Iterative Information Retrieval Using Fast Clustering and Usage-Specific Genres

Jussi Karlgren, Ivan Bretan, Johan Dewe, Anders Hallberg, Niklas Wolkert
SICS, University of Helsinki, and Telia Research AB

Abstract

This paper describes how collection specific empirically defined stylistics based genre prediction can be brought together together with rapid topical clustering to build an interactive information retrieval interface with multi-dimensional presentation of search results. The prototype presented addresses two specific problems of information retrieval: how to enrich the information seeking dialog by encouraging and supporting iterative refinement of queries, and how to enrich the document representation past the shallow semantics allowed by term frequencies.

Searching For More Than Words

Today's tools for searching information in a document database are based on term occurrence in texts. The searcher enters a number of terms and a number of documents where those terms or closely related terms appear comparatively frequently are retrieved and presented by the system in list form. This method works well up to a point. It is intuitively understandable, and for competent users and well edited document bases it will give a predictably mediocre result. It also has obvious drawbacks.
Term frequencies provide a representation of document content which suffers from being both shallow and sparse: shallow, in that a text has more facets and features than a list of terms will be able to represent; and sparse, in that it usually is difficult for users to pinpoint which term to search for in a multi-laced hierachy of hyponymial relations and sets of near synonyms, related terms, or other variants.

We focus on two specific drawbacks of the traditional information retrieval search process. Firstly, as has been acknowledged by many recent research projects and some recent applications, searches are seldom one-shot affairs. Typically a search is improved and refined iteratively, until the retrieved set seems good enough, by some metric. Thus, the interface should support persistence and incremental refinement. Secondly, the objects of study are more complex than usually is assumed: document topic is more than term frequencies, and documents are more than the topic they are about.

Information Seeking Dialog

Dialog design is not one of the central fields of computational linguistics; most design principles have been made outside the field, in by researchers generally interested in human-computer interaction. This is a pity, for two reasons. First, linguists -- even computational -- have a lot to offer dialog design: language has a lot to do with dialog. Second, language technology applications need dialog design as well as parsing algorithms or terminology statistics.

We have put some thought in our dialog design. We want our system to transcend the typical one-shot dialogs of most fielded information systems today: ``enter your search terms'' -- ``browse the resulting list''. Our dialog builds on incremental specialization. The pipeline

architecture of the system allows users to specialize a query by specializing the resulting set to only some of the clusters retrieved initially.

The query and the resulting clusters are represented continuously: retrieved documents are inserted in the appropriate cluster irrespective of ranking. Specialization of a query can be done without figuring out specialized terms, since the enriched representation of documents can be used instead. The clusters themselves represent topic and genre: they can be selected for further inspection.

Nowhere is there a list format - it should be uninteresting to judge documents by internal processing order if topic and genre are displayed instead.

Seeking information is seldom done with a clear picture of the goal in one's mind. Typically users will need to familiarize themselves with the available materials to find how they will express their information need to the system -- by some preliminary searches, some back-tracking, and some reformulation of previous tries. An information retrieval interface should support persistent and manipulable dialog objects which represent the system's understanding of the search.

Topic and Document Representation

What effect a search term has in terms of retrieval is usually surprising to a user. The documents it retrieves may vary: if the term or term set is broad, by topic; if the topic or set of topics is broad, by style. An information retrieval interface should not try to shield the user from this variation but display it and allow the user to act on the results, iteratively.

Interaction - A Prototype Implementation

The assumption of interface designers is often that users cannot cope with complex interfaces and that the goal is to reduce the amount of information users are subjected to. We believe this not to be true. If the information is presented in a manner well founded in the task and usage context users will accept it.

We have built a fully functional prototype system, DropJaw, to experiment with iterative search. Our experiment database is the World Wide Web, and DropJaw bases its searches for web documents by the user entering terms, as in a traditional system. Rather than producing ranked lists of output based on term occurrence, DropJaw displays the distribution of the resulting set over two dimensions: dynamically generated topical clusters and user-defined, document-base oriented genre. The two-dimensional document space is displayed on a work board or matrix for further user processing.

DropJaw consists of two main modules: Easify, a presentation module, and Chunkify, a classification and clustering engine. DropJaw is implemented in C++ and runs as a stand-alone application under Microsoft Windows. It does not include a indexing or search engine -- currently we make use of any of several commercial search engines as indexers. DropJaw retrieves the documents the indexing service returns and reanalyzes them locally. The architecture is a pipeline model, where different processing components in Chunkify can be connected or disconnected -- by user requests to Easify -- to the pipeline
which delivers a stream of documents from the World Wide Web to the user.

