Spectral Clustering of the Google Distance

Jan POLAND and Thomas ZEUGMANN
Graduate School of Information Science and Technology
Hokkaido University

Abstract

The World Wide Web provides a huge amount of documents reflecting (parts of) the humans' view of the world and its history. Recently, Cilibrasi and Vitányi have suggested a way to make this data usable for supervised and unsupervised learning, by employing the popular Google search engine in order to define a distance function on pairs of terms. In this work, we propose to apply *spectral* clustering to the resulting distance function, which gives a much faster and fully automatic algorithm for decomposing a list of terms into semantically related groups.

1 Introduction

The Google distance has been suggested by Cilibrasi and Vitányi [1] as a semantical distance function on pairs of words or terms. For instance, for most of today's people, the terms "Claude Debussy" and "Béla Bartók" are much tighter related than "Béla Bartók" and "Michael Schumacher". The World Wide Web represents parts of the world we live in as a huge collection of documents, mostly written in natural language. By just counting the relative frequency of a term or a tuple of terms, we may obtain a *probability* of this term or tuple. From there, one may define conditional probabilities and, by taking logarithms, complexities. Li et al. [4] have proposed a distance function based on Kolmogorov complexity, which can be used for other complexities, such as those derived from the WWW frequencies.

Spectral clustering is an increasingly popular method for analyzing and clustering data by using only the matrix of pairwise distances. It was invented more than 30 years ago for partitioning graphs, based on the fact that by considering the second eigenvector of the Laplacian of the adjacency matrix, one can infer an approximation to the min-max cut of the graph (see e.g. [7] for a brief history).

The main aim of this work is to suggest replacement of the computationally expensive phylogenetic trees used by [1] by a much faster spectral clustering. We will show how state-of-the-art techniques can be combined in order to achieve quite accurate clustering of natural language terms with surprisingly little effort. We will also observe that there is a vast variety of related but slightly different techniques available, which makes it hard to actually choose the method for the particular application. This supports the demand for more theoretical research in this area.

There is a huge amount of related work, in computer science, in linguistics, as well as in other fields. Text mining with spectral methods has been for instance studied in [2]. A variety of statistical similarity measures for natural language terms has been

listed in [8]. For literature on spectral clustering, see the References section and the references in the cited papers.

This paper is structured as follows. In the next section, we will introduce the similarity metric, the Google distance, and spectral clustering, and we will state our algorithm. Section 3 describes the experiments and their results, and Section 4 gives discussion and conclusions.

2 Theory

2.1 Similarity Metric and Google Distance

We start with a brief introduction to Kolmogorov complexity (see [5] for a much deeper introduction). Let us fix a universal Turing machine (which one we fix is not relevant, since each universal machine can interpret each other by using a "compiler" program of constant length). For concreteness, we assume that its program tape is binary, such that all subsequent logarithms referring to program lengths are w.r.t. base 2 (which is also not relevant for our algorithms). The output alphabet is ASCII or UTF-8, according to which character set we are actually using. (For simplicity, we will always use the term ASCII in the following, which is to be replaced by UTF-8 if necessary.) Then, the (prefix) Kolmogorov complexity of a character string x is defined as

$$K(x) = \text{length of the shortest self-delimiting program generating } x,$$

where by the requirement "self-delimiting" we make sure that the programs form a prefix-free set and therefore the Kraft inequality holds: $\sum\{2^{-K(x)} : x \text{ ranges over all ASCII strings}\} \leq 1$. The Kolmogorov complexity is a well-defined quantity regardless of the choice of the universal Turing machine, up to an additive constant.

