# Community Finding in Large Social Networks Through Problem Decomposition

Anand Narasimhamurthy, Derek Greene, Neil Hurley, and Pádraig Cunningham

School of Computer Science and Informatics, University College Dublin
{anand.narasimhamurthy,derek.greene,neil.hurley,padraig.cunningham}@ucd.ie

**Abstract.** The identification of cohesive communities is a key process in social network analysis. However, the algorithms that are effective for finding communities do not scale well to very large graphs, since their time complexity is worse than linear in the number of edges in the graph. This is an important issue as there is considerable interest in applying social network analysis to large datasets, such as networks of mobile phone subscribers. In this respect the contributions of this paper are two-fold. First we demonstrate these scaling issues using a prominent community-finding algorithm as a case study. We then show that a two-stage process, whereby the network is first decomposed into manageable subnetworks using a multilevel graph partitioning procedure, is effective in finding communities in networks in networks that cannot be handled by existing community finding techniques alone.

## 1 Introduction

After several years of academic research on social network analysis (SNA), considerable commercial interest in exploiting SNA has recently emerged. The research reported in this paper is motivated by the prospect of using SNA in large-scale applications, such as exploring online communities [1], mining web usage data [2], identifying research trends in bibliographic networks [3], or analysing relationships among mobile telephone subscribers. The identification of cliques or communities in network data has received a lot of attention in SNA research, where a number of promising techniques have emerged [4–6]. These techniques can identify subgraphs such that the density of edges within the subgraph is greater than the density of edges connecting the subgraph to the rest of the network. They can also identify overlapping communities, where an individual belongs to more than one group. This is an important facility for network analysis in many real-world domains.

Another highly significant issue in this area is that of scalability. Unfortunately, it is in the nature of the analysis entailed in common community-finding algorithms that they do not scale well to very large graphs. In fact, existing techniques in this area generally cannot handle graphs with more than a few tens of thousands of nodes (*e.g.* [6]). In contrast, many real-world networks will be substantially larger. For instance, in a study reported by Abello et al. [7], a

one-day telephone call graph at AT&T consisted of 53,767,087 vertices and more than 170 million edges.

To deal with this issue of scalability, we propose a pragmatic problem decomposition strategy: use a "top-down" graph partitioning technique to decompose the network into smaller subnetworks, on which it is then feasible to apply a more computationally intensive community-finding algorithm. To perform the initial partitioning, we use the multilevel method proposed by Dhillon et al. [8], referred to as Graclus. For the second stage of the process, the CFinder algorithm [5] is employed to discover small, overlapping communities. The results of evaluations on synthetic and real data are presented in order to demonstrate the effectiveness of this strategy. In both cases, we focus on networks with community structures similar to those occurring in large-scale, real-world networks such as those pertaining to mobile phone subscribers – *i.e.* small, dense, localised communities embedded in a very large sparse graph. We show that we can discover communities in large graphs in a computationally efficient manner, without adversely affecting the "quality" of these communities.

The remainder of this paper is organised as follows. The next section provides a summary of existing methods which are relevant in SNA. We describe our proposed problem decomposition strategy in Section 3. The datasets used in our experiments are presented in Section 4, and the findings of our evaluation are subsequently discussed in Section 5. The paper finishes with some conclusions in Section 6. Note that an extended version of this paper is available as a technical report with the same title [9].

## 2 Related Work

While the distinction between the community-finding and the graph partitioning algorithms discussed here is not always clear-cut, we can make general distinctions. Community-finding algorithms are designed to find regions of dense structure that are not well connected with the rest of the network. These regions will often exhibit a certain degree of overlap. In contrast, the graph partitioning problem involves finding the optimal division of a graph into $k$ disjoint parts according to a chosen criterion, such that the partition covers the entire graph. Real applications include VLSI module placement, load balancing for parallel processing, and image segmentation.

