A Concept-Driven Algorithm for Clustering Search Results

Stanisław Osiński and Dawid Weiss, *Poznań University of Technology*

Search engines rock! Right? Without search engines, the Internet would be an enormous amount of disorganized information that would certainly be interesting but perhaps not very useful. Search engines help us in all kinds of tasks and are constantly improving result relevance. So, does even the tiniest scratch exist on this perfect image? We're afraid so.

The Lingo algorithm combines common phrase discovery and latent semantic indexing techniques to separate search results into meaningful groups. It looks for meaningful phrases to use as cluster labels and then assigns documents to the labels to form groups.

Contrary to popular belief, search engines don't answer questions. They merely provide fast access to the information people put on the Web. In fact, popular search engines, as opposed to query-answering systems, return Web pages matching the user's question rather than the question's answer. Luckily, this works most of the time, because people tend to place questions together with answers. The real problem arises when the information need expressed by the query is vague, too broad, or simply ill-defined. Then the range of covered topics becomes unmanageably large, confusing, and of doubtful value to the user.

Search-results clustering aims to present information about the matching documents (see the "Related Work in Text Clustering" sidebar). It's like taking a step backward to grasp a bigger picture—we no longer care about individual documents, but about some underlying semantic structure capable of explaining why these documents constitute a good result to the query. To find this structure, we set a few goals:

- identify groups of similar documents,
- discover a textual description of the property making the documents similar, and
- present these descriptions to the user in document clusters.

Our approach reverses the traditional order of cluster discovery. Instead of calculating proximity between documents and then labeling the discovered groups, we first attempt to find good, conceptually varied cluster labels and then assign documents to the labels to form groups. We believe that only the commercial search engine Vivisimo (www.vivisimo com) uses a similar order of cluster discovery, but the details of that algorithm are unknown.

The Lingo algorithm

According to the *Collins English Dictionary*, *lingo* is "a range of words or a style of language which is used in a particular situation or by a particular group of people." Each time a user issues a query on the Web, a new language is created, with its own characteristic vocabulary, phrases, and expressions. A successful Web-search-results clustering algorithm should speak its users' lingoes—that is, create thematic groups whose descriptions are easy to read and understand. Users will likely disregard groups with overly long or ambiguous descriptions, even though their content might be valuable. A Web search clustering algorithm must therefore aim to generate only clusters possessing meaningful, concise, and accurate labels.

In conventional approaches, which determine group labels after discovering the actual cluster content, this task proves fairly difficult to accomplish. Numerical cluster representations might "know" that certain documents are similar, but they can't describe the actual relationship.

In the Lingo *description-comes-first* approach, careful selection of label candidates is crucial. The algorithm must ensure that labels are significantly different while covering most of the topics in the input snippets. To find such candidates, we use the vector space model (VSM) and singular value decomposi-

Related Work in Text Clustering

Originally derived from full-text clustering and classification, topic-grouping of search results has its subtleties. Contextual descriptions (*snippets*) of documents returned by a search engine are short, often incomplete, and highly biased toward the query, so establishing a notion of proximity between documents is a challenging task.

Clustering systems initially used classic information retrieval algorithms, which converted documents to a *term-document matrix* before clustering. We use the same technique but combine clustering and smart cluster label induction to provide stronger cluster descriptions. The Scatter-Gather system,[1] for example, used the Buckshot-fractionation algorithm. Other researchers used agglomerative hierarchical clustering (AHC) but replaced single terms with *lexical affinities* (2-grams of words) as features.[2]

Unfortunately, strictly numerical algorithms require more data than is available in a search result. Raw numerical outcome is also difficult to convert back to a cluster description that human users can understand. *Phrase-based* methods evolved to address this problem. The suffix tree clustering (STC) algorithm[3] and the Multisearch Engine with Multiple Clustering system[4] form clusters based on *recurring phrases* instead of numerical frequencies of isolated terms. STC, for instance, implicitly assumes correlation between a document's topic and its most frequent phrases. Clustering in STC is thus basically finding groups of documents sharing a high ratio of frequent phrases; cluster descriptions are a subset of the same phrases used to form the cluster.

Phrase-based methods, albeit simple, usually yield good results. Unfortunately, when one topic highly outnumbers others, the algorithms usually discard smaller clusters as insignificant. Recently, researchers have applied matrix decomposition methods to the term-document matrix to fully explore the underlying latent topic structure and provide a diverse cluster structure. For example, researchers have used singular value decomposition for this purpose,[5] and nonnegative matrix factorization in a more general context of document clustering.[6]

To our knowledge, no other researchers have successfully integrated numerical and phrase-based methods. Lingo bridges existing phrase-based methods with numerical cluster analysis to form readable and diverse cluster descriptions.

