DHQ: Digital Humanities Quarterly
Preview
2022
Volume 16 Number 1

# Tesserae Intertext Service

Nozomu Okuda <nozomuok_at_buffalo_dot_edu>, SUNY University at Buffalo, Department of Classics
Jeffery Kinnison <jkinniso_at_nd_dot_edu>, University of Notre Dame, Department of Computer Science and Engineering

Patrick Burns <Patrick_dot_Burns_at_austin_dot_utexas_dot_edu>, University of Texas at Austin, Department of Classics

Neil Coffee <ncoffee_at_buffalo_dot_edu>, SUNY University at Buffalo, Department of Classics
Walter Scheirer <walter_dot_scheirer_at_nd_dot_edu>, University of Notre Dame, Department of Computer Science and Engineering

## Abstract

The Tesserae Intertext Service Application Programming Interface (TIS-API) enhances the machine-accessibility of the intertext discovery capabilities of the Tesserae software. Instead of requiring inputs through a human-accessible webpage, the TIS-API accepts inputs according to a web development standard. Two case studies demonstrate the contributions of the TIS-API to computer-assisted literary criticism, particularly in increased software development and maintenance flexibility as well as in easier integration of Tesserae software into research workflows. Those interested in integrating the TIS-API into their digital projects can find documentation at https://tesserae.caset.buffalo.edu/docs/api/. For exact implementation details, the source code is available at https://github.com/tesserae/apitess.

## Introduction

The term “intertext” is used to name a passage of text that gives evidence of one author using another author’s words. Scholarly consensus indicates that intertexts exist ([Juvan 2008] [Allen 2011] for Latin literature, see [Edmunds 2001] [Coffee 2012a] for Greek literature, see [Berti 2016]). While there is great variation in the motivation for the use of an intertext, perhaps the reuse signals an author’s erudition or enhances the effect of an expression. For example, when Vergil in his epic *Aeneid* describes the sound of a horse galloping across the fields: 1

> quadrupedante putrem **sonitu quatit ungula** campum (Aeneid 8.596).
>
> The horses’ hooves shake the soft-soiled plain with four-footed thunder.

He is adapting a line from his epic predecessor Ennius: 2

> . . . summo **sonitu quatit ungula** terram. (Ennius Annals 8.264)
>
> The horses’ hooves shake the earth with great thunder.

With this recollection, Vergil is reusing available poetic materials, demonstrating familiarity with his predecessor Ennius, and inviting the reader to compare the parallel passages. 3

Various attempts have been made at computer-assisted intertext discovery ([Lee 2007], [Mastandrea 2011], [Büchler 2013], [Chaudhuri 2015])[1], but none have made a conscious effort to standardize machine-accessibility of the intertext discovery process and results. This paper presents the Tesserae Intertext Service Application Programming Interface 4

(TIS-API), which specifies the rules of machine-accessibility for the intertext discovery tool known as “Tesserae”.

5 We argue that machine-accessibility to Tesserae’s capabilities, as provided by the TIS-API, is an enhancement to the existing Tesserae software. We first demonstrate that the TIS-API does, in fact, make Tesserae’s capabilities machine-accessible by showing how the TIS-API is designed on standard principles in the web development community, and by walking through an example of how the TIS-API can be used to conduct a Tesserae search in a machine-accessible way. Two case studies further explore how the TIS-API enhances the Tesserae software. The first case study demonstrates how the TIS-API aids collaboration among members of the Tesserae project team, particularly in the implementation of the new Tesserae website interface. The second case study demonstrates how someone outside of the Tesserae project team sees value in the TIS-API for making the software easier to integrate into research workflows that aid reproducibility in literary criticism research.

## Tesserae

6 The Tesserae Project develops software for discovering intertexts, with particular focus on the intertexts that appear in ancient Greek and Latin literature [Coffee 2012b]. The project’s main tool, called Tesserae (https://tesserae.caset.buffalo.edu/), accomplishes this by analyzing and comparing two texts for regions of word reuse.

