Document Classification Techniques using LSI

Barry Britt
University of Tennessee
Department of Computer Science
Knoxville, TN

GTP and LSI

Latent Semantic Indexing (LSI) is a vector space model in which a weighted term-by-document matrix A can be decomposed into

$$A = U\Sigma V^{T}$$

where U is the matrix of left orthonormal eigenvectors of left singular vectors of A, Σ is the diagonal matrix of singular values in decreasing order of magnitude, and V^{T} is the matrix of right orthonormal eigenvectors of right singular vectors of A[1]. This decomposition facilitates the representation of documents and terms in a *k*-dimensional vector space. Typically the leading *k* left and right singular vectors (denoted by U_k and V_k respectively) are used to suppress noise and simplify searching of the reduced-rank vector model.

Using this model, queries are relatively simple. A query is defined as

$$q^{'} = q^{T} U_k \Sigma_k^{-1}$$

where q is a sparse vector of global weights corresponding to terms in the query, $q^{T} U_k$ is the rank *k* approximation of *q* into *k* space, and Σ_k^{-1} is the matrix of reciprocal singular values that appropriately weights each of the *k* separate dimensions. Query results are returned as the cosine similarity value,

$$\cos\Theta = q \bullet d / \| q \| \times \| d \|$$

where q is the query vector, d is a document vector, and ||q|| denotes the Euclidean vector norm of q.

The benefit of using LSI is evident in the quality of returned query results. Rather than simple keyword searches, LSI query results are contextually related to the query terms or document. Rank-reduction in the LSI model typically forces semantically linked terms and documents to be grouped together in the *k*-dimensional vector space [2]. When cosine similarity values are computed, these documents and terms rank much higher than terms and documents of little or no semantic similarity.

General Text Parser (GTP) is a program used to generate vector space models, such as LSI [3]. GTP supports local and global weighting and document frequencies, document normalization schemes, keyword lengths, vector space decomposition schemes, and much more. All of the LSI models used for classification of documents in this study were built using GTP.

Document Sets

Our test document sets consisted of varying number of Technology Readiness Reports. Each of these reports described a “readiness level” – a subjective score on a scale from 1 to 9 (9 being the highest score) – that represents the capability of a group to provide a technology, or service. Each document set was no larger than 4000 documents, most of which contained fewer than 50 words. The documents were in a “bag-of-words” format with no inherent structure, and there was no metadata we could use to gather any extra information from the documents.

For this report, we generated results for three different document sets. All three document sets consisted of two classes, denoted “Phase1” and “Phase2.” The first set (DS1) contained 4000 documents. The average word count of the documents was 85.4 terms per document. The second set (DS2) contained 4000 documents with an average word count of 49.2 terms per document. The final set (DS3) contained 455 documents with an average word count of 37.1 terms per document (Figure 1-a).

	# Docs	# Classes	Avg. Terms/Doc
DS1	4000	2	85.4
DS2	4000	2	49.2
DS3	455	2	37.1

Figure 1-a: Document Set general information

	# Class 1	# Class 2	% Class 1	% Class 2
DS1	3244	756	81.10%	19.90%
DS2	2955	1045	73.88%	26.13%
DS3	166	289	36.48%	63.52%

Figure 1-b: Document Set Class information

The class distributions for the sets varied quite a bit. As you can tell from Table 2, the first class was dominant in DS1 and DS2. Though DS3 is much smaller, the second class dominated with over 63% of the documents belonging to that class.

Document Labels

Each document had a predetermined label corresponding to a specific Technology or Manufacturing Readiness Level (TRL or MRL respectively) that was determined internally by each organization that submitted the reports. It is important to note that labels were subjectively assigned to projects based on the composition of the document that describes the manufacturing or technological process involved. Training and test sets were used to judge the accuracy of the LSI-based classifier.

