Giga-Mining

Corinna Cortes and **Daryl Pregibon**
AT&T Labs-Research
180 Park Ave Bldg 103
Florham Park, NJ 07932 USA

Abstract

We describe an industrial-strength data mining application in telecommunications. The application requires building a short (7 byte) profile for all telephone numbers seen on a large telecom network. By large, we mean very large: we maintain approximately 350 million profiles. In addition, the procedure for updating these profiles is based on processing approximately 275 million call records per day. We discuss the motivation for massive tracking and fully describe the definition and the computation of one of the more interesting bytes in the profile.

Introduction

Data mining is the name given to the process of extracting information from large data sets. It is a multidisciplinary field that brings together the latest technology and research in database design, exploratory data analysis, algorithms for very large data sets, model fitting, visualization and systems.

Many will claim that data mining is nothing new. Database systems have been around for decades, and for centuries scientists have studied data sets for correlations and dependencies between the elements. They have developed models to describe data and invented ways of displaying both the data and the models.

However, data mining differs from "traditional" data analysis on one single, but important dimension: scale. The size of data sets has grown tremendously with modern computers and data storage capabilities. Ten years ago, a large data set would consist of thousands of observations/cases stored in 10MB. Today, a large dataset consists of millions of cases stored in 10GB. The size and complexity of the available data is often so overwhelming that most algorithms from machine learning, statistics, and visualization become impossible to use in practice. An important problem in data mining is to determine how to scale these algorithms (Glymour *et al.* 1996; Chiang *et al.* 1996).

Large data sets are ubiquitous in science, technology, and the commercial sector (Brachman *et al.* 1996; Garvey 1996; Metcalfe 1996). In this paper, we focus on the commercial sector since this is where our recent experience lies. Here, the data sets may be the result of credit card transactions, retail sales, or calls carried by a phone company. In a highly competitive marketplace these data sets are an extremely valuable asset. From these data, businesses can learn who their customers are, where they are, what their needs are, how they use existing services and products, what makes customers stop using or buying the offered services and product, and what offers could attract new customers. Businesses without methods for efficient recording, handling and analyzing data are at a serious competitive disadvantage. Thus, data mining is more than a research area, it is a commercial need.

In this paper, we demonstrate some techniques and methods for performing meaningful mining of gigabytes of data. As an example, we introduce the *universe list*, an industrial-strength data mining initiative.

The Universe List

AT&T is one of the largest phone companies in the world. It carries approximately 275 million (M) calls on a weekday — less over the weekend. The total number of calls carried in 1997 was 76 billion. On a typical weekday the calls involve approximately 65M unique phone numbers. Over a 40 day period we observe approximately 350M unique phone numbers.

The universe list is the set of unique telephone numbers (TNs) observed on our network, together with a few bytes of information on each to support operations such as billing, customer care, usage management, and fraud prevention and detection. Currently four distinct variables are monitored through time:

- **inactivity:** the number of days since a TN was last observed on the network (In),
- **minutes:** the average daily number of minutes a TN is observed on the network (Min),
- **frequency:** the estimated number of days between observing a TN (Freq),
- **bizocity:** the degree to which a TN displays business-like behavior (Biz).

These variables are maintained separately for inbound calling and outbound calling, and in some cases, distinctions are made for toll and toll-free calls.

The inactivity variable is determined by incrementing values in a data structure according to the expression:

$$\mathrm{In}(TN)_{new} = \begin{cases} 0 & \text{if observed today} \\ \mathrm{In}(TN)_{old} + 1 & \text{if } \textit{not} \text{ observed today} \end{cases}$$

The basic statistical device we use for tracking the other three variables is an exponential weighted average. This strategy is typical for tracking large numbers of entities in transaction-oriented applications (Burge & Shawe-Taylor 1996; Denning 1987; Lunt 1993). For example, letting $0 < \lambda < 1$, the expression:

$$\mathrm{Min}(TN)_{new} = \lambda \mathrm{Min}(TN)_{today} + (1 - \lambda)\mathrm{Min}(TN)_{old}$$

illustrates how today's minutes for a TN are combined with the old average daily minutes to provide a smooth current estimate of average daily minutes. Similar computations are made for frequency and bizocity.