7 For example, in order to find intertexts between Vergil’s *Aeneid* and Lucan’s Bellum Civile, Tesserae goes through both works line-by-line to find where pairs of words occur in lines from both works. Because the *Aeneid* precedes the Bellum Civile in publication, we can assume that the*Aeneid*served as a source text from which the later poet could draw upon for poetic raw material. Accordingly, the results from this Tesserae search can help us find places where Lucan may have borrowed words from Vergil (see [Coffee 2012b] for examples).

8 Given the highly inflected nature of both ancient Greek and Latin, Tesserae also allows for matching by stem. This allows for finding examples of Lucan reusing Vergil’s words in different grammatical and morphological forms. It is also useful to exclude common words, that is stopwords, from Tesserae searches, on the grounds that the reuse of common words signals not so much Vergil’s influence on Lucan as much as the common language available to the poets in composing their respective works. In summary, when Tesserae is given (1) two texts to compare, (2) a manner in which to compare the words of the texts (i.e., by exact word match or by stem), and (3) a list of stopwords to ignore, instances of text reuse can be found, which in turn can serve as evidence of intertexts.

9 Originally conceived of as a command line tool, the advent of the Tesserae website interface in 2009 (https://tesseraev3.caset.buffalo.edu/) made the delivery of its intertext discovery capabilities much more human-accessible [Coffee 2012b]. However, this same interface made machine-accessibility difficult for two main reasons. First, the submission of Tesserae searches relied on user input via web forms. Second, the results of the Tesserae searches were returned as an HTML webpage. While these two reasons did not make the Tesserae results impossible to access automatically, they did present difficulties with the automatic submission of search queries and the parsing of results because there were no guarantees that the input forms or the results webpage would remain unchanged. Any tools built on the Tesserae site looking and performing in exactly one way would most likely fail in the face of website updates that changed the input forms or the layout of the search results. The lack of a machine-accessible design created a situation in which Tesserae could be difficult to integrate with other digital projects, short of installing and hosting a standalone instance of Tesserae.

10 The TIS-API eases the burden of integrating Tesserae’s intertext discovery capabilities into other projects by permitting queries to Tesserae’s database of texts, submission of Tesserae search requests, and retrieval of search results — all in a machine-accessible way. The TIS-API was designed, developed, and deployed at the SUNY University at Buffalo, in collaboration with developers at the University of Notre Dame, where backend libraries on which the TIS-API relies were developed. The University of Notre Dame also developed a web frontend powered by the TIS-API, demonstrating one instance in which the TIS-API eased the burden of integrating Tesserae’s intertext discovery capabilities into new software. As an important perspective from outside of the Tesserae Project, a researcher at the Quantitative Criticism Lab (University of Texas at Austin) has tested and commented on the TIS-API.

## TIS-API Design

Following standard practice for machine-accessible communication across the Internet, the TIS-API was designed according to the five REpresentational State Transfer (REST) principles [Fielding 2000], [Fielding 2017]. Since the basic rules for communication over the Internet are already defined by the HyperText Transfer Protocol (HTTP), the five REST principles provide further guidelines on how to use HTTP for effective machine-to-machine communication. 11

The fundamental assumption of the five REST principles is that the only thing a machine can request from another machine is a resource (Figure 1). 12

