Visualizing Content Based Relations in Texts

Edgar Weippl
Software Competence Center Hagenberg
Hauptstr. 99, A-4232 Hagenberg, Austria
E-Mail: edgar.weippl@scch.at

Abstract

Our goal is to efficiently visualize a medium sized hypertext database containing 500 - 20000 articles. The visualization technique we propose is an Information Landscape. Basically, the information landscape maps texts into a 2D plane so that related texts are placed next to each other. The hypertexts' location is calculated according to their content and not according to their links. Combining already published algorithms the clustering works very well. An important issue, however, is a well-designed user interface (UI). In this paper we present two 2D interfaces and an improved 3D version. This paper covers all aspects from preprocessing and clustering to the final UI and its functionality.

Keywords

Text clustering, stemming/morphological analysis, information visualization for IR, UIs/visualization for hypertext search and navigation, text categorization

1. Introduction

Our goal is to efficiently visualize a medium sized text database containing 500 - 20.000 articles. The information landscape - a visualization technique we propose in this paper - is useful for many different tasks like organizing one's literature database or for gaining an overview over text collections like conference proceedings and for hypertext-based textbooks in computer-based training.

Basically, the information landscape maps texts into a 2D plane so that related texts are placed next to each other. The hypertexts' location is calculated according to their content and not according to their links. This is useful because (1) authors tend to forget to set links and (2) text collections that do not contain links at all can be visualized, too.

Combining already published algorithms the clustering works very well. An import issue, however, is a well-designed user interface (UI). In this paper we summarize two 2D interface options we implemented and tested. A simple UI is ideal for novice users and a more complex one gives content creators and expert users a much more powerful tool to keep an overview over a text collection. Each text is visualized as a dot in the 2D map (see Figure°1). The results of full-text searches and other operations can be marked using color-coding and glyphs.

As the graphic cards in standard PCs became much more powerful during the last year, we continued the development of a 3D version that we had already canceled due to performance reasons. The third dimension can be used to visualize whether a text contains overview information or detailed data. Each mountain in the information landscape represents one text. Big mountains contain overview information, whereas small mountains symbolize detailed texts.

This paper covers all aspects needed to build such a 3D information landscape. Section 2 explains the best practices to preprocess texts so that the clustering works well. The last part of Section 2 summarizes other people's work that visualizes text relations, too. Section 3 describes our UI in detail and section 4 explains the technical details on how the texts have to be processed.

2. Related Work

This work encompasses different aspects of computer science. First of all, the texts have to be preprocessed so that the clustering algorithms can perform well. The last