References

1. M.A. Hearst and J.O. Pedersen, "Reexamining the Cluster Hypothesis: Scatter/Gather on Retrieval Results," *Proc. 19th ACM SIGIR Int'l Conf. Research and Development in Information Retrieval,* ACM Press, 1996, pp. 76–84.
2. Y.S. Maarek et al., *Ephemeral Document Clustering for Web Applications*, tech. report RJ 10186, IBM Research, 2000.
3. O. Zamir and O. Etzioni, "Grouper: A Dynamic Clustering Interface to Web Search Results," *Computer Networks*, vol. 31, no. 11–16, 1999, pp. 1361–1374.
4. P. Hannappel, R. Klapsing, and G. Neumann, "MSEEC: A Multisearch Engine with Multiple Clustering," *Proc. 99 Information Resources Management Assoc. Int'l Conf.,* Idea Group Publishing, 1999.
5. Z. Dong, *Towards Web Information Clustering*, doctoral dissertation, Southeast Univ., Nanjing, China, 2002.
6. W. Xu, X. Liu, and Y. Gong, "Document Clustering Based on Nonnegative Matrix Factorization," *Proc. 26th Ann. Int'l ACM SIGIR Conf. Research and Development in Information Retrieval*, ACM Press, 2003, pp. 267–273.

tion (SVD), the latter being the fundamental mathematical construct underlying the latent semantic indexing (LSI) technique.[1]

VSM is a method of information retrieval that uses linear-algebra operations to compare textual data. VSM associates a single multidimensional vector with each document in the collection, and each component of that vector reflects a particular key word or term related to the document. (We use "term" to refer to a single word and "phrase" to refer to a sequence of terms.) This lets us represent a set of documents by arranging their vectors in a *term-document matrix*. The value of a single component of the term-document matrix depends on the strength of the relationship between its associated term and the respective document.

Unlike VSM, LSI aims to represent the input collection using concepts found in the documents rather than the literal terms appearing in them. To do this, LSI approximates the original term-document matrix using a limited number of orthogonal factors. These factors represent a set of abstract concepts, each conveying some idea common to a subset of the input collection. From Lingo's viewpoint, these concepts are perfect cluster label candidates; unfortunately, however, their matrix representation is difficult for humans to understand. To obtain concise and informative labels, we need to extract the verbal meaning behind the vector representation of the LSI-derived abstract concepts.

Among all term co-occurrences in a collection of documents, phrases that appear at least a certain number of times in the input collection seem to be most meaningful to users, and at the same time are relatively inexpensive to find. They're often collocations or proper names, making them both informative and concise. We therefore use them to approximate the verbal meaning of the SVD-derived abstract concepts. Lingo uses these frequent phrases as cluster labels.

Once we've discovered diverse cluster labels, we assign documents to them using standard VSM. Figure 1 gives an overview of the Lingo algorithm's phases, which we discuss in the following sections.

Preprocessing

At this stage, we typically use a combination of three common text-preprocessing methods:

- *stemming*, a technique for finding a semantic representation of an inflected word (usually a lemma) to decrease the impact of a language's syntax;
- *ignoring stop words*, a common technique for dealing with terms that occur frequently but have no meaning (conjunctions, articles, and so on); and
- *text-segmentation heuristics*, a technique for dividing text into words and sentences that has many implementations.

