



ZIPPING OUT RELEVANT INFORMATION

By Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto

THE WORD *INFORMATION* CAN DESCRIBE THE CULTURAL STRUCTURES OF SCIENCE, LEGAL AND MARKET INSTITUTIONS, ART, MUSIC, KNOWLEDGE, AND EVEN BELIEFS. IT CAN ALSO REFER TO THE STRUCTURES AND PROCESSES OF PHYSICAL AND BI-

ological phenomena, which we can represent as sequences of characters. Experimental investigations of physical processes, for instance, typically produce sequences or time series of data. Other systems, such as DNA and protein sequences or human language, are intrinsically represented as strings of characters. Treating information as sequences of characters helps make this information searchable—a necessary first step in navigating the overwhelming mass of data facing us today.

Although the abundance of information and its accessibility represents an important cultural advance, it also introduces a new challenge: retrieving *relevant* information. Imagine entering the largest library in the world and seeking all the relevant documents on your favorite topic. Without an efficient librarian's help, this task would be difficult if not impossibly hopeless. The references you wanted likely would remain buried under tons of irrelevancies.

On a more positive note, the growing body of available data provides an ideal test bed for theoretical constructions and models. This opportunity has stimulated considerable interest from researchers in many different communities—physicists, mathematicians, economists, and statisticians, to name a few.

In this spirit, we seek to discover the most suitable tools for examining large masses of data and extracting useful information from it.

Information Extraction

To accomplish the ambitious task of finding the proverbial needle in a haystack, we must first define what useful or relevant information is and where and how it is coded. This is a nontrivial problem because information means different things in different contexts. Moreover, it has no absolute value, depending instead on the specific filters observers impose on their data. Consider a simple coin-toss experiment. A gambler is probably only interested in the toss's outcome (heads or tails), but a physicist might be interested in whether the outcomes reveal anything about the coin's nature (such as whether it is honest or dishonest).

We extract information via a two-step process. The *syntactic* step is where we first identify the structures present in messages without associating any specific meaning to them. It is only in the second (or *semantic*) step that comprehension of meaning occurs; it is the step in which we connect the syntactic information to previous experience and knowledge.

As an example of this two-step process, consider how to identify the language in which a given text is written. In the first step, we scan through the text and identify syntactic structures: articles, verbs, adjectives, and so on. But only someone who knows the language can carry out the second phase—summarizing the incoherent jumble of syntactic data in the sentences' specific meaning.

Similarly, for a DNA sequence, this process would entail identifying the subsequences that encode the genes and then identifying their specific functions. In studying time-series data (for example, earthquake seismograms or stock prices), we might seek specific features and trends that could help us characterize such sequences' sources.

Keep in mind that the syntactic and semantic levels are not always strictly related; the correlation between them could depend a lot on the specific source of information. In other words, suppose we could efficiently extract a given sequence's syntactic information. Could we obtain from this measure the information (in the semantic sense) we were trying to extract from the sequence? The answer to this question is again far from trivial. This article will show that we can draw at least partial answers within a posteriori reasoning.

Having this plan in mind, the first logical step is to provide tools that can measure the amount of syntactic or structural information contained in a given string. Obviously, we would also like to be able to compare pairs of strings in terms of their relative infor-

mation content. We approach the problem of constructing these tools from Claude Shannon's information-theoretic viewpoint.^{1,2}

The Entropy Concept in Information Theory

Shannon's seminal paper founded information theory in the context of electronic communications.¹ Information theory has subsequently acquired a leading role in such arenas as computer science, cryptography, biology, and physics.²

In information theory, the word *information* acquires a precise meaning—namely, a string's entropy. In a sense, entropy measures the surprise that the source emitting the sequences can give. Suppose the surprise you feel on learning that an event E has occurred depends only on the probability of E occurring. If the event occurred with probability one (certainty), your surprise at its occurrence would be zero. On the other hand, if the probability of E 's occurrence were quite small, your surprise would be proportionally larger.

For a single event occurring with probability p , the degree of surprise is proportional to $-\ln p$. Generalizing the result to a random variable X (which can take N possible values x_1, \dots, x_N with probabilities p_1, \dots, p_N) is easy. In this case, the average surprise you receive on learning the value of X is precisely the entropy of the source emitting X (meaning, $-\sum p_i \ln p_i$).

