Circle Graphs: New Visualization Tools for Text-Mining

Yonatan Aumann, Ronen Feldman, Yaron Ben Yehuda, David Landau, Orly Liphstat, Yonatan Schler

Department of Mathematics and Computer Science
Bar-Ilan University
Ramat-Gan, ISRAEL
Tel: 972-3-5326611
Fax: 972-3-5326612
feldman@cs.biu.ac.il

Abstract. The proliferation of digitally available textual data necessitates automatic tools for analyzing large textual collections. Thus, in analogy to *data mining* for structured databases, *text mining* is defined for textual collections. A central tool in text-mining is the analysis of *concept relationship,* which discovers connections between different concepts, as reflected in the collection. However, *discovering* these relationships is not sufficient, as they also have to be presented to the user in a meaningful and manageable way. In this paper we introduce a new family of visualization tools, which we coin *circle graphs,* which provide means for visualizing concept relationships mined from large collections. Circle graphs allow for instant appreciation of multiple relationships gathered from the entire collection. A special type of circle-graphs, called *Trend Graphs,* allows tracking of the evolution of relationships over time.

1 Introduction

Most informal definitions [2] introduce *knowledge discovery in databases* (KDD) as the extraction of useful information from databases by large-scale search for interesting patterns. The vast majority of existing KDD applications and methods deal with structured databases, for example, client data stored in a relational database, and thus exploits data organized in records structured by categorical, ordinal, and continuous variables. However, a tremendous amount of information is stored in documents that are essentially unstructured. The availability of document collections and especially of online information is rapidly growing, so that an analysis bottleneck often arises also in this area. Thus, in analogy to *data mining* for structured data, *text mining* is defined for textual data. *Text mining* is the science of extracting information from hidden patterns in large textual collections.

Text mining shares many characteristics with classical data mining, but also differs in some. Thus, it is necessary to provide special tools geared specifically to text mining. A central tool, found valuable in text mining, is the analysis of *concept*

J.M. Żytkow and J. Rauch (Eds.): PKDD'99, LNAI 1704, pp. 277–282, 1999.

relationship [3,4], defined as follows. Large textual corpuses are most commonly composed of a collection of separate *documents* (e.g. news articles, web pages). Each document refers to a set of *concepts* (*terms*). Text mining operations consider the distribution of concepts on the *inter-document* level, seeking to discover the nature and relationships of concepts as reflected in the collection as a whole. For example, in a collection of news articles, a large number of articles on politician X and "scandal" may indicate a negative image of the character, and alert for a new PR campaign. Or, for another example, a growing number of articles on both company Y and product Z may indicate a shift of focus in the company's interests, a shift which should be noted by its competitors. Notice that in both of these cases, the information is not provided by any single document, but rather from the totality of the collection. Thus, *concept relationship* analysis seeks to discover the relationship between concepts, as reflected by the totality of the corpus at hand.

Clearly, discovering the concept relationships is only useful insofar as this information can be conveyed to the end-user. In practice, even medium-sized collections tend to give rise to a very large number of relationships. Thus, a mere listing of the discovered relationships is of little practical use for the end-user, as it is too large to comprehend. In addition, a linear list fails to show the structure arising from the entirety of relationships. Thus, we find that in order for mining of concept-relationship to be a useful tool for text-analysis, proper visualization techniques are necessary for presenting the results to the end user in a meaningful and manageable form.

In this paper we introduce a new family of visualization tools for text-mining, which we call *Circle Graphs*. Circle graphs prove to be an effective tool for visualizing concept relationships discovered in text-mining. The graphs provide the user with an instant overall view of many relationships at once. Thus, circle graphs provide the extra benefit of surfacing the overall structure emerging from the multitude of relationships.

We describe two specific types of circle graphs:

1. Category Connection Graphs: Provide a graphic representation of relationships between concept in different categories.
2. Context Connection Circle Graphs: Provide the user with a graphic representation of the connection between entities in a give context.

2 Circle Graphs

We now describe the circle graphs visualization. We first give some basic definitions and notations.

2.1 Definitions and Notations

Let T a be taxonomy. T is represented as a DAG (Directed Acyclic Graph), with the terms at the leaves. For a given node $v \in T$, we denote by $Terms(v)$ the terms which are decedents of v.

Let D be a collection of documents. For terms e_1 and e_2 we denote $sup_D(e_1,e)$ the number of documents in D which indicate a relationship between e_1 and e_2. The nature of the indication can be defined individually according to the context. In the current implementation we say that a document indicates a relationship if both terms appear in the document in the same sentence. This has proved to be a strong indicator. Similarly, for a collection D and terms e_1, e_2 and c, we denote by $sup_D(e_1,e_2,c)$ the number of documents which indicate a relationship between e_1 and e_2 in the context of c (e.g. relationship between the UK and Ireland in the context of peace talks). Again, the nature of indication may be determined in many ways. In the current implementation we require that they all appear in the same sentence.