Appearance

The main goal for the graphic design and the interaction design has been to build an interface which appeals to the untrained Internet user and is interesting and fun to use.

In most other cases information visualization tools have a somewhat scientific look and support an interaction designed for specialist users. Most are graphically designed with mainly the function in mind, rather than aesthetic quality. This most likely is a result of using standard elements from standard programming tools and little or no involvement of graphic or industrial designers in the design process.

Information visualization before the computer era (Tufte, 1983), in many cases is aesthetically very beautiful and pleasant to look at -- without compromising clarity or usability. There is no reason why the introduction of computers into the information visualization process should compromise aesthetics. Lack of aesthetic qualities may distract and bore the user and indirectly cause the loss of potentially useful information; usability is a function of both underlying functionality and aesthetic quality.

The graphic interface, Easify, is designed to feel playful and fun but without sacrificing a sense of strict reliability. Because of the high information density in the interface, colors have been used to help user choice, without dominating the interface look.
The use of strong colors and extreme shapes many times bores users after extensive use; this interface has a combination of soft, sober colors and simple but interesting shapes built from carefully chosen and composed non-standard graphic elements.

Functionality

The prototype was designed with a high degree of interactivity in mind. Users initiate the interaction by entering a query and clicking the search button. Easify sends the current parameter set -- number of initial clusters, i.a. -- to Chunkify, and starts the background pipeline. Chunkify consults the indexing engine for a list of likely documents, and starts retrieving candidate documents from the Internet. After initial clustering, Chunkify starts delivering documents to Easify for presentation.

The pipeline design leaves the user in control of the interface at all times instead of locking the interaction: there are indicators to show that the classification engine is running in the background. A stop button lets the user halt the background processes;
a clear button clears the current document set from the display.

There are many choices that can be made at each step of processing: the user can drag and drop subsets of the presented document set to a regrouping panel to request Chunkify to regroup the clusters. This does not stop the first filter in the pipeline from continuing its work: DropJaw simply adds a finer-grained Chunkify filter to the end of the pipeline. The first set keeps accruing, and can be returned to if the finer analysis turns out less useful. More information about the documents can be found in pop up menus, and the documents themselves are available for perusal at all times.

Document Representation

Rather than defined solely by their list ranking, Easify represents documents as members of topically and stylistically homogenous clusters in the interface.

Stylistic variation among texts shows through stylistic items: observable choices of linguistic items. Stylistic items can be observed on any level of linguistic abstraction: lexical, for the choice between words of similar meaning but different connotations; syntactic, for the choice between equivalent constructions with different communicative import; textual, for decisions of textual organization. Each stylistic item is of little import in itself, but taken together the entire set is indicative of systematic differences. A set of documents with a perceived

consistent tendency to make the same stylistic choices is called a genre or, specifically, if it has an established communicative function, a functional style (see e.g. Enkvist, (1973). Stylistic variation between genres or language varieties can be detected reliably using a large battery of quite simple stylistic items such as pronoun counts or relative frequencies of certain types of constructions such as agentless passives (Karlgren and Cutting, 1994), utilized for authorship determination by simple calculations of average word length distributions (Mendenhall, 1887), and with some success predictively for information retrieval (Harman, 1996).

Balanced Corpora for Testing

There is no well-established genre palette for Internet materials, such as one can find for printed documents. We need to create one to know what test materials to collect. This involves the risk of circular evaluation of self-established criteria for success, the interpretation of vaguely expressed and imperfectly understood user expectations, and the need to face the very real engineering problem of putting genre distinctions to predictive use for retrieval purposes.

In most computational stylistics, genre has mostly been equated or based on text source: Wall Street Journal text archive, personal letters, technical documentation (Francis and Kucera, 1982; Källgren, 1990; Karlgren and Cutting, 1994). We find this unsatisfying, and wish to find a better foundation for analysis; we believe user perceptions are central to this task. We have built our genre palette (see Table 1) through interviewing users: trying to define genres that are both reasonably consistent with what users expect and observable and conveniently computable using measures of stylistic variation as outlined in the previous section. This work is d escribed in a previous report (Dewe et al, 1998).

