Clustering with Random Indexing K-tree and XML Structure

Christopher M. De Vries, Shlomo Geva, and Lance De Vine

Faculty of Science and Technology,
Queensland University of Technology, Brisbane, Australia
chris@de-vries.id.au s.geva@qut.edu.au l.devine@qut.edu.au

Abstract. This paper describes the approach taken to the clustering task at INEX 2009 by a group at the Queensland University of Technology. The Random Indexing (RI) K-tree has been used with a representation that is based on the semantic markup available in the INEX 2009 Wikipedia collection. The RI K-tree is a scalable approach to clustering large document collections. This approach has produced quality clustering when evaluated using two different methodologies.

Key words: INEX, XML, Mining, Documents, Clustering, Structure, K-tree, Random Indexing, Random Projection

1 Introduction

The cluster hypothesis suggests that documents that cluster together tend to have relevance to similar queries. The clustering task at INEX 2009 aims to evaluate the utility of clustering in collection selection. The goal of clustering is to minimise the spread of relevant results of ad-hoc queries over a clustering solution. The purpose of clustering in this context is to determine the distribution of a collection over multiple machines. We have a dual optimisation problem - it is desirable to maximise the number of clusters while minimising the spread of relevant results of ad-hoc queries over the clusters. Search efficiency can be increased with the distribution of clusters (sub-collections) on more machines. However, since it is not possible to produce clusters that split the collection to perfectly satisfy all conceivable ad-hoc queries, a good clustering solution is expected to optimise the distribution such that for most ad-hoc queries most of the results can be found in a small set of clusters. The goal of collection selection is then to rank the clusters (sub-collections) to identify the order in which they should be searched to satisfy any given query.

We have used K-tree [1, 2] to generate clustering solutions. The scalability of K-tree in a document clustering setting has been discussed by De Vries and Geva [3, 4]. The original contribution to K-tree in INEX 2009 is the use of Random Indexing (RI) to represent the documents. The K-tree algorithm has also been modified to work with the RI representation. RI facilitates an efficient and economical vector space representation. The RI K-tree provides a scalable approach

to clustering large collections at multiple granularities. The latest Wikipedia collection has included semantic markup that is based on the YAGO ontology. This markup had been used in encoding the documents, and two simple approaches are described in Section 4.

This paper introduces and defines Random Indexing in Section 2 and explains its use with K-tree in Section 3. The representation of semantic markup is discussed in Section 4. The combination of RI K-tree and representation of semantic markup introduced in earlier sections is applied to the INEX clustering task in Sections 5, 6 and 7. The paper ends with a conclusion in Section 8

2 Random Indexing

RI [5] is an efficient, scalable and incremental approach to the implementation of a word space model. Word space models use the distribution of terms in documents to create high dimensional document vectors. The directions of these document vectors represent various semantic meanings and contexts.

Latent Semantic Analysis (LSA) [6] is a popular word space model. LSA creates context vectors from a document term occurrence matrix by performing Singular Value Decomposition (SVD). Dimensionality reduction is achieved through projection of the document term occurrence vectors onto the subspace spanned by the vectors with the largest singular values in the decomposition. This projection is optimal in the sense that it minimises the sum of squares of the difference between the original matrix and the projected matrix components. In contrast, Random Indexing first creates random context vectors of lower dimensionality and then combines them to create a term occurrence matrix in the dimensionally reduced space. Each term in the collection is assigned a random vector and the document term occurrence vector is then a superposition of all the term random vectors. There is no matrix decomposition and hence the process is efficient.

RI is also known as Random Projection and is explained by the Johnson and Linden-Strauss lemma [7]. It states that if points in a high dimensional space are projected into a lower dimensional, randomly selected subspace of sufficient dimensions they will approximately retain the same topology. Any n point set in Euclidean space can be embedded in $O(\log n/\epsilon^2)$ dimensions without distorting the pair-wise distances between points by more than $1 \pm \epsilon$, where $0 < \epsilon < 1$. Dasgupta and Gupta [8] have provided a proof for the Johnson and Linden-Strauss lemma, showing that the proposed bounds of the lemma hold.