**Community-finding algorithms.** We briefly review three prominent algorithms that have been employed for community-finding in SNA tasks. The GN algorithm presented by Newman & Girvan [4] is essentially a top-down hierarchical clustering strategy, but with an alternative partitioning criterion. This criterion is based on the shortest path betweenness measure [10], which is well-established as a measure of node centrality in SNA. In the GN algorithm the "edge betweenness" refers to the number of shortest paths in a network that pass along an edge. Edges that score highly on this criterion are likely to be inter-cluster edges (*i.e.* edges that link adjacent clusters). The CONGA technique [6] extends the GN algorithm to handle overlapping communities. This approach

employs a divisive hierarchical strategy. In order to allow nodes to belong to more than one cluster, CONGA permits nodes to be split, i.e. a real node $v$ will be split into $\{v_1, v_2\}$ with the incoming edges to $v$ divided between $v_1$ and $v_2$ and a new *virtual* edge created to link $v_1$ and $v_2$. In contrast to the GN method which is top-down, the Clique Percolation Method proposed by Palla *et al.* [5] is a bottom-up technique that uses cliques as building blocks for large groups. Their approach first extracts all maximal complete subgraphs (*i.e.* cliques) that do not form part of larger complete subgraphs, and composes these into larger structures. Although an efficient approach for finding all cliques in a general network may not be feasible since determining a maximum clique is NP-hard, Palla *et al.* nevertheless claim that their approach performs well on real networks.

It is important to note that techniques for optimising an appropriate *connectedness* criterion will often have bad time complexity, thus limiting their applicability to very large graphs. Newman & Girvan say the time complexity of their algorithm is $O(n^3)$ on sparse graphs, so it could be applied to graphs of up to 10,000 nodes at the time of publication in 2003. Gregory states that the worst case time complexity of this algorithm is $O(e^3)$, where $e$ is the number of edges. In practice CONGA has been shown to handle networks of up to 4,000 nodes and 7,000 edges. Palla *et al.* point out that it is difficult to analyse the time complexity of CFinder given the nature of the algorithm.

**Graph partitioning.** Of the vast array of approaches that have been proposed in the literature, two of the most important classes are spectral clustering and multilevel partitioning. Spectral methods produce a partition based on the eigendecomposition of the graph. Spectral approximations for a variety of partitioning criteria have been formulated, including the minimum cut [11], ratio cut [12], and normalised cut [13]. Many multilevel approaches for graph partitioning have been developed over the years, the Metis algorithm [14] being the most well-known example. In these approaches a sequence of successively smaller, coarser hypergraphs is computed. A bisection of the smallest hypergraph is computed and is successively projected to the next finest level. At each level, an iterative refinement algorithm is used to improve the bisection. Most multilevel algorithms are based on the seminal work by Kernighan & Lin [15]. Recently, Dhillon *et al.* [8] developed a fast multi-level algorithm that directly optimises various weighted graph clustering objectives. They show that a general weighted $k$-means objective is mathematically equivalent to a weighted graph clustering objective and exploit this equivalence. The main advantage of their method is that it optimises this objective without requiring an eigendecomposition, which can be computationally costly for large graphs. Another advantage is that, unlike other popular multilevel approaches, it does not require the partitions to be of equal sizes.

## 3 Problem decomposition strategy

While community-finding algorithms have been aimed at SNA applications, these techniques are computationally expensive and the communities they discover are small – normally comprising some tens of nodes. When dealing with very large

graphs, which may potentially contain hundreds of thousands or even millions of nodes, algorithms with running time $O(e^2)$ or $O(e^3)$ will not be practical. Therefore the question arises, how can we find small communities in such large networks? The approach we take in this paper is to employ a two-stage problem decomposition strategy outlined below:

**Stage 1:** Given a large graph, apply a computationally tractable graph partitioning algorithm that minimises the number of links broken in the process.

**Stage 2:** Once the original graph has been split into manageable subgraphs, apply a more computationally intensive community-finding algorithm to each subgraph and combine the results.

For the first stage of the process, we suggest the use of the implementation of the multilevel partitioning method proposed by Dhillon *et al.* [8], referred to as Graclus[1]. We choose this approach as it allows us to optimise the objectives for spectral clustering, which have previously proved successful in other areas [13], but without requiring a costly spectral decomposition. Thus Graclus will provide a partitioning of a large graph with a small running time. For the second stage of the process, we employ the CFinder algorithm [5], as it supports the discovery of overlapping, localised and nested communities. It should be noted that, while we focus on the pairing of Graclus and CFinder in the remainder of this paper, any reasonable combination of algorithms can be used.

## 4 Experimental Datasets

In this paper we perform two sets of experiments, one using synthetic data, the other using a real-world network. We now describe the details of these datasets.