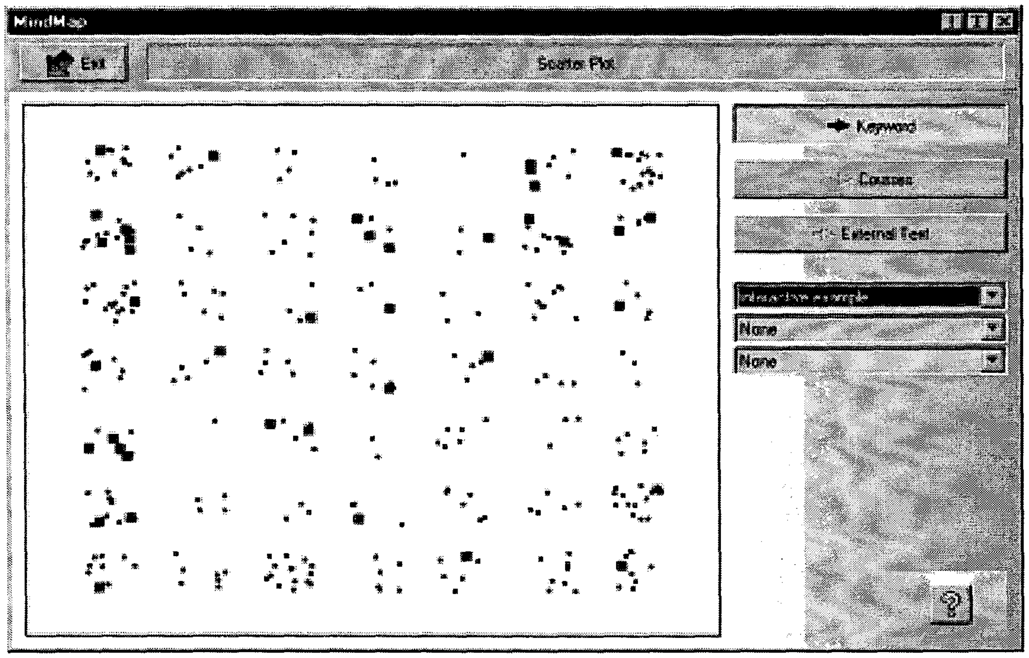


Figure 1: The information landscape offers an easy way of exploring how concepts are related and supports students in quickly gaining an overview over the whole content. In this Figure all texts that contain the term "Interactive Example" are displayed as boxes.

step is visualizing the results so that users can reach their goal quickly. Many researchers have dealt with all these topics. By improving and combining existing technologies we were able to implement the Information Landscape described in this paper.

2.1. Preprocessing of Texts

The idea to describe texts using term vectors was mentioned by Salton [26], [27], [4] and is clearly summarized by Baezo-Yates ([2] page 27). Wise [30] also used this vector-based model to calculate his document clustering.

Another very important issue in preprocessing is stemming. Stemming is used to reduce the number of distinct words and helps to improve both the quality and the speed of the clustering.

The first stemming algorithm was proposed by Lovins [18]. Porter [21], [2] developed a very popular stemming algorithm (see also appendix in [2]) that has several drawbacks.

§ Terms are incorrectly stemmed, e.g. "police" and "policy" are both stemmed to "polic". This is called overstemming [20].

§ Stems cannot always be used for classification. E.g. the term "bank" can mean a bank where people can sit on or a financial institution.

Using Xu et al [31] ideas, the two categories of mistakes concerning stemming and classification can be solved. By "training" the stemmer for a special dataset, false classifications can be minimized.

Krovetz [15] developed yet another stemmer based on machine-readable dictionaries and morphological rules. Despite addressing problems of the Porter stemmer, KSTEM - as it is now called - does not produce consistently better recall/precision performance.

As the importance of words depends upon how often they occur in different texts, the vectors should be modified using techniques proposed by Beazo-Yates ([2], page 29).

2.2. Clustering of Texts

Clustering is a very common task and it is beyond this paper's scope to mention all relevant work. Instead, we focus on the basic techniques we used.

A rather well known project is Websom [10]. It is based on Kohonen's ideas first published as book in 1989 [13]. This algorithm was used to explore databases [14]. A summary on research results since 1989 can be found in [12].

Other approaches to classify texts are the Principal Component Analysis, Error backpropagation networks [32] and genetic algorithms. Weippl [29] gives an overview over all these options and evaluates them.

2.3. User Interfaces for Text Collections

The Scatter/Gather system [8] is an approach that is very effective for browsing search results. Every text can be found in a cluster but the paper does not describe any form of visualizing the inter-cluster relationship.

The Bead system [6] uses multidimensional scaling (MDS) to calculate a 2D visualization. However, the main drawback of the MDS based algorithm is that clusters may lie very far apart and the density within a cluster may be very high. Due to this fact the resulting data clouds are hard to visualize. We described [29] a similar situation for a PCA based approach.

The Galaxy of News [24] uses an Associative Relation Network to classify news texts based upon assumptions of which words (keywords, location, time, etc.) are important. This implies that the information in the text body is ignored. For news articles this may not matter but for a general solution this is definitely a drawback.

Andrews [1] developed Information Landscapes to visualize a hyperlink structure. Our information landscape is different because it uses both the information contained in the texts and the hyperlink structure to train a neural net. The advantage is that if an author forgets to set some links, the information landscape will still represent the knowledge domain fairly well.

Robertson's Data Mountain [25] allows interactively sorting documents via a 3D user interface (UI). Once the user has sorted the documents the approach is comparable to information landscapes. The differences, however, are that (1) information landscapes are computed automatically and (2) that the user can choose whether he wants to see all the documents or only those containing overview information.

