Trade-offs of Heuristic Vs. Rigorous Algorithms in Text Mining

Andrew Matheny

Masters Defense

Committee
Tim Menzies, Ph.D (chair)
Tim McGraw, Ph.D
James Mooney, Ph.D

West Virginia University, LCSEE

July 21, 2010

Outline

Abstract

Abstract

Within the field of text mining, rigorous methods for dimensionality reduction and clustering don't scale.

We examine the trade-offs between standard rigorous approaches and a few heuristic alternatives.

The findings indicate that heuristic approximations aren't so bad.

National Archival Needs

- Research arose out of NETL-funded look into archiving needs of NARA (National Archives and Records Administration)
- Vast amounts of textual documents with no infrastructure for exploring them

Navigating the Technical Document Space

Specifically, NARA needs to explore thousands of projects worth of technical documents.
To name a few...

- STEP/EXPRESS documents
- Source code
- Engineering artifacts (requirements, use cases, etc.)

$1M Question

What other designs are similar to this one?

But why?

- Estimated that over 80% of potentially usable business information in an unstructured form [Grimes, 2008]
- Heaps' Law tells us that the growth rate of the term space, or vocabulary, is approximately equivalent to the square root of the total number of words in the document set [Grootjen et al., 2003].

Outline

What is Text Mining?

Definition

The exploration of large libraries of natural language with the goal extracting information not previously known.

Example Applications

- Spam filtering
- Identifying similar news items
- Targeted advertising
- Program comprehension
- Search indexing
- etc...

Basic Process

The basic operation involved in text mining are:

- Defining the scope of a document
- Processing the documents into manageable data structures
- Computational analysis of document collection as a whole
- Output of new information

Explicit Process

A more descriptive explanation of these steps and how they are used in this study are as follows:

- **Preprocess:**
  Translate documents into a vector space model [Salton, 1991]
- **Reduce Dimensionality:**
  Remove non-descriptive terms and extract valuable terms
- **Cluster:**
  Group similar documents

Preprocessing Techniques

A few of the preprocessing techniques used in this study are:

- Tokenization
- Stop Lists
- Stemming
- Conversion to TF*IDF Space

Tokenization

Definition

The processes of dividing written text into meaningful units, such as words, sentences, or topics

In addition to dividing the text into words, some house keeping is often done in this step such as removing punctuation and removing any variation in case (i.e. sending all upper case to lower)

Stop Lists

Definition

A collection of words that provide no lexical context and can be removed to aid in classification

```
a about across again against almost alone along already also although always
am among amongst amongst amount an and another any anyhow anyone anything
anyway anywhere are around as at ... ... ... ... ...
```

Figure: 24 of the 262 stop words used in this study.

Stemming

Definition

The process for reducing inflected (or sometimes derived) words to their stem, base or root form. [Porter, 1980]

- CONNECT
- CONNECTED
- CONNECTING
- CONNECTION
- CONNECTIONS

Figure: Stemming example for the word *connect*

TF*IDF

A weight assigned to a given word in a given document [Jones, 1993]

Definition

Term frequency * inverse document frequency

Term Frequency

Count of occurrence of term t_i in document d_j

- $n_{i,j}$ = number of occurrences of term t_i in document d_j
- $\displaystyle\sum_k n_{k,j}$ = sum of the number of occurrences of all terms in document d_j

Term Frequency Equation

$$\mathrm{TF}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$$

Inverse Document Frequency

Normalized count of occurrence of term t_i in all documents

- D = set of all documents
- $\{d : t_i \epsilon d\}$ = set of documents that contain term t_i

Inverse Document Frequency Equation

$$\mathrm{IDF}_i = log \frac{|D|}{\{|d : t_i \epsilon d\}|}$$

TF*IDF

Term-Frequency Inverse-Document-Frequency

- $\mathrm{TF}_{i,j} = \frac{n_{i,j}}{\sum_k n_{k,j}}$
- $\mathrm{IDF}_i = log \frac{|D|}{\{|d:t_i \epsilon d\}|}$

TF*IDF Equation

$\mathrm{TF*IDF}_{t_i,d_j} = TF_{i,j} x IDF_i$

Preprocessing Example

An example of applying preprocessing to documents and achieving a vector space model of terms

Document$_j$	**TFIDF(t_i, d_j)**					
	t_1: woodchucks	t_2: chuck	t_3: lumberjacks	t_4: wood	t_5: chop	t_6: norris
Woodchucks can chuck wood	1	0.6	0	0.3	0	0
Lumberjacks can chop woodchucks	1	0	0.6	0	0.6	0
Chuck Norris can chop lumberjacks and woodchucks	1	0.6	0.6	0	0.6	0.3

What is Dimensionality Reduction?