Classification Using a “Majority Rules” Classifier

The most obvious choice for a classifier was the “majority rules” classifier. For this technique, we selected a single document from our document set, stripped its label, and retrieved its pre-computed cosine similarity values with all other documents. We then took the top-ranked subset *S* (usually 10 documents, but this varied depending on the classifier and document sets) and we counted the number of documents belonging to each class. For classification, we assigned the label with the greatest frequency to the selected query.

The majority rules classifier performed well for the dominant class, as it had a majority of the "votes" when determining the class of a document. This suggested that any minority classes would not be represented well in the classification.

	Phase 1	Phase 2
Phase 1	3239	5
Phase 2	752	4

Figure 2-a: "Majority Rules" confusion matrix for DS1

	Phase 1	Phase 2
Phase 1	2875	80
Phase 2	833	212

Figure 2-b: "Majority Rules" confusion matrix for DS2

	Phase 1	Phase 2
Phase 1	98	68
Phase 2	53	236

Figure 2-c: "Majority Rules" confusion matrix for DS3

	Class 1	Class 2	Overall
DS1	99.85%	0.53%	81.08%
DS2	97.29%	20.29%	77.18%
DS3	59.04%	81.66%	73.41%

Figure 2-d: Precision calculations for DS1, DS2, and DS3

In each case, the class that contained the majority of documents had a stronger showing than the other classes, as shown in Figure 2. For example, in Figure 2-d, since DS1 contained 80% "Phase1" documents; there is a strong probability that documents will cluster around the documents containing this label. The only case where the second class performed better than the first was in DS3; most of the documents in this set had a label of "Phase2".

Improving the Classification Scheme via Class Weighting

Since the results in Figure 2 were not very good for the second class, we needed to find an alternative to a straight majority classification. By adding a weight factor to the classification, we increased the importance of document similarity in the underrepresented class.

The class weight was applied after the results of the query had been tallied. Each class was assigned a separate weight; the label with the greatest value was applied to the document. The determination of class weights was based on the distribution of documents across the labels. For example, if there were 3000 “Phase1” documents and 1000 “Phase2” documents, we could assign a weight of 1 to the “Phase1” documents and 3 to the “Phase2” documents. A trial and error process to determine the optimal weight scheme for a set of documents could further refine this weight scheme.

For DS1, our optimal two-class score function was defined as

$$\max[n_1/4, n_2]$$

where n_i was the frequency of documents that belonged to class *i*. This function was based on the distribution of documents in the set in which “Phase1” documents outnumbered “Phase2” documents four to one. Not every document set had the same scoring function. The optimal DS2 scoring function was $\max[n_1/2, n_2]$, and the optimal DS3 scoring function was $\max[n_1, 2n_2/3]$.

Figure 3 shows the weighted classification of our three document sets.

	Phase 1	Phase 2
Phase 1	2355	889
Phase 2	323	433

Figure 3-a: Weighted classification confusion matrix for DS1

	Phase 1	Phase 2
Phase 1	2381	574
Phase 2	498	547

Figure 3-b: Weighted classification confusion matrix for DS2

	Phase 1	Phase 2
Phase 1	132	34
Phase 2	105	184

Figure 3-c: Weighted classification confusion matrix for DS3

	Class 1	Class 2	Overall
DS1	72.60%	57.28%	69.70%
DS2	80.58%	52.34%	73.20%
DS3	79.52%	63.89%	69.60%

Figure 3-d: Precision calculations for DS1, DS2, and DS3

To come up with an optimal weight function, we always started with a ratio based on the set composition. This would not always provide the best classification, but it did provide a good classification scheme that could be further adjusted by testing. From there, we tested weights by systematically adjusting the weight of the under-represented class. For each document set, we wrote a script that systematically tested all class weight combinations in the range $0 < weight \leq 10$

Further Improvements to Class Weighting

The addition of class weights greatly improved the Majority Rules classification as the second class was better represented (though the first class did not perform as well). Since we were trying to classify documents into either set, Figure 3 was more representative of our desired results. Further improvements could be achieved by considering the relative size of the documents in the set.