**Synthetic data.** We made use of artificial data for two main purposes. Firstly, we wished to examine how community finding techniques scale to large, sparse graphs, approaching the size of those occurring in real-world problems such as measuring churn in mobile subscriber networks. Secondly, we wished to assess to what extent graph partitioning techniques preserve communities intact.

To construct the artificial datasets[2] for our evaluation, we followed a procedure similar to that outlined by Newman & Girvan [4]. The basic idea is that nodes in each of the communities acquire edges among other members of the same community, as well as nodes outside the community, with certain probabilities. Let $p_{in}$ denote the probability with which a node in a community acquires edges with other members of the same community, and $p_{out}$ denote the probability with which it acquires edges randomly with the rest of the nodes. When constructing artificial graphs, values for these two parameters can be reached in a number of ways. For instance, Newman & Girvan [4] fix the expected degree of each vertex, and then compute values for $p_{in}$ and $p_{out}$ accordingly.

[1] Available from `http://www.cs.utexas.edu/users/dml/Software/graclus.html`

[2] Available for download at `http://mlg.ucd.ie/datasets/graph.html`

While the term "community" is not clearly defined in the literature, in most cases it is used to refer to a group of nodes which have a high degree of connectivity among themselves, relative to the rest of the network. In order to capture this notion of a community, we introduce the measure *assoc*, which quantifies the connectivity between a community $C$ and a graph $G$, where $C \subset G$,

$$assoc(C, G) = \sum_{u \in C, v \in G, u \neq v} w(u, v) \tag{1}$$

and where $w(u, v)$ is the weight of the edge connecting a pair of nodes $u$ and $v$. Note that in the data described here, edge weights are either 1 or 0. In this case $assoc(C, C)$ is a count of the number of edges connecting nodes within $C$, and we can use $p_{in}(C)$ as a quality measure where

$$p_{in}(C) = \frac{assoc(C, C)}{c \times (c-1)/2} \tag{2}$$

and $c = |C|$. Another measure of community quality is the ratio $r(C)$ of the within-community connectivity in $C$ to the connectivity of the community's nodes to the entire graph:

$$r(C) = \frac{assoc(C, C)}{assoc(C, G)} \tag{3}$$

To construct the graphs used in our experiments, we fix the values for the two measures $p_{in}(C)$ and $r(C)$ for all communities – thus the graphs are defined by the parameters $p_{in}$ and $r$. From these values, we can compute the other parameter $p_{out}$ used by Newman & Girvan [4]. We chose to fix $p_{in}$ as opposed to $p_{out}$, since it has a straightforward interpretation – it is a measure of how close the community is to forming a maximal clique. For evaluation purposes, we constructed "unit" graphs consisting of 1,000 nodes, containing embedded communities of different sizes, ranging from 10 to 40 nodes. Successively larger graphs were then generated such that the number of nodes and communities were scaled by the same integral ratio. For example, a graph of 2,000 nodes had exactly twice the number of communities of the same respective sizes as a graph of 1,000 nodes.

**Real data.** The CORA bibliographic dataset [16] contains information and annotations such as authors, cited papers and topic for over 50,000 research papers[3]. For our evaluation, the paper citation graph derived from the CORA dataset was used. Only the papers with available authors were selected, where the total number of these was 28,400. We suggest that this graph is a reasonable proxy for real-world data, such as mobile subscriber networks, since the communities are small despite the scale of the graph – typically containing 10 to 20 members. The edge directions and weights were ignored and the largest weakly connected component was used for the experiments described in the next section. This resulted in a graph consisting of 24,542 nodes.

[3] See `http://www.cs.umass.edu/~mccallum/code-data.html`

## 5 Experimental Evaluation

The evaluation presented in this section has two objectives: (i) to highlight the scalability advantages of the two stage process, (ii) to demonstrate that the prior application of graph partitioning does not adversely affect our ability to locate communities in a large graph. For the synthetic data, we ran experiments on multiple datasets generated using a range of values for the parameters $p_{in}$ and $r$. Although the actual running times varied, the trends were highly similar. Hence we discuss in detail results corresponding to one set of values, namely $p_{in} = 0.7$ and $r = 0.7$. These values correspond to a problem where communities have reasonably well-defined "signatures", but where it is still potentially challenging for graph partitioning algorithms to recover these communities intact. Note that, due to space constraints, we refer the reader to [9] for a full discussion of the results produced during the course of our evaluation.

