Pp. 54–65 in *Towards the Semantic Web and Web Services, Proceedings of the XML Finland 2002 Conference*, edited by E. Hyvönen and M. Klemettinen. HIIT Publications.

Building and Maintaining Web Taxonomies

Wray Buntine, Sami Perttu and Henry Tirri

Helsinki Inst. of Information Technology
HIIT, P.O. Box 9800
FIN-02015 HUT, Finland
{wray.buntine,sami.perttu,henry.tirri}@hiit.fi

Abstract. A recognized problem for internet commerce is the task of building a product taxonomy from web pages, without access to corporate databases, and then populating a database with link information about service, spare parts, reviews, product specifications, product family, etc. A key precursor for this task is the ability to build classification hierarchies in an unsupervised manner of web pages potentially useful. However, a nasty aspect of the real world is that most web pages have multiple facets. A web page might contain information about cameras and computers, as well as having both specification and sale data. We are interested in methods for unsupervised learning of multiple facet models which automatically allow these multiple facets to be extracted or predicted from examples. We will present some examples using the Reuters 2000 XML Corpus containing 800,000 Reuters newswires from the year beginning 20th August 1996. Our system, called Ydin, inputs XML/XHTML pages and displays results using a custom web interface. It extracts different facets and provides a navigation tool for investigating the results.

1 Introduction

A recognized problem for internet commerce is the task of building *a consumer guide*, building a product taxonomy from web pages, without access to corporate databases, and then populating a database with link information about service, repair, spare parts, reviews, product specifications, product family and company home pages, and purchase information from retailers. While within individual corporations this task is arguably well-handled by using the emerging standards and methodology of the semantic web, for web services such as consumer guides, where collections of disparate information sources would be used, top-down standards enforcement is not a realistic option, and nor will be labor intensive knowledge engineering in general. Many believe more automatic methods are required to build the ontologies and other knowledge structures needed [1], such as semantic-based search and the automated construction of taxonomies from web data.

While we do not believe full automation of the task (building consumer guides) is generally feasible, significant support for the task is provided by the ability to build classification hierarchies, that is, taxonomies, in a supervised or

unsupervised manner. However, a nasty aspect of the real world is that most webpages have multiple facets. A web page might contain information about cameras and computers, as well as having both specification and sale data. Whereas another page might mix a product index with partial specification and sales data. We are interested in methods for supervised and unsupervised learning of multi-faceted models, where facets cover both the content and the style (e.g., index, sales, specs, etc.).

Clustering or unsupervised learning is a standard method for analysing discrete data such as documents, and is now being used in industry to create taxonomies from web pages. A rich variety of methods exist borrowing theory and algorithms from a broad spectrum of computer science: spectral (eigenvector) methods, kd-trees [3], using existing high-performance graph partitioning algorithms from CAD [4], hierarchical algorithms [5] and data merging algorithms [6], etc. These methods are used both to structure a corporation's document database, and to organise results from a web search.

These methods have one significant drawback for typical application in areas such as document analysis: each document is to be classified exclusively to one class. Their models make no allowance for instance, for a product page to have 40% digital camera content and 30% laptop computer content, and 10% of strictly sales jargon. Rather these methods require pages to be 100% one way or another, and any uncertainty is only about whether to place the 100% into one or the other class. In practice documents invariable mix a few topics, readily seen by inspection of the human-classified Reuters newswire, so the automated construction of topic hierarchies needs to reflect this. One alternative is to make clustering *multi-faceted* whereby a document can be assigned proportionally (i.e., using a convex combination) across a number of clusters rather than uniquely to one cluster.

A different task, supervised multi-faceted learning, covers the same kind of problem: a document can belong to several classes simultaneously, though perhaps with varying degrees. However, in this case, the taxonomy (or classification hierarchy) is pre-supplied. Given a set of pre-assigned classes for each document, one is to build a predictive classifier that will assign a set of classes to new documents. This supervised multi-faceted task has been addressed quite well in the literature, oftentimes by modifying an existing classification algorithm, for instance the multi-class multi-label perceptron of Crammer and Singer [7]. In our internet commerce example, the way to use these techniques is to have an expert build a taxonomy, assign say 20 pages to each node, train a classification system on these examples, and then classify the remaining pages under consideration. However, because this requires a pre-existing taxonomy, it does not address the general discovery task we proposed.

