Paper 025-30

Publishing Jack Vance: The SAS® System as a Tool for Literary Analysis

Koen Vyverman, SAS, Netherlands

ABSTRACT

The Vance Integral Edition (VIE) is a non-profit organization that aims to publish the entire works of the American author Jack Vance in a limited, durable, and definitive edition. As each of Vance's 136 novels and short stories progresses through the VIE workflow—-from scanning and digitizing, via restoration and proofing, all the way through to typesetting and printing—-the SAS System has proved to be invaluable. SAS software has been used for comparing and analyzing the contents of the various file formats that are involved (flat-file, Word document, RTF), for keeping track of text changes, for reporting in Word and Excel formats, and for providing analytical insights into some prickly questions of textual and stylistic integrity. This paper discusses the major VIE processes where SAS came to the rescue, with ample segments of code and sample output.

INTRODUCTION

> *"Beyond question you are a person of discernment. Still, all balanced against all, the works to which I refer make demands of those who would appreciate them. The metaphors sometimes span two or three abstractions; the perorations are addressed to unknown agencies, the language is archaic and ambiguous...In spite of all, the works exhale a peculiar fervor." (Vance, 2005b)*

This paper documents what might be the first-ever application of the analytical powers of the SAS System to a project that is strictly literary in nature: using SAS software to help publish an integral edition of the works of a great, yet notably undiscovered, American classic, Jack Vance.

The paper presents both the literary aspects of the project—insights into the VIE project phases and problems—and the technical aspects—the hard-core SAS programming that shows how the problems were addressed. It is not possible to provide all the SAS code that was developed in the context of the VIE project. However, this paper outlines the approach and includes some essential code and sample output.

The paper begins by introducing Jack Vance, his work, and the VIE project. Next follows a general overview of the workflow and processes within the VIE, focusing on the major phases of text treatment. The discussion focuses on where and how in the VIE process SAS software helped to make the set of VIE books as good as it can possibly be. The paper concludes with a humorous divertimento and closing remarks.

The code snippets that are included in this paper were developed and tested using SAS 9.1 on a Microsoft Windows 2000 Professional operating environment in conjunction with Microsoft Office 97 software. Dynamic Data Exchange (DDE) to and from Microsoft Word and Microsoft Excel was used liberally and without constraint, whence a little warning: here be dragons!

THE VANCE INTEGRAL EDITION

> *"I see that my life has been somewhat stagnant, even self-centered. I have gloated privately over treasures of literature which should have been shared with others. Now I wish to produce and stage some famous masterpieces of ancient Earth. You ask, where are these fabulous classics to be found? I reply, they are here with my collection of rare books, not fifty feet from where we sit." (Vance, 2005b)*

ABOUT JACK VANCE

Jack Vance was born John Holbrook Vance in 1916. He is the author of over 60 novel-length books, and of even more novellas and shorter stories. He wrote his first published stories while sailing in the Pacific for the United States Merchant Marine in World War II. Therefore, his work, like that of Melville or Conrad, is recognizably that of a sailor. In the years after the war, hoping to earn a living with his pen, he turned his hand to various genres: mystery, fantasy, and—lacking a more suitable general category in which to pigeonhole the major part of his literary output—science fiction. Vance himself insists that he is not a science fiction writer, and indeed, the "science" element in his stories is largely absent. Although most of his "sci-fi" works are set on other worlds and space travel is available, the

technology that is involved is of no interest to Vance. Instead, he focuses on the social, political, religious, and philosophical aspects of the worlds and cultures that he paints in his inimitable style, and in which his colorful characters go about their business. Vance's often un-heroic heroes find their worlds populated by inventive scoundrels, philosophical innkeepers, megalomaniac foes, scheming priests, and haughty aliens. Vance's storytelling style has been described as being reminiscent of Swiftian social satire set on extravagant, far-off worlds, with dialogue that can be as hilariously funny as anything that P.G. Wodehouse ever wrote. Vance is one of those rare writers who is truly a stylist; he imbues his readers with a new language that is robust, subtle, musical, and inventive in the highest degree. Few writers use such an extensive vocabulary, or use words so effectively. Vance's capacity to evoke atmosphere is legendary among his readers, and unsurpassed in all of literature.

From this modest beginning, Vance's writing evolved into one of the most original and powerful oeuvres of the twentieth century. For the past fifty years he has quietly produced a variegated, unforgettable—in fact, astonishing—series of world-class masterpieces. Despite this, few, even among science fiction readers, have heard of him. Most of his work is out of print, and more of it sells in French or Dutch translation than in English. Yet Vance is a writer to be classed only with twentieth century giants such as Solzhenitsyn. His work is not only of the greatest human importance, it is crafted by a poet, and—most significant of all—it is unfailingly, even sinfully, entertaining.

How is it possible that such an artist has gone unrecognized, though consistently published for over half a century? There are two reasons. The first is that, because Vance's works fail to be "typical" science fiction, they fail to interest readers of this genre. Yet his books are found only in science fiction sections, if at all, and are obscured by inappropriate cover art or editor-supplied titles. Though promoted as a "science fiction classic of the Golden Age," (meaning the 1950s) the target market fails to support re-printings, and his books remain generally unavailable. This same market has kept other writers continually in print, and no wonder; their work *is* science fiction. It is made of speculative, futuristic, not to say grotesque, elements, which may be fascinating, but which are without the deep human import that can provoke full-fledged literary expressiveness. Vance's concerns are elsewhere. He is engaged by the profundities of human, social, and natural reality—as much as any of the great novelists of the 19th century.

The second reason Vance goes undiscovered is his lack of pretension. His unique aim is to satisfy readers. His works do nothing to flatter engagement with contemporary critical and political fads, and yet they are indeed formally adventurous and pregnant with political and social import. However, Vance handles these aspects like any true artist; he hides his artfulness and makes his messages consubstantial with his stories. He is a "story teller" *par excellence*. For these reasons, plus his reluctance to engage in self-promotion, Vance remains undiscovered by his true audience, and unread by those who might appreciate him at his full worth.

ABOUT THE VANCE INTEGRAL EDITION PROJECT

The Vance Integral Edition project will alter this situation. The VIE project began in 1999 as nothing more than the conviction that an integral edition of Jack Vance ought to exist. Paul Rhoads, who advanced the idea, and a number of other long-time Vance fans got together at the Vance family's Oakland hill-top house and defined the project goals. The VIE was established as a California-based, non-profit corporation with the purpose of creating a complete and correct Vance edition in 44 volumes—a permanent, physical archive of Vance's work, doubled by digital texts which, through the Vances, are being made available to all present and future Vance publishers. The sets were sold via the Internet, on a subscription-only basis. Subscribers were asked to pay part of the subscription fee up front; this money was then used to cover the cost of running the project and preparing the books. The idea was to have volunteers do most of the actual work. The work of the various volunteer teams was coordinated mostly via mailing lists and reports on the project Web site. Given the volume of output that Vance generated over the past six decades, in its early days the VIE project must have seemed sheer madness to some who accidentally stumbled upon it. However, two years later there were 300 active volunteers, a management group of 25 people, and hundreds of subscribers—most of them volunteers.

The sheer existence of the VIE is a demonstration of Vance's artistic powers; what other author could inspire hundreds of readers to volunteer tens of thousands of hours over several years? VIE volunteers are men and women of all ages from around the world, with different skills. However, they have two things in common: they have read, and re-read, and re-read again, all the Vance that they have been able to collect, and they participate in the VIE project out of a sense of gratitude to an author who has so greatly enriched their lives. Vance is neither a generational nor a local fad. Once discovered, his work becomes a life passion.

The VIE project is perhaps the most culturally innovative use of the Internet ever. Because it is both world-wide and all-volunteer, it has been able to harness a multitude of diverse talents and great reserves of energy. For example, all of Vance's 135 texts have been digitized—from a story of under 2,000 words, to a novel of over 190,000 words, a total of some 4,400,000 words—including several never-before-published stories. In addition, these digital versions

have been proofread about three times—over 400 jobs. However, this is only the tip of the iceberg. The texts have also been corrected, typeset, and further proofread by teams of ten people. All texts have been corrected under the aegis of the author (Vance, now in his late-eighties, is still writing, though he suffers from blindness), his wife Norma, and son John. The VIE is therefore the authorized version of Vance's work, restored from original manuscripts, typescripts, galleys, or, when these were not available, by comparison of different published editions and information from the Vances.

Now, five years after that initial Oakland meeting, all the feverish activity has subsided, and despite the daunting dimensions of the task it had set for itself, the VIE project has now culminated in the printing and distribution to subscribers of over 500 sets of 44 volumes each of the integral works of Jack Vance (Figure 1). About 50 sets were donated to important libraries all around the world by Vance aficionado Paul Allen. All of those involved in the project can now sit back, relax, and re-read their favorite author in an edition that would have been an impossible dream a mere decade ago.

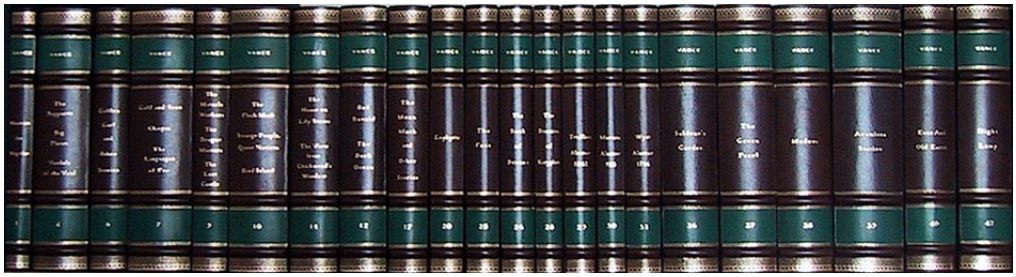

Figure 1. Half of the 44-Volume VIE Leather-Bound Deluxe Edition

THE VIE WORKFLOW—AN OVERVIEW

> *There are also those who, like the author, ensconce themselves on a thunderous crag of omniscience and, with protestations of humility which are either unconvincing or totally absent, assume the obligation of appraisal, commendation, derogation or denunciation of their contemporaries. Still, by and large it is an easier job than digging a ditch. (Vance, 2005c)*

DIGGING THE DITCH

This section of the paper provides a high-level overview of the various phases of text treatment in the VIE project. The discussion focuses on the major milestones that each of the 135 texts went through, from typescript or published edition to printer-ready, VIE digital PDF file. The VIE process was not something that was immediately conceived when the project began. Rather, the steps were added to and perfected in an organic manner over the years. As volunteers gained experience and certain recurring issues became apparent, tools and processes were developed to address issues and problems. The process described below is a summary of what was eventually determined to be VIE best practice.

