Volume xx (200y), Number z, pp. 1–12

Constructive Visual Analytics for Text Similarity Detection

A. Abdul-Rahman[1], G. Roe[3], M. Olsen[4], C. Gladstone[4], R. Whaling[4], N. Cronk[2], R. Morrissey[4], and M. Chen[1]

[1]Oxford e-Research Centre, and [2]Faculty of Medieval and Modern Languages, University of Oxford.
[3]College of Arts and Social Sciences, Australian National University.
[4]Department of Romance Languages and Literatures, University of Chicago.

Abstract

Detecting similarity between texts is a frequently encountered text mining task. Because the measurement of similarity is typically composed of a number of metrics, and some measures are sensitive to subjective interpretation, a generic detector obtained using machine learning often has difficulties balancing the roles of different metrics according to the semantic context exhibited in a specific collection of texts. In order to facilitate human interaction in a visual analytics process for text similarity detection, we first map the problem of pairwise sequence comparison to that of image processing, allowing patterns of similarity to be visualized as a 2D pixelmap. We then devise a visual interface to enable users to construct and experiment with different detectors using primitive metrics, in a way similar to constructing an image processing pipeline. We deployed this new approach for the identification of commonplaces in 18th-century literary and print culture. Domain experts were then able to make use of the prototype system to derive new scholarly discoveries and generate new hypotheses.

1 Introduction

In the text mining literature, there has been a huge volume of work on detecting similarity between or among texts (e.g., [SS95, CGPW97, FEC05, BCMB12]). Detection techniques often have to address particular requirements related to a specific context, such as languages, periods of articles, genres of articles, and so on. Hence the metrics used for measuring similarity are sensitive to subjective interpretation. For example, in some cases, measuring trigram similarity is more effective than unigram similarity, but vice versa in other cases. In some cases, it is desirable to include numbers in a comparison, but undesirable in other cases. This places a huge weight on a generic detector, typically devised algorithmically or obtained using machine learning, in balancing the roles of different metrics according to the semantic context exhibited in a specific collection of texts. Considering all detection models together, all possible variants of these models define a *model space*, which is enormous and complex. In bioinformatics, *sequence alignment* [Mou04] exhibits a similar phenomenon, but this is beyond the scope of this work.

In some domains of text analysis, the term "text alignment" has been used for detecting similarity between or among texts (e.g., [LMD01, BG07, HOR11]). Visualization techniques have been used to support such analytical processes (e.g., [HKBE12, JKM12, RPSF15]). This work explores the use of a visual analytics approach [DNKS10, OSSK10, DWS*12] to help analysts explore a model space effectively through closely integrated visualization, interaction, machine analysis, and human reasoning. In particular, we focus on a subset of models for *pairwise sequence comparison* in text analysis, and draw a parallel between text alignment and image processing.

Dot plots [GM70] are pixel-based visual representations for sequence comparison, more widely used in bioinformatics than in text analysis. They can be used to depict intermediate and final results of a detection model as "images". Hence a detection model is conceptually similar to an image processing pipeline. This parallel suggests that interactive visualization could support the construction of a text alignment pipeline, visualization of the model and corresponding results, and testing and improvement of the pipeline.

This work was conducted in collaboration with a team of domain experts in the fields of literary studies, intellectual history, and digital humanities. The domain experts have worked on computational models for detecting text alignment for a number of years. As it is costly to test a model over a large text corpus in terms of computational resources as well as the time required to run the model and analyze the results, the domain experts wished to observe behavior of a model and its variations in relation to various scenarios in

text alignment. In order to enable domain experts to explore the model space, we defined a set of primitive metrics, and a number of processing components. The domain experts referred to models as *methods*, and primitive metrics and processing components as *tools*. Each text alignment pipeline is thus a method composed of a number of tools. This was followed by the design and prototyping of a visual analytics interface for constructing a text alignment pipeline, and testing it with various sample texts. We deployed this new approach for the identification of commonplaces in 18th-century literary and print culture. Domain experts were then able to make use of the prototype system to derive new scholarly discoveries and generate new hypotheses. In summary, our contributions are as follows:

- We have helped the advancement of visual analytics in the scope of model development and optimization through the design and implementation of a novel software system and an application case study.
- We have mapped the models for text alignment to image processing pipelines (Section 4), which facilitates a mechanism for decomposing a text alignment method (i.e., model) to a set of tools (i.e., primitive metrics and processing components) (Section 5).
- We developed a web-based visual analytics interface to enable domain experts to construct a text alignment pipeline, visualize the components and connections inside a method (i.e., a model) as well as the corresponding testing inputs and outputs. The interaction with the pipeline editing facility essentially becomes a visual programming interface (Section 6).
- The domain experts tested the software prototype with huge enthusiasm, and observed a number of new phenomena in text alignment in the context of 18th-century literary and print culture. As examples, we describe two such phenomena in the context of usage of function words in text mining and the detection of disjointed segments of commonplaces due to passage re-usage in Section 7.

2 Related Work

There are many visual representations for visualizing text and documents. These include a variety of statistics graphics (e.g., [HHN00]), tag clouds (e.g., [Fei]), pixel-based visualization (e.g., [OSSK10]), multivariate visualization (e.g., [ARLC*13]), and graphs and networks (e.g., [Fel98, Hol06, CCP09, vHWV09]).

In recent years, a number of visual analytic tools have been developed for text and document visualization in the context of digital humanities. Vuillemot *et al.* [VCPK09] present a multi-view visual interface (called *POSvis*) for examining named entities in literary texts. Correll *et al.* [CWG11] construct a collection of visualization tools for analyzing large collections of texts that have been preprocessed using natural language processing techniques. Oelke *et al.* [OKK13] describe the *Fingerprint Matrices* technique for analyzing the changing relationships between characters in a novel. Koch *et al.* [KJW*14] present a method called *VarifocalReader* for examining a large corpus of text with multi-level abstract representations. Jänicke *et al.* [JGBS14] present several visualization techniques for highlighting differences and similarities of various English Bible translations. Our work differs from these tools by focusing on the visualization capability for model construction and improvement.

Several approaches have been developed in model construction and visualization. Bosch *et al.* [BTH*13] describe a method called *ScatterBlogs2* for allowing users to construct message filters based on existing events. Koop *et al.* [KSC*08] present a predictive workflow constructor called *VisComplete*. Our work shares the same overall objective for empowering users in model development, while bringing text alignment in line with image processing.

Pairwise sequence alignment is a technique that is commonly used in bioinformatics for comparing genetic sequences [Mou04, HHB07]. Dot plot, introduced by Gibbs and McIntyre [GM70], is one of the techniques that is used to compare genetic sequences and help identify similarity between them. Dot plot is a 2D matrix where two sequences of symbols are placed along the vertical and horizontal axes respectively, and a dot is placed when the symbol at the intersecting vertical and horizontal axes matches. Many of the tools available for visualizing genetic sequences can be found in a review by Nielsen *et al.* [NCD*10]. Dot plot technique has also been used in the identification of text reuse (or text plagiarism) (e.g., [CH93, Lee07, JGBS14]).

A number of systems have been developed to help with the detection of software plagiarism (or similarity in software code). In most of these systems [Aik, JL99, GT99], the result is presented to the user as a ranked list that has been computed through a distance matrix. Instead of a ranked list view, Freire [Fre08] presents the result of the distance matrix using a combination of graph and histogram representations. Ribler and Abrams [RA00] propose an *n*-gram based visualization called *Categorical Patterngram* to help detect software plagiarism. Sequence alignment has also been used to help in identifying text reuse (e.g., [CGPW97, LMD01, BG07, HOR11]). Jankowska *et al.* [JKM12] present a *n*-gram approach of visualizing documents at a word-level. There are commercial systems for detecting text plagiarism (e.g., [Wri, Tur]). White and Joy [WJ04] present an approach of identifying text plagiarism using sentence-level comparison. Clough [Clo03] presents a review of automatic techniques for detecting text plagiarism.