Definition

The mapping of a feature space N, to a subspace $N_{\prime}$, with $|N_{\prime}| < |N|$

- The preprocessing steps mentioned above a preliminary form of dimensionality reduction
- Given the large amount of different terms in most document collections, almost any type of analysis is computationally impossible

What is Clustering?

Definition

Assignment of items in a set into subsets, where the items of each subset have similar attributes, based on a given criteria

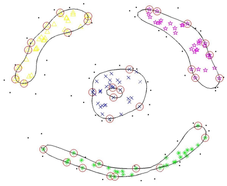


Figure: Clustering of points based on two dimensional distance

Outline

Principal Components Analysis

Involves a mathematical procedure which includes computing the eigenvalue decomposition of the covariance matrix (SVD) [Jolliffe, 2002]. The new space ranks the variation of projections and places them in the new coordinates in decending order.

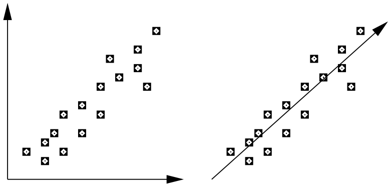


Figure: The two features in the left plot can be transferred to the right plot via one latent feature.

Principal Components Analysis

Assumptions

- The new coordinate system can be defined by a linear combination of existing features
- That mean and variance are stastically important
- That large variances have important dynamics

- Recently shown to provide the relaxed solution for K-means clustering [Ding and He, 2004]
- Run-time of $O(n^2)$ where n is the number of initial dimensions
- Does *not* scale

Fastmap

Performs a linear transformation of points based on the euclidian geometry of the initial space. [Faloutsos and Lin, 1995]

Algorithm

- $n = D$, the number of dimenions in the original space
- While $n > d$ (new dimensionality)
  - Uses heuristic to find an aproximation of the two points which lie the furthest from each other in constant time. These are the pivot elements
  - Uses the consine law to project all other points onto the plane that lies perpendicular to the line between the two pivot elements
  - Decrement n

Fastmap

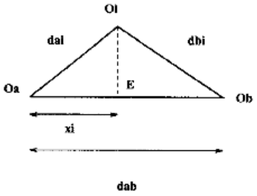


Figure: Example of using the cosine law to find the position of *Oi* in the dimension *k*

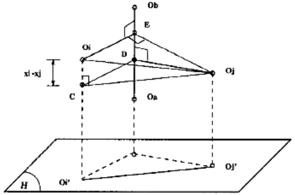


Figure: Projects of points O_i and O_j onto the hyper-plane perpendicular to the line $O_a O_b$

TF*IDF Sort

A simple linear time method of determining the most useful terms in a document collection. First reported by [Ramos, 2003]

Algorithm

```
• for each term, $t_i$
    • Compute tfidfSum $= \sum_{d \epsilon D} TFIDF(t_i, d)$
• sort terms by tfidfSum in decending order
• return top $n$ terms
```

TF*IDF Sort: Example

Document$_j$	**TFIDF**(t_i, d_j)					
	t_1: woodchucks	t_2: chuck	t_3: lumberjacks	t_4: wood	t_5: chop	t_6: norris
Woodchucks can chuck wood	1	0.6	0	0.3	0	0
Lumberjacks can chop woodchucks	1	0	0.6	0	0.6	0
Chuck Norris can chop lumberjacks and woodchucks	1	0.6	0.6	0	0.6	0.3

Top 4 terms as determined by TF*IDF Sort:

- woodchucks
- chuck
- lumberjacks
- chop

K-means

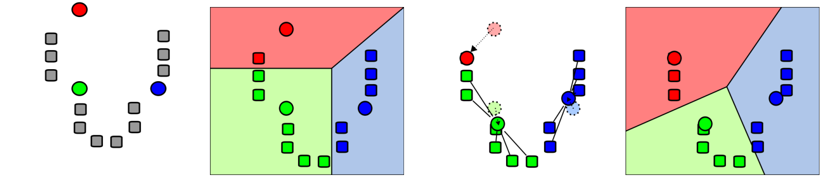
Algorithm

```
• initialize i to 0
• randomly select $k$ initial cluster centroids
• while ($i \leq maxIterations$ or no point changes set membership)
    • assignment: assign each point to the cluster with the nearest centroid
    • update: recompute centroids using the mean of all items in the cluster
    • increment $i$
```

