Janus: the intertextuality search engine for the electronic *Manipulus florum* project

Andrew Kane and Frank Wm. Tompa
David R. Cheriton School of Computer Science, University of Waterloo, Canada

Abstract

The Electronic *Manipulus florum* Project (<http://manipulusflorum.com/>) is digitizing a large collection of Latin quotations. We examine the design of the Janus search engine, which finds overlaps between keywords, supplied text, and *Manipulus florum* quotations in the presence of typographic, orthographic, and complex variants. Janus can be used to determine whether, and to what extent, the *Manipulus florum* was used as a resource by later writers. It can also be used to determine which previous works Thomas of Ireland used to compile the *Manipulus florum*. In addition, the Janus engine could be used to find overlaps between two arbitrary texts.

Correspondence:
Andrew Kane,
David R. Cheriton School of Computer Science,
University of Waterloo,
Waterloo, ON, Canada.
E-mail:
arkane@cs.uwaterloo.ca

Downloaded from http://llc.oxfordjournals.org/ at UB Leipzig on January 4, 2012

1 Introduction

Scholars often need to compare multiple texts to explore borrowings from pre-existing documents and dependencies on earlier or contemporary authors. Given a fixed *baseline text*, they wish to check one *query text* after another to determine whether there are any common passages between the baseline text and each query text. One text that is often the source for borrowings is the *Manipulus florum*, which comprises quoted passages from earlier texts and was created to be used by others in preparing sermons, papers, etc. (Rouse and Rouse, 1979). With the *Manipulus florum* as baseline text, the Janus search engine allows scholars to explore whether specific candidate documents were used by Thomas of Ireland as sources to create the *Manipulus florum* and whether certain documents (and thus their authors) used passages from the *Manipulus florum*. We call this exploration mechanism between a baseline text and a query text an *intertextuality* search.

Like other search engines, the Janus search engine can use keywords to search *Manipulus florum* quotations. What is new, however, is that instead of supplying one or more keywords as input for a search, a user can supply any text as input, whereupon Janus will find excerpts that are common to that query text and any of the *Manipulus florum* quotations. The query text may be extracted from any published or unpublished document available to the user. After finding the shared excerpts, Janus displays the overlaps in order that the scholar can determine if the matches are significant.

To find all matches that will be of interest to scholars, the text matching algorithms used for keyword search and intertextuality search must handle:

(1) typographic variants, including formatting, punctuation, white space, and capitalization;
(2) orthographic variants, such as reading 'nichil' and 'nihil' interchangeably; and

Literary and Linguistic Computing, Vol. 26, No. 4, 2011. © The Author 2011. Published by Oxford University Press on behalf of ALLC. All rights reserved. For Permissions, please email: journals.permissions@oup.com

doi:10.1093/llc/fqr009 Advance Access published on 4 May 2011

(3) complex variants, such as interpolation, omission, or substitution of one or more words; transposition of word order; and paraphrasing.

Furthermore, for intertextuality search, it is important to some scholars that the query text they supply not be preserved within the search engine, because they wish to retain full control of all copies. We, therefore, add a fourth requirement for an intertextuality search engine:

(4) Copyright assurance for query texts.

Existing technologies that could be used to address these requirements include traditional search engines, plagiarism detection software, duplicate phrase detection software, and software for biological gene alignment. Each of these technologies finds matches between queries and texts, but fails to meet some of the four criteria listed above. Further details are included in the next section, and Table 1 summarizes the capabilities for these existing technologies. Janus meets all four requirements, combining text normalization with a plagiarism detection algorithm and a search engine.

From the more technical perspective, the *Manipulus florum* contains approximately 6,000 Latin quotations, which implies that a large query text might raise performance concerns. It is important that the search engine can find all intertextuality matches sufficiently quickly so that scholars can use the facilities repeatedly on query texts of any size. Again, Janus satisfies this requirement.

2 Related Work

Standard search engines, such as Google (<http://google.com/>), do not handle user-specified orthographic variants, but they do handle typographic variants. An open source search engine could be modified to handle orthographic variants, but the problems with complex variants cannot be handled cleanly. In particular, complex variants cannot be matched by search engines: searching using a phrase query (e.g. 'Welcome to our Janus technical demonstration') will fail when any part of the text varies, searching using an AND between each word pair (e.g. Welcome AND to AND our AND Janus AND technical AND demonstration) will fail when some words are omitted, and searching using an OR between each word (e.g. Welcome OR to OR our OR Janus OR technical OR demonstration) will find superfluous results containing those words. Handling complex variants can be approximated by searching for a list of sub-phrases combined using ORs, and then relying on the ranking algorithm to place the best results at the top of the result list; however, choosing which sub-phrases to include is problematic.