The "aging" factor λ allows a trade-off between smooth scoring of a TN (small λ) and fast tracking of changes in a TNs calling behavior (large λ). If the recursion is expanded, the new estimate can be expressed as a weighted sum of all previous daily values, where the weights decrease smoothly in time. That is, the weight given to a TNs call volume k days ago is:

$$w_k = (1 - \lambda)^k \lambda$$

Thus old data is "aged out" as new data is "blended in." This is the main principle behind the universe list.

The universe list is based both on statistical modeling and an efficient computational approach. In the next section, we present the statistical method we use. We outline the need and the method for monitoring the bizocity score. Bizocity is an interesting behavioral factor that requires challenging statistical modeling and tracking methods. We then present the computational engine necessary to maintain the call detail stream and to update data structures, and briefly mention some of the characteristics of these data structures. A careful design of the data structures is of critical importance: when processing hundreds of millions of calls per day, the I/O processes tend to dominate CPU time, and thus can considerably slow down the system (*e.g.*, in the worst case it might take more that 24 hours to process a days worth of data). Finally, we describe the interface to our data mining results. It provides convenient access to the data and makes testing possible when the size of the data exceeds several gigabytes.

Bizocity

One natural piece of information that a telecommunications service provider is interested in concerns whether a phone number belongs to a business (Biz) or a residential (Res) customer. Historically, the company has split its operations along the Biz/Res boundary for billing, collections, customer care, and fraud prevention and detection. So, it is natural to infer Biz/Res status of any TN observed on the network to aid internal operations. For example, when a call comes into a customer care center, knowledge of the Biz/Res status of the calling party allows the call to be automatically routed to the appropriate care channel. This is an efficient use of the channel and a convenience to the calling party. Similarly, fast, efficient investigation of toll fraud requires knowledge of the Biz/Res status of the involved lines. For example, long calls originate quite commonly from residences late at night, but not from a business.

For about 30% of the 350M lines seen on our network, we have a provided value of Biz/Res status: when a customer signs up for phone service with the local phone company, they must specify their status. If the customer chooses AT&T as their long distance service provider, this information is (usually) passed on to AT&T. However, information from the local phone companies is frequently incomplete (businesses may have their central number listed as a business but not all the additional lines), or it can be out of date (the information may arrive much later than the date the customer starts using AT&T long distance service).

The universe list solves the problem of the missing Biz/Res status for the remaining 250M lines (or 70% of all lines seen) by deriving and maintaining a *behavioral* estimate of the Biz/Res status of *all* phone numbers observed on the network (over the most recent 40 days). We call this estimate the *bizocity* score of a phone number. Bizocity is a probabilistic measure of how much a line behaves like a business. It is constructed from characteristics of the calls going into or out of each line. We emphasize that the bizocity of a line is not the same as the provided Biz/Res status. The two pieces of information can be different, for example when a residential line is used for a home business.

Our methodology estimates the bizocity of a phone number by tracking the behavior of the line over time. Every day we score all lines with calling activity for how business-like they behave today. This score is combined with the bizocity score from past history of the line to form a new bizocity score. These new scores are stored for next day's processing.

Modeling Bizocity

The basic building block for the universe list is a call detail record generated at call completion. Most of the fields in the record relate to how the call is routed on the network. Only a few fields contain useful information for bizocity modeling: the originating TN, the dialed TN, the connect time, and the duration of the call. For credit card and collect calling, other fields identify the billing number. Such calls are of minor importance for bizocity scoring. It is apparent that these records do not provide customer identification, only pairs of phone numbers.

We can add value to the plain call detail records by dipping into other available data bases. For about 100M TNs we have a provided Biz/Res status so we can add two fields to each record that identify the Biz/Res status of the originating and dialed TN. This database also enables us to construct a training set to learn call-

