RC22315 (W0201-111) January 24, 2002
Mathematics/Computer Science

IBM Research Report

Experimental Evaluation of Taxonomy Mapping Algorithms

Guy T. Hochgesang, Bhavani Iyer, Bernard S. Landman, Zvi P. Weiss
IBM Research Division
Thomas J. Watson Research Center
P.O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Delhi - Haifa - India - T. J. Watson - Tokyo - Zurich

Experimental Evaluation of Taxonomy Mapping Algorithms

Guy Hochgesang, Bhavani Iyer, Bernard Landman, Zvi Weiss
IBM T. J. Watson Research Center
P. O. Box 704
Yorktown Heights, NY 10598

Abstract

We investigated several algorithms for automatically mapping one topical taxonomy onto another, assuming that each taxonomy is populated with documents. We also devised an experimental method of evaluating taxonomy mapping algorithms. This paper describes the taxonomy mapping algorithms, the experiments to evaluate them, and the results of the experiments. We conclude that a Centroid-like algorithm produces the best results when mapping one taxonomy onto a very similar taxonomy, but that a kNN-like algorithm performs better when mapping one taxonomy onto a dissimilar taxonomy.

Introduction

Taxonomies are used to navigate large document repositories. A well-known example is the Yahoo taxonomy for browsing documents on the worldwide web. Given a particular large document repository, different users of that repository may prefer to use different taxonomies to navigate through the documents. For example, in a large corporate document repository, the marketing and sales representatives may prefer to navigate with a taxonomy organized by lines of business, while the engineering staff may prefer a technical subject taxonomy.

This paper describes the initial results of experiments with several algorithms for automatically mapping one topical taxonomy onto another. It also describes an experimental means of evaluating a taxonomy mapping.

Taxonomy Mapping

First, let us define what we mean by *taxonomy mapping*. Suppose that we have two taxonomies which we will call the *adjunct taxonomy* and the *base taxonomy*. A mapping of the adjunct taxonomy onto the base taxonomy maps each adjunct taxonomy category onto zero, one, or more of the base taxonomy categories. The adjunct taxonomy may then be used to navigate the documents of the base taxonomy by following the mapping of an adjunct category onto base categories.

In order to use our techniques to perform such a taxonomy mapping, the adjunct and base taxonomies are subject to the following constraints.

1. The adjunct and base taxonomies must both be populated with document sets. These document sets should be adequate as "training sets," as described in [1].

2. As in [1], the labels of the categories are not used in the mapping. The mapping is performed on the basis of the content of the documents in each category. Thus categories which do not contain documents are not mapped. In this regard, the term *category mapping* might be more appropriate than *taxonomy mapping*.
3. If either taxonomy has a hierarchical structure, that structure is not mapped.

The Experiment

Suppose we have devised an algorithm to perform taxonomy mapping, as described above. We now describe how to evaluate experimentally such a taxonomy mapping algorithm.

Take one of the standard test collections of documents, such as REUTERS-21578 [1]. Following the notation of [1], this test collection consists of a set of categories $C = \{c_1, \ldots, c_m\}$, a set of documents $D = \{d_1, \ldots, d_n\}$, and a correct decision matrix ca_{ij}, $i = \{1, \ldots, m\}$ and $j = \{1, \ldots, n\}$. If $ca_{ij} = 1$, then document d_j should be categorized in category c_i [according to the judgments of human experts]. If $ca_{ij} = 0$, then document d_j should not be categorized in category c_i.

Next, as in [1], divide the document set D into two sets, which we will call the *adjunct document set*, *AD*, and the *base document set*, *BD*. Obviously, the correct decision matrix can be divided into adjunct and base decision matrices. These two decision matrices tell how the documents of the adjunct and base document sets should be classified into the original category set, C. In our experiments we used the REUTERS-21578 document collection and divided it according to the Mod-Apte split [2], with the adjunct document set being the test documents, and the base document set being the training documents. Because some of our categorization tools (i.e., the kNN classifier and ATG) do not handle documents which are not assigned to any category, we removed from the REUTERS-21578 document collection any documents which the experts did not assign to any category. For brevity, we will use the term "Reuters" to mean "REUTERS-21578, divided by the Mod-Apte split."

The first step of the experiment was a basic "sanity check," testing if the taxonomy mapping algorithm to be evaluated performs reasonably. Create an *adjunct category set AC* and a *base category set BC* which are the same as the original category set C (i.e., $AC \equiv BC \equiv C$). Next, the taxonomy mapping algorithms were used to map the categories of the adjunct category set onto the categories of the base category set, using the documents of the adjunct and base document sets. Any reasonable taxonomy mapping algorithm should probably result in the "identity" mapping (i.e., each adjunct category maps to the identical base category), or something very close to it.