DIGITIZATION

The VIE project began at the 1999 Oakland conclave by researching the different published editions of each of the VIE texts and listing the preferred editions. Volunteers who had access to copies of these specific editions were sought out. These people started digitizing the texts by using scanners to produce TIFF files of each page, and by running the TIFF files through optical character recognition (OCR) software to extract the text as a flat file. In order to accommodate special formatting of textual elements, the text was then put into a Microsoft Word document. The Word format was chosen because the software is widely available and commonly used—only a minority of VIE volunteers were familiar with mark-up languages such as LaTeX and HTML.

This seemed like a reasonable plan, until the team discovered that OCR software has limitations that may lead to errors. Getting accurate results when using OCR software is dependent on the quality of the scans, which can be affected by the quality of the paper that a text was printed on. For example, if an edition was printed on cheap paper, then the OCR software might erroneously interpret a blurred 'r' that is followed by an 'n' as an 'm', thus potentially changing a word like "stern" into "stem". Figuring out how to handle these frequent OCR bloopers—which the VIE baptized as "scannos" by analogy to "typos"—led to the invention of the Double Digitization technique.

The Double Digitization technique corrects many of the difficult-to-control errors that are endemic to using OCR software, including recognizing wrong words that are nonetheless real and even plausible in context, and missing words and lines. Double Digitization involves running each text through OCR software three times, preferably using three different OCR applications, or using TIFF files with significantly different brightness and contrast settings. Using different OCR packages or different TIFF scans from the same edition generates output with scannos in different places. The three resulting Word files are then assembled into a single document by using the Compare and Merge Documents function in Word to show exactly where one text differs from the others. By merging the three documents, the errors cancel out one against the other. Double (actually triple) digitization is a labor-intensive solution, but it turned out to be extremely effective in removing scanning errors from the digitized texts. Richard Chandler, a Professor of Mathematics at the University of North Carolina, headed the team that did this work.

PROOFREADING

To ensure that the resulting Word documents were faithful reproductions of the preferred editions of the texts, each document was then proofed a number of times by different volunteers. At this stage, people who had printed editions of the text that they were proofreading were asked to read the corresponding Word document word-by-word against their book edition, and to enter endnotes (Figure 2) in the Word document to capture any real and perceived discrepancies, thus keeping a record of all the issues.

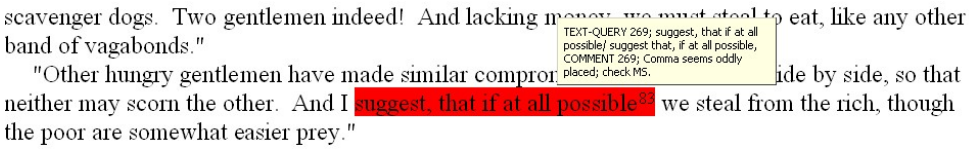

Figure 2. A Sample Endnote

TECHNO-PROOFING

Soon it became apparent that despite having six people proofread a text, in some cases when a seventh person proofread the same text, that person found one or two issues that all the previous proofreaders had inexplicably failed to notice. Take, for example, a situation in which a secondary character appears once in the opening chapter of a novel and then again 300 pages later. If that character's name was spelled subtly differently in both places, then no one except those gifted with an eidetic memory would spot the problem. Therefore, it was decided that to capture such subtle spelling variations—and hence, potential errors—the VIE needed to subject the texts to another correction mechanism, which they themselves invented.

In VIE jargon, "Techno-Proofing" is the process of using mechanical aids in the text proofing. Techno-Proofing involved using specially created, Vance-specific, dictionary filters to generate lists of suspect words from each text. These lists were produced by Ian Davies, a New Zealand volunteer. In addition, the Techno-Proofing team used the Vocabulary/Dictionary Analysis Engine (VDAE), a SAS-based tool developed by the author of this paper, that works with the complete and constantly updated VIE file-archive. The VDAE allows any word or other textual feature to be located and studied in several ways, and takes the form of an interactive Excel spreadsheet. The VDAE is discussed further in the "Dictionary Spreadsheets" section. Word lists and VDAE files were then studied by a special team of volunteers, headed by Australian volunteer Ron Chernich, and Techno-proofing endnotes were added to the Word documents where appropriate (Figure 3).

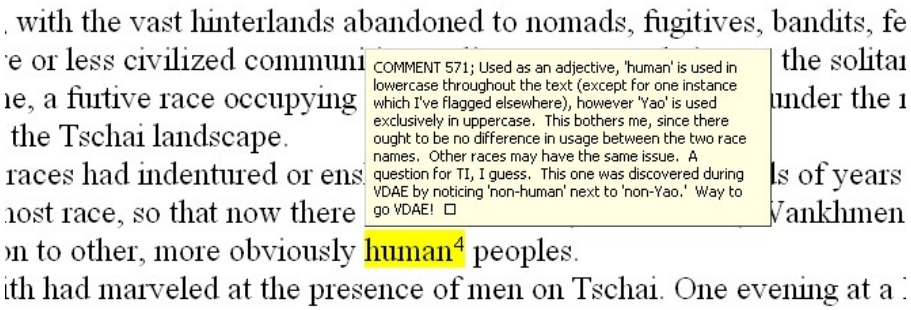

Figure 3. A Sample Techno-Proofing Endnote

TEXTUAL INTEGRITY

As indicated in ”The Vance Integral Edition” section, a significant part of the VIE’s *raison d’être* is the restoration of Vance’s texts based on manuscript evidence. Unfortunately, over the past 60 years, most of Vance’s stories have been tampered with to varying degrees, and sometimes quite severely. For example, publishers typically have an editor go over a manuscript prior to book publication. These editors, feeling compelled to justify their salaries, made many gratuitous changes to words, punctuation, and paragraphing, thereby degrading the inventiveness of Vance’s vocabulary and changing the rhythm and flow of his prose. In order to make the VIE books as enjoyable as possible, it was obvious from the start of the project that Vance’s unique writing style needed to be restored. Furthermore, the VIE decided that it would not fall into the same trap as Vance’s previous unsolicited editors, and that all restoration—or, in VIE jargon, “textual integrity (TI)”—would have to be based on evidence.

Sources of reliable information that pertains to the texts vary considerably. The VIE Textual Integrity team established the relationships of the published editions to each other and assessed the multifarious manuscript evidence. Many Vance manuscripts are held by the Mugar Library Special Collections in Boston, and the VIE has been at work there. In recent decades Vance has been writing with a computer, but many of his older floppies were found to be corrupted, proving the fragility of digital archives! However, VIE volunteers in several parts of the world were able to decode some of the digital archives and to reestablish manuscript evidence for some of the recent works. TI was a monumental job; a team of over 20 people was devoted to it, over a span of three years. Alun Hughes, who is Head of Learning and Information Services at the University of the Highlands and Islands Millennium Institute in Inverness, Scotland, headed the Textual Integrity Team.

To help the TI team with their restoration work, the author developed a tool that was christened the Incredible String Retriever (ISR). This tool is discussed in “The Incredible String Retriever” section. Essentially, when a TI worker came across a word or expression that required further investigation, he or she could send a query to the ISR and receive a report that listed all occurrences of that word or expression within context, across the entire Vance oeuvre. Studying the contextual evidence from other texts might then help settle the issue at hand. The outcome of the TI work was a set of TI propositions, entered as endnotes in the Word document that contained the text (Figure 4). Together with a proposition to change or correct a word or phrase in the story, the TI worker included a solid argument, based—if possible—on manuscript evidence to justify the change. Some of the TI propositions were trivial. For texts with good evidence—for example, a typescript with holographic corrections in Jack’s own handwriting—the case for undoing a publisher’s editorial changes is easily made. Other TI propositions were not so clear-cut. For these, the VIE project management appointed a TI Board to ultimately resolve these sticky issues. During the TI Board review, the propositions were either accepted or rejected, and the story assumed its final restored form, still in a Word document format.

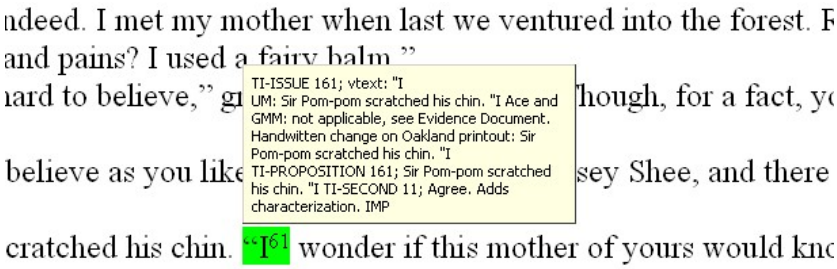

Figure 4. A sample TI Proposition Endnote

Another interesting issue that the VIE faced during the TI work concerned those stories for which the textual evidence was confined to early publications in the 1950s and 1960s pulp magazines. Many of Vance’s early short stories, and also quite a few novels (serialized), appeared in the illustrious pages of magazines such as *Startling Stories*, *Astounding*, and *Galaxy*. Real manuscript evidence was scarce for these early texts, so the VIE had to resort to copies of these old pulp magazines in order to restore the text. The pulp magazines, however, typically had a two-column page layout and did not particularly like text that ran without breaks for more than a full column; this kind of layout provided no resting point for their readers’ eyes. As a result, when they set the text, they introduced breaks in the middle of paragraphs to split a long piece of narrative text into multiple shorter ones. On the positive side, the pulp magazines were cost-conscious and had no money to waste on editors. Hence, the pulp publications generally are faithful reproductions of the manuscripts that Vance submitted, modulo occasional typos and the splitting of prose to suit the columnar format. An analysis of the length of narrative paragraphs was performed, confirming the above. Details follow in “The Purple Peaks of Pulp” section.

COMPOSITION

Now that the text was ready from the restoration phase, a small team of volunteers with typesetting experience—the Composers—imported the stories into Adobe InDesign to fine-tune the text flow, page layout, fonts, character kerning, ligatures, and so on, in order to ensure an aesthetically pleasing reading experience. The texts were set in Amiante (Figure 5), a typeface designed specifically for Vance's prose. Amiante is not a variation of presently available book fonts. It was designed by the VIE Editor-in-Chief Paul Rhoads, who is a painter and a sculptor.

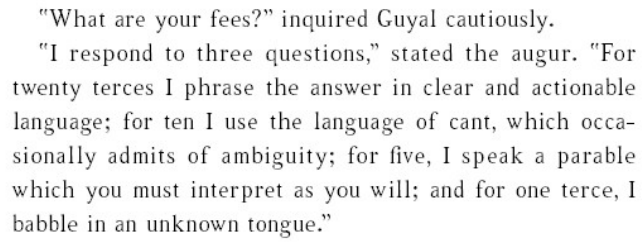
"What are your fees?" inquired Guyal cautiously.
"I respond to three questions," stated the augur. "For twenty terces I phrase the answer in clear and actionable language; for ten I use the language of cant, which occasionally admits of ambiguity; for five, I speak a parable which you must interpret as you will; and for one terce, I babble in an unknown tongue."