Genre		**Sample**
1	Informal, Private Personal home pages.	128
2	Public, commercial Home pages for the general public.	197
3	Interactive pages Pages with feed-back: customer dialogue; searchable indexes.	73
4	Journalistic materials Press: news, editorials, reviews, popular reporting, e-zines.	94
5	Reports Scientific, legal, and public materials; formal text.	113
6	Other running text	160
7	FAQs	12
8	Link Collections	148
9	Other listings and tables	225
10	Discussions Contributions to discussions; Usenet News material.	24
11	Error Messages	184
Total		**1358**

Table 1: The current genre palette

Recognizing genres automatically

The genre palette, besides being intuitively understandable, needs to be workable for automatic analysis. We calculate a quite large number of textual features for each individual text and work them together for a categorization decision using a machine learning algorithm. The pioneering work by Douglas Biber(1989) on computational corpus-based stylistics has been descriptive rather than predictive, aiming to find distinctions between different registers opr varieties of spoken and written language. It has made use of large numbers of stylistic features collected from previous, non-computational work and weighing them together using standard methods from multivariate statistics. We use this work as a basis for ours. Most of Biber's features we use here are rather lexical in nature, for ease of processing: the relative frequency of certain classes of words such as personal pronouns, emphatic expressions, or downtoning expressions, for instance. We add more general textual and genre specific features: relative number of digits, or average word length, for instance. Others yet are vectored specifically to the Internet material we have been using for experimentation: number of images or number of HREF links in the document, for instance. We normalize the measurements by mean and standard deviation, and combine them -- 40 of them, at present -- into simple if-then categorization rules using C4.5, a non-parametric categorization tool (Quinlan, 1993).

If
- there are more "because" than average,
- longer words than average,
- type-token ratio is above average,
then
- the object is of class Textual
with
- a certainty of 90.0%.

We have a few dozen rules to categorize texts into one of the eleven genres defined in the above sections. The genres partition into two major hypercategories: textual (04, 05, 06, 07, 10) and non-textual (01, 02, 03, 08, 09, 11); each of them in turn splits to one of five or six sub-categories. These splits are of varying quality: the first does quite well, something like a ninety per cent success rate, while the subsplits make the wrong choice somewhere between once in three or four times.

With additional features and a better defined genre palette results will improve. However, to get really useful results the categorization should not be exclusive. Every object should potentially be of several genres.

Clustering By Content -- A Very Simple Clustering Algorithm

The similarity measure for comparing topical document representations with each other and with cluster centroids is in most respects based on a standard tfc metric, as defined by Salton and Buckley (1982): standard term frequencies, cosine length normalization, and a standard collection frequency (idf) measure to factor in collection and domain specific terminology variation.

Since the emphasis is on a high degree of interactivity, a quick and dirty clustering must be used for the initial document sets. We work on the assumption that low number, up to 5, of clusters in the interface is desirable (Hearst and Pedersen, 1996). Since the search engine

itself is not included in the system setup, there is no time to wait for all the data to arrive. If the algorithm would have recourse to the entire document set, the initial clusters could be formed from a random set, as in Scatter/Gather (Cutting et al, 1992); in our case the first clustering must proceed on the assumption that the first documents to arrive are a representative subset of the entire retrieved set. This is a daring assumption and most likely overly optimistic, but enables us to start clustering sooner, and to restrict the use of the computationally expensive -- on the order of N squared -- hierarchical clustering by defining the first clusters on a small number of documents: the first 10-50 documents, which number is adjustable in the interface. The clustering itself is a variant of the standard metrics: a hierarchical agglomerative group-average algorithm (e.g. Jain and Dubes, 1988).

After deciding the first i clusters (with i adjustable in the interface) the following documents are each routed to one of them. A simple assign-to-nearest algorithm is used to decide cluster membership. The clusters are represented by their centroids -- the N-dimensional centre points -- as a list of the terms with highest weight, and the matching distance is the same as for the hierarchical clustering algorithm above. Currently the clusters are not refined on the fly; this would be easier to implement with the entire clustering stage coupled tighter to the search engine itself -- these algorithms are substitutes for an integrated system.

Evaluating The Method And The Design

Developer satisfaction

With rather small effort, this both user- and technique-centered development method has pinpointed certain weaknesses in the design while at the same time encouraging us to pursue its strong sides further. Users liked the basic idea, and since most components are based on empirically evaluated knowledge, they proved immediately useful.

Subject satisfaction

The clustering algorithms are very simple; the genre classification just barely flies; some interface niceties not completely in place yet -- but the prototype runs well enough for the interaction to be tested and evaluated and the design to be refined.

We gave a small number of subjects two simple retrieval tasks each, one using Easify and the other using Altavista. The order between questions and interfaces was varied between subjects.

12 subjects were chosen to be a representative test population. 6 subjects were male, 6 female; all of age 25-30; and while averages are a poor measure for small populations, in this case the subject pool hit the mark quite well: they assessed themselves as averagely experienced internet who use search services occasionally, and have a reasonable understanding of how search services work.