3 Application Background and Requirements Analysis

Application Background. This work was conducted in the context of 18th-century literary and print culture. Example texts used for testing and developing our visual analytics web application *ViTA*, *Visualization for Text Alignment*,

were drawn from the ARTFL-FRANTEXT database of 18th-century French literature [ARTa], housed at the University of Chicago, and the ECCO (Eighteenth-Century Collections Online) [ARTb] database provided to the project by the online publisher Gale-Cengage Learning [Cen]. Historically, the 18th century can be seen as one of the last in a long line of "commonplace cultures" extending from Antiquity through the Renaissance and Early Modern periods. Here the term *commonplacing* denotes the thematic organization of quotations and other passages for later recall and reuse. In other words, two similar sequences in texts are potentially *commonplaces*. Recent scholarship has demonstrated that the various rhetorical, mnemonic, and authorial practices associated with Early Modern commonplacing were highly effective strategies for dealing with the perceived "information overload" of the period, as well as for functioning successfully in polite society (e.g., [All10, Bla11]). The research goal of the domain experts has been to examine this paradigm shift in 18th century culture from the perspective of commonplaces and through their textual and historical deployment over large scale text collections and across various contexts of collecting, reading, writing, classifying, and learning [EMR13].

While text reuse (or text borrowing) was a common phenomenon in 18th-century print culture, the majority of authors then did not practice modern conventions of citation, even in reference works such as the French *Encyclopédie* [EMR13]. In addition, it was common practice for authors to "borrow" extensive passages from elsewhere, without attempting to cover their traces, in the full knowledge that readers "in the know" might recognize these borrowings or re-writings. The 18th-century French philosopher Voltaire (also known as François-Marie Arouet (1694–1778)) even borrows extensively from himself. These stylistic complexities have been until now extremely difficult to analyze. Hence detecting commonplaces, or other forms of shared passages, in 18th-century digital collections is both of scholarly interest and technologically challenging.

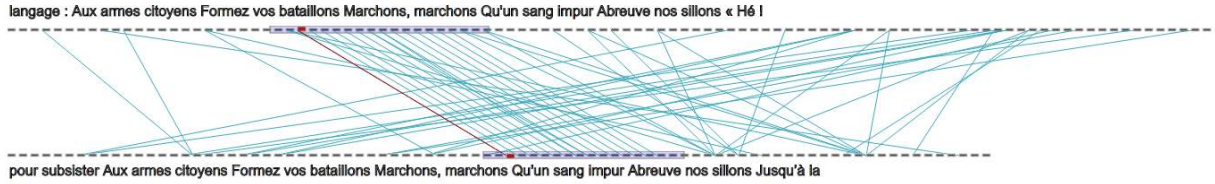
Requirements Analysis Before this work started, the domain experts had substantial experience in using automated methods, such as machine learned models, to detect commonplaces in collected texts. While these methods achieved significant successes, they also resulted in many false positives, which were time-consuming to eliminate. Many of these false positives are artifacts of the various digital archives from which the domain experts draw their texts – namely large scale literary and historical databases in French and English. As the digitization process used to build these archives is largely automatic, high frequency sequences such as publisher information, running headers or titles, and bookseller advertisements have been included along with the literary or historical works themselves, leading to a high number of uninteresting or spurious commonplace passages. The domain experts first shared many examples of these false positives with the interdisciplinary team, which led to the decision to adopt a visual analytics approach to bring humans back into the loop. By allowing domain experts to examine a decomposed detection model, apply it to testing texts and visualize the results, we can empower them to optimize a model based on their knowledge of the related background (e.g., authors, period, context, etc.) and the research questions most relevant to investigations into 18th century print culture.

As the partners of the interdisciplinary team are distributed across three different continents, we made use of two sessions of face-to-face meetings, each over several days, as the major landmarks of the collaboration. In the first set of meetings in May 2014, we explored the design of two visual representations, and specified most primitive metrics and processing components as *tools* for constructing *methods*. We identified that the main visualization task was to help users create and optimize a method or several methods for detecting commonplaces with a number of short texts. The visual representations should allow users to observe the alignments identified by a method as well as the impact of varying its tool components and their parameters. This process would allow the users to obtain a set of relatively optimized methods for large scale computation to identify commonplaces in a large text corpus. We also recognized the difficulty for humanities scholars to gain access to text analysis software in general. Hence, offering a web-based software service would be attractive to a broad range of users in the humanities. Of course, this would require the design of the software to accommodate the diversity of users and the constraints of a web-based application. The team decided to accommodate both requirements, developing a web-based application for experimenting with methods (i.e., algorithmic pipelines) in the context of commonplace identification in English and French.

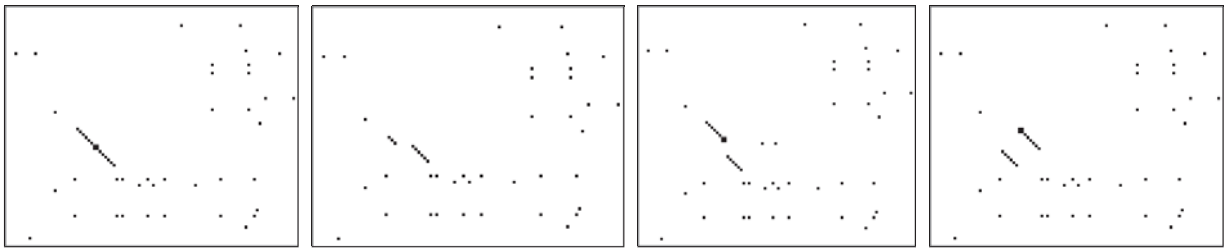
The web-based interface thus needed to be both highly flexible from a methodological perspective and intuitively designed for humanities users with little or no programming experience. The ability to identify commonplaces is deeply subjective, and thus open to many interpretations. As such, humanities scholars need methods and tools that can both identify potential commonplaces and provide a snapshot of the matching process itself, allowing users to modify methods and tools iteratively after initial results analysis. This process in effect becomes a scholarly feedback loop, in which lessons learned from low-level matching pipelines can be combined and added to subsequent methods aimed at more fine-grained matching and analysis. In this way scholars can engage in the algorithmic process in ways that would have been otherwise impossible or technologically intractable for the average humanities user.

4 Text Similarity and Image Processing

A text is an ordered sequence of words. In this work, we denote a text as $T = \{t_1, t_2, \ldots, t_l\}$, where t_i is a *word ele-*



(a) An example of perfect text alignment using bipartite graph representation



(b) A perfect alignment (c) Text deletion (d) Text insertion (e) Text swapping

Figure 1: *Example of text alignment using (a) bipartite graph representation and (b)-(e) 2D pixelmap.*

ment that is an independently-meaningful sequence of characters. To compare two texts, $T_a = \{t_{a,1}, t_{a,2}, \ldots, t_{a,n}\}$ and $T_b = \{t_{b,1}, t_{b,2}, \ldots, t_{b,m}\}$, one can apply a similarity metric $M(t_{a,i}, t_{b,j})$ to all pairs of word elements in T_a and T_b. The metric M returns a value to indicate the level of similarity. Without losing any generality, we assume that the returned similarity values are in the domain $[0,1]$, where 0 implies totally different, and 1 the same. It is not necessary for M to restrict its comparison to $t_{a,i}$ and $t_{b,j}$, and the comparison often involves neighboring word elements in a given sliding window. Nevertheless, each returned measure is always associated with a unique 2D position $(i, j), 1 \leq i \leq n, 1 \leq j \leq m$. Hence, one may write a generalized form of a similarity metric as $M(T_a, T_b, i, j)$.

Given all similarity measures, $s_{i,j}$, we can organize them into an $m \times n$ matrix. As a convention, the first text T_a is referred to as *source text*, and the positions of its word elements correspond to column indexes. The second text T_b is referred to as *target text*, and the positions of its word elements correspond to row indexes. The similarity matrix $\mathbf{S}$ thus takes the following form:

$$\mathbf{S}_{ab} = \begin{bmatrix} s_{1,1} & s_{1,2} & \cdots & s_{1,n} \\ s_{2,1} & s_{2,2} & \cdots & s_{2,n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{m,1} & s_{m,2} & \cdots & s_{m,n} \end{bmatrix}$$

One intuitive way to visualize the similarity between two texts is to display the word elements of T_a and T_b along two parallel lines as shown in Fig. 1(a). One then represents each non-zero $s_{i,j}$, as a connection path between position i on the first line and j on the second. This is in effect a *bipartite graph*. The value of $s_{i,j}$ can be encoded using visual channels such as opacity, color, and thickness. In Fig. 1(a), the level of similarity [0, 1] is mapped to the range of opacity [0, 1]. From this figure, we can observe some text alignment patterns suggested by a sequence of parallel or near parallel connection paths between the two text lines. It is not difficult to notice that occlusion and cluttering can be a major issue. When many connection paths cross each other, observing the alignment patterns becomes difficult.