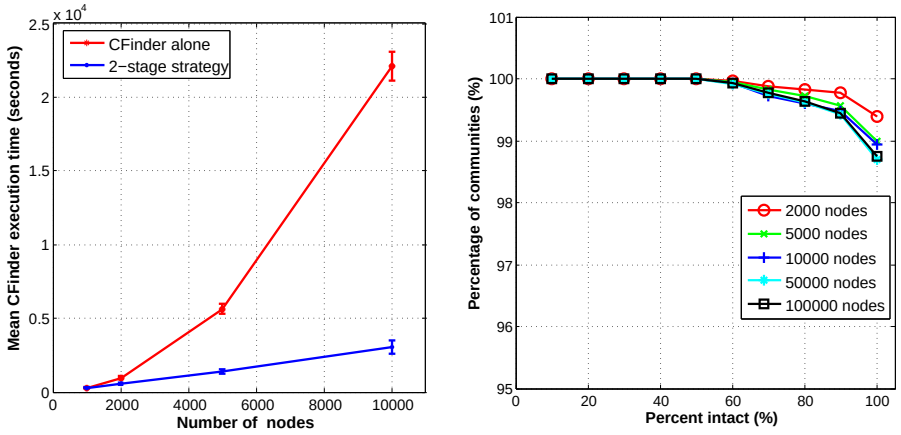
### 5.1 Scalability

Firstly, we consider the problem of scaling community-finding algorithms such as CFinder to very large graphs.

**Synthetic data.** We illustrate the results pertaining to running time of CFinder on graphs generated as described in Section 4. The average size of communities remains constant as the graphs grow – *i.e.* the average community size in a large network is not greater than a smaller network. The experiments were repeated multiple times. The average execution times for CFinder (computed from 50 runs) are shown in Figure 1. We observe a steep increase in running time as the graphs grow in size, even though the average community size is the same. For example, in Figure 1(a) we see that the average execution time on a graph consisting of 5,000 nodes is nearly 18 times greater and for a 10,000 node graph is nearly 70 times greater than that of a graph of 1,000 nodes.

The running times for the problem decomposition strategy are also shown in Figure 1(a). It can be clearly seen that this strategy results in a dramatic reduction in total running time. The combined times are computed as follows. Let the graph be partitioned into $k$ parts. The total running time on a graph may be expressed as $T = G_k + \sum_{i=1}^{k} C_i$, where $G_k$ denotes the execution time for partitioning the graph into $k$ parts using Graclus, and $C_i$ denotes the execution time of CFinder on the $i^{th}$ subgraph resulting from the graph partitioning. For our datasets, the running time for Graclus was usually a fraction of a second for graphs consisting of up to a few tens of thousands of nodes. Thus the total running time was dominated by the CFinder execution times on the individual subgraphs.

We have only shown the combined running times for graphs containing up to 10,000 nodes in Figure 1 for the sake of clarity of illustration and direct comparison. We found that graphs of 10,000 nodes and more than 80,000 edges were close to the limit of what can be handled on a 2GHz dual core server with 4GB RAM, running 64-bit Linux. However, the problem decomposition strategy




(a) (b)

**Fig. 1. (a)** Scalability : Mean execution times against graph size for CFinder on synthetic data for (i) CFinder alone, (ii) two-stage strategy; **(b)** Solution quality of graph partitioning – *i.e.* fraction of communities preserved intact to different degrees

permits discovery of communities in graphs of sizes significantly larger than possible using the community finding algorithm on its own.

We remark here that the actual running time of both CFinder and Graclus vary somewhat depending on the actual graph, as well as the configuration of the machine they are run on. In many cases, it is hard to exactly quantify the size dependence of these algorithms. However, spectral clustering techniques in general and Graclus in particular (which avoids a computationally costly eigendecomposition) can usually be run on much larger graphs than community-finding algorithms. Thus in most cases the two-stage problem decomposition approach would help to "extend the reach" of community finding algorithms. Another point to note is that while the total combined execution times are shown in Figure 1(a), the community finding algorithm can be run in parallel on the individual subgraphs resulting from the graph partitioning, thereby minimising the real time required for processing the full graph.