We note that the mapping of the adjunct category set onto the base category set can be regarded as a classification of the documents from the adjunct document set into the categories of the base category set. That is, a mapping of an adjunct category onto a base category describes a classification of the documents of the adjunct category into the base category. Since we have the correct decision matrix for the adjunct document set, we can quantitatively measure the effectiveness of the classification of the adjunct documents by computing the *precision* and

recall, as in [1]. Since there is usually a tradeoff between precision and recall (i.e., increasing precision usually decreases recall), we use a measure called *F1*, which combines precision and recall, to measure the effectiveness of the taxonomy mapping [1].

The next step in the experiment was to use an adjunct taxonomy which was different from the base taxonomy. There are many possible ways to derive a different adjunct taxonomy. In our experiment, we chose to use the Automatic Taxonomy Generation ("ATG" for short) feature of the Lotus Knowledge Discovery Server [3, 4, 5]. We processed the test documents of the Reuters collection with the Knowledge Discovery Server. The Knowledge Discovery Server automatically generated a taxonomy and classified the test documents into the categories of that taxonomy. We then used the ATG taxonomy and the test documents as our adjunct taxonomy. The taxonomy mapping algorithms were used to map the ATG adjunct taxonomy into the original Reuters base taxonomy. Again, the category mapping can be viewed as a classification of the adjunct documents into the base categories. Since the base categories were not modified from the original categories, again we have a correct decision matrix for the classification, and again we can compute precision and recall for the classification.

Taxonomy Mapping Algorithms

This section describes the first two taxonomy mapping algorithms which we investigated.

kNN Taxonomy Mapping

The first algorithm is called a "kNN Taxonomy Mapping" algorithm, because it operates in a manner similar to a kNN document classifier [6]. It is assumed that there is a document classifier[1] which, given a document from the adjunct taxonomy, returns a list of base taxonomy categories relevant to that document and a relevance score for each base category. Relevance scores are assumed to be in the range 0.0 to 1.0, with 0.0 meaning no relevance and 1.0 meaning maximum relevance.

In order to map an adjunct taxonomy category onto the base taxonomy, the kNN Taxonomy mapping algorithm invokes the document classifier for each document in the adjunct category, thus obtaining a list of base categories relevant to that document. For each base category, the relevance scores are summed to compute a *similarity score* for the adjunct category to the base category. The similarity scores are normalized to be in the range 0.0 to 1.0. A *policy* then uses the similarity scores to decide to which base categories the adjunct category maps. The policies tested are described in a following section.

The kNN algorithm is described in more detail in Appendix 1.

[1] The document classifier does not need to be a kNN classifier, although the classifier which we used in our experiments was a kNN classifier [6].

Centroid Taxonomy Mapping

The Centroid Taxonomy Mapping Algorithm is similar to the "Centroid-Based Document Classification" algorithm described in [7]. Each document is represented by its *tf-idf* (*term frequency-inverse document frequency*) vector [7, 1]. The *centroid vector* of a category is the average of the td-idf vectors of the documents contained in the category.

Once the centroid vectors of the adjunct and base categories are calculated, the similarity score of an adjunct category *A* to a base category *B* is the cosine of the angle between the centroid vectors of the categories.

Given an adjunct category *A*, for each base category *B*, the similarity score of *A* to *B* is calculated. As with the kNN algorithm, a policy is applied to the similarity scores to decide to which base categories the adjunct category is mapped.

The Centroid algorithm is described in more detail in Appendix 2.

Policies

As described above, a taxonomy mapping algorithm produces a list of similarity scores which describe the similarity of each adjunct category to zero, one, or more base categories. A policy is then applied to decide to which base categories the adjunct categories are mapped. Separating the policy decision from the generation of similarity scores facilitates experimenting with various combinations of policies and taxonomy mapping algorithms. We now describe the policies that we evaluated.

Map By Threshold (MBT)

The *Map By Threshold* policy maps each adjunct category to each base category for which the similarity score exceeds a specified threshold. This results in each adjunct category being mapped to zero, one, or more base categories. In our experiments, we varied the threshold from 0.0 to 0.9 in increments of 0.1 and then chose the threshold which maximized F1.

Map Top Category (MTC)

The *Map Top Category* policy maps each adjunct category to that base category which has the highest similarity score. Thus, each adjunct category is mapped to exactly one base category.