If x and y are ASCII strings and x^* and y^* are their shortest (binary) programs, respectively, we can define $K(y|x^*)$, which is the length of the shortest self-delimiting program generating y where x^*, the program for x, is given. $K(x|y^*)$ is computed analogously. Thus, we may follow [4] and define the *universal similarity metric* as

$$d(x,y) = \frac{\max\big\{K(y|x^*), K(x|y^*)\big\}}{\max\big\{K(x), K(y)\big\}} \tag{1}$$

This can be interpreted as (approximately) the ratio by which the complexity of the more complex string decreases, if we already know how to generate the less complex string. The similarity metric is almost a metric according to the usual definition, as it satisfies the metric (in)equalities up to order $1/\max\big\{K(x), K(y)\big\}$.

Given a collection of documents like the World Wide Web, we can define the probability of a term or a tuple of terms just by counting relative frequencies. That is, for a tuple of terms $X = (x_1, x_2, \ldots, x_n)$, where each term x_i is an ASCII string, we set

$$p^{\text{www}}(X) = p^{\text{www}}(x_1, x_2, \ldots, x_n) = \frac{\#\text{ web pages containing all } x_1, x_2, \ldots, x_n}{\#\text{ relevant web pages}}. \tag{2}$$

Conditional probabilities can be defined likewise as $p^{\text{www}}(Y|X) = p^{\text{www}}(Y \cup X)/p^{\text{www}}(X)$, where X and Y are tuples of terms and $\cup$ denotes the concatenation. Although the

probabilities defined in this way do not satisfy the Kraft inequality, we may still define complexities

$$K^{\text{www}}(X) = -\log\big(p^{\text{www}}(X)\big) \text{ and } K^{\text{www}}(Y|X) = K^{\text{www}}(Y \cup X) - K^{\text{www}}(X). \quad (3)$$

Then we use (1) in order to define the *web distance* of two ASCII strings x and y, following [1], as

$$d^{\text{www}}(x,y) = \frac{K^{\text{www}}(x \cup y) - \min\big\{K^{\text{www}}(x), K^{\text{www}}(y)\big\}}{\max\big\{K^{\text{www}}(x), K^{\text{www}}(y)\big\}} \quad (4)$$

Since we use Google to query the page counts of the pages, we also call d^{www} the Google distance. Since the Kraft inequality does not hold, the Google distance is quite far from being a metric, unlike the universal similarity metric above.

A remark concerning the "number of relevant web pages" in (2): This could be basically the number of all pages indexed by Google. This quantity is not appropriate for two reasons: First, since some months ago there seems to be no way to directly query this number. Hence, the implementation by [1] used a heuristic to estimate this value, which however yields inaccurate results. Second, not all web pages are really relevant for our search. For example, billions of Chinese web pages are irrelevant if we are interested in the similarity of "cosine" and "triangle" (they would be relevant if we were searching for the corresponding Chinese terms). Therefore, we use a different way to fix the relevant database size: We add the search term "the" to each query, if we are dealing with English language. This is one of the most frequent words used in English and therefore gives a reasonable restriction of the database. The database size is then the number of occurrences of "the" in the Google index. Similarly, for our experiments with Japanese, we add the term の ("no" in hiragana) to all queries.

2.2 Spectral Clustering

Consider the block diagonal matrix $\begin{pmatrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}$. Its top two eigenvectors (i.e. the eigenvectors associated with the largest two eigenvalues) are $[1\ 1\ 0]^T$ and $[0\ 0\ 1]^T$, i.e. they separate the two perfect clusters represented by this similarity matrix. In general, there are conditions under which the top k eigenvectors of the similarity matrix or its Laplacian result in a good clustering, even if the similarity matrix is not perfectly block diagonal [6, 10]. In particular, it was observed in [3] that the transformed data points given by the k top eigenvectors tend to be aligned to lines in the k-dimensional Euclidean space, therefore the kLines algorithm is appropriate for clustering. In order to get a complete clustering algorithm, we therefore only need to fix a suitable kernel function in order to proceed from a distance matrix to a similarity matrix.