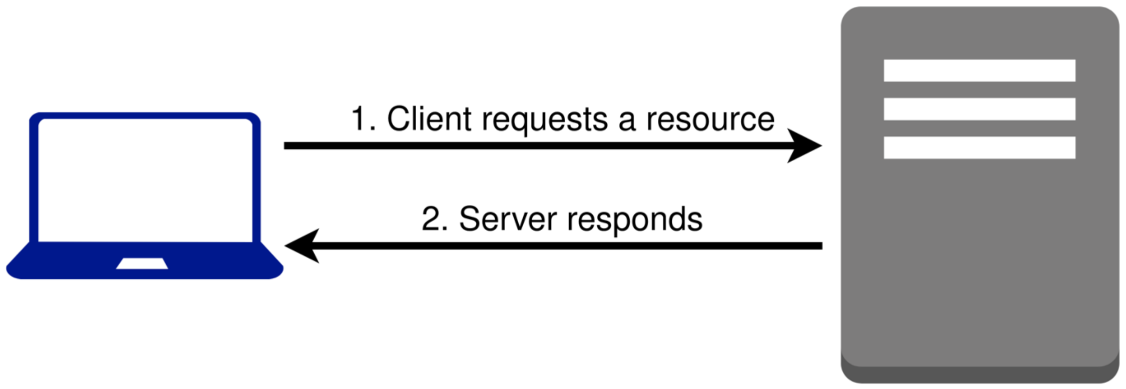


**Figure 1.** Diagram of the fundamental communication pattern in REST. The machine on the left, known as the “client,” makes a request for a resource. The machine on the right, called the “server”, responds to the client’s request. The language used between the computers is known as HTTP.

In the original vision of the five REST principles, a resource typically took the form of a webpage or some multimedia asset, like an image. For Tesserae’s intertextual discoveries, one resource would be the discoveries themselves — the results from a Tesserae search. Other resources, such as text information and search options, aid in the production of those discoveries. 13

Starting from these basic assumptions about resources, the first REST principle is to have a name for every resource (Figure 2). 14

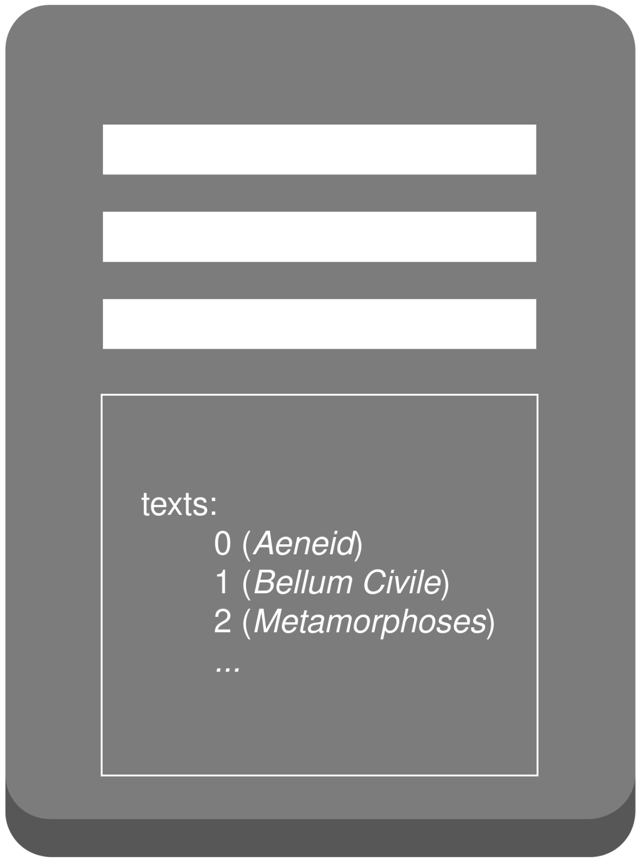


**Figure 2.** First REST principle: every resource that a server makes available must have a name. One resource that the TIS-API makes available is the catalog of texts that Tesserae has access to, which can be accessed by the name "/texts/". Furthermore, metadata for each text in the catalog can also be requested. Therefore, the TIS-API assigns a name to each resource. In the example figure, the name "0" is associated with *Aeneid*, the name "1" is associated with *Bellum Civile*, and so forth. Note that these example names are not the ones used in the actual deployment currently running through the Tesserae project.

