Discovering Literary *Topoi* by Computer

Ian Lancashire

Topoi are hypothetical entities: conventional building blocks of literary content that writers associated in space and time agree to share. Yet delineating and identifying these 'repeaters' is not easy. Most literary communities lack definition even at the peak of their life cycle. Should an Aristotle be alive to chronicle the behaviour and properties of such a community, that critic will face uncertainty about the most basic matters, such as the readiness of its members—sometimes competing and idiosyncratic—to affirm that any communal properties exist at all. Modern European authors have not had scientific witnesses to catalogue and classify their topical commonplaces. We are left with the texts alone; they must be made to reveal their own *topoi*.

The process of *topoi* discovery begins by recognizing a subset of texts from the entire literature of a language, what may be called a possible galaxy of discourse in a literary universe. Renaissance English drama will do, or 17th-18th-century French novels, or modern poetry. In selecting, we resort to broad and obvious criteria like genre or period.

Next we have to ask what kind of commonplace or 'repeater' a *topos* is? It appears to be more extensive than a word or an idiomatic phrase, less complex than an action or a character, liable to be found in a single sentence or paragraph rather than over a page, and suggestive of a concept or generally-applicable thematic topic rather than a rhetorical figure of speech or syntactic regularity. We are looking for something of a medium order of complexity, like a chemical compound or a strand of DNA rather than a fundamental particle or a molecule. *Topoi* are limited complexities that characterize, neither the language (idioms, for instance, are linguistic entities) nor the world as we make sense of it in history or myth (plots and characters) but the minds of writers as they concentrate on a discrete event or thing or idea.

Modern neuro-psychology gives a useful theoretical name for these 'limited complexities.' Semantic nets consist of a group of concepts, images, sounds, or other 'nodes' of thought linked by association in such a way that bringing up or 'activating' one arouses others in the net, exactly how many to depend on the intensity of the thought.[1] The anatomy of the nervous system offers a parallel physical structure or storage mechanism for such associational nets. Neurons consist of a cell body, long axon, and branching filament-like dendrites that, while not touching other neurons, permit inter-neuron synaptic contact or 'activation.' It is also worth remembering that language processing in the cerebral cortex recognizes the difference between syntactical (what is called above 'linguistic') and semantic functions by locating them in two different places, the first in Broca's area, the second in Wernicke's area. By focusing on semantic elements in defining *topoi*, by distinguishing between them and rhetorical figures of speech, we (as it were) recognize a functional distinction that has been a commonplace of neuroscienee for over a century.[2]

All *topoi* may be semantic nets, but not all semantic nets are *topoi*. *Topoi* are networks shared by a society of texts, not ones unique to any given writer. Research on *topoi*, then, is less psychological than sociological: it aims to identify part of the basis of literary tradition.[3] Those who study *topoi*, a non-random recurrence of semantic nets in a society of texts, engage in a structural analysis that tries to falsify a literary theory proposing that there is such a thing as a 'global text' of texts. Text analysis is thus experimental; it belongs to a field that might be called text science which begins with a hypothesis, develops procedures to test it, and alters that hypothesis according to the results obtained. *Topoi* researchers use text science, then, to test whether intertextual dynamics operate through a 'cloud' of thematic commonplaces.

Computers as Textual Laboratories

The computer is a tool for text science. Computer programs are collections of artificial 'speech acts.' Expressions in a programming language

[1] For representative computational approaches, see Roger C. Schank, "Language and Memory," *Cognitive Science* 4.3 (1980) 243–84.

[2] An excellent introduction to this matter for non-scientists may be found in *The Brain* (San Francisco: W.H. Freeman, 1979), especially two chapters: Eric R. Kandel, "Small Systems of Neurons," pp. 29–38, and Norman Geschwind, "Specializations of the Human Brain," pp. 108–117.