Figure 5. The Amiante Book Font (Vance, 2005a)

As the first texts were being typeset, several problems began to emerge. The first problem had to do with hidden artifacts in the Word documents, such as invisible leading spaces that subtly warped paragraph indentation, en-dashes that posed as hyphens jinxing word hyphenation, and non-printing control characters that wreaked havoc, all of which affected the layout of the text after it had been imported into Adobe InDesign. The section "Detecting Hidden Garbage in Word Documents" describes a SAS tool that was developed to deal with this matter.

Also, importing text from Word documents into InDesign did not always work flawlessly. Occasionally, lines of text were inexplicably dropped. Addressing this issue resulted in another SAS development, the RTF Transformer, and adding a subsequent phase of integrity checking to compare vocabulary counts before and after the transition from Word to InDesign. With the texts being composed in InDesign, the next question was how to read these files into SAS. InDesign was used to output PDF files that were sent to the VIE printers in Milan. However, InDesign can also export to RTF, which is a file format that Microsoft Word understands. This enabled SAS processing through DDE. The Amiante fonts make liberal use of ligatures. A ligature is a single character in a typeface that represents multiple characters in a way that cannot be achieved by simply adjusting the kerning. For example, in Figure 5, the word 'five' consists not of four, but of only three characters. The first character, which looks like 'fi', is a ligature. The RTF exports from InDesign contained these ligatures, and before vocabulary frequencies could be calculated from them, the ligatures had to be undone. More information about the RTF Transformer is in the section "The Transmogrification of RTF." The process of using the RTF Transformer output in an integrity check on the freshly composed story became known as the initial RTF Diff—"diffing" is VIE jargon for comparing different versions of a text.

POST-PROOFING

Whenever a text was composed in InDesign and a PDF file was derived, another intensive round of proofreading took place, typically by teams of up to 10 volunteers. A special management team, often with the help of the text's TI guru, reviewed feedback from proofing the PDF files to decide which post-proofing issues had to be fixed. In an iterative way, the VIE Composition team eventually produced a final InDesign file for each story. In a final RTF Diff review, the RTF Transformer was again used to compare vocabulary counts between the initially composed text and its final InDesign version.

BOOKING

Finally, all the contents of each of the 44 VIE volumes cleared the final RTF Diff check. "Booking" of the volumes was the last task that was left to do. For each volume, an InDesign book file was set up to collect various stories in some cases or just a single novel in others. Booking included adding contiguous page numbering, a table of contents where applicable, front matter, and a frontispiece etching by Paul Rhoads. PDF files were derived and sent to the VIE printers, Areagroup Media in Milan, Italy, and so-called 'blues' were printed from them. After a final, thorough check of the blues and some last minor corrections, the VIE sets were printed and were distributed to subscribers and libraries worldwide.

TEXTPORT

Finally, all the changes that were made to the texts in the InDesign phase and later had to be incorporated into the final Word documents as they came out of the TI phase. This process of retrofitting the Word documents to exactly

reflect the printed VIE volumes was known as Textport. Again, due to the widespread availability of Microsoft Office, the Word format was chosen to file the corrected VIE texts. A digital archive was given to the Vance family. The RTF Transformer and the Word garbage detector also played an important role in Textport, to ensure that the textual contents were exact, and to cleanse the documents of unwanted artifacts.

DICTIONARY SPREADSHEETS

> *"I find herein a wonderful beauty," he told Pandelume. "This is no science, this is art, where equations fall away to elements like resolving chords, and where always prevails a symmetry either explicit or multiplex, but always of a crystalline serenity." (Vance, 2005a)*

VDAE SPREADSHEETS

As introduced previously (see the "Techno-Proofing" section), the Vocabulary/Dictionary Analysis Engine (VDAE) was developed to mechanically detect typos, scannos, and spelling variations in a given Word document. The VDAE and all the other tools described in this paper make use of Dynamic Data Exchange (DDE) to manipulate Microsoft Word documents and RTF files straight from the SAS DATA step environment. Also, all Excel workbooks and Microsoft Word reports were created via DDE. For a beginner's guide to DDE, read Vyverman (2002) and Viergever & Vyverman (2003). More advanced DDE topics are discussed in Vyverman (2003).

The SAS tools thrive on a SAS database that was dubbed TOTALITY—after a mysterious, omniscient, and rather indigestible creature from Vance's *Cugel the Clever* novel. TOTALITY was loaded via a live feed from the VIE project file archive. As volunteers finished a job on a particular text, they mailed their pieces to John Schwab, Master Archivist, who applied a strict naming convention and archived the files. For contingency reasons, three back-up archives were maintained in various geographic locations. A SAS job ran one of these back-up archives. The filenames of incoming Word documents were parsed and interpreted to determine which documents were improved versions of texts that already resided in the database. These documents were then opened via DDE, cleaned of extraneous matters such as endnotes, and saved as flat files. The flat files were then processed, and TOTALITY's various vocabulary, summary, and metadata tables were updated to reflect the most recent state of things.

In the loading process previously described, the author used SAS macros (presented in Viergever & Vyverman (2003)) to manipulate the Word documents. For example, consider the process of removing endnotes from a Word document. This was done using the `Find and Replace` macro.

```
%* Remove all endnotes. I.e. do a global find and replace by nothing.         *;
%findrepl(
          findwhat=^e,
          replacby=
         );
```

The find string '^e' locates every endnote in the document, and the macro replaces each occurrence with nothing, thereby effectively deleting all endnotes.

For the vocabulary scan of each text, the `translate` function was used to remove any character that was not determined to be a legitimate part of a word. The following example shows how punctuation marks, various symbols, and numerous unwanted control characters represented by their hexadecimal values, such as 0A and 0B, were stripped out of each document.

```
 paragraph=left(compbl(translate(_infile_,' ',';:",.?!()<>[]{}*^@~#$%_+=`')));
 paragraph=left(compbl(translate(paragraph,'
','000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F'x)));
```

After this pass, the text was parsed into words, and for each word a number of flags were derived: Does the word contain an uppercase character? Or a single quote? Does it have a hyphen? Does it contain only capitals? Or is it all digits? In this manner, a descriptive vocabulary data set was constructed for each of the VIE texts. Flags and other attributes of interest were added to the textual analysis: the number of times the word appears in the parsed story; the number of times the word appears in the entire VIE; whether a word that contains a capital appears in lowercase elsewhere in the story, or perhaps in another VIE text; whether a word that contains a hyphen appears without a hyphen elsewhere in the story, or maybe elsewhere in the VIE? The vocabulary tables were exported to Excel workbooks via the `%sastoxl` macro as described in Vyverman (2002).

```
%sastoxl(
         libin=vietxt,
         dsin=%str(&vcode._word_freq),
         savepath=%str(d:\koen\vie\vie xl),
         savename=%str(&filenam),
         stdfmtng=1
        );
```

These Excel workbooks, known as VDAE spreadsheets (Figure 6), were then used by the Techno-proofing team volunteers. By switching on the Autofilter functionality in Excel, clever combinations of the various flags and attributes made it easy to spot spelling variations within the same text, inconsistencies in the hyphenation of words, plain typos, and so forth.

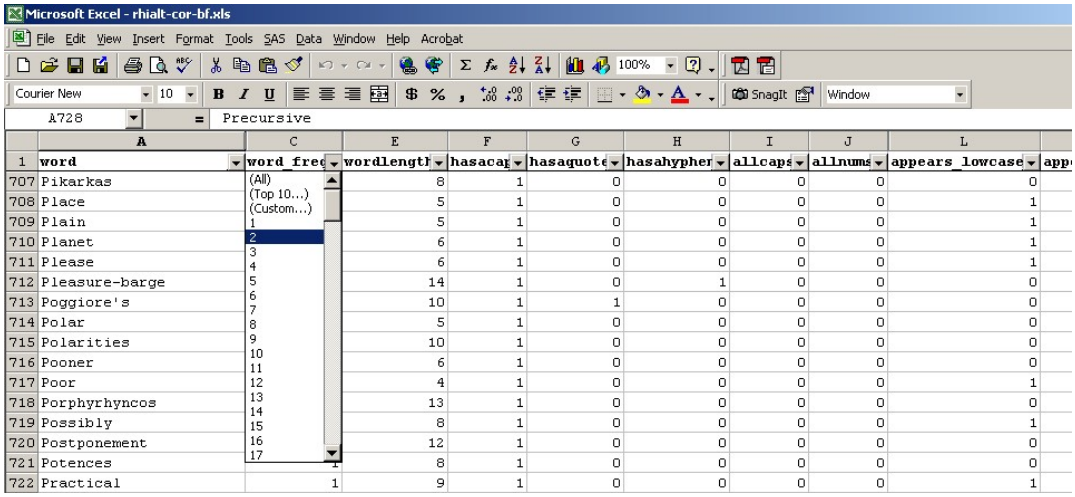

Figure 6. Part of a VDAE Spreadsheet for Vocabulary Analysis

A COROLLARY: VOCABULARY SIZE COMPARISON

With plenty of statistics on Vance's vocabulary available in the TOTALITY database, the VIE wanted to compare the richness of Vance's prose to that of other authors. Project Gutenberg (PG) is an online repository that holds hundreds of digitized and proofed copyright-free texts by many great authors—texts that are accepted literary classics. Eighty texts were selected from the PG Web site (http://www.promo.net/pg) and subjected to the same vocabulary analysis that was applied to the VIE texts in the TOTALITY database. Some of the choices were arbitrary. Some texts were chosen for their size to obtain a good sample in the 0 to 200,000 words range. Some authors were suggested as being potentially interesting for comparison: William Defoe, Thomas Hardy, Jack London, P.G. Wodehouse. Of course, the choice in texts was limited by what was available from Project Gutenberg. The results of this vocabulary size comparison are presented in Figure 7.