Consider the entropy of a symbolic sequence that a given source S emits. For simplicity, regard S as stationary (meaning the stochastic mechanism generating the sequence does not change with time). In this case, we can give a coherent definition for the entropy through the so-called N -block (or vectorial) entropies. A symbolic sequence X_1, X_2, X_3, \dots (here X_i is the

symbol emitted at time $t = i$ and each X can assume one of M different values), the " N -block entropy" is

$$H_N = - \sum_{\{C_N\}} P(C_N) \ln P(C_N), \quad (1)$$

where $P(C_N)$ is the probability of the N -word $C_N = [X_i, X_{i+1}, \dots, X_{i+N-1}]$. We define the Shannon entropy for an ergodic stationary process as the asymptotic limit of the normalized N -block entropy:

$$h(S) = \lim_{N \rightarrow \infty} \frac{H_N}{N}. \quad (2)$$

A theorem Shannon and McMillan developed^{1,3} expresses in a precise way how h in Equation 2 quantifies the source's "complexity": if N is large enough, we can partition the set of N -words $\{C_N\}$ into two classes, $\Omega_1(N)$ and $\Omega_2(N)$, such that all the words $C_N \in \Omega_1(N)$ have probability $P(C_N) \sim \exp(-bN)$ and

$$\sum_{C_N \in \Omega_1(N)} P(C_N) \rightarrow 1 \quad \text{for } N \rightarrow \infty \quad (3a)$$

$$\sum_{C_N \in \Omega_2(N)} P(C_N) \rightarrow 0 \quad \text{for } N \rightarrow \infty. \quad (3b)$$

The theorem also implies that the effective number of typical sequences $N_{\text{eff}}(N)$ (those in $\Omega_1(N)$) effectively observable is $N_{\text{eff}}(N) \sim e^{bN}$. (In nontrivial cases in which $b < \ln m$, $N_{\text{eff}}(N) \ll m_N$, where m_N is the number of possible N -words.) For processes without memory, the Shannon–McMillan theorem is essentially the law of large numbers. Moreover, $N_{\text{eff}}(N) \sim e^{bN}$ is the information theory equivalent of the Boltzmann relation in statistical thermodynamics: $S \propto \ln W$, where W is the

number of possible microscopic states, and S is the thermodynamic entropy. Under rather natural assumptions, the Shannon entropy h is, apart from a multiplicative factor, the unique quantity that characterizes the surprise.³

You might now ask how to extend the definition of the entropy for a generic string of characters without any reference to its source. Among the many equivalent definitions of entropy, the best for this case is the Chaitin–Kolmogorov complexity (or algorithmic complexity). It states that the algorithmic complexity of a string of characters is given by the length (in bits) of the smallest program that produces the string as output.

A string is called *complex* if its complexity is proportional to its length. Obviously, this definition is quite abstract. For one thing, finding such a program, even in principle, is impossible.⁴ Because the definition is not based on the amount of time the best program should take to reproduce the sequence, you can never be sure that a shorter program doesn't exist that could eventually output the string in a longer (eventually infinite) time.

Data Compression and Entropy Measurement

To overcome the intrinsic difficulty of measuring the entropy of a generic string of characters, we use the relation between the entropy h and the maximum compression rate of a sequence X_1, X_2, X_3, \dots expressed in an alphabet with M symbols. If the length T of the sequence is large enough, then compressing it into another sequence (with an alphabet with M symbols) whose size is smaller than $Tb/\ln M$ is not possible. Therefore, noting that the number of bits needed for a symbol in an alphabet with M symbols is $\ln M$, the maximum allowed compression rate is $b/\ln M$.

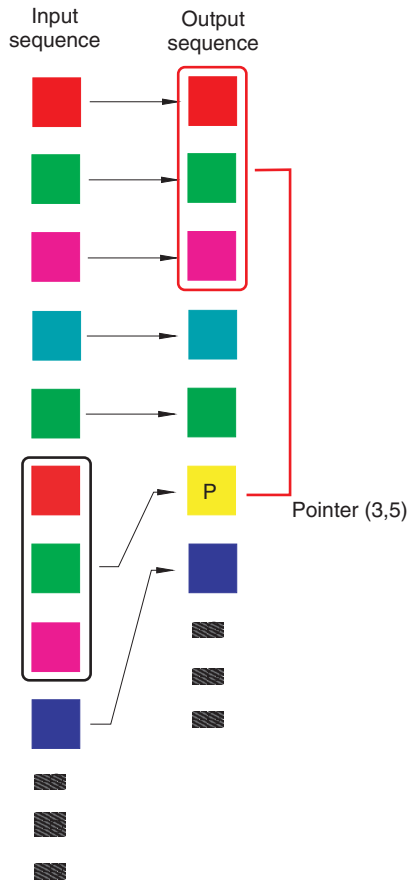


Figure 1. The Lempel-Ziv 77 algorithm. LZ77 searches in the look-ahead buffer for the longest substring (in this case, a substring of colors) that has already occurred and replaces it with a pointer represented by two numbers—the length of the matching and its distance.