[3] D.F. McKenzie's *Bibliography and the Sociology of Texts* (London: The British Academy, 1986) has actively propounded this view for some time.

are often intended to have immediate and tangible effects, like a conductor's cry "Move to the back, please" in a crowded streetcar. Together, these acts form algorithms, which are experimental procedures for getting a desired result from some initial conditions. The algorithms of interest in *topoi* discovery belong to well-known fields in computer science like text retrieval, processing, and analysis, natural language understanding, and computational linguistics.[4] They have been widely applied in such areas as word processing, online information services, database management, automatic indexing or document classification, and expert systems. Literary scholars do not need to invent new concepts in software to discover *topoi*. Much basic technology has been developed long ago by programmers who wanted a reliable, rapid method of editing long computer programs and is freely available now.

There are three steps in computer analysis of *topoi*:

1) Prepare the texts for searching (do 'text mark-up').
2) Choose the search strategy and conduct the search.
3) Use the results of the search to modify the text mark-up or the search strategy.

This 'looping' or repetitive procedure does not in itself discover anything but only retrieves for the researcher what he or she has actually requested. The benefits of computational analysis are clerical and methodological. Once texts have been put in machine-readable form, they may be searched with far greater speed than may be done manually. More important, the computer carries out with impressive literalness the instructions of the researcher, not infrequently to expressions of astonishment as the true meaning of those instructions becomes clear.

As a drudge, the computer will generate both invaluable evidence of textual structures, and heaps of useless data. Its clerical function, then, is less important than its role in helping critics refine their understanding of what they are trying to do. Critics should not assume that existing programs will relieve them from the need to think out acceptable methodology. I do not see any ready-made computational solution to the problem of *topoi* discovery yet, perhaps because no literary critic has so far tried to describe the problem clearly to a programmer.

Preparing the Texts

It might be thought that the first step in finding commonplaces by computer

[4] For a general survey of the field, see Ian Lancashire and Willard McCarty, *Humanities Computing Yearbook* (Oxford: Oxford University Press, 1988).

is to devise a suitable database structure to receive the results of searching, but that is certainly not the case. Relational database management systems (e.g., Oracle, Ingres, and other SQL-like programs) and traditional systems (e.g., dBase IV) offer only the technology to store and sort on already-known data, such as bibliographical records. There is no controversy about what is and is not a book or about how to name its parts. Only when *topoi* have been identified and classified are they ready for cataloguing. Only then will we know what will constitute the main 'record' (the *topos* name? the concepts out of which *topoi* are built?) and what parts or 'fields' this record will have (if we choose concepts as the building blocks of *topoi*, then one field for each block might list the *topoi* that may be derived with it). Database management systems organize for automatic inquiry the results of discovery but cannot very well generate those results from raw text.

Topoi discovery begins by entering the collection of base texts into computer-readable files with an optical scanner or with a word processor; and the business of preparing a 'textbase' has more to it than appears at first sight.

Legal considerations come into play quickly, because copying a text electronically without permission infringes the author's copyright if the text has been created within the past sixty years or so. This caution may also apply to editions of texts themselves well out of copyright, because any editor's decisions about choice of textual variants, spelling, punctuation, and text format are themselves subject to copyright protection, even if the original author is unknown and the text is ancient. By un-editing the text, by reproducing what exists in the original text rather than what someone has thought the text *really* is, a researcher can avoid legal entanglements as well as ensure a basic text for experimentation that most scholars will accept.

On the other hand, as only a little thought will show, in creating an electronic copy of a text someone determined to be faithful to the original will still be put in the position of providing a substantial amount of new (and thus copyrighted) editorial apparatus. This is so because a reader of a paper book routinely, unthinkingly makes decisions about the status of the text at any point that no computer can yet make. For example, a text-searching program 'reading' through a play-text reproduced verbatim would not be able to distinguish among preliminary material, running-titles, page numbers, stage directions, speech prefixes, and dialogue. Chapter headings, table of contents, footnotes, and epigraphs, in a similar way, would all be blended in with the text of a novel. We rou-

tinely and unconsciously disambiguate true hyphens, end-of-line hyphens, and dashes, or apostrophes and closing single quotes, or 'scare' quotes and marks enclosing words spoken by a character. We know that page numbers mark the location of a section of text in a work linearly from start to finish and recognize that years or ages rendered in arabic numbers in running text do not indicate a new page.