- An iterative refinement clustering algorithm
- With each iteration of the algorithm, cluster centroids are updated according to the mean of the items in the cluster
- Iteration continues until certain criteria is reached
- Variation on expectation-maximimization algorithm
- Closely related to PCA as recently show by [Ding and He, 2004]

[Kanungo et al., 2000]

K-means: Illustrated



(a) Randomly select cluster centroids

(b) Assign each object to it's nearest centroid

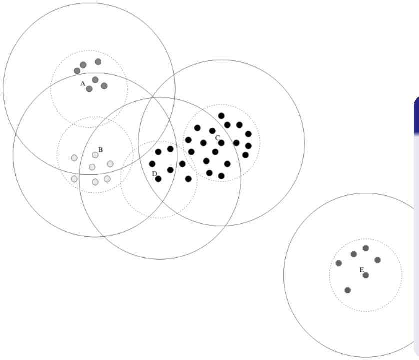
(c) Compute new centroids by using the mean value from the centroids picked in step

(d) Repeat steps *b and* c until the convergence criteria has been reached

Canopy

- Often used as a preprocessor for K-means [McCallum et al., 2000]
- Uses a cheap distance metric to build canopies
- Canopies are then used in clustering to reduce the number of times the expensive distance metric is used
- Cheap distance metric is highly dependant on the domain

Canopy



Canopy Clustering: The darker circle represents all points in a given canopy ($distance(p, p_{\prime}) < T1$). Points in the smaller circles cannot be used as a new canopy center ($distance(p, p_{\prime}) < T2$).

Canopy Creation Algorithm

```
while $|P| > 0$
    pick a point, $p$, at random and create a cluster here
    remove $p$ from the set $P$
    foreach $p_{\prime} \epsilon P$
        if $distance(p, p_{\prime}) < T1$
        then; add $p_{\prime}$ to the cluster
        if $distance(p, p_{\prime}) < T2$
        then; remove $p_{\prime}$ from the set $P$
```

Genic

- Generalized incremental clustering algorithm developed by [Gupta and Grossman, 2004]
- Only requires single pass
- Highly scalable

Genic Parameters

- k: number of initial
- m: number of initial candidate centers
- n: size of a generation

Genic: Algorithm

Algorithm

- **Initialization**
  - Select m points, $c_1, \ldots, c_m$ to be the initial candidate centers.
  - Assign a weight of $w_i = 1$ to each of these candidate centers.
- **Incremental Clustering** For each subsequent data point p in the stream: do
  - $Count = Count + 1$
  - Find the nearest candidate center c_i to the point p
  - Move the nearest candidate center using $c_i = \frac{(w_i * c_i + p)}{w_i + 1}$
  - Increment the corresponding weight $w_i = w_i + 1$
  - When $Count \ mod \ n = 0$, go to last bullet
- **Generational Update of Candidate Centers**

  When $Count$ equals $n, 2n, 3n, \ldots$, for every center c_i in the list L of centers, do:
  - Calculate its probability of survival, $p_i = \frac{w_i}{\sum_{i=1}^{n} w_i}$
  - Select a random number δ uniformly from [0,1].
  - If $p_i > \delta$, retain the center c_i in the list L of centers
  - Set the weight $w_i = 1$ back to one.
  - Go back to second bullet and continue processing the input stream
- **Calculate Final Clusters** The list L contains the m centers. These m centers can be grouped into the final k centers based on their Euclidean distances.