Algorithms explicitly conceived to approach the theoretical limit of optimal coding are called file compressors or zippers; a typical zipper, besides reducing the space a file occupies on a memory storage device, can be considered an entropy meter. The better the compression algorithm, the closer the zipped file’s length is to the minimal entropic limit, and hence the better the zipper-provided entropy estimate will be. Compression algorithms provide a powerful tool for measuring entropy and (more generally) estimating more sophisticated complexity measures.^{1,3}

The Lempel-Ziv 77 algorithm (LZ77) represents a great improve-

ment in the field of data compression.⁵ This algorithm zips a file by exploiting the presence of repeated subsequences of characters within it. Its compression efficiency becomes optimal as the file’s length goes to infinity.⁶ Let’s review briefly how it works.

The LZ77 algorithm first looks for duplicated strings in the input data (see Figure 1). It replaces the second occurrence of a string with a pointer to the previous string. This pointer consists of two numbers: a distance, representing how far back into the window the sequence starts, and the length in characters of that subsequence. The original LZ77 algorithm defines the window as the section of the sequence already scanned sequentially. Commercial implementations generally use a sliding buffer, typically of 32,768 characters, as follows: if at a given time the part of the sequence already scanned is $\sigma_0 \dots \sigma_{i-1}$, the algorithm is allowed to seek in a look-ahead buffer the longest subsequence that has already occurred in the window $\sigma_{i-32,768} \dots \sigma_{i-1}$. If i is less than 32,768, the window will be $\sigma_0 \dots \sigma_{i-1}$.

For example, in the compression of an English text, an occurrence of the sequence **the** will be represented by the pair $(d, 3)$, where d is the distance between this occurrence of **the** and the previous one. The zipper does not recognize **the** as a dictionary word—only as a specific sequence of characters without reference to a specific dictionary. The sequence will then be encoded with a number of bits of the order of $(\log_2(d) + \log_2(3))$ —the number of bits necessary to encode d and 3. The average distance between two consecutive instances of **the** in an English text is roughly 10 characters. Therefore the subsequence **the** will be encoded with less than 1 byte instead of 3 bytes.

The Relative Entropy

An important quantity, the *relative entropy* (or Kullback-Leibler divergence⁷) measures the statistical remoteness between two distributions. You can easily grasp its essence in the following example.

Consider two ergodic sources \mathcal{A} and \mathcal{B} emitting sequences of zeroes and ones: \mathcal{A} emits 0 with probability p and 1 with probability $1 - p$ whereas \mathcal{B} emits 0 with probability q and 1 with probability $1 - q$. The previously described compression algorithm can encode a sequence emitted by \mathcal{A} almost optimally—coding a 0 with $-\log_2 p$ bits and a 1 with $-\log_2(1 - p)$ bits. However, this \mathcal{A} -optimal coding is not optimal for the sequence that \mathcal{B} emits. In fact, this sequence’s entropy per character in the \mathcal{A} -optimal coding will be $-q \ln p - (1 - q) \ln(1 - p)$. The entropy per character of the sequence that \mathcal{B} emits in its own optimal coding is $-q \ln q - (1 - q) \ln(1 - q)$. The number of bits per character wasted to encode the sequence that \mathcal{B} emits with the \mathcal{A} -optimal coding is the relative entropy of \mathcal{A} and \mathcal{B} ,

$$\begin{aligned}
 D(\mathcal{B} \parallel \mathcal{A}) &\equiv D(q \parallel p) \\
 &= -q \ln \frac{p}{q} \\
 &\quad - (1 - q) \ln \frac{1 - p}{1 - q} \quad . \quad (4)
 \end{aligned}$$

A linguistic example might clarify the situation. Transmitting an Italian text in Morse code optimized for the English language’s letter frequency requires extra bits compared with the same text transmitted using a coding scheme optimized for the Italian language. The difference is a measure of the relative entropy.