Fowler [8] describes a system based on inter-document similarity. The main display shows keywords and in another frame a list of corresponding documents. The keywords are connected to each other so that a Concept Map (or associative term thesaurus) is displayed. In our opinion the explicit connections can become quite confusing when visualizing large text collections.

Lin [16] uses a self-organizing map for a prototype system in HyperCard. The system uses latent semantic indexing [9] to build the original high-dimensional document space. According to the authors, the extended vector model offers advantages but has not (yet) been implemented in the system.

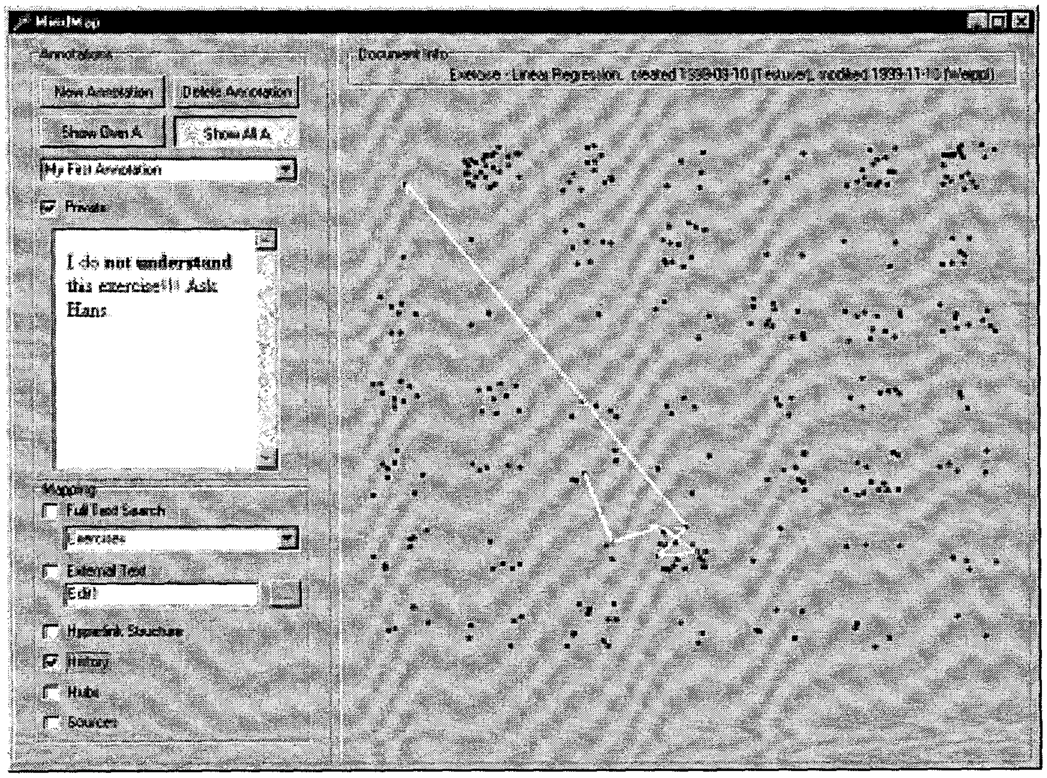
3. HCI Metaphor

Two years ago we started developing a prototype of the 3D information landscape. Due to performance issues we decided to implement a 2D map first. This map is very useful on slow machines and as a quick navigational aid. The 3D version, which can now be rendered at acceptable frame-rates, provides additional hints so that navigating larger text collections is possible.

3.1. 2D Content Based Clustering

The first task is to find a two-dimensional representation containing all texts so that related texts are placed next to each other. The technical details are explained in the section "Calculating the Visualization".

During the last two years we developed and tested navigation tools for hypertext-based authoring software. We realized that authors tend to forget important links to related material not authored by them. We therefore started investigating on how information landscapes can be designed. During the initial stages we tested variations of UIs and finally found an easy-to-use UI (see Figure°1). This information landscape - in the final product Teach/Me [17] it is called MindMap - offers some important features like integration of index searches and visualizing predefined subsets. For example, texts on a certain topic can be placed in a subset. Every subset can then be selected and the specified texts are marked using glyphs. This UI was mainly designed for non-expert users who need easy content-based and non-hierarchical access to information.