This allows one machine to make a request on a specific resource from another machine. For example, the name http://www.buffalo.edu/ refers to the resource that is the main web page for the University at Buffalo. The TIS-API specifies naming conventions for the resources necessary to carry out Tesserae searches. For example, the naming conventions require that "/texts/" be part of the name for resources referring to texts available in a Tesserae database. More examples of names and the TIS-API naming conventions will be shown later in the paper. 15

The second REST principle is that the way one specific request behaves for one resource is similar to the behavior of the same request on a different resource (Figure 3). 16

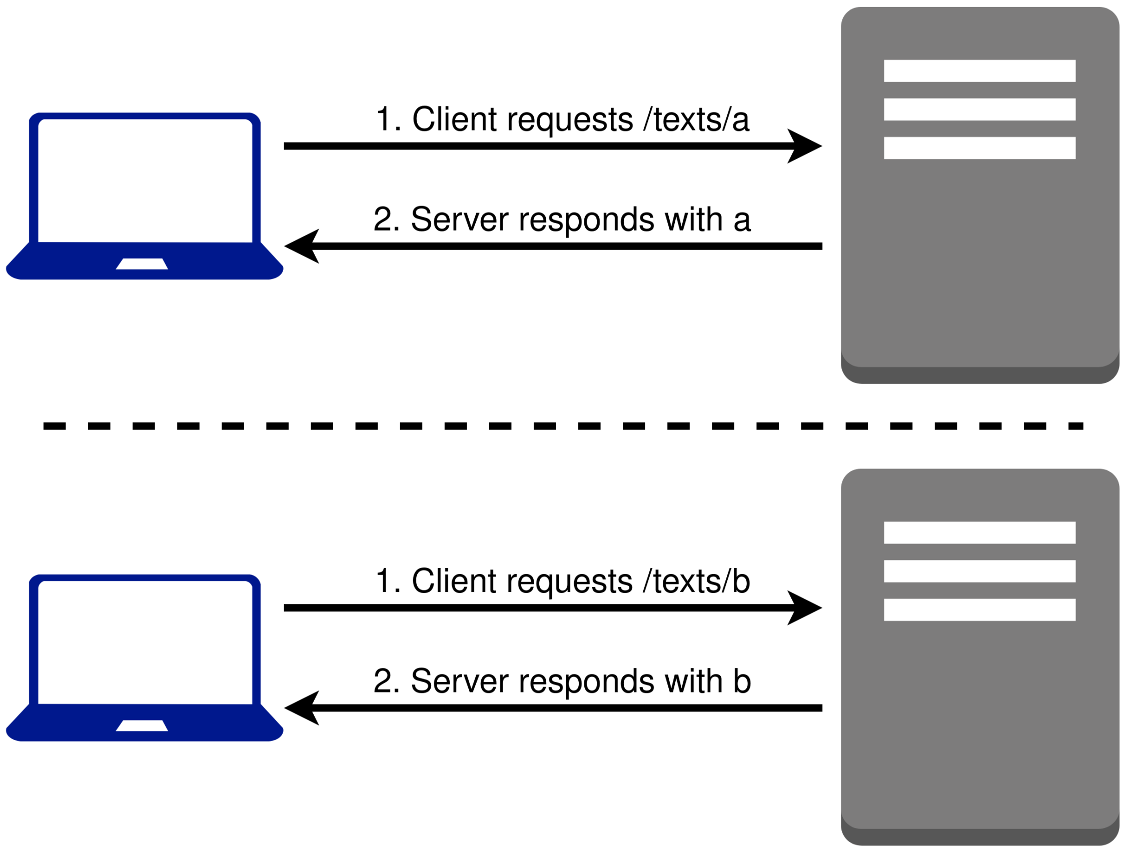

1. Client requests /texts/a
2. Server responds with a
1. Client requests /texts/b
2. Server responds with b

**Figure 3.** Second REST principle: requests should behave similarly across resources. In this example, the client has requested a resource named "/texts/a". In response, the server provides information on "a". When the client requests for a resource named "/texts/b", the server responds in a similar fashion, except that it provides information on "b" instead of on "a".