Logic suggests that applying these meth-

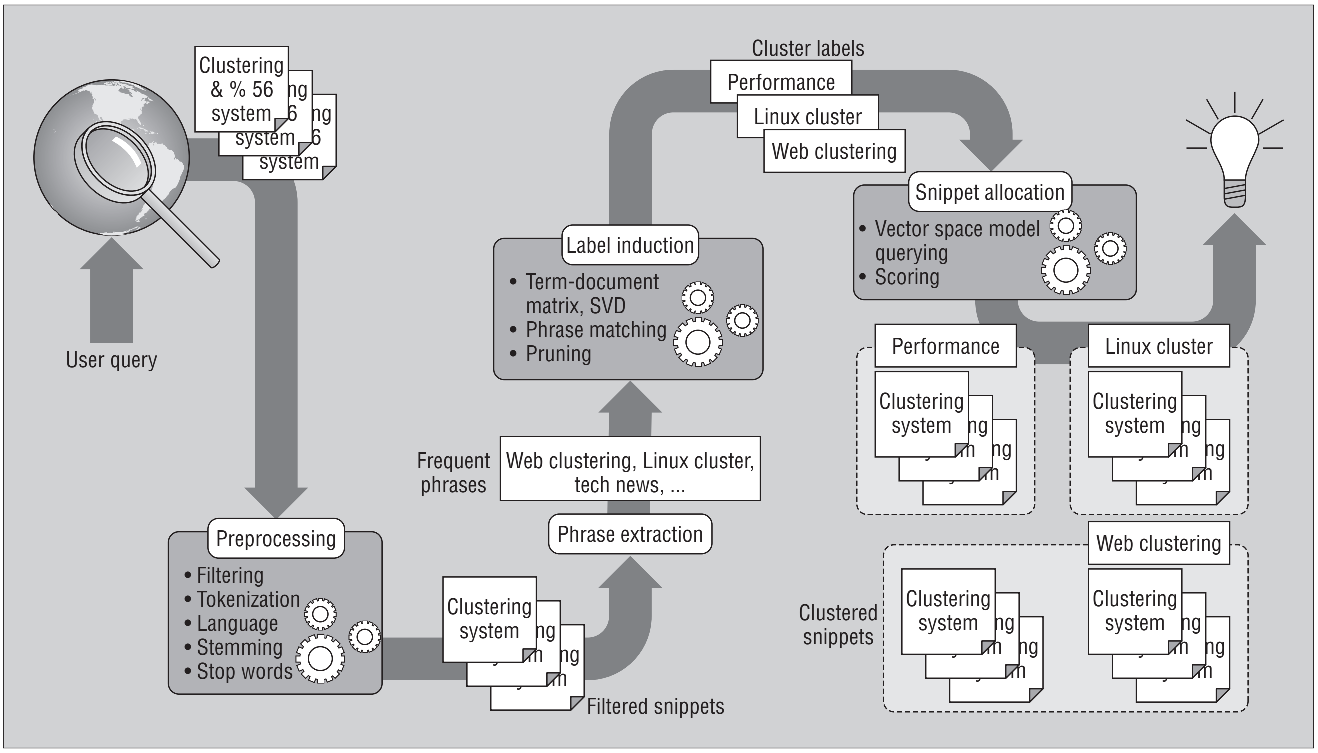


Figure 1. Overview of the Lingo algorithm's phases.

ods in information retrieval should improve results. In reality, some results contradict this assumption, especially when large quantities of text are available. However, our previous analysis of the influence of text-preprocessing methods[2] and experience with Lingo indicate that both stemming and stop-word identification are important for small and noisy textual information such as snippets.

Lingo recognizes each snippet's language separately and applies adequate preprocessing to it. Most algorithms remove stop words from the input entirely. Our algorithm only marks stop words, leaving them in the input because they can help users understand the meaning of longer phrases (for example, "Chamber Commerce" versus "Chamber of Commerce").

Phrase extraction

The phrase-extraction phase aims to discover phrases and single terms that could potentially explain the verbal meanings behind the SVD-derived abstract concepts. Like the online semantic hierarchical clustering (SHOC) algorithm,[3] Lingo requires that cluster labels

- appear in the input snippet at least a specified number of times.
- not cross sentence boundaries. Sentence markers indicate a topical shift, therefore a phrase extending beyond one sentence is unlikely to be meaningful.
- be a complete (that is, the longest possible) frequent phrase. Complete phrases should allow clearer cluster descriptions than partial phrases (for example, "Senator Hillary Rodham Clinton" versus "Hillary Rodham").
- neither begin nor end with a stop word. Again, we don't discard stop words appearing in the middle of a phrase.

Lingo uses a modified version of SHOC's phrase discovery algorithm. The algorithm uses a variant of suffix arrays extended with an auxiliary data structure—the longest common prefix (LCP) array—and identifies all frequent complete phrases in $O(N)$ time, where N is the total length of all input snippets.

Cluster-label induction

During the cluster-label-induction phase, Lingo identifies the abstract concepts that best describe the input snippet collection and uses frequent phrases to construct a human-readable representation of these concepts. This produces a set of labels, each of which will determine one cluster's content and description.

To illustrate the ideas introduced in this phase, we analyze how Lingo would deal with an example collection of $d = 7$ document titles (figure 2a), in which $t = 5$ terms (figure 2b) and $p = 2$ phrases (figure 2c) appear more than once and thus are treated as frequent.

First, Lingo builds a term-document matrix from the input snippets. In such a matrix, a column vector represents each snippet, and row vectors denote terms selected to represent documents' features. In our example (shown in figure 3), the first row represents the word "information," the second row represents "singular," and so on through the terms in figure 2b. Similarly, column one represents D1 in figure 2a, "large-scale singular value computations," column two represents D2, "software for the sparse singular value decomposition," and so on.

