

# Knowledge Base Maintenance Using Knowledge Gap Analysis

Scott Spangler and Jeffrey Kreulen  
IBM Almaden Research Center  
650 Harry Road, San Jose, CA 95120-6099  
408-927-2887, 408-927-2431  
email: {spangles, kreulen}@almaden.ibm.com

## ABSTRACT

As the web and e-business have proliferated, the practice of using customer facing knowledge bases to augment customer service and support operations has increased. This can be a very efficient, scalable and cost effective way to share knowledge. The effectiveness and cost savings are proportional to the utility of the information within the knowledge base and inversely proportional to the amount of labor required in maintaining the knowledge. To address this issue, we have developed an algorithm and methodology to increase the utility of the information within a knowledge base while greatly reducing the labor required.

In this paper, we describe an implementation of an algorithm and methodology for comparing a knowledge base to a set of problem tickets to determine which categories and subcategories are not well addressed within the knowledge base. We utilize text clustering on problem ticket text to determine a set of problem categories. We then compare each knowledge base solution document to each problem category centroid using a cosine distance metric. The distance between the "closest" solution document and the corresponding centroid becomes the basis of that problem category's "knowledge gap". Our claim is that this gap metric serves as a useful method for quickly and automatically determining which problem categories have no relevant solutions in a knowledge base.

We have implemented our approach, and we present the results of performing a knowledge gap analysis on a set of support center problem tickets.

## Categories and Subject Descriptors

H.2.8 [Information Systems]: Data Mining;  
H.3.3 [Information Search & Retrieval]: Clustering;  
I.5.3 [Clustering]: Similarity Measures

## General Terms

Algorithms

## Keywords

Text Mining, Clustering, Knowledge Management, Gap Analysis

## 1. INTRODUCTION

In this paper, we focus on analyzing text produced by service and support helpdesks and corresponding knowledge bases. Human helpdesk operation is very labor intensive and therefore expensive. Consequently, automation of helpdesk problem solving represents a key objective for providers of electronic customer services. The first step in automation of a task is to understand it, and one of the first steps in understanding is to segment examples into meaningful categories [1]. The thesis of this paper is that there is much value derived from using our algorithms and methodology to automatically leverage the text within problem tickets to identify areas and specific problems within a knowledge base that need to be improved. Our approach uses the analysis, mining, and summarization of problem tickets text using an automated unsupervised clustering algorithm in concert with a data analyst. Our algorithms then automatically discover those categories of problem tickets which are not well represented within a knowledge base. The goal in this comparison is not to find the best solution to each problem category, but to see which problem categories are least addressed by the knowledge base.

In this paper, we will illustrate our results and methodology on a text data set containing over 10,000 helpdesk examples obtained from an IBM Global Services support center. A typical text document (known as a *problem ticket*) from this data set consists of a brief description by the helpdesk agent of the exchanges between an end user and an expert helpdesk advisor, for example:

```
1836853 User calling in
with WORD BASIC error when
opening files in word. Had
user delete NORMAL.DOT and
had her reenter Word, she
was fine at that point.
00:04:17 ducar May
2:07:05:656PM
```

Problem tickets may be comprised only of a single symptom and resolution pair as in the above example, or they may span multiple questions, symptoms, answers, attempted fixes, and resolutions—all pertaining to the same basic issue. Problem tickets are *opened* when the user makes the first call to the helpdesk and *closed* when all user problems documented in the first call are finally resolved in some way. Helpdesk operators enter problem tickets directly into the database. Spelling, grammar and punctuation are inconsistent. The style is terse and the vocabulary very specialized.

Given the inconsistency of problem ticket sentence structure, we determined that natural language processing techniques were out of the question. Nevertheless, significant information might still be obtained from a tabulation of word occurrences in each problem ticket. After eliminating common stop words and (high- and low-frequency) non-content-bearing words, we represented the text data set as a vector space model, that is, we represented each problem ticket as a vector of certain weighted frequencies of the remaining words [6]. We used the **txn** weighting scheme [7]. This scheme emphasizes words with high frequency in a document, and normalizes each document vector to have unit Euclidean norm. For example, if a document were the sentence, “We have no bananas, we have no bananas today,” and the dictionary consisted of only two terms, “bananas” and “today”, then the unnormalized document vector would be [2 1] (to indicate two bananas and one today), and the normalized version would be:  $\left[ \frac{2}{\sqrt{5}}, \frac{1}{\sqrt{5}} \right]$ .