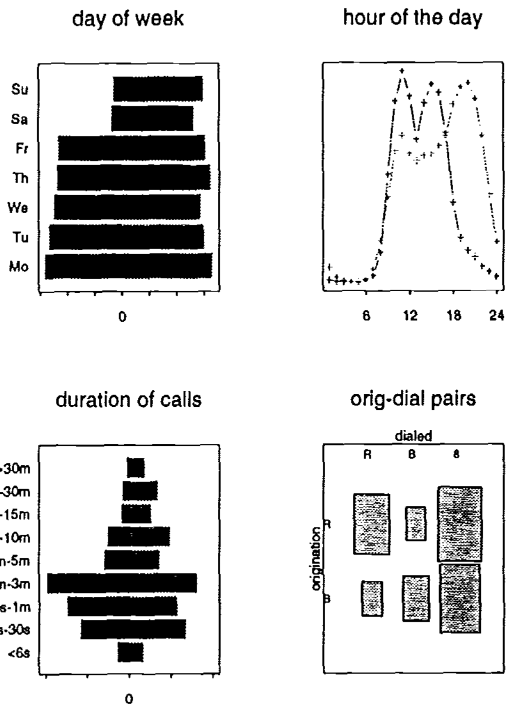

Figure 1: Historical tendencies for business and residential calling. In the first three panels, dark grey represents business calling and light grey represents residential calling. In the fourth panel, the size of the rectangle is proportional to the amount of calls between origination and dialed pairs.

ing behavior. Using the training data we illustrate the (statistical) signal on bizocity in call detail.

Figure 1 illustrates how businesses and residential customers make different use of the telephone. Businesses make calls during the week but infrequently over the weekend. Businesses call from 9AM-5PM with a lunch break, while residential calls peak in the evening. Businesses make short calls, and they call other businesses or 800 services. While individual call detail records are very simple and may be very noisy, Figure 1 shows that this simplicity and noise can be overcome by sheer volume: from aggregation we obtain a strong signal for differentiating businesses from residences.

The statistical patterns in Figure 1 are aggregated over all TNs with a provided Biz/Res status for a typical week of calling. If we study a single call from a TN we get a much weaker signal. For instance, we cannot derive much information for a TN based on a single three minute call on a weekday afternoon. But by staging processing and aggregating calls, bizocity scoring of individual TNs can be significantly enhanced.

We stage processing over a 24 hour period. It both serves the purpose of collecting more data (and hence getting a better signal from the single TN), and it reduces the I/O burden on call processing. Recall that on a single day the AT&T network carries approximately 275M calls but that only 65M unique TNs are involved. Therefore, processing call detail every 24 hours can reduce the I/O by 75%. The flip side of staging the processing is that we need to store 24 hours of data, and then sort it so that all the calls from a single TN can be processed together. Storage is a minor problem: disks are cheap and we compress the records in an efficient manner. Specifically, we translate individual ASCII records to 32 byte binary records so that a day of call detail occupies a total of about 8GB. Sorting is done directly on these binary records. It takes about 30 minutes on an SGI Challenge with 3GB RAM using a single processor.

Once we have all the calls from a single TN, we form its calling profile by binning its calls into a four dimensional array, *e.g.*, a data cube. The dimensions are day-of-week, time-of-day, duration, and Biz/Res/800 status of the dialed TN. The size of the entire cube is $7 \times 6 \times 5 \times 3$.

From training data collected on known businesses and residences we have (off-line) formed logistic regression models for scoring a TN based on its calling profile. Logistic regression seems to do a reasonable job for this two group classification problem, but due to the near-rank deficiency of the binned data, regularization via a "ridge" penalty is necessitated. An advantage of logistic regression is that it leads to easily interpretable models. The coefficients of one of these models are shown in Figure 2. It shows how long calls in the evening contribute towards scoring a TN as a residence while short calls in the middle of the day favor business-like behavior. We also tried boosted stumps (one level deep decision trees) (Freund & Schapire 1995) as a model for the bizocity score. This model had similar behavior as the ridge regression. We prefer the ridge regression model because of its interpretability.

We use the models on the calling profile for the single TN to estimate today's business-like behavior. This is done for all TNs with calling activity. Using just today's calls as a bizocity score would give a noisy picture of a TN. Therefore, we *track* the TNs and smoothly aggregate their daily scores. We update the bizocity score of a TN, Biz(TN), using exponential weighting as previously mentioned:

$$\mathsf{Biz}\ (\mathsf{TN})_{new} = \lambda \mathsf{Biz}(\mathsf{TN})_{today} + (1-\lambda)\mathsf{Biz}(\mathsf{TN})_{old} \quad (1)$$

where $0 < \lambda < 1$. With a reasonably small λ, the new bizocity score can contain weeks of historic information about a TN without storing more than one set of scores.

From training data of known business and residential customers, we have estimated probabilities of how likely a TN is to be a business given that it makes *no* calls on a weekday or over a weekend. If a TN does *not* have any calling activity, its bizocity is also updated using Equation (1), but employing a style of *lazy evaluation*. That is, updates of TNs with no daily calling are delayed until they are observed again.

