Unsupervised document clustering with ART

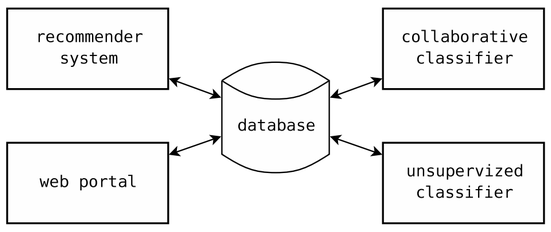
Ciprian D. Craciun

Faculty of Mathematics and Informatics, West University of Timisoara, Romania

ccraciun@info.uvt.ro

Scientific document recommender

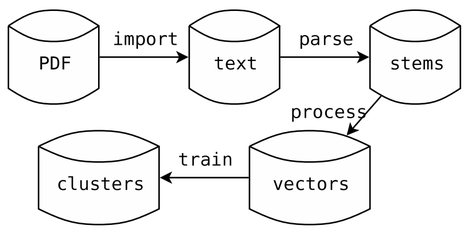
Our goal is to build a recommender system for scientific documents, based both on user expertise and / or automatic classification, that will aid and guide researches to find more easily relevant documents for their work. [5]



Automatic document classifier

The implementation started with the **prototype** of the automatic classifier, with the following simple architecture:

- a couple of tools that communicate with each other only by modifying some shared data structures, based on a workflow;
- data structures are simple BerkeleyDB tables (key – value maps);
- the tools were written in C (the heavy duty and processor intensive tools) and Python (the classification algorithms);



Document preprocessing

- obtaining only text (ASCII) content for documents;
- **stemming** using the Porter stemmer for English;
- eliminating English (stemmed) **stop words**;
- computing the frequency for each stem in each document;
- **rejecting** stems which appear in more than 1/200 or less than 1/5 documents; (these values were determined by empirical methods;)
- computing the total frequency for each stem in the entire data set;
- computing the feature vector:

$$x_i = \frac{dsf_i}{\sum_j dsf_j} * log(\frac{n}{gsf_i})$$

- **normalizing** the feature vector;

Unsupervised clustering with ART

Based on the classic version of the **ART** neural network [1]:

- parametrized with **vigilance**, **learning rate** and number of runs;
- for each document we find the prototype with the best match above the given vigilance, adjust it, and normalize it;
- if the best match is below the given vigilance we create a new prototype based on the document vector;
- we execute the previous steps for each round;
- the match and activation functions are based on **cosine**;

Adjusting functions:

- **variant A** – classic:

$$w_i = \frac{max(w_i + l * (x_i - w_i), 0)}{|w|}$$

- **variant B** – classic, usually used for spatial data sets:

$$w_i = \frac{(1 - l) * w_i + l * min(w_i, x_i)}{|w|}$$

- **variant C** – customized, it takes into consideration the **similarity** between the cluster and the document:

$$w_i = \frac{max(w_i + l * e^{-|w-x|^2} * d(x_i, w_i), 0)}{|w|}$$

Where:

- w – cluster prototype;
- x – document feature vector;
- l – learning rate;
- $c(x, y)$ – cosine similarity, $c(x, y) = (x \cdot y)/(|x| * |y|)$;
- $d(x, y)$ – cosine dissimilarity, $d(x, y) = 1 - c(x, y)$;

Clustering with K-means

Classic version of **K-means**:

- parametrized with the **number of prototypes**, and minimum adjustment;
- for each document we find the closest matching prototype, and we add the document vector to its corresponding list;
- for each category we take the vector list and compute the average, which is going to be used as the new prototype vector (of course normalized);
- we sum differences between old an new vectors for each prototype, and if the value is above the minimum adjustment we re-execute the previous steps;

Clustering validation

Validation indices:

- **external** – based on an apriori partitioning of the documents – usable only for training and testing data sets; [2]
- **internal** – based only on the data set (the vectors) itself and does not need any other inputs – usable for any data set (for example real data sets); [2]

External validation indices

As described in [2] (the closer to 1 the better):

- **Rand statistic** – $R = (SS + DD)/M$;
- **Jaccard coefficient** – $J = SS/(SS + SD + DS)$;
- **Folkes and Mallows index**: $FM = SS/\sqrt{(SS + SD) * (SS + DS)}$;

