

Similarity Patterns in Language

Jonathan Isaac Helfman
AT&T Bell Laboratories
Murray Hill, NJ 07974-0636
jon@research.att.com

ABSTRACT

Dotplot is a technique for visualizing patterns of string matches in millions of lines of text and code. Patterns may be explored interactively or detected automatically. Applications include text analysis (author identification, plagiarism detection, translation alignment, etc.), software engineering (module and version identification, subroutine categorization, redundant code identification, etc.), and information retrieval (identification of similar records in results of queries). Patterns are interpreted through a visual language. Squares identify unordered matches (documents with lots of matching words or subroutines with lots of matching symbols), while diagonals identify ordered matches (copies, versions, and translations). Patterns of squares and diagonals have more complex interpretations that identify subtler relationships.

1. Introduction

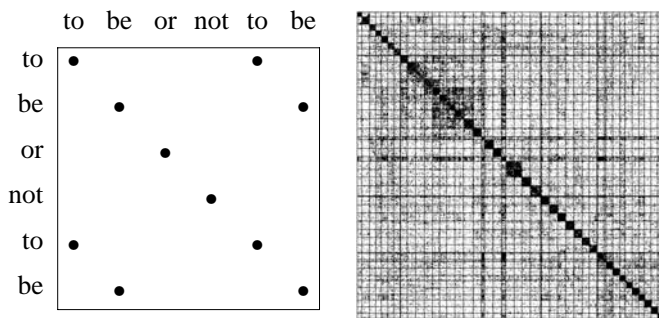


Fig. 1 a) Six words, b) A million words of Shakespeare

The dotplot technique is illustrated in Fig. 1a. A sequence is tokenized and plotted from left to right and top to bottom with a dot where the tokens match. Dots off the main diagonal indicate similarities. While Fig. 1a shows six words of Shakespeare, Fig. 1b shows “The Complete Works” [6]. Grid lines show the boundaries between the concatenated files. Dark areas show a high density of matches. Unlike Fig. 1a, weighting and reconstruction methods are used to display matches from more than one pair of tokens in a single pixel. Weighting prevents matches between frequent tokens from saturating the plot. Additional details of the dotplot technique and associated browser are described elsewhere [2].

Small dark squares along the main diagonal in Fig. 1b are caused by names of characters, which generally match within a single work, but not across different works. The exceptions are the European Histories, which share vocabulary and form a large dark cluster near the upper left. Squares are also formed by the character sequence of Fig. 3a in which the a’s match each other, but not the b’s, and vice versa. In general, one square indicates a high density of unordered matches, usually due to common vocabulary, while two squares indicate a change in vocabulary.

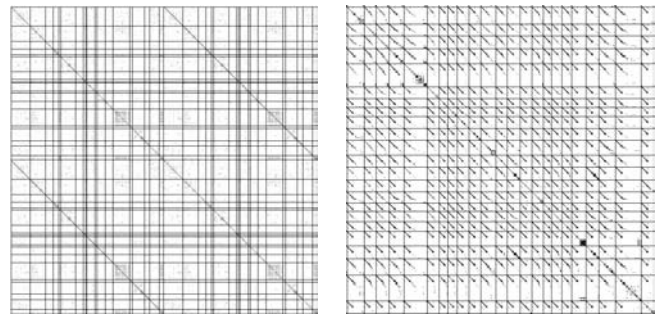


Fig. 2 a) Two versions of xmh (20000 lines of C code)
b) Repeated macros (5000 lines of manual pages)

Fig. 2a plots two versions of the xmh program. The software examples in this paper use C code from the X11R5 and X11R6 Window System [5]. Software is tokenized into lines so that a dot appears where two entire lines of code match. In Fig. 2a, diagonals are formed in the grid boxes that compare the different versions of each file. Diagonals are modeled by the character sequence of Fig. 3b. In general, diagonals indicate ordered matches such as copies or versions.

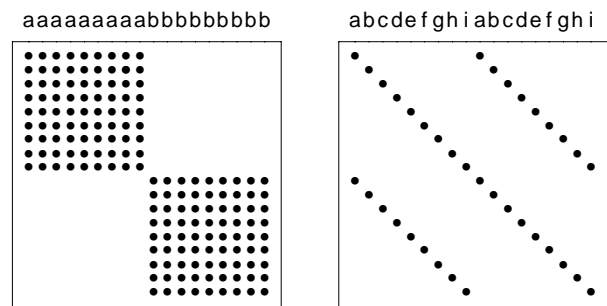


Fig. 3 a) Squares, b) Diagonals

While concatenating two copies creates diagonals, concatenating multiple copies creates a diagonal texture. Fig. 2b shows a diagonal texture formed by repeated macros in X11 manual pages. In the following section, additional patterns of squares and diagonals are enumerated and described in terms of a generalized concatenation operation: shuffling. Similar work is briefly described.