Programs searching a text have to be instructed in advance about all these things. Only by tagging or encoding a text as it is being entered into machine-readable form can a computer be protected from blunders about the nature of what it is recording.

Text mark-up (as tagging is sometimes called) at present has no standards. A group of researchers in the Association of Literary and Linguistic Computing, the Association for Computing and the Humanities, and the Association for Computational Linguistics led by Nancy Ide has recently been funded by the National Endowment for the Humanities to develop a standard mark-up procedure for literary texts. This will probably be based on an International Standards Organization (ISO) standard named SGML, Standard Generalized Markup Language, which has been adopted by publishers in the United States and many other countries as a format for the exchange of machine-readable texts. SGML represents a syntax only. It does not list any actual tags for handling literary texts, but some guidance now may be found in research underway by George Logan and David Barnard.[5]

Tags are simply labels, pieces of code in a rigid format (or syntax) that a program has been told to recognize as labels and *not* as text. Normally a tagging system identifies one character as the start of a tag; this letter cannot appear in the text. I use the opening diamond bracket. Consider Robert Frost's sonnet "Design," prepared with tags.[6]

```
<texttitle "Design">
<textauthor "Robert Frost">
<compositiondate "1936">
<textpoem>
```

[5] D.T. Barnard, C.A. Fraser, and G.M. Logan, "Generalized Markup for Literary Texts," *Literary and Linguistic Computing* 3.1 (1988) 26–31; and D.T. Barnard, R. Hayter, M. Karababa, G. Logan, and J. McFadden, "SGML-Based Markup for Literary Texts: Two Problems and Some Solutions," *Computers and the Humanities* 22 (1988) 265–76.

[6] *The Norton Anthology of Modern Poetry*, ed. Richard Ellmann and Robert O'Clair (New York: Norton, 1973) 212.

```
<poemtype "sonnet">
<stanzatype "octave">
I found a dimpled spider, fat and white, //
On a white heal-all, holding up a moth //
Like a white piece of rigid satin cloth/- //
Assorted characters of death and blight //
Mixed ready to begin the morning right, //
Like the ingredients of a witches' broth/- //
A snow-drop spider, a flower like a froth, //
And dead wings carried like a paper kite. //

<stanzatype "sestet">
What had that flower to do with being white, //
The wayside blue and innocent heal-all? //
What brought the kindred spider to that height, //
Then steered the white moth thither in the night? //
What but design of darkness to appall?/- //
If design govern in a thing so small. //
</textpoem>
```

The terms 'texttitle,' 'textauthor,' 'poemtype,' 'compositiondate,' 'text[poem],' and 'stanzatype' all give information about the text that is not part of the text, and all belong to a special artificial tagging vocabulary. A double virgule marks end-of-verse-line (so as to be distinguishable from ordinary line-ends in prose), and the dash in "broth—" is distinguished from the hyphen in 'heal-all' by a preceding virgule. From a computational point-of-view, every word in the second stanza may be considered to have attached to it the title 'Design,' the author 'Robert Frost,' the date '1936,' the poem form 'sonnet,' and the stanza-type 'sestet.' These six lines will be retrieved should a search be made for text marked with any of these tags.

Until a standard emerges, researchers will use any reasonable tagging notation, including the so-called 'COCOA' markers identified with the Oxford Concordance Program and the COCOA program from which it came.[7] Converting from one set of codes to another may be done later with any number of search-and-replace functions in word-processing programs, text-analysis systems, or string-handling languages such as SNOBOL4. Thoughtful tagging rather than the use of a given syntax is what counts.

[7] See *Humanities Computing Yearbook*, pp. 320–23.

Search Strategies

A search strategy is just an intelligent sieve into which the text is 'poured' and in which parts of the text collect that cannot pass through.