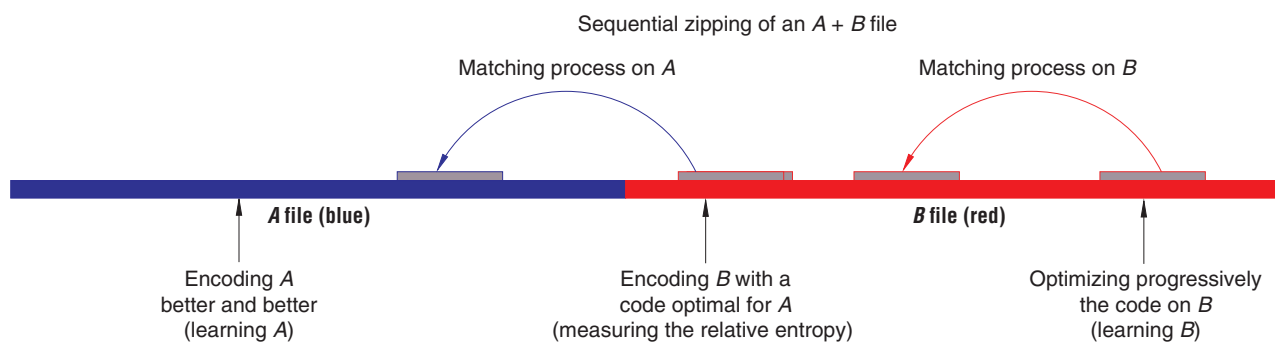


Figure 2. Sequential zipping of two texts A and B . A sequential zipper optimizes its features at the interface between two sequences A and B while zipping the sequence $A + B$ obtained by simply appending B after A .

From the Syntactic to the Semantic Level

We already mentioned that extracting information from a string proceeds in two steps. Any algorithm that measures the entropy of a string of characters (with arbitrary statistical properties) only carries out the first, or syntactic, step. To proceed to the semantic level, we must add other ingredients that bridge the gap between a sequence's syntactic properties and its semantic aspects. With this precise aim, we recently proposed a general method for context recognition and classification of strings of characters or other coded information.^{8,9} Other researchers have proposed similar approaches.⁹⁻¹²

The key point of our new method is simple. Suppose you want to estimate the distance (or similarity) between texts A and B in terms of their informational content. For instance, for two texts written in different languages (such as English and Italian), their distance is a measure of the difficulty a typical speaker of tongue A experiences in understanding the text written in language B .

We can measure one text's remoteness from the other by estimating the relative entropy.¹² To do this, we perform a procedure so simple that it is reproducible on virtually any modern personal computer. We take a long English text and append to it an Italian text; we then zip the resulting text.

The zipper begins reading the file,

starting with the English text. After a while, it encodes optimally the English file (this is the compression procedure's aim). Naturally, when the Italian part begins, the zipper starts encoding it in a way that is optimal for the English. So, the first part of the Italian file is encoded with the English code, thus the zipper "learns" Italian and changes its rules.

Therefore, if the Italian file's length is small enough, the difference between the length (in bits) of the zipped English/Italian text and the length (in bits) of the English text zipped alone will give a measure of the distance between the two texts. Figure 2 illustrates this learning process when using the LZ77 algorithm.

A universal scaling (or learning) function rules the way the compression algorithm learns a sequence B after compressing a sequence A .⁹ There exists a cross-over length for B that depends on the relative entropy between A and B , below which the compression algorithm does not learn B (measuring in this way the relative entropy between A and B) and above which it starts learning B ; it optimizes the compression using B 's specific features.

More precisely, to compute the relative entropy of two sources \mathcal{A} and \mathcal{B} , we extract a long sequence A from the source \mathcal{A} and a long sequence B from the source \mathcal{B} . We create a new sequence $A + B$ by simply appending B to A . If B is short enough (shorter than the cross-over length), we can measure

the relative entropy by zipping the sequence $A + B$ (using gzip or an equivalent sequential compression program). The measure of B 's length in the coding optimized for A will be $\Delta_{AB} = L_{A+B} - L_A$, where L_X indicates the length in bits of the zipped file X . If, on the other hand, B is longer than the cross-over length, we must change strategies and implement an algorithm that does not zip the B part but that simply reads it with the (almost) optimal coding of part A . The relative entropy S_{AB} per character between \mathcal{A} and \mathcal{B} is thus estimated by

$$S_{AB} = (\Delta_{AB} - \Delta_{BB}) / |B|. \quad (5)$$

The relative entropy is not a distance (or metric) in the mathematical sense: it is neither symmetric, nor does it satisfy the triangle inequality. In many applications (such as phylogenesis), defining a true metric that measures the actual distance between sequences is vital.