An alternative way of visualizing similarity patterns is to display the similarity matrix as an $m \times n$ dot plot, similar to an image, except that each dot may be shown as a block of pixels. Figs. 1(b)-(e) show examples of four dot plots. In general, the dot plots do not suffer from occlusion and cluttering issues, and are more likely to successfully scale than bipartite graphs in terms of space utilization. The advantage of dot plots over the bipartite graphs was also shown in the context of genome comparison [HHB07]. However, showing of texts with a dot plot is not as intuitive as displaying them along the two lines in a bipartite graph. Hence these two visual representations are complementary.

Once a user is accustomed to the pixel-based visualization, different alignment patterns are also more observable. In Fig. 1(b), a continuous line of dots suggests that there is a perfect text alignment between two common passages in the two texts respectively. In (c), the horizontal gap between two aligned segments suggests that some word elements in the source text have been deleted from a common passage in the target text. In (d), the vertical gap suggests that some new word elements have been inserted into the common passage of the target text. In (e), the relative locations of two aligned segments suggest that there is a common passage between two texts, except that it has been divided into two sections with their order swapped in the target text.

When dot plots are considered as images, we can utilize image processing techniques (e.g., [GW08]) to improve such plots. Any improvement of a dot plot corresponds directly to the improvement of text alignment. For example, if one uses an image processing tool to remove isolated dots

Table 1: *The 41 tools that are currently available in the framework.*

Tools	Categories	Descriptions
Any Word	Word Matching	Arbitrary word matching using n-gram
Any Word (with order invariance)	Word Matching	Arbitrary word matching using n-gram, while disregarding word ordering
Any Word (without function words)	Word Matching	Arbitrary word matching using n-gram, while excluding function words
Any Word (without numbers)	Word Matching	Arbitrary word matching using n-gram, while excluding numbers
Any Word (with punctuations)	Word Matching	Arbitrary word matching using n-gram, with punctuations
Any Word (with flatten accents)	Word Matching	Arbitrary word matching using n-gram, while flattening accented characters
Any Word (with synonyms)	Word Matching	Arbitrary word matching using n-gram, while allowing matching with synonyms using the online French and English thesauri of OpenOffice [The]
Keywords	Word Matching	Specific word matching using n-gram
Keywords (with order invariance)	Word Matching	Specific word matching using n-gram, while disregarding word ordering
Keywords (without function words)	Word Matching	Specific word matching using n-gram, while excluding function words
Keywords (without numbers)	Word Matching	Specific word matching using n-gram, while excluding numbers
Keywords (with punctuations)	Word Matching	Specific word matching using n-gram, with punctuations
Keywords (with flatten accents)	Word Matching	Specific word matching using n-gram, while flattening accented characters
Keywords (with synonyms)	Word Matching	Specific word matching using n-gram, while allowing matching with synonyms (as Any Word with synonyms)
Any Word and Keywords	Word Matching	Arbitrary and specific word matching using n-gram
Any Word and Keywords (with order invariance)	Word Matching	Arbitrary and specific word matching using n-gram, without word ordering
Any Word and Keywords (without function words)	Word Matching	Arbitrary and specific word matching using n-gram, without function words
Any Word and Keywords (without numbers)	Word Matching	Arbitrary and specific word matching using n-gram, while excluding numbers
Any Word and Keywords (with synonyms)	Word Matching	Arbitrary and specific word matching using n-gram, while allowing matching with alternative synonyms (as Any Word with synonyms)
Word Length	Language Processing	Scaling each value in the similarity matrix by word length
Word Frequency	Language Processing	Scaling each value in the similarity matrix input by word usage frequency
Inverting	Visual Processing	Inversion of the similarity matrix
Banding	Visual Processing	Grouping the values in the similarity matrix into a number of bands
Darkening	Visual Processing	Increase or decrease brightness of the similarity matrix
Contrasting	Visual Processing	Change the contrast of the similarity matrix by changing the range of the luminance values
Thresholding	Visual Processing	Set all value above a certain threshold level to white, and the rest to black
Box Smoothing	Visual Processing	Modify the similarity matrix by convolution using a Box Smoothing filter
Gaussian Smoothing	Visual Processing	Modify the similarity matrix by convolution using a Gaussian filter
Sharpening	Visual Processing	Modify the similarity matrix by convolution using a Sharpening filter
Unclutter	Visual Processing	Remove single, unconnected points
Diagonal Grow	Visual Processing	Expand an alignment pattern diagonally
Diagonal Fill	Visual Processing	Connect line segments with diagonal gaps between them
Horizontal Fill	Visual Processing	Connect line segments with horizontal gaps between them
Vertical Fill	Visual Processing	Connect line segments with vertical gaps between them
Union	Operator	Merge two or three pipelines using the union function
Intersection	Operator	Merge two or three pipelines using the intersection function
Blend	Operator	Merge two or three pipelines using the blending function
Pass	Operator	Passes on the input similarity matrix to the output
Difference	Operator	Merge two or three pipelines using the difference function
Mult	Operator	Merge two or three pipelines using the multiplication function
Add	Operator	Merge two or three pipelines using the addition function

in Figs. 1(b-e), it corresponds to the elimination of isolated unigram matching in text alignment. As image processing techniques are typically organized into a pipeline, it suggests that elementary tools of text alignment can also be organized into a pipeline. At every stage of an image processing pipeline, one can observe the intermediate images as a means of debugging. Similarly at every stage of a text alignment pipeline, one could observe the intermediate dot plots as a means of debugging. When observing a dot plot while constructing and debugging such an "image" processing pipeline, it is necessary to maintain a stable spatial mapping from words to dots. It is not desirable to compress out the white space in a dot plot at this stage as they may vary with any slight change to the pipeline or parameters.

5 Constructive Text Alignment

In this section, we present a framework for *Constructive Text Alignment*, which defines the mathematical constraints of the tools that are used as the components of a text alignment pipeline (i.e., a method). In this framework, each tool is a function that operates essentially on one or more input similarity matrices and generates a similarity matrix as an output. As mentioned previously, a similarity matrix is equivalent to a grayscale image, these tools or functions are analogous to filters in image processing. Table 1 shows the collection of tools that have been defined in our framework. They are grouped into four categories: *Word Matching* (**M**), *Language Processing* (**L**), *Visual Processing* (**V**), and *Operator* (**O**).

All word matching tools have access to two texts and a similarity matrix as the inputs and a similarity matrix as the output. Hence they take a common form of $\mathbf{M} : \mathbf{T}_n \times \mathbf{T}_m \times \mathbf{S}_{m\times n} \rightarrow \mathbf{S}_{m\times n}$, where $\mathbf{T}_x$ is the set of all texts with x word elements, and $\mathbf{S}_{m\times n}$ are the set of all $m \times n$ similarity matrices. When a word matching tool appears at the beginning of a pipeline, the input $\mathbf{S}$ is assumed to contain all 0s. The word matching tools are divided into three subcategories, "Any Word", "Keyword", and both. The *Any Word* tools will match any words in the two input texts as long

as they meet the matching criteria, such as n-gram matching. The *Keyword* tools will match only those words in a list of keywords specified by the user. This allows for a more targeted search for common passages. For example, a user may be only interested in common passages that contain words "sun", "moon", "star", etc. The third category of word matching tools allow for matching arbitrary words, and return a higher similarity value for keyword matching and a lower value for non-keyword matching.

Each category consists of several variations. For example, one matches n-grams without considering the order of word elements in each n-gram. Another matches each word element with its synonyms. For the synonyms match, we use the online French and English thesauri of OpenOffice [The], which allow real-time retrieval of synonyms of a given word. Others ignore the matching of function words or numbers. We purposely designed multiple tools for different variations in order to minimize the number of parameters in each tool. We found that users in arts and humanities are more accustomed to choosing the appropriate tool than customizing a generic tool.