Outline

Experimental Design

Factorial or fully-crossed design (a 8x3x3x6x6 factorial design) with a total of 2,592 (8x3x3x6x6) different treatments

Experimental Factors

Factor	**Levels**	**Level count**
dataset	ap203, ap214, BBC, BBCSport, ngBias3, ngBias8, ngBal3, ngBal8	8
clusterer	KMeans, Canopy, Genic	3
reduction method	PCA, FastMap, TF-IDF Ranking	3
k	3, 5, 8, 15, 40, 75	6
d	3, 15, 25, 50, 100, 200	6

Classification of Algorithms

	Clustering	**Dimensionality Reduction**
Exhaustive	K-Means	PCA
Heuristic	Canopy Genic	FastMap TF-IDF Ranking

Dataset Statistics

Documents	AP203	AP214	BBC	BBCSport	ngBias3	ngBias8	ngBal3	ngBal8
D documents	484	1373	2224	737	1500	2395	1499	3999
T terms	1103	3050	9635	4613	8631	9826	8158	14984
Mean document length	143	164	130	104	116	87	91	90
Natural Classes	N/A	N/A	5	5	3	8	3	8
Stemming used	-	-	x	x	x	x	x	x
Stop-words removed	-	-	x	x	x	x	x	x

Table: Statistics on our datasets.

Natural Classes of Supervised Datasets

Dataset	Natural classes
BBC	business(509), entertainment(386), politics(417), sport(511), tech(302)
BBCSport	athletics(100), cricket(124), football(265), rugby(147), tennis(57)
ngBias3	graphics(136), hockey(591), windows(587)
ngBias8	atheism(704), autos(202), crypt(204), hockey(205), mac(202), mideast(202), space(204), xwindows(205)
ngBal3	graphics(463), hockey(459), mideast(453)
ngBal8	atheism(471), autos(463), crypt(467), hockey(461), mac(466), mideast(470), space(461), xwindows(469)

Table: Supervised datasets along with class values.

BBCSports & BBC datasets found at [at UCD, 2009]
ngBias3, ngBias8, ngBal3, & ngBal8 datasets found at [Lang, 1995], [Bay et al., 2000]

Assessment Criteria

MWU tests, AUC plots, and relative AUC sums of the following data items.

- Run-time
- Cluster Purity
- Internal Cluster Similarity
- External Cluster Similarity
- Similarity Loss

Cluster Purity

- A method for ranking cluster heterogeneity
- Answers the question: How dominant is the the most frequently occurring class in a cluster?
- Can only be used on supervised datasets (No STEP/EXPRESS)

Purity Equations

$$purity(C_j) = \frac{1}{|C_j|} max_i(|C_j|_{class=i})$$
$$purity = \sum_{j=1}^{k} \frac{|C_j|}{|D|} purity(C_j)$$

Internal Similarity

- Measure of how similar the things within the same class are in terms of geometric distance
- Can be used with supervised and unsupervised datasets
- Closely related to external similarity

Internal Similarity Equation

$$iSim_i = \sum_{d \in C_i} \sum_{d' \in C_i} \frac{cos(d, d')}{n_i^2}$$

$$iSim = \sum_i \frac{n_i}{N} iSim_i$$

External Similarity

- Measure of how different the things within the same class are in terms of geometric distance
- Can be used with supervised and unsupervised datasets
- Closely related to internal similarity

External Similarity Equation

$$eSim_{ij} = \sum_{d \in C_i} \sum_{d' \in C_j} \frac{cos(d, d')}{n_i n_j}$$

$$eSim = \sum_{i} \frac{n_i}{N} eSim_{ij}$$

Similarity Loss

Measure of difference between internal and external similarity

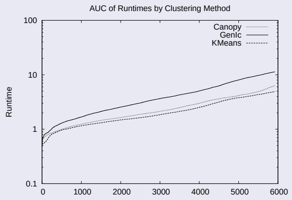
Similarity Loss Equation

$$\mathbf{SimilarityLoss} = eSim - iSim$$

AUC Plots

- Area under the curve (AUC)
- Used to display the total scope of values received for a given treatment.
- Plotted by sorting all results for a treatment and plotting this as the y value while the x axis is incremented by one for each value
- Useful for helping to explain disparities between MWU and other results

Sample AUC Plot



Runtimes

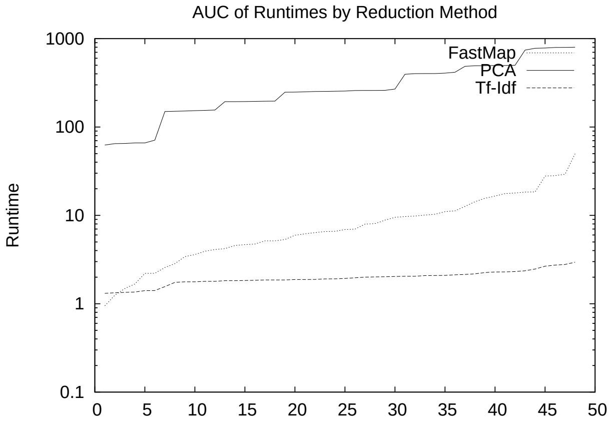
	Run-time MWU results			
Reduction Method	**ties**	**wins**	**losses**	**wins-losses**
tfidf	0	2	0	2
FastMap	0	1	1	0
PCA	0	0	2	-2