Content analysis involves two kinds of search strategy. In the first, implemented in programs such as the General Inquirer and TextPackV,[8] a researcher lists all *topos*-signalling words or 'strings' in an especially-created thesaurus or dictionary beside all *topoi* to which it might conceivably belong. Then the program checks each word in the text consecutively against the dictionary. If a text word appears in it, the program writes the *topos* name, followed by the location in which it has been 'found' in the text. The second strategy does not look for 'literal' strings or words but rather for generalized patterns or formulae that may contain no actual words at all. For instance, "[infinitive1] + [conjunction] + [negative] + [infinitive1]" would catch Hamlet's "To be or not to be" and an unpredictable number of parallel phrases that may or may not be echoes of the Prince.

Computer-based analysis may seem to work most productively in the first case, when we know what we are looking for. The search function of most word-processing programs, after all, asks the user to specify the word or phrase to be found. If researchers already have a catalogue of *topoi* and their defining features, then thesaurus-driven content analysis will do an acceptable job of amassing all examples obeying those known criteria. Yet a moment's thought will show that this kind of sieve can neither find out new kinds of *topoi* nor improve the defining criteria. If we know what we want to find, can we be said to be discovering anything new when we find it? Automatic retrieval procedures become truly useful when they complement manual searching, not when they duplicate it.

For this reason, *topoi* discovery should also use generalized search algorithms that find word patterns—structures—rather than literals. 'Regular-expression' searching by the so-called *grep* command in Unix, designed for relatively uninflected languages such as English, is probably the best known pattern-matching utility. A regular expression is built from both literal alphanumeric characters (letters and numbers) and 'metacharacters,' which may stand for any zero or more alphanumeric characters or for one or more of a selected list of them. A regular expression in this way may describe a pattern of characters which many words will satisfy.[9]

Three important types of regular expression are wildcards, closure,

[8] See *Humanities Computing Yearbook*, pp. 327–28, 331.

[9] Two text-analysis and retrieval microcomputer programs for MS-DOS sys-

and character class.

wildcard: A question mark may stand for any character. The regular expression 's?ng,' then, will capture 'sing,' 'sang,' 'song,' and 'sung.'

closure: An asterisk may stand for any zero or more occurrences of the character or metacharacter preceding it. The regular expression 'so*n' will capture 'sn,' 'son' and 'soon.'

character class: Square brackets enclose a 'class' of characters that are regarded, for searching purposes, as being identical. The regular expression 's[ou]n' will capture 'son' and 'sun' but not 'sin' (this as well would be caught with 's[iou]n'). Specifying ranges of alphanumeric characters may be done: e.g., '[a-zA-Z0-9]' catches any alphanumeric character, and '[.,;:!?]' any punctuation mark. A caret prefixing characters inside square brackets yields any alphanumeric character except those listed: e.g., 'ho[^aeiou]d' would capture 'hold' and 'hord' but not 'hood.'

By combining all three metacharacters within the same regular expression we are able to match all inflected forms of some words. For example, 's[aio]n[a-z]*' will capture 'sang,' 'songs,' and 'singing.' A pattern matching *most single words* is [a-zA-Z][a-zA-Z]*, which combines closure with character-class metacharacters.

Ad hoc individual searching for regular expressions can yield many surprising new possible 'repeaters' but most will only be single words or phrases which turn out not to be *topoi* but rather different morphological forms of the one word, or syntactic 'frames,' or semi-formulaic and popular turns-of-phrase, both embedded in common speech of the time. For example, we can be sure that matches for common syntactic frames like '[a-zA-Z][a-zA-Z]* is [a-zA-Z][a-zA-Z]*, [a-zA-Z][a-zA-Z]* [a-zA-Z][a-zA-Z]*' (e.g., '*Truth* is *beauty*, *beauty truth*.' or '*Henry* is *intelligent*, *it seems*.') do not belong to the same *topos*. Of course, in reading the output of the search we might well find a syntactic ingredient found in all examples of a *topos* largely defined by a common semantic node and not included in the actual specifications for the search.

Searching for *topoi* means searching for collocations, that is, co-occurrences of words or groups of words. Regularly associated words, collocations, are the semantic networks that I have identified with *topoi*.

