Textual Data Mining of Service Center Call Records

Pang-Ning Tan
Department of Computer Science
University of Minnesota
Minneapolis, MN 55455
ptan@cs.umn.edu

Hannah Blau
Department of Computer Science
University of Massachusetts
Amherst, MA 01003-4610
blau@cs.umass.edu

Steve Harp and Robert Goldman
Honeywell Technology Center
3660 Technology Drive
Minneapolis, MN 55418
steve.a.harp,robert.goldman@honeywell.com

ABSTRACT

In this project, we developed a technique for extracting useful information from databases that contain both fixed-format and free-text fields. The present state of the art in data mining is a schism between techniques that handle only fixed-format data (pattern recognition, classification algorithms from machine learning), and techniques designed for free-form text (information retrieval). Advanced knowledge discovery technologies have been developed in both research areas, but systems that can categorize or cluster records containing both kinds of data are still lacking. Specifically, we examined database records from a Honeywell service center to extract information about the expected cost of different kinds of service requests. Our goal was to test the hypothesis that incorporating information from free-text fields would provide a better categorization of these records; in this case, better predictions of the cost of the service call. In our work, we have integrated feature extraction and clustering techniques from information retrieval with classification algorithms from machine learning in order to categorize the hybrid fields. Our preliminary results suggested that incorporating free-form text could potentially induce better classification models.

Categories and Subject Descriptors

H.2.8 [**Database Management**]: Database Applications—*Data mining*

1. INTRODUCTION

Textual data mining is a rapidly growing application of knowledge discovery in databases (KDD) due to the ever-increasing volume of structured and unstructured documents produced by large organizations. In this project, we developed an automatic technique for extracting useful information from databases that contain both fixed-format and free-text fields. At present, most of the techniques developed in the area of data mining and information retrieval are designed to handle only fixed-format data or free-form text, but not a combination of both. There are many application domains in which both fixed-format and free-form data are available. For example, an e-commerce Web site may collect profiles of their customers as fixed-format data as well as users' reviews about their products in free-form text. Incorporating both types of information into the data mining task enhances our understanding of the domain, but also increases the complexity of the problem. In this project, we examined database records containing both types of data from a Honeywell technical assistance center, which provides telephone support for industrial control products such as distributed control systems, automatic valves, sensors, etc. We were interested in gaining insights about the nature of problems handled, and the expected cost of different kinds of service requests.

Service center directors would like to determine what types of cases are the most expensive to service (where cost is measured in terms of call frequency and call duration). A director can easily find out what it costs to provide support for each product, by summing over all cases with the same code in the product field. But knowing which are the most expensive products to support is not enough, the director has to know what makes them expensive to choose an appropriate corrective action. Are these cases due to hardware failures, poorly-written documentation, installation problems or configuration problems? These questions cannot be answered with a simple database query, because the information is buried in the engineers' notes. Our goal is to develop a classification model for service center case records containing both free-text and fixed-format data fields. This model will be useful for improving customer services as well as identifying flaws in existing products. For example, [4] developed a system to automatically categorize in-coming email messages at a call center in order to improve customer care. However, they used only free-form email messages to classify the calls into one of the 47 different categories. The accuracies of their system varied between 22% up to 56%, depending on the classification algorithms and linguistic preprocessing methods used.

In our work, we have combined techniques from information retrieval and machine learning into a novel method of categorizing hybrid fixed-format and free-text fields. Our initial results suggest that incorporating free-text information can potentially induce better classification models.

2. SYSTEM OVERVIEW

In this section, we present an overview of our service center textual data mining system (Figure 1). It is an off-line knowledge discovery system, consisting of 3 main components : data extraction, preprocessing and data mining modules. The functionality of each module is described below.

2.1 Data Extraction

At the Honeywell Service Center, all requests for customer support are recorded in a call tracking database. The information stored in this database contains both fixed-format and variable-format (free-text) fields. The fixed-format fields contain attributes such as the case number for each service request, type of problem encountered, the number of engineers who handled the case, product ID, customer information and the amount of time spent to resolve the problem. These fields have strict formatting restrictions in terms of attribute type, range and precision of the attribute values. For example, the total time field is measured up to one-tenth of an hour.

The free-text field contains a case description written by the engineers who handled the case. Because this field is unstructured, it may contain both relevant and irrelevant information for categorizing the particular case. This is further complicated by the different styles and languages used for writing the description. For example, some of the description may contain Spanish text, followed by an English translation of the text.