Results of Experiments

Reuters-To-Reuters Results

In our first experiments, we tried mapping the Reuters taxonomy onto itself. More exactly, we used the test documents of the REUTERS-21578 Mod-Apte split as the adjunct document set and the training documents as the base document set. Note that the set of categories containing test documents is not exactly the same as the set of categories containing training documents: there are 93 categories containing test documents and 115 categories containing training documents; thus, a perfect "identity" mapping of each adjunct category onto a corresponding base category is not possible.

Chart 1 shows the results of mapping the Reuters taxonomy onto itself, using the Centroid mapping algorithm. The values of F1, precision, and recall are plotted for the Map By Threshold policy, varying the threshold from 0.0 to 0.9. Note that, as expected, precision increases and recall decreases as the threshold is increased. For Threshold = 0.0, an adjunct category is mapped to a base category if it has even a slight similarity (as measured by the centroid vectors). This results in excellent recall (there are very few "False Negatives") but extremely poor precision (there are many "False Positives"). Conversely, for Threshold = 0.9, recall is very poor but precision is good.

However, the best value of F1 is obtained not by the Map By Threshold policy, but by the Map Top Category policy. The results of the Map Top Category policy are shown in the rightmost column of Chart 1, above the label "MTC." Note that the values of F1, precision, and recall for Map Top Category are all so close to 1.0 that they are superimposed on each other at the top of the MTC column. However, the Map Top Category policy did not produce a perfect "identity" mapping. In the instances where the Map Top Category policy did not map an adjunct category onto the identical base category, it mapped onto a closely related category. For example, the category "soy-meal" was mapped to "soybean."

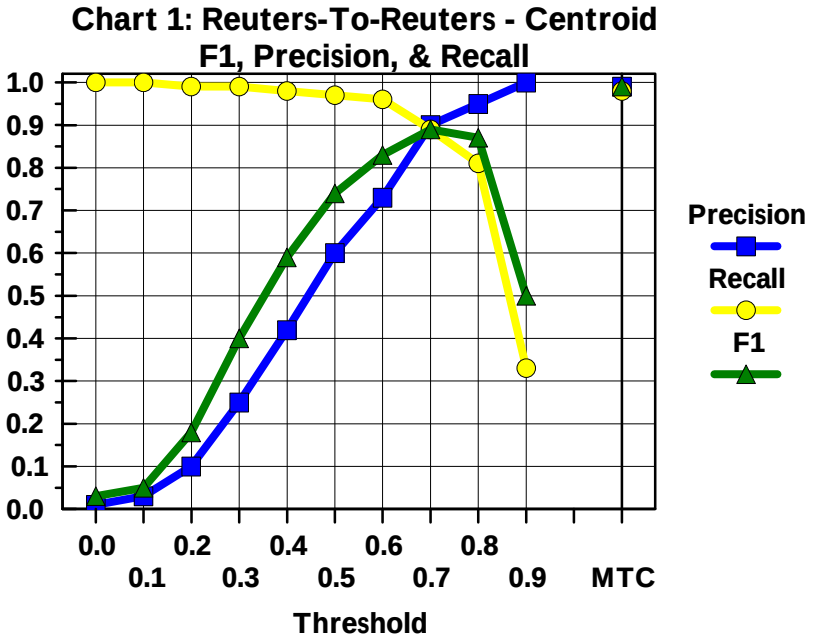


Chart 2 shows how the number of adjunct categories mapped to base categories decreases as the threshold of the Map By Threshold policy becomes more selective. At Threshold = 0.0, every one of the 93 adjunct categories is mapped to some base category. At the maximum F1 value for the Map By Threshold policy (F1 = 0.89 at Threshold = 0.7), only 37 of the 93 adjunct categories are mapped to base categories. For the Map Top Category policy, every one of the 93 adjunct categories is mapped to a base category.