Language processing tools are used to modify values in the input similarity matrix based on the two texts. For example, the *Word Length* tool allows each value at position (i, j) in the similarity matrix to be scaled according to the lengths of the word elements $t_{a,j}$, and $t_{b,i}$. Such a scaling can be used to weight the matching of long word elements more than short ones. Hence, these tools can also take a common form of $\mathbf{L} : \mathbf{T}_n \times \mathbf{T}_m \times \mathbf{S}_{m\times n} \rightarrow \mathbf{S}_{m\times n}$.

The *Visual Processing* tools are very similar to traditional image processing filters. Reading the names of these tools in Table 1, such as "Contrasting", "Gaussian Smoothing", one can easily relate them to similar filters in image processing. In several cases, domain experts preferred to rename them to avoid clashing with certain connotations in arts and humanities. Tools in this category do not handle texts, and all have one similarity matrix as the input and another as the output. They thus take the same form of $\mathbf{V} : \mathbf{S}_{m\times n} \rightarrow \mathbf{S}_{m\times n}$.

The *Operator* tools apply real-value constructive operations to one or more input similarity matrices. For example, "Union" and "Intersection" tools can be used to combine two or more input similarity matrices. Because they are real-value operators, we use pixel-based (or piecewise) MAX and MIN functions to achieve the union and intersection as in *Constructive Volume Geometry* [CT00]. All of the mathematical definitions of the *Operator* tools can be found in [CT00] with the exception of the *Pass* tool. Tools in this category take the form of $\mathbf{O} : \mathbf{S}^k_{m\times n} \rightarrow \mathbf{S}_{m\times n}$, where $k = 1, 2, 3$. Although many tools in this category can take a number of inputs, we limit k to 3, to reduce the memory requirements for multiple similarity matrices since $m \times n$ in text processing is typically a large number.

All tools in the four categories can be slightly customized. Most customization facilities are unavoidable, such as specifying n for n-gram matching. Some tools allow for the specification of a sliding window or the size of a convolution kernel. We made a conscious effort to keep the same look and feel for all properties windows for setting parameters. For example, each property window starts with an input filtering threshold, defining only certain values (e.g., > 0.5) in the input similarity matrix will be processed.

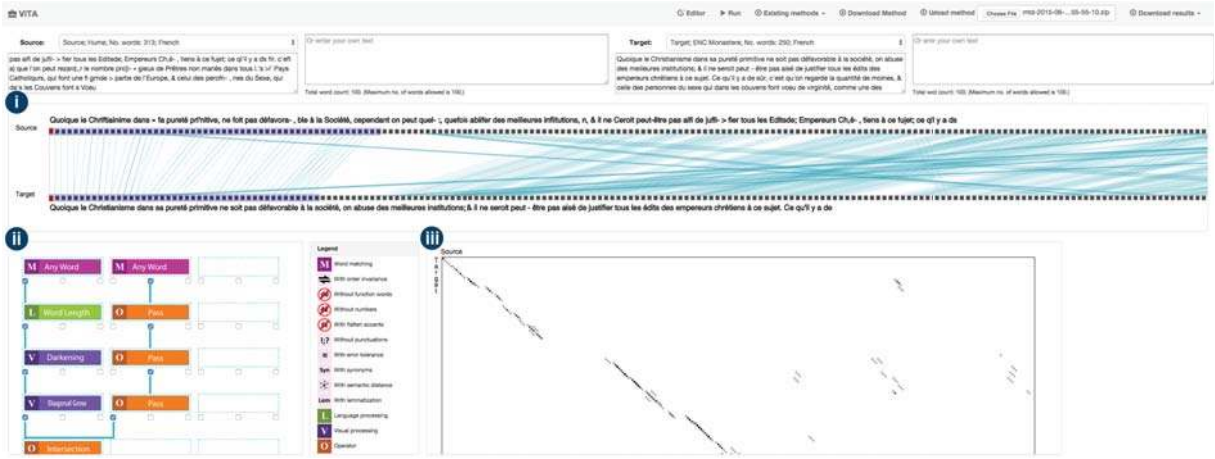
6 Visual Interface

We developed a visual analytics web application called *ViTA*, *Visualization for Text Alignment*. ViTA is designed to enable users to create specific methods that are suited to the users' particular context, such as languages and genres of articles, to be deployed on a larger, offline text alignment pipeline. Fig. 2 shows a screenshot of ViTA. ViTA uses the concept of visual programming [BD94] to enable users to construct a method by selecting appropriate tools from a menu and connecting them together. This empowers users, who may not be expert programmers, with the capabilities and abilities to construct a "program" to serve their research objectives. It also allows users to explore new hypotheses.

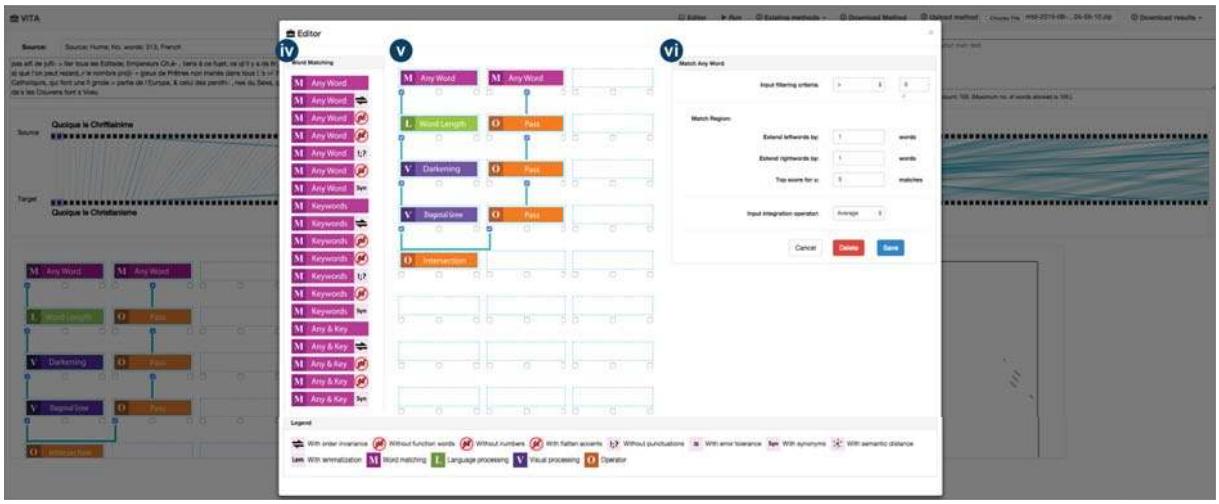
There are two ways in which users are able to analyze a text in ViTA: either by selecting one piece of the pre-stored texts or pasting one into the input text areas. Fig. 2(a) shows the main visual interface, which has three main components. Fig. 2(a)(i) shows the *Bipartite Graph View Panel* for displaying the connection paths between the two input texts. Users are able to view the text by brushing along any of the two parallel lines. A connection path between the two parallel lines indicates that the two word elements in the two texts are similar or related, and its strength of similarity is represented by the opacity channel.

Fig. 2(a)(ii) shows the *Method Overview Panel* for displaying the method that has been used to detect text alignment. Clicking on the *Method Overview Panel* opens up the *Method Editor* for creating a text alignment pipeline, i.e., a method. While Fig. 2(a)(iii) shows the *Pixelmap View Panel* representing the similarity matrix through a dot plot where the first input text (i.e., the source text) is located along the x-axis and the second input text (i.e., the target text) is located along the y-axis. Once again the similarity between the two input texts is encoded using the grayscale channel. Clicking on any of the pixels in the *Pixelmap View Panel* would highlight the corresponding texts in the *Bipartite Graph View Panel* as well as its connecting path.

Fig. 2(b) shows the *Method Editor*. The list of available tools in ViTA can be accessed through the *Tools Panel* (Fig. 2(b)(iv)). Each of the tools that is defined in Table 1 is represented as a single block in the panel. The users are able to *drag-and-drop* a tool from the *Tools Panel* to a desired position in the *Grid Layout Panel* (Fig. 2(b)(v)). The *Tools Panel* is an accordion panel component where each group of



(a) The main visual interface for ViTA. (i) A *Bipartite Graph View Panel* showing the connecting edges between the two texts. (ii) A *Method Overview Panel* showing a method being developed for text alignment. (iii) A *Pixelmap View Panel* showing the similarity matrix as a dot plot.