The data extraction module extracts only relevant features from the call-tracking database. These features were initially identified with the help of domain experts. Our data extraction module is implemented with SQL scripts to extract the important fields and store them in flat files for further processing.

2.2 Preprocessing and Feature Selection

To classify the case records, each record has to be converted to a vector in a feature space. For the fixed-format fields such as customer and product, the feature space is defined by their limited possible values. For free-text fields, statistics based on the frequencies of certain words and phrases form the feature space. The classification algorithm then uses the feature vector corresponding to the case record to evaluate some desired attribute of the case.

In our text mining system, the fixed-format fields require minimal preprocessing. For example, we translate the alphanumeric values of some of the fixed-format fields into suitable numeric ids, and consolidate the different variations of NULL values. For free-text fields, construction of the feature vector involves two steps : preprocessing and dimension reduction. Preprocessing of the free-text fields requires more robust techniques to handle problems such as spelling errors, abbreviations, multilingual text, etc. Figure 2 illustrates the steps needed to transform the raw free-text data into a feature vector representation.

The first step of preprocessing is text conversion, which converts a portion of the free-text description into a form more suitable for subsequent preprocessing steps. This includes removing irrelevant sentences within the free-text field (such as the signature or timestamp of email messages); transforming the different morphological variants of a word into the same lexical unit; handling common spelling mistakes, run-ons and word splits; substituting common expressions with keywords (such as replacing 128.08.10.10 with "IP address"), etc. This step was implemented with a set of conversion rules specifying the patterns to look for within the free-text field and the lexical units with which they should be replaced.

Tokenization breaks the free-text sentences into smaller units called tokens. Our system uses white-space characters as delimiters for a token. The disadvantage of this method is that multi-word phrases ("shut down", "IP address", etc.), are broken up into separate tokens. We reduce the severity of this problem by identifying the most common word phrases and replacing them with a single lexical unit during the text conversion step.

The term separation step partitions the tokens into five distinct categories : keywords, proper names, alphabetical strings, symbolic terms, and low-frequency terms. A list of keywords containing important product and parameter names is used to distinguish between keywords and non-keywords appearing in the free-text field. Irrelevant proper names, symbolic tokens (consisting of digits and other non-alphabetical symbols), and low-frequency tokens are removed during this step. A list of proper names of people and places is used to filter out irrelevant names from the list of tokens generated by the tokenization step. The low-frequency terms are removed by applying a user-specified threshold on the token list. Only keywords and alphabetical strings (tokens containing only alphabetical letters) are used to construct the feature vectors.

The alphabetical strings then undergo further preprocessing steps, which include conversion to lower case, stopword removal, spelling check, and stemming with Porter's suffix removal algorithm [10]. However, stemming may produce conflicts between the stemmed words and keywords. For example, the word *basically* will be stemmed to *BASIC*, a product name, while the stemmed word for *dies* may conflict with the product *DI*. As a result, we have to identify potential conflicts and circumvent the problem by manually undoing the stemmed words or adding new rules to the text conversion step. The filtered words are combined with the list of keywords found during the term separation step to form the set of features representing the free-text field. The last step of the free-text preprocessing stage involves the construction of a feature vector for every case description extracted by the data extraction module. For each term T_j of the description D_i, we calculate its weight w_{ij} according to the metric

$$w_{ij} = tf_{ij} \log \frac{N}{df_j}$$

where N is the total number of case records, term frequency tf_{ij} is the number of occurrences of T_j in D_i, and document frequency df_j is the number of case descriptions that contain T_j. Terms T_j of sufficiently high weight are retained as keywords for description D_i. This is a widely used measure in information retrieval [14, 13].