Special pattern-matching programming languages like SNOBOL4,

tems produced at the University of Toronto, Micro Text-Analysis System (MTAS) and TACT, an interactive retrieval and category analysis program, both have a *grep* pattern-matching built-in.

which capture most of the functionality of the *grep* function, in addition allow for collocational searches. These can trap the multiple simultaneous patterns that we find in semantic networks. Interactive text-retrieval programs such as WordCruncher,[10] working on pre-indexed texts, or concordance systems such as Micro-OCP, will also handle collocations.[11] Such programs can be instructed to search only for co-occurring words, or collocations, but will only act on well-formed, user-specified search patterns. We have to know in advance what we want to find and be able to state it precisely.

The more general the formula, the more interesting its results. The ideal search engine would operate automatically to catch all repeated semantic networks, *no matter what their content was*. This procedure does not ask the researcher in advance what he or she wants to find. It would be possible, rather, to ask for all instances where *any* three content words[12] collocated within (say) a block of 100–200 words within the entire collection of texts being examined. This procedure could look for all combinations of any three content words found in a given paragraph or stanza that occur anywhere inside all other paragraphs within the corpus.

Consider, for instance, the beginning of Frost's "Design." This has 33 content words, which reduce to 32 after conversion into 'canonical' or 'lemmatized' forms, as might be done by a computer using routine dictionary look-up and a group of morphological rules before the actual search.

assorted => assort	ingredients => ingredient
begin	kite
bright	mixed => mix
broth	morning
carried => carry	moth
characters => character	paper
cloth	piece
dead, death => die	ready
design	right

[10] *Humanities Computing Yearbook*, pp. 357–8.

[11] A concordance is a word-index where each occurrence of a word-type (or keyword) is sorted under its spelling, the so-called headword, along with a context that normally consists of the rest of the line in which the word occurred, and a citation reference to the original text.

[12] Content words are normally considered to be nouns, verbs, adjectives, and adverbs, in contrast to so-called function words, which comprise determiners, auxiliary verbs, prepositions, conjunctions, etc.

dimpled => dimple	rigid
fat	satin
flower	snow-drop
found => find	spider
froth	white
heal-all	wings => wing
holding => hold	witches => witch

The number of possible combinations of any three of these words, while substantial (well over 4000), is not so large that they could not be automatically searched in each other paragraph in a corpus. This would yield many otherwise hard-to-discover combinations in a systematic way.

Yet in a large corpus the number of uninteresting matches (from the viewpoint of *topoi*) would be too many in practice. Almost any combination of any three of 'begin,' 'find,' 'piece,' 'ready,' and 'right,' for instance, would net many matches of no consequence. The critic's time would be wasted in considering most of these if the object were to find conventional elements in Frost's poem.

The procedure needs refining. Several ways of improving the 'sieve' have been used with some success in other applications. Possible collocations could be limited to ones involving *frequent* words in Frost's lines, such as 'spider' (twice) and 'white' (three times). Or the 32 target words could be compared to the total vocabulary of all texts to see which of them in Frost's poem appear to be distinctive—i.e. important to the context, and not shared routinely by other texts—and then the program could search for collocations involving only distinctive words. Other statistical techniques have been used to evaluate just how significant a collocation in fact is.[13]

Other *topoi* may be identified as much by grammatical form as by content words. Hamlet's "To be or not to be," for instance, might be echoed in passages with entirely different content words. In order to catch complexities of this sort, a search pattern of the form "To X or not to X" might be sufficient (employing wildcards for the missing content words), but here again the researcher has to know in advance exactly what

[13] For research on 'automatic document classification' by statistical analysis of the vocabulary of a text, see Martin Phillips' *Aspects of Text Structure: An Investigation of the Lexical Organization of Text* (Amsterdam: North-Holland, 1985) and my "Using a Textbase for English-language Research" in *The Uses of Large Text Databases*, Proceedings of the Third Annual Conference of the UW Centre for the New Oxford English Dictionary (Waterloo, Ont.: UW Centre for the New OED, 1987), pp. 51–64.

patterns to retrieve. A more general solution would have the corpus of texts pre-tagged with part-of-speech and even full syntactic structure, a feat that only a natural-language-understanding system of substantial size and power could manage. The search-engine might then look for any unusual repetition or combination of syntactic patterns found in Frost's lines, not just for collocating words. For this purpose, the textbase would have to be represented to the computer in a form quite unlike the very long linear 'string' it now assumes.