As our primary tool for automated classification, we used the k-means algorithm [2] [3] using a cosine similarity metric [5] to automatically partition the problem tickets into k disjoint clusters. The algorithm is very fast and easy-to-implement. See [5] for a detailed discussion of various other text clustering algorithms. The k-means algorithm produces a set of disjoint clusters and a *centroid* for each cluster that represents the cluster mean.

Once a text categorization has been determined, the next step is to compare the centroid of each category to the contents of a knowledge base. The distance from each centroid to the nearest solution document is the basis for the “knowledge gap”. Once this knowledge gap is determined for each problem category, we can easily determine categories where additional solution documents may be required.

In Section 2, we describe our text classification methodology in some detail. In Section 3, we describe how we systematically compare problem classifications to a set of solution documents. In Section 4, we empirically compare our gap analysis metric to other potential metrics on some typical text data sets. Additionally, we discuss the results of applying our algorithm to an actual problem and solution set. Finally, in Section 5, we discuss future work and how our approach might be applied in other domains

## 2. TEXT CLASSIFICATION METHODOLOGY

In this section we describe the general process used for creating a classification of the problem ticket information. This classification is a critical step in the process of finding knowledge gaps between the problems and solutions. In the absence of such a classification we would need to do a detailed analysis of individual problem tickets against individual solution documents. Typically this is accomplished by sampling the problem tickets and manually categorizing and counting. Depending on the size of our problem ticket data set, this could be a daunting, expensive and error prone undertaking. This approach usually leads to the most common questions being addressed, but can miss many important problems and sometimes whole categories. Particularly, as more emphasis is placed on leveraging knowledge bases to augment service and support operations more and increasingly complex issues must be addressed within the knowledge base. The approach to manually categorize and count simply does not scale to this problem.

We use the k-means text clustering algorithm to obtain an initial clustering of the data. First, a dictionary of terms is generated by finding those words in the text that occur with the highest frequency. A generic *stop word* list is then subtracted to eliminate common non-informative words such as “and” and “the”. What usually remains, in addition to the informative words, is a set of proper names. Some of these, such as “Lotus”, may be useful features for clustering. Others, such as “Bob”, may be a distraction to the clustering algorithm. Our software provides features for automatically locating proper names (via capitalization) and for manually refining the dictionary to improve clustering performance. A stemming algorithm is employed to combine all variants of a word into a single term (e.g. print, prints, printer, printed, etc.). Each problem ticket is converted into a vector of floating point values by counting in each problem ticket the number of occurrences of each dictionary term. This integer vector is then normalized to have unit Euclidean norm. The k-means algorithm is then executed starting with k random seeds selected from the population of problem tickets. The distance metric employed is the cosine similarity metric. Thus two points are considered identical if the cosine of the angle between them is 1.0 and considered most dissimilar if the cosine of this angle is 0.0.

The k-means approach starts by finding the nearest of the k problem ticket seeds to each problem ticket example (using the cosine distance metric). Each problem ticket is then said to “belong” to the cluster corresponding to that nearest seed. For each cluster, a centroid is then calculated, which is the mean of the examples contained in that cluster. In the next iteration, each problem ticket is compared to every centroid and each problem ticket is moved to the cluster of the nearest centroid. The centroids are then recalculated and the process continues until an iteration does not result in any cluster membership changes from the previous iteration. The result of the clustering is the set of centroids

(one per cluster) along with an integer vector describing which cluster each problem ticket example belongs to.

### 3. FINDING THE KNOWLEDGE GAP

This section discusses our solution to the problem of how to find the knowledge gap between a set of categorized problems and a knowledge base. The goal in this comparison is not to find the best solution to each problem category, but to see which problem categories are least addressed by the knowledge base.

Our solution to the problem starts by measuring the cosine distance between every solution document and the centroid of every problem category. To do this, we first reduce each solution document to a vector of word occurrence counts, using the same dictionary and synonym list that was used in clustering the problem tickets. The *most similar* document in the solution set to a given cluster of problems is defined to be the document whose word occurrence vector is nearest to the centroid of the given cluster (using the cosine distance metric).

$$\cos(X, \text{centroid}(c)) = \frac{X \bullet \text{centroid}(c)}{\|X\| \cdot \|\text{centroid}(c)\|}$$