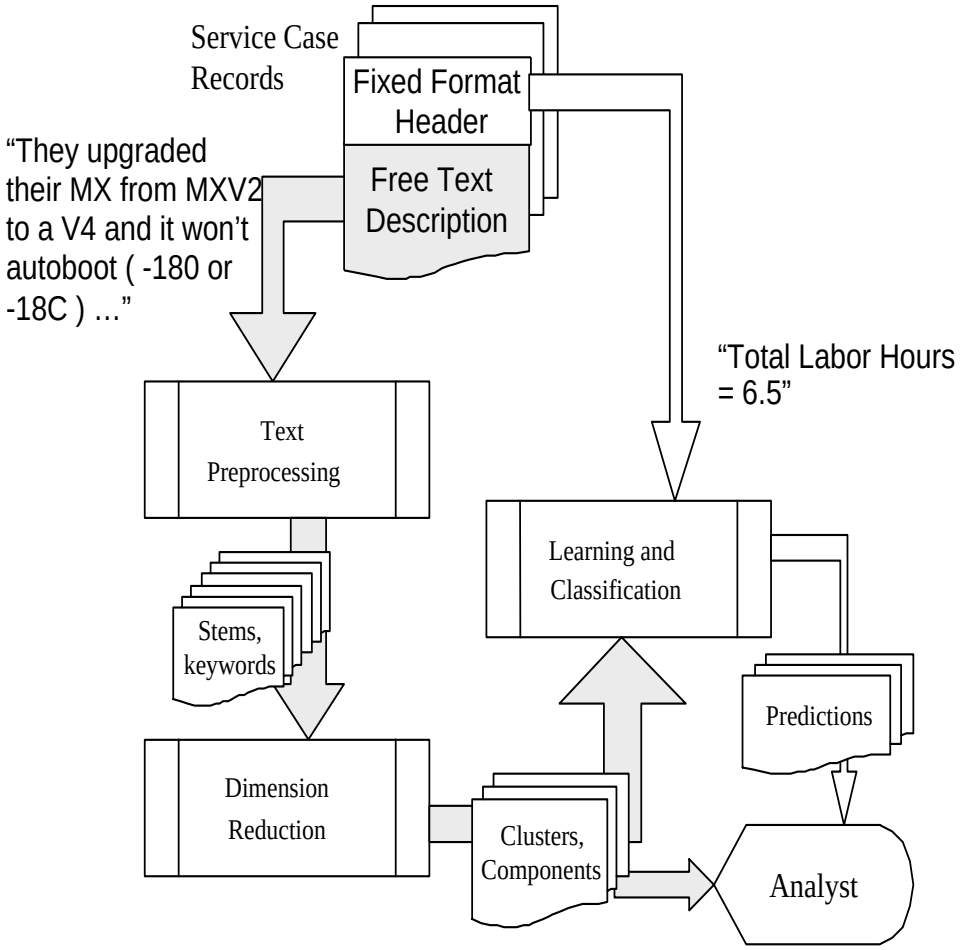

Figure 1: System Overview

From the terms selected in the preprocessing step, we derive the free-text feature space we need to classify the case records. Using each word T_j as a feature would lead to an unmanageable number of dimensions. Instead, we empirically reduce the dimensionality of the feature space by grouping together related terms. This strategy works because there are strong correlations between the original term features. By exploiting this redundancy, the dimension of the feature space can be reduced to mere hundreds. We used two different techniques for dimension reduction: singular value decomposition (SVD) [6] of the original matrix of feature vectors and term clustering using a hypergraph partitioning software, called hMETIS [8, 7]. SVD is a well-known method for dimensionality reduction because it is capable of extracting most of the salient features of the text documents and removing the noise present in the data. Clustering using hMETIS partitioning technique has been shown to produce higher quality clusters compared to traditional hierarchical agglomerative clustering techniques and Autoclass [3, 9]. In [3, 9, 5], hypergraphs were constructed using frequent itemsets generated by the Apriori algorithm [2, 1]. Each vertex would represent a term while each hyperedge would correspond to a frequent itemset. However, it is not clear what is the best way to assign an appropriate weight to the hyperedge. For instance, [3, 9] have suggested using the average confidence of association rules generated from the frequent itemset while [5] proposed using an *interest* factor. In contrast, our approach was slightly easier. From the original matrix of free-text features, we computed the similarity between each pair of words using an L_2 distance measure [1]. We sparsified the matrix by choosing the k-largest values for each row and setting the rest of the column entries to zero. We then assigned each non-zero entry in the sparsified matrix as the weight for the corresponding edge in our graph. Finally, we use hMETIS to partition the graph into highly-connected clusters of words.

Both SVD and graph partitioning techniques proved to be effective. The results of the dimension reduction phase (the hmetis-discovered or SVD-derived clusters) are themselves interesting, since they characterize the cases in a novel manner. For example, cluster f57 (Table 1) contains terms such as badpvfl, latch, partfail, softfail, switchover, swap, unet; these terms point to a special control network product family that features automatic switching to redundant units when a failure has been detected.

[1] This approach was suggested in [14]. Other similarity measures such as cosine measure can also be adopted.