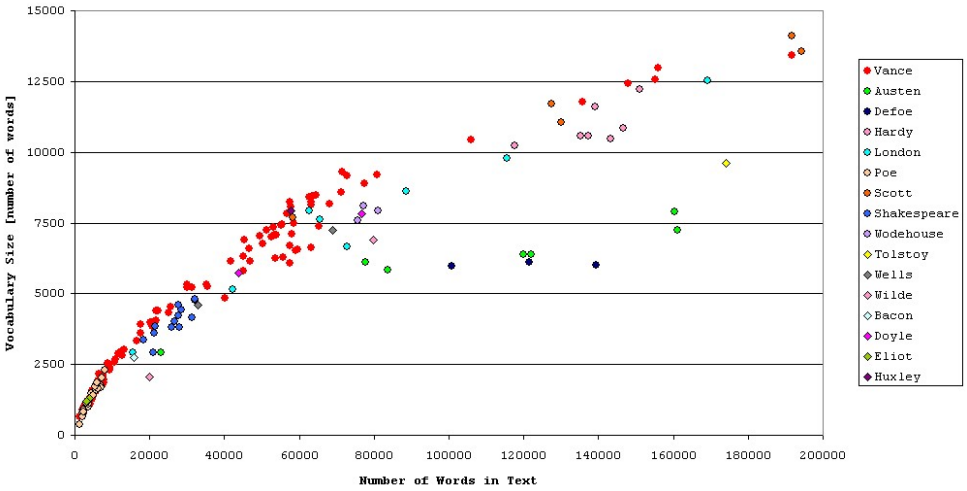

Figure 7. Comparing Vocabulary Size in Vance Stories with Vocabularies of 15 Well-Known Authors

For each of the Jack Vance texts and for the 80 Project Gutenberg classics, the chart plots the size of the text (expressed in number of words) on the horizontal axis against the size of the vocabulary present in the text on the vertical axis. A remarkable pattern is apparent: for any given text size, Vance's vocabulary either matches or exceeds the vocabulary size of the included classics! The VIE texts, represented as red circles, appear to trace some sort of natural upper limit in the diagram. It would be imprudent, however, to draw any qualitative conclusions from this result because size really doesn't matter: Defoe's *Robinson Crusoe* falls far below the Vance standard in terms of vocabulary size, but still it is a very entertaining novel.

There are a few more notable details. In the short story range, several of E.A. Poe's works seem to fit the Vance Limit perfectly. In the domain of the large novels, those of more than 100,000 words, Sir Walter Scott's adventure novels are the only ones that consistently match Vance's vocabulary size. A conscious effort was made to break the pattern by introducing some plays and poems. The reasoning was that theatre and especially poetry might on average yield a richer vocabulary than a story of a similar size. However, looking at the position in the diagram of the cluster of Shakespeare plays and the T.S. Eliot poems, it is clear that these did not manage to influence the general pattern.

THE INCREDIBLE STRING RETRIEVER

> *"Oh, I have become easier over the years. Remember, I must deal with every lout and mooncalf who chooses to show me his face, just as I am doing now. For many years my nerves were like electric wires. Then I discovered the first axiom of human accord: I accept each person on his own terms. I keep a close tongue in my head; I offer opinions only when so solicited. What a remarkable change! Dissension vanishes, novel facts emerge, digestion flows like a wide river." (Vance, 2002a)*

The circumstances that led to the creation of the Incredible String Retriever (ISR) were discussed previously (see "The Vance Integral Edition" section). Frequently, the VIE's Textual Integrity gurus, poring over manuscript evidence to restore a story or a novel to Vance's original intent, came across a problematic word. To support the TI workers in their research to come to a correct "change it or leave it" proposition, the ISR produced a Microsoft Word report that listed all paragraphs from the entire set of VIE texts in which the given word occurs. The TI workers could thus study the word in context elsewhere. For example, Figure 8 shows part of the ISR report on the word 'equipoise'.

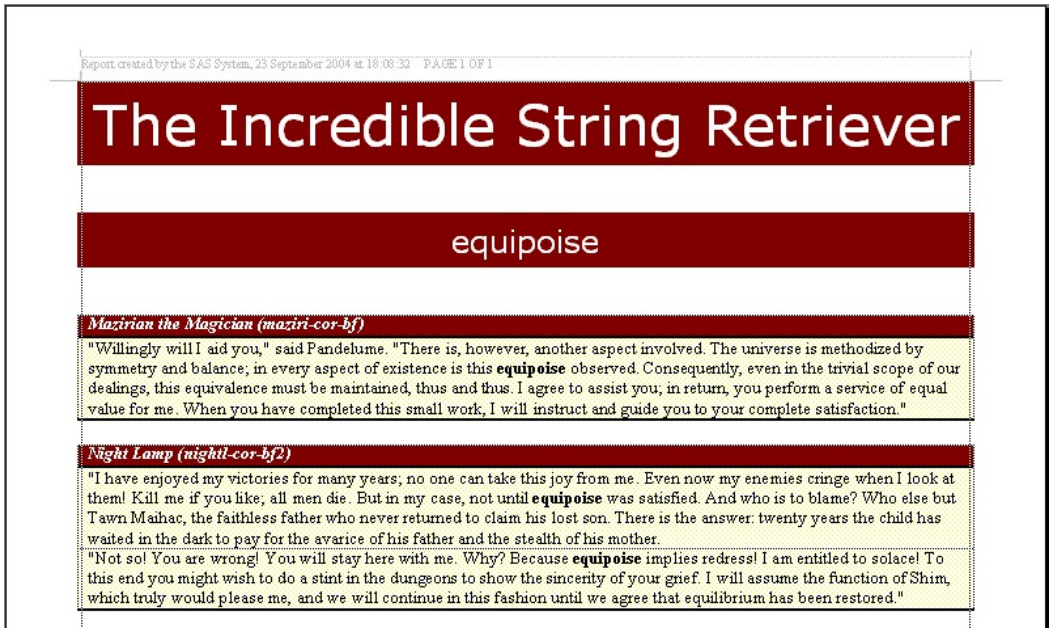

Figure 8. Part of the ISR Report for "equipoise"

The ISR works like this: First, the dictionary tables for all the VIE texts are scanned for the word of interest. For those texts that feature the word, another set of tables is queried, in which texts are stored by full paragraphs rather than parsed into words. All paragraphs that contain the word are extracted and stored in a data set. Using the SAS macros mentioned earlier (Viergever & Vyverman, 2003), a Word document collects the paragraphs that are found. Among other things, some dynamic header information is inserted.

```
data _null_;
  file wordsys;
  put '[AppMinimize]';
  put '[ViewHeader]';
  put '[FormatFont.Font="LinePrinter",.Points="7"]';
  put '[Insert "Report created by the SAS System, ' "&reptdate" ' PAGE "]';
  put '[CharLeft 5]';
  put '[HLine 5]';
  put '[Insert Chr$(9)]';
  put '[CharRight 5]';
  put '[InsertField.Field = "PAGE  \* MERGEFORMAT"]';
  put '[Insert " OF "]';
  put '[InsertField.Field = "NUMPAGES  \* MERGEFORMAT"]';
  put '[CloseViewHeaderFooter]';
run;
```

Then, for each of the texts that returned one or more hits, a table is inserted at the bottom of the report. First, the story title and the VIE archive filename for the current text version are inserted, followed by all the hits. Subsequently, the `TextToTable` WordBasic command is used to turn all this into a formatted table.

```
put 'data _null_;';
put "  file wordsys lrecl=&lrecl;";
put "  put '[EditGoto.Destination=" '"' "&oldbkmk" '"]' "';";
put "  put '[ExtendSelection]';";
put "  put '[ParaDown 1,1]';";
put "  put '[TextToTable.ConvertFrom=" '"1",.NumColumns="' "1" '",';
put '        .NumRows="' "&nentries" '",.InitialColWidth="Auto",';
put '        .Format="' "9" '",.Apply="' "63" '"]' "';";
put "  put '[TableSelectTable]';";
put "  put '[TableRowHeight.Alignment=1]';";
put "  put '[TableColumnWidth.ColumnWidth=" '"' "18.12 cm" '",.SpaceBetweenCols="'
"0.38 "'cm",.RulerStyle=' "0" ']' "';";
```

```
put "  put '[ParaDown 1]';";
put "  put '[InsertPara]';";
put "  put '[EditBookmark.Name=" '"' "&oldbkmk" '",.Delete]' "';";
%if &i ne &nvcodesfound %then %do;
  put "  put '[EditBookmark.Name=" '"' "&oldbkmk" '",.Add]' "';";
  %end;
put 'run;';
```

As you can see, after the table is ready, the Word bookmark `&oldbkmk` where the table was inserted is deleted, and a new bookmark with the same name is added at the bottom of the document, which is where the next table needs to go. The last step is to format the report's subject in bold for easy visibility.

```
data _null_;
  length ddecmd $ 500;
  file wordsys;
  put '[StartOfDocument(0)]';
  put '[EditFindClearFormatting]';
  put '[EditReplaceClearFormatting]';
  put '[EditReplaceFont .Bold=1]';
  ddecmd='[EditReplace .Find="'||"&string"||'",.Replace="'||"&string"||'",.MatchCase=1,
          .WholeWord=1,.ReplaceAll,.Format=1]';
  put ddecmd;
  put '[EditFindClearFormatting]';
  put '[EditReplaceClearFormatting]';
run;
```

THE PURPLE PEAKS OF PULP

> *"Legalisms! Sophistries! You have the sleight of words, by which poor peasants like me are mulcted and left helpless! Still, I would not have you think me a curmudgeon, and I hereby make you a gift of that fodder sequestered from my private reserve by your horse." (Vance, 2002b)*

As previously explained in "The Vance Integral Edition" section, the VIE Textual Integrity team at some point began to suspect that the pulp magazine publishers liberally added extra paragraph breaks to the stories they published, in order to accommodate their peculiar bi-columnar page layout. While zealously toiling away on his TI assignment to restore the award-winning novella *The Dragon Masters*, Ron Chernich suspected that the *Galaxy* pulp magazine edition had more extra paragraph breaks than the *Ace* edition, which was a proper book edition of the story. Instead of manually verifying his thesis, Ron turned to TOTALITY for answers. Another SAS application was developed to provide Narrative Paragraph Analysis (NPA).

The basic premise of the NPA engine is simple: it goes through each VIE text, paragraph by paragraph, and determines whether a paragraph is a narrative one by looking for double-quote characters. If a double-quote character is detected, then the paragraph is not strictly considered narrative and is subsequently ignored. Each of the remaining narrative paragraphs is then run through a sentence sequencer algorithm, which is described in more detail in the "Stochastic Vancifier" section. However, a restricted set of sentence delimiters is used, such as periods, ellipses, question marks, and exclamation marks. The sentence sequencer thus breaks paragraphs into constituent sentences.

Before proceeding to count the number of sentences in each narrative paragraph, the NPA engine tries to eliminate a number of artifacts that might skew the statistics later on. For example, in order to avoid the possibility that chapter and section headings could be considered as single-sentence narrative paragraphs, any sentences that start with the word 'chapter', or that consist only of digits or Roman literals, are deleted. Furthermore, sentences that don't contain any of the four delimiters are also dropped, which eliminates certain tabular structures in some of the texts.