2.2 Category Connection Maps

Category Connection Maps provide a means for concise visual representation of connections between different categories, e.g. between companies and technologies, countries and people, or drugs and diseases. In order to define a category connection map, the user chooses any number of categories from the taxonomy. The system finds all the connections between the terms in the different categories. To visualize the output, all the terms in the chosen categories are depicted on a circle, with each category placed on a separate part on the circle. A line is depicted between terms of different categories which are related. A color coding scheme represents stronger links with darker colors.

Formally, given a set $C=\{v_1,v_2,\ldots,v_k\}$ of taxonomy nodes, and a document collection D, the category connection map is the weighted G defined as follows. The nodes of the graph are the set $V=terms(v_1)\cup terms(v_2)\cup\ldots\cup terms(v_k)$. Nodes $u,w \in V$ are connected by an edge if:

1. u and w are from different categories
2. $sup_D(u,w)>0$.
3. The *weight* of the edge (u,w) is $sup_D(u,w)$.

An important point to notice regarding Category Connection Maps is that the map presents in a single picture information from the entire collection of documents. In the specific example, there is no single document that has the relationship between all the companies and the technologies. Rather, the graphs depicts aggregate knowledge from hundreds of documents. Thus, the user is provided with a bird's-eye summary view of data from across the collection.

The Category Connection Maps are dynamic in several ways. Firstly, the user can choose any node in the graph and the links from this node are highlighted. In addition, a double-click on any of the edges brings the list of documents which support the given relationship, together with the most relevant sentence in each document. Thus, in a way, the system is the opposite of search engines. Search engines point to documents, in the hope that the user will be able to find the necessary information. Circle category connection maps present the user with the information itself, which can then be backed by a list of documents.

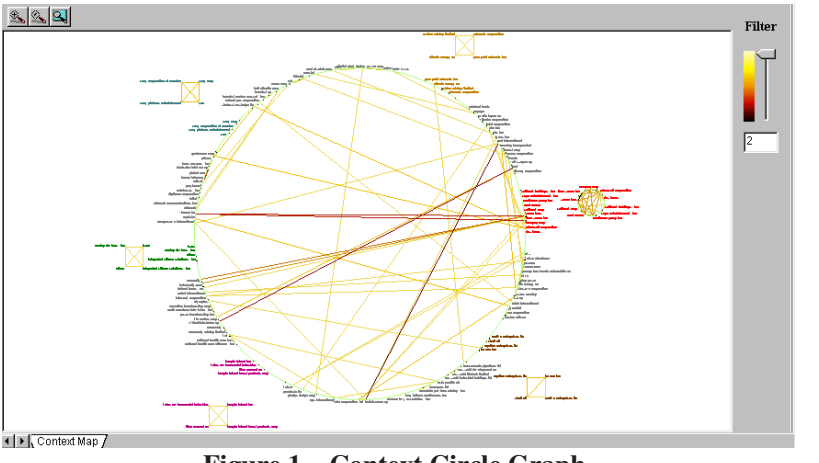

Figure 1 – Context Circle Graph.

The graph presents the connections between companies in the context of "joint venture". Clusters are depicted separately. Color coded lines represent the strength of the connection. The information is based on 5,413 news articles obtained from Marketwatch.com. Only connection with weight 2 or more are depicted.

2.3 Context Circle-Graphs

Context Circle-Graphs provide a visual means for concise representation of the relationship between many terms in a given context. In order to define a context circle graph the user defines:

1. A taxonomy category (e.g. "companies"), which determines the nodes of the circle graph (e.g. companies)
2. An optional context node (e.g. "joint venture"): which will determine the type of connection we wish to find among the graph nodes.

Formally, for a set of taxonomy nodes vs, and a context node C, the Context Circle Graph is a weighted graph on the node set $V=terms(vs)$. For each pair $u,w \in V$ there is an edge between u and w, if there exists a context term $c \in C$, such that $sup_D(u,w,c)>0$. In this case the weight of the edge is $\Sigma_{c \in C}\ sup_D(u,w,c)$. If no

context node is defined, then the connection can be in any context. Formally, in this case the root of the taxonomy is considered as the context.

A Context Circle Graph for "companies" in the context of "joint venture" is depicted in Figure 1. In this case, the graph is clustered, as described below. The graph is based on 5,413 news documents downloaded from Marketwatch.com. The graph gives the user a summary of the entire collection in one visualization. The user can appreciate the overall structure of the connections between companies in this context, even before reading a single document!