Plagiarism detection software, such as turnitin.com (<http://turnitin.com/>) and MOSS (<http://theory.stanford.edu/~aiken/moss/>), try to identify overlaps between an uploaded/submitted document and an existing repository of documents to find directly copied text. The algorithms can handle typographic and complex variants, but these packages are closed source and cannot be modified to handle non-standard orthographic variants. Furthermore, these packages often maintain copies of uploaded documents, which could restrict their usage by scholars who wish to retain control of their intellectual property. We have chosen to adapt the algorithms from such software in a system that can avoid their shortcomings.

Research has begun to measure and model phrase-level duplication in various data sets, such

Table 1 Capabilities of existing technologies

Criteria	Traditional search engines	Plagiarism detection software	Duplicate phrase detection software	Gene alignment software
Remove typographic variants	Yes	Yes	Yes	No
Remove orthographic variants	No	No	No	No
Find complex variants	No	Yes	Yes	Yes
Provide copyright protection assurance	No	No	No	No

Downloaded from http://llc.oxfordjournals.org/ at UB Leipzig on January 4, 2012

as the Web and digital libraries of scanned books (Fetterly *et al.*, 2005; Kolak and Schilit, 2008; Schilit and Kolak, 2008). The techniques employed typically use phrases of k words, with k in the range of 4–7. These techniques usually handle typographic variants and some complex variants, but they do not handle orthographic variants since they are word-based rather than based on arbitrary sequences of characters. Since many complex variants include partial modifications of words, such as suffix changes or modified verb endings, character-based algorithms, such as those underlying Janus, typically outperform word-based ones.

Systems for bioinformatics research, most notably BLAST (Altschul *et al.*, 1990), have been based on extensive work on finding overlapping sequences of genes. In this area, the genes are treated as long sequences of characters encoding the base pairs of the gene, and the algorithms in these tools are similar to, but more specialized than, those used in search engines and plagiarism detection software. The variants of interest to biologists are not typographic, orthographic, or complex, but instead they result from mutations at the base pair level. Therefore, an edit distance approach is used to measure the similarity of two sequences with assigned costs for insertions, deletions, and replacements to characterize the extent of mutation required to convert one gene to another. For the purpose of assessing complex variants, the edit distance approach over-penalizes subsequences being swapped, such as when the order of two words in a sentence is reversed. Thus the similarity measurement adopted by these systems is not as well suited to scholar's needs as are those in Janus.

There are several software tools, including those used for managing software evolution, which compare two sequences of text to determine their differences. These tools assume that the sequences are quite similar and hence only the differences are displayed or highlighted. This assumption does not hold for our purposes. Many of these tools have direct lineage from the Unix *diff* utility (Hunt and McIlroy, 1975; Myers, 1986), which compares two text inputs on a line-by-line basis using the longest common subsequence. Because such systems miss many possible similarities, they were extended to use a model that handles edit distance with moves (Tichy, 1984; Bourdaillet and Ganascia, 2006). These systems identify complex variants, and they could be modified to handle typographic and orthographic variants with some complication to the algorithms. However, all these algorithms assume input of two text sequences to be compared, but our system needs to work with one query text and many baseline texts. As a result, the algorithms used within the Janus search engine achieve much better performance than could be obtained using an approach based on *diff*. Interestingly, the visualization of the text alignment used in the Janus search engine is similar to that found in the MEDITE system (Bourdaillet and Ganascia, 2006), but it displays the similarities rather than the differences in the texts.

3 Core Algorithms

The implementation of the Janus search engine builds on Lucene (<http://lucene.apache.org/>), an open source search engine, and adapts the text comparison algorithm used in MOSS (<http://theory.stanford.edu/~aiken/moss/>), a program for detecting plagiarism. The resulting algorithm allows us to match texts that differ because of typographic, orthographic, and complex variations.

To compare two texts, each is viewed as an ordered sequence of characters (also called a *string*), starting from the first and continuing to the last. For a given integer value n, an *n-gram* is a *sub-string* starting at any character in the text and continuing from each character to the next until n characters are included. Any text thus corresponds to a collection of overlapping n-grams, one for each starting position.