**Equation 1:** Where  $X$  is the document vector, and  $\text{centroid}(c)$  is the cluster centroid vector.

Another important component to determining the gap between a cluster and a solution document is to look at the cohesion of the cluster. The cohesion of a cluster,  $c$ , is a metric that is analogous to the *standard deviation* metric for random variables. It is the square root of the average cosine distance of the members of the cluster from their centroid.

$$\text{cohesion}(c) = \sqrt{\frac{\sum_{x \in c} \cos(\text{centroid}(c), x)}{|c|}}$$

**Equation 2:** Cohesion as a function of a cluster centroid a member document vectors.

The distance of each cluster in the problem categorization from its corresponding *most similar* solution document divided by the cohesion of the cluster is calculated and stored as a *g score*.

$$g(c) = \frac{\text{Max}_{y \in \text{solutions}} (\cos(\text{centroid}(c), y))}{\text{cohesion}(c)}$$

**Equation 3:** The g-score is the distance of the centroid to nearest solution divided by cohesion

Note that we use the maximum cosine distance to select the “most similar” document because cosine distance returns a value of 1.0 for identical documents and 0.0 for completely distinct documents. As the g-score increases, therefore, we expect the likelihood of a matching solution document to the cluster to also increase. A low g-score indicates that no matching solution document is present (thus a low g-score indicates a large “knowledge gap”). Clusters with cohesion approaching 1.0 (e.g. those consisting of nearly identical documents) will require solution documents nearer their centroid to achieve the same g-score as clusters with less cohesion. This inverse proportionality of cohesion is important when using the cosine distance metric to avoid the smallest g-score always corresponding to the cluster with the lowest cohesion since, generally, centroids from clusters with low cohesion tend not to match any particular document very exactly.

There are some inherent assumptions underlying this approach. The principle assumption is that solution documents contain many of the same terms as the problems that they intend to solve. While there are applications and domains where this may not be the case, we have observed across a broad set of helpdesks and knowledge bases this assumption to hold. We have found that in practice most helpdesk solution documents contain at least a brief summarization of the problem that they propose to solve, and this summary invariably will contain words that are similar to those used in the problem ticket descriptions. In addition, the problem tickets themselves will often contain brief descriptions of what was done to resolve the issue. In many cases, the terms in this solution description match those used in the solution documentation. Therefore, we believe this assumption is valid and will hold for many other domains as well.

### 4. EVALUATION OF THE KNOWLEDGE GAP METRIC

In order to evaluate the efficacy of our gap metric, we establish a situation where we know that a definitive knowledge gap exists and run our algorithm to determine if it correctly identifies the actual knowledge gap. For comparison, we also evaluate various other related metrics against the same data sets. One of the alternative metrics is the cosine distance metric, which is essentially the same as the g-score with the denominator (cohesion) always equal to 1.0. The second alternative is to use an Euclidean distance metric instead of the cosine distance metric. In the second case the denominator is also set to 1.0.

For this evaluation, our test sets are drawn from three distinct collections of test documents. One of these is the well-known Reuters-21578 data set [4] containing 9603 training documents, and the other two are helpdesk problem tickets from two different help centers. For the Reuters data set, we made use of both the existing classification of the data and a new classification generated via our text clustering approach. For the two help desk data sets, we created classifications using text clustering.