Several authors have recently developed methods that address the task of multi-faceted clustering. These are discrete analogues to principle components analysis (PCA) intended to handle discrete or positive only count data of the kind used in the bag-of-words representation of web pages. By their discrete relatively efficient nature, they offer significant computational advantages over the more

general non-linear methods of Karhunen and others [8]. These new methods include probabilistic latent semantic analysis [9] latent Dirichlet allocation [10], multinomial PCA [11]. A good discussion of the motivation for these techniques can be found in [9].

In this paper we present an application of multinomial PCA to the new Reuters Corpus[1], containing over 806,791 news items from 1996 and 1997. This gave us the opportunity to evaluate the effectiveness and scaling potential of the algorithm for the task of automatically constructing taxonomies from large collections of web pages.

2 Contrasting Clustering and Multi-Faceted Clustering

For concreteness, consider the problem in terms of the usual "bag of words" representation for a document [12], popular for analysing web pages. Here the items making up the sample are documents and the features are the counts of words in the document, and counts of any stylistic features considered relevant (though ignored here). A document is represented as a sparse vector of words and their occurrence counts. All positional information is lost. With J different words/features, the dimensionality for words/features, each document becomes a vector $\boldsymbol{x} \in \mathcal{Z}^J$, where the total $\sum_j x_j$ might be known. Traditional clustering becomes the problem of forming a mapping into a single integer (i.e., a class assignment or discretization, $\mathcal{Z}^J \mapsto \{1, \ldots, K\}$, where K is the number of clusters). Whereas techniques such as PCA form a mapping into a real-valued vector of lower dimension (i.e., $\mathcal{Z}^J \mapsto \mathcal{R}^K$ where K is considerably less than J).

The problem we consider, however, is to represent the document as a convex combination, thus to form a mapping into a lower dimension probability vector (i.e., $\mathcal{Z}^I \mapsto \mathcal{C}^K$ where $\mathcal{C}^K$ denotes the subspace of $\mathcal{R}^K$ where every entry is non-negative and the entries sum to 1, $\boldsymbol{m} \in \mathcal{C}^K$ implies $0 \leq m_k \leq 1$ and $\sum_k m_k = 1$).

For instance, suppose we are performing a coarse clustering of newswires into topics: the topics found might be "sports", "business", "travel", "international", "politics", "domestic", and "cultural". Consider a document about a major sports-star and the overlap of his honeymoon with a big game. Then traditional clustering might output the following: "the document is about sports". A more refined clustering system that represents uncertainties as well might output: "with 90% probability the entire document is about sports, with 7% probability it is about cultural, and 3% probability about something else". General multinomial PCA considered in this paper might output: "50% of the document is about sports, 35% of the document is about cultural, 7% about business, 5% about international". The supposed business content is really a discussion of the hotel for the honeymoon and the supposed international content comes from the location of the honeymoon. Note here general multinomial PCA plays the role of dimensionality reduction, and places similar kinds of words into the same bucket for compression purposes rather than any real topic identification.

[1] Volume 1: English Language, 1996-08-20 to 1997-08-19.

The problem we consider is also to perform multi-faceted clustering, which serves the purpose of extracting multiple mutually occurring topics from a document. Suppose $\boldsymbol{m}$ is the probability vector ($\boldsymbol{m} \in \mathcal{C}^K$) corresponding to a particular document. For multi-faceted clustering, $\boldsymbol{m}$ should have most entries zero, and only a few entries significantly depart from zero. A measure we shall use for this is entropy, $H(\boldsymbol{m}) = \sum_j m_j \log(1/m_j)$. Thus multi-faceted clustering prefers low entropy vectors from $\mathcal{C}^K$, i.e., those where the entropy is much less than $\log K$. In the limit, when the average entropy of the probability vectors is 0, the mapping becomes equivalent to standard clustering. With the simple honeymoon example above, the output could be reduced to: "70% of the document is about sports, 30% of the document is about cultural". This makes the document have $2^{H(\boldsymbol{m})} = 1.85$ effective topics, as opposed to the original PCA example above with more proportions (0.5,0.35,0.07,0.05) which had $2^{H(\boldsymbol{m})} = 3.17$ topics.

