A Common Lisp Framework for Document Classification and Retrieval

Andrew J. Blumberg
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
blumberg@ai.mit.edu

April 24, 1996

Abstract

This paper describes the Document Classification Substrate (DCS) and accompanying protocols. The DCS is a framework of Lisp support code facilitating the prototyping and deployment of systems for automatic document classification and retrieval applications. The DCS design reflects the following observations concerning the problem of classification of texts.

1. Initial preprocessing (lexical feature extraction) is at least as significant as the choice of classification algorithm in terms of impact on classification performance. Richer lexical features will enable better classification.
2. The preprocessing stage needs to adapt in response to the particular document space just as the classification algorithm does.
3. Adaptive classification algorithms do not need to learn everything - encoding a priori operator information is important.
4. For purposes of rapid experimentation, the specific functional modules of a classification system should be strongly separated by an abstraction barrier.

The DCS provides separate management code for lexical feature extraction and classification, dividing the system into two layers. Interaction between these layers is strictly managed by a protocol which enables each layer to be abstracted away from the other layer, thereby facilitating independent development and experimentation. The lexical classification layer provides support for managing arbitrarily expressive lexical features which can dynamically adapt in response to the particular runtime environment. The categorization layer contains facilities for controlling arbitrary compositions of classification algorithms (each packaged and abstracted away from the others) and actively supports hybrid categorization algorithms; categorization algorithms mixing adaptive (typically statistical) classification algorithms with a priori symbolic rules.

The DCS was implemented in Common Lisp and has been tested with a set of lexical features based on interface code to the Wordnet database and a series of classification algorithms mixing various statistical approaches and probabilistic inference networks on an extensive document corpus.

1 Introduction

In recent years both the volume of information available electronically as well as the capacity to encode information in electronic format have increased tremendously. Accordingly, the need for tools enabling users to access this information has also grown. Access to stored information (in all formats) depends on indexing and classifying the data. In an informationally diverse, rapidly expanding, and highly decentralized medium like the World-Wide Web, this is a tremendous problem. Many difficulties with information retrieval and access are a direct result of exploding corpora coupled with a lack of adequate manpower or indexical concept to adequately index the incoming data.

The clear solution to such problems lie in systems which perform automatic document classification and retrieval - algorithmic methods for determining content information. However, such systems must satisfy fairly stringent constraints. The system must be efficient (both in time and space) due to the large volume of information in most realistic corpora and the fairly sharp limits of operator patience. Furthermore, to be directly utilizable by humans such a system must encode a functional approximation of human mechanisms for indexing content. To date, the most popular solution has been the Salton algorithm [S80] which employs frequency-based keyword matching coupled with clustering algorithms to categorize classes of documents. Although contemporary implementations of the Salton algorithm are extremely efficient and scale well to very large data sets (note the relative success of Digital's Alta Vista service operating on over 30 gigabytes of indexing data), they all suffer from the inherent problems associated with keyword based approaches. Specifically,

1. Keyword matching is only effective for content domains that are characterized by the repetition of particular indicator words at a rate significantly higher than background repetition (over the document universe as a whole).

2. Algorithms based on keyword matching are rather difficult to focus - typically vastly more information than is needed is returned to ensure a sufficient return.

Although the first objection stated above may appear tautological, it is a significant criticism to note. Consider situations in which each document may contain several disjoint (or only loosely related) topics - even if each topical domain is well characterized by keywords, the whole document may not present sufficiently frequent usage of the keywords relative to background usage. And of course categorizations based on stylistic ground (e.g. type of presentation) are not well characterized by keywords. The second criticism will no doubt be familiar to anyone who has employed a search over the World-Wide Web.

These are very difficult problems - and there are not yet clear and direct solutions. The DCS system was designed to facilitate the development and evaluation of new document classification systems - as well as embody certain design principles we believe are integral to improving on the performance of traditional document classification methods.

2 Design Principles

The following observations about the problem of document classification were embodied in the DCS.