Figure 1: Comparing Gap Metrics by Accuracy

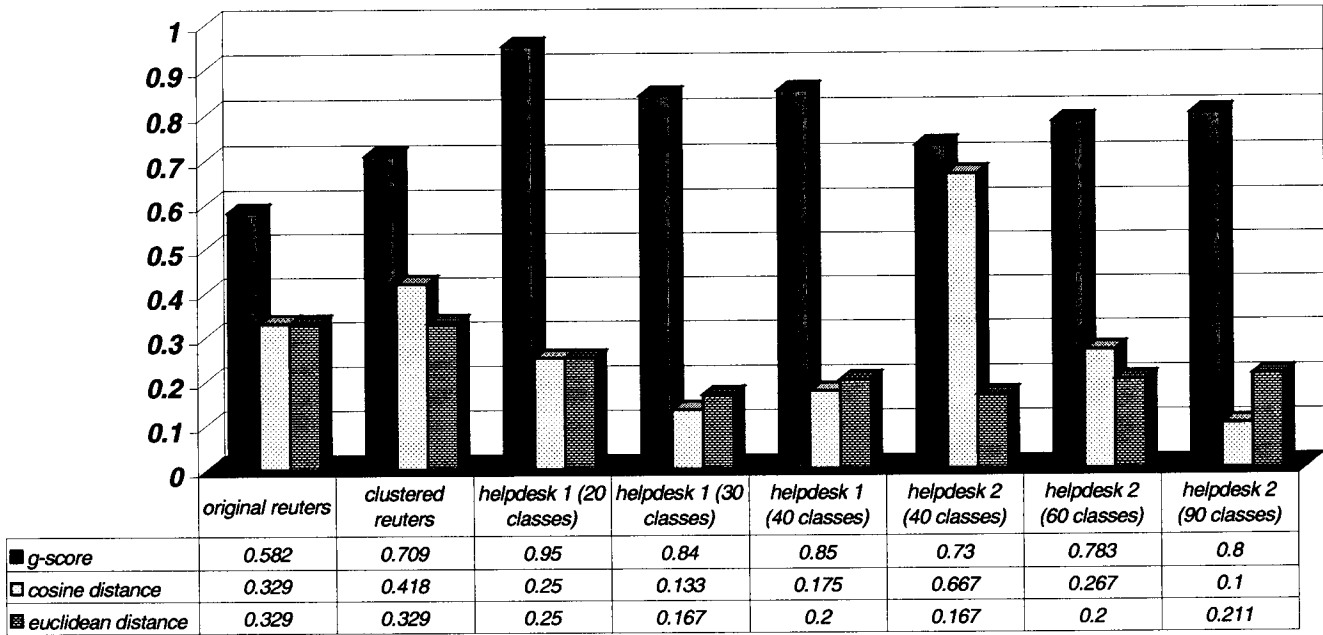
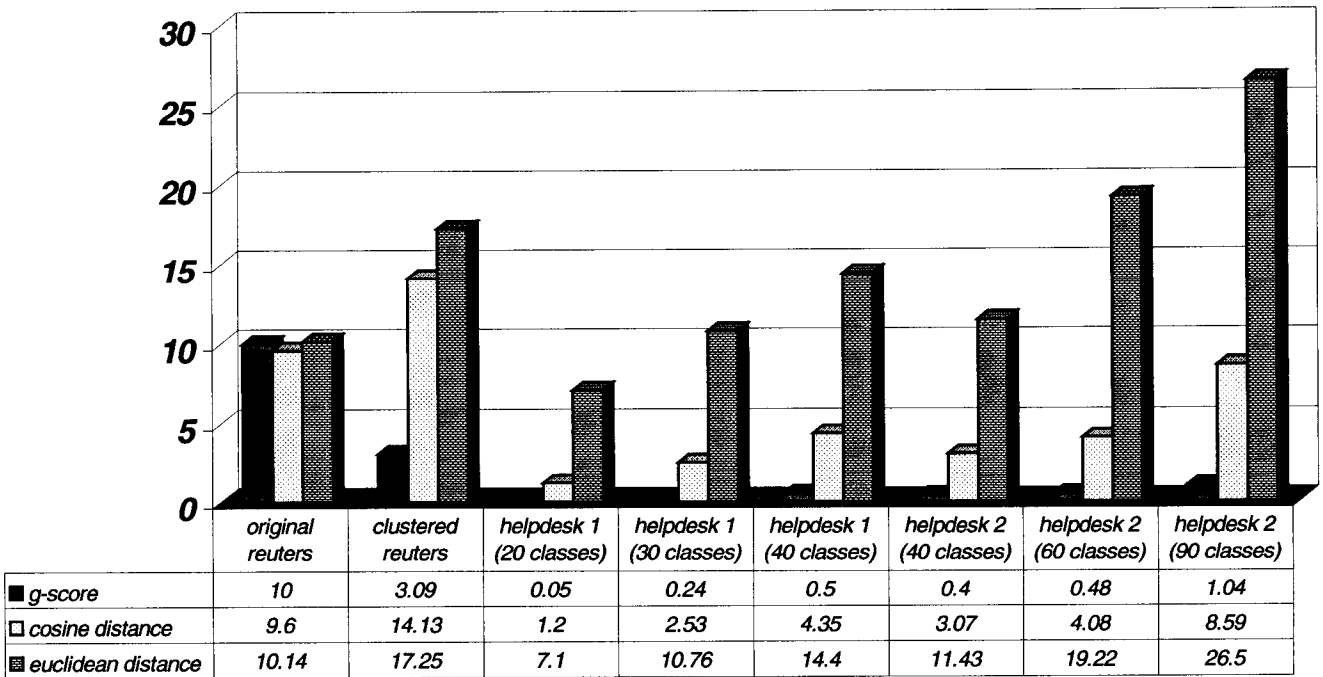