3 Overview of the Multinomial PCA Method

We give a brief review of the multinomial PCA method and algorithm here as a basis for the experimental work presented later.

3.1 The Basic Model

To begin, consider Tipping *et al.*'s representation of standard PCA [13]. The hidden variable $\boldsymbol{m}$ is sampled from K-dimensional Gaussian noise. Each entry represents the strength of the corresponding component and can be positive or negative. This is folded with the $J \times K$ matrix of component means $\boldsymbol{\Omega}$ and then used as a mean for J-dimensional Gaussian noise.

$$\begin{aligned} \boldsymbol{m} &\sim \mathit{Gaussian}(0, \mathbf{I_K}) \\ \boldsymbol{x} &\sim \mathit{Gaussian}(\boldsymbol{\Omega m} + \boldsymbol{\mu}, \mathbf{I_J}\sigma) \end{aligned}$$

This relies on the data being real-valued and somewhat Gaussian, which fails badly for some image and document data. Techniques such as ICA address this problem using a more general non-linear framework [8].

A discrete analogue to the above formulation is first to draw a probability vector $\boldsymbol{m}$ that represents the proportional weighting of components, and then to mix it with a matrix $\boldsymbol{\Omega}$ whose columns represent a word probability vector for a component. This yields a distribution over the word counts $\boldsymbol{x}$:

$$\begin{aligned} \boldsymbol{m} &\sim \mathit{Dirichlet}(\boldsymbol{\alpha}) \\ \boldsymbol{x} &\sim \mathit{Multinomial}(\boldsymbol{\Omega m}, L) \end{aligned}$$

Here L is the total number of words in the document, and $\boldsymbol{\alpha}$ is a vector of K-dimensional parameters to the Dirichlet. Thus, the mean of each entry x_j is a convex combination of a row of $\boldsymbol{\Omega}$.

3.2 The Basic Algorithm

Basic algorithms for this problem as casted use the hidden variable framework of the Expectation-Maximization (EM) algorithm and its variants widely used in clustering (e.g., [14]). This yields an algorithm that estimates the model parameters $\boldsymbol{\Omega}$, the distribution of words per component, from the document mixing proportions $\boldsymbol{m}$, then in turn estimates the mixing proportions $\boldsymbol{m}$ from the model parameters $\boldsymbol{\Omega}$. This iterative approach is termed a re-estimation algorithm. The version we use applies a so-called variational extension of the EM algorithm [11], and was first applied to this problem by Blei, Ng, and Jordan [10]. However, they ran their algorithm on the old Reuters Corpus maintained by David Lewis that contains about 20,000 documents, using a network of computers in parallel, and only built 20 component models. It is difficult to assess the properties of their results from the simple experiments they reported.

3.3 Scaling Up the Algorithm

If 1000 component models are to be built, that would mean $806,791,000$ floats are needed to store the estimate of $\boldsymbol{m}$, denoted here as $\widehat{\boldsymbol{m}}$, which would require 3.2Gb if stored naively.

Thus, in order to run the system on the full Reuters data set, we needed to make some changes to our software. First, we store each entry of $\widehat{\boldsymbol{m}}$ in 16-bits as a factor of 2^{-16}, which reduces the storage requirement to 1.6Gb. Note that part way through the run, this becomes sparse, but initially at least the full $1.6Gb$ is used. Second, we process the $\widehat{\boldsymbol{m}}$ data using sparse vector processing. Part way through the run, $\widehat{\boldsymbol{m}}$ becomes sparse and a single cycle runs by up to three times faster. Third, we process the $\widehat{\boldsymbol{m}}$ data directly off the disk using memory mapping (`mmap()` in Linux) and do not keep the full matrix in memory. It turns out that the processing per document is sufficiently slow that this use of disk makes no difference in speed. The $\boldsymbol{\Omega}$ matrix, of size 260Mb (with 1000 components and $65,000$ words), needs to be stored in main memory since access to it is effectively random.