Documents with a lower TRL/MRL contained fewer significant words than those with a higher MRL or TRL. We realized that we could count the number of words in each document and only consider documents whose size (number of terms) was within *x* words of the query document. For simplicity, this classifier will be denoted WS, or "Weight and Document Size Classification."

Formally, let *s* be the number of words parsed from the document *S*. Let *s – x* be a lower bound on document size and let *s + x* be the upper bound (for any positive integer *x*). If *q* is the number of words parsed from a returned query document *Q*, and $(s - x) \leq q \leq (s + x)$, then *Q* is considered to be in the set for classifying document *S*.

This classification was used in conjunction with the previous weighted classification technique. For the document sets, we set *x = 5*. The weights remained the same as in the previous technique.

The choice of *x* in these document sets was not an arbitrary decision. We tested values for *x* ranging from 2 to 50. Five was the optimal choice because the number of returned values was large enough to provide a fair classification, but small enough that it seemed to cluster well. This choice may not be proper for all document sets, but it seemed to work fairly well for these document sets.

	Phase 1	Phase 2
Phase 1	2379	865
Phase 2	151	605

Figure 4-a: WS classification confusion matrix for DS1

	Phase 1	Phase 2
Phase 1	2494	461
Phase 2	364	681

Figure 4-b: WS classification confusion matrix for DS2

	Phase 1	Phase 2
Phase 1	130	36
Phase 2	79	210

Figure 4-c: WS classification confusion matrix for DS3

	Class 1	Class 2	Overall
DS1	73.34%	80.03%	74.60%
DS2	84.40%	65.17%	79.38%
DS3	78.31%	72.66%	74.73%

Figure 4-d: Precision calculations for DS1, DS2, and DS3

One important note is that the overall precision for the document sets dropped between this classification and the first. Initial classifications were not representative of the document set because the minority class performed poorly. For each document set, the WS classifier - which accounted for document size and class distribution - increased the probability of the minority class being correctly identified.

Alternative Classification Techniques - Terms Classification and Reclassifying Documents with Confidence Measurements

Within the LSI vector space model, terms are classified along with their representative documents. Rather than documents, we can use any of the previous classification schemes to classify the terms which are clustered near documents in the vector space. That is, instead of retrieving $\cos(\vec{q},\vec{d}_i)$ for a query vector $\vec{q}$ and the *i*th document vector $\vec{d}_i$, we retrieve $\cos(\vec{q},\vec{t}_i)$. We can use the weighted classification scheme (the WS classification is not relevant in this case) to classify the closest terms to a document, and assign a label to that document based on its similar terms.

	Phase 1	Phase 2	Weights
Phase 1	2432	812	1
Phase 2	343	413	8

(Figure 5-a: Term classification confusion matrix for DS1)

	Phase 1	Phase 2	Weights
Phase 1	2256	699	1
Phase 2	411	643	4

(Figure 5-b: Term classification confusion matrix for DS2)

	Phase 1	Phase 2	Weights
Phase 1	148	18	3
Phase 2	45	244	1

(Figure 5-c: Term classification confusion matrix for DS3)

	Class 1	Class 2	Overall
DS1	74.97%	54.63%	71.13%
DS2	76.35%	61.01%	72.48%
DS3	89.16%	84.43%	86.15%

Figure 5-d: Precision calculations for DS1, DS2, and DS3

This technique was not consistently superior to the WS classifier, though the small document set did very well. This was likely due to the relative size of document set and the smaller number of total words in the set.

However, this technique was comparable to the WS classifier. When we checked the actual results, we realized that this technique correctly classified some documents that the WS classifier misclassified. Several of these instances were not *confident* classifications from the WS classifier. To test this, we defined a confidence value by

$$c = v_i / \sum_{j=0}^{n} v_j$$

where v_i is the weighted value of class *i* and n is the total number of class labels in the document set.