At this stage we disregard terms marked as stop words (such as "for" in figure 2a) and terms that haven't appeared more than a certain number of times (such as "software" in figure 2a). This not only significantly increases the algorithm's time efficiency but also reduces noise among cluster labels. We use the popular Salton's *tfidf* term-weighting scheme[4] to eliminate strong bias in the snippets toward the query words. We calculate values in matrix $\boldsymbol{A}$ (figure 3) using *tfidf* and then normalize each column vector's length.

We base the actual process of discovering abstract concepts on the SVD of the term-document matrix $\boldsymbol{A}$, which breaks it into three matrices ($\boldsymbol{U}$, $\boldsymbol{S}$, and $\boldsymbol{V}$) such that $\boldsymbol{A} = \boldsymbol{USV}^T$.

One of SVD's properties is that the first r columns of $\boldsymbol{U}$ (where r is $\boldsymbol{A}$'s rank) form an orthogonal basis for the input matrix's term space. In linear algebra, a linear space's basis vectors can serve as building blocks for creating vectors over that space. Following this intuition, in our setting each building block should carry one of the ideas referred to in the input collection. Thus, from Lingo's viewpoint, basis vectors (that is, the column vectors in $\boldsymbol{U}$) are exactly what it has set out to find: a vector representation of the snippets' abstract concepts.

In most practical situations, taking all r basis vectors as abstract concepts would result in an unmanageable number of clusters—usually close to the number of snippets. Therefore, Lingo uses the singular values of the $\boldsymbol{A}$ matrix (lying on the diagonal of the SVD's $\boldsymbol{S}$ matrix) to calculate how many columns of $\boldsymbol{U}$ should actually proceed to the next stage of the algorithm. Assume that the calculation results in $k = 2$ being set as the desired number of clusters for our example. Consequently, in further processing we use $\boldsymbol{U}_k$, which consists of the first k columns of $\boldsymbol{U}$ (shown in figure 4a).

Two characteristics of the basis vectors are important here. First, the vectors are pairwise orthogonal, which should result in a rich diversity among the discovered abstract concepts. Second, basis vectors are expressed in the $\boldsymbol{A}$ matrix's term space and thus can be the frequent phrases discovered in the previous phase. Lingo can therefore use the classic cosine distance measure to compare them and calculate which phrase or word will be an abstract concept's best verbal representation. We consider single terms at this stage because it's possible that none of the frequent phrases describes an abstract concept better than a single term.

For every frequent phrase and single frequent word, Lingo creates a column vector over the $\boldsymbol{A}$ matrix's term space. When assembled together, these vectors form a term-document matrix $\boldsymbol{P}$ of frequent phrases and words (figure 4b). In our example, column one corresponds to phrase "singular value," column two corresponds to "information retrieval," column three corresponds to "information" (single word), and so on.

Assuming column vectors of both $\boldsymbol{U}$ and $\boldsymbol{P}$ are length-normalized, as in our example, the problem of calculating the cosine distance between every abstract-concept–phrase-or-term pair reduces to simple matrix multiplication.

D1: Large-scale *singular value computations*
D2: Software for the sparse *singular value decomposition*
D3: Introduction to modern *information retrieval*
D4: Linear algebra for intelligent *information retrieval*
D5: Matrix *computations*
D6: *Singular value* cryptogram analysis
(a) D7: Automatic *information* organization

T1: Information
T2: Singular
T3: Value
T4: Computations
(b) T5: Retrieval

P1: Singular value
(c) P2: Information retrieval

Figure 2. Input data in the label induction phase: (a) documents, *d* = 7; (b) terms, *t* = 5; and (c) phrases, *p* = 2.

$$\boldsymbol{A} = \begin{bmatrix} 0.00 & 0.00 & 0.56 & 0.56 & 0.00 & 0.00 & 1.00 \\ 0.49 & 0.71 & 0.00 & 0.00 & 0.00 & 0.71 & 0.00 \\ 0.49 & 0.71 & 0.00 & 0.00 & 0.00 & 0.71 & 0.00 \\ 0.72 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.83 & 0.83 & 0.00 & 0.00 & 0.00 \end{bmatrix}$$

Figure 3. Term-document matrix in the label-induction phase.