Space requirements for runtime are $O(K*(I+J)+S)$ where S is the size of the input data represented as sparse vectors, and each iteration takes $O(K*(I+J+S))$. Thus the computational requirements are comparable to an algorithm for extracting the top K eigenvectors usually used for PCA. Convergence is maybe 10-30 iterations, depending on the accuracy required, slower than its PCA counterpart. Our code is written in C using Open Source tools and libraries such as the GNU Scientific Library (GSL), and we intend to release an Open Source version once development is sufficiently advanced.

4 Reuters Experiments

We collected bag-of-words data from the new Reuters Corpus, containing $806,791$ news items from 1996 and 1997. On average, each news item is 225 words long,

which translates to a total of 180 million word instances. About 390,000 of these are distinct words; we kept the most frequent 65,000 in the dictionary, accounting for 99.995% of the data. This lets us store the full set of documents in a bag-of-words format in 220 MB which can be kept for random access in a computer's main memory. The individual documents themselves, when individually compressed, take up about 3 GB on disk. We were able to process the full 806,791 Reuters news documents on a single desktop Linux computer with 1 GB of memory and a 1.3 GHz processor in an overnight run. Parallel processing (e.g., [10]) is the next technique which we will need when running the algorithms on 20 GB web/HTML repositories. Runs were done training on 700,000 documents and using the remainder as a hold-out set to ensure safe stopping and prevent over-fitting.

4.1 Inspecting Results

With data sizes of this magnitude, it is no longer feasible to analyze the results from a static printout. For instance, a cross-referenced report listing details of components, words and documents and their statistics can easily reach a few megabytes for even 10,000 documents. Hence, we have written a custom web server in C++ that allows the user to examine different aspects of a model via a web browser. The model data takes many gigabytes of space with the larger models; only a small part of it can be kept in memory at once.

Documents, components and words each have their own pages that are dynamically generated and fully cross-referenced. Fig. 1 shows a part of a (origi-

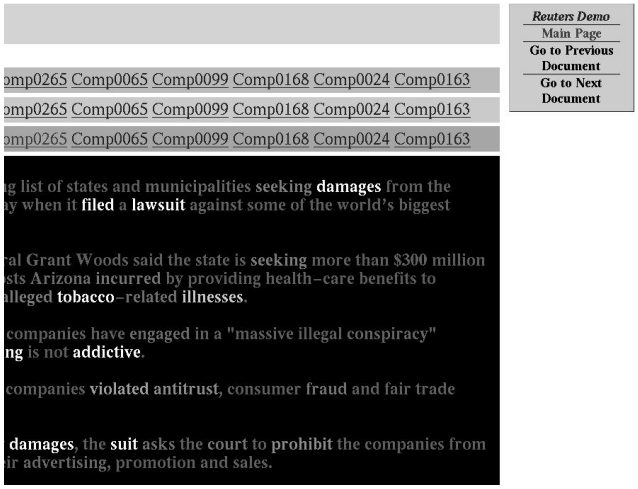

Fig. 1. Color-Coded Document Display

nally) color-coded display of a document. In our interface, the three color components red, green and blue can be assigned individually to different topics; in the figure, they are all assigned to a single topic to get a gray-level display. Each word is colored according to its frequency in the component divided by its frequency in the data. The highlighted component seems to be about lawsuits against tobacco companies.

Naturally, components in the model reflect the Reuters Corpus content: more than a half of the components are related to subareas of finance and economy. In the 1000 component run, many components correspond to breaking out these reports into topics like "Australian stock market closing prices", "USA Research alerts," "European union shipping reports," "Corporate profits," "Nuclear power companies," etc.

Table 1 summarises the components for a $K = 20$ component run, ordered with the most numerous components at the top. Listed in the table, the effective

Code	Description	Prop.	Eff. words	Eff. docs. (1000s)
01	investigations/interviews/legal	0.08	310	347
10	corporate finance	0.07	307	279
07	stock exchange	0.07	275	226
11	international finance/currency	0.07	343	212
17	international/government relations	0.06	525	196
06	corporate research/intelligence	0.08	700	191
18	EU politics, economy	0.04	528	155
19	business/markets in greater europe	0.03	405	145
12	business in asia/sth africa	0.03	492	128
05	politics	0.03	940	123
13	corporate results/forecasts	0.09	199	122
00	production/capacity	0.03	552	120
15	bonds	0.06	587	119
16	commodities	0.06	584	113
09	terrorism	0.04	656	108
14	EU records, imports/exports	0.02	617	105
03	sports, conflicts	0.03	702	104
08	soccer and cricket	0.02	692	71
04	shipping reports, tennis/golf	0.01	725	57
02	indicators/ecomony	0.01	112	38

Table 1. Component Summary for K=20

number of words in a component is the log2 entropy of the component word vector (entry in $\boldsymbol{\Omega}$) raised to the power 2. The effective number of documents containing a component is the log2 entropy of documents given components (probability of being in a document given a word from the component) raised to the power 2.