MOSS considers a text to be a set of overlapping n-grams and selects some of these to represent the text, a process known as *fingerprinting*. The specific selection algorithm, called *winnowing* (Schleimer *et al.*, 2003), guarantees that the fingerprint contains at least one n-gram within every sequence of w n-grams, where w is called the *window size*. For example, consider the text X depicted in Fig. 1, and assume that the chosen n-gram size is 5 and the

Downloaded from http://llc.oxfordjournals.org/ at UB Leipzig on January 4, 2012

Baseline text *X*:
"Welcome to the Janus demonstration."
Normalized text:
"weltometotheianusdemonstration"
5-grams:
welto, *eltom*, *ltome*, *tomet*, *ometo*, ..., *trati*, *ratio*, *ation*
Winnowing selection of 5-grams:
eltom from {*welto*, *eltom*, *ltome*}
ltome from {*eltom*, *ltome*, *tomet*}
...
ation from {*trati*, *ratio*, *ation*}
Query text *Y*:
"Welcome to our Janus technical demonstration ...."

Fig. 1 Depiction of *n*-gram winnowing algorithm.

window size is chosen to be 3. The algorithm first normalizes the text by removing typographic markers, such as whitespace characters and capitalization, and by standardizing orthographic markers, such as replacing *j* with *i* to handle differing interpretations of Medieval Latin and *c* with *t* to handle variant spellings such as *predicacio* and *predicatio*. The algorithm then considers all overlapping 5-grams from the normalized text. Each 5-gram is converted through a hash function into an integer, which is called its *signature*. Because the window size is 3, the algorithm then chooses one of the 5-grams (the one with the lowest integer signature) from each overlapping sequence of three 5-grams, as shown in Fig. 1. The full text is thus characterized by the selected 5-grams. Now, given the query text *Y*, also shown in Fig. 1, the same characterization algorithm (i.e. normalizing, forming 5-grams, winnowing with a window of size 3) is applied. As a result, some of the same 5-grams will be selected from the overlapping sub-texts 'Welcome to', 'Janus', and 'demonstration'. The algorithm follows these three steps consistently, so if two normalized texts have overlapping character sequences of length at least $n+w$, then it is guaranteed that at least some shared *n*-grams will be selected.

The Janus search engine is first given a collection of baseline texts. For each baseline text, a document is formed consisting of the sequence of normalized words (i.e. with typographic and orthographic variants removed) followed by the set of selected *n*-grams (each treated as if it were an additional 'word' in the baseline text). When the search engine indexes all the word occurrences in these constructed documents (as it would any collection of documents), it can be used to find conventional text (for keyword and phrase search) as well as *n*-gram values (for intertextuality search) within the baseline text collection.

To create intertextuality queries, we normalize and fingerprint the query text to produce *n*-grams and then form the query:

$$n\text{-gram}_1 \text{ OR } n\text{-gram}_2 \text{ OR } \ldots$$

This query is processed against all the quotations. If matches share more or longer sub-strings with the query text, they will have more *n*-grams in common, and thus they will automatically be assigned a higher rank by the search engine.

Intertextuality search can be combined with standard text search. If a search includes keywords in addition to a query text, *n*-gram matching is also combined with (normalized) keyword matching.

After an intertextuality search, the resulting sub-string overlaps are displayed to the user, who can determine if the match is significant. To accomplish this, the code maintains the start and end location in the original text for each *n*-gram, and these are used to locate the overlapping *n*-grams for each quotation appearing in the result.

Importantly, the Janus search engine does not maintain a copy of the query text after the results are sent to the user. As a consequence the user can maintain control over all copies of the text.

Continuing the previous example, given the baseline text *X* and the query text *Y*, the overlapping *n*-grams and corresponding locations are identified. These locations allow us to map the *n*-grams back to the query text *Y* and to highlight these locations, as shown in Step 1 at the bottom of Fig. 2. The highlighted text easily compensates for the presence of overlapping shared *n*-grams, such as *eltom* and *ltome*. However, it does not detect the shared letter *w* occurring just before these *n*-grams, because that letter does not appear in the set of *n*-grams constituting the shared parts of the fingerprints. Also, it does not find that the word *Janus* is shared, because the overlap is too small for it to be preserved in the fingerprint. In general, although it is guaranteed that some *n*-grams will be shared when large text segments overlap (and more

Downloaded from http://llc.oxfordjournals.org/ at UB Leipzig on January 4, 2012

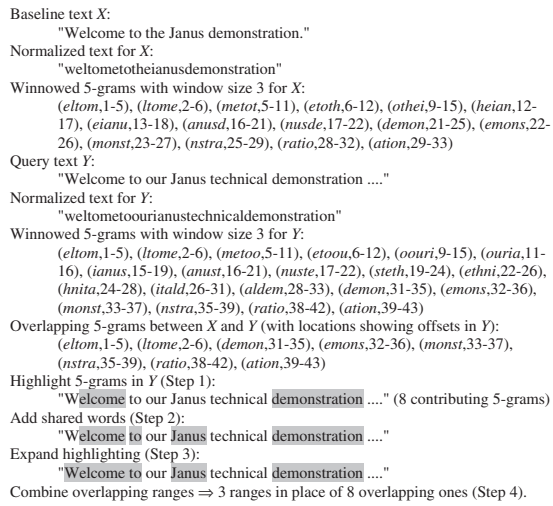


Fig. 2 Depiction of highlighting algorithm

n-grams will be shared when the overlap is larger), shared strings just before and after matched *n*-grams and small shared strings might go undetected.