For the kernel, we use the Gaussian $k(x,y) = \exp\big(-\frac{1}{2}d(x,y)^2/\sigma^2\big)$. We may use a globally fixed kernel width σ, since the Google distance (4) is scale invariant. In the experiments, we compute the mean value of the entries o of the distance matrix D and then set $\sigma = mean(D)/\sqrt{2}$. In this way, the kernel is most sensitive around $mean(D)$.

The final clustering algorithm for a known number of clusters k is stated below. We will discuss in the next section how to estimate k if the number of clusters is not known in advance.

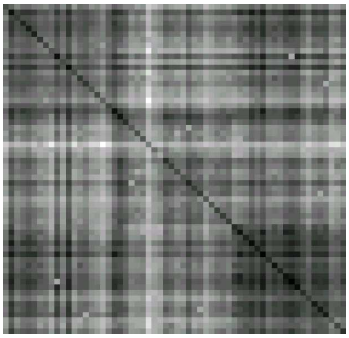

Figure 1: Complexities

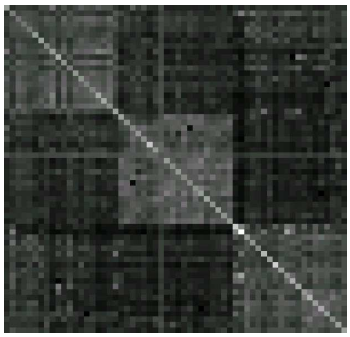

Figure 2: Laplacian

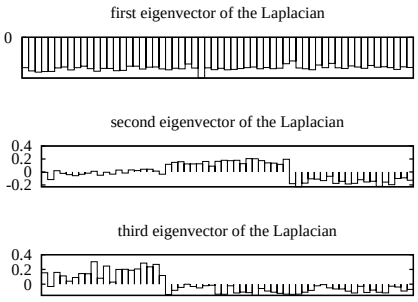

Figure 3: Eigenvector plot

Algorithm Spectral clustering of a word list
Input: word list $X = (x_1, x_2, \ldots, x_n)$, number of clusters k
Output: clustering $c \in \{1 \ldots k\}^n$

1. for $x, y \in X$, compute Google relative frequencies $p^{\text{www}}(x)$, $p^{\text{www}}(x, y)$
2. for $x, y \in X$, compute complexities $K^{\text{www}}(x)$, $K^{\text{www}}(x, y)$
3. compute distance matrix $D = \left(d^{\text{www}}(x, y)\right)_{x,y \in X}$
4. compute $\sigma = mean(D)/\sqrt{2}$
5. compute similarity matrix $A = \left(\exp(-\frac{1}{2} d(x, y)^2/\sigma^2)\right)$
6. compute Laplacian $L = S^{-\frac{1}{2}} A S^{-\frac{1}{2}}$, where $S_{ii} = \sum_j A_{ij}$ and $S_{ij} = 0$ for $i \neq j$
7. compute top k eigenvectors $V \in \mathbb{R}^{n \times k}$
8. cluster V using kLines [3]

3 Experiments

In this section, we present the experiments of our clustering algorithm applied to four lists of terms. First we show step by step how the algorithm acts on the first data set, which is the following list of 60 English words:

`axiom,average,coefficient,probability,continuous,coordinate,cube,denominator,`
`disjoint,domain,exponent,function,histogram,infinity,inverse,logarithm,`
`permutation,polyhedra,quadratic,random,cancer,abnormal,abscess,bacillus,`
`delirium,betablocker,vasomotor,hypothalamic,cardiovascular,chemotherapy,`
`chromosomal,dermatitis,diagnosis,endocrine,epilepsy,oestrogen,ophthalmic,`
`vaccination,traumatic,transplantation,nasdaq,investor,obligation,benefit,`
`bond,account,clearing,currency,deposit,stock,market,option,bankruptcy,`
`creditor,assets,liability,transactions,insolvent,accrual,unemployment`