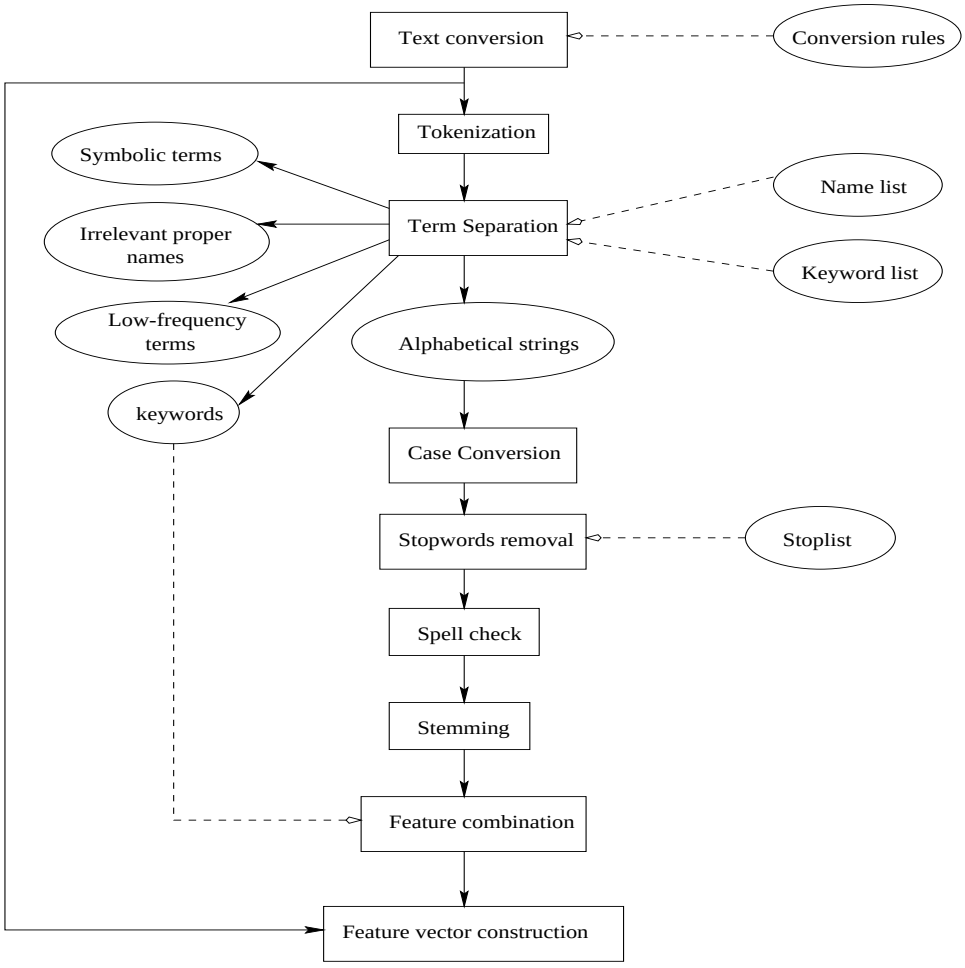

Figure 2: Preprocessing of free-text field

2.3 Pattern Discovery

As mentioned earlier, one of the main goals of our text mining system is to build a classification model using historical case records. The classifier supplies information that can be used to improve future customer support or existing products. Our text mining system uses the free-form text description and/or the fixed-format header information to predict whether a new case record would be resolved rapidly (within one hour) or would take a longer time. These two classes of cases are labeled *LESS* and *MORE*, respectively.

We have experimented with two inductive learning methods for building our classifier :

- Decision trees.
- Naive Bayesian.

We construct decision trees with the C4.5 algorithm [11]. The algorithm begins by considering the entire feature space as having the same class label. It then recursively partitions this space in a greedy manner to minimize impurity due to mixed class labels within a particular region in the feature space. C4.5 uses the information gain ratio as the criteria for splitting. The output of this algorithm is a decision tree in which each intermediate node denotes the attribute chosen to partition the feature space, and each leaf node is one of the possible class labels.

We used RoC[2] as our naive-Bayes classifier for this project [12]. Unlike decision tree induction algorithms, a naive Bayes classifier constructs a classification model by estimating the conditional probability of each class label given a new instance of the feature vector. The basic assumption in a naive Bayes classifier is that the classes are mutually exclusive and exhaustive, while the features are independent of each other. RoC handles only discrete variables; continuous variables are discretized. There are two ways to discretize continuous variables. The range between the minimum and maximum values can be partitioned into equally spaced intervals, or into intervals having the same number of cases.