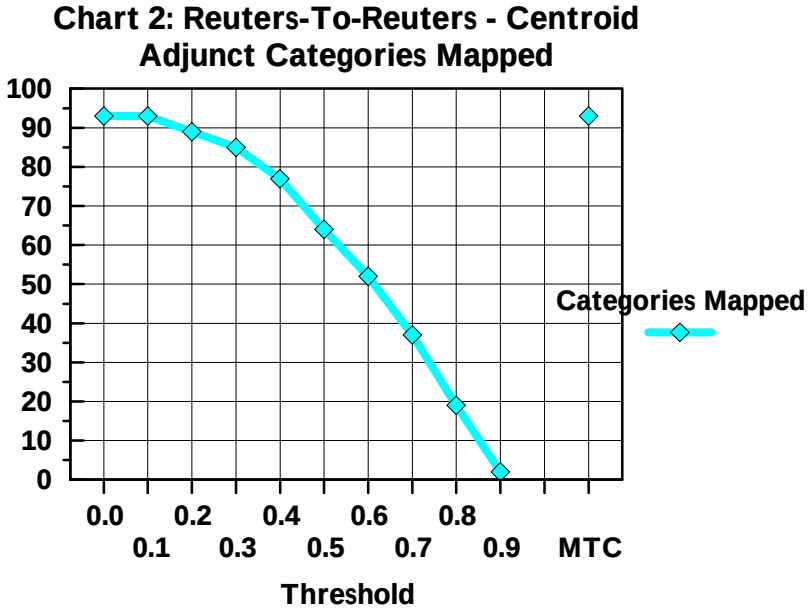


Chart 3 shows the results of the Reuters-To-Reuters mapping, using the kNN mapping algorithm. Once again, F1, precision, and recall are plotted for the Map By Threshold policy and the Map Top Category policy. The maximum F1 value for the Map By Threshold policy (F1 = 0.88 at Threshold = 0.5) is very close to the F1 value for the Map Top Category policy (F1 = 0.87).

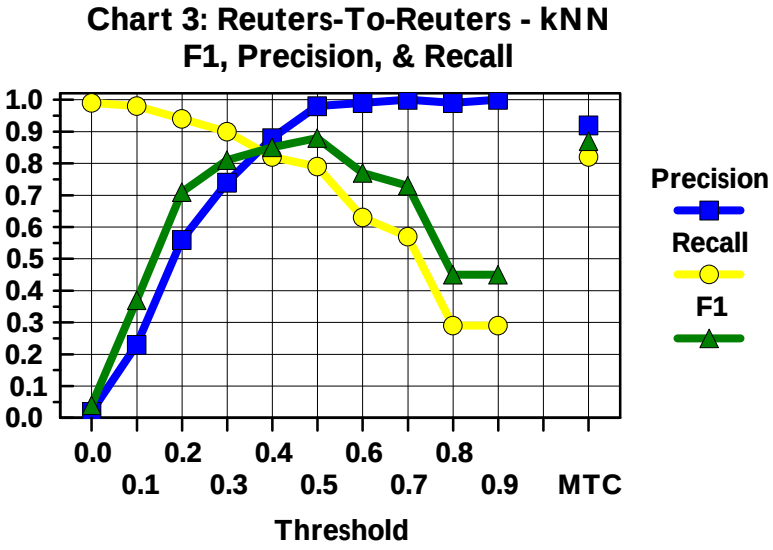
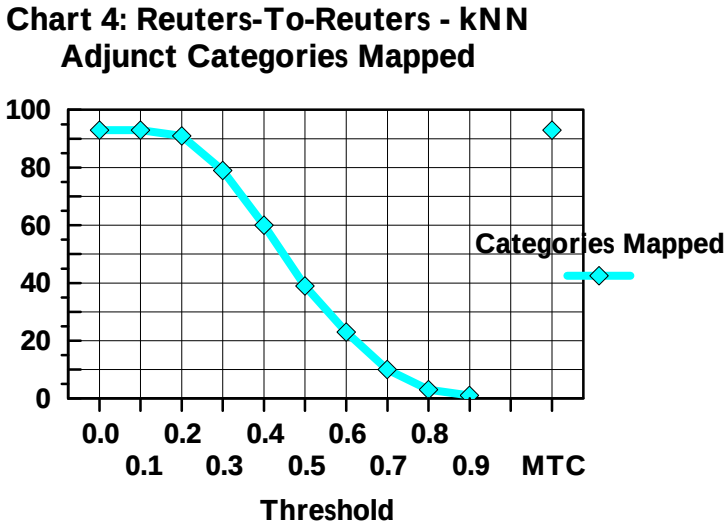


Chart 4 shows the number of adjunct categories mapped to base categories for the kNN mapping algorithm. At the maximum F1 value for the Map By Threshold policy (F1 = 0.88 at Threshold = 0.5), 39 of the 93 adjunct categories are mapped to base categories.



The results of the Reuters-To-Reuters experiments are summarized in Table 1. F1 is used as a measure of the effectiveness of the mapping algorithm (Centroid vs. kNN) combined with the policy (Map Top Category vs. Map By Threshold).