The first 20 words are commonly used in mathematics, the next 20 words have been taken from a medical glossary, and the final 20 words are financial terms. The matrix containing the complexities is depicted in Figure 1 (large complexities are white, small complexities black). Clearly, the single complexities on the diagonal are smaller than the pairwise complexities off-diagonal. After transformation to the Laplacian (Figure 2), the block structure of the matrix is clearly visible. Figure 3 shows the top three eigenvectors of the Laplacian. The first eigenvector having only negative entries seems not useful at all for the clustering (but in fact it is useful for the kLines algorithm). The second eigenvector separates the medical terms (positive entries) from

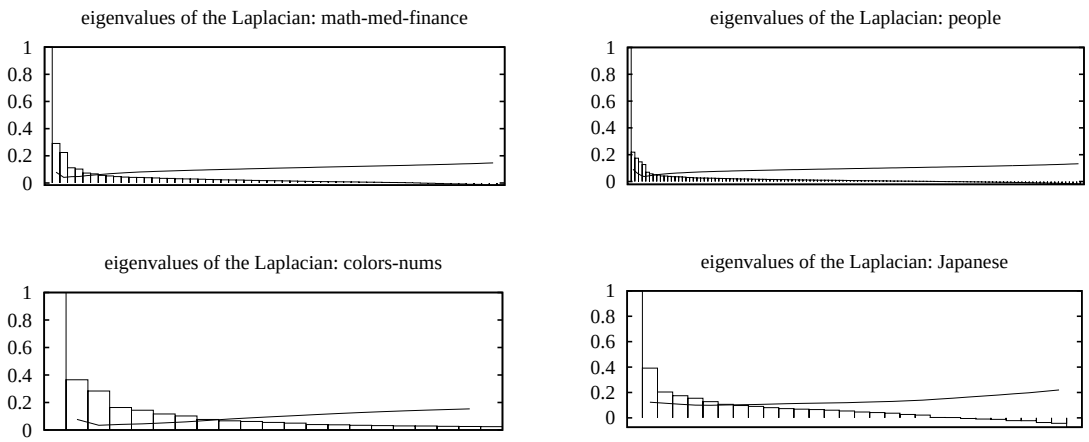

Figure 4: Plots of the eigenvalues of the Laplacian (bars) and the s.s.e. score for determining the number of clusters (lines) for the four data sets

the union of mathematical and financial terms (negative entries). This indicates that the mathematical and financial clusters are closer related than each is related to the medical terms, in a hierarchical clustering we would first split off the medical terms and then divide mathematical and financial terms. The final clustering correctly groups all terms except for "`average`", which is assigned to the financial terms instead of the mathematical terms (this is also visible from Figure 3). As `average` also occurs often in finance, we cannot even count this as misclustering. We stress that the clustering algorithm is of course invariant to permutations of the data, i.e. yields the same results if the terms are given in a different order. It is just convenient for the presentation to work with an order corresponding to the correct grouping.

In case that we do not know the number of clusters k in advance, there is a way to estimate this quite reliably from the eigenvalues of the Laplacian, if the data is not too noisy. Consider again the case of a perfect block diagonal matrix, i.e. all intra-cluster similarities are 1 and all other entries 0. Then the number of non-zero eigenvalues of this matrix is equal to the number of blocks/clusters. If the matrix is not perfectly block diagonal, we may still expect some dominant eigenvalues which are clearly larger than the others. Figure 4 top left shows this for our first example data set. The top three eigenvalues of the Laplacian are dominant, the fourth and all subsequent ones are clearly smaller. (Observe that the smallest eigenvalues are even negative: This indicates that the distances we used do not stem from a metric. Otherwise all eigenvalues should be nonnegative, since the Gaussian kernel is positive definite.)