**Real data.** For the synthetic data, the exact number and size of communities was known in advance. While we cannot assume such a "ground truth" in the case of the CORA dataset, it is straightforward to check whether any communities discovered by running CFinder on the entire graph are missing when CFinder is run on the individual subgraphs obtained from graph partitioning. In the case of the two-stage process, the communities obtained from the different subgraphs were pooled together, with duplicates removed. As a first step we compared the total number of communities obtained from the entire graph with those obtained from partitioning the graph into $\{2, 3, \ldots, 10\}$ parts. These are illustrated in Figure 2, with the actual numbers shown in Table 1.

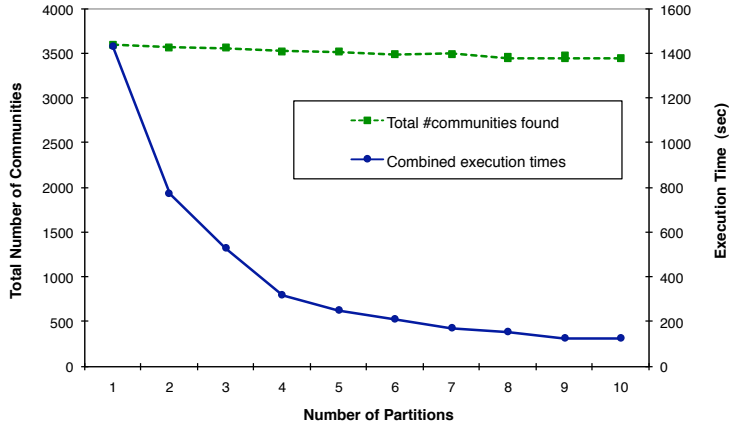



**Fig. 2.** The first plot shows the total number of communities found on the CORA dataset, while the second plot shows actual execution times (in seconds). Note that the first data point in both plots corresponds to the single stage process (*i.e.* CFinder on its own), while the remaining points correspond to the combined two-stage process.

| # Partitions | # Communities (combined) | Total execution time |
|---|---|---|
| 1 | 3601 | 1430.98 |
| 2 | 3572 | 773.55 |
| 3 | 3559 | 529.42 |
| 4 | 3524 | 318.52 |
| 5 | 3517 | 250.46 |
| 6 | 3490 | 209.66 |
| 7 | 3501 | 173.18 |
| 8 | 3470 | 156.15 |
| 9 | 3476 | 126.18 |
| 10 | 3448 | 128.83 |

**Table 1.** Comparison of CFinder on the paper citation graph and its subgraphs, in terms of the number of communities discovered and execution times (in seconds).

Next we compared the specific communities discovered by both processes and found very significant overlap. For example, the total number of communities obtained by running CFinder on the full graph was 3,601 and the total number corresponding to partitioning into two subgraphs was 3,572. The number of communities in the two sets that were identical was 3,454. Of the remaining 147 communities obtained from the full graph, there were close matches amongst the remaining 118 obtained from two subgraphs. When the graph was partitioned into ten subgraphs, the total number of communities was 3,448, of which 2,967 had exact matches from among the communities of the full graph. Thus, while a significant reduction in execution time was achieved when the two-stage procedure was employed, there was also minimal loss of information overall.

### 5.2 Solution Quality

The second objective of our evaluation was to assess the extent to which graph partitioning preserves communities when applied to a large graph. For this purposes, the synthetic data described in Section 4 is useful for assessing solution quality, since it allows us to directly determine whether the communities inserted into the graph are preserved by the first phase of the two stage process (*i.e.* graph partitioning using Graclus). We adopt the following procedure to assess to what extent communities are preserved:

- For each experimental run, compute the proportion of communities where all the community members are grouped together in a single cluster (*i.e.* kept fully intact) by the graph partitioning approach.
- Average this value over multiple runs – *i.e.* compute the mean percentage of communities preserved fully intact.
- Repeat the above steps for different extents of intactness ($10\%, \ldots, 90\%$).

The results of this evaluation are illustrated in Figure 1(b), where the the $x$-axis represents the degree of intactness (percentage) and the $y$-axis the proportion (percentage) of communities preserved to that degree. The different plots correspond to graphs of different sizes (2,000 to 100,000 nodes). We see from the figure that, communities were fully intact at least 98% of the time on average, across all graph sizes. In our experiments, even when a community was not 100% intact, a relatively large section of the community remained intact. Also most of the communities broken up were the smallest ones. Thus the picture is even more optimistic when only larger communities are considered.

