A Description of the CIDR System as Used for TDT-2

Dragomir R. Radev[*] *Vasileios Hatzivassiloglou* *Kathleen R. McKeown*

Department of Computer Science
450 CS Building
Columbia University
1214 Amsterdam Avenue, MC 0401
New York, NY 10027

{radev,vh,kathy}@cs.columbia.edu

ABSTRACT

We describe several experimental parameters and a parallelization technique used in our online document clustering system, CIDR. These modifications were introduced into CIDR to reduce the running time so that incoming documents be clustered in almost real time. We discuss how several of these parameters are justified on linguistic grounds and report preliminary quantitative results on the effects that these parameters have on speed and accuracy.

1. Introduction

We report our experience with the development and testing of CIDR, a system for the automated placement of text documents into topical clusters. Our focus in CIDR is somewhat unusual. We have started from the assumption that our clustering system should aim for maximal efficiency, so that it will be able to classify tens of thousands of documents in real time. This puts a premium on operational speed rather than classification accuracy, and raises a number of interesting research questions, namely, what modifications to a standard document clustering approach offer significant speedup gains and what penalty in classification accuracy each of them incurs. We offer preliminary answers to these questions in this paper.

Our development of CIDR was constrained by limited availability of time and manpower resources. We did not participate in the pilot TDT study in 1997, and did not become officially involved in TDT-2 until November 1998. As a result, we were not able to participate in the formative TDT workshops, and we submitted our first results on the development subset of the TDT corpus only days before we submitted the official evaluation run. Unlike other competitors, CIDR is the product of essentially a single implementor, spanning a development period of six person-weeks. Thus, we view our current system as less of a finished competitive product and more of a prototype that embodies a basic clustering technique and allows us to explore modifications to it in order to investigate speed-accuracy tradeoffs. We have designed CIDR in a modular way that facilitates the replacement of components for rapid prototyping. We are particularly interested in augmenting CIDR with language-informed techniques and knowledge sources, a topic we return to in the last section of this paper.

2. Experimental Parameters and System Description

CIDR uses at its core a fairly standard single-pass clustering algorithm [1] with a deferral of zero. The innovative elements of our work lie in modifications to that algorithm that offer significant speed gains without sacrificing much accuracy. We describe first the general algorithm, and then discuss five experimental parameters and other modifications that were introduced for the purpose of reducing running time.

Our algorithm takes one article at a time and assigns it to the cluster that looks most similar to it. Matching between a new document and existing clusters is based on a comparison between the new document and the centroids of the existing clusters. Each document is represented as a vector of word frequencies modified by inverse document frequencies (a TF*IDF product, as is standard in information retrieval). Centroids of clusters are represented in a similar manner but the TF*IDF values associated with them are the weighted averages of the corresponding TF*IDF values of the documents already assigned to that cluster. The algorithm initially places the first document by itself in the first cluster, and this single cluster makes the initial working collection of clusters. As new documents are processed, they are compared to the centroids of the clusters in the working collection and are either placed in the most similar existing cluster or in a newly created cluster, consisting of just one new document. Similarity between a document and a centroid is measured by the cosine (normalized inner product) of the corresponding TF*IDF vectors, and a predetermined cutoff threshold specifies when the similarity is unacceptably low and a new cluster should be created instead.

In order to satisfy the on-line restriction, we estimate the inverse document frequencies from a separate collection, rather than the articles we are clustering. We use the documents in the TDT collection between January and April 1998 for this purpose. In addition, we introduce the following modifications to the algorithm:

- We ignore all but the first DECAY_THRESHOLD (typically 50-200) words in input documents. This speeds up the construction of the TF*IDF vectors for documents, and also hopefully focuses the comparisons to the most important words for each document. Earlier summarization research (see for example [2]) indicates that the first paragraph of a document typically contains the most salient points, at least for news articles.
- We ignore any words in the documents with inverse document frequencies (IDF) less than IDF_THRESHOLD (typi-

[*] The author's current address is IBM T. J. Watson Research Center, 30 Saw Mill River Road, Hawthorne, NY 10532.

cally around 3), since such words are not likely to affect the comparisons significantly. This significantly reduces the size of the vector representations for articles and clusters.