We propose a simple method for detecting the gap between the dominant eigenvalues and the rest: Tentatively split after the second eigenvalue, compute the means of the eigenvalues in the two groups "dominant" and "non-dominant" (ignore the top eigenvalue, which is always much larger), and calculate the sum square error (s.s.e.) of all eigenvalues w.r.t. their means. Compute this s.s.e. score also for the split after the third eigenvalue, the fourth eigenvalue and so forth. Choose the split with the lowest score. We have depicted the s.s.e. scores in Figure 4 by solid lines. For the math-med-finance data set, the minimum score is at the correct number of $k = 3$ clusters.

The next data set is the "colors-nums" data set from [1]:

purple, three, blue, red, eight, transparent, black, white, small, six, yellow, seven, fortytwo, five, chartreuse, two, green, one, zero, orange, four

Although the intended clustering has two groups, colors and numbers (where "small" is supposed to be a number and "transparent" a color), the eigenvalues of the Laplacian in Figure 4 bottom left indicate that there are three clusters. Indeed, in the final clustering, "fortytwo" forms a singleton group, and "white" and "transparent" are misclustered as numbers.

The next data set,

Takemitsu, Stockhausen, Kagel, Xenakis, Ligeti, Kurtag, Martinu, Berg, Britten, Crumb, Penderecki, Bartok, Beethoven, Mozart, Debussy, Hindemith, Ravel, Schoenberg, Sibelius, Villa-Lobos, Cage, Boulez, Kodaly, Prokofiev, Schubert, Rembrandt, Rubens, Beckmann, Botero, Braque, Chagall, Duchamp, Escher, Frankenthaler, Giacometti, Hotere, Kirchner, Kandinsky, Kollwitz, Klimt, Malevich, Modigliani, Munch, Picasso, Rodin, Schlemmer, Tinguely, Villafuerte, Vasarely, Warhol, Rowling, Brown, Frey, Hosseini, McCullough, Friedman, Warren, Paolini, Oz, Grisham, Osteen, Gladwell, Trudeau, Levitt, Kidd, Haddon, Brashares, Guiliano, Maguire, Sparks, Roberts, Snicket, Lewis, Patterson, Kostova, Pythagoras, Archimedes, Euclid, Thales, Descartes, Pascal, Newton, Lagrange, Laplace, Leibniz, Euler, Gauss, Hilbert, Galois, Cauchy, Dedekind, Kantor, Poincare, Godel, Ramanujan, Wiles, Riemann, Erdos, Thomas Zeugmann, Jan Poland, Rolling, Stones, Madonna, Elvis, Depeche, Mode, Pink, Floyd, Elton, John, Beatles, Phil, Collins, Toten, Hosen, McLachan, Prinzen, Aguilera, Queen, Britney, Spears, Scorpions, Metallica, Blackmore, Mercy

consists of five groups of each 25 (more or less) famous people: composers, artists, last year's bestseller authors, mathematicians (including the authors of the present paper), and pop music performers. We deliberately did not specify the terms very well (except for our own much less popular names), in this way the algorithm could "decide" itself if "Oz" meant one of the authors Amos and Mehmet Oz or one of the pop music songs with Oz in the title. The eigenvalue plot in Figure 4 top right shows clearly five clusters. From the 125 names, 9 were not clustered into the intended groups. The highest number of incorrectly clustered names (4 misclusterings) occurred in the least popular group of the mathematicians (but our two names were correctly assigned). We also observed that the clustering gets disproportionately harder when the number of clusters increases: Clustering only the first 50, 75, and 100 names gives 0, 2, and 5 clustering errors, respectively. We also tried clustering the same data set w.r.t. the Japanese web sites in the Google index, this gave 0, 1, 4, and 16 clustering errors for the first 50, 75, 100, and 125 names, respectively.

The last data set consists of 20 Japanese terms from finance and 10 Japanese terms from computer science (taken from glossaries):

依頼, 為替, 営業, 円高, 株価, 環境, 金利, 景気, 雇用, 購入, 財政, 株価, 落札, 輸出, 税金, 売上高, 破綻, 流動性, 有価証券, 販路, 回路, 画像処理, 画像圧縮, 関数, 近似, 係数, 形式, 論理, 実験, 算術.