Within the rules of HTTP, there are various requests that can be made. One common request is GET, which requests retrieval of the resource. Going back to the University at Buffalo main web page example, a computer requesting a GET for http://www.buffalo.edu/ is asking another computer to retrieve the University at Buffalo main webpage. The TIS-API specifies that requesting a GET for "<webhost>/texts/" (where "<webhost>" is replaced with the URL referring to a server on which the TIS-API is installed) will retrieve information about the texts available in a Tesserae database. 17

The third REST principle is that modification of a resource is mediated through instructions, which are called representations (Figure 4). 18

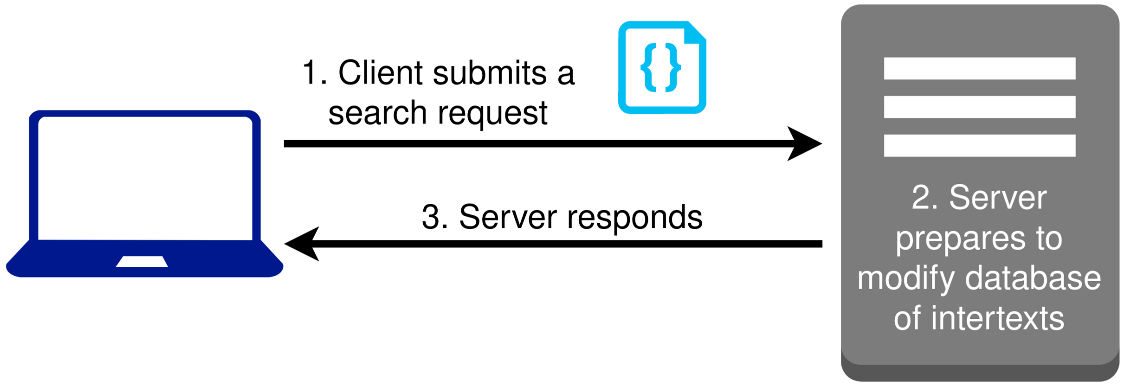

1. Client submits a search request
3. Server responds
2. Server prepares to modify database of intertexts

**Figure 4.** Third REST principle: modifications to a resource are mediated through instructions, known as a representation. When a client requests that a Tesserae search be run, the TIS-API modifies the "parallels" resource by appending newly found intertexts. What intertexts should be added to "parallels" is determined by the representation that the client sent (shown as the blue icon), which contains instructions specifying how the search is to be run.

For the TIS-API, this principle is highlighted particularly in the way Tesserae searches are submitted. To submit a Tesserae search, a machine must request a POST (like GET, POST is one of the requests that can be made through HTTP) on the "<webhost>/parallels/" resource and provide Tesserae search options in a particular format. The particularly formatted search options would be a representation to which the third REST principle refers. 19

The fourth REST principle is that representations are self-descriptive. Admittedly, the TIS-API does not conform to this principle directly. However, one way in which the TIS-API adheres to the spirit of this principle is in the search results resource. Search results resources are defined to contain the search options used to produce the results available in the resource. In this way, it would be possible to know exactly what options produced these results (Figure 5). This also enables caching of results, which can lead to a better user experience in the form of reduced search times. 20