2. Patterns

Fig. 4a shows the manual pages of Fig. 2b along with the C code for two versions of X11 (the dark grid lines show the boundary between the versions). The manual pages appear as a small dark square centered inside a light cross. This feature is modeled by the character sequence of Fig. 4b in which a subsequence of b's is inserted into a larger sequence of a's. Insertions are also identified by broken diagonals. Fig. 5a shows two versions of the X Toolkit code. The second version is larger than the first. New code is indicated by breaks in the diagonals. Fig. 5b models broken diagonals by inserting extra tokens into an otherwise copied sequence. In general, light crosses indicate insertions into unordered sequences, while broken diagonals indicate insertions into ordered sequences.

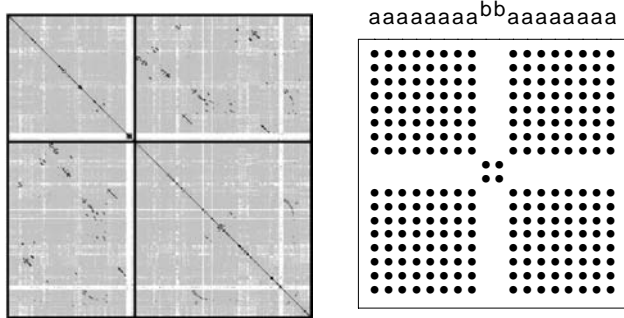


Fig. 4 a) Two versions of X11 (1900000 lines of C Code)
b) Light Cross

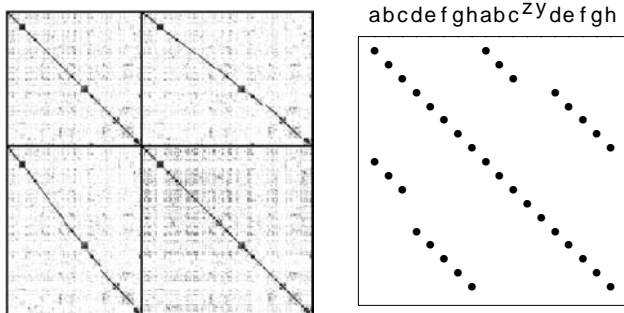


Fig. 5 a) Two versions of Xt (66600 lines of C Code)
b) Broken Diagonals

The two versions of X11 code in Fig. 4a do not plot as diagonals due to reordering in X11R6. A portion of the upper right grid box of Fig. 4a is shown magnified in Fig. 6a. Fig.

6b shows the same sequence that plots diagonals in Fig. 3b, but in a different order. Reordering of versions is typically caused by changes to file pathnames. Reordering is also indicated by square textures. Fig. 7a shows alternating initializations in the file ram.bif. Fig. 7b models square textures with the same sequence that plots squares in Fig. 3a, but in a different order.

In some cases it is useful to automatically reorder sequences. The concatenation order of the files in Fig. 1b was determined by a combinatorial optimization algorithm that emphasizes clusters of similar documents. This is useful in information retrieval systems for visualizing document clusters in the results of a query. Image processing algorithms that identify the clearest diagonal on a strip of dotplots are useful for identifying file pairs in reordered versions.

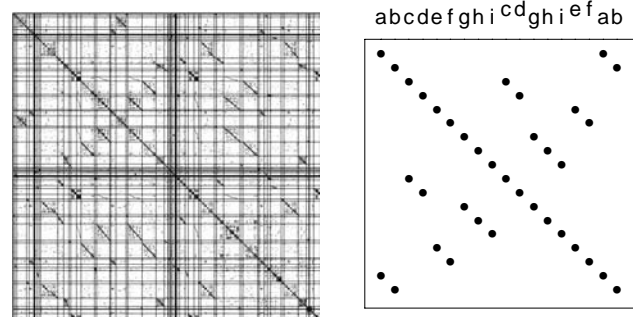


Fig. 6 a) Reordered versions of dix (8000 lines of C Code)
b) Reordered Copies

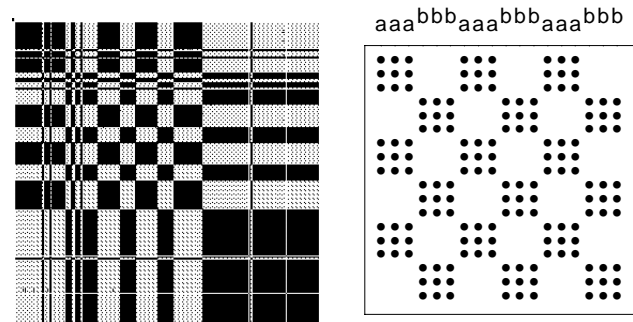


Fig. 7 a) Reordered Initializations (2500 lines of C Code)
b) Reordered Squares

Shuffling is a generalization of insertion and reordering. Fig. 8a shows a seemingly contradictory pattern. Dark squares on the main diagonal indicate that each file has unique vocabulary, while the diagonal texture indicates that each file is a copy. Fig. 8b models the combination of squares and diagonals by shuffling together a sequence that forms squares with one that forms diagonals. Fig. 8a is a chapter of a manual in Dutch, French, German, Italian, Spanish, and Swedish. The input is tokenized into 4-grams.