(b) The *Method Editor* for creating a pipeline with a composition of tools. (iv) The *Tools Panel* holds a selection of tools available in ViTA. (v) The *Grid Layout Panel* is a grid of 8 rows $\times$ 3 columns for representing up to three parallel flows in the method. (vi) The *Attributes Panel* is for modifying the parameters of the currently-selected tool.

Figure 2: *An overview of ViTA, a visual analytics tool for developing detection models for text alignment.*

tools can be "expanded" or "collapsed" by clicking on the header of the group.

To help with the memorization of the tool block, logologic links are used to relate the types of tools to colors, such as **L** for Language/Lime, **M** for Matching/Magenta, **O** for Operator/Orange, and **V** for Visual/Violet. The design of each tool icon is divided into three parts (starting from left to right) indicating its specific category (i,.e., whether it is *Language Processing*, *Word Matching*, etc.), its subcategories, for example, "Any Word", "Keywords", etc., in its main group category of *Word Matching*, and then its variations of the subcategories, for example, "With order invariance" or "Without function words". This is also represented by the changing of the color, for example, from a dark magenta to a lighter version of magenta.

The *Grid Layout Panel* comprises a grid of 8×3 cells where the ordering of grid cells begins at the top-left corner. In comparison to a canvas-based block diagram, the grid restricts the freedom of placing a tool block arbitrarily and connecting them using polylines. However, as humanities scholars wish to focus on scholarly questions rather than engineering activities, they viewed the restriction in block placement, connection drawing as well as the relatively smaller grid size as advantages. This design also allowed the model and the visualization results to be displayed side-by-side.

Each tool in the *Grid Layout Panel* is linked to another using a connector, accepting one input connection. The exception are those in the *Operator* category, which can accept a maximum of three input connections. Only the *Operator*

tools cannot be placed in the first row. Underneath each dotted cell there are three checkboxes representing the *on*/*off* switches for the output connections to the tools in the row below. All tools can have up to three output connections, and each switch corresponds to one of the three parallel pipelines, i.e., first switch for the left, and second middle and third right. A pipeline is valid when the last filled row contains only a single tool. Users are able to adjust the parameters for each tool using the *Attributes Panel* (Fig. 2(b)(vi)).

Once a method has been created, users are able to run the created pipeline of tools (i.e., a method) by clicking on the *Run* option at the top-right corner in the main visual interface. A number of other features, such as the downloading and uploading of created methods as well as the functions of saving the generated image of the 2D pixelmap and the downloading of the matching text alignment, can also be found at the top-right corner in the main visual interface.

7 Testing and Evaluation

The main evaluation session was carried out in the second set of meetings in October 2014. The domain experts were those directly involved in the requirements analysis (see Section 3). The domain experts spent considerable time testing different tools and their parameters together with the visualization scientists, with the visualization scientists responded to the feedback by making various minor improvements concurrently. During the evaluation meetings, the domain experts used their wealth of knowledge about the literature, and pulled out various texts from different articles contained in the ARTFL-FRANTEXT [ARTa] and ECCO [ARTb] databases to test the software. Although domain experts had never used dot plots before, it soon became apparent that the domain experts found the dot plot more effective than the bipartite graph, and had little difficulty using the pipeline editor to construct different methods. In particular, they discovered a number of unexpected phenomena in text alignment.

One of the phenomena was the use of function (or stop) words. The conventional wisdom in text mining has been to remove all function words before applying any similarity metrics (e.g., n-gram matching). Using the dot plots, domain experts discovered a commonplace that the previous automated method failed to detect (i.e., false negative). They quickly hypothesized that the removal of all function words might have caused other false negatives. With their knowledge of the literature, they soon discovered a few other examples that illustrate this phenomena.

During a discussion about connecting disjointed segments of commonplaces, one domain expert noticed another phenomenon. When a passage was reused in another article, some parts of the passage were often deleted or pieces of new text were inserted. Both deletion and insertion resulted in disjointed line segments in the corresponding dot plots, as well as separated commonplaces in the detection results. Hence a specific tool would be required to reconnect the disjointed line segments, which would subsequently derive a merged commonplace. When observing various dot plots, the domain expert hypothesized that the average length of deletion might be longer than that of insertion. This led to further examination of various passages, and a decision to separate the connection tool into a horizontal connection tool (for deletion) and a vertical tool (for insertion). The former tool allows for connecting line segments that are further apart than the latter.

Fig. 3 illustrates these two phenomena. Fig. 3(a) shows the two passages that are being compared, where the red text was detected by PhiloLine [ART09], a sequence alignment extension to the ARTFL Project's PhiloLogic search engine, as a match and the blue text can be considered as a match as well, but was missed by PhiloLine. Comparing the red texts in the source and target, we can observe that some part of the source text was deleted in the target. Meanwhile the text in blue should still logically be part of the common sequence, but was missed by the automated detection system.

As shown in Fig. 3(b), when function words are not considered in the text alignment, the patterns in the dot plot are rather sparse. When function words are included in the similarity measures, the alignment patterns in Fig. 3(c) are more clearly observable, especially the third long segment towards the bottom-right of the plot. Such visualization helped domain experts to reassess the traditional wisdom about function words in text alignment.

One domain expert commented: "What I find particularly interesting in the function word examples is that it may be a way to control for the extremely dirty OCR. Given any error rate, it is more likely that longer 'content' words will contain an error, thus eliminating it as a valid match from any exact match scheme. Shorter function words will have less chance of error. ... More generally, there have been various kinds of studies, in areas like author attribution and stylometry, which are based on patterns of function words. So, ViTA made this kind of initial experiment both easily performed and quite compelling."

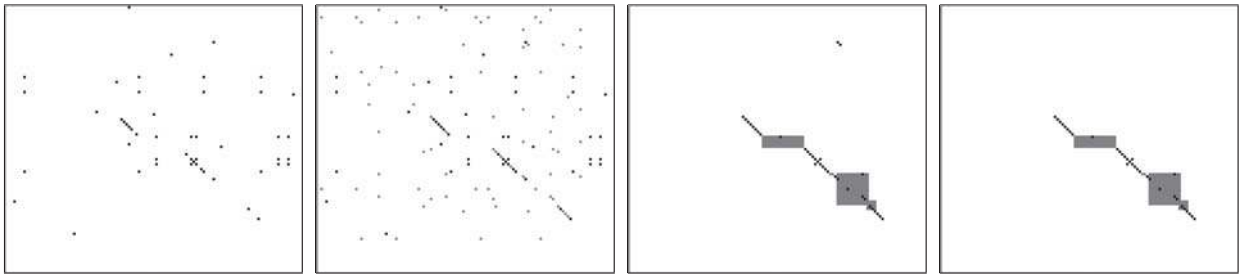
Another domain expert commented: "The general idea here is that for text mining it's often taken for granted that you throw out the most common words (which are by and large small function words), for reasons of efficiency and to reduce dimensionality. This may, however, result in us missing certain sequences if they contain mainly function words. ... So this [example] tells us that by including function words, we can, in some cases, extend matches beyond where our traditional sequence alignment approach stops."

After the meetings in October 2014, we developed three new image processing tools, *Diagonal Fill*, *Horizontal Fill*, and *Vertical Fill*, following the specification by the domain experts. Fig. 3(d) shows the results of applying two imaging processing tools to the dot plot in (c). The first is an integrated erosion-dilation tool that removed all single points in (c). The second is a horizontal fill tool, which extends at both

Source: Rousseau, Jean-Jacques, 1712-1778., [1758], *Lettre à Mr. d'Alembert sur les spectacles*:
publics, en voici un rapporté par Plutarque . Il y avoit, dit-il, toujours trois danses en autant de bandes, selon la différence des âges; et ces danses se faisoient au chant de chaque bande. Celle des vieillards commençoit la première, en chantant le couplet suivant, nous avons été jadis, jeunes, vaillans et hardis . Suivoit celle des hommes, qui chantoient à leur tour en frappant de leurs armes en cadence, nous le sommes maintenant, à l'épreuve à tout venant . Ensuite venoient les enfans qui leur répondoient, en chantant de toute leur force, et nous bientôt le serons, qui tous vous surpasserons . Voilà, monsieur, les spectacles qu'il faut à des républiques. Quant à celui dont votre article Genève