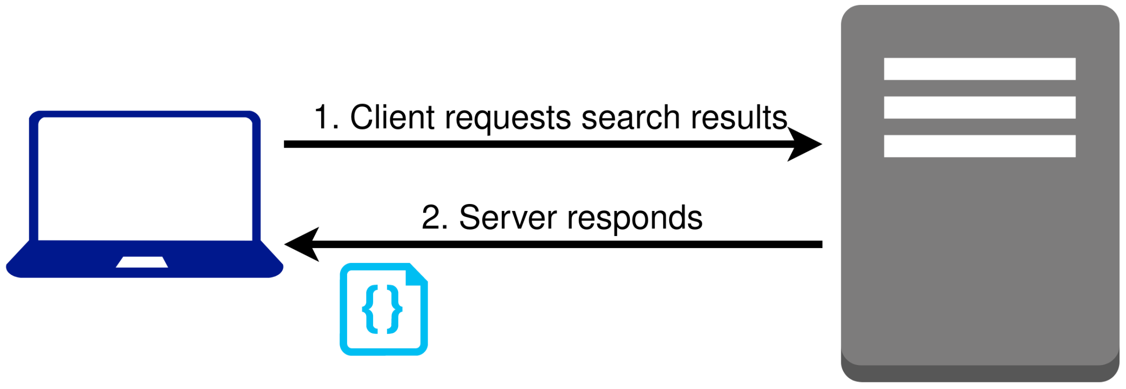


**Figure 5.** Fourth REST principle: representations should be self-descriptive. Although the TIS-API is not strict in following this principle, one way in which it practices the spirit of this principle is in the server response to a request for search results. The representation used in submitting the current search (shown as the blue icon) is given as part of the server's response.

The fifth and final REST principle is that application state information should not be stored; instead, it should be passed as a representation. To unpack the meaning of this, it is important to consider another assumption about the REST principles: when one computer communicates with another computer, the two are in a server-client relationship. The server is the computer that holds the resource that a client wishes to make a request on. As a result, the client always initiates contact with the server. Application state refers to a step in some process that the server and client navigate together. 21

A common server-client application is online shopping. The first step is for the client to submit a set of items to purchase; then, the server needs to calculate various things (like whether the items are in stock, how much the order will cost, taxes to collect, and other business-related tasks); then, the client needs to provide authorization to pay for the purchase; finally, the server should notify the client that the order has been placed and that the purchase has been billed. According to the fifth REST principle, neither the client nor the server should store information specifying which step they are on. Rather, the exchange of information they conduct as part of that process should specify which step they are on. 22

The TIS-API exhibits compliance with the fifth REST principle in the way it handles Tesserae search submissions (Figure 6). 23