- In order to speed up the comparisons between cluster centroids and documents, we only keep the most important words for each centroid. This is accomplished by imposing a maximum number of words for each centroid (KEEP_WORDS); the words with the highest TF*IDF values are selected for this set. A second experimental parameter, KEEP_THRESHOLD, selects additional words that are included in the centroid, on top of the fixed number specified by KEEP_WORDS, if their individual TF*IDF values meet or exceed that threshold. Typical values are 10–20 for KEEP_WORDS and around 3 for KEEP_THRESHOLD, although experiments have indicated that many times even three or four words are sufficient to accurately describe the cluster. Three example cluster centroids are shown below, demonstrating that ten (and sometimes three) words give a clear picture of what each cluster is about:
  - CLUSTER00001 (90 documents): [grand 1.16, jury 1.07, whitewater 1.00, mcdougal 0.76, susan 0.43, testify 0.38, privilege 0.26, contempt 0.16, ewing 0.13, smaltz 0.07]
  - CLUSTER00008 (113 documents): [space 1.98, shuttle 1.17, station 0.75, nasa 0.51, columbia 0.37, mission 0.33, mir 0.30, astronauts 0.14, steering 0.11, safely 0.04]
  - CLUSTER00026 (10 documents): [universe 1.50, expansion 1.00, bang 0.90]
- In addition to the parameters mentioned above, a fifth parameter, SIM_THRESHOLD, controls when a new cluster is created. Low values for this parameter result in a more fine-grained separation of the input documents. We have found that a value around 0.1 offers a good compromise between precision and recall (or false positives and misses).

3. Parallelized Version

We have also experimented with a quasi-parallel version of the sequential algorithm described above. In this parallel model, we make a distinction between a designated ”main processor” and k ”auxiliary processors”. The main processor takes the first m articles (typically one quarter to one third of the total) and clusters them using the techniques of the previous section. In this way, centroids of C(m) clusters are established. The main idea behind parallelization is that these profiles do not change significantly as new articles are added to the clusters, although some new clusters will be formed. For example, Figures 1 and 2 show two sample cluster centroids after 10,000 documents are processed, and after all 22,443 documents are processed, indicating little change to the centroids. The auxiliary processors use this fixed collection of C(m) clusters as their set of working clusters, and either assign the remaining documents to these clusters or set them aside as too different from the existing clusters. In a final pass, the main processor reassigns the documents that have been set aside during the parallel phase, creating new clusters as needed. To observe the on-line restriction, auxiliary processors do not actually assign documents to clusters or modify centroids. They only store their assignment recommendations in a “hint” file. The documents are assigned to clusters, in sequential order, by the main processor during the final phase.

word	score
suharto	2.48
jakarta	0.58
habibie	0.47
students	0.45
student	0.22
protesters	0.20
asean	0.11
campuses	0.05
geertz	0.04
medan	0.04

word	score
suharto	2.61
jakarta	0.58
habibie	0.53
students	0.43
student	0.21
protesters	0.19
asean	0.10
campuses	0.04
geertz	0.04
medan	0.04

Figure 1: Centroid for cluster 44 (the two scores are after 10,000 (left) and all 22,443 documents (right).

word	score
microsoft	3.31
justice	1.06
department	1.01
windows	0.90
corp	0.60
software	0.51
ellison	0.09
hatch	0.06
netscape	0.05
metcalfe	0.03

word	score
microsoft	3.24
justice	0.93
department	0.88
windows	0.98
corp	0.61
software	0.57
ellison	0.07
hatch	0.06
netscape	0.04
metcalfe	0.03

Figure 2: Centroid for cluster 62 (the two scores are after 10,000 (left) and all 22,443 documents (right).

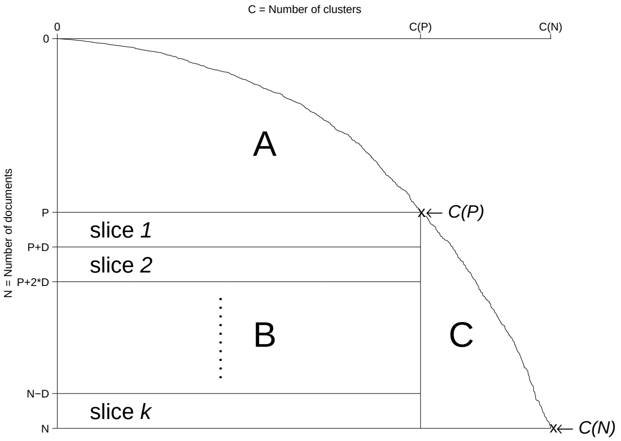

Figure 3: Parallelization diagram. The two axes are not drawn to the same scale, but the area under the curve and the positions of labeled points on the two axes correspond to real data.

Figure 3 indicates the gains in run-time when running the parallel version of the algorithm. The main processor operates on the curve $C(N)$. It starts at the origin and moves down and to the right. The figure is drawn in a way such that the area under the curve is equal to the number of comparisons between documents and cluster centroids. At the beginning, there are only a few clusters, so fewer comparisons are required to classify a document. Towards the end of the run, many clusters exist and at that stage, more comparisons are required. With no parallelization, the main processor would have to take time proportional to the total area under the curve in Figure 3.

We introduce a number of documents, P, chosen in a way such that a reasonable number of clusters, $C(P)$, have already been created. The role of the auxiliary processors is to compare documents numbered $P + 1$ and higher with all clusters in the range $1 \dots C(P)$. When this task is completed, the main processor re-examines these assignments. For each document d ($d > P$), the main processor checks whether the similarity between document d and its assigned cluster is still above the established threshold. If it is, document d is immediately added to cluster C_d. If, however, the centroid for cluster C_d has changed in such a way that the similarity between d and C_d is now below the threshold, the main processor compares d against all existing clusters before making the final placement decision.

Clustering, and especially on-line clustering, is an inherently non-parallel operation, so making the decisions in parallel is bound to introduce some additional errors. By analyzing the results of this process, we have found that 78% of the cluster assignment decisions made during the parallel phase are the same as if the full sequential model were employed, and that only 5% of the documents examined by the auxiliary processors are set aside for the final phase. With ten auxiliary processors, the parallelized version is at least three times faster in our experiments. This comes at a cost of a potentially 22% higher error rate, which can be a defensible compromise.

4. Results

To select the values of the input parameters, we ran two experiments using parts of the development (dev-test) corpus (we chose the 3,851 articles included in the sample index file from the "examples" subdirectory of the TDT distribution). We ran our system on this small corpus on up to 13 machines at the same time, then we measured its performance using the master tables included on the TDT CD-ROM. In this way, we selected our first approximation of the best values for the four parameters SIM_THRESHOLD, DECAY_THRESHOLD, IDF_THRESHOLD, and KEEP_WORDS. Parameter KEEP_THRESHOLD was set equal to IDF_THRESHOLD in our experiments. Table 1 lists several choices of parameters and the corresponding scores obtained during this training phase.

This experiment led to the selection of the following combination of parameters, which we used for the official submission: SIM_THRESHOLD = 0.1, DECAY_THRESHOLD = 100 words, IDF_THRESHOLD = 3, and KEEP_WORDS = 10. They gave a story-weighted average detection cost of 0.0089 on our internal set (without parallelization), and, as we found later, 0.0095 (using parallelization) and 0.0077 (without parallelization) on the official test set.

After we sent out the official submission, we continued experimenting with the input parameters, and we selected two more parameter sets. Our later experiments offer a significant improvement on average cost detection (0.0051–0.0068, measured on our internal test set), as we were able to better tune the clustering parameters. We submitted the results from these two selected combinations of parameters, produced with the non-parallel version of our algorithm, as contrasting submissions. We also submitted as a contrasting submission the results obtained when our initial set of parameters (used for the official results) was run with the non-parallel version of the algorithm (the parallel version takes 27 hours on the evaluation corpus, versus 73 hours for the non-parallel version, on a Sparc Ultra 2/300). Table 2 summarizes our submitted runs, while Figure 4 shows the miss–false alarm rate graph for one of these runs.

SIM	DECAY	IDF	KEEP	Story Weighted		
				P(miss)	P(fa)	C_{detect}
0.01	100	3	10	0.9643	0.0038	0.0230
0.03	100	3	10	0.8214	0.0063	0.0226
0.02	100	3	20	0.8571	0.0038	0.0208
0.02	1000	3	10	0.9286	0.0017	0.0202
0.02	100	8	30	0.9643	0.0008	0.0201
0.02	100	3	8	0.9643	0.0006	0.0199
0.02	50	3	10	0.6071	0.0044	0.0164
0.02	100	2	10	0.6800	0.0026	0.0162
0.02	100	3	10	0.4900	0.0061	0.0158
0.02	100	1	50	0.4100	0.0054	0.0135
0.05	100	3	5	0.4286	0.0048	0.0133
0.05	100	3	15	0.5357	0.0025	0.0132
0.05	100	4	10	0.5000	0.0031	0.0131
0.02	100	3	1	0.3600	0.0048	0.0119
0.02	100	8	10	0.4500	0.0023	0.0113
0.25	100	3	10	0.5357	0.0004	0.0111
0.20	100	4	10	0.5307	0.0004	0.0111
0.06	100	3	10	0.3929	0.0027	0.0105
0.05	100	3	10	0.4000	0.0025	0.0105
0.05	100	3	30	0.3200	0.0040	0.0103
0.09	100	3	10	0.4000	0.0014	0.0094
0.15	100	3	10	0.4286	0.0006	0.0092
0.10	100	3	5	0.4000	0.0011	0.0091
0.20	100	3	10	0.4286	0.0004	0.0090
0.13	100	3	10	0.4286	0.0004	0.0090
0.10	**100**	**3**	**10**	**0.3800**	**0.0014**	**0.0090**
0.10	100	3	15	0.3800	0.0010	0.0086
0.14	100	3	10	0.4000	0.0001	0.0081
0.10	100	4	10	0.3600	0.0009	0.0081
0.10	67	3	10	0.3100	0.0018	0.0080
0.12	100	3	10	0.3500	0.0007	0.0077
0.11	100	3	10	0.3500	0.0007	0.0077
0.08	100	3	10	0.3100	0.0014	0.0076
0.10	200	3	10	0.2500	0.0024	0.0073
0.10	100	3	20	0.3100	0.0010	0.0072
0.10	**50**	**3**	**10**	**0.2400**	**0.0021**	**0.0068**
0.10	**100**	**2**	**10**	**0.1900**	**0.0013**	**0.0051**

Table 1: Parameters (KEEP refers to KEEP_WORDS) and corresponding scores for several training runs, listed in increasing order of performance. Lines in bold indicate runs that are included in the official evaluation.

5. Conclusion and Future Directions

We have explored several ways to cut down the running time of text clustering algorithms without a disproportionate penalty on the accuracy of cluster assignments. Some of these not only offer a speed benefit but also help focus the similarity measure to the most important part of a document or the core elements of a cluster, thus improving performance over the unmodified version of the same clustering algorithm.

CIDR was inspired by our work on document grouping and summarization within an NSF STIMULATE grant. A major characteristic of our approach on that task is to complement information retrieval techniques with shallow text analysis, so that the former are informed by linguistic knowledge. We intend to introduce such linguistic elements into the comparisons performed by CIDR, and we have started doing so by already weighing likely proper nouns (cap-

Run	Parallelization?	SIM	DECAY	IDF	KEEP	Story Weighted			Topic Weighted		
						P(miss)	P(fa)	C_{detect}	P(miss)	P(fa)	C_{detect}
1	yes	0.1	100	3	10	0.3861	0.0018	0.0095	0.3309	0.0018	0.0084
2	no	0.1	100	3	10	0.3164	0.0014	0.0077	0.3139	0.0014	0.0077
3	no	0.1	100	2	10	0.3178	0.0014	0.0077	0.2905	0.0014	0.0072
4	no	0.1	200	3	10	0.5045	0.0014	0.0114	0.3201	0.0014	0.0077

Table 2: Official evaluation of CIDR.

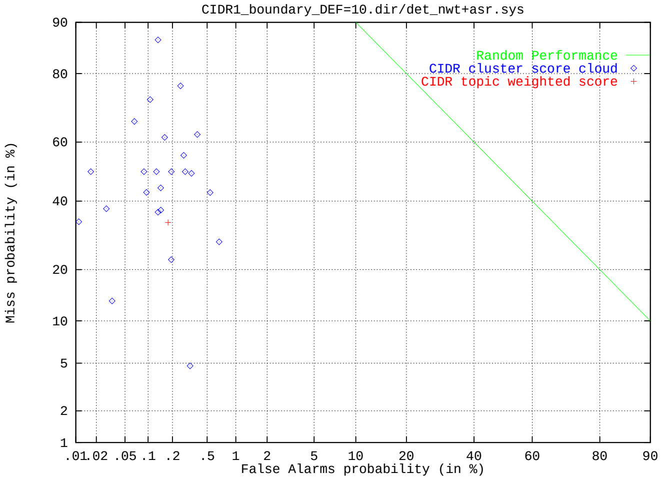

Figure 4: Scatterplot of miss rate versus false alarm rate for one variant of CIDR on the official test set. The corresponding parameters for this run were SIM_THRESHOLD=0.1, DECAY_THRESHOLD=100, IDF_THRESHOLD=KEEP_THRESHOLD=3, KEEP_WORDS=10, no parallelization.

italized words within a context of mixed case text) twice as much as their TF*IDF value would indicate. We will extend this approach by giving privileged status to elements of a document such as location, time period, and major named participants. We will also explore the possible use of external knowledge sources, such as the CIA FactBook and the database of named entities collected by our Profile tool [3], to automatically identify the elements within each event that should be given priority during matching.

We are also interested in exploring cluster centroid stability over the course of an event and use it for detecting sub-events. This idea is related to our work on summary generation from multiple articles.

Acknowledgements

This material is partly based upon work supported by the National Science Foundation under Grants No. IRI-96-19124 and IRI-96-18797. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

The authors are grateful to Luis Gravano for fruitful discussions and to Jonathan Fiscus for all the help during the evaluation.

References

1. William B. Frakes and Ricardo Baeza-Yates. *Information Retrieval: Data Structures and Algorithms*. Prentice Hall, Englewood Cliffs, New Jersey, 1992.
2. Chin-Yew Lin and Eduard Hovy. “Identifying Topics by Position”. In *Proceedings of the 5th ACL Conference on Applied Natural Language Processing*, pp. 283–290, Washington, D.C., April 1997.
3. Dragomir R. Radev. “Learning Correlations Between Linguistic Indicators and Semantic Constraints: Reuse of Context-Dependent Descriptions of Entities”. In *Proceedings of the Joint 17th International Conference on Computational Linguistics and 36th Annual Meeting of the Association for Computational Linguistics (COLING-ACL’98)*, Montreal, Canada, August 1998.
4. Dragomir R. Radev and Kathleen R. McKeown. “Generating natural language summaries from multiple on-line sources”. In *Computational Linguistics*, 24(3):469–500, September.