To overcome these problems, the *n*-gram matches are clustered together if their locations in the query text and also in the baseline text appear close to each other (within 100 characters). The overlapping portions of the clusters are then highlighted in the query text using the following four steps:

(1) Find the locations of shared *n*-grams in the texts; call these *S*.
(2) Examine the texts directly to find the locations of shared words between adjacent values in *S*, or at the start or end of the cluster; call these *T*.
(3) Expand the values in *S* and *T* when the query text and the baseline text match. Maintain one unhighlighted character between values in *S* and *T* unless they are adjacent and align in both the query text and the baseline text.
(4) Combine overlaps that are adjacent and align in both the query text and the baseline text to produce fewer, but longer, overlaps.

The example presented in Fig. 2 demonstrates this algorithm. In the winnowed *n*-grams, locations are shown as *i*–*j*, where *i* is the offset of the start of the *n*-gram within the normalized text (counting from zero for the first character in the text), and *j* is the offset of the end of the *n*-gram. For example, 15–19 indicates that the corresponding 5-gram starts at the sixteenth character of the normalized text (the first character being numbered zero) and extends to the twentieth character of the normalized text. In this example, eight 5-grams are found to be in common between the baseline text and the query text. Between the second and third overlapping matches, we find that the two texts have the words *to* and *Janus* in common as well (Step 2). The letter *w* before the first 5-gram and a space after the second one are found to occur in both texts, so the highlighting is extended to cover these (Step 3), and finally overlapping 5-gram regions within the query text are merged to result in three distinct matches between the query text and the baseline text (Step 4).

We have implemented these algorithms and tested them extensively using quotations from the *Manipulus florum* as the baseline text collection. In particular, we have partitioned the *Manipulus florum* so that each quotation is indexed as a separate document. The search engine matches a normalized query against the normalized quotations and their extracted *n*-grams, the quotations are ranked by the number of matches, the position of each match is identified in the query text and the corresponding quotation, and the highlighting algorithm described above is applied. A sample display window is shown in Section 5, where we describe the user interface.

Our current implementation of the highlighting algorithm assumes that each *n*-gram appearing in a quotation has exactly one corresponding position in that quotation; that is, it does not handle multiple occurrences of an *n*-gram in a quotation. This greatly simplifies the task of aligning corresponding locations in the quotation and the query text when determining whether there are additional shared characters or words in their immediate contexts. Relaxing this assumption would require the system to consider all possible locations for each match, unnecessarily complicating the task of determining an appropriate highlighting to aid the user in assessing the matches. Similarly, our implementation also assumes that common words

Downloaded from http://llc.oxfordjournals.org/ at UB Leipzig on January 4, 2012

that appear in the context of a match (such as the words 'to' and 'Janus' in Fig. 2) do not repeat, thus simplifying the task of choosing which word occurrences in the two texts to highlight. As a third simplification, when identifying word overlaps and expanding overlaps (for example, in Fig. 2, to include the 'W' before 'elcome' and the blank after it), our implementation requires exact (albeit case insensitive) matches and, therefore, does not accommodate variations in orthography or typography (other than capitalization). These simplifications do not affect the determination of which baseline texts share text fragments with any query text, but only the highlighting of the precise overlaps themselves. However, if users of Janus find these limitations too severe, we will address them in future implementations.

4 Performance

The overlap detection algorithm used here has two input parameters, the n-gram size (n) and the winnowing window size (w). We ran our algorithm with various values of n and w using the *Moralium dogma philosophorum* (Holmberg, 1929) as the query text against approximately 2,700 *Manipulus florum* quotations as baseline texts. The matches were compared against a list of expected results obtained from prior manual methods. From this we produced a list of correct matches, which can then be used to measure relevance.

The algorithm's performance is displayed in Tables 2 and 3 using standard information retrieval definitions of precision and recall:

$$\text{Precision} = \frac{|\text{relevant} \cap \text{retrieved}|}{|\text{retrieved}|}$$

$$\text{Recall} = \frac{|\text{relevant} \cap \text{retrieved}|}{|\text{relevant}|}$$

Precision reflects the fraction of matched quotations that are correct (a scholar would agree that they share some text with the query), and recall reflects the fraction of all correct matches that are identified by the algorithm.