Exercise - Linear Regression, created 1999-09-10 (Textuser), modified 1999-11-10 (fwiegol)

My First Annotation

[✓ Private]

I do not understand this exercise!!! Ask Hans

[✓ History]

Figure 2: The 'History' function shows the last six pages that have been read and the one currently open. The white line connects them so that the user can see his trail through the information landscape.

Having created hypertext textbooks ourselves and through discussions with colleagues who use our software (www.coimbra.at) we noticed that the information landscapes lack features for expert users. Therefore we decided to implement an improved version.

3.2. The Third Dimension

Although the expert-user version of our information landscape proves useful in everyday work, we did not give up our first attempts to create 3D information landscapes. The alpha-version we planned to integrate into Teach/Me (see Figure°3) did not make it to the final product, as the hardware requirements were too high. During the last two years the increasing speed of CPUs and especially accelerated 3D video cards changed the situation. Even quite large VRML worlds can now be explored at an acceptable speed.

The first obvious advantage of 3D is that we now have a third spatial dimension for visualization. Each mountain represents one text. The mountains are placed in the 2D base plane the same way as in the 2D information landscape previously described.

The third dimension, i.e. the mountains' height depends on whether the mountain contains detailed or overview information. This visual metaphor is very intuitive: In real life many of us have experienced the overview one has standing on a high mountain. Thus high mountains contain overview information like a table of contents or e.g. an overview over various forms of neural networks. High mountains can easily be spotted from even large distances and helps people reading detailed information to keep the general concept

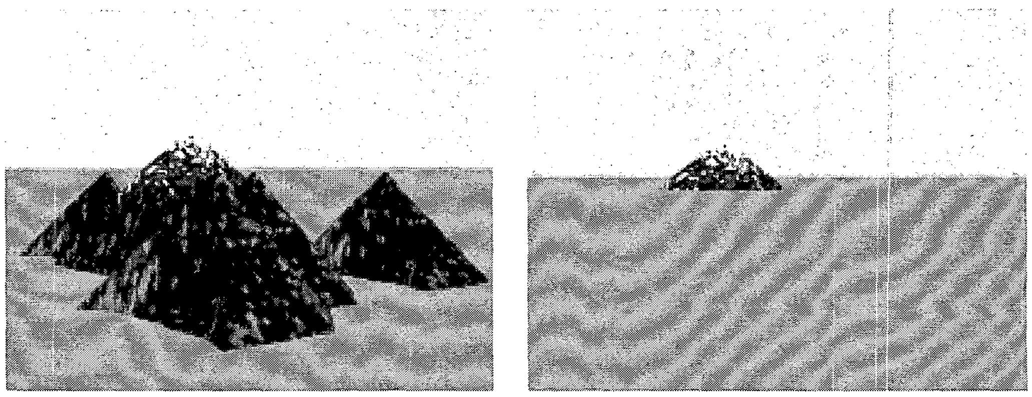


Figure 3: Information landscape: In the left image all mountains can be seen due to the low sea level; the rising sea level in the right image lets the small mountains disappear. As higher mountains represent basic chapters they can be spotted easily during high tide.

in mind all the time. Small mountains, on the other hand represent texts dealing with detailed information. These mountains can only be seen in the vicinity of big mountains and thus people are not confused when first entering the 3D information landscape.

Using the mountains' size as described, enabled us to implement yet another very appealing feature: tidal variations of the sea level. As the information landscape is not a realistic world (e.g. it is not a sphere), the user can control high tide and low tide. During high tide - the sea level is obviously high - small mountains are covered by water and cannot be seen. When entering a world with 3000 mountains this feature is extremely useful. It filters the texts so that only the overview texts can be seen. Once the user has gained an overview or found an interesting region, he or she can make the sea level fall until more detailed mountains appear (see Figure°3).

4. Calculating the Visualization

This section focuses on how information that is necessary to offer user interfaces like the ones described in the previous sections, can be derived from text collections.