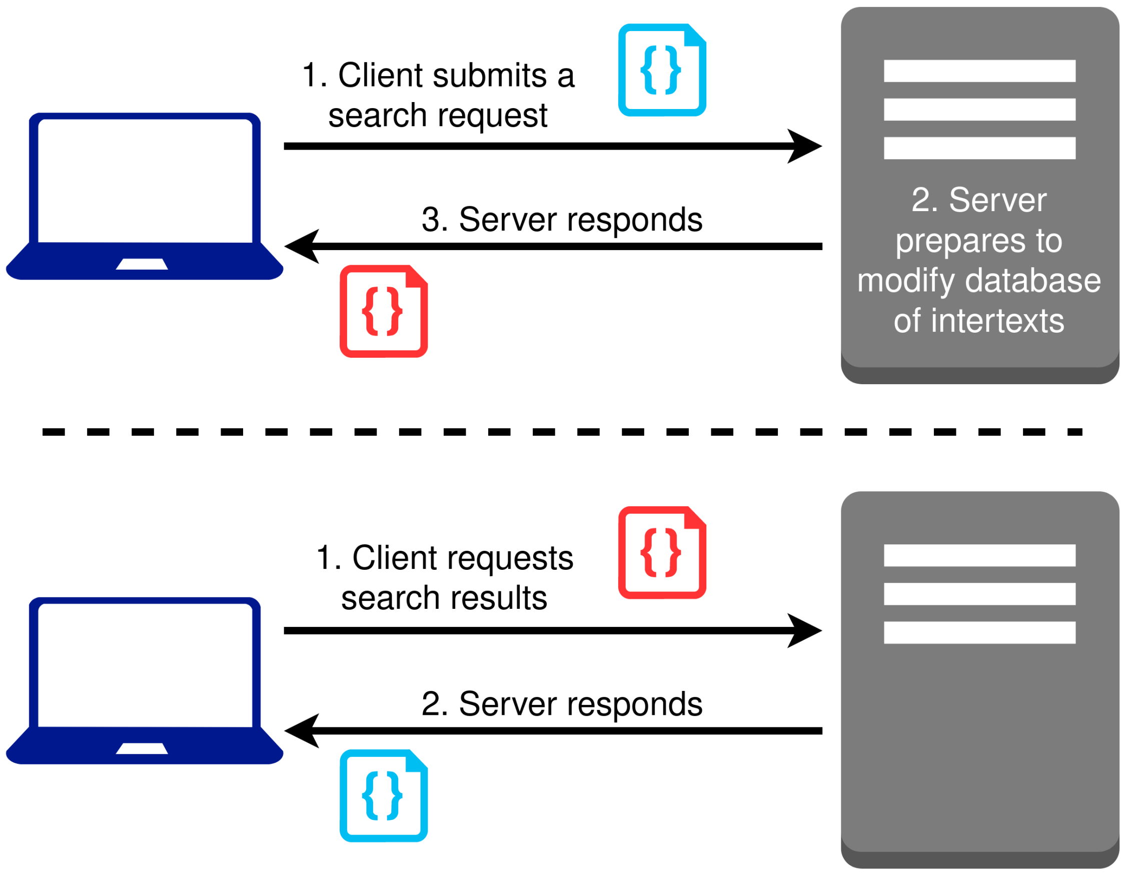

1. Client submits a search request
3. Server responds
2. Server prepares to modify database of intertexts
1. Client requests search results
2. Server responds

**Figure 6.** Fifth REST principle: application state should not be stored in the API. In the example of running a Tesserae search, the application is expected to 1) run a search as specified and 2) display the results. The TIS-API separates these two states of the application into individual requests, thereby avoiding the need to store application state either on the client or the server. Instead, the application state is mediated by representation: the client knows how to request search results based on information the server has given (shown as the red icon). Note that the client is not required to immediately request search results; because it is not tied to an application state, it can make other requests, such as requests for certain texts or for a new stopwords list, before requesting the search results.

For the Tesserae search application, the client first submits a Tesserae search; the server then needs to run the search and store the results; finally, the client should be able to access the results. Since Tesserae search may take longer than a user is willing to wait, it seemed important to design the search application in such a way that the client would not be stuck waiting for the server to respond with search results. Designing the application so that the server would first respond to the client with the resource name for completed search results not only freed the client from waiting on the server but also made storing application state in either the server or the client unnecessary. The client can make a request on the completed search results, and if the server responds that they aren't ready yet, the client can wait to make a request later. 24

The previous points demonstrate that each of the five REST principles were considered in certain aspects of the TIS-API. While these examples do not describe all the ways in which the five REST principles were employed in the design of the TIS-API, these examples should serve to substantiate the claim that the five REST principles were important considerations during the design of the TIS-API. Because of this, the TIS-API provides machine-accessibility to Tesserae's capabilities. 25

## Tesserae Search via the TIS-API

Though claims of machine-accessibility can be made through the REST principles, it can be more convincing to show machine-accessibility by walking through the process of submitting a Tesserae search through the TIS-API (see Figure 7 for a high-level workflow of the process). For those who prefer directly reading the behavior prescribed by the TIS-API instead of going through this example, the TIS-API documentation site is available at 26 https://tesserae.caset.buffalo.edu/docs/api/. For those who want to know exact details about how the TIS-API was implemented, source code is available at https://github.com/tesserae/apitess.