When performing intertextuality search with these texts, values of $n < 15$ produce slow queries, and the first column of Table 2 shows that they have low precision, i.e., many of the results returned are obviously coincidental matches and not proper variants of the *Manipulus florum* quotations. Values of $n > 20$ and values of $w > 20$ produce fast queries, but the recall is below 1, i.e., some of the expected results are not returned. The Janus search

Table 2 Results for various window (rows) and n-gram (columns) sizes (cells contain: *precision*/*recall*)

$w\backslash n$	10	15	20	25	30	35	40
10	0.04/1.00	0.38/1.00	0.92/0.96	0.99/0.94	1.00/0.86	1.00/0.80	1.00/0.74
15	0.05/1.00	0.49/1.00	0.94/0.95	0.99/0.90	1.00/0.83	1.00/0.76	1.00/0.69
20	0.05/1.00	0.54/0.98	0.98/0.94	1.00/0.86	1.00/0.82	1.00/0.75	1.00/0.68
25	0.06/1.00	0.62/0.95	0.97/0.88	1.00/0.86	1.00/0.76	1.00/0.70	1.00/0.68
30	0.07/1.00	0.67/0.93	0.97/0.86	1.00/0.82	1.00/0.74	1.00/0.70	1.00/0.64
35	0.08/1.00	0.68/0.93	0.97/0.83	1.00/0.76	1.00/0.71	1.00/0.68	1.00/0.62
40	0.08/0.99	0.72/0.90	0.97/0.82	1.00/0.74	1.00/0.69	1.00/0.64	1.00/0.58

Table 3 Results for various window (rows) and n-gram (columns) sizes (cells contain: *precision*/*recall*)

$w\backslash n$	15	16	17	18	19	20
15	0.49/1.00	0.65/1.00	0.71/0.99	0.87/1.00	0.85/0.98	0.94/0.95
16	0.50/0.99	0.70/0.98	0.73/0.99	0.88/1.00	0.85/0.98	0.95/0.95
17	0.50/0.98	0.70/0.96	0.73/0.99	0.88/1.00	0.85/0.98	0.98/0.95
18	0.51/0.98	0.71/0.96	0.75/0.98	0.88/1.00	0.85/0.98	0.98/0.94
19	0.52/0.98	0.72/0.96	0.74/0.96	0.89/0.99	0.87/0.98	0.98/0.94
20	0.54/0.98	0.73/0.96	0.75/0.95	0.89/0.99	0.87/0.96	0.98/0.94

Downloaded from http://llc.oxfordjournals.org/ at UB Leipzig on January 4, 2012

engine is intended to be a tool to reduce the amount of work required of a scholar to find *all* the locations where *Manipulus florum* quotations are used in a query text. As such, missing possible overlap locations should be avoided, and thus our choice of n and w should be constrained to values that yield perfect recall. Table 3 presents results for n and w between 15 and 20. When recall is 1, the highest precision occurs at $n = 18$ and $w = 18$. Therefore, these settings are used in the Janus search engine to maintain good performance, produce few false matches, and find all the desired results. Based on Table 3, when using these settings we expect to find all correct matches plus an additional 12% of false matches that will have to be pruned out using scholarly judgements.

5 User Interface

Janus can display matched results using excerpts from the query text and/or the full query text. The former choice displays a list of baseline texts (the quotations) that overlap the query text. For this display, each quotation is followed by those passages from the query text that contain the overlaps, where each passage excerpted from the query text is cut along sentence boundaries and the overlaps with the quotation are highlighted. When displaying the full query text, the portions of the query text that overlap with *Manipulus florum* quotations are again highlighted. If the user requests both the excerpt display and the full-text display, then each passage in the excerpt list contains a link to the location of that passage in the full-text display.

A previous version of the Janus search engine was demonstrated at the International Medieval Congress (Nighman *et al.*, 2008). The current search interface, shown in Fig. 3, allows the user to specify the query text and/or keywords as well as the display format and then execute the search. The results appear on one HTML page as shown in Fig. 4.