4.1. Key-term Vector

To visualize text clusters based on the texts' content, we have to find means to describe a text. A common way to represent a text is to use its words. A very straightforward way is to count how often words are used in each text. All words of all texts build an index, i.e. the list of words. On the basis of this index we build a matrix; each column represents a word of the index, each row one text. We count how often a word is used in a certain text.

Textual data is of high dimensionality, as each word (or term) contained in the set of modeled documents effectively defines an independent axis in a vector space used to describe the set [5]. Each document is represented by the set of words occurring in it and each single word's relative frequency of occurrence, collectively known as a term vector ([30], [2]). These vectors span a high dimensional vector space in which every text is represented as a single point.

4.2. Stemming

The main problem with this basic word-count algorithm is that the vectors quickly become very large. Therefore we had to find ways to classify terms as key-terms.

Stemming is an efficient and sensible way to reduce the number of distinct terms. Porter [22] developed an algorithm to stem English words. Although Porter's algorithm works quite well and reduces the number of distinct terms in our test data sets by approximately 60% the algorithm makes some mistakes. However, due to the redundancy of natural language texts, these errors do not have a noticeable influence on the overall classification.

4.3. Stopwords

A second step is to eliminate words that are not suited well for classification. These words are called stopwords. The central idea of a text is normally communicated using nouns. In general, verbs, adverbs and adjectives are thus not so important for classifying texts.

The first step is to add all verbs, adverbs and adjectives that cannot be a noun to our stopword list. For example, we add "went", "easily" and "big". However, we do not add "show" because "To show" is a verb, but "the show" can also be a noun.

In the second step we add words that should be explicitly excluded. These are words like "the", "and" etc.

Well-designed stopword lists can reduce the number of terms to be included into our key-term vector by approximately 50%-70%. For our system we use word lists from the WordNet project [19], which are available on the Web.

4.4. Self-Organizing Maps (SOMs)

After having evaluated various algorithms (principal component analysis, error backpropagation networks, self-organizing maps and genetic algorithms) we chose the popular SOM algorithm to calculate the projection from the high dimensional key-term vector space to the 2D plane. Kohonen [13] originally uses the dot product to measure the distance between two vectors because in most cases this distance measure is superior to the Euclidean distance. The main drawback of the Euclidean distance is that it is very error-prone in case of outliers. P tzelsberger [23] proposes a method that allows to combine these two distance measures and to select a metric that lies anywhere in-between the two.

4.5. Hubs and Sources

Kleinberg [11] developed a method to improve Web search engines. Web pages are classified as hubs and sources. A good hub is a page that contains links to various good sources. Hubs can be considered to be index pages, whereas a source is a page that contains detailed information. A source is classified to be a good source, if many good hubs point to this source.

Kleinberg's algorithm is quite complex to understand, because he has to find ways to classify hubs and sources as good and bad. As hubs are considered to be good if they reference good sources and good sources are classified as good if they are pointed to by many good hubs, the algorithm has to deal with loops of references.

In our case, the task is much simpler. As the goal is to visualize all hypertexts of a database, all pages can be considered to be relevant. This is a major difference to Kleinberg's algorithm that has been designed to improve Web searches, i.e. to select pages.

All pages having a fan-out of more than a certain threshold are hubs and pages that are not hubs are sources. We do not use Boolean logic to classify a page as hub or source but we use a fuzzy-continuous scale that determines to what degree a page is hub or source. If a text is a hub it is assigned "1" as z-coordinate, a source "0". Other values in the range [0;1] are assigned to those texts that lie somewhere in-between.

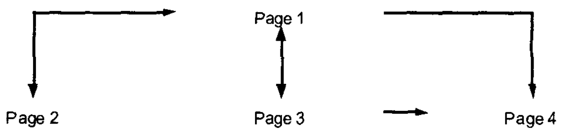


Figure 4: The graph shows the link structure of some HTML documents. Page 1 has a fan-out of 3 (FO(P1)=3) and a fan-in of 2 (FI(P1)=2).

Using the data given in Figure 4, page 1 (P1) has the highest fan-out (FO(P1)=3). Therefore it is classified as hub with the z-coordinate z=1. The table shows the z-coordinates for all pages of this example.