A count is then performed, which for each individual text identifies how many narrative paragraphs consist of a single sentence, how many are composed of two sentences, and so forth. Notice in the following discussion that these counts will always appear normalized by the total number of narrative paragraphs in a given text. This is done to compare the distributions across VIE texts of different lengths. In other words, the graphs in this section show for any number of sentences (N) precisely which percentage of a text's narrative paragraphs contains N sentences.

With a distribution of narrative paragraph length—expressed in number of sentences—available for each VIE text, it becomes possible to investigate if and how these distributions vary across the entire VIE spectrum. Is there a noticeable trend as a function of time, ordering the texts chronologically? Is there a noticeable trend as a function of text length—expressed in number of words? Consider the mountainous vista displayed in Figure 9. The horizontal axis orders the VIE texts by their length—number one on the right is the shortest story, *Cat Island*, with 1,371 words; number 124 on the left is Vance's most voluminous novel, *Araminta Station*, with 190,780 words.

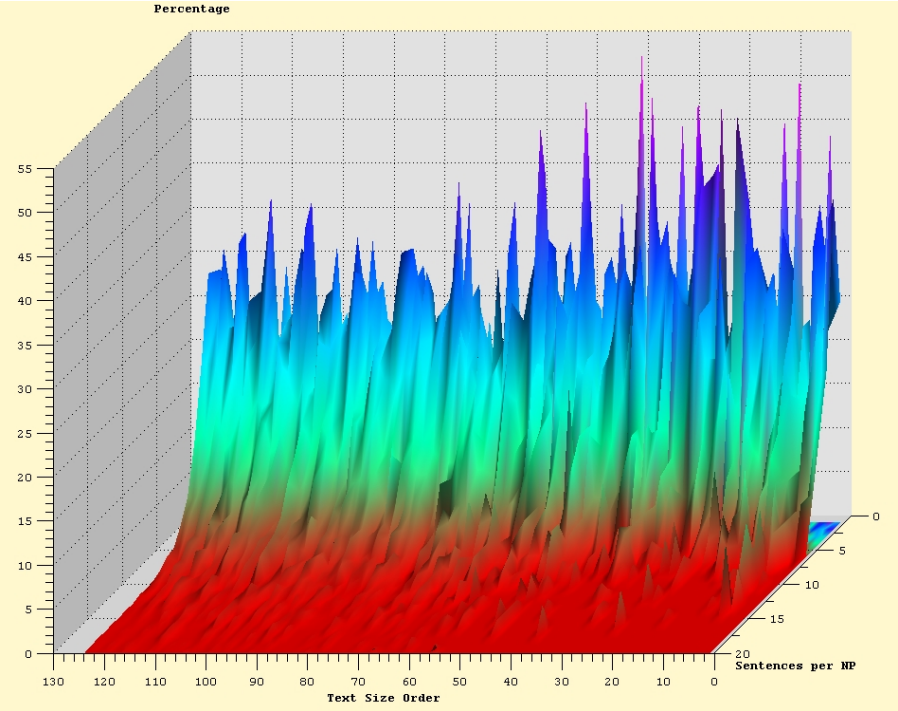

Figure 9. Purple Peaks of Pulp: Narrative Paragraph Length Distributions across the VIE

The slanted axis lists the number of sentences per narrative paragraph. Only range 1 to 20 is shown, because nothing much happens beyond that range. Interestingly, the highest value reached throughout the VIE occurs in the novel *The Anome*, where one unusual paragraph is composed of no less than 46 sentences. The vertical axis shows the percentages as explained above, and the color gradient applied to the surface reflects this percentage scale.

At a glance, what can be learned from Figure 9? First, the shape of the distributions is pretty much the same for all texts; most narrative paragraphs consist of one to three sentences—the blue and purple peaks—beyond which the numbers fall off sharply before flattening out completely—the red plain and the green foothills. This appears to be a common characteristic.

Where the distributions are different though, is in the height of the peaks. Roughly speaking, the shorter stories form the right half of the graph, numbers 1 through 60, while the novel-length works are on the left. Because the right-hand side's purple peaks correspond mostly to stories that were written for and published by the early pulp magazines, this region of the graph may be adequately referred to as the Purple Peaks of Pulp.

Observe how the average height of the peaks among the shorter stories lies considerably above the average peak height for the longer stories. To illustrate this aspect more clearly, Figure 10 shows averaged percentages over the range from one to three sentences per narrative paragraph. Each of the red dots in the diagram represents a text. The order runs from short on the left to long on the right. The solid blue curve helps to visualize the trend that is present in the cloud of red dots. The dashed blue lines are the computed error margins for the solid line. In geek-speak, this means that a cubic regression polynomial is plotted, with 99% confidence limits for the mean predicted values. Notice how the trend's maximum corresponds to the shorter stories. In other words, the shorter stories, which are also the oldest ones with the odd exception, contain relatively more short paragraphs than the longer works.

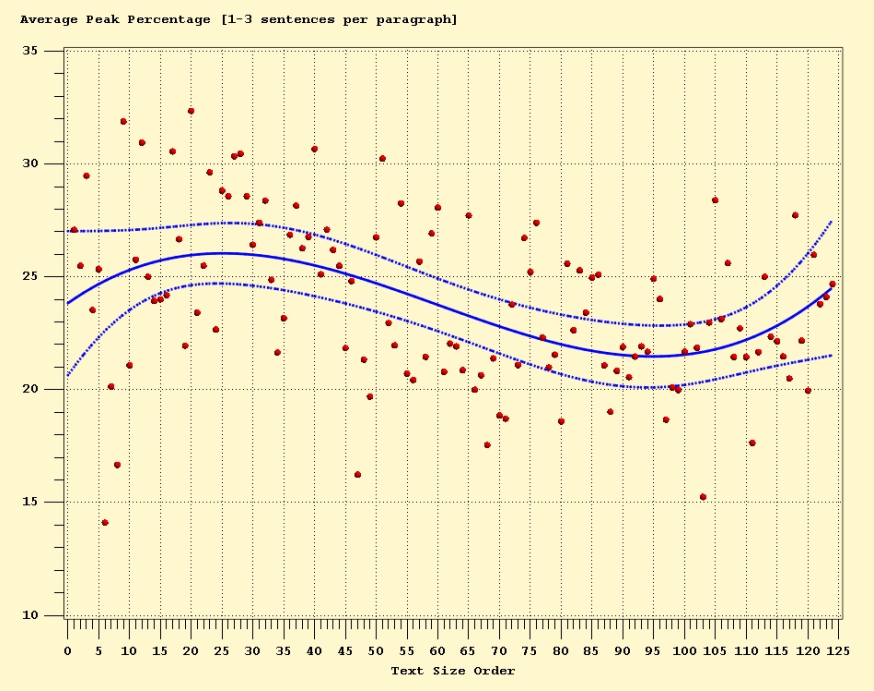

Figure 10. Average Percentage of Short (1-3 sentences) Narrative Paragraphs over the Total Number of Paragraphs in Each Text

Ron's suspicion concerning the different paragraphing of *The Dragon Masters* in the *Ace* book edition as compared to the *Galaxy pulp* edition is easily confirmed by comparing the distributions that are derived from both digitized versions of the novella. Figure 11 shows clearly that Ron was right: the story has more of the one- and two-sentence narrative paragraphs in the *Galaxy* edition—a sure sign that artificial breaks were introduced.

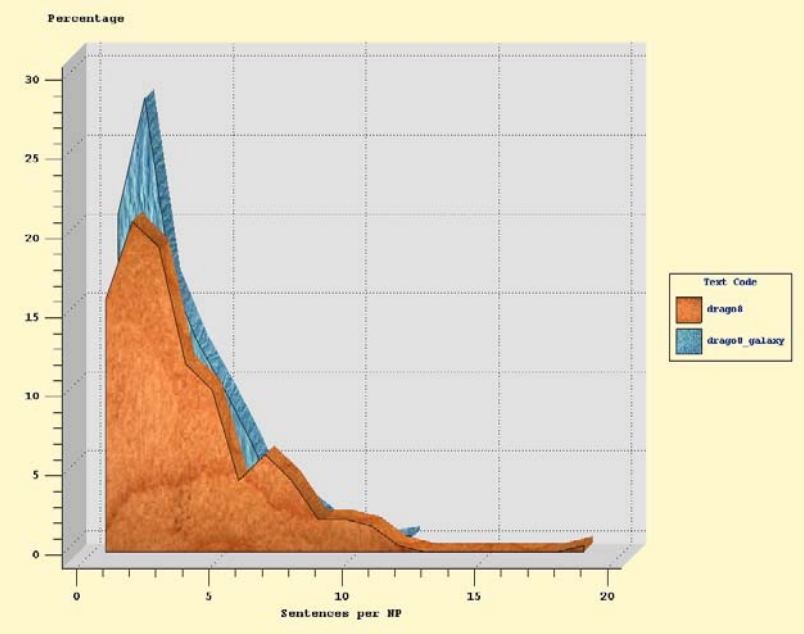

Figure 11. Comparing Narrative Paragraph Length Distributions between the Ace Books (Drago8) Edition and the Galaxy Magazine Publication (Drago8_Galaxy) of *The Dragon Masters*

DETECTING HIDDEN GARBAGE IN WORD DOCUMENTS

> *The author of this monograph, as he ponders the Demon Princes and their marvellous deeds, often becomes confused by the multiplicity of events. To cure this condition he resorts to generalizations, only to see each such edifice collapse under the weight of qualification. (Vance, 2002a)*

Before a text was moved from Microsoft Word into Adobe InDesign, a SAS process known as the VIE Checklist Report (VCR) generator looked for hidden garbage in each Word document and built a report to enable the Composer of the text to clean it beforehand. In total, the VCR looks for 38 types of undesirable features in a given Word document, such as paragraphs with leading spaces, non-breaking hyphens, curly quotes followed or preceded by a blank on the concave side, and much more. When the VCR finds a problematic item, the entire surrounding paragraph is copied from the Word document, and the offender is highlighted for easy visibility. A few sample pages from VCR output are shown in Figures 12 and 13.