In the excerpt display, as shown at the top of Fig. 4, the list of matched quotations (each

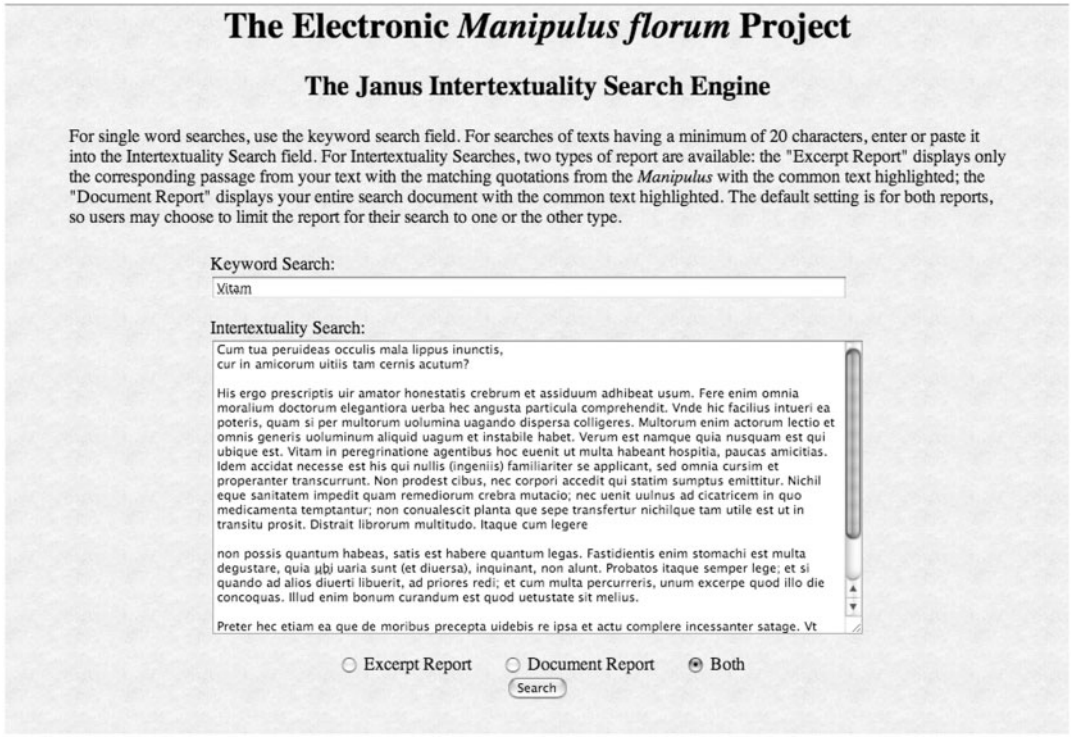
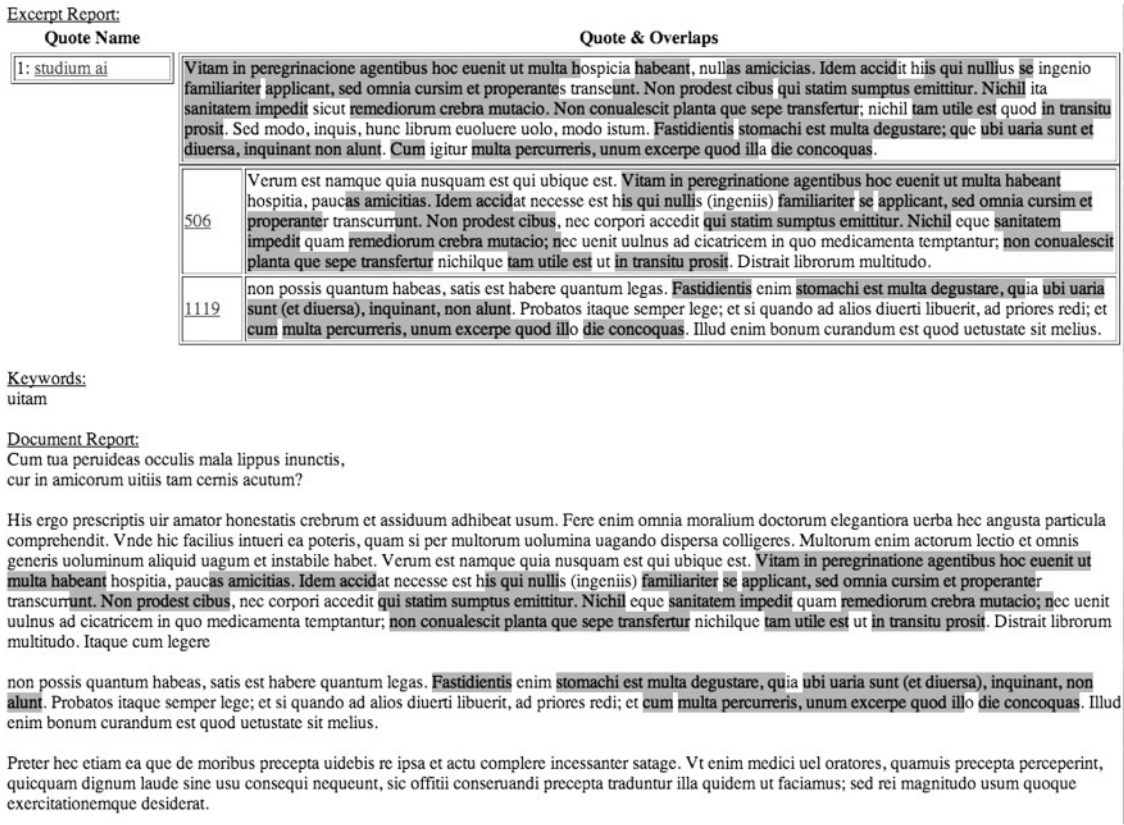