We chose C4.5 and naive Bayes as our classification algorithms because they are commonly used in text categorization and their results are easy to understand. Other classification schemes that can be potentially useful include k-nearest neighbors, Support Vector Machines, etc. Compar-

[2]available at http://kmi.open.ac.uk/projects/bkd/

Table 1: Clusters obtained using hMETIS software.

Cluster	terms
f57	mls, mms, mmph, softfail, interpret, remaind, io, uldm, statechg, accident, ml, dn, mph, mml, ladder, mde, mpa, swap, right, flat, latch, badpvfl, iom, partfail, unet, extender, inadvert, aliv, iolink, mmpa, offnet, rcn, pref, siom, rack, circumst, switchover, full, consecut, hlai, odd, partner
f80	overwrit, loopback, flip, lit, introduc, hub, lrc, rippl, cnet, dh, eos, gh, ii, ny, st, teak, detector, recd, light, ckpt, valv, main, netway, cope, nic, diagnost, cmo, plcx, crm, femal, translator, pcig, decoder, octal, upset, ktx, instrum, dih, sam

isons with these classification methods will be for future research.

3. EXPERIMENTAL RESULTS

Our dataset contains 20,816 cases collected over a duration of one year. About 75% of the cases (in this sample) were resolved within one hour. We built the classification model on a subset of the available data and evaluated the different combinations of classifiers and dimension reduction techniques.

The best combination was provided by the naive Bayesian classifier with SVD (Table 2). We used SVD to decompose the original matrix of features, X, into two matrices of singular vectors, U and V, and a diagonal matrix of singular values, D :

$$X = UDV$$

By selecting k of the largest singular values in the diagonal matrix (and setting the rest of the entries to zero), we can obtain a new matrix $\tilde{X}$ which best approximates X in a least square sense. $\tilde{X}$ is obtained by deleting the rows and columns in U and V corresponding to small diagonal values in D. In our experiments, we had used $k = 100$. Notice that with fixed-format data, the classifier is biased towards the $LESS$ class labels. However, when free-form text is incorporated, a less-biased model is obtained.

With C4.5 classifier, the best result is obtained using hMETIS clustering. In this approach, each cluster becomes a new, composite feature of the dimensionally-reduced feature space. The C4.5 classifier with hMETIS consistently obtained a hit rate near 53% with a false alarm rate less than 10% — imperfect but well above chance and the results produced by fixed-format data. Figure 3 illustrates the hit rate (of $MORE$) and false alarm (Type II error) for all the classification models built using C4.5 (with hMETIS clustering). These models were generated after performing a 10-fold cross-validation on the dataset. Because this is an observational dataset, we have no analytical means to assess the maximum attainable prediction accuracy. In short, our experimental results revealed that there is some improvement in the predictive accuracy of classification models when free-form text is incorporated.

Table 3 shows an example of a decision tree built with C4.5 using both header and cluster data. The learned cluster features (100 of them) are all assigned arbitrary names, e.g. f12, f92. The initial splits are frequently on the header fields "call status" and "product" (the product family).

Cluster f80, highlighted in Table 3, is a good free-text indicator of longer ($MORE$) cases. Examination of f80 words shows it has a large proportion of computer network terms, such as "loopback", "hub", "nic", and "gateway". Cluster f86, another trouble indicator, includes some interesting terms: "overload", "fear", "sleep", "attack", "nervou", "explicit". A precise interpretation of these clusters is a job for domain experts. Our results indicate how the initial assembly can be significantly automated.

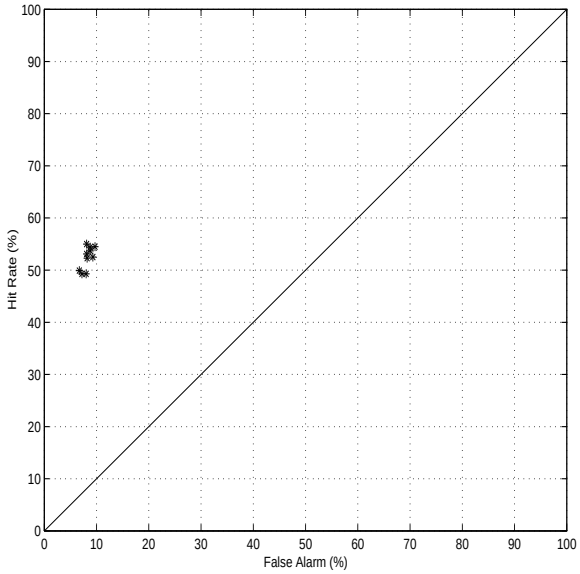

Figure 3: Hit rate versus false alarm rate for C4.5 classifer using mixed features (with hmetis clustering).