For this run with components, each document might have between 2 and 8 components contributing to its word distribution, many with one dominant one. The top component (with code 01) reflects this because while it occurs in effectively 43% of documents, it only makes up 8% of the component contributions, thus when it occurs, it occurs on average as 20% of a document. Note the documents with a high proportion of the component with code 01 are all legal cases, whereas in general the component might occur in articles including an interview of a CEO or politician.

The following components are taken from a run for $K = 300$ components. Table 2 shows two components from the model. Typical words are ordered ac-

Component 37		Component 79	
Typical Words	Unexpected Words	Typical Words	Unexpected Words
0.029 church	9.53 resplendent	0.053 internet	8.89 Vebacom
0.027 mayor	9.51 Tuckey	0.047 telecommunicat.	8.89 Viinanen
0.017 hundreds	9.51 Perafan	0.038 telecom	8.88 Dancall
0.017 pope	9.50 cobras	0.032 phone	8.88 xylitol
0.016 catholic	9.50 botulism	0.032 access	8.88 Sudirman
0.016 children	9.50 Balabagan	0.030 communications	8.88 Rhenus
0.015 people	9.50 Arreckx	0.027 telephone	8.88 Hotwired
0.014 who	9.49 Burman	0.025 service	8.88 Sunpage
0.013 roman	9.49 manifestations	0.022 mobile	8.87 Scaglia

Table 2. Top Words In Components 37 and 79

cording to their frequency in the component. Unexpected words are scored by the $\log_2$-difference of their frequency from word prior. Component 37 is broadly religion-related, with a large expected number of words, 580. The typical documents contain a news item about the war on flies declared by the mayor of Manila and an item about the Pope blessing Christmas crib figurines.

Component 153 in table 3 is interesting in that its typical words are per-

Component 153		Component 48	
Typical Words	Unexpected Words	Typical Words	Unexpected Words
0.193 we	6.62 Ospel	0.880 loss	10.47 Penril
0.043 very	6.62 Maucher	0.015 widens	10.47 Reddi
0.042 our	6.62 Jagmetti	0.012 write	10.47 Rayrock
0.027 do	6.62 Zinkernagel	0.011 narrows	10.47 Puretec
0.023 are	6.62 Kulczyk	0.009 charges	10.47 Yuoka
0.023 good	6.61 Peltola	0.005 restructuring	10.47 jpe
0.021 can	6.61 Integrion	0.003 writedown	10.47 Dorman
0.020 is	6.61 Bols	0.003 discontinued	10.46 Schmiedeknecht
0.020 going	6.60 Cees	0.003 disposal	10.46 Digene

Table 3. Top Words In Component 153 and 48

sonal pronouns and opinionated words. The typical documents are invariably interviews or press releases.

Component 48 in table 3 is an example of a low-entropy component with an expected number of words of only 2.36. It is dominated by the word "loss", which has a frequency of 0.88. The typical news items are financial news about losses. This component is basically triggered by the occurrence of the word "loss", making all the other high frequency words in the component more likely.

4.2 Explaining Component Dimensionality

To understand components and their relationship to the task of multi-faceted clustering, we plotted some diagnostics extracted from the result matrices $\boldsymbol{\Omega}$ and $\boldsymbol{m}$. The effective number of components in a document is the log2 entropy of its component proportion vector (entry in $\boldsymbol{m}$) for that document raised to the power 2. For instance, a document where the $\boldsymbol{m}$ vector is $(0.33, 0.33, 0.33, 0, 0, 0, \ldots)$ would have an effective number of components of 3, and a document where the $\boldsymbol{m}$ vector is $(1/K, 1/K, \ldots, 1/K, 0, 0, 0, \ldots)$ would have an effective number of components of K. Fig. 2 shows the relationship between document size and