Once you have defined the entropy-based remoteness between two texts, or more generally, between two strings of characters with this "kitchen physics" procedure, you have in your hands a powerful tool. This tool will help you implement suitable algorithms for recognizing a given sequence's context. For example, for a sequence of characters representing a text, you might want to recognize the language in which it is written, discover its author, or see its subject.

Table 1. Our author recognition experiment. The two columns at the far right show the number of times another text by the same author was ranked in either the first position or in one of the first two positions.

Author	Number of texts	Number of success 1	Number of success 2
Alighieri	8	8	8
D'Annunzio	4	4	4
Deledda	15	15	15
Fogazzaro	5	4	5
Guicciardini	6	5	6
Macchiavelli	12	12	12
Manzoni	4	3	4
Pirandello	11	11	11
Salgari	11	10	10
Svevo	5	5	5
Verga	9	7	9
Bacon	6	6	6
Brown	3	2	3
Chaucer	6	6	6
Marlowe	5	4	4
Milton	8	8	8
Shakespeare	37	37	37
Spencer	7	5	5
Total	162	152	157

Language Recognition

Suppose you are interested in automatically recognizing the language of a text X . We can summarize the procedure to use as follows:

- Take as large a collection as possible of long texts (a corpus) in different (known) languages: English, French, Italian, Tagalog, and so on.
- Consider all the resulting files obtained by appending in turn all the corpus elements to the unknown file X .
- Now measure all the “distances” between the unknown file X and all the corpus elements.
- The corpus file for which the distance from file X is minimal will indicate either the language closest to the X file's or its actual language (if the corpus contained the unknown language).

Authorship Attribution

How can you use this technique to recognize automatically the author of a given text X ? Imagine as large a collection as possible of texts by known au-

thors, written in the language of the unattributed text X . We then seek the text A_i for which the difference $L_{A_i+X} - L_{A_i}$ is minimum.

To collect statistics, we performed such an experiment using a corpus of 162 different texts (from Italian and English literature). For each run, one of the texts in the corpus was an unknown text (see Table 1). Our rate of success was 93.8 percent, which is the ratio between the number of texts whose author is recognized and the total number of texts considered.

For each unknown text, we rank the relative entropy between the unknown text and each reference text. We count it as a success if either the unknown text or another text by the author of the unknown text is ranked in the first position. (Of course, this works only for a posteriori experiments in which we know in advance the unknown text's author.)

There are, of course, fluctuations in each author's success rate. This must be expected, because defining writing style precisely is difficult; moreover, it can vary a lot within a single author's works.

Context Recognition and Classification of Sequences

Determining a given text's subject is already possible within our method's framework. It proceeds in much the same way as determining a text's language or author. In this case, we must compile a collection of texts that treat known subjects. Then, we can represent each known subject as a box to which we can associate the unknown texts. Given an unknown text, we can now determine which box is the closest and conclude that the unknown text's subject most closely resembles that of the closest.


Automated subject determination is fundamental in another important application: automatic universal classification. This term implies a hierarchical organization of a corpus of sequences (texts) on the basis of their content, a feature likely to appeal to librarians, bibliographers, publishers, or search engine managers. With our method, we have obtained accurate results in classifying large collections of legal and literary texts as well as in finding messages newsgroup archives. (For more information on this experiment, visit www.ai.mit.edu/~jrennie/20Newsgroups and <http://babbage.sissa.it/abs/cond-mat/0203275>.)

We can imagine a further step forward, wherein we try to construct in a self-consistent way the boxes' structure. Suppose we have a large collection of texts and wish to classify them; an example might be classifying Web pages for a search engine. The ability to guess algorithmically a text's subject without having to read it would permit automated classification.

In this case, we must define a distance (in the mathematical sense)^{8,13} between all pairs of elements in the corpus. The pairwise distances $\rho(X, Y)$ are then elements of the distance ma-

trix. From the distance matrix, we can build tree representations: phylogenetic trees, spanning trees, and so on. This would provide a graphical representation of the corpus structure, letting us visualize different relationships.⁸ Although we have not yet found a rigorous metric, we can do almost as well in practice by imposing the triangular inequality by hand.

The information-theoretic method described in this article applies to any kind of corpora of character strings, independent of the type of coding behind them. The method has great potential for fields where human intuition might fail: DNA and protein sequences, geological time series, stock market data, medical monitoring, and so on. In these examples, we see no barrier to developing algorithms for automatic recognition and classification or for identifying specific significant patterns.