The RI mapping is performed by producing r dimensional index vectors for each term in a collection, where r is the desired dimensionality of the reduced space. We have chosen these vectors to be sparse and ternary. Ternary index vectors were introduced by Achlioptas [9] as being friendly for database environments. Bingham and Mannila [10] have found that the sparsity of index vectors does not effect the distortion of the embedding via experimental analysis. Sparse index vectors reduce the time to complete RI as only 10 percent of the dimensions are non-zero. However, other choices exist for index vectors such as binary

spatter codes [11] which are randomly selected binary vectors and holographic reduced representations [12] that are dense randomly selected real valued vectors. When indexing the INEX 2009 collection, the index vectors are multiplied by the BM25 weight for each term in each document and added to the RI document vector. The document vector becomes a superposition of the index vectors multiplied by the term weights as determined by BM25.

RI can be viewed as a matrix multiplication of a document by term matrix D and a random projection matrix I resulting in a reduced matrix R. Row vectors of I contain index vectors of r dimensions for each term in D. Moreover, n is the number of documents, t is the number of terms and r is the dimensionality of the reduced spaced. R is the reduced matrix where each row vector represents a document. Equation 1 defines RI as a matrix multiplication.

$$D_{n\times t} I_{t\times r} = R_{n\times r} \tag{1}$$

Note that the RI document vectors themselves are not random. They are composed of a superposition of random term vectors and the superposition result depends on BM25 term weights and document content.

Another way to view RI is to interpret each index vector as a code. These codes are nearly orthogonal to all other codes produced, resulting in minimal interference between terms in the reduced vector space. Orthogonality can be measured by creating a pair-wise distance matrix between index vectors using cosine similarity as a distance measure. If two vectors are orthogonal their cosine similarity will be zero. The closer the vectors are to orthogonal the closer their cosine similarity will be to zero. Therefore, it is expected that the pair-wise distance matrix will contain values close to zero in every position except the main diagonal. Finding truly orthogonal codes is computationally expensive and therefore avoided. Nearly orthogonal codes are found by drawing values in the vector from a normal distribution. Figure 1 shows the addition of index vectors (nearly orthogonal codes) to create a document representation.

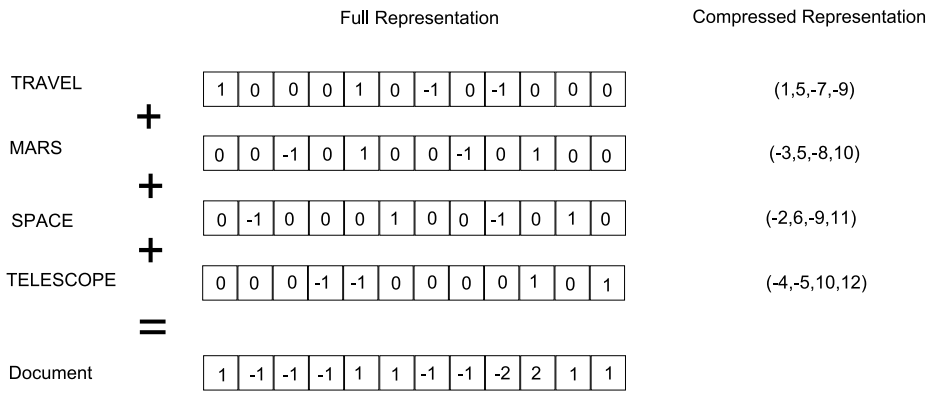

Fig. 1. Random Indexing Example

3 Random Indexing K-tree

The K-tree is an online and dynamic clustering algorithm that scales well by making many local decisions resulting in a hierarchical tree structure. It is a hybrid of the B^+-tree and k-means algorithms where the B^+-tree has been adapted for multi-dimensional data and the k-means algorithm is used to perform splits in the tree. It is built in a bottom-up manner as data arrives. De Vries and Geva [2–4] discuss the algorithm and its application to document clustering, including the scalability of the algorithm. K-tree was compared to the popular CLUTO clustering toolkit and found to cluster significantly faster when a large number of clusters are required [4]. The Random Indexing (RI) K-tree [13] combines K-tree with RI to improve the quality of results and run-time performance.