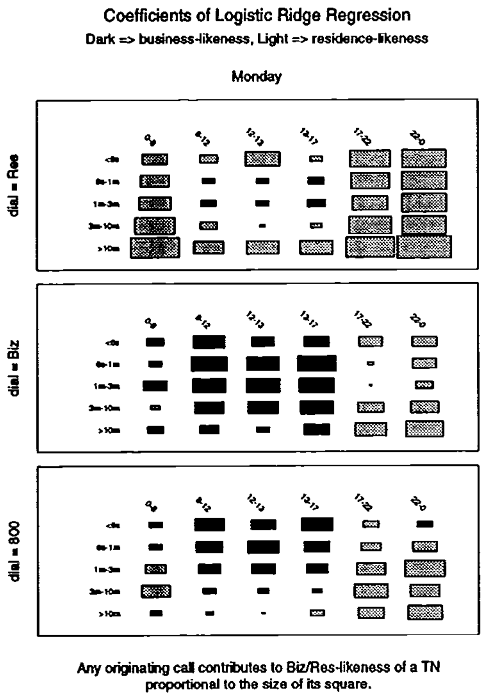

Figure 2: Coefficients of a logistic regression model for scoring a TN based on its outbound weekday calling profile.

One interesting and important aspect of the universe list is that the bizocity scores are self-consistent and self-maintaining. Recall that in our off-line training data, the calling profile of a TN is formed by binning new calls with respect to the Biz/Res/800 status of the dialed TN. But for on-line production scoring, we do not have a provided status for the majority of the TNs. We overcome this deficiency by always using the bizocity score for the dialed TNs. Consider for example a call to a TN that is a Biz with probability 0.7 and a Res with probability 0.3. The profile used in the scoring splits the single call into it's fractional components, and populates two cells of the profile: adding 0.7 in the Biz margin of the data cube and 0.3 in the Res margin of the data cube. The scoring function itself remains as the dot product of the logistic regression coefficients and the entries in the data cube.

The universe list has been running in this fully autonomous mode for more than a year and we still observe fine separability of Biz and Res lines: the bizocity scores have not collapsed into a single value.

The accuracy of our bizocity scoring is hard to assess since a TN may very well *behave* differently from it's provided status. This will be the case for example when a pure residential line migrates to home business use. Our accuracy for predicting provided status is 75%.

Computational Issues

The universe list requires serious hardware and robust, efficient software. Call detail records are stored in 32 byte binary records. At no point in the processing do we perform an ASCII conversion. We have highly efficient programs that work directly on the binary records. For example, sorting 300M records can be accomplished in about 30 minutes (recall that records need to be sorted two ways since we are scoring TNs based both on calls into and out of them). We have also developed a set of tools for operating on the binary records to support filtering and tabulation.

The universe list itself is implemented in C. In a couple of hours it makes two passes over the data, first to score each TN seen with outbound calling, then to score each TN seen with inbound calling. This processing is done on an SGI Challenge with 20 processors, 3GB RAM, and 5GB of swap space. As one would expect when discussing large volumes of data, we maintain a several terabyte disk farm.

The scores are stored in special purpose data structures that allow for fast sequential *and* fast random access. These data structures reflect the semantics of 10 digit telephone numbers into the exchange (first 6 digits) and line (last 4 digits). First, they exploit the fact that the space of exchanges is sparsely populated — only 150,000 of the possible 1M exchanges are currently in use. Second, we exploit the fact that exchanges in use are often densely populated, so we store all 10,000 possible line numbers for an exchange in a contiguous block whether or not the exchange is actually fully populated. We quantize inbound/outbound scores for a TN into one byte, so our data structures can take up about 2GB for each variable (score). This is not much more than is minimally required since a lower bound for the space required is four bytes for the TN and one byte for the score, suggesting a 1.5GB data structure. We initially considered this smaller data structure, but migrated to the other for fast random access. The data structure also deals effortlessly with growth within an exchange, and quite simply with area code splits.

Besides the three inbound/outbound measurement variables that we maintain for each TN, we maintain activity maps for all TNs. We keep track of how many days we have *not* seen calling activity on a TN inbound, outbound, or to 800-numbers. The upper bound for the number of passive days is 40. This allows us to pack the three activity values for the single TN into two bytes. The size of the activity maps is about 4GB, so it is necessary to use 64-bit programming and to run our programs on 64-bit machines.