These techniques are plainly experimental. They will undoubtedly identify new *topoi* and new examples of known *topoi*, but perhaps computational methods will also bring about a revision of the grounds of *topoi* creation itself, to the end that we can better understand why some repeating patterns become *topoi*, and not others.

A Sample Search

In 'Design,' Frost draws our eye to something small: a white spider having caught a white moth on a white flower that ought to have been blue. He hints at a sinister power bringing together the unusual flower, a spider that climbs when it should not, and a moth attracted to a whiteness impossible to see in the night. Then, in a surprisingly eerie change of mind, he makes this little death the more striking by doubting the possibility of any design or order in something so easy to overlook.

Is this modern fable a *topos* in 20th-century verse? Perhaps it is, because a similar theme, presented comically, emerges in don marquis' "a spider and a fly," where a fly argues with a spider that he should not eat him who serves "a great purpose / in the world." The fly lives to carry germs on its wings

> into the households of men
> and give them diseases
> all the people who
> have lived the right
> sort of life recover
> from the diseases
> and the old soaks who
> have weakened their systems
> with liquor and iniquity
> succumb it is my mission
> to help rid the world
> of these wicked persons
> i am a vessel of righteousness

scattering seeds of justice[14]

The eloquent spider replies forcefully that he serves "the gods of beauty" by creating webs and that the fly, a servant of merely "utilitarian deities," has no right to starve the superior artist,

a creator and a demi god
it is ridiculous to suppose
that i should be denied
the food i need in order
to continue to create
beauty i tell you
plainly mister fly it is all
damned nonsense for that food
to rear up on its hind legs
and say it should not be eaten

This rebuttal convinces the fly, who can only protest weakly that he could have made a stronger case if he had "had a better line of talk," a point accepted by the spider with the cynical observation,

of course you could said the spider
clutching a sirloin from him
but the end would have been
just the same if neither of
us had spoken at all

a pronouncement that archie the cockroach, at the end of the poem, says makes him "think/ furiously upon the futility/ of literature."

don marquis amusingly touches on Frost's first implication, that a dark design has worked out an end for us no matter what we may have to say about it. A greater poet, Frost gives this almost trite thought a much more sinister twist at the sonnet's close. That marquis read Frost's early version of the sonnet, dated 1912, before he wrote archy's verse sometime after 1916 is unlikely. It is possible, however, that the spider–fly/moth–fate theme existed as a literary *topos* in the world of modern American popular verse.

Suppose, then, that such a *topos* exists. Could a computer draw attention to it in a textbase containing the work of both writers?

The search method of trying every three-word combination in Frost's octave would have retrieved marquis' verse when Frost's (lemmatized)

[14] don marquis, *archy and mehitabel* (Garden City, New York: Dolphin Books, 1960) 40–2.

words, 'spider,' 'wing,' and 'right,' were tried with archy's first 100 words. If the sieve caught not just exact matches, but synonyms, then it would have trapped a fourth match, Frost's word 'design' (i.e. archy's word 'purpose').

At this point in the development of computer tools for text research, it might be expected that 'intelligent' systems, rather than methodical brute-force processing, would be available to literary scholars for their work. To a degree they are, in natural-language understanding systems, but in practice these promise not only to refine the searches we want to do but to complicate them unpredictably. A system that can search for a match to a passage by trawling for the exact words, their canonized word-forms, their known synonyms or antonyms, and even aspects of the syntactical structure of the passage in which they are found would give us too many choices (if it waited on instructions from us) or too much output (if it ran automatically, exhausting all options). The parallel-processing machines on which this kind of search should be made will be within reach of the average scholar in the next few years. Yet until we have machines with something of the intelligence of ourselves—we know when to take short cuts—the simple brute-force approach outlined here might do well enough, both to collect representative sets of *topoi* and to improve theories of *topoi* formation so that we can pose better still better questions.

University of Toronto