Fig. 3 A screen shot of the Janus search page

Downloaded from http://llc.oxfordjournals.org/ at UB Leipzig on January 4, 2012



Excerpt Report:

Quote Name	Quote & Overlaps
1: studium ai	Vitam in peregrinacione agentibus hoc euenit ut multa hospicia habeant, nullas amicicias. Idem accidit hiis qui nullius se ingenio familiariter applicant, sed omnia cursim et properantes transcunt. Non prodest cibus qui statim sumptus emittitur. Nichil ita sanitatem impedit sicut remediorum crebra mutacio. Non conualescit planta que sepe transfertur; nichil tam utile est quod in transitu prosit. Sed modo, inquis, hunc librum euoluere uolo, modo istum. Fastidientis stomachi est multa degustare; que ubi uaria sunt et diuersa, inquinant non alunt. Cum igitur multa percurreris, unum excerpe quod illa die concoquas.
506	Verum est namque quia nusquam est qui ubique est. Vitam in peregrinatione agentibus hoc euenit ut multa habeant hospitia, paucas amicitias. Idem accidat necesse est his qui nullis (ingeniis) familiariter se applicant, sed omnia cursim et properanter transcurrunt. Non prodest cibus, nec corpori accedit qui statim sumptus emittitur. Nichil eque sanitatem impedit quam remediorum crebra mutacio; nec uenit uulnus ad cicatricem in quo medicamenta temptantur; non conualescit planta que sepe transfertur nichilque tam utile est ut in transitu prosit. Distrait librorum multitudo.
1119	non possis quantum habeas, satis est habere quantum legas. Fastidientis enim stomachi est multa degustare, quia ubi uaria sunt (et diuersa), inquinant, non alunt. Probatos itaque semper lege; et si quando ad alios diuerti libuerit, ad priores redi; et cum multa percurreris, unum excerpe quod illo die concoquas. Illud enim bonum curandum est quod uetustate sit melius.

Keywords:
uitam

Document Report:
Cum tua peruideas occulis mala lippus inunctis,
cur in amicorum uitiis tam cernis acutum?

His ergo prescriptis uir amator honestatis crebrum et assiduum adhibeat usum. Fere enim omnia moralium doctorum elegantiora uerba hec angusta particula comprehendit. Vnde hic facilius intueri ea poteris, quam si per multorum uolumina uagando dispersa colligeres. Multorum enim actorum lectio et omnis generis uoluminum aliquid uagum et instabile habet. Verum est namque quia nusquam est qui ubique est. Vitam in peregrinatione agentibus hoc euenit ut multa habeant hospitia, paucas amicitias. Idem accidat necesse est his qui nullis (ingeniis) familiariter se applicant, sed omnia cursim et properanter transcurrunt. Non prodest cibus, nec corpori accedit qui statim sumptus emittitur. Nichil eque sanitatem impedit quam remediorum crebra mutacio; nec uenit uulnus ad cicatricem in quo medicamenta temptantur; non conualescit planta que sepe transfertur nichilque tam utile est ut in transitu prosit. Distrait librorum multitudo. Itaque cum legere

non possis quantum habeas, satis est habere quantum legas. Fastidientis enim stomachi est multa degustare, quia ubi uaria sunt (et diuersa), inquinant, non alunt. Probatos itaque semper lege; et si quando ad alios diuerti libuerit, ad priores redi; et cum multa percurreris, unum excerpe quod illo die concoquas. Illud enim bonum curandum est quod uetustate sit melius.

Preter hec etiam ea que de moribus precepta uidebis re ipsa et actu complere incessanter satage. Vt enim medici uel oratores, quamuis precepta perceperint, quicquam dignum laude sine usu consequi nequeunt, sic offitii conseruandi precepta traduntur illa quidem ut faciamus; sed rei magnitudo usum quoque exercitationemque desiderat.