Target: Guys, M. (Pierre-Augustin), 1721-1799., [1771], *Voyage littéraire de la Grèce, ou, Lettres sur les Grecs, anciens et modernes, avec un parallèle de leurs moeurs*:
jeunes et vieux chanter en dansant, de maniere que les plus âgés répondent aux enfans qui les provoquent par leurs chansons, je me rappelois ces choeurs de Lacédémone , où, suivant la traduction d'Amiot , dont vous aimez tant la naïveté, les vieillards chantoient: nous avons été jadis jeunes, vaillans, et hardis. À quoi les jeunes répondoient: nous le sommes maintenant, à l'épreuve à tout venant . Et les enfans, pour n'être pas en reste, ajoûtoient: et nous un jour le serons qui tous vous surpasserons. Lorsque j'entends une jeune grecque se plaindre de ce qu'elle ne peut pas aller danser avec ses compagnes, je crois entendre

(a)



(b) (c) (d) (e)

Figure 3: *(a) The two sample texts being compared. The red text was detected by PhiloLine [ART09] as alignment. The blue text can be considered as also a match, but was missed by the automated detection system. (b) Word matching without function words. (c) Word matching with function words, where the lighter shade indicates the function words in the sample texts. (d) Word matching with a horizontal fill, where the gray boxes indicate possible text deletion and insertion. (e) Word matching with horizontal fill where short line segments have been removed.*

ends of each line segment in search for a possible connection. The search region is set differently for *Diagonal Fill*, *Horizontal Fill*, and *Vertical Fill*. Fig. 3(d) shows the successful connections of the three segments in this example. A further image processing tool is then applied to remove short line segments. The final result is shown in Fig. 3(e).

Fig. 4 illustrates an example that PhiloLine [ART09] found difficult to process. Fig. 4(b) shows the dot plot resulted from applying a trigram alignment method, which is the top tool in the middle column in Fig. 2(v) with an opened attribute panel in Fig. 2(vi). We can observe that there is a diagonal line pattern among many short line segments. One feature that may be used to differentiate these false positives (i.e., short line segments) and the true positive is that the true positive in this case contain a few relatively long words. We thus introduce a second pipeline as shown in the left column in Fig. 2(v).

This second pipeline first identifies all unigram matches, and then applies a language processing tool *Word Length* to select all those matches with 6 or more letters in a word. As shown in Fig. 4(c), there are only a few dots along the potential commonplace. The pipeline applies five repeated steps of a morphological operator dilation to these dots, resulting in Fig. 4(d). In ViTA, this is achieved by using a *Diagonal Grow* tool.

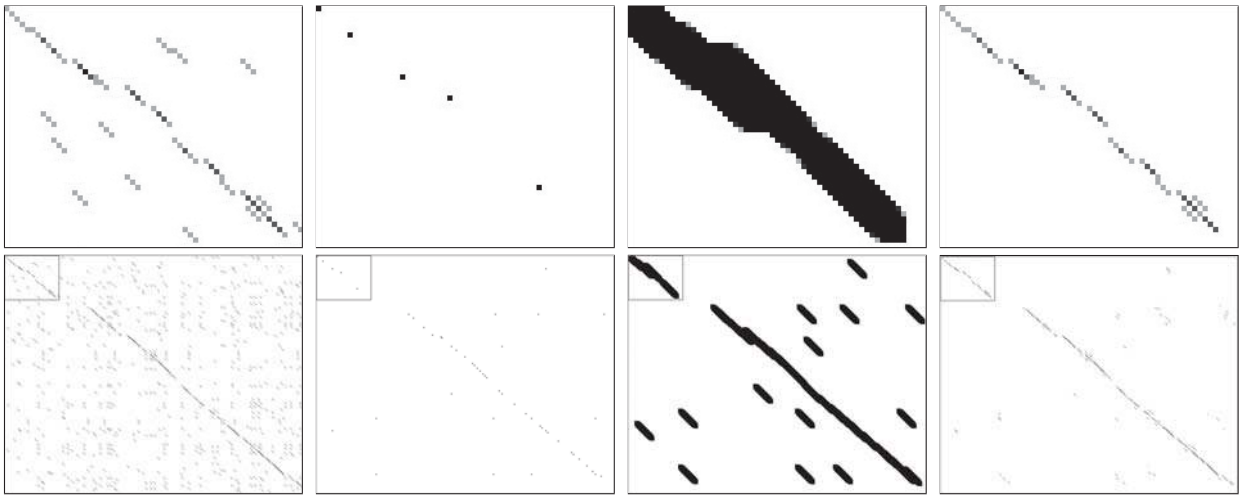
After the two pipelines are combined using an *Intersection* operator, the result shown in Fig. 4(e) is a diagonal line pattern reflecting the true positive. We can observe that there are minor modifications to the source text as well as a major deletion. The constructive method confirms the alignment where minor modifications were made, while removing many false positives in this case. Similar to Figs. 3(d-e), further image processing tools can be used to make the connections and remove the remaining noise.

The above examples serve to demonstrate that from a domain-specific perspective – centered on the literary and print culture of the 18th-century – and considering the most common presumptions of standard text-mining practices, the combination of image processing tools and methods with existing sequence alignment approaches for constructive text alignment can yield interesting, and previously unforeseen, results. The use of darkening for example, based on thresholds that determine what constitutes a match using a sliding scale of 0 to 1 allows users to highlight possible matches in otherwise unremarkable sequences. Extending match parameters beyond normal constraints and conditions, such as the darkening of potentially matching heptagrams (which are very rare, and would normally not be considered), allows us to leverage our capacity to extend patterns visually beyond the strict parameters of the sequence alignment algorithm. This capacity also allows us to refine these same matching parameters and algorithms iteratively using the ViTA interface, moving from obvious to less-obvious matches and building a set of rule-based pipelines for different text mining tasks. This ability to treat multiple data formats and quality is all the more relevant given the mixed nature of the various datasets from which we are drawing our texts. For example, the construction of a stop word or high-frequency list for exclusion at the pre-processing stage is standard practice

Source: Hume:
Quoique ie Chriftiainime dans » fa pureté pri'nitive, ne foit pas défavora- , ble à la Société, cependant on peut quel- :, quefois abiifer des meilleures inflitutions, n, & il ne Ceroit peut-être pas aifi de jufli- > fier tous les Editsde; Empereurs Ch,é- , tiens à ce fujet; ce ql'il y a ds fir. c'eft a) que l'on peut rezard,.r le nombre proJi- » gieux de Prêtres non mariés dans tous l.'s >l' Pays Catholiqurs, qui font une fi grnde > partie de l'Europe, & celui des perofn- , nes du Sexe, qui da's les Couvens font s Vceu de virginité, comme une des prin- cipales caiifcs de la difette de Peuple, ' dans les Pays qui font fous la domination niais il eft affez vrai-femblable qu'an- *, du Souverain Pontife. «c On ne doit pas être furpris que des Auteurs Proteflans tien- nent ce langage, lorfque les Ecriv. ins Ca- tholiques les plus judicieux & les plus atta- chés à la Religion ne peuvent s'empécher de former les mêmes plaintes. Si lEipagne, autrefois fi peuplée, efl aujourd'hui dérerte, c'eft fur-tout au trop grand nombre de Couvens, qu'il faut s'en prendre felon les Auteurs Efpagnols. ,j Je , laiffe, dit le célébre DoN DIEGO DE D, SAAVEDRA, dans fon En:blme LX Vl. . , ceux dont c'ef le devoir à examiner fi le , nombre exceffif des Eccléfiaftiques & des » Couvens, eft proportionné aux Facultés » de la Société des Laïques qui doit les en- s, tretenir, & s'il n'eft pas contraire aux ,t vues mêmes de l'Eglife. Le Confeil de » Caflille dans le projet de Réforme qui fut :; préfenté à Philippe III. en 1619. fupplie » le Roi d'obtenir du Pape, qu'il mette des),bornes à ce nombre exceffif de Reli- , gieux, d'Ordres & de Couvens, qui s'ac- » croît tous les jours; & de lui repréfente

Target: ENC Monastere:
Quoique le Christianisme dans sa pureté primitive ne soit pas défavorable à la société, on abuse des meilleures institutions; & il ne seroit peut - être pas aisé de justifier tous les édits des empereurs chrétiens à ce sujet. Ce qu'il y a de sûr, c'est qu'on regarde la quantité de moines, & celle des personnes du sexe qui dans les couvens font voeu de virginité, comme une des principales causes de la disette de peuple dans tous les lieux soumis à la domination du souverain pontife. On ne doit pas être surpris que des auteurs protestans tiennent ce langage, lorsque les écrivains catholiques les plus judicieux & les plus attachés à la religion, ne peuvent s'empêcher de former les mêmes plaintes.
Si l'Espagne, autrefois si peuplée, est aujourd'hui deserte, c'est sur - tout à la quantité de monasteres qu'il faut s'en prendre, selon les auteurs espagnols. « Je laisse, dit le célebre dom Diego de Saavedra dans un de ses emblèmes, à ceux dont le devoir est d'examiner si le nombre excessif des ecclésiastiques & des monasteres est proportionné aux facultés de la société des laïques qui doit les entretenir, & s'il n'est pas contraire aux vûes mêmes de l'Eglise. Le conseil de Castille, dans le projet de réforme qui fut présenté à Philippe III. en 1619, supplie le roi d'obtenir du pape qu'il mette des bornes à ce nombre prodigieux d'ordres & de monasteres qui s'accroît tous les jours, & de lui représenter les inconvéniens qui en résultent.