(a)
$$\boldsymbol{U} = \left[\begin{array}{cc|ccc} 0.00 & 0.75 & 0.00 & -0.66 & 0.00 \\ 0.65 & 0.00 & -0.28 & 0.00 & -0.71 \\ 0.65 & 0.00 & -0.28 & 0.00 & 0.71 \\ 0.39 & 0.00 & 0.92 & 0.00 & 0.00 \\ 0.00 & 0.66 & 0.00 & 0.75 & 0.00 \end{array}\right]$$

(b)
$$\boldsymbol{P} = \begin{bmatrix} 0.00 & 0.56 & 1.00 & 0.00 & 0.00 & 0.00 & 0.00 \\ 0.71 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 & 0.00 \\ 0.71 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 & 0.00 \\ 0.00 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 & 0.00 \\ 0.00 & 0.83 & 0.00 & 0.00 & 0.00 & 0.00 & 1.00 \end{bmatrix}$$

(c)
$$\boldsymbol{M} = \boldsymbol{U}_k^T \boldsymbol{P} = \begin{bmatrix} 0.92 & 0.00 & 0.00 & 0.65 & 0.65 & 0.39 & 0.00 \\ 0.00 & 0.97 & 0.75 & 0.00 & 0.00 & 0.00 & 0.66 \end{bmatrix}$$

Figure 4. Matrices in the label-induction phase: (a) abstract concepts matrix *U*, (b) cluster-label-candidate matrix *P*, and (c) abstract-concept–cluster-label-candidate matrix *M*.

Rows of the resulting matrix $\boldsymbol{M}$ (figure 4c) represent abstract concepts, its columns represent phrases and single words, and individual values are the cosine similarities in question. Thus, in a single row, the maximum component indicates the phrase or single word that best approximates the corresponding abstract concept. In our simple example, the first abstract concept is related to "singular value," while the second is related to "information retrieval." As the values in the $\boldsymbol{M}$ matrix range from 0.0 (no relationship) to 1.0 (perfect match), Lingo also uses the maximum components as cluster label scores.

Cluster-content allocation

The cluster-content allocation process resembles VSM-based document retrieval except that instead of one query, Lingo matches the input snippets against a series of queries, each of which is a single cluster label. So, for a certain query label, if the sim-

Information retrieval (1.0)
D3: Introduction to modern *information retrieval*
D4: Linear algebra for intelligent *information retrieval*
(a) D7: Automatic *information* organization

Singular value (0.95)
D2: Software for the sparse *singular value* decomposition
D6: *Singular value* cryptograms analysis
(b) D1: Large-scale *singular value computations*

Other topics
(c) D5: Matrix *computations*

Figure 5. Cluster-content-allocation final results. We normalized group scores to the 0.0–1.0 range. Numbers in parentheses are cluster scores.

Table I. Experiment categories.

Category code	Number of documents	Contents
BRunner	77	Information about the *Blade Runner* movie
LRings	92	Information about the *Lord of the Rings* movie
Ortho	77	Orthopedic equipment and manufacturers
Infra	15	Infrared-photography references
DWare	27	Articles about data warehouses (integrator databases)
MySQL	42	MySQL database
XMLDB	15	Native XML databases
Postgr	38	PostgreSQL database
JavaTut	39	Java programming language tutorials and guides
Vi	37	Vi text editor

ilarity between a snippet and the label exceeds a predefined threshold (snippet assignment threshold), Lingo allocates the snippet to the corresponding cluster.

Snippet assignment threshold values fall within the 0.0–1.0 range. Higher threshold values result in more documents being put in clusters, which can decrease the assignment precision. Lowering the value leads to smaller groups with better assignment precision but smaller snippet recall. The snippet assignment threshold value is therefore largely a matter of user preference. We've empirically verified that thresholds within the 0.15–0.30 range produce the best results.

This assignment scheme naturally creates overlapping clusters and nicely handles cross-topic documents. We can also use the similarity values to sort snippets within their groups, making the most relevant snippets easier to identify. Finally, we created an "other topics" group for those snippets that don't match any of the cluster labels.

The last operation of the cluster-content allocation phase is calculating group scores as a product of the label score and the number of snippets in the group. Figure 5 shows the results for our example.

Evaluation

We can evaluate clustering quality through empirical evaluation, user surveys, or a merge-then-cluster approach, which compares a known cluster structure to the results of clustering the same set of documents algorithmically.

Yet, a cluster's quality or a label's descriptive power is subjective—even humans are inconsistent when asked to manually group similar documents. This doesn't mean human experts are making mistakes; rather, the inconsistencies reflect the fact that people perceive things differently. In this context, any measure of "quality" relative to some predefined clustering is skewed by the choice of an "ideal" solution (also called *ground truth*).