As expected, the quick linear time method *tfidf* performs better than both *PCA* and *FastMap* while *PCA* performs the worst.

Reduction Method	**Run-time AUC Relative to PCA**
tfidf	<1
FastMap	3
PCA	100

Notice the massive computational requirements of PCA compared to FastMap and tfidf. tfidf runs in less than 1% of the time of PCA.

Runtime AUC Plot



Purity

Reduction Method	Purity MWU results			
	ties	wins	losses	wins-losses
PCA	0	2	0	2
TfIdf-Sort	0	1	1	0
Fastmap	0	0	2	-2

Reduction Method	Purity AUC Relative to PCA
pca	100
fastmap	83
tfidf	83

As expected, *PCA* wins with purity though *FastMap* and *tfidf* are not far behind.

External Similarity

Reduction Method	External Sim MWU results			
	ties	wins	losses	wins-losses
pca	0	2	0	2
tfidf-sort	1	0	1	-1
fastmap	1	0	1	-1

Reduction Method	External Sim AUC Relative to PCA
fastmap	102
pca	100
tfidf	81

For external similarity, lower is better. Surprisingly, tfidf is coming out on top of PCA.

Internal Similarity

	Internal Sim MWU results			
Reduction Method	**ties**	**wins**	**losses**	**wins-losses**
pca	0	2	0	2
fastmap	0	1	1	0
tfidf-sort	0	0	2	-2

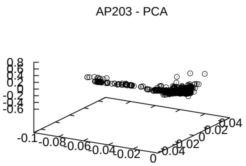
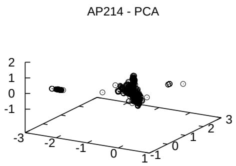
Reduction Method	**Internal Sim AUC Relative to PCA**
fastmap	118
pca	100
tfidf	88

Again with internal similarity, we have an example of a heuristic algorithm, *fastmap*, outperforming our rigorous baseline, *PCA*

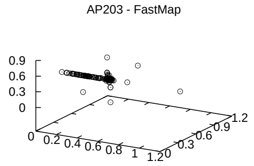
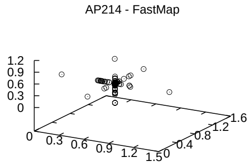
Similarity Loss

Reduction Method	Similarity Loss
tfidf	6
fastmap	30

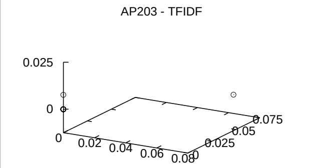
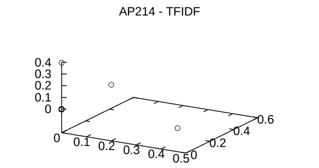
Visualizing the Reduction: PCA

PCA $(D - T^2)$		

Visualizing the Reduction: Fastmap

FASTMAP $(2N)$		

Visualizing the Reduction: TF*IDF

Tf-IDF $(t - N)$		

Runtimes

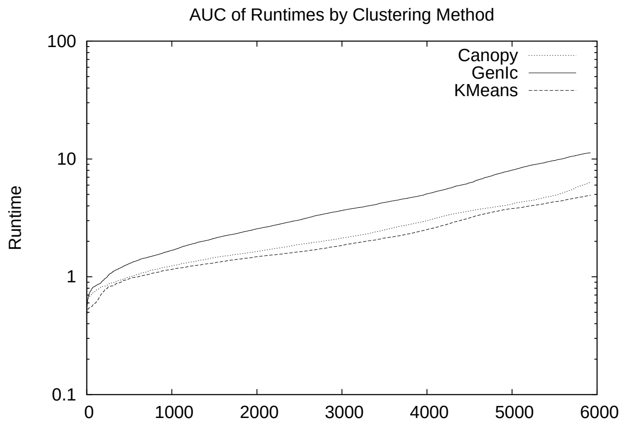
Clustering Method	Run-time MWU results			
	ties	wins	losses	wins-losses
genic	0	2	0	2
kmeans	1	0	1	-1
canopy	1	0	1	-1

.