2.3.1 Clustering

For Context Circle Graph we use clustering to identify clusters of nodes which are strongly inter-related in the given context. In the example of figure 1, the system identified six separate clusters. The edges between members of each cluster are depicted in a separate small Context Circle Graph, adjacent to the center graph. The center graph shows connections between terms of different clusters, and those with terms which are not in any cluster. We now describe the algorithm for determining the clusters.

Note that the clustering problem here is different from the classic clustering problem. In the classic problem we are given points in some space, and seek to find clusters of points which are close to each other. Here, we are given a graph in which we are seeking to find dense sub-graphs of the graph. Thus, a different type of clustering algorithm is necessary.

The algorithm is composed of two main steps. In the first step we assign weights to edges in the graph. The weight of an edge reflects the strength of the connection between the vertices. Edges incident to vertices which are in the same cluster should be associated with high weights.

In the next step we identify sets of vertices which are dense-subgraphs. This step uses the weights assigned to the edges in the previous one.
We first describe the weight-assignment method. In order to evaluate the strength of a link between a pair of vertices u and v, we consider two criteria:
Let u be a vertex in the graph. We use the notation $\Gamma(u)$ to represent the neighborhood of u. The *cluster weight* of (u,v) is affected by the similarity of $\Gamma(u)$ and $\Gamma(v)$. We assume that vertices within the same clusters have many common neighbors.
Existence of many common neighbors is not a sufficient condition, since in dense graphs any two vertices may have some common neighbors. Thus, we emphasize the neighbors which are close to u and v in the sense of *cluster weight*. Suppose $x \in \Gamma(u) \cap \Gamma(u)$, if the *cluster weights* of (x,u) and (x,v) are high, there is a good chance that x belongs to the same cluster as u and v.

We can now define an *update operation* on an edge (u,v) which takes into account both criteria: $w(u,v)=\sum_{x\in\Gamma(u)\cap\Gamma(v)} w(x,u)+\sum_{x\in\Gamma(u)\cap\Gamma(v)} w(x,v)$

The algorithm starts with initializing all weights to be equal, $w(u,v)=1$ for all u,v. Next, the update operation is applied to all edges iteratively. After a small number of iterations (set to 5 in the current implementation) it stops and outputs the values associated with each edge. We call this the *cluster weight* of the edge.

The cluster weight has the following characteristic. Consider two vertices u and v within the same dense sub-graph. The edges within this sub-graph mutually affect each other. Thus the iterations drive cluster weight $w(u,v)$ up. If, however, u and v do not belong to the dense sub-graph, the majority of edges affecting $w(u,v)$ will have lower weights, resulting in a low *cluster weight* assigned to (u,v).

After computing the weights, the second step of the algorithm finds the clusters. We define a new graph with the same set of vertices. In the new graph we consider only a small subset of the original edges, whose weights were the highest. In our experiments we took the top 10% of the edges. Since now almost all of the edges are likely to connect vertices within the same dense sub-graph, we thus separate the vertices into clusters by computing the connected components of the new graph and considering each component as a cluster.

Figure 1 shows a circle context graph with six clusters. The cluster are depicted around the center graph. Each cluster is depicted in a different color. Nodes which are not in any cluster are colored gray. Note that the nodes of cluster appear both in the central circle and in the separate cluster graph. Edges within the cluster are depicted in the separate cluster graph. Edges between clusters are depicted in the central circle.

References

1. Eick, S.G. and Wills, G.J.; Navigating Large Networks with Hierarchies. *Visualization '93,* pp. 204-210, 1993.

2. Fayyad, U,; Piatetsky-Shapiro, G.; and Smyth P. Knowledge Discover and Data Mining: Towards a Unifying Framework. In *Proceedings of the 2nd International Conference of Knowledge Discovery and Data Mining*, 82-88, 1996.

3. Feldman, R.; and Dagan, I.; KDT - Knowledge Discovery in Texts. In *Proceedings of the 1sr International Conference of Knowledge Discovery and Data Mining.* 1995.

4. Feldman, R.; Klosgen, W.; and Zilberstein, A.; Visualization Techniques to Explore Data Mining Results for Document Collections. In *Proceedings of the 3rd International Conference of Knowledge Discovery and Data Mining*, 16-23. 1997.

5. Hendley, R.J., Drew, N.S., Wood, A.M., and Beale R.; Narcissus: Visualizing Information. In *Proc. Int. Symp. On Information Visualization,* pp. 90-94, 1995.