Test data and the experiment

We took our ground truth and test data from the Open Directory Project (http://dmoz.org), a human-collected directory of Web page links and descriptions. Documents and groups (clusters) inside ODP are a result of the commonsense agreement of many people and one individual's subjective choice. In addition, unlike most classic information retrieval test suites, which contain full documents, ODP contains only short descriptions of documents, which serve as snippet replacements.

We selected 10 categories (see table 1) related to four subjects: movies (2 categories), health care (1), photography (1), and computer science (6). Documents within each subject should theoretically have enough in common to be linked into a cluster. We also wanted to verify how Lingo would handle separation of similar but not identical topics within one subject, so we took some categories from a parent branch of ODP—for example, four categories were related to various database systems. Finally, as table 2 shows, we created seven test sets mixing categories in various ways so we could verify specific questions about Lingo. We clustered the test sets with snippet assignment threshold values between 0.150 and 0.250.

Empirical evaluation

We manually investigated each cluster's contents and label for every test set at the 0.250 threshold level. Table 3 presents the topmost labels. Cluster descriptions were generally satisfactory ("federated data warehouse" and "foot orthotics," for example), even if elliptical because of truncated sentences in the input snippets ("information on infrared [photography]").

In tests G1 to G3, Lingo highlighted all mixed categories as the topmost positions, indicating good topic-separation capabilities. Diversity test G3 showed that Lingo discovered and appropriately described the significantly smaller "Infra" category. The algorithm also properly divided closely related topics in test set G4 into clusters. In test G5, "Java" and "Vi" were the topmost clusters, followed by database-related clusters. In tests G6 and G7, Lingo successfully highlighted outlier subjects: In G6, clusters related to "Ortho" were at positions 3 ("foot orthotics") and 5 ("orthopedic products") of the result. In recent work, we show that suffix tree clustering, for example, has a tendency to obscure small distinct topics with subgroups of a major subject.[5]

Analytical evaluation

We can numerically compare similarity between two cluster structures in several ways—for example, using mutual-information measures.[6] But these measures usually attempt to aggregate similarity between individual clusters into a single figure, whereas we wanted to show the differences in allocation of objects to clusters between Lingo and the suffix tree clustering (STC) algorithm.

We used a *cluster contamination* measure—that is, a cluster is considered pure if it consists only of documents from a single original category (or its subset). Cluster contamination for cluster *K* is defined as the number of pairs of objects found in the same cluster *K* but not in any of the *partitions* (groups of documents in the ground truth set we're comparing against), divided by the maximum potential number of such pairs in *K*.

Cluster contamination of pure clusters equals 0. Cluster contamination of a cluster consisting of documents from more than one partition is between 0 and 1. An even mix of documents from several partitions is the worst case; such a cluster is said to be fully contaminated and has a measure equaling 1. Due to space limitations, we omit the mathematical formulae describing contamination measure here but present it online (www.cs.put.poznan.pl/dweiss/site/publications/download/ieee-cont-appx.pdf).

Figure 6 shows contamination measures for input data test set G7, clustered independently using Lingo and STC. As the figure shows, Lingo creates significantly purer clusters than STC, especially at the top of the clusters' ranking. In addition, STC can't distinguish between informative and uninformative clusters—see, for example, "includes" or "information" groups, essentially common words with no meaning specific to any cluster and hence with a high contamination ratio. Lingo also produces contaminated clusters, such as "Web site" or "movie review," but these can be understood ("Web site" is a common phrase in ODP) or explained (the "movie review" cluster is contaminated because it merges documents from two categories, "LRings" and "Brunner"). The latter example shows that blind analysis (including aggregate measures) can lead to incorrect conclusions; the "movie review" cluster is a generalization of two original categories and as such isn't bad, even though it was marked as contaminated.

We omit the remaining tests because the results and conclusions are similar (including the range of threshold settings for Lingo). Interestingly, adjusting STC's thresholds doesn't improve document allocation much; it merely affects the clusters' size.

The experiment revealed a minor drawback in Lingo's document-assignment phase. Recall that Lingo assigns documents to clusters using VSM after choosing the cluster label. Because VSM is phrase-unaware, it sometimes assigns incorrect documents to a cluster—for example, the "MySQL database" cluster includes documents from other database categories that contain the keyword "database." This misassignment problem is one of the issues we need to resolve in the future.

Table 2. Merged test sets.