Clustering Method	Run-time AUC Relative to K-means
genic	6
canopy	52
kmeans	100

Not as drastic of increases as with reduction, though still much faster. Genic is only 6% of K-means

Runtime AUC Plot



Purity

	Purity MWU results			
Clustering Method	**ties**	**wins**	**losses**	**wins-losses**
kmeans	0	2	0	2
genic	0	1	1	0
canopy	0	0	2	-2

Clustering Method	**Purity AUC Relative to K-means**
kmeans	100
canopy	72
genic	64

As expected, $K - means$ wins with purity.

External Similarity

	External Sim MWU results			
Clustering Method	ties	wins	losses	wins-losses
genic	0	2	0	2
kmeans	0	1	1	0
canopy	0	0	2	-2

Clustering Method	External Sim AUC Relative to K-means
genic	91
kmeans	100
canopy	113

Internal Similarity

	Internal Sim MWU results			
Clustering Method	**ties**	**wins**	**losses**	**wins-losses**
kmeans	0	2	0	2
genic	0	1	1	0
canopy	0	0	2	-2

Clustering Method	**Internal Sim AUC Relative to K-means**
canopy	83
genic	91
kmeans	100

Similarity Loss

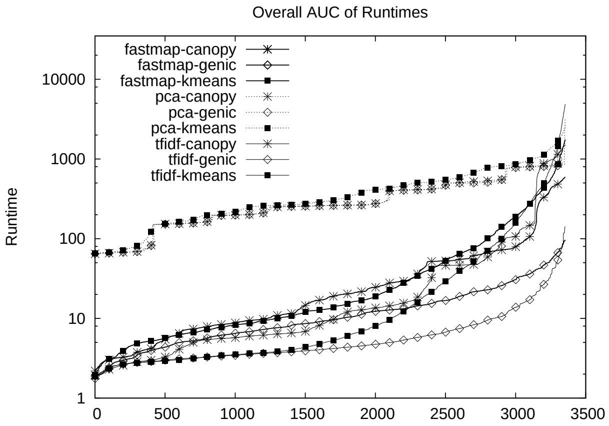
Clustering Method	Similarity Loss
canopy	26
genic	22

Runtimes

Reducer-Clusterer	Run-time MWU results			
	ties	wins	losses	wins-losses
tfidf-genic	0	8	0	8
tfidf-kmeans	0	7	1	6
fastmap-genic	0	6	2	4
fastmap-kmeans	0	5	3	2
tfidf-canopy	0	4	4	0
fastmap-canopy	0	3	5	-2
pca-genic	0	2	6	-4
pca-canopy	0	1	7	-6
pca-kmeans	0	0	8	-8

Reducer-Clusterer	Run-time AUC Relative to PCA-Kmeans
tfidf-genic	1
fastmap-genic	3
tfidf-canopy	5
fastmap-canopy	7
tfidf-kmeans	12
fastmap-kmeans	20
pca-genic	71
pca-canopy	76
pca-kmeans	100

Runtime AUC Plot



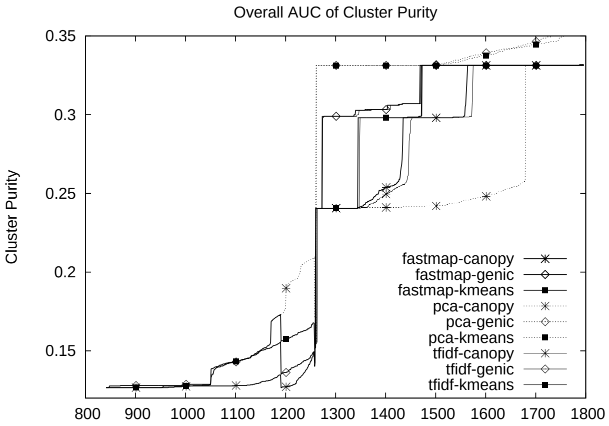
Purity

Reducer-Clusterer	Purity MWU results			
	ties	**wins**	**losses**	**wins-losses**
pca-kmeans	0	8	0	8
pca-genic	0	7	1	6
fastmap-genic	1	5	2	3
fastmap-canopy	1	5	2	3
tfidf-kmeans	1	3	4	-1
fastmap-kmeans	1	3	4	-1
tfidf-canopy	0	2	6	-4
tfidf-genic	0	1	7	-6
pca-canopy	0	0	8	-8