The time complexity of K-tree depends on the length of the document vectors. K-tree insertion incurs two costs, finding the appropriate leaf node for insertion and k-means invocation during node splits. It is therefore desirable to operate with a lower dimensional vector representation.

The combination of RI with K-tree is a good fit. Both algorithms operate in an on-line and incremental mode. This allows it to track the distribution of data as it arrives and changes over time. K-tree insertions and deletions allow flexibility when tracking data in volatile and changing collections. Furthermore, K-tree performs best with dense vectors, such as those produced by RI.

Given the scalable and dynamical properties of the RI K-tree algorithm we propose it is a good solution for clustering large volatile document collections. The logarithmic lookup time of K-tree [13] to find the most similar cluster is also of use in a functioning information retrieval system relying on collection selection. This allows a query broker to direct queries to the most relevant sub-collection in sub-linear time with respect to the size of the collection.

4 Document Representation

Document structure has been represented by using a bag of words and a bag of tags representation derived from the semantic markup in the INEX 2009 collection. Both are vector space representations.

The bag of words is made up of term frequencies contained within any entity tags in the collection. The term frequencies were weighted with BM25 [14] where $K1 = 2$ and $b = 0.75$. We hypothesise that terms contained within entity tags are more likely to indicate the topic of a document. Therefore, documents with the same topic will fall closer together in the vector space representation. This indirectly exploits the XML structure.

The bag of tags representation is made up in an analogous manner of XML entity tag frequencies. The tag frequencies were not weighted. Entity tags consist of concepts such as scientist, location and person. We conjecture that documents with similar tags will belong to the same topic. Future work may compare tag frequency based vectors to set based vectors. In set based vectors each tag would be recorded as existing in a document or not. This way it can be determined if

the use of tag frequencies is worthwhile. If a power law distribution exists in tag frequencies the Inverse Document Frequency heuristic may also prove useful as it did with link graphs [3]. The entity tags directly exploit structure by indexing it.

The bag of words and tags representations were combined. This is done by adding the two vector space representations together and then normalising the resulting vector to unit length. As both of the representations are based on RI, the codes between representations will be nearly orthogonal. However, a larger number of dimensions may be required to accommodate the extra information.

5 Run-time Performance

The performance of RI K-tree has been measured when operating in main memory. The concern is with the performance of the clustering algorithm. Efficiency was not taken into account when indexing or loading the final representation into memory.

All performance figures are for processing all 2,666,190 XML Wikipedia documents. The RI operations took a total of 1860 seconds for the entity text representation. The randomly selected lower dimensional space had 1000 dimensions. The run time of the K-tree algorithm varies between 1200 and 1500 seconds depending on the tree order selected between 15 and 50. This includes the process of re-inserting all vectors to their nearest neighbour leaves upon completion of the tree building process. This produces clustering at many different granularities at once. Table 1 lists the different sized clusters found by trees of order 20, 40, 60, 80 and 100, where m is the tree order.

Level	$m = 20$	$m = 40$	$m = 60$	$m = 80$	$m = 100$
1	12	3	8	3	92
2	111	89	356	129	2011
3	542	1260	5610	3090	53174
4	2161	12865	89612	67794	
5	8529	154934			
6	37230				
7	197299				

Table 1. K-tree Clusters

6 Experimental Setup

The Random Indexing (RI) K-tree has been used to cluster all 2,666,190 XML documents in the INEX 2009 Wikipedia collection. Bag of words and tags representations were used to create different clusters. Both representations were also

combined to create a third set of clusters. Clusters were created as close as possible to the 100, 500, 1000, 5000 and 10000 clusters required for evaluation. The RI K-tree produces clusters in an unsupervised manner where the exact number of clusters can not be precisely controlled. It is determined by the tree order and the randomised seeding process. The algorithm produces clusters of many sizes in a single pass. The desired clustering granularity is selected by choosing a particular level in the tree.