Interface

Implicitly, our methodology involves estimating several hundred million parameters per day. How can one explore such a huge parameter space and ensure that the algorithm is doing the "right thing"? Our approach exploits the fact that TNs are aggregated at several levels of a natural hierarchy:

- **bank:** the group of 100 TNs that share the first 8 digits of the TN,
- **exchange:** the group of 10000 TNs that share the first 6 digits of the TN,
- **area code:** the group of (<10M) TNs that share the first 3 digits of the TN,
- **state:** the group of TNs that reside in the same state (i.e., a group of area codes),
- **region:** the group of TNs that reside in the same region of the county (i.e., a group of states).

We make the data available at all these levels through a variety of visualization tools. In effect, the tools are designed to be used in a top-down fashion whereby one starts at the highest level of the hierarchy. As more detail is required, the user is able to "drill-down" and check results at a finer level. Depending on the application, information at the TN level is provided. For example, corporate security applications require direct access to a random-access data warehouse of call records.

For interactive use, we employ a web browser as the user interface. This has proven quite effective in disseminating the results to the user community over the corporate intranet. The web server can of course only be accessed from inside the AT&T firewall, and the pages are password protected so only people with "need to know" can gain access. From the web, the user can, with a few clicks, display scores for a full exchange. The 10,000 lines in an exchange are presented to users as 100×100 images. The cells of the image are indexed by the individual line numbers such that the first 2 digits of the line indexes the row and the trailing 2 digits the column. Since the universe list contains about 150,000 exchanges, all images have to be computed on-the-fly. Snappy performance and rapid response is obtained by our special purpose data structures and C-code that directly computes images in gif format.

Conclusion

The universe list proves that it is indeed possible to do meaningful mining of gigabytes of data. The universe list has now been running *continuously* for more than a year, *every day* updating 7 bytes of information on *every TN* with calling activity. The universe list has demonstrated how it is possible to establish and effectively maintain short, yet powerful, behavioral profiles on millions of TNs through tracking of simple features.

The bizocity score described in this paper relates to a classification task: statistical models are used to categorize telephone numbers as business or residential lines. Another interesting task is that of capturing deviations: new transactions are compared to the existing profile of the telephone number, and large deviations are flagged for further investigation. This is the methodology behind AT&T's international toll fraud detection system. In this application the number of TNs tracked is much smaller (about 15M) while the profiles are much larger (512 bytes). In addition, this system is not time-driven, but rather event-driven: calls are processed as they occur rather than once per day.

While our experience is with call detail data from the AT&T network, the methods for tracking simple profiles described in this paper can, of course, be applied to other very large transactional data set such as airline reservations, credit card usage, and WWW traffic.

References

Brachman, R. J.; Khabaza, T.; Kloesgen, W.; Piatetsky-Shapiro, G.; and Simoudis, E. 1996. Mining business databases. *Communications of the ACM* 39:11:42–48.

Burge, P., and Shawe-Taylor, J. 1996. Frameworks for fraud detection in mobile telecommunications networks. In *Proceedings of the Fourth Annual Mobile and Personal Communications Seminar.* University of Limerick.

Chiang, Y. J.; Goodrich, M.; Grove, E.; Tamassia, R.; Vengroff, D.; and Vitter, J. 1996. External memory graph algorithms. In *ACM-SIAM Symposium on Discrete Algorithms.*

Denning, D. E. 1987. An intrusion-detection model. In *IEEE Trans Soft Eng Vol 13, No 2.*

Freund, Y., and Schapire, R. E. 1995. A decision theoretical generalization of on-line learning and an application to boosting. In *Proceedings of the Second European Conference on Computational Learning Theory, 2.*

Garvey, M. J. 1996. Citibank mines data. *Information Week* 600.

Glymour, C.; Madigan, D.; Pregibon, D.; and Smyth, P. 1996. Statistical inference and data mining. *Communications of the ACM* 39:11:35–41.

Lunt, T. F. 1993. Detecting intruders in computer systems. In *Proceedings of Auditing and Computer Technology.*

Metcalfe, B. 1996. A collaborative filter can help you mine your data for jewels. *InfoWorld* 18:49.