We found that with a proper confidence threshold, we could correctly reclassify documents that were misclassified with the WS classifier. We researched defining a confidence threshold that would allow us to use the alternate classifier for documents that do not have a high enough confidence value, thereby increasing the effectiveness of our overall classification (see Figure 6).

Class Scheme	Doc #	Weighted Class 1	Weighted Class 2	Confidence	Classification	
					Predicted	Actual
WS	3997	5	4	0.556	0	1
Term	3997	8	16	0.667	1	1

Figure 6 - An example of reclassification of a document from DS1

This is still an open topic. We have not yet been able to determine a confidence threshold for which reclassification consistently yields better results than the original classifier. Right now, lies in providing a “second opinion” on the class of a document rather than a primary classification technique.

A Java Front End for GTP

Currently under development is a Java program that assists the user in running GTP, querying document sets, and classifying documents. This program has a graphical user interface and customizable configuration options that will keep a user’s GTP settings every time the user restarts the program.

The program also allows a user a “save state.” If the user has built an LSI model and generated the term and document query list, the state of the program can be save for later retrieval. All internal objects will be restored with this state. This program supports document sets that are up to 12 classes. Current MRL/TRL classifications range from one to nine and three to twelve, so this should be sufficient for the foreseeable future. All of the results for this report have been generated by data obtained from this tool.

Windows GTP

A port of GTP to the Windows environment is also currently under development. This program is functionally equivalent to the *NIX version of GTP, with some minor modifications. The original GTP code forks a UNIX *sort* command to sort the document set by terms. Those terms are then passed to GTP via pipe where they are counted.

The Windows version uses an external sort to accomplish the same task. The documents are sorted out of core memory and placed into a temporary file on disk. This file is then read back into GTP where they are counted.

A few other minor changes exist in the program. The program still uses the GNU Database Manager to build a hash table of keys and entries, but this keys database is not compatible with the *NIX version. There is also no *gzip* library available in Windows, so compression is not yet included in the Windows version.

Conclusions

The LSI vector space model provides a good environment to cluster and classify documents based on similarity measures. Properties of the model ensure that similar documents cluster according to underlying word usage patterns rather than simple word co-occurrence.

The decision of which classification scheme is used should be based on the size and composition of a document set. Sets that contain a dominant majority will be benefited by the weighted or WS classifier defined above. The determination of weights – and possibly upper or lower bounds - for these classifiers should be based on the document distribution, but the best choice may need to be determined by systematic testing.

The Term classifier provides a good alternative to straight document classification. By analyzing the document collection beforehand, a good weighted classifier can be determined. It may be possible to reclassify documents that do not have a good document confidence value, but this is still an open research question.

Acknowledgements

Support for this research project was provided by a subcontract to the Department of Computer Science at the University of Tennessee by InRAD, LLC of Knoxville, TN as part of a Phase I NSF SBIR Grant entitled “Technology Assessment and Readiness Analysis System TARAS.”

I would like to thank everyone that has been involved in the project. Maryann Merrell first contacted Dr. Berry about this project last summer, allowing me the opportunity to work on this. Dr. Berry was an invaluable asset as this project has. We heavily relied on his experience throughout the project. Murray Browne acted as a liaison between the University of Tennessee group and the InRAD group working on this project. Nathan Fisher, a former undergraduate student at the University of Tennessee wrote much of the term classification and confidence value code that was adapted for the Java implementation.

References

1) *"Using Linear Algebra for Intelligent Information Retrieval*." Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien, December 1994. Published in SIAM Review 37:4 (1995), pp. 573-595.
2) Understanding Search Engines: Mathematical Modeling and Text Retrieval. M. Berry and M. Browne, SIAM Book Series: Software, Environments, and Tools. (2005), ISBN 0-89871-581-4.
3) *"GTP (General Text Parser) Software for Text Mining."* J. T. Giles, L. Wo, and M. W. Berry, Statistical Data Mining and Knowledge Discovery. H. Bozdogan (Ed.), CRC Press, Boca Raton, (2003), pp. 455-471.