Random Indexing (RI) is an efficient dimensionality reduction technique that projects points in a high dimensional space onto a randomly selected lower dimensional space. It is able to preserve the topology of the points. In the context of document representation, topology preserving dimensionality reduction is preserving document meaning, or at least this is the conjecture which we test here. The RI projection produces dense document vectors that work well with the K-tree algorithm.

Cluster quality has been measured with two metrics this year. Purity is a commonly used metric and it is measured against an external ground truth. In the case of the INEX 2009 collection, the categories were created by YAWN. Purity is the fraction of documents with the majority category in a cluster. Micro purity is the average across all clusters in a solution where each cluster's contribution is weighted by the fraction of documents it contains from the whole collection. Thus, smaller clusters have less influence and larger clusters have more influence on the average. Normalised Cumulative Cluster Gain (NCCG) is a new measure based on relevance judgments from search queries in the ad-hoc track. The ad-hoc track at INEX provides most relevant documents for each topic based on manual human evaluations. Given the relevant results an oracle cluster ranking system can be built, where clusters are sorted in descending order by the number of relevant documents they contain. NCCG measures the spread of relevant documents over the clusters. A score of one is achieved if all relevant documents appear in the first cluster and a score of zero is achieved if relevant documents are evenly spread across all clusters. NCCG rewards placing all relevant documents together. Therefore, it is testing the clustering hypothesis that states that relevant documents for a query tend to cluster together.

7 Experimental Results

Table 2 lists micro purity and NCCG scores for all submissions that clustered the full INEX 2009 collection. The table is split into sections corresponding to the required cluster sizes specified for the track. The RI K-tree, using the entity text representation is clearly the best approach when it comes to finding high purity clusters using an approach that can scale to the full collection at all cluster sizes. The NCCG metric for collection selection favours the combination of entity text and tags over either representation. It changes the ordering of results when compared to the traditional ground truth based approach. The C3m based approach produced higher quality clusters with respect to the NCCG metric on two occasions at 100 and 10,000 clusters.

Guyon et. al. [15] argue that the context of clustering needs to be taken into account during evaluation. The evaluation of this INEX task tests the clustering hypothesis in the information retrieval specific. Clustering is intended to facilitate document distribution and collection selection for ad-hoc retrieval, and it is tested in that setting. This differs greatly from evaluation where authors assign categories to documents and the categories are then used as the ground truth for the evaluation of clustering. Guyon et. al. [15] argue, and we agree, that ground truth based evaluations are unsound. This is particularly true when it comes to an information retrieval setting where the number of potential topics (clusters) is virtually unconstrained. It is a virtually impossible task to compare alternative clustering possibilities by inspecting large numbers of documents in clusters. In contrast, the evaluation of topics represented as queries in an ad-hoc retrieval system achieves high levels of inter-judge agreement. These relevance judgments have been the backbone of ad-hoc information retrieval system evaluations for many years. They have also been exposed to criticism and review by many of the top researchers in the field. By exploiting this high quality, human generated information, we can have great confidence that we are testing clustering in the context of its use. The context is specifically clustering of documents in an information retrieval setting.

Guyon et. al. go as far to say "In our opinion, this approach [ground truth approach] is dangerous. The underlying assumption is that points with the same class labels form clusters. This might be the case for some data sets but not for others.". If the ground truth reflected the application of clustering in an information retrieval context, then the scores would agree between purity and NCCG. However, they do not. Therefore, we argue that the NCCG scores based on ad-hoc queries are more meaningful in an information retrieval setting.

Relevance of documents to queries can also be derived from click-through data in an operational search engine. This provides a potential mountain of relevance judgments.

8 Conclusion

In conclusion the RI K-tree provided a scalable approach to clustering at multiple granularities in a single pass with quality comparable to other approaches. The hypothesis that combining entity text and tag based representations will improve quality held true for the new ad-hoc based evaluation. Furthermore, the evaluation provided insights into why it is important to take context of use into account when evaluating clustering.