Figure 2: Comparing Gap Metrics by Average Rank



Multiple values for  $k$  (the number of clusters) were applied to the helpdesk datasets to insure that the results were not sensitive to this parameter. For each data set, we evaluate each of the gap analysis metrics by creating  $k$  solution sets, one for each class in the taxonomy of the data. Each solution set is nothing more than the original data set without the documents comprising one of the classes. The best result for each of  $k$  executions of the gap analysis algorithm is to correctly identify the missing class of data by giving the cluster corresponding to the missing data the lowest score. In analyzing the results we calculated two measures of goodness for each gap analysis approach. In figure 1, *accuracy* is the percentage of trials where the correct answer was the class with the lowest metric score, and in figure 2, *average rank* is the position of the missing class in the list of classes sorted from lowest to highest metric score (0.0 being the best possible). Thus the baseline, or value of a metric which always gave the lowest score to the first cluster, would be  $accuracy = 1/k$ ,  $average\ rank = (k-1)/2$ .

The results clearly show that the g-score metric was the best overall indicator of the actual knowledge gap. It had the best accuracy rate for all eight data sets and the best average rank in seven of them. The one case where g-score does worse is the Reuters data set using the original classification of the documents. Since for this data set each document is not necessarily in the class of the nearest centroid to that document, it is not surprising that the g-score approach does somewhat worse.

The simple Cosine distance metric did poorly overall, especially when measured by accuracy. A detailed analysis revealed that the cosine distance algorithm tended to select the same cluster in every trial. This cluster was invariably the one with the lowest cohesion. Since the g-score formula makes a correction for classes with low cohesion, it does not seem to suffer as greatly from this problem.

Additionally, we have tested our approach in an operational help desk environment. IBM helpdesk personnel applied the Knowledge Gap Analysis software to a local helpdesk data set from the IBM Almaden Research Lab. The helpdesk personnel analyzed 9 months worth of data, creating 30 classes via clustering. They then performed Knowledge Gap Analysis on an intranet knowledge base. They focussed their efforts on the clusters having the three lowest g-scores. Additional analysis of the problem tickets in these three clusters revealed twenty potential solutions that should be added to the knowledge base. The total time required for this analysis was a few hours, whereas before such a study would have taken weeks of an expert's time.

## 5. FUTURE WORK AND OTHER POTENTIAL APPLICATIONS

Knowledge Gap Analysis has many potential applications beyond helpdesk problem ticket analysis. Virtually any domain with unstructured problem descriptions and a knowledge base of text solutions will be amenable to this type of analysis. One such example would be to order the results of a keyword search query so that documents which are most unlike any previously seen documents are listed first.

Another potential application of this approach as an adjunct to web crawlers to search for areas of the web with content that is most unlike anything seen before.

## 6. ACKNOWLEDGEMENTS

The authors gratefully acknowledge Mike Lamb for first suggesting the problem which Knowledge Gap analysis solves and Tracy Knoblauch, our first user of the technology who provided invaluable feedback.

## 7. REFERENCES

- [1] Brachman, R. and Anand T. (1996). The Process of Knowledge Discovery in Databases. In Fayyad, U.M., Piatetsky-Shapiro, G., Smyth, P., and Uthurusamy, R., editors, *Advances in Knowledge Discovery and Data Mining*, Chapter 2, pages 37-58. AAAI/MIT press.
- [2] Duda, R. O. and Hart, P. E. (1973). *Pattern Classification and Scene Analysis*. Wiley.
- [3] Hartigan, J. A. (1975) *Clustering Algorithms*. Wiley.
- [4] D. D. Lewis (1999). Reuters-21578 text categorization test collection distribution 1.0. <http://www.research.att.com/~lewis>.
- [5] Rasmussen, E. (1992). Clustering algorithms. In Frakes, W. B. and Baeza-Yates, R., editors, *Information Retrieval: Data Structures and Algorithms*, pages 419-442. Prentice Hall, Englewood Cliffs, New Jersey.
- [6] Salton, G. and McGill, M. J. (1983). *Introduction to Modern Retrieval*. McGraw-Hill Book Company.
- [7] Salton, G. And Buckley, C. (1988). Term-weighting approaches in automatic text retrieval. *Information Processing & Management*, 4(5):512:523.