1. The choice of lexical features is as important as the choice of specific classification algorithm. Ample experimental evidence as well as certain theoretical results suggest that most statistical learning techniques will perform at a roughly equivalent level - and thus the real difference in performance will come from the decision about what kinds of information to extract from the raw data. It's all in the features.

2. The preprocessing stage needs to adapt in response to the particular document space just as the classification algorithm does. Given the importance of the lexical features, the choice of which features are applied should adapt to specific runtime experience and the document universe.

3. Adaptive classification algorithms do not need to learn everything. In many domains classification performance can be dramatically improved by employing a priori rules to cover cases where the operator can provide specialized knowledge. In addition, this reduces the burden on the classification algorithm - which can be a significant consideration for statistical methods.

4. For purposes of rapid experimentation, the specific functional modules of a classification system should be strongly separated by an abstraction barrier. Given the roughly equal levels of importance of the classification algorithm and the lexical features, the development of these parts of the overall system should be decoupled to allow independent research.

3 General Architectural Overview

The DCS is organized into two layers - a lexical classification layer and a categorization layer. The two layers interact via a storage system (which might for example be a persistent database). This interaction is mediated by a strict protocol which enforces sufficient abstraction barriers so as to sufficiently modularize the system to enable mostly independent development of the levels. The roles of the layers are :

1. The lexical classification layer is responsible for the transformation of raw text into a sequence of tokens - where the tokenization process extracts lexical features.
2. The categorization layer is responsible for performing the actual categorization of documents based on the coded output of the lexical classification layer as well as stored information derived from previously presented documents (often with supervisory categorization information).
3. The storage layer is responsible for mediating the interaction of the two layers and for management of any long-term information necessary for adaptive algorithms (operating in either layer).

4 The Lexical Classification Layer

The lexical classification layer provides code which manages the extraction of lexical features from the raw text of the document. There are two major components to this transformation of the text. First, the text is converted to tokens via the action of one or more tokenizing functions - these perform very simple processing, such as removing all punctuation and extracting word-chunks by grouping characters delimited by spaces. Then, instances of lexical features are extracted from the sequence(s) of tokens. Lexical feature archetypes are stored as objects encapsulating a constraint function - this function serves to specify which tokens or token-groups are examples of the feature. The constraint facility possesses a subset of the functionality of the general constraint described in the CL-HTTP WWW walker [MBV96]. The constraint function can be arbitrarily expressive and can reference other existing constraints (which are managed by a simple internment facility). This constraint system enables less technically proficient individuals to write lexical feature constraints by applying simple combination rules to a library of existing constraints. The lexical features can be dynamically rebound and modified during runtime in response to the specific environment.

5 The Categorization Layer

The categorization layer provides code managing the actual assignment of a category labeling to a specific document. There are two major kinds of objects handled in this layer.

1. Primitive classification objects hold slots for a classification algorithm (typically statistical) and a series of symbolic rules.

   These rules are implemented as constraints - and activate in response to the constraints being satisfied. The interaction of the symbolic rules and the classification algorithm is controlled by precedence constraints. These are somewhat more general than the other classes of constraints found in this system, in that they compute an arbitrary function in response to the conditions being satisfied and hence do not necessarily have a binary output.
2. Higher-order classification objects encapsulate a set of primitive classification objects and precedence constraints controlling the weighting of the final output.

This architecture encourages the use of hybrid classification strategies - it facilitates the introduction of a priori information for classification. Furthermore, the abstraction barriers between different segments of the classification object permit easy change of the specific classification algorithm or the interaction of a group of algorithms without impacting the a priori knowledge - which can be a significant problem in some algorithms where incorporating existing knowledge is a nontrivial and domain-specific endeavor. Also note that this architecture supports both retrieval and categorization - retrieval is implemented simply by performing categorization based on synthesized document data produced from input keywords or sample document text.

6 The Storage Layer and Interaction Protocol

The storage module mediates the interaction between the lexical classification layer and the categorization layer. Presently the storage module is implemented as a object database storing appropriately indexed hash tables, but the specific implementation is irrelevant as long as the following interaction protocol is obeyed.