Table 1: Reuters-To-Reuters Results			
Algorithm - Policy	**F1**	**Precision**	**Recall**
Centroid - Map Top Category	0.99	0.99	0.98
Centroid - Map By Threshold = 0.7	0.89	0.90	0.89
kNN - Map By Threshold = 0.5	0.88	0.98	0.79
kNN - Map Top Category	0.87	0.92	0.82

ATG-To-Reuters Results

As described previously, we used the Automatic Taxonomy Generation ("ATG") feature of the Lotus Knowledge Discovery Server to create an adjunct taxonomy from the test documents of the Reuters collection. The ATG taxonomy contained 94 categories, compared to the 93 categories of the Reuters collection. Many of the ATG categories are noticeably different from the Reuters categories. For example, ATG created a category named "crop," while Reuters has categories such as "grain," "cocoa," "wheat," "corn," "sugar," "rubber," "orange," etc. We did not attempt to analyze the differences between the ATG and Reuters taxonomies, since our goal was to investigate taxonomy mapping algorithms, not taxonomy generation.

Chart 5 shows the results of using the kNN algorithm to map the ATG taxonomy to the Reuters taxonomy. F1, precision, and recall are plotted for the Map By Threshold policy and the Map Top Category policy. The maximum F1 value for the Map By Threshold policy (F1 = 0.520 at Threshold = 0.5) is very close to the F1 value for the Map Top Category policy (F1 = 0.517).

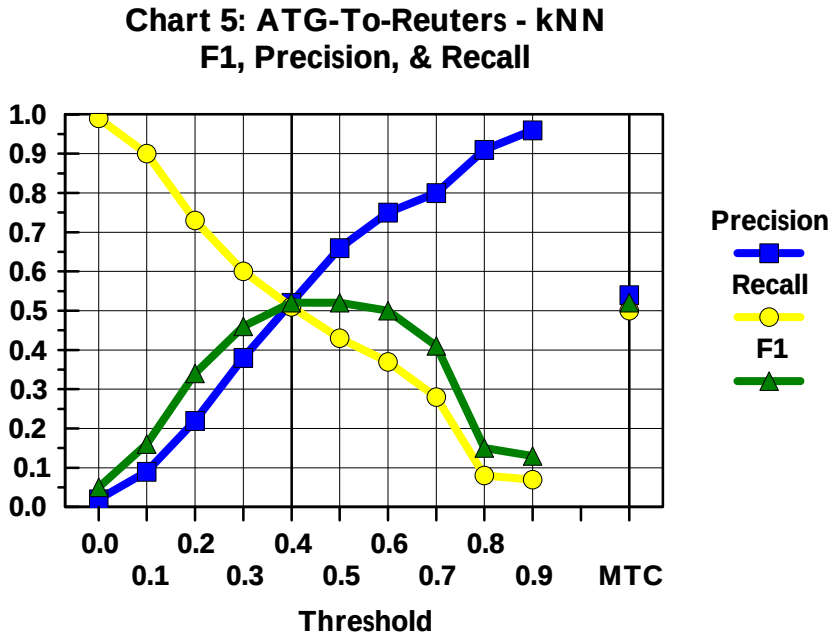


Chart 6 shows how the number of adjunct categories mapped to base categories decreases as the threshold of the Map By Threshold policy becomes more selective. At the maximum F1 value for the Map By Threshold policy (F1 = 0.52 at Threshold = 0.5), 59 of the 94 adjunct categories are mapped to base categories. For the Map Top Category policy, every one of the 94 adjunct categories is mapped to a base category.

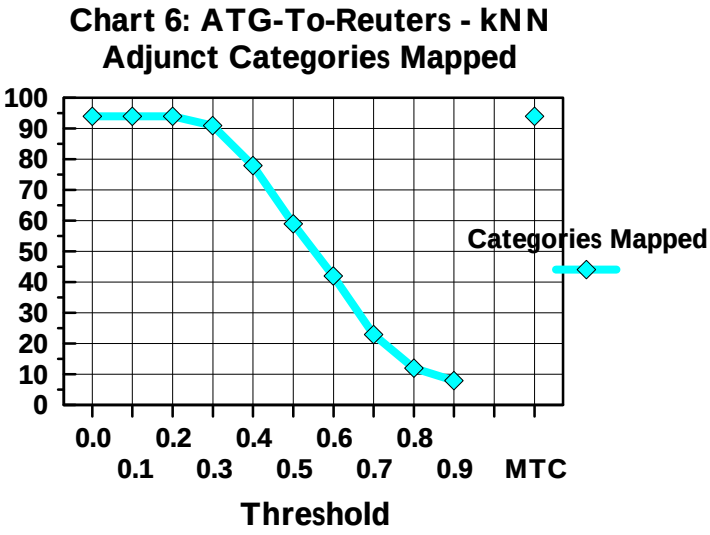


Chart 7 presents the results of the ATG-To-Reuters mapping using the centroid algorithm. The best value of F1 (F1 = 0.42) was produced by the Map Top Category policy. Using the Map By Threshold policy, Threshold = 0.9 prevents *any* adjunct category from being mapped to a base category; thus F1, precision, and recall are all zero.