Reducer-Clusterer	Purity AUC Relative to PCA-Kmeans
pca-kmeans	100
pca-genic	95
tfidf-kmeans	73
fastmap-kmeans	73
tfidf-genic	67
fastmap-genic	68
tfidf-canopy	50
fastmap-canopy	50
pca-canopy	38

As expected, $K - means$ wins with purity.

Purity AUC Plot



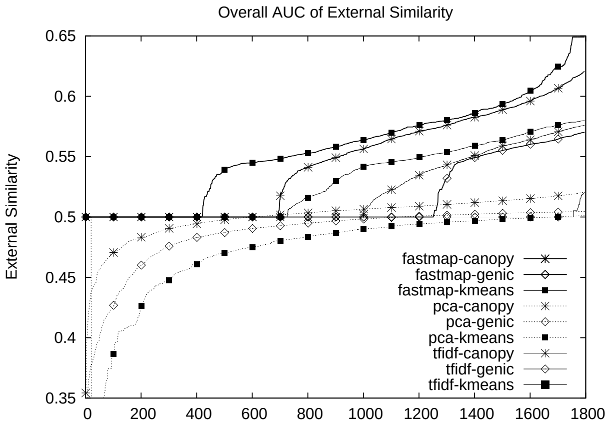
External Similarity

	External Sim MWU results			
Reducer-Clusterer	**ties**	**wins**	**losses**	**wins-losses**
pca-kmeans	0	8	0	8
tfidf-genic	0	7	1	6
pca-genic	0	6	2	4
tfidf-kmeans	1	4	3	1
pca-canopy	0	4	4	0
tfidf-canopy	1	3	4	-1
fastmap-genic	0	2	6	-4
fastmap-kmeans	0	1	7	-6
fastmap-canopy	0	0	8	-8

Reducer-Clusterer	**External Sim AUC Relative to PCA-Kmeans**
tfidf-canopy	78
fastmap-canopy	79
tfidf-kmeans	80
pca-canopy	81
fastmap-kmeans	87
pca-kmeans	100
pca-genic	102
tfidf-genic	103
fastmap-genic	107

Lower means better performance and Genic, one of the heuristic approaches, is defnitely lower that K-means

External Similarity AUC Plot

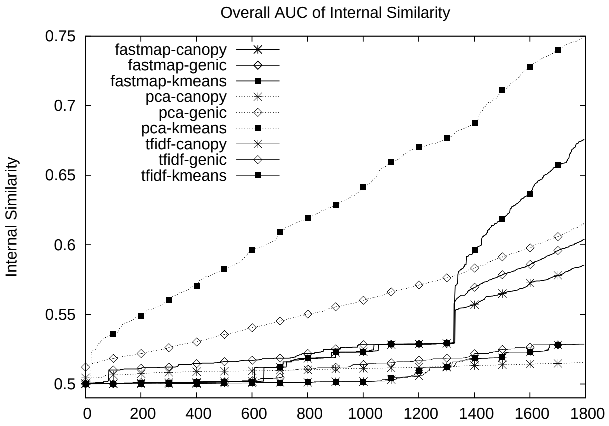


Internal Similarity

	Internal Sim MWU results			
Reducer-Clusterer	**ties**	**wins**	**losses**	**wins-losses**
pca-kmeans	0	8	0	8
pca-genic	0	7	1	6
fastmap-genic	1	5	2	3
fastmap-canopy	1	5	2	3
tfidf-kmeans	1	3	4	-1
fastmap-kmeans	1	3	4	-1
tfidf-canopy	0	2	6	-4
tfidf-genic	0	1	7	-6
pca-canopy	0	0	8	-8

Reducer-Clusterer	**Internal Sim AUC Relative to PCA-Kmeans**
pca-kmeans	100
fastmap-kmeans	90
pca-genic	88
fastmap-genic	82
fastmap-canopy	81
tfidf-kmeans	79
tfidf-genic	75
tfidf-canopy	74
pca-canopy	71