The lexical classification layer is permitted the following interaction with the storage module :

1. Storage of either arrays of feature-tokens or hash tables consisting of feature-tokens keyed by frequency along with indexing feature and document.
2. Extraction of information about the magnitudes of category memberships and specific classification of a given document.
3. Extraction of numerical information about the stored document universe - e.g. number of documents.

The categorization layer is permitted the following interaction with the storage module :

1. Extraction of the full feature-token profile for a specific input document.
2. Extraction of token counts over documents specified via constraint.
3. Extraction of token relevance metrics over documents specified via constraint.
4. Extraction of numerical information about the stored document universe - e.g. number of documents.
5. Storage of categorization information for a given document.

Furthermore, both layers are permitted arbitrary storage of shielded data - data that will only be extracted on the same side of the protocol.

7 Example

To illustrate and make concrete the discussion above, let us follow the progress of a document through a system implemented on the DCS system.

1. A document is presented to the system for processing.
2. The document is tokenized into a sequence of words (punctuation removed).
3. A set of feature-token sequences are produced by the application of lexical feature objects to the raw token sequence (one feature-token sequence corresponding to each lexical feature object).
4. These feature-token sequences are admitted to the storage module.
5. The categorization layer removes the present feature-token sequences from the storage module.
6. The active higher order classification object is applied to the input data.
7. The classification algorithm is run over the data and the symbolic rules are checked for activation.
8. In the course of this processing some token distribution information is extracted from the storage module.
9. Precedence constraints are applied to mediate the output of the classification algorithm with the output of the symbolic rules.
10. A categorization decision is output to the user.

8 Taxonomic Document Hierarchies

A special module has been implemented for use in situations where there is a preexisting set of categories which are arranged in a taxonomic hierarchy. In this situation there are typically constraints between different categories (for example, categories higher in the hierarchy subsume lower categories) and it can be advantageous to perform overall classification by descending the hierarchy, making restricted decisions at each stage. A general framework for storing this class of information is provided, as well as precedence constraints which can activate distinct classifiers at different levels of the hierarchy. Further, there are facilities provided to easily extract the specific influences of a particular category and allow for the ordering of category checking in order to maximize the pruning power of a given test.

9 Applications

The DCS has been tested on the corpus of White House released documents - employing a traditional Salton classification algorithm as well as a more sophisticated probabilistic inference technique, with lexical features performing simple syntactic extractions based on the Wordnet lexicon. As expected, experiments verify that richer lexical features and a priori information provide for improved classification performance. It was also interesting to note that this corpus provides an example of documents for which traditional keyword classification performs particularly poorly.

10 Conclusions

The DCS provides an architecture for research on automatic document classification methods and for actual deployment and use of a specific document classification system. It facilitates rich lexical feature preprocessing and hybrid classifiers supporting the encoding of operator knowledge as symbolic rules. The constraint formalism used broadly throughout the classifier provides a means of controlling the interaction and extraction processes in a quantifiable fashion. Furthermore, the interaction protocol and associated abstraction barriers enable rapid local experimentation to be done and independent development on the lexical feature modules and categorization algorithms.

11 Acknowledgments

Thanks go to John Mallery for a great deal of useful discussion and suggestion, Roger Hurwitz for helpful conceptual insights into the essentials of the classification process, and Mark Nahabedian for technical assistance and the original Wordnet interface code.

12 Bibliography

(M94) Mallery, J.C. "A Common Lisp Hypermedia Server," Proceedings of the First International Conference on the World-Wide Web, Geneva: CERN, 1994.
(MVB96) Mallery, J.C. and A.J. Blumberg and C.R. Vincent. "A Constraint-Guided Web Walker for Specialized Activities," - in these proceedings.
(S80) Salton, G. "Automatic Information Retrieval," Computer, 1980, 13(5):41-57.
(S91) Salton, G. "Developments in Automatic Text Retrieval," Science, 1991, 253:974-980.