Further potentially fruitful areas of application might be in identifying the base sequences encoding genes, searching for premonitory patterns in geological or medical data, and the demarcation of boundaries between regions with different properties. The boundary demarcation problem, sometimes called the segmentation problem, is central to defining and optimizing suitable observables that can reveal sudden changes in complex data sequences. 

References

1. C.E. Shannon, "A Mathematical Theory of Communication: Part II, The Discrete Channel with Noise," *The Bell System Technical J.*, vol. 27, 1948, pp. 623–656.
2. W.H. Zurek, ed., *Complexity, Entropy, and Physics of Information*, Addison-Wesley, 1990.
3. A.I. Khinchin, *Mathematical Foundations of Information Theory*, Dover, 1957.
4. M. Li and P. Vitányi, *An Introduction to Kolmogorov Complexity and Its Applications*, Springer, 1997.
5. A. Lempel and J. Ziv, "A Universal Algorithm for Sequential Data Compression," *IEEE Trans. Information Theory*, May 1977, pp. 337–343.
6. A.D. Wyner and J. Ziv, "The Sliding-Window Lempel-Ziv Algorithm Is Asymptotically Optimal," *Proc. IEEE*, vol. 82, 1994, pp. 872–877.
7. T. Cover and J. Thomas, *Elements of Information Theory*, John Wiley & Sons, New York, 1991.
8. D. Benedetto, E. Caglioti, and V. Loreto, "Language Trees and Zipping," *Physical Rev. Letters*, vol. 88, 2002, p. 048702.
9. A. Puglisi et al., "Data Compression and Learning in Time Sequences Analysis," cond-mat/0207321, submitted to *Physica D*, 2002.
10. D. Loewenstern et al., *DNA Sequence Classification Using Compression-Based Induction*, tech. report 95-04, DIMACS, 1995.
11. O.V. Kukushkina, A.A. Polikarpov, and D.V. Khmelev, "Opredeleniye Avtorstva Teksta s Ispol'zovaniem Bukvennoi i Grammaticheskoi Informacii" ["Using Letters and Grammatical Statistics for Authorship Attribution"], *Problemy Peredachi Informatsii*, vol. 37, no. 2, 2000, pp. 96–108 (in Russian).
12. N. Merhav and J. Ziv, "Universal Schemes for Sequential Decision from Individual Data Sequences," *IEEE Trans. Information Theory*, vol. 39, 1993, pp. 1280–1292.
13. M. Li et al., "An Information-Based Sequence

Distance and Its Application to Whole Mitochondrial Genome Phylogeny," *Bioinformatics*, vol. 17, 2001, pp. 149–154.

Dario Benedetto is a researcher of mathematical physics in the Mathematics Department at La Sapienza University in Rome. His interests include fluid dynamics and kinetic equations. Contact him at benedetto@mat.uniroma1.it.

Emanuele Caglioti is an associate professor of mathematical physics in the Mathematics Department at La Sapienza University in Rome. His interests include statistical mechanics, kinetic theory, and Information Theory. Contact him at caglioti@mat.uniroma1.it.

Vittorio Loreto is a researcher of physics in the Physics Department at La Sapienza University in Rome. His interests include statistical mechanics and information theory. Contact him at loredo@roma1.infn.it.

Career Opportunities

TOYOTA TECHNOLOGICAL INSTITUTE AT CHICAGO Computer Science at TTI-Chicago

Toyota Technological Institute (TTI-Japan) is founding a new Department of Computer Science (TTI-Chicago) adjacent to the University of Chicago campus. Applications are invited for tenure-track and tenured faculty positions at all ranks.

TTI-Chicago will have exclusive use of the interest on a fund of \$100 million being set aside by TTI-Japan for this purpose. TTI-Chicago will be dedicated to basic research, education of doctoral students, and a small masters program. Faculty members will receive continuing research grants and will have a teaching load of at most one course per year. TTI-Chicago will have close ties with the Computer Science Department of the University of Chicago.

Initial faculty appointments will commence in Autumn 2003, though some appointments may begin earlier by mutual agreement. The Department is projected to grow to a steady-state of thirty faculty by 2007.

Faculty are particularly sought with research programs in

- computational geometry
- databases and data mining
- human-computer interaction
- large-scale scientific simulation
- machine learning
- networking and distributed computing
- software and programming systems
- theoretical computer science

An advisory committee from the University of Chicago and Argonne National Laboratory will recruit the founding faculty, who will then assume leadership to determine the character of the department.

For more information, contact:

Mr. David McAllester
Professor

Toyota Technological Institute at Chicago
mcallester@tti-c.org