The tasks were

- Find an album or a concert review about Oasis.
- Find a list of hotels on Malta.

The subjects were given an introduction to the main ideas with Easify, and shown an example search with the system: all users had used Altavista previously. They were then given the tasks. The experiment supervisor gave tips on query formulation if the subjects seemed to get stuck. The subjects used about 5 minutes on each task.

The subjects did not do well on the tasks: Altavista search produced better results. In spite of this, the subjects liked and understood the interface prototype, with some remarks. Some subjects were confused by the changing cluster texts, but were comfortable with it after a short explanation. Some wanted the genres to be refined as well, to sub-genres. In spite of the low response times, all but one of the subjects liked the interface in itself.

Most subjects used the interface as we had intended them to, and many searched for documents in the genres we intended the results to show up in. The genre determination algorithm in its current state makes a mistaken choice too often for some of the genres: the hit rate must be raised to nine out of ten for the concept to work better. Flexible genre determination would help here -- a document should be allowed to fall into several genres rather than exclusively one.

Better cluster headings would improve subjects' chances of finding their way through the document space. This will be absolutely necessary if Easify is to work with very large document bases, where inspection of documents will be impossible until after a large number of categorization iterations.

The search engine only delivers the 200 top ranked documents for each query, and this gives the formulation of the query too large weight in determining if a correct answer can be found. With a closer integration this bottleneck can be eliminated.

With these improvements -- all of them included in future planned project work and none of them surprising or dismaying to the design team -- Easify will be able to fulfil its promises to compete successfully with single-shot single-modal single-channel search machinery.

References

Douglas Biber. 1989. "A typology of English texts", Linguistics, 27:3-43.

Douglass R. Cutting, David R. Karger, Jan O. Pedersen, John W. Tukey. 1992. Scatter/Gather: A Cluster-based Approach to Browsing Large Document Collections. Proceedings of the Fifteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Copenhagen. New York: ACM.

Johan Dewe, Ivan Bretan, and Jussi Karlgren. 1998. "Assembling a Balanced Corpus from the Internet". 11th Nordic Computational Linguistics Conference, Copenhagen. Copenhagen University.

Nils Erik Enkvist. 1973. Linguistic Stylistics. The Hague: Mouton.

W. N. Francis and F. Kucera. 1982. Frequency Analysis of English Usage. Houghton Mifflin.

Donna Harman (ed.). 1996. The Fourth Text REtrieval Conference (TREC-4). National Institute of Standards Special Publication 500-236. Washington.

Marti A. Hearst and Jan O. Pedersen, Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results. 1996. Collections. Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Zurich. New York: ACM.

Anil K. Jain and Richard C. Dubes. 1988. Algorithms for Clustering Data. Engelwood Cliffs, New Jersey: Prentice Hall.

Jussi Karlgren. 1996."Stylistic Variation in an Information Retrieval Experiment" In Proceedings NeMLaP 2, Bilkent, September 1996. Ankara: Bilkent University. (In the Computation and Language E-Print Archive: cmp-lg/9608003).

Jussi Karlgren and Douglass Cutting. 1994. "Recognizing Text Genres with Simple Metrics Using Discriminant Analysis", Proceedings of the 15th International Conference on Computational Linguistics (COLING 94), Kyoto. (In the Computation and Language E-Print Archive: cmp-lg/9410008).

Gunnel Källgren. 1990. The First Million is Hardest to Get: Corpus Tagging. Proceedings of the 13th International Conference on Computational Linguistics (COLING-90) Hans Karlgren (ed.), Helsinki.

T.C. Mendenhall. 1887. "The Characteristic Curves of Composition." Science 9: 237-49.

J. Ross Quinlan. 1993. C4.5: Programs for Machine Learning. San Mateo: Morgan Kaufmann.

Gerard Salton and Christopher Buckley. Term-Weighting Approaches in Automatic Text Retrieval. Information Processing and Management 24 (5) 513-23.

Tomek Strzalkowski, Louise Guthrie, Jussi Karlgren, Jim Leistensnider, Fang Lin, Jose Perez-Carballo, Troy Straszheim, Jin Wang, Jon Wilding. 1996. "Natural Language Information Retrieval: TREC-5 Report" Proceedings of The Fifth Text REtrieval Conference (TREC-5). Donna Harman (ed.). National Institute of Standards Special Publication. Washington.

Edward R. Tufte. 1983. The Visual Display of Quantitative Information.