Internal Similarity AUC Plot



Similarity Loss

Reducer-Clusterer	Similarity Loss Relative To pca-kmeans
fastmap-kmeans	-3
fastmap-canopy	-2
tfidf-kmeans	1
tfidf-canopy	4
pca-canopy	10
pca-genic	14
fastmap-genic	25
tfidf-genic	28

Reducer-Clusterer	ties	wins	losses	wins-losses
pca-kmeans	0	8	0	8
tfidf-kmeans	1	6	1	5
fastmap-kmeans	1	6	1	5
pca-genic	0	5	3	2
tfidf-canopy	2	2	4	-2
fastmap-genic	2	2	4	-2
fastmap-canopy	2	2	4	-2
tfidf-genic	0	1	7	-6
pca-canopy	0	0	8	-8

Reducer-Clusterer Similarity Difference MWU test results

Outline

Conclusions

- **Dimensionality Reduction:**
  - PCA wins at all validity contents, but is also the largest potential performance bottleneck
  - Fastmap can do the job if run-time contraints exist
  - TFIDF-sort is your best bet if even larger run-time contraints exist
- **Clustering:**
  - As expected, K-means wins at most validity contests
  - Genic outperforms Canopy in every validity and even beats K-means on external similarity
  - Canopy performance leaves much to be desired

Conclusions

- **Combined:**
  - Fastmap-Genic provide a highly scalable solution without sacraficing too much in the way of validity
  - Simply replacing PCA can vastly reduce run-times and still maintain the validity of K-means
  - TfIdf and KMeans combination shows astounding performance in validity at only 12% of the base line run-time

Summary

Heuristic methods are worth it

Paths Going forward

- Closer examination of in-use applications
- More in-depth analysis on the effect of the various algorithms parameters
- Use more current algorithms (i.e. LSI)
- Find out what Genic is capable of
- Explore heuristic ways of finding the optimal/inherent dimensionality
- Explore heuristic ways of determining the number of clusters

Questions?

at UCD, M. L. G. (2009).
Mlg datasets.
http://mlg.ucd.ie/datasets.

Bay, S. D., Kibler, D. F., Pazzani, M. J., and Smyth, P. (2000).
The UCI KDD archive of large data sets for data mining research and experimentation.
SIGKDD Explorations, 2(2):81–85.

Ding, C. and He, X. (2004).
K-means clustering via principal component analysis.
In *ICML '04: Proceedings of the twenty-first international conference on Machine learning*, page 29, New York, NY, USA. ACM.

Faloutsos, C. and Lin, K.-I. (1995).
Fastmap: a fast algorithm for indexing, data-mining and visualization of traditional and multimedia datasets.
In Carey, M. and Schneider, D., editors, *Proceedings of the 1995 ACM SIGMOD International Conference on Management of Data*, pages 163–174. ACM Press.

Grimes, S. (2008).
Unstructured data and the 80 percent rule.
Experts Corner: Seth Grimes, Clarabridge Bridgepoints, Issue 3, 2008.
White Paper.

Grootjen, F., van Leijenhorst, D., and van der Weide, T. P. (2003).
A formal derivation of heaps' law.

Gupta, C. and Grossman, R. (2004).
Genic: A single pass generalized incremental algorithm for clustering.

In *In SIAM Int. Conf. on Data Mining*. SIAM.

Jolliffe, I. (2002).
Principal component analysis. 2nd edition.
Springer.

Jones, K. S. (1993).
A statistical interpretation of term specificity and its application in retrieval.
Journal of Documentation, 28:11–21.

Kanungo, T., Mount, D., Netanyahu, N., Piatko, C., Silverman, R., and Wu, A. (2000).
The analysis of a simple k-means clustering algorithm.
In *UMD*.

Lang, K. (1995).
Newsweeder: Learning to filter netnews.
In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 331–339.

McCallum, A., Nigam, K., and Ungar, L. H. (2000).
Efficient clustering of high-dimensional data sets with application to reference matching.
In *KDD '00: Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 169–178, New York, NY, USA. ACM.

Porter, M. F. (1980).
An Algorithm for Suffix Stripping.
Program, 14(3):130–137.

Ramos, J. (2003).
Using tf-idf to determine word relevance in document queries.

Salton, G. (1991).
The smart document retrieval project.
In *SIGIR '91: Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 356–358, New York, NY, USA. ACM.