(a)



(b) (c) (d) (e)

Figure 4: *(a) The two sample texts being compared. The blue text can be considered as an alignment, but was missed by the automated detection system [ART09]. (b) Word matching with a trigram alignment (above: detailed view and below: full matrix view). (c) Word matching with a unigram alignment followed by a selection of words containing 6 or more letters and the darkening of input values* ≥ 0.4*. (d) Grow the alignment pattern in (c) diagonally to extend the influence of the matched long words. (e) The results of combining the two pipelines using the intersection operator.*

for many text-mining tasks, but these exclusions can have unintended results when applied to very noisy data. The assumptions for one sort of data are thus not always applicable for other data sources, and the "one-size-fits-all" model of sequence alignment is not sufficiently robust to include the level of flexibility that humanities scholars come to expect. Our aim in the project was thus to be able to visualize both sorts of data, i.e., texts drawn from highly curated and corrected collections such as the ARTFL-FRANTEXT database (Fig. 3), as well as those that are the result of uncorrected OCR processes and therefore contain an unknown, and often significant, amount of textual inaccuracy (Fig. 4).

Thus, in order to leverage the scale of the digital collections we are targeting in this project, text alignments must be able to be made not only based on exact word or *n*-gram matches, but also with a high degree of match flexibility. This flexibility is essential from a humanities perspective: some matches will include deletions, modifications, or insertions (Fig. 3) that are significant for literary scholars to identify and analyze; other matches will be difficult to find given the noisy information space of uncorrected OCR collections (Fig. 4), and thus require more expansive matching criteria than what is normally required in the generic text alignment. ViTA thus addressed both of these concerns by way of its constructive visual analytic approach to text alignment and matching algorithms.

In February 2015, before the release of ViTA (*beta*) online, we organized a small workshop, and invited eight literary scholars to test the software. None of these scholars was involved in the design and early testing of the software. None of them had used visual or non-visual software tools for text alignment. The workshop consisted of a 30-minute introduc-

tion, 60 minutes of hands-on experience, and 30 minutes of discussion and feedback. It did not take long for the scholars to understand the two visual representations and common patterns shown in the dot plots. The scholars found the *Method Editor* easy to use, and would like more online help about the functions and parameters of each tool. Scholars worked on a number of pieces of texts selected by them just before the workshop, including 19 pieces in French from the *Questions sur l'Encyclopédie* [ARTc] and 201 short passages in English from *Night and Day* [Woo19]. The known commonplaces were discovered, stimulating discussions among scholars. The workshop also helped us to identify a number of aspects for future improvement, including the speed of the web-server for dealing with texts over 1000 words, online help, and small-screen devices. The scholars considered that making such a software system available to them is "fantastic" and "amazing". They were pleasantly surprised that they were actually constructing programs shortly after the 30-minute introduction.

ViTA (*beta*) was released online in March 2015 and is available at `www.ovii.org/vita`.

8 Conclusion

In this paper, we have presented a visual analytics approach to help improve the existing text alignment methods that rely on analytical models. This approach brings back human analytical capabilities through the use of visualization of text alignment pipelines in conjunction with testing inputs and outputs, and the use of interaction for constructing and improving such pipelines. As the research was embedded in an application directly, namely, the context of 18th-century print culture, this approach was developed in an interdisciplinary manner, and was evaluated in intensive meetings at the design stage as well as after prototyping. The evaluation confirmed that the domain experts can quickly accustom themselves to the pixel-based visualization that appears to be less intuitive than bipartite graphs. More importantly, they can use the visual interface to construct and modify text alignment pipelines. During the testing of the prototype software, domain experts discovered several interesting phenomena, which stimulated new hypotheses and further research and development activities.

Although this work was conducted in a specific application context, the capability of visual analytics in model visualization and development demonstrated in this work can easily be applied to many other applications that rely solely on analytical models. We will continue to explore further applications of this visual analytics approach. We are also making further improvement to the online help and the scalability of dot plots.

Acknowledgments

This work was funded jointly by NEH (USA) and JISC (UK) under the *Digging into Data Challenge* Program (III). The authors would like thank Professor I. Foster (University of Chicago) for his organizational support to this project, A. Himpson (University of Oxford) for his technical assistance in recording the workshop and video editing, and E. Maguire (CERN) for designing the ViTA logo. The authors would also like to thank the literature scholars who participated in the workshop. They are A. Breathe, P. Cotsapas, L. Ludtke, T. Nurmikko-Fuller, G. Pink, K. Rubin-Detlev, R. Sciuto, and C. Warman (University of Oxford).

References

[Aik] AIKEN A.: MOSS: A system for detecting software plagiarism. http://theory.stanford.edu/~aiken/moss/. (3 Dec 2014). 2

[All10] ALLAN D.: *Commonplace Books and Reading in Georgian England*. Cambridge University Press, 2010. 3

[ARLC*13] ABDUL-RAHMAN A., LEIN J., COLES K., MAGUIRE E., MEYER M., WYNNE M., JOHNSON C., TREFETHEN A., CHEN M.: Rule-based visual mappings – with a case study on poetry visualization. *Computer Graphics Forum 32*, 3 (2013), 381–390. 2

[ARTa] ARTFL PROJECT: ARTFL-FRANTEXT. https://artfl-project.uchicago.edu/content/artfl-frantext. 3, 8

[ARTb] ARTFL PROJECT: ECCO-TCP. https://artfl-project.uchicago.edu/content/ecco-tcp. 3, 8

[ARTc] ARTFL PROJECT: TOUT VOLTAIRE. http://artfl-project.uchicago.edu/tout-voltaire. 11

[ART09] ARTFL PROJECT: PhiloLine. https://code.google.com/p/text-pair/, Mar 2009. 8, 9, 10

[BCMB12] BÜCHLER M., CRANE G., MORITZ M., BABEU A.: Increasing recall for text re-use in historical documents to support research in the humanities. In *Proc. 2nd Int. Conf. Theory & Practice of Digital Libraries* (2012), pp. 95–100. 1

[BD94] BRAGG S., DRISKILL C.: Diagrammatic-graphical programming languages and dod-std-2167a. In *IEEE Systems Readiness Technology Conf. 'Cost Effective Support Into the Next Century'* (Sep 1994), pp. 211–220. 6

[BG07] BOURDAILLET J., GANASCIA J.: Alignment of noisy unstructured text data. *IJCAI Workshop on Analytics for Noisy Unstructured Text Data* (Jan 2007). 1, 2

[Bla11] BLAIR A.: *Too Much to Know: Managing Scholarly Information before the Modern Age*. Yale University Press, New Haven, 2011. 3

[BTH*13] BOSCH H., THOM D., HEIMERL F., PUTTMANN E., KOCH S., KRUGER R., WORNER M., ERTL T.: Scatterblogs2: Real-time monitoring of microblog messages through user-guided filtering. *IEEE Tran. Visualization & Comp. Graphics 19*, 12 (2013), 2022–2031. 2

[CCP09] COLLINS C., CARPENDALE M. S. T., PENN G.: Docuburst: Visualizing document content using language structure. *Computer Graphic Forum 28*, 3 (2009), 1039–1046. 2