Method	Clusters	Micro Purity	NCCG
RI K-tree Text	88	**0.1744**	0.7859
RI K-tree Tags	99	0.1427	0.7851
RI K-tree Text and Tags	105	0.1450	0.8003
C3m Content Only (bildb)	101	0.1566	**0.8205**
RI K-tree Text	420	**0.1918**	0.6770
RI K-tree Tags	477	0.1526	**0.7546**
RI K-tree Text and Tags	509	0.1668	0.7330
RI K-tree Text	1009	**0.2140**	0.6450
RI K-tree Tags	1026	0.1699	0.7021
RI K-tree Text and Tags	963	0.1690	**0.7092**
C3m Content Only (bildb)	1001	0.1617	0.6614
RI K-tree Text	2450	**0.2136**	0.6100
RI K-tree Tags	2407	0.1769	0.6348
RI K-tree Text and Tags	2536	0.1928	**0.6575**
BM25 BicMsGrowingKMeans (mark)	2263	0.1698	0.6349
RI K-tree Text	4914	**0.2384**	0.5581
RI K-tree Tags	4993	0.2020	0.5729
RI K-tree Text and Tags	4978	0.2038	**0.6003**
RI K-tree Text	9725	**0.2719**	0.4736
RI K-tree Tags	10453	0.2321	0.5274
RI K-tree Text and Tags	9896	0.2509	0.5492
C3m Content Only (bildb)	10001	0.1942	**0.6035**
BM25 BicMsGrowingKMeans (mark)	12636	0.2416	0.5885

Table 2. Clusters

References

1. : K-tree project page, http://ktree.sourceforge.net. (2009)
2. Geva, S.: K-tree: a height balanced tree structured vector quantizer. Proceedings of the 2000 IEEE Signal Processing Society Workshop Neural Networks for Signal Processing X, 2000. **1** (2000) 271–280 vol.1
3. De Vries, C., Geva, S.: Document clustering with k-tree. Advances in Focused Retrieval: 7th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2008, Dagstuhl Castle, Germany, December 15-18, 2008. Revised and Selected Papers (2009) 420–431
4. De Vries, C., Geva, S.: K-tree: large scale document clustering. In: SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval, New York, NY, USA, ACM (2009) 718–719
5. Sahlgren, M.: An introduction to random indexing. In: Methods and Applications of Semantic Indexing Workshop at the 7th International Conference on Terminology and Knowledge Engineering, TKE 2005. (2005)
6. Deerwester, S., Dumais, S., Furnas, G., Landauer, T., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science **41**(6) (1990) 391–407
7. Johnson, W., Lindenstrauss, J.: Extensions of Lipschitz mappings into a Hilbert space. Contemporary mathematics **26**(189-206) (1984) 1–1
8. Dasgupta, S., Gupta, A.: An elementary proof of the Johnson-Lindenstrauss lemma. Random Structures & Algorithms **22**(1) (2002) 60–65
9. Achlioptas, D.: Database-friendly random projections: Johnson-Lindenstrauss with binary coins. Journal of Computer and System Sciences **66**(4) (2003) 671–687
10. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: KDD '01: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, New York, NY, USA, ACM (2001) 245–250
11. Kanerva, P.: The spatter code for encoding concepts at many levels. In: ICANN94, Proceedings of the International Conference on Artificial Neural Networks. (1994)
12. Plate, T.: Distributed representations and nested compositional structure. PhD thesis (1994)
13. De Vries, C., De Vine, L., Geva, S.: Random indexing k-tree. In: ADCS09: Australian Document Computing Symposium 2009, Sydney, Australia. (2009)
14. Robertson, S., Jones, K.: Simple, proven approaches to text retrieval. Update (1997)
15. Guyon, I., von Luxburg, U., Tubingen, G., Williamson, R., Canberra, A.: Clustering: Science or Art?