4. CONCLUSIONS

Our initial results show that incorporating free-text information in service databases can potentially improve the classification model. We are working with the service organization to further develop these results. These results come in two areas:

- the function that classifies the duration of a service case based on information in the database record, and
- information about the structure of the domain that is implicit in the clustering of terms.

Early conversations with our domain expert indicate that this clustering provides useful information about classes of service call. This information could be used either to im-

Table 2: Summary of classification results using various feature types and dimensional reduction strategies.

Classifier	dimension reduction	Fixed Format	Free-form Text	% Hit (MORE)	% Miss (MORE)	% Hit (LESS)	% Miss (LESS)
RoC	-	Yes	No	39.4	60.6	96.0	4.0
	SVD	Yes	Yes	79.3	20.7	62.5	37.5
C4.5	-	Yes	No	48.8	51.2	92.8	7.2
	Clustering	Yes	Yes	52.4	47.6	91.7	8.3
	SVD	Yes	Yes	43.2	56.8	94.0	5.1

prove handling of service calls, or to improve products (and their documentation) to avoid the need for support.

This research could help service centers to identify case categories that are expensive to service. At the present time, service centers cannot automatically analyze which types of cases are consuming the most engineering resources. Case categorization and cost summaries such as the ones we have illustrated could enable the service center to locate its problem areas, and to quantify the resources expended in those areas. This information can be used to solve the underlying problems that are generating service requests: poor documentation on a particular product, faulty hardware design, etc.

The methodology developed here could be applied to other service centers. Some effort would be required to adapt to a new domain (different database format, new acronyms), but the methodology would remain unchanged.

Acknowledgments

This project would not have been possible without the advice of our principal domain expert, Keith Curtis, Honeywell Hi-Spec Solutions. We would also like to thank George Karypis (hMETIS), Marco Ramoni (RoC) and Ross Quinlan (C4.5) for making available the software used in our experiments.

5. REFERENCES

[1] R. Agrawal, T. Imielinski, and A. Swami. Database mining: a performance perspective. *IEEE Transactions on Knowledge and Data Engineering*, 5:914–925, 1993.

[2] R. Agrawal, T. Imielinski, and A. Swami. Mining association rules between sets of items in large databases. In *Proc. ACM SIGMOD Intl. Conf. Management of Data*, pages 207–216, Washington D.C., USA, 1993.

[3] D. Boley, M. Gini, R. Gross, E. Han, K. Hastings, G. Karypis, V. Kumar, B. Mobasher, and J. Moore. Document categorization and query generation on the world wide web using webace. *AI Review*, 12(5/6):365–391, 1999.

[4] S. Busemann, S. Schmeier, and R. Arens. Message classification in the call center. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, Seattle, WA, 2000.

[5] C. Clifton and R. Cooley. Topcat: Data mining for topic identification in a text corpus. In *Proceedings of the 3rd European Conference of Principles and Practice of Knowledge Discovery in Databases*, 1999.

[6] S. Deerwester, S. Dumais, G. Furnas, T. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[7] G. Karypis and V. Kumar. Multilevel k-way hypergraph partitioning. In *Proceedings of the 36th Conference on Design Automation (DAC)*, pages 343–348, New Orleans, LA, 1999.

[8] G. Karypis, A. R., V. Kumar, and S. Shekhar. Multilevel hypergraph partitioning: Applications in vlsi domain. In *Proceedings of the 34th Conference on Design Automation (DAC)*, pages 526–529, Anaheim, CA, 1997.

[9] J. Moore, E. Han, D. Boley, M. Gini, R. Gross, K. Hastings, G. Karypis, V. Kumar, and B. Mobasher. Web page categorization and feature selection using association rule and principle component clustering. In *7th Workshop on Information Technologies and Systems*, December 1997.

[10] M. Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, 1980.

[11] J. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann, 1993.