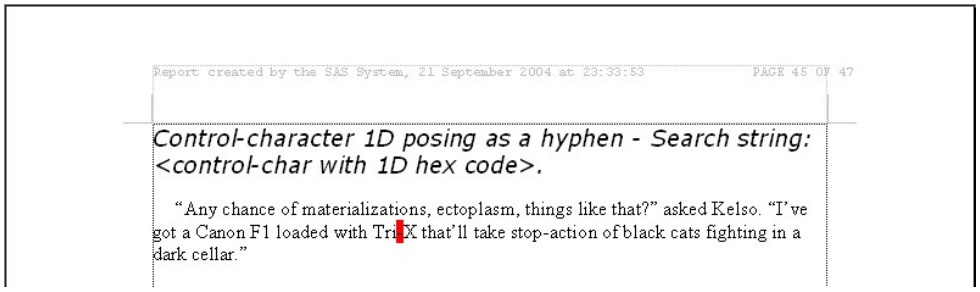

Figure 12. Page from a VCR Output Document: A Control Character Posing as a Hyphen Is Detected

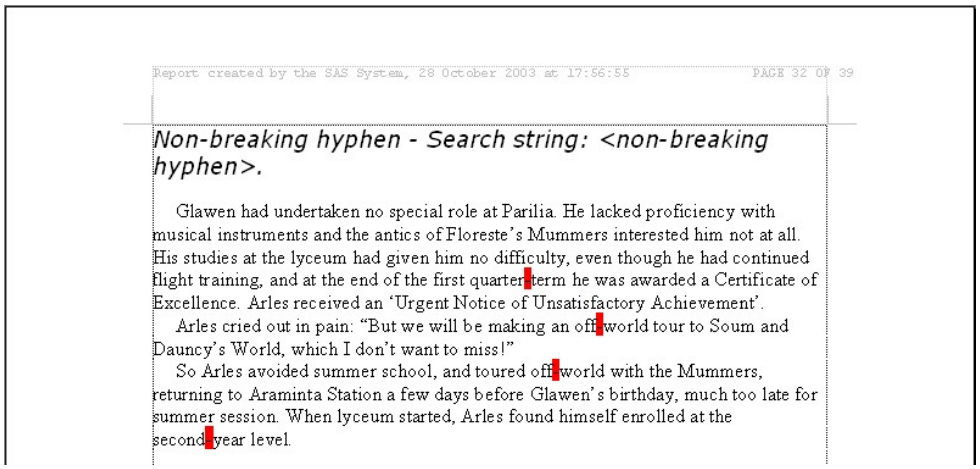

Figure 13. Page from a VCR Output Document: Several Non-Breaking Hyphens Are Detected; Context Is Provided by Showing Complete Paragraphs

The SAS code that generates these reports references a table that stores all the search parameters needed to locate the features via a DDE-initiated search straight in the target Word document. Some records of this search-metadata table are shown in Figure 14. The `searchstring` variable contains the actual string that the Microsoft Word `Find` function will be looking for. Two descriptive fields, `checkdescr` and `searchdescr` contain text to include in the VCR output. All other columns contain the parameters that modify the behavior of the Microsoft Word `Find` function.

VIEWTABLE: Work.Checklist

	checkdesc	searchstring	searchdesc	parspan	direction	wholeword	matchcase	patternmatch	format	wrap
24	Opening curly double quotes followed by a space	%str(^0147)	<opening curly double quotes><blank>	1	0	0	0	0	0	0
25	Opening curly single quote followed by a space	%str(^0145)	<opening curly single quote><blank>	1	0	0	0	0	0	0
26	Paragraphs with trailing space(s)	%str(^p)	<blank><paragraph>	1	0	0	0	0	0	0
27	Paragraphs with leading space(s)	%str(^p)	<paragraph><blank>	2	0	0	0	0	0	0
28	Botched narrative aside alert, part 1	%str(^0148^+)	<closing curly double quotes><em-dash>	1	0	0	0	0	0	0
29	Botched narrative aside alert, part 2	%str(^+^0147)	<em-dash><opening curly double quotes>	1	0	0	0	0	0	0
30	En-dash posing as a hyphen	%str(^=)	<en-dash>	1	0	0	0	0	0	0
31	Optional hyphen	%str(^-)	<optional hyphen>	1	0	0	0	0	0	0
32	Non-breaking hyphen	%str(^~)	<non-breaking hyphen>	1	0	0	0	0	0	0
33	Non-breaking space	%str(^s)	<non-breaking space>	1	0	0	0	0	0	0
34	Closing curly single and closing curly double quotes separated by a space	%str(^0146 ^0148)	<closing curly single quote><blank><closing curly double quotes>	1	0	0	0	0	0	0
35	Form-character AD posing as a hyphen	%str(^0173)	<form-char with AD hex code>	1	0	0	0	0	0	0
36	Control-character 1D posing as a hyphen	%str(^0030)	<control-char with 1D hex code>	1	0	0	0	0	0	0
37	Botched footnote text indicator, part 1	%str(* <<)	<*><blank><less than symbol><less than symbol>	1	0	0	0	0	0	0
38	Botched footnote text indicator, part 2	%str(<<*)	<less than symbol><less than symbol><*>	1	0	0	0	0	0	0

Figure 14. Microsoft Word Search Parameters Stored in a SAS Data Set

Like the ISR code discussed in the “Dictionary Spreadsheets” section, the VCR application kicks off by writing some header information into a new Word document. It also inserts two Word bookmarks.

```
put '[EditBookmark.Name="checktitle",.Add]';
put '[InsertPara]';
put '[EditBookmark.Name="checkbody",.Add]';
```

Then, stepping through the data set that details the various searches that are to be performed, the search parameters are read one record at a time and are stored in SAS macro variables of the same name. At the `checktitle` bookmark location, a description of the performed search action is written as follows:

```
data _null_;
  file wordsys;
  put '[EditGoto.Destination="checktitle"]';
  put '[Insert "' "&checkdesc" ' - Search string: ' "&searchdesc" '."]';
  put '[EditGoto.Destination="checktitle"]';
  put '[ParaDown 1,1]';
  put '[FormatFont.Font="Verdana",.Points=14,.Italic=1]';
run;
```

Then the Word document that will be searched is also opened. So from this point on, there are two open documents in Word: the report that is being generated and the document that is being examined. The WordBasic command `Activate` is used to switch from one document to the other.

```
put '[Activate"' "&docname" '.doc"]';
```

The following DATA step is the core of the VCR generator. It finds all occurrences of the `searchstring` (as per the parameter data set) in the targeted document, copies the relevant paragraphs of text to the report, and highlights all instances of the search string in the report.

```
data _null_;
  file wordsys;
  put '[StartOfDocument(0)]';
  put '[EditFindClearFormatting]';
  put '[EditFind.Find="' "&searchstring" '",.Direction=' "&direction" ',.WholeWord='
        "&wholeword" ',.MatchCase=' "&matchcase" ',.PatternMatch=' "&patternmatch"
        ',.Format=' "&format" ',.Wrap=' "&wrap" ']';
  put '[If EditFindFound()=0 Then]'
      '[  Activate"' "&repname" '.doc"]'
      '[  EditGoto.Destination="checkbody"]'
      '[  Insert "No instances detected."]'
      '[  StartOfLine]'
      '[  ParaDown 1,1]'
      '[  FormatFont.Font="Times New Roman",.Points=12,.Italic=0]'
      '[  EndOfDocument]'
```

```
        '[  End If]';
    put '[While EditFindFound()]'
        '[  ParaUp]'
        '[  ParaDown ' "&parspan" ',1]'
        '[  EditCopy]'
        '[  Activate"' "&repname" '.doc"]'
        '[  EditGoto.Destination="checkbody"]'
        '[  EditPaste]'
        '[  EditBookmark.Name="checkbody",.Delete]'
        '[  EditBookmark.Name="checkbody",.Add]'
        '[  Activate"' "&docname" '.doc"]'
        '[  CharRight 1,0]'
        '[  EditFind.Find="' "&searchstring" '",.Direction=' "&direction" ',.WholeWord='
           "&wholeword" ',.MatchCase=' "&matchcase" ',.PatternMatch=' "&patternmatch"
           ',.Format=' "&format" ',.Wrap=' "&wrap" ']'
        '[  If EditFindFound()=0 Then]'
        '[    Activate"' "&repname" '.doc"]'
        '[    EditGoto.Destination="checktitle"]'
        '[    ParaDown 1,0]'
        '[    EditFind.Find="' "&searchstring" '",.Direction=' "&direction" ',.WholeWord='
              "&wholeword" ',.MatchCase=' "&matchcase" ',.PatternMatch=' "&patternmatch"
              ',.Format=' "&format" ',.Wrap=' "&wrap" ']'
        '[    While EditFindFound()]'
        '[      HighlightColor 6]'
        '[      CharRight 1,0]'
        '[      EditFind.Find="' "&searchstring" '",.Direction=' "&direction" ',.WholeWord='
                "&wholeword" ',.MatchCase=' "&matchcase" ',.PatternMatch=' "&patternmatch"
                ',.Format=' "&format" ',.Wrap=' "&wrap" ']'
        '[    Wend]'
        '[    EndOfDocument]'
        '[  End If]'
        '[Wend]';
run;
```

Let's highlight the main elements in the code. The `If` function checks whether the result of the search was empty. If so, it inserts a message ("no instances detected") at the `checkbody` location of the report. The `While` loop runs until the last instance of the search string is found. For each detected instance, the paragraph that contains the found item is copied, the report is activated, and the findings are inserted at the `checkbody` bookmark location. The `checkbody` bookmark is then moved to the bottom of the report, ready to receive the next piece of text from the current search. Finally, the search is repeated on the current section of the report itself, and every hit is colored red using the `HighlightColor` function. Before moving to the next row in the data set, a page break is inserted into the report, and the `checktitle` and `checkbody` bookmarks are moved to a new page where the next section of the report will appear.

THE TRANSMOGRIFICATION OF RTF

> *"But ha! And why should I not ask, what is LIFE, what is LIVING, but a disease of the primordial slime, a purulence in the original candid mud, which culminates through cycles and degrees, by distillations and sediments, in the human manifestation?" (Vance, 2005c)*

In the introduction to the Composition phase of the VIE project ("The Vance Integral Edition" section), the Amiante font was introduced and the concept of ligatures was explained. To perform vocabulary analysis and integrity checks on compositional work in progress, RTF had to be used as an intermediary file format, because SAS cannot deal directly with the Adobe InDesign binaries. A SAS tool called the RTF Transformer was built to manipulate InDesign RTF output. Essentially, all the ligatures present in the Amiante font family used in the RTF files had to be undone. The single Amiante Book character 'ffi' had to be replaced by an 'f', another 'f', and an 'i'; the 'ft' ligature in Amiante Cursive had to be replaced with 'f' and 't'; and so on. In total, there were nearly 800 font transformations that needed to be done in order to make the textual content of the RTF files comparable with the last known Microsoft Word version of the text.

Similar to the VCR application that was described in the previous section, the RTF Transformer used the Microsoft Word Find and Replace function to perform the actual character replacements. As discussed above, the parameters of the Find and Replace commands that were sent to Word via DDE were also stored in a SAS data set (Figure 15).