Fig. 4 A screen shot of the Janus result page

hyperlinked to the corresponding *fons primus* document identifying its original source) is ordered by the underlying Lucene engine such that quotations with more *n*-gram or keyword matches appear before quotations with fewer. This is roughly, but not exactly, equivalent to ordering the quotations by the amount of overlapping text. Evaluating the appropriateness of this ordering and, if necessary, improving it is left for future work.

6 Conclusions

The Janus search engine finds overlaps between keywords, supplied text and *Manipulus florum* quotations in the presence of typographic, orthographic and complex variants. These overlaps are displayed so the user can quickly determine their significance. Janus simplifies tracking of the lineage of *Manipulus florum* quotations in historical texts. It is also valuable in determining the resources used by Thomas of Ireland to compile the *Manipulus florum*. For example, Janus was used extensively to explore the influence of John of Wales' *Communiloquium* on the *Manipulus florum* (Nighman, 2010).

Janus maintains no information after the results are displayed, thus avoiding copyright infringement for the supplied text, and making Janus more broadly applicable to scholarly research.

Online access to the Janus search engine is freely available, linked from the main *Manipulus florum* web page (<http://manipulusflorum.com/>).

The Janus engine is extendible to be used by scholars to compare two arbitrary texts. The first

Downloaded from http://llc.oxfordjournals.org/ at UB Leipzig on January 4, 2012

text should be broken into paragraphs and each paragraph entered into the Janus engine as if it were a *Manipulus florum* quotation. The second text would then be used as the query text. Thus Janus could be made more generally available for scholarly research if the demand exists.

Acknowledgements

Dr Chris Nighman posed the problem of designing an intertextuality search engine for the *Manipulus florum* project and suggested basing it on plagiarism detection software.

Funding

This work was supported by the University of Waterloo and the Natural Sciences and Engineering Research Council of Canada.

References

Altschul, S. F., Gish, W., Miller, W. *et al.* (1990). Basic local alignment search tool. *Journal of Molecular Biology*, **215**: 403–410.

Bourdaillet, J. and Ganascia, J. (2006). MEDITE: A unilingual textual aligner. *Paper presented at Advances in Natural Language Processing, Fifth International Conference on NLP, FinTAL*. Finland: Turku.

Fetterly, D., Manasse, M., and Najork, M. (2005). Detecting phrase-level duplication on the World Wide Web. *Paper presented at SIGIR '05: Proceedings of the 28th annual international ACM SIGIR Conference on Research and Development in Information Retrieval*. Brazil: Salvador.

Holmberg, J. (1929). *Das Moralium Dogma Philosophorum des Guillaume de Conches*. Uppsala, Sweden: Almqvist and Wiksell.

Hunt, J. W. and McIlroy, M. D. (1975). An algorithm for differential file comparison. *Computing Science Technical Memorandum* 75-1271-11. Bell Laboratories.

Kolak, O. and Schilit, B. N. (2008). Generating links by mining quotations. *Paper presented at HT '08: Proceedings of the Nineteenth ACM Conference on Hypertext and Hypermedia*. PA, USA: Pittsburgh.

Myers, E. W. (1986). An O(ND) difference algorithm and its variations. *Algorithmica*, **1**(2): 251–66.

Nighman, C. (2010). Revisiting the connection between John of Wales. Communiloquium and Thomas of Ireland's *Manipulus florum*: Results from the Janus Intertextuality Search Engine. *Third International Margot Conference, The Digital Middle Ages: Teaching and Research*. New York, USA.

Nighman, C., Kane, A., and Tompa, F. (2008). The intertextuality search eEngine for the electronic manipulus florum project. *Demonstration at the International Medieval Congress 2008*. Leeds, UK.

Rouse, R. H. and Rouse, M. A. (1979). *Preachers, florilegia and sermons: Studies on the Manipulus Florum of Thomas of Ireland*, Vol. 47. Toronto: PIMS Texts and Studies.

Schilit, B. N. and Kolak, O. (2008). Exploring a digital library through key ideas. *Paper presented at JCDL '08: Proceedings of the 8th ACM/IEEE-CS Joint Conference on Digital Libraries*. Pittsburgh PA, USA.

Schleimer, S., Wilkerson, D. S., and Aiken, A. (2003). Winnowing: Local algorithms for document fingerprinting. *Paper presented at SIGMOD '03: Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*. San Diego, CA, USA.

Tichy, W. F. (1984). The string-to-string correction problem with block moves. *ACM Transactions on Computer Systems*, **2**(4): 309–21.

Downloaded from http://llc.oxfordjournals.org/ at UB Leipzig on January 4, 2012