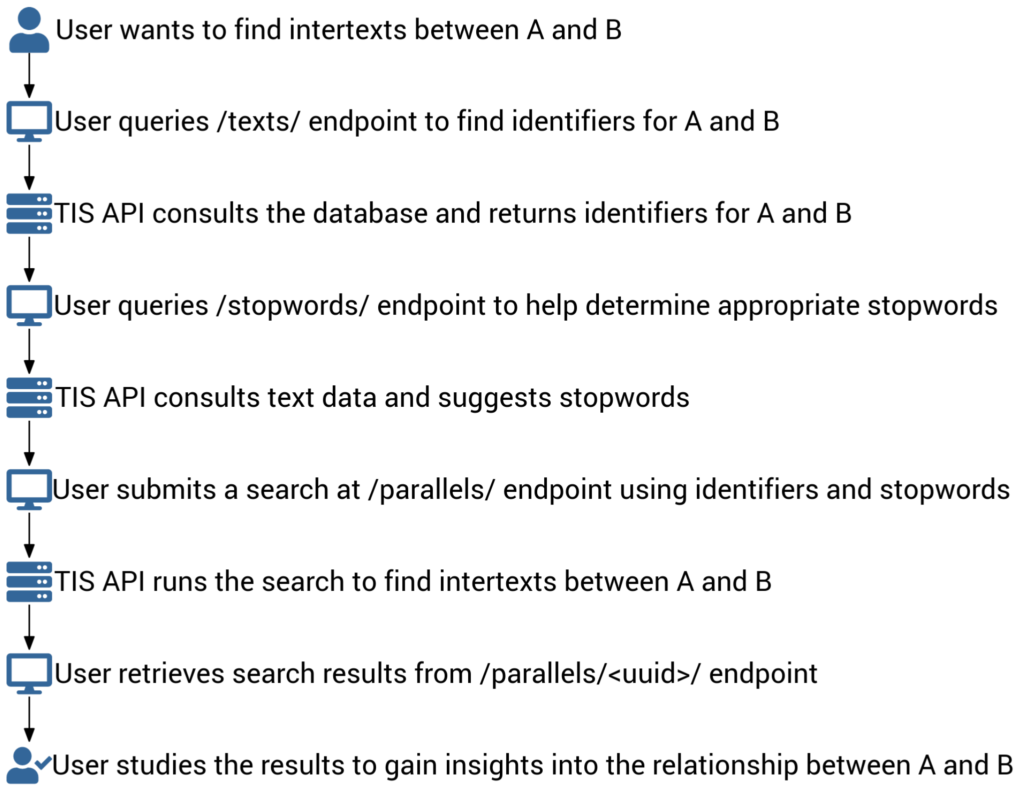

User wants to find intertexts between A and B
User queries /texts/ endpoint to find identifiers for A and B
TIS API consults the database and returns identifiers for A and B
User queries /stopwords/ endpoint to help determine appropriate stopwords
TIS API consults text data and suggests stopwords
User submits a search at /parallels/ endpoint using identifiers and stopwords
TIS API runs the search to find intertexts between A and B
User retrieves search results from /parallels/<uuid>/ endpoint
User studies the results to gain insights into the relationship between A and B

**Figure 7.** A high-level overview of the workflow used to submit a Tesserae intertext search through the TIS-API. This image is also used at https://tesserae.caset.buffalo.edu/docs/api/getting-started/workflow/, where a similar workflow is described.

## Anatomy of a Tesserae Search Query

To start the discussion of how the TIS-API can be used to perform a Tesserae search, the component parts of a Tesserae search query should be reviewed. Although the details for the Tesserae search process have already been described [Forstall 2014], the focus here is to have a simplified but strong conceptual grasp on the parts involved in a Tesserae search. By understanding what the building blocks are for defining a search, it will be easier to understand how each of these blocks are addressed by the TIS-API. The aspects of the Tesserae search input discussed in this paper are: 27

- source text
- target text
- type of feature used to find matches
- stopwords

To frame the discussion of these inputs, it will be helpful to state what the Tesserae search is supposed to do: the objective of Tesserae search is to find intertexts between two works. Thinking of Tesserae search in this way, it becomes immediately obvious that two works need to be specified in the input. Because Tesserae was originally developed as a tool for intertext detection, a relationship between the two works is assumed: namely, that one work