The eigenvalue plot in Figure 4 does not clearly indicate the correct number of $k = 2$ clusters. However, when using $k = 2$, only the term "環境" (which means "environment") is non-intendedly grouped with the computer science words.

The computational resources required by our clustering algorithm are much lower than those needed by Cilibrasi and Vitányi's algorithm [1]. Their clustering tries to

optimize a quality function, a task which is NP hard. The approximation costs hours even for small lists of $n = 20$ words. On the other hand, our spectral clustering's naive time complexity is cubic $O(n^3)$ in the number of words. This can be improved to $O(n^2k)$ by using Lanczos method for computing the top k eigenvectors. On our largest "people" data set with $n = 125$, our clustering needs about 0.11sec on a 3GHz Pentium4 processor with the ATLAS library.

4 Discussion and Conclusions

We have shown that it needs surprisingly little effort, just a few queries to a popular search engine together with some state-of-the-art methods in machine learning, to automatically separate lists of terms into clusters which make sense. We have focused on unsupervised learning in this paper, but for other tasks such as supervised learning, appropriate tools are available as well (e.g. SVM). Our methods are theoretically quite well founded, basing on the theories of Kolmogorov complexity on the one hand and spectral clustering on the other hand. Yet, there are many possible modifications and variants of our algorithm. For instance, we could use the Ng et al. spectral clustering algorithm [6], which gives almost the same results as those reported. Or we could try a context dependent similarity measure as suggested in [3], this gives slightly better results except for the Japanese data set. More generally, trying just all combinations of clustering algorithms from [9] with distance measures from [8] gives too many variants in order to evaluate them all empirically and find significant differences. It is therefore desirable to develop an accurate mathematical theory for this and related tasks, such that the choice of distance measure, learning algorithm, and other relevant components can be even more thoroughly theoretically founded in the future.

References

[1] R. Cilibrasi and P. M. B. Vitányi. Automatic meaning discovery using Google. Technical report, CWI, Amsterdam, 2006.

[2] I.S. Dhillon. Co-clustering documents and words using bipartite spectral graph partitioning. In *Proceedings of the 7th ACM SIGKDD Int. Conference on Knowledge Discovery and Data Mining(KDD)*, pages 269–274, ACM Press, 2001.

[3] I. Fischer and J. Poland. New methods for spectral clustering. Technical Report IDSIA-12-04, IDSIA, 2004, Manno, Switzerland.

[4] M. Li, X. Chen, X. Li, B. Ma, and P. M. B. Vitányi. The similarity metric. *IEEE Transactions on Information Theory*, 50(12):3250–3264, 2004.

[5] M. Li and P. M. B. Vitányi. *An introduction to Kolmogorov complexity and its applications*. 2nd edition, Springer, Berlin, 1997.

[6] A. Ng, M. Jordan, and Y. Weiss. On spectral clustering: Analysis and an algorithm. In *Advances in Neural Information Processing Systems 14*, pages 849-856, The MIT Press, 2001.

[7] D. A. Spielman and S. Teng. Spectral partitioning works: Planar graphs and finite element meshes. In *37th Annual Symposium on Foundations of Computer Science*, pages 96–105, IEEE Computer Society, 1996.

[8] E. L. Terra and C. L. A. Clarke. Frequency estimates for statistical word similarity measures. In *HLT-NAACL 2003: Main Proceedings*, pages 244–251, Edmonton, Alberta, Canada, 2003.

[9] D. Verma and M. Meila. A comparison of spectral clustering algorithms. Technical Report CSE-03-05-01, University of Washington, 2003.

[10] U. von Luxburg, O. Bousquet, and M. Belkin. Limits of spectral clustering. In *Advances in Neural Information Processing Systems (NIPS) 17*. MIT Press, 2005.