the only leaf classes of the artificial hierarchy. For this to happen
  1. We generate a standard binary classifier for each class
  2. We apply the clustering algorithm not to the training documents, but to the classifiers, since we consider them “representatives” of the classes.

4. Conclusion

- The CompCat project is still ongoing: stay tuned ...
- Its central idea is

  Give hierarchies generated by humans to humans, give hierarchies generated by machines to machines.

- If this principle proves successful, this may well revolutionize the way we conceive classification, since it can be instantiated with
  - any "flat" text classification technique
  - any technique for exploiting the hierarchical structure in classification
  - any hierarchical agglomerative clustering technique