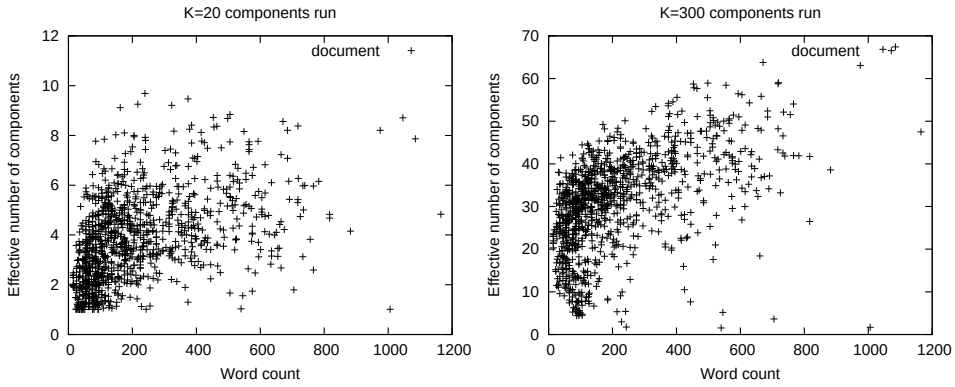

Fig. 2. Document Size versus Components. (a) 20 components, (b) 300 components

number of components for the runs using $K = 20$ and $K = 300$ components. Clearly, in this case, the number of components for each document indicates we are operating in the regime of dimensionality reduction. Most components are not synonym sets, they are topical sets of words. While for each component there will be a corresponding set of documents where this dominates, the majority of documents mix many different components.

Fig. 3 shows the effective number of words for different components runs. For instance, for $K = 20$ this plots data from the fourth column of Table 1. Components have been ordered in terms of their effective number of words, and

then the effective number of words plotted (which is now increasing). For the $K = 300$ run for instance, some components are very specific, the extreme being component 48 above, and some components are far more topical with a large number of effective words. Note, however, that the components for the different runs are very different from each other and amenable to hierarchical combination (unlike the components obtained using PCA).

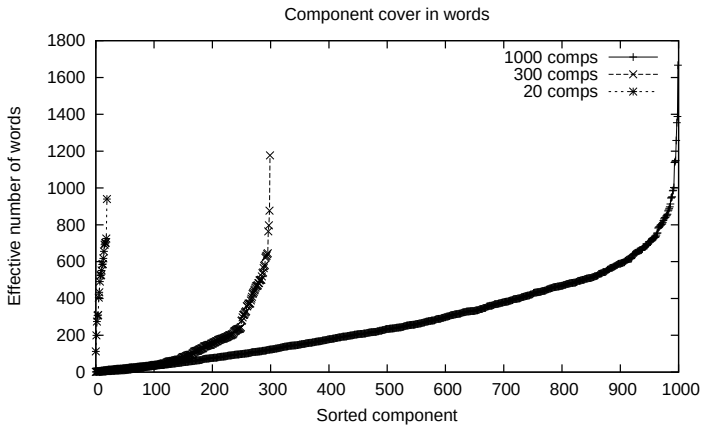

Fig. 3. Effective words per component

5 Overview of the Ydin Server

The six modules comprising the Ydin server, some of which are described above, are shown in Fig. 4:

XML/XHTML parser: Reads and parses XML/XHTML documents, creating a tree model of each document. XML elements are leaves in the tree.

Document analyzer: Converts document content to words and sentences. Words are initially stemmed, i.e. reduced to a root word, and placed in a dictionary and a word index.

Document database: Distributed storage for processed and compressed documents and information about them: dictionary, inverted index, bag-of-words data.

MPCA: Statistical component described above to generate component models ($\boldsymbol{\Omega}$) and document-component probability vectors ($\boldsymbol{m}$).