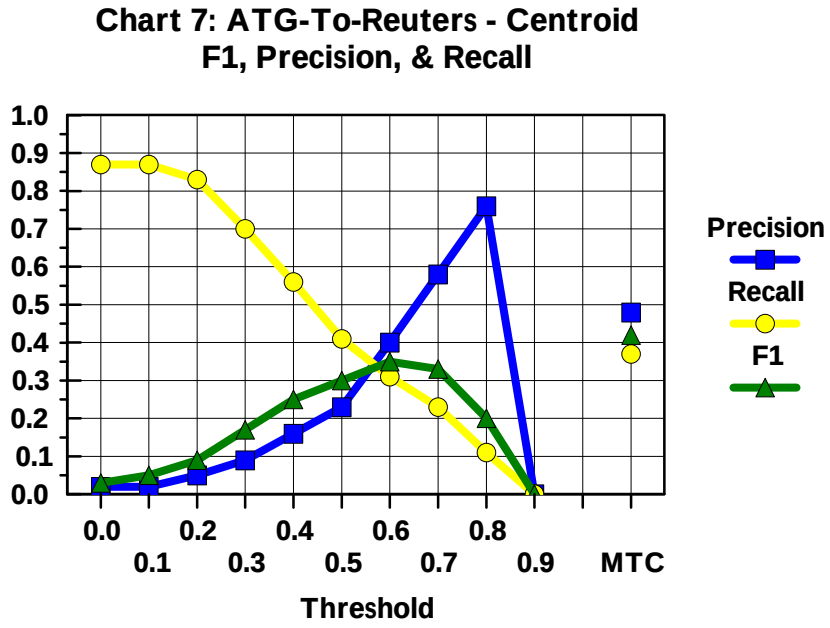


Chart 8 shows how, as the Threshold of the Map By Threshold policy becomes more restrictive, the number of adjunct categories mapped to base categories decreases.

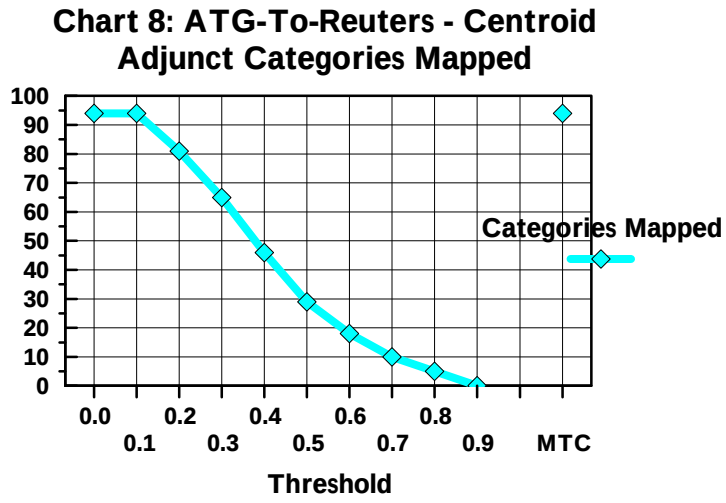


Table 2 presents the results of experiments with the ATG-To-Reuters taxonomy mapping. The different combinations of mapping algorithm (kNN vs. Centroid) and policy (Map By Threshold vs. Map Top Category) are shown in descending order of effectiveness, as measured by F1.

Table 2: ATG-To-Reuters Results			
Algorithm - Policy	**F1**	**Precision**	**Recall**
kNN - Map By Threshold = 0.5	0.520	0.66	0.43
kNN - Map Top Category	0.517	0.54	0.50
Centroid - Map Top Category	0.42	0.48	0.37
Centroid - Map By Threshold = 0.6	0.35	0.58	0.23

Conclusion

As shown in Table 1, the Centroid mapping algorithm combined with the Map Top Category policy performed the best in mapping the Reuters taxonomy onto itself, as indicated by the near-perfect scores for F1, precision, and recall. Thus, we conclude that the Centroid-Map Top Category combination is best (of the algorithm-policy combinations we tested) for mapping a taxonomy onto a very similar taxonomy.