[12] M. Ramoni and P. Sebastiani. Bayesian methods for intelligent data analysis. In M. Berthold and D. Hand, editors, *Intelligent Data Analysis: An Introduction*. Springer, New York, NY, 1999.

[13] G. Salton and B. C. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, 24:513–523, 1988.

[14] G. Salton and M. McGill. *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, New York, 1983.

Table 3: Portion of automatically generated decision tree using mixed features.

```
Decision tree:
   CallStatus in Busy_1X,Busy_2X,Busy_3X,Dispatch_-_Direct,Accepted: LESS (9.0)
   CallStatus in Paged,AHE,No_Answer_2X,No_Answer_3X,VOX:
   |   CallStatus in Paged,No_Answer_2X,No_Answer_3X,VOX: LESS (24.0/10.0)
   |   CallStatus in Last_Time,EMAIL,Busy_1X,Busy_2X,Busy_3X,Wrong,
   |                 Message_1X,Message_2X,Message_3X,Redispatched,
   |                 No_Answer_1X,Contacted,Dispatch_-_Direct,Accepted,WEB,
   |                 N/A,FAX: MORE (0.0)
   |   CallStatus in AHE:
   |   |   Product in IO,LIU,P100,OHMUX : MORE (16.0/4.0)
   |   |   Product = XMTR : MORE (5.0/2.0)
   |   |   Product = IPC : MORE (3.0/1.0)
   |   |   Product in CLM,LAN,HINTRF,SPICED,STI,ANLG,SOX,BOS,EPLGW,RECD,MIN,GWY,
   |   |                AXV,XMP,IP,ALGOL,FCN,WMA,CB,MAX,MS-SCF,AFT,DPH,BC,R50MC,
   |   |                BATCH, LANG, PUN, 1540, SI: MORE (106.0)
   |   |   Product in EPCD,MDCM,DUC,UPGRADE,X9001: LESS (5.0)
   |   |   Product in SM,AOS,_42_,_90_,EOS,FSC_SMM,MPCI,FATBUS,UNKNOWN,TTREND,
   |   |                TOCTOOL,TSX_INFRASTRUCTURE,LOPS,SUW,IDS,_03_,APP,SAM,
   |   |                LIPPKE,RED50S,HPOWR,TSX_APP_DIRECTOR,HCM,UC,SM-PICON,
   |   |                DMCI,MAS,HVTS,CTL_BLDR,HOTTUNAII,ABE,DISK,SILCONIX,PCSI,
   |   |                XPI_DOCUMENTATION,GRXPLOT,AWOL_RDI,BPC,NG,
   |   |                DOCUMENTATION,MACRO,CNI,MISCELLANEOUS_PRODUC,HTD,NT,DMPC,
   |   |                NETWORK,DEBR,RHS,USRCDR,IOP_STREAMBUS,HPK2,DROSS,OPENDDA,
   |   |                SCADA,DOCPTDX,OPUS,UTILIMOLD,PMC,TSX_WINDOWS_NT,LEXCON,
   |   |                APPS,BRIC,ROC2K,TFIC,_31_,TSX_HARDWARE,_07_,MSH800,PCMM,
   |   |                OPR,MAC,RULA,MULTI_SCHEMATIC,40XLAN,TSX_BLDR,UNIX,PCTCTL,
   |   |                TIM,AWOL_SUPT_PRG,GRANT,AUX,PHLM_BUILDER,CRM,MOM80N,
   |   |                25,BATCH40,XPBX,XPOD,XRPCII,PARTS,49,XHS,DOSFILETRANS,
   |   |                URLBOOK,LINE_BUILDER,URTK,IO_MUX,PAE,PLANER,
   |   |                KITS_&ENHANCEMENTS,ODBC,U2LAN,HER: MORE (0.0)
   |   |   Product in LM,UPM,LLPIU,XXM,EC,LAN,UI,GAS,AWOL_MS,HARM,LEPIUS,
   |   |                UON,PLCG,LAN5000,HM,PM,SANDSCAPE,AM,RCD,FSC,PATHWAY,BSM,
   |   |                OM&S_SUPT_PRGM,MFC,HPROCMOD,GCI:
   |   |   |    f80 = 1: MORE (48.0)
   |   |   |   f80 = 0:
   |   |   |   |   f86 = 1: MORE (20.0)
   |   |   |   |   f86 = 0:
   |   |   |   |   |   f99 = 1: MORE (15.0)
   |   |   |   |   |   f99 = 0:
   |   |   |   |   |
```