Identifier	Merged categories	Test set rationale (hypothesis to verify)
G1	LRings, MySQL	Can Lingo separate two unrelated categories?
G2	LRings, MySQL, Ortho	Can Lingo separate three unrelated categories?
G3	LRings, MySQL, Ortho, Infra	Can Lingo separate four unrelated categories, one significantly smaller than the rest (Infra)?
G4	MySQL, XMLDB, DWare, Postgr	Can Lingo separate four similar, but not identical, categories (all related to databases)?
G5	MySQL, XMLDB, DWare,Postgr, JavaTut, Vi	Can Lingo separate four very similar categories (databases) and two distinct but loosely related ones (computer science)?
G6	MySQL, XMLDB, DWare, Postgr, Ortho	Can Lingo separate four dominating conceptually close categories (databases) and one outlier (Ortho) (outlier highlight test)?
G7	All categories	Can Lingo generalize categories (into movies and databases)?

Table 3. Labels of the test sets' topmost clusters.

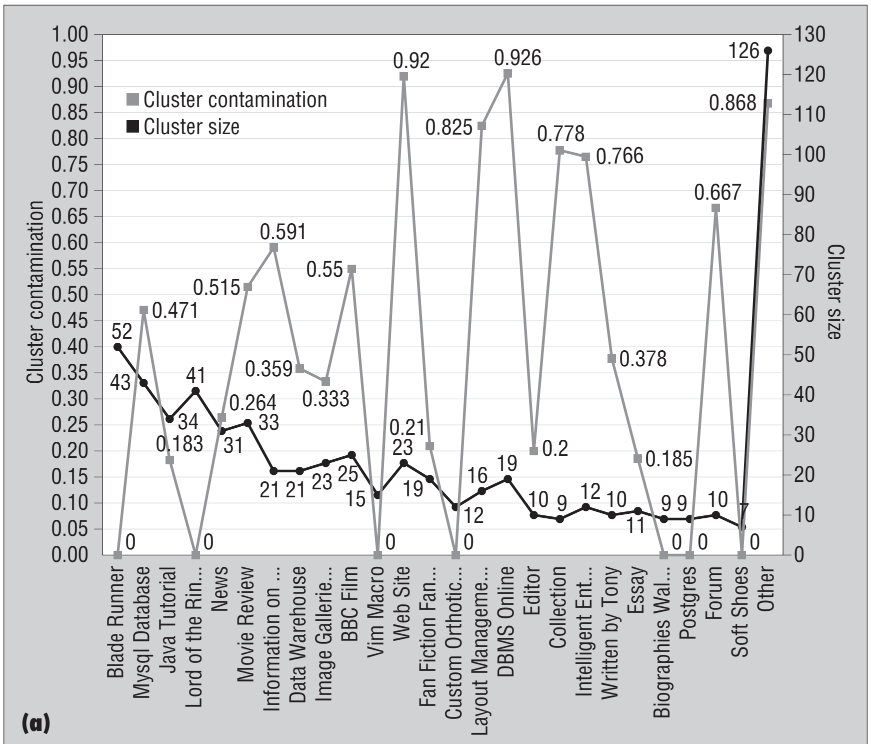
Test set identifier	Topmost clusters
G1	Fan fiction/fan art, image galleries, MySQL, wallpapers, LOTR humor, links
G2	News, MySQL database, image galleries, foot orthotics, *Lord of the Rings*, wallpapers, information on the films
G3	MySQL, news, information on infrared, images galleries, foot orthotics, *Lord of the Rings*, movie
G4	Federated data warehouse, XML database, Postgresql database, MySQL server, intelligent enterprise, magazine, Web based
G5	Java tutorial, Vim page, federated data warehouse, native XML database, Web, Postgresql database
G6	MySQL database, federated data warehouse, foot orthotics, orthopedic products, access Postgresql, Web
G7	*Blade Runner*, MySQL database, Java tutorial, *Lord of the Rings,* news, movie review, information on infrared, data warehouse

Lingo's time complexity is high and mostly bound by the cost of term-vector matrix decomposition; the best-known algorithms perform this operation in $O(m^2n + n^3)$ for an $m \times n$ matrix. Moreover, the high number of matrix transformations leads to demanding memory requirements. However, we designed Lingo for a specific application—search results clustering—and, in this setting, scalability to large data sets is of no practical importance (the information is more often limited than abundant). Obviously, improving the algorithm's efficiency and investigating possibilities for distributing the processing is an important and interesting direction of future research.

Is search-results clustering a panacea for all search-related problems? Certainly not. It's merely a small step forward to what we believe is an ideal search engine: one that can explain the query's context and hence stimulate interaction between the user and the system—something even the best ranking algorithms aren't good at.