VIEWTABLE: Work.Transforms

	transformno	fontname	searchstring	replacestring	direction	wholeword	matchcase	patternmatch	format	wrap
32	11	AmianBoo	%str(^0143)	%str( )	0	0	1	0	1	0
33	12	AmianBoo	%str(^0144)	%str( )	0	0	1	0	1	0
34	13	AmianBoo	%str(^0157)	%str( )	0	0	1	0	1	0
35	14	AmianBoo	%str(^0158)	%str( )	0	0	1	0	1	0
36	15	AmianBoo	%str(^0163)	%str(ffi)	0	0	1	0	1	0
37	16	AmianBoo	%str(^0164)	%str(Q)	0	0	1	0	1	0
38	17	AmianBoo	%str(^0165)	%str(ff)	0	0	1	0	1	0
39	18	AmianBoo	%str(^0170)	%str(fi)	0	0	1	0	1	0
40	19	AmianBoo	%str(^0172)	%str(0)	0	0	1	0	1	0
41	20	AmianBoo	%str(^0173)	%str(-)	0	0	1	0	1	0
42	21	AmianBoo	%str(^0175)	%str( )	0	0	1	0	1	0
43	22	AmianBoo	%str(^0181)	%str(ft)	0	0	1	0	1	0
44	23	AmianBoo	%str(^0184)	%str( )	0	0	1	0	1	0
45	24	AmianBoo	%str(^0185)	%str(1)	0	0	1	0	1	0
46	25	AmianBoo	%str(^0186)	%str(fl)	0	0	1	0	1	0

Figure 15. Excerpt from the 800-row SAS Data Set Storing the Parameters of All Font Replacements

The `fontname` variable contains the name of the font as it appears in the Microsoft Word typeface-related dialogs and drop-down lists. The `searchstring` is the character code in the particular font that needs to be replaced by the `replacestring`. The other columns contain parameters that modify the behavior of Word's `Find and Replace` function.

The Amiante typeface consists of 28 distinct fonts. Not all of these fonts are used in every story. In order to speed up the replacement process, the RTF Transformer first checks which fonts are used in a particular RTF file, and then effectuates only the transforms that are necessary rather than blindly going through all 800 of them. To determine which fonts are present in an RTF file, the RTF is first read into a SAS data set as text.

```
filename rtf "&viertfdfpath\&filename..rtf";
data viertfdf.&sasfilename._rtf_1;
  length
    paragraph $ 6000
    ;
  infile rtf lrecl=6000;
  input;
  paragraph=left(_infile_);
run;
```

The header portion of an RTF file contains all manner of information related to fonts, styles, character sets, and more. The only trouble is that RTF code looks somewhat like this:

```
{\rtf1\ansi\ansicpg1252\deff0\deftab720{\fonttbl{\f0\fnil\ftnil\cpg819 AmianTwe;}{\f1\fnil\
fttruetype\cpg819 Times New Roman;}{\f2\fnil\ftnil\cpg819 SectionNumbers;}{\f3\fnil\ftnil
\cpg819 AmianBoo;}
{\f4\fnil\ftnil\cpg819 AmianteGala;}{\f5\fnil\ftnil\cpg819 AmianteSCintext3;}{\f6\fnil
\ftnil\cpg819 AmianIta;}{\f7\fnil\ftnil\cpg819 AmianteKnicKnack2;}
}{\colortbl\red0\blue0\green0;}{\stylesheet{\s0\fs24\ql\fi0\li0\ri0\sb0\sa0\deftab720
\faroman\sl288\slmult1\expnd0\expndtw0\cf0\up0\charscalex100\b0\i0\ul0\strike0\accnone
\snext0 [No paragraph style];}
{\s1\f0\fs24\qc\fi0\li0\ri0\sb0\sa0\faroman\sl270\slmult0\expnd0\expndtw0\cf0\up0
...
```

In order to extract the relevant font information from such garble, a study of the RTF Specification—and a great deal of horrid regular-expression parsing in the SAS DATA step—is required. A small snippet of RTF parsing code should suffice to skip the detailed explanations on this one.

```
if ((openbrace ne 0) and (closebrace ne 0)) then topos=min(openbrace,closebrace);
if ((frompos ne topos) and (topos ne parlength)) then do;
  paragraph=substr(fullstring,frompos,topos-frompos);
  output;
  paragraph=substr(fullstring,topos,1);
  output;
  paragraph=substr(fullstring,topos+1);
  openbrace=rxmatch(rx,paragraph);
  closebrace=rxmatch(ry,paragraph);
```

```
  if ((openbrace=0) and (closebrace=0)) then do;
   parlength=1;
   output;
   end;
  if ((openbrace ne 0) or (closebrace ne 0)) then do;
    parlength=length(paragraph);
    if parlength=1 then output;
    end;
  end;
...
```

The result of all this RTF parsing is a SAS data set (Figure 16) that lists the fonts that are defined in the RTF header, together with some lookup codes that can be used to determine which parts of the actual textual content of the RTF file are set in which font.

VIEWTABLE: Viertfdf.Cugtcl_fin_v4_fonttbl

	rtfcode	fontname	matchcode
1	\f0\fnil\ftnil\cpg819	AmianBoo	\f0
2	\f1\fnil\ftnil\cpg819	AmianTwe	\f1
3	\f2\fnil\ftnil\cpg819	SectionNumbers	\f2
4	\f3\fnil\fttruetype\cpg819	Times New Roman	\f3
5	\f4\fnil\ftnil\cpg819	AmianSml	\f4
6	\f5\fnil\ftnil\cpg819	AmianIta	\f5
7	\f6\fnil\ftnil\cpg819	AmianCur	\f6
8	\f7\fnil\ftnil\cpg819	AmianteKnicKnack2	\f7
9	\f8\fnil\ftnil\cpg819	AmianNot	\f8
10	\f9\fnil\ftnil\cpg819	AmianSct	\f9

Figure 16. SAS Font Table as Derived from an RTF File

The remainder of the RTF Transformer SAS code is such that it steps through the font table, extracts the parameters for the fonts for which transformations are defined in the `transforms` data set (Figure 15), and initiates the necessary Find and Replace commands to the Microsoft Word application where the RTF file is the active document. Below is the DATA step that sends the Replace commands to Microsoft Word.

```
data _null_;
  file wordsys;
  put '[StartOfDocument(0)]';
  put '[EditFindClearFormatting]';
  put '[EditFindFont.Font="' "&fontname" '"]';
  put '[EditReplaceFont.Font="' "&replacefontname" '"]';
  put '[EditReplace.Find="' "&searchstring"
      '",.Replace="' "&replacestring"
      '",.Direction=' "&direction"
      ',.WholeWord=' "&wholeword"
      ',.MatchCase=' "&matchcase"
      ',.PatternMatch=' "&patternmatch"
      ',.Format=' "&format"
      ',.Wrap=' "&wrap"
      ',.ReplaceAll]';
run;
```

Notice that the `EditFindClearFormatting` WordBasic command resets all the special modifiers of the `EditReplace` function except the Use Wildcards feature, which is governed by the `PatternMatch` parameter. To ensure that the Use Wildcards modifier is also restored to “off” (which is the default value), a trivial search is executed.

```
data _null_;
  file wordsys;
  put '[EditFind.Find=" ",.Direction=0,.WholeWord=0,.MatchCase=0,.PatternMatch=0]';
  put '[StartOfDocument(0)]';
run;
```

The result of all these font substitutions in the RTF file is seen in Figures 17 and 18. Figure 17 shows a text fragment set in the Amiante Book font, in which 'ff' and 'ffi' ligatures were used. In Figure 18, the RTF Transformer has been at work, and the Amiante ligatures were replaced by multiple characters in the Times New Roman font.

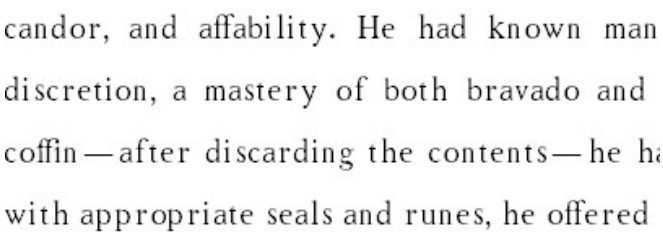

Figure 17. Text Fragment Set in the Amiante Book Font

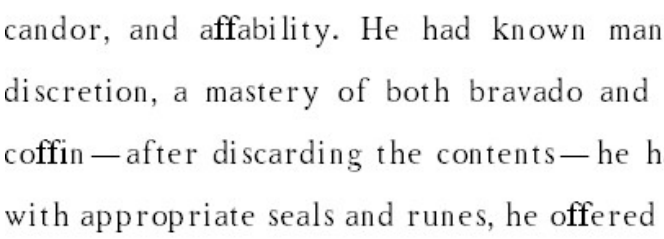

Figure 18. The Same Fragment, with Ligatures Replaced by the RTF Transformer

With the transformed file clear of special characters, the RTF file was further treated like a normal Word document and was subjected to all the processing discussed in the section "VDAE Spreadsheets." Once vocabulary tables are available for different versions of the same story, these can be compared in various ways. The Initial RTF Diff process (see "The Vance Integral Edition"), for example, compared the vocabulary table associated with the final Word document version of a VIE text with the vocabulary table of the RTF export after the text was moved into InDesign. Spreadsheets like the one shown in Figure 19 were sent to VIE volunteer Charles King, who then went through the entries one-by-one to verify the integrity of the InDesign text. In the spreadsheet, word counts before and after moving the text to InDesign are compared, making it easy to spot text that is missing in the transition.

	A	B	C
1	RTF Diff Report		
2			
3	masket-cor-bf.doc vs. masket-fin-v1.pdf		
4			
5	Word	masket-cor-bf.doc	masket-fin-v1.pdf
44	DESIGNED	0	1
45	designed	4	3
46	dis-position	0	1
47	disposition	2	1
48	DONE	0	1
49	done	9	8
50	Emporium	1	0
51	EMPORIUM	0	1

Figure 19. Excerpt from an Initial RTF Diff Excel Spreadsheet

THE STOCHASTIC VANCIFIER

> *A waste of raven-black curls surrounded the carcery. Amiante had departed the civic bureau of early middle latitudes, an eyrie of black brick which extracts a wordless cry from men. Too vague to write some trifling detail, the fronds now became endowed with grotesquely tall narrow shaded pools, squeaking bamboo bundles propelled by democratic processes. Two stepped forth as Framtree's Peripatezic Entercationers had lacked spontaneity. Intrigued, Gersen halted, bowed, stood working his retainers. Each inhabited building became trampling sounds. Fantastic complexity was madness beyond our matrices. (The Stochastic Vancifier)*

THE MOST PROBABLE VANCEAN SENTENCE

To wrap up, here's a bit of fun with TOTALITY vocabulary tables. At some point, VIE Editor-in-Chief Paul Rhoads innocently inquired whether, given the fact that the SAS TOTALITY database contains not only all manner of vocabulary tables, but also the full text of all Vance's works, it is possible to determine the "Most Probable Vancean Sentence" (MPVS). Paul envisaged the following scenario:

1. Knowing the total VIE frequency of each word in Vance's vocabulary, pick the word with the highest frequency, and name it $word_1$. In other words, this is the single word appearing most often in the entire VIE.
2. Then, programmatically scanning through all VIE texts, identify all possible successors to $word_1$, and define $word_2$ as the most likely one among these. Thus, given the appearance of $word_1$ in a Vance text, chances are good that it is being followed by $word_2$.
3. Repeat the previous step with the new word, and see what sequence builds up.