HTML interface: Interfaces with the MPCA and document database modules to provide the user with a high-level view of the documents. The interface has a number of optional functions:

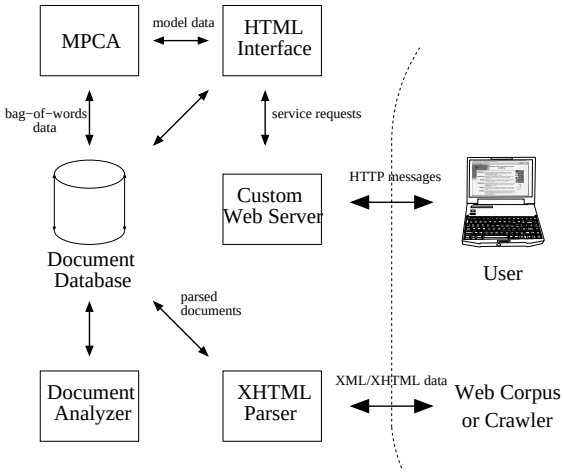

Fig. 4. Module diagram of the Ydin Server

search-by-topic: enter topical words to do a topic search
find-similar: find documents similar to the current document
component viewer: view a component in terms of common words and typical documents
word viewer: view a word in terms of its semantically related words

Custom web server: Our server responds to HTTP requests and dispatches them to the HTML interface, which then generates the desired pages dynamically. We use a custom server for maximum efficiency but a standard web server such as Apache could be used in its place.

We are extending Ydin to the task of extracting web taxonomies, which goes beyond component analysis.

For instance, to be able to provide links to reviews of a product the user is interested in, we need to be able to identify which documents contain product reviews. This entails matching the components generated by a statistical model with the semantics we are interested in. In other words, the statistical components need to be identified; we could use hand-made or supervised criteria for inferring such information. Other services such as comparisons of product features require similar functionality.

A move beyond simple bag-of-words analysis to richer context and sentential models may be mandatory to reach this level of sophistication.

6 Conclusion

We have demonstrated that recent extensions to PCA for multinomial data provide some support for multi-faceted clustering. We argued that while multinomial

PCA is really a dimensionality reduction algorithm, and not designed for multi-faceted clustering, if used right components can be extracted that are useful for the purpose of building a taxonomy. For this, we would need at least the ability to generate full topic hierarchies, and to perform automatic labelling/naming of topics in the hierarchy. We expect that by doing this for multiple companies at once, useful hierarchies could be established.

We have also presented our Ydin server which has been developed for the purpose of exploring the results of a multinomial PCA analysis on a large repository of HTML/XML data. We are currently extending the tool with view to building an Open Source system for information navigation and assisting the maintenance of semantic web.

Acknowledgements Thanks to Reuters for providing their Corpus as described in Footnote 1.

References

1. Cherry, S.: Weaving a web of ideas. IEEE Spectrum **39** (2002) 65–69
2. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000) 888–905
3. Moore, A.: Very fast EM-based mixture model clustering using multiresolution kd-tree. In: Neural Information Processing Systems, Denver (1998)
4. Han, E.H., Karypis, G., Kumar, V., Mobasher, B.: Clustering based on association rule hypergraphs. In: SIGMOD'97 Workshop on Research Issues on Data Mining and Knowledge. (1997)
5. Vaithyanathan, S., Dom, B.: Model-based hierarchical clustering. In: UAI-2000, Stanford (2000)
6. Bradley, P., Fayyad, U., Reina, C.: Scaling clustering algorithms to large databases. In: Proc. KDD'98. (1998)
7. Crammer, K., Singer, Y.: A new family of online algorithms for category ranking. In: 25th Annual Intl. ACM SIGIR Conference. (2002)
8. Karhunen, J., Pajunen, P., Oja, E.: The nonlinear PCA criterion in blind source separation: Relations with other approaches. Neurocomputing **22** (1998) 520
9. Hofmann, T.: Probabilistic latent semantic indexing. In: Research and Development in Information Retrieval. (1999) 50–57
10. Blei, D., Ng, A., Jordan, M.: Latent Dirichlet allocation. In: NIPS*14. (2002) to appear.
11. Buntine, W.: Variational extensions to EM and multinomial PCA. In: ECML 2002. (2002)
12. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison Wesley (1999)
13. Tipping, M., Bishop, C.: Mixtures of probabilistic principal component analysers. Neural Computation **11** (1999) 443–482
14. Kontkanen, P., Myllymaki, P., Silander, T., Tirri, H.: BAYDA: Software for bayesian classification and feature selection. In: Knowledge Discovery and Data Mining. (1998) 254–258