To test algorithms for mapping a taxonomy onto a different taxonomy, we used the Automatic Taxonomy Generation (ATG) feature of Lotus Knowledge Discovery Server to generate a taxonomy from the test documents of the Reuters collection. We then used our taxonomy mapping algorithms to map the ATG taxonomy onto the Reuters taxonomy. As shown in Table 2, the kNN mapping algorithm combined with the Map By Threshold policy (with Threshold = 0.5) produced the most effective mapping.

Appendix 1: kNN Taxonomy Mapping Algorithm

For each category ***A*** in the adjunct taxonomy:

If category ***A*** contains no documents:

category ***A*** does not map to any category in the base taxonomy.

Else:

Set $N = 0$.

For each category ***B*** in the base taxonomy:
Set $S(B) = 0.0$.
[***S*** is an associative array, indexed by base category name, which will contain the similarity score of adjunct category ***A*** for each base category ***B***.]

For each document ***D*** in category ***A***:

Invoke the categorizer to categorize document ***D*** in the base taxonomy. The categorizer returns pairs (C_i , R_i), $i = 1, 2, \ldots, n$, $n \geq 0$, where each C_i is the name of a category in the base taxonomy and each R_i is the relevance score of document ***D*** to category C_i, where $0.0 \leq R_i \leq 1.0$.

For $i = 1, 2, \ldots, n$:
$S(C_i) = S(C_i) + R_i$

$N = N + n$

For each category ***B*** in the base taxonomy:
Set $S(B) = S(B) \div N$

[This normalizes each $S(B)$ such that $0.0 \leq S(B) \leq 1.0$. It also has the property that if every document ***D*** in adjunct category ***A*** has a relevance score of 1.0 for base category ***X*** and a relevance score of 0.0 for every other base category, then $S(X) = 1.0$ and for $Z \neq X$, $S(Z) = 0.0$.]

The array ***S*** now contains the similarity score of adjunct category ***A*** for each base category ***B*** in the base taxonomy. These similarity scores are then used by a policy to determine the base categories onto which adjunct category ***A*** is mapped.

Appendix 2: Centroid Taxonomy Mapping Algorithm

The centroid algorithm employs the vector-space model for representing documents and centroids. We now introduce several definitions and notational conventions to facilitate the algorithm description in terms of the model.

We indicate a vector quantity (document or centroid) in boldface, e.g., $\boldsymbol{A}$. We display the components of a vector as a parenthesized ordered sequence:

$$\boldsymbol{A} = (a_1, a_2, ..., a_n)$$

We denote the dot (or scalar) product of two vectors $\boldsymbol{A}$ and $\boldsymbol{B}$ as

$$\boldsymbol{A} \bullet \boldsymbol{B} = \sum_{i=1}^{n} a_i b_i$$

The above expression is predicated on the implicit assumption that *both* vectors are defined over the same vector space, i.e., the components of both vectors are measured along a *common* set of basis vectors that span the space. This needs to be generalized to accommodate the case where the two vectors are defined over spaces which are different, but which *do* contain some common basis vectors. To do this we postulate the existence of a translation function, $\Im_{AB}$, which - given the index of a component of $\boldsymbol{A}$ - returns the index of the corresponding component of $\boldsymbol{B}$, if it exists, or returns some invalid index if there is no corresponding component in $\boldsymbol{B}$. Now the dot product may be written as

$$\boldsymbol{A} \bullet \boldsymbol{B} = \sum_{i=1,\ j=\Im_{AB}(i)}^{n} a_i b_j$$

where $a_i b_j = 0$ if j is an invalid index.

We measure the length of a vector using the L_2 (Euclidean) norm:

$$\|\boldsymbol{A}\|_2 = \left(\sum_{i=1}^{n} |a_i|^2 \right)^{\frac{1}{2}}$$

We indicate the number of elements of a set S by use of the symbol $|S|$.

Having established the notational conventions above, we now present the description of the algorithm.

For each document d in each taxonomy, generate the vector representation of that document $\hat{\boldsymbol{D}}$ in term-frequency space (each component of the document vector is a tf-idf value [7,1] for that document):

$$\hat{\boldsymbol{D}} = (tf_1, tf_2, ..., tf_n)$$

For each document vector $\hat{\boldsymbol{D}}$ calculate the corresponding normalized document vector $\boldsymbol{D}$ by dividing each of the components of $\hat{\boldsymbol{D}}$ by the norm of $\hat{\boldsymbol{D}}$:

$$\boldsymbol{D} = (tf_1, tf_2, ..., tf_n) \div \left\| \hat{\boldsymbol{D}} \right\|_2 = \hat{\boldsymbol{D}} \div \left\| \hat{\boldsymbol{D}} \right\|_2$$

This results in $\|\boldsymbol{D}\|_2 \equiv 1$ for each normalized document vector and so the length of a document d does not influence the centroid calculation.