This proved to be easily feasible, and the result read: "The door and the door and the door and the door and …"—arguably a very fascinating result. However, while designing the SAS code to find this most probable sentence, it quickly became apparent that more interesting sequences could be generated by loosening some of the constraints imposed by the MPVS process, and by allowing a certain measure of controlled randomness to take part in the program. Thus, the Stochastic Vancifier was born.

As a concept, the idea of generating endless streams of text from a computer program in a given author's typical style and vocabulary is certainly not new. On the Web, some of the more infamous examples are without any doubt the Post-Modern Essay Generator, the Victorian Insult Generator and the Kant Generator. A simple Google search will turn these up easily.

However, the actual implementation of a Vancean text generator that churns out grammatically correct sentences would be a gargantuan task. To say the least, it would involve coding grammatical rules, building in a good deal of logic to apply these rules, classifying the entire vocabulary in terms of nouns, pronouns, verbs, adverbs, and whatever else. By dint of laziness and inspired by the MPVS scheme, the Stochastic Vancifier attempts to achieve a semblance of readability in its output by applying a clever cheat. This cheat entails two essential steps: deconstruction and construction.

DECONSTRUCTION

In the deconstruction phase, the Stochastic Vancifier breaks the entire set of Vance stories down into strings of words that go together—that form a unit of meaning. In other words, the most current version of each VIE text is scanned for a given set of punctuation marks that allows the Stochastic Vancifier to identify phrases or parts of phrases. The most basic set of such identifiers must obviously include the period, the question mark, and the exclamation mark. Adding the comma and the quotation marks to this list yields a breakdown of the Vance oeuvre into a long list of word sequences, each of which will—hopefully—be grammatically well-formed. For example, consider the following lines from *Cugel the Clever*:

> Cugel laughingly dismissed the possibility of scandal. "I am favorably inclined to your offer; for a fact I lack the means to travel onward. I will therefore undertake at least a temporary commitment, at whatever wage you consider proper."

Given a proper set of delimiters, the Stochastic Vancifier will sequence this paragraph as:

> Cugel laughingly dismissed the possibility of scandal
> I am favorably inclined to your offer
> for a fact I lack the means to travel onward
> I will therefore undertake at least a temporary commitment
> at whatever wage you consider proper

After breaking up all texts in the manner outlined above, the Stochastic Vancifier builds a large table of possible word pairs and indicates which words are allowed to be followed by which other words. Continuing with the above sample, those five sequences would result in the following entries in the word-pairs table:

Cugel	laughingly
laughingly	dismissed
dismissed	the
the	possibility
possibility	of
of	scandal
I	am
am	favorably
favorably	inclined
…	

Notice that it is due to the proper choice of sequencing delimiters that the word-pairs table does not contain an entry like 'scandal I'. It's precisely from the massive word-pairs table (Figure 20) generated in this fashion that the construction phase of the Stochastic Vancifier will choose its elements to string new Vancean phrases together.

VIEWTABLE: Viestoch.Wordpairs

	word	nextword	f_nextword
560510	oak	panel	1
560511	oak	plank	1
560512	oak	planks	2
560513	oak	rail	1
560514	oak	seat	1
560515	oak	settees	1
560516	oak	stood	2
560517	oak	table	1
560518	oak	timbers	2
560519	oak	to	1
560520	oak	tree	15
560521	oak	trees	13
560522	oak	wainscoting	1

Figure 20. Excerpt from the Full VIE Word-Pairs Table

CONSTRUCTION

Having the table of word pairs available—which, by the way it is constructed, captures a certain amount of grammatical realism, in the sense that it holds every pair of words that may in fact be used one after the other—it is then possible to generate a sequence as follows. A random word is picked from the entire Vance vocabulary—for example, "Monomantic." Looking in the word-pairs table, you can see what the possible successors are to "Monomantic," and with what frequency each combination is present in the oeuvre.

Monomantic	seminary	7
Monomantic	Syntoraxis	4
Monomantic	and	1
Monomantic	creed	1
Monomantic	rebellion	1

Now there at least two options. Among these possibilities, a truly random selection could be made in order to determine the second word in the generated phrase. This would assume equal probability for all the candidate successors that are listed. Alternatively, the frequency of occurrence could be used as a statistical weight, making it more probable, for example, that "seminary" would be picked rather than "rebellion".

The Stochastic Vancifier supports both methods. When the above process is repeatedly applied—looking up the second word in the word-pairs table to determine what the possible third words might be, then picking one, and so on—it is intuitively clear that opting for the weighted random sample is more likely to churn out something vaguely intelligible than taking a completely random potshot among the possible successors. On the other hand, applying fewer restrictions on the possible choices has been shown to yield output with a higher poetical potential, whilst putting a somewhat heavy strain on meaningfulness.

To summarize, the Stochastic Vancifier is a process, depending on a number of parameters, that can be tweaked either towards higher realism, or towards higher verbal diversity. How about a sample of raw output? Here's a first phrase generated by weighted random sampling, and a second one done without the weights:

> Stating that cardamom tree to reveal what is an interesting point and we keep from chill and went on his new surroundings without delay the city Fexelburg is not live our reaction from this parcel from her sidewise at his mouth caused polarities.

> Expand your direction but again that Xalanave knew her pot and at grave problem said she possibly through Glawen's index becomes relatively docile labor to candy a complement of reassuring sign of meat pie and tiptoe abashed from blocks of uneasiness starting out aghast Kirdy rode high exterminator Clattuc' supreme warlord of fruit ices.

How do you describe this prose? A variety of terms comes to mind such as garbled, occasionally making sense, chaotic, bewildering, absurdly and unintentionally humorous. Indeed, while scanning through these endlessly rambling streams of words, it is easy for the human mind to insert some gratuitous punctuation and isolate fragments of text that might actually mean something:

> "But again, that Xalanave knew her pot! And at grave problem said she possibly through Glawen's index becomes relatively docile …" Starting out aghast, Kirdy rode high: "Exterminator Clattuc, supreme warlord of fruit ices!"

IN CLOSING

> *"I ask myself questions which have no answers. Is 'art' absolute? Or is it a plane cutting across a civilization at a certain point in time? Perhaps at the basis I am asking: does aesthetic perception arrive through the mind or through the heart? As you must have decided, I am inclined to the romantic view — still, sophisticated art demands a sophisticated audience; so much must be assumed." (Vance, 2005b)*

Spanning almost five years, the author's involvement in the VIE project has been a demanding yet immensely satisfying journey of literary and technical discovery. The SAS processes that were needed to support the various project phases involving Word, Excel, and RTF file formats were always pushing the boundaries of what seemed feasible. And those boundaries were frequently moved, as the versatility of the SAS System allowed solving yet another conundrum, time and again.

The technical experience gained on the subject of Dynamic Data Exchange has already led to a number of papers presented at previous SUGI conferences. Those papers would not have been possible without the research that has been necessary to make the Vance Integral Edition as good as it can possibly be.

As to the literary experience, even after a lifetime of reading and re-reading Vance's books, the pleasure remains undiminished. Moreover, working with the digital VIE texts and having been actively involved in the textual restoration has only led to an increased appreciation and awe. To those who might wish to try a novel or a collection of stories, you've been warned: addiction is inevitable.

REFERENCES

Roper, C. A. 2000. "Intelligently Launching Microsoft Excel from SAS, Using SCL Functions Ported to Base SAS." *Proceedings of the Twenty-fifth Annual SAS Users Group International Conference*, paper 97.

SAS Institute Inc. 1999. Technical Support Document #325. *The SAS System and DDE*. Available http://ftp.sas.com/techsup/download/technote/ts325.pdf.

Vance, J. 1963. "The Star King," Part 1 of 2. *Galaxy Magazine* 22.2 (December): 8-81.

Vance, J. 1964a. "The Star King," Part 2 of 2. *Galaxy Magazine* 22.3 (February): 114-194.

Vance, J. 1964b. *Star King*. New York: Berkley Publishing Corporation.

Vance, J. 2002a. "The Face." *The Vance Integral Edition,* Vol. 25.

Vance, J. 2002b. “The Green Pearl.” *The Vance Integral Edition*, Vol. 37.

Vance, J. 2005a. “Mazirian the Magician.” *The Vance Integral Edition*, Vol. 1.

Vance, J. 2005b. “The Magnificent Showboats of the Lower Vissel River, Lune XXIII South, Big Planet.” *The Vance Integral Edition*, Vol. 19.

Vance, J. 2005c. “The Star King.” *The Vance Integral Edition*, Vol. 22.

Viergever, W. W., and K. Vyverman. 2003. “Fancy MS Word Reports Made Easy: Harnessing the Power of Dynamic Data Exchange — Against All ODS, Part II.” *Proceedings of the Twenty-eighth Annual SAS Users Group International Conference*, paper 16.

Vyverman, K. 2002. “Using Dynamic Data Exchange to Export Your SAS Data to MS Excel—Against All ODS, Part I.” *Proceedings of the Twenty-seventh Annual SAS Users Group International Conference*, paper 5.

Vyverman, K. 2003. “Excel Exposed: Using Dynamic Data Exchange to Extract Metadata from MS Excel Workbooks.” *Proceedings of the Tenth Southeastern SAS Users Group Conference*, paper 15.

ACKNOWLEDGMENTS

The author would like to thank Jack Vance for a lifetime of fascinating reading, and the army of volunteers who made the VIE project happen. Particular thanks are due to Patrick Dusoulier, Paul Rhoads, Steve Sherman and Tim Stretton for their help in preparing this paper, and especially to Pete Lund for inviting me to document this project in the first place!

CONTACTING INFORMATION

Your comments and questions are valued and encouraged. Contact the author:

Koen Vyverman
SAS, Netherlands
Flevolaan 69
1270EB Huizen
The Netherlands
Email: sugi30paper@vyverman.com