Important enhancements to Lingo will include

- creating a hierarchical structure of clusters, either directly inferred from the source, or using man-made ontologies such as WordNet;
- improving the document-to-cluster assignment phase—the current method based on the VSM is the algorithm's weakest spot; and



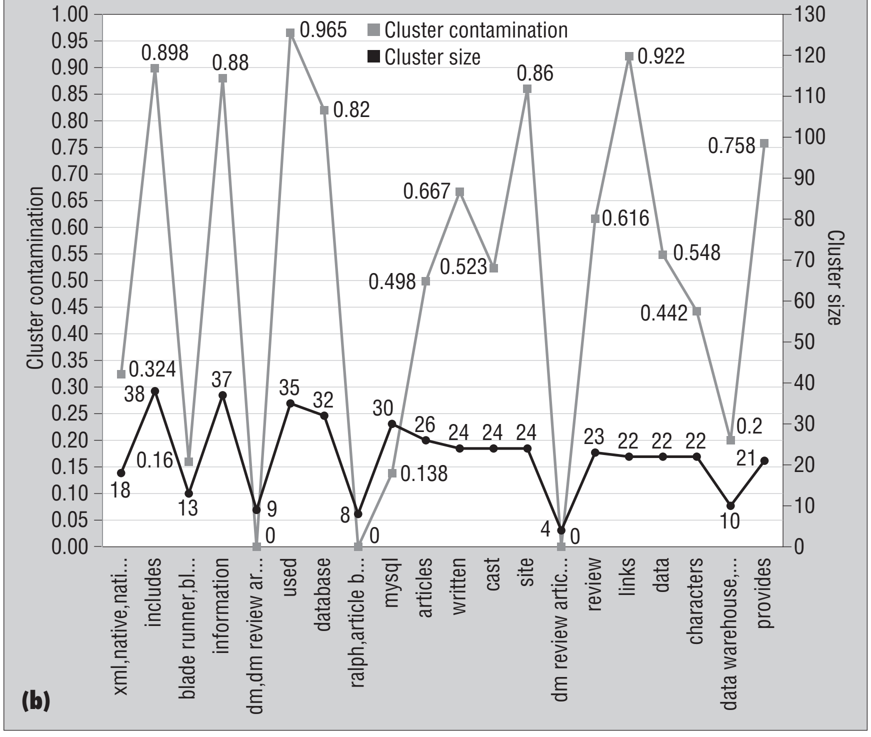


Figure 6. Contamination measure and cluster size for test set G7 clustered using (a) Lingo and (b) suffix tree clustering (STC). We've truncated cluster labels to fit the chart. The STC algorithm has no explicit "others" group. We ordered clusters on the horizontal axis from the most important clusters (left side) to the least significant ones (right).

- improving the phrase-selection method to prune elliptical or ambiguous phrases.

You can try Lingo in the publicly available online demo of the Carrot^2 system at http://carrot.cs.put.poznan.pl.

The Authors

Stanisław Osiński is a software developer with Poznań Supercomputing and Networking Center. His research interests include Web mining, text processing, and software engineering. He received his MS in computer science from Poznań University of Technology, Poland. Contact him at stanislaw.osinski@man.poznan.pl.

Dawid Weiss is a research associate at the Laboratory of Intelligent Decision Support Systems, Poznań University of Technology, Poland. His research interests include Web mining, text processing, computational linguistics, and software engineering. He received his MS in software engineering from Poznań University of Technology. Contact him at dawid.weiss@cs.put.poznan.pl; www.cs.put.poznan.pl/dweiss.

References

1. M.W. Berry, S.T. Dumais, and G.W. O'Brien, *Using Linear Algebra for Intelligent Information Retrieval,* tech. report UT-CS-94-270, Univ. of Tennessee, 1994.
2. J. Stefanowski and D. Weiss, "Carrot^2 and Language Properties in Web Search Results Clustering," *Proc. Web Intelligence, 1st Int'l Atlantic Web Intelligence Conf.* (AWIC 2003), LNCS 2663, E.M. Ruiz, J. Segovia, and P.S. Szczepaniak, eds., Springer, 2003.
3. Z. Dong, *Towards Web Information Clustering*, doctoral dissertation, Southeast Univ., Nanjing, China, 2002.
4. G. Salton and M. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, 1983.
5. S. Osiński and D. Weiss, "Conceptual Clustering Using Lingo Algorithm: Evaluation on Open Directory Project Data," *Proc. Int'l IIS: Intelligent Information Processing and Web Mining Conf.*, Springer, 2004, pp. 369–378.
6. B.E. Dom, *An Information-Theoretic External Cluster-Validity Measure*, IBM research report RJ 10219, IBM, 2001.