For each category C_i in each taxonomy, calculate its centroid vector $\boldsymbol{C}_i$ as the average of the normalized document vectors corresponding to the documents in that category (using vector summation):

$$\boldsymbol{C}_i = \left(\sum_{d \in C_i} \boldsymbol{D} \right) \div |C_i|$$

At this point we have calculated a centroid vector for each category of each taxonomy. Note in passing $\|\boldsymbol{C}_i\|_2 \leq 1$ since we are averaging normalized vectors.

Anticipating we will be using the dot product between centroid vectors from the two taxonomies to measure category similarities, we need to realize the postulated index translation function $\Im_{AB}$ introduced earlier. We first introduce some practical considerations. When representing documents using the vector space model, the basis vectors are the so-called terms - words, word fragments, or short phrases - that appear in the document. The tf-idf value of a document vector component is proportional to the number of times the corresponding term appears in that document. The generation of document vectors for a given taxonomy is conceptually done by first collecting all terms over all documents in that taxonomy, which produces a "master" term list (i.e., the set of basis vectors which span the taxonomy's vector space). Then a determination is made, for each document, which - and how many - of these terms it contains. Returning to the implementation issue for the function $\Im_{AB}$, we want to take dot products of (centroid) vectors from two different taxonomies - whose master term lists are usually different - so we proceed as follows:

(1) Identify those terms which are common between the two master term lists.

(2) For each common term, take its master term list index in the first taxonomy as an element of the function's domain (i.e., as an argument) and its master term list index in the second taxonomy as the corresponding element of the function's range (i.e., as its value).
(3) For each non-common term in the master term list of the first taxonomy, its list index becomes an element of the function's domain and we force the corresponding element of the function's range to be some index which is invalid for the master term list of the second taxonomy (e.g., -1).

With $\Im_{AB}$ implemented we are now equipped to compute meaningful dot products.

We next calculate, for each possible ordered pair consisting of an adjunct category C_A and a base category C_B, a similarity score for these two categories. We take this similarity score to be the cosine of the angle θ between the centroid vectors $\boldsymbol{C}_A$ and $\boldsymbol{C}_B$ corresponding to these categories; this cosine is calculated by the formula:

$$\cos\theta = (\boldsymbol{C}_A \bullet \boldsymbol{C}_B) \div (\|\boldsymbol{C}_A\|_2 \times \|\boldsymbol{C}_B\|_2)$$

At this point we have a similarity score between each category of the adjunct taxonomy and each category of the base taxonomy. We also note that since the term frequencies cannot be negative, each similarity score must lie in the range [0.0, 1.0].

Finally we apply a policy to these similarity scores - as we did with the kNN algorithm - which decides to which adjunct categories each base category will be mapped.

References

[1] Fabrizio Sebastiani. A Tutorial on Automated Text Categorization. *Proceedings of ASAI-99,* Analia Amandi and Ricardo Zunino, editors. 1st Argentinian Symposium on Artificial Intelligence, pages 7-35, Buenos Aires, AR, 1999

[2] David Lewis. The Reuters-21578 Text Categorization Test Collection, Distribution 1.0. README.txt file in http://www.research.att.com/~lewis/reuters21578.html

[3] Lotus White Paper. Building Enterprise Taxonomies with the Lotus Discovery Server. Lotus Development Corporation. 2001. Ftp://ftp.lotus.com/pub/lotusweb/product/discovery/KMT-2001-DS2.pdf

[4] Shivakumar Vaithyanathan and Byron Dom. "Model-Based Hierarchical Clustering". *The Sixteenth Conference on Uncertainty in Artificial Intelligence.* June 30 - July 3, 2000 at Stanford University; Stanford, CA.

[5] Think Research. Natural Selection. IBM Corporation. http://domino.research.ibm.com/comm/wwwr_thinkresearch.nsf/pages/selection200.html

[6] Think Research. Paw and Order on the Internet. IBM Corporation. http://domino.research.ibm.com/comm/wwwr_thinkresearch.nsf/pages/webcat498.html

[7] Eui-Hong (Sam) Han and George Karypis. Centroid-Based Document Classification: Analysis & Experimental Results. Technical Report #00-017, University of Minnesota, Department of Computer Science/Army HPC Research Center, Minneapolis, MN, 2000.