[Cen] CENGAGE LEARNING: Eighteenth Century Collections Online (ECCO). http://gale.cengage.co.uk/product-highlights/history/eighteenth-century-collections-online.aspx. 3

[CGPW97] CLOUGH P., GAIZAUSKAS R., PIAO S. S. L., WILKS Y.: METER: MEasuring TExt Reuse. In *Proc. 40th Anniversary Meeting for the Asso. for Computational Linguistics* (1997), pp. 152–159. 1, 2

[CH93] CHURCH K. W., HELFMAN J. I.: Dotplot: A program for exploring self-similarity in millions of lines of text and code. *J. Computational & Graphical Statistics 2*, 2 (1993), 153–174. 2

[Clo03] CLOUGH P.: Old and new challenges in automatic plagiarism detection. In *National UK Plagiarism Advisory Service* (2003), pp. 391–407. 2

[CT00] CHEN M., TUCKER J. V.: Constructive volume geometry. *Computer Graphics Forum 19*, 4 (2000), 281–293. 6

[CWG11] CORRELL M., WITMORE M., GLEICHER M.: Exploring collections of tagged text for literary scholarship. *Computer Graphics Forum 30*, 3 (2011), 731–740. 2

[DNKS10] DIAKOPOULOS N., NAAMAN M., KIVRAN-SWAINE F.: Diamonds in the rough: Social media visual analytics for journalistic inquiry. In *IEEE VAST* (Oct 2010), pp. 115–122. 1

[DWS*12] DOU W., WANG X., SKAU D., RIBARSKY W., ZHOU M.: Leadline: Interactive visual analysis of text data through event identification and exploration. In *IEEE VAST* (Oct 2012), pp. 93–102. 1

[EMR13] EDELSTEIN D., MORRISSEY R., ROE G.: To Quote or not to Quote: Citation Strategies in the Encyclopédie. *Journal of the History of Ideas 74*, 2 (Apr 2013), 213 – 236. 3

[FEC05] FORMAN G., ESHGHI K., CHIOCCHETTI S.: Finding similar files in large document repositories. In *Proc. 11th ACM SIGKDD Int. Conf. Knowledge Discovery in Data Mining* (2005), pp. 394–400. 1

[Fei] FEINBERG J.: Wordle. http://www.wordle.net/. (4 Dec 2014). 2

[Fel98] FELLBAUM C. (Ed.): *WordNet: An Electronic Lexical Database*. MIT Press, Cambridge, MA, 1998. 2

[Fre08] FREIRE M.: Visualizing program similarity in the ac plagiarism detection system. In *Proc. Working Conf. AVI* (2008), pp. 404–407. 2

[GM70] GIBBS A. J., MCINTYRE G. A.: The diagram, a method for comparing sequences. *European Journal of Biochemistry 16*, 1 (1970), 1–11. 1, 2

[GT99] GITCHELL D., TRAN N.: Sim: A utility for detecting similarity in computer programs. In *Proc. 13th SIGCSE Tech. Symp. Computer Science Educ.* (1999), pp. 266–270. 2

[GW08] GONZALEZ R. C., WOODS R. E.: *Digital Image Processing*, 3rd ed. Pearson, 2008. 4

[HHB07] HENLEY M., HAGEN M., BERGERON R. D.: Evaluating two visualization techniques for genome comparison. In *Proc. Information Visualization* (2007), pp. 551–558. 2, 4

[HHN00] HAVRE S., HETZLER B., NOWELL L.: ThemeRiver: Visualizing theme changes over time. In *Proc. IEEE Symp. Information Visualization* (2000), pp. 115–123. 2

[HKBE12] HEIMERL F., KOCH S., BOSCH H., ERTL T.: Visual classifier training for text document retrieval. *IEEE Trans. Visualization & Comp. Graphics 18*, 12 (2012), 2839–2848. 1

[Hol06] HOLTEN D.: Hierarchical edge bundles: Visualization of adjacency relations in hierarchical data. *IEEE Trans. Visualization & Comp. Graphics 12*, 5 (Sept. 2006), 741–748. 2

[HOR11] HORTON R., OLSEN M., ROE G.: Something borrowed: Sequence alignment and the identification of similar passages in large text collections. *Digital Studies / Le champ numérique 2*, 1 (2011). 1, 2

[JGBS14] JÄNICKE S., GESSNER A., BÜCHLER M., SCHEUERMANN G.: Visualizations for text re-use. In *IVAPP* (2014), pp. 59–70. 2

[JKM12] JANKOWSKA M., KESELJ V., MILIOS E.: Relative n-gram signatures: Document visualization at the level of character n-grams. In *IEEE VAST* (Oct 2012), pp. 103–112. 1, 2

[JL99] JOY M., LUCK M.: Plagiarism in programming assignments. *IEEE Trans. on Educ. 42*, 2 (May 1999), 129–133. 2

[KJW*14] KOCH S., JOHN M., WORNER M., MULLER A., ERTL T.: VarifocalReader - In-depth visual analysis of large text documents. *IEEE Trans. Visualization & Comp. Graphics 20*, 12 (Dec 2014), 1723–1732. 2

[KSC*08] KOOP D., SCHEIDEGGER C. E., CALLAHAN S. P., FREIRE J., SILVA C. T.: VisComplete: Automating suggestions for visualization pipelines. *IEEE Trans. Visualization & Comp. Graphics 14*, 6 (Nov. 2008), 1691–1698. 2

[Lee07] LEE J.: A computational model of text reuse in ancient literary texts. In *Proc. 45th Anniversary Meeting for the Asso. for Computational Linguistics* (2007), pp. 472–479. 2

[LMD01] LYON C., MALCOLM J., DICKERSON B.: Detecting short passages of similar text in large document collections. In *Proc. Conf. Empirical Methods in Natural Language Processing* (2001), pp. 118–125. 1, 2

[Mou04] MOUNT D. W.: *Bioinformatics: Sequence and Genome Analysis*, 2nd ed. Cold Spring Harbor Lab. Press, 2004. 1, 2

[NCD*10] NIELSEN C. B., CANTOR M., DUBCHAK I., GORDON D., WANG T.: Visualizing genomes: Techniques and challenges. *Nature Methods 7*, 3 Suppl (Mar 2010), S5–S15. 2

[OKK13] OELKE D., KOKKINAKIS D., KEIM D. A.: Fingerprint Matrices: Uncovering the dynamics of social networks in prose literature. *Computer Graphics Forum 32*, 3 (2013), 371–380. 2

[OSSK10] OELKE D., SPRETKE D., STOFFEL A., KEIM D.: Visual readability analysis: How to make your writings easier to read. In *IEEE VAST* (Oct 2010), pp. 123–130. 1, 2

[RA00] RIBLER R. L., ABRAMS M.: Using visualization to detect plagiarism in computer science classes. In *Proc. IEEE Symp. Information Vizualization* (2000), pp. 173–178. 2

[RPSF15] RIEHMANN P., POTTHAST M., STEIN B., FROEHLICH B.: Visual assessment of alleged plagiarism cases. *Computer Graphics Forum 34*, 3 (2015). 1

[SS95] SINGHAL A., SALTON G.: Automatic text browsing using vector space model. In *Proc. Dual-Use Technologies and Applications Conference* (1995), pp. 318–324. 1

[The] THESAURUS:. http://thesaurus.altervista.org/. (4 Dec 2014). 5, 6

[Tur] TURNITIN:. http://turnitin.com/. (4 Dec 2014). 2

[VCPK09] VUILLEMOT R., CLEMENT T., PLAISANT C., KUMAR A.: What's being said near "Martha"? Exploring name entities in literary text collections. In *IEEE VAST* (2009), pp. 107–114. 2

[vHWV09] VAN HAM F., WATTENBERG M., VIEGAS F. B.: Mapping text with phrase nets. *IEEE Trans. Visualization & Comp. Graphics 15*, 6 (Nov. 2009), 1169–1176. 2

[WJ04] WHITE D. R., JOY M. S.: Sentence-based natural language plagiarism detection. *J. on Educ. Res. in Computing 4*, 4 (Dec. 2004). 2

[Woo19] WOOLF V.: *Night and Day*. Duckworth, 1919. 11

[Wri] WRITECHECK:. http://en.writecheck.com/. (4 Dec 2014). 2