(An n -gram tokenization is obtained by sliding a fixed-width window over the input, one byte at a time.) The dark squares on the main diagonal are formed by tokens matching within the same language. The diagonal texture is formed by names and numbers that are the same in each language. These words form identical sequences that are shuffled into each file. Alignments computed from the identical sequences are used to construct multi-lingual concordances for terminology research [1].

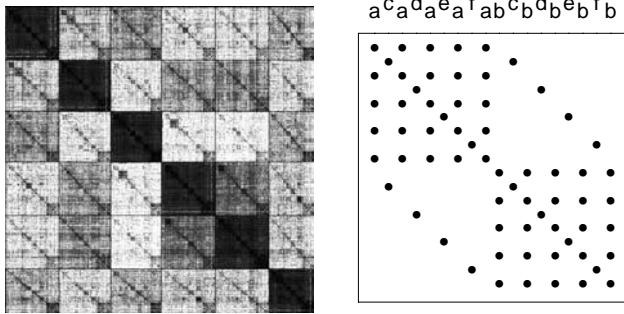


Fig. 8 a) Manual Chapter in 6 languages (1 million 4-grams)
b) Shuffle of Squares and Diagonals

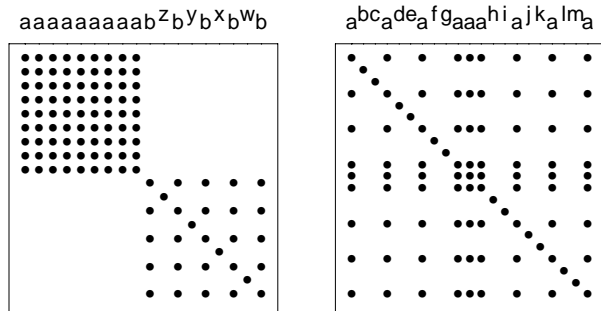


Fig. 9 a) Density Variation, b) Dark Cross

The squares off the main diagonal in Fig. 8a have different grey values or densities of matches. Squares of different densities are modeled in Fig. 9a. In Fig. 8a, the grey value of a square indicates the relative similarities of different languages. Matches between files in relatively different languages (e.g., Spanish and German) show up as light grey, while matches between files in very similar languages (e.g., Spanish and Italian) show up as dark grey.

Shuffling also explains the dark crosses near the center of Fig. 1a, which correspond to the poems *The Rape of Lucrece* and *Venus and Adonis*. Dark crosses are modeled in Fig. 9b. The poems do not have large casts of characters to distinguish them from the other works. Instead they share tokens common to the other works (e.g. doth, eyes, love, life, death, etc.). While light crosses indicate unusual subsequences, dark crosses indicate representative subsequences.

3. Similar Work

Dotplots were originally used in biology to study similarities in genetic sequences [3], [4]. This work is most closely related to string matching and sequence alignment (e.g. suffix arrays, UNIX *diff*, etc.). Algorithmic approaches may be comparable to dotplots for some applications, but for noisy data, the human visual system is still better suited to pattern recognition than non-graphical alternatives. In addition, there are no known algorithmic approaches for detecting diagonal textures (Fig. 2b), reordered diagonals (Fig. 6), or density variations (Figs. 8 and 9).

4. Conclusion

There is a distinction between “control languages” that are used to direct behavior (e.g. programming languages) and “interpretation languages” that are used to understand or assimilate information (e.g. visualization languages). A language for the interpretation of similarity patterns has been presented. It is a language because the primitive patterns (squares and diagonals) have specific meanings (unordered and ordered matches) and the combination operations (copying and shuffling) adapt those meanings in specific ways. Understanding the meaning of the primitives and the combinations allows one to understand an infinite number of patterns. Applications have been described in a variety of fields (text analysis, software engineering, and information retrieval).

REFERENCES

1. Church, K.W. Char_align: A Program for Aligning Parallel Texts at the Character Level. In *31st Annual Meeting of the Association for Computational Linguistics*, Association for Computational Linguistics, June 22-26, 1993, pp. 1-8.
2. Church, K.W. and Helfman, J.I. Dotplot: A Program for Exploring Self-Similarity in Millions of Lines of Text and Code. *Journal of Computational and Graphical Statistics* 2, 2 (June 1993), 153-174.
3. Maizel, J.V. and Lenk, R.P. Enhanced Graphic Matrix Analysis of Nucleic Acid and Protein Sequences. *Proceedings of the National Academy of Science, USA* 78, 12 (December 1981), 7665-7669. Genetics.
4. Pustell, J. and Kafatos, F.C. A High Speed, High Capacity Homology Matrix: Zooming Through SV40 and Polyoma. *Nucleic Acids Research* 10, 15 (1982), 4765-4782.
5. Scheifler, R.W. and Gettys, J. The X Window System. *ACM Transactions on Graphics* 5, 2 (April, 1986), 79-109.
6. Wells, S. and Taylor, G., Eds. *William Shakespeare the Complete Works*. Oxford University Press, 1989.