Textno.	FI	FO	Z-coordinate
Page 1	2	3	1

Page 2	1	1	0.33
Page 3	1	2	0.66
Page 4	2	0	0

One problem we had to solve was "backlinks", i.e. links from a source-page back to a hub like a table-of-contents or index page. Kleinberg has not dealt with this issue because the effects are thought to be neglectable in the context of Web wide searches. Our solution is that we do not count "index", "next" and "back" links.

4.6. Examples

We have thoroughly tested the information landscape with Teach/Me [17] and various abstract collections ranging from 360 to 1400 abstracts. During the Beta tests of Teach/Me twenty-five university students and sixty-two high-school students were interviewed and observed in order to optimize the UI.

Using the information landscape to visualize Reuters-21578 text collection and 20000 Usenet articles [3] show that the training of the SOM does not scale well. Beyond 30000 articles the SOM quickly becomes the bottleneck in preprocessing. We are currently investigating how we can modify our approach to deal with larger text collections.

5. Conclusions and Future Work

We have shown a way to develop a powerful user interface for exploring hypertext databases. Our next step will be to improve the SOM's training stage so that large text collections can be dealt with, too. We assume that using prototypes representing subsets of the high dimensional space [28] will be the best way.

Another important issue is that the third dimension can only be calculated for hypertexts and not for text collections like Reuters-21578. We are convinced that this is possible by analyzing the variety of words a text uses. Texts containing many different key-phrases just once or twice are likely to present overview information, whereas texts with a limited scope of key-phrases are probably dealing with very detailed information.

6. References

1. Keith Andrews et al. Towards Rich Information Landscapes for Visualising Structured Web Spaces. Proc. of 2nd IEEE Symposium on Information Visualization. 1996.

2. Ricardo Baezo-Yates and Berthier Ribeiro-Neto. Modern Information Retrieval. Addison-Wesley, Harlow, England, Reading Massachusetts. 1999

3. S. D. Bay, The UCI KDD Archive [http://kdd.ics.uci.edu]. Irvine, CA: University of California, Department of Information and Computer Science, 1999.

4. Chris Buckley and Gerard Salton and James Allan, The Effect of Adding Relevance Information in a Relevance Feedback Environment. Proceedings of ACM SIGIR. 1994

5. Stuart K. Card and Jock Mackinlay, The Structure of the Information Visualization Design Space. Proceeding of the IEEE Symposium on Information Visualization. IEEE Computer Society Technical Committee on Computer Graphics. IEEE Computer Society, 1997

6. Matthew Chalmers and Paul Chitson, Bead: Explorations in Information Visualization. Proceedings of ACM SIGIR. 1992

7. Douglass R. Cutting and Jan Pedersen and David Karger and John W. Tukey, Scatter / Gather: A Cluster-based Approach to Browsing Large Document Collections. Proceedings of ACM SIGIR. 1992

8. Richard H. Fowler and Wendy A. L. Fowler and Bradley A. Wilson, Integrating Query, Thesaurus, and Documents Through a Common Visual Representation. Proceedings of ACM SIGIR. 1991

9. G.W. Furnas and S. Deerwester and T.K. Landauer and R.A. Harshman and L.A. Streeter and Lochbaum, Information retrieval using a singular value decomposition model of latent semantic structure. 11th International Conference on R&D in Information Retrieval. 1988

10. Timo Honkela and Samuel Kaski and Krista Lagus and Teuvo Kohonen, WEBSOM - Self-Organizing Maps of Document Collections. Proceedings of WSOM'97, Workshop on Self-Organizing Maps, Espoo, Finland, June 4-6. Helsinki University of Technology, Neural Networks Research Centre, 1997

11. Jon M. Kleinberg, Authoritative Sources in a Hyperlinked Environment, Communications of the ACM, 1999

12. Teuvo Kohonen, Self-Organizing Maps. Springer, Berlin, Heidelberg, New York, London, Paris, Tokyo. 1997

13. Teuvo Kohonen, Self-Organization and Associative Memory. Springer, Berlin, Heidelberg, New York, London, Paris, Tokyo. 1989