For the purpose of illustration in Figure 1, we chose the number of partitions $k$ to be the size of the graph divided by the size of the smallest graph (*i.e.* 1,000 nodes). Thus, a value of $k = 2$ was used for a graph containing 2,000 nodes. In practical applications, the value of $k$ can be selected based upon the largest graph that can be handled in a "reasonable" time. Another possibility is to recursively partition the graph until all the subgraphs are small enough for the community finding algorithm to handle. Although we tried different numbers of partitions for each of the graphs in our experiments, changing the value of $k$ did not appear to significantly impact upon the ability of the two-stage process to preserve communities intact. However, this behaviour may not necessarily be the case in general.

## 6 Conclusion

In this paper, we have proposed a two-stage problem decomposition strategy for SNA that facilitates the application of computationally expensive community-finding algorithms to much larger networks than would otherwise be possible if these algorithms were applied on their own. The two-stage strategy involves using a "top-down" graph partitioning technique to divide the network into smaller subnetworks, on which it is then feasible to apply a more computationally

intensive community-finding algorithm. We have demonstrated the usefulness of this strategy in empirical evaluations on both artificial and real-world datasets, where the computational cost of the network analysis process was significantly reduced without adversely affecting the quality of the communities that were discovered.

## References


1. Paliouras, G., Papatheodorou, C., Karkaletsis, V., Spyropoulos, C.D.: Clustering the users of large web sites into communities. In: Proc. 7th International Conference on Machine Learning (ICML'00). (2000) 719–726
2. Srivastava, J., Cooley, R., Deshpande, M., Tan, P.: Web usage mining: discovery and applications of usage patterns from Web data. ACM SIGKDD Explorations Newsletter **1** (2000) 12–23
3. He, Y., Cheung Hui, S.: Mining a Web Citation Database for author co-citation analysis. Information Processing and Management **38** (2002) 491–508
4. Newman, M.E.J., Girvan, M.: Finding and evaluating community structure in networks. Physical Review E **69** (2004)
5. Palla, G., Derenyi, I., Farkas, I., Vicsek, T.: Uncovering the overlapping community structure of complex networks in nature and society. Nature **435** (2005) 814–8
6. Gregory, S.: An algorithm to find overlapping community structure in networks. In: Proc. 11th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'07). (2007) 91–102
7. Abello, J., Pardalos, P.M., Resende, M.G.C.: On maximum clique problems in very large graphs. In: External Memory Algorithms. DIMACS Series in Discrete Mathematics and Theoretical Computer Science (1999) 119–130
8. Dhillon, I., Guan, Y., Kulis, B.: Weighted graph cuts without eigenvectors a multilevel approach. IEEE Transactions on Pattern Analysis and Machine Intelligence **29** (2007) 1944–1957
9. Narasimhamurthy, A., Greene, D., Hurley, N., Cunningham, P.: Community finding in large social networks through problem decomposition. Technical Report UCD-CSI-2008-04, University College Dublin, School of Computer Science and Informatics (2008) `http://csiweb.ucd.ie/Research/TechnicalReports.html`.
10. Freeman, L.: Centrality in social networks: Conceptual clarification. Social Networks **1** (1979) 215–239
11. Pothen, A., Simon, H.D., Liou, K.P.: Partitioning sparse matrices with eigenvectors of graphs. SIAM J. Mathematical Analysis and Applications **11** (1990) 430–452
12. Chan, P., Schlag, M., Zien, J.: Spectral k-way ratio cut partitioning. IEEE Transactions CAD-Integrated Circuits and Systems **13** (1994) 1088–1096
13. Shi, J., Malik, J.: Normalized cuts and image segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **22** (2000) 888–905
14. Karypis, G., Kumar, V.: A fast and high quality multilevel scheme for partitioning irregular graphs. SIAM J. Sci. Comput. **20** (1999) 359–392
15. Kernighan, B.W., Lin, S.: An efficient heuristic procedure for partitioning graphs. The Bell System Technical Journal **49** (1970) 291–307
16. McCallum, A., Nigam, K., Rennie, J., Seymore, K.: Automating the construction of internet portals with machine learning. Information Retrieval Journal **3** (2000) 127–163