Where:

- SS – the number of document pairs that belong to the same cluster and partition;
- SD – same cluster but different partitions;
- DS – different clusters but the same partition;
- DD – different clusters and different partitions;
- $M = SS + SD + DS + DD = N * (N - 1)/2$;
- N – the number of documents;

And as described in [4]:

- **entropy** (the closer to 0 the better):

$$E = \sum_{k=1..K} \frac{n_k}{N} * E_k \qquad E_k = -log(P) * \sum_{p=1..P} \frac{n_k^p}{n_k} * log(\frac{n_k^p}{n_k})$$

- **purity** (the closer to 1 the better):

$$P = \sum_{k=1..K} \frac{n_k}{N} * P_k \qquad P_k = 1/n_k * max_{p=1..P}(n_k^p)$$

Where:

- K – the number of clusters;
- P – the number of partitions known apriori;
- n_k – the number of documents that belong to the cluster k;
- n_k^p – the number of documents in the partition p that were assigned to the cluster k;

Internal validation indices

Based on PBM described in [3] (the greater the better):

$$PBM = (\frac{1}{K} * \frac{E_1}{E_K} * D_K)^2$$

Where:

- x – a document in the data set;
- $E_1 = \sum_{i=1..N} d(x_i, w_0)$
- $E_K = \sum_{i=1..N} \sum_{j=1..K} u(x_i, w_j) * d(x_i, w_j)^2$
- $D_K = max_{i,j=1..K} d(w_i, w_j)$
- w_0 – the geometric center of the data set vectors;
- d – a distance function;
- u – a similarity function;

Document classifier testing

Data sets

col.	data sets	documents	categories	features
col-a	Cisi+Cran+Med	3887	3	1590
col-b	News20	18846	20	2842

Testing results

Values for the validation indices:

test	K	R	J	FM	E	P
col-a-1	3	0.94	0.84	0.91	0.16	0.96
col-a-2	3	0.94	0.86	0.92	0.16	0.96
col-a-3	3	0.98	0.95	0.97	0.05	0.99
col-b-1	19	0.92	0.17	0.30	0.52	0.48
col-b-2	20	0.91	0.18	0.31	0.63	0.38
col-b-3	20	0.92	0.22	0.37	0.53	0.45

Parameters for ART:

test	variant	vigilance	learning	runs
col-a-1	A	0.02	0.2	3
col-a-2	C	0.02	1.0	3
col-b-1	A	0.02	0.2	3
col-b-2	C	0.02	1.0	3

Parameters for K-means:

test	min. adjustment
col-a-3	0.001
col-b-3	0.001

Conclusions

- the classic variant of ART is **comparable** with K-means;
- our own variant of ART shows a slight **improvement** over the classic version;
- col-a yields better results because the apriori partitions are well created;
- col-b yields worse results for all the tests as the partitions are not based on actual content, but on other factors (in this case mailing list);

Questions

- Which is the best solution for the document preprocessing part (stemming and feature vector building)?
- How can we combine the strengths of ART and K-means to obtain a **hybrid** clustering algorithm?
- If we would like to make a **fuzzy** classification, which would be the best algorithm? How can we adapt ART to this task?
- How can we obtain a **hierarchical** classification? How can we match this hierarchy with existing ontologies?
- How would we optimize the run-time by parallelizing the entire process?

References

[1] Andrea Baraldi and Ethem Alpaydin — Constructive Feedforward ART Clustering Networks Part I — 2002;

[2] Maria Halkidi, Yannis Batistakis, and Michalis Vazirgiannis — Cluster Validity Methods: Part I — 2002;

[3] Marta V. Modenesi, Myrian C. A. Costa, Alexandre G. Evsukoff, and Nelson F. F. Ebecken — Parallel Fuzzy c-Means Cluster Analysis — 2007;

[4] Ying Zhao and George Karypis — Criterion Functions for Document Clustering: Experiments and Analysis — 2002;

[5] http://web.info.uvt.ro/~ccraciun/mindsoft

Acknowledgment: This work was supported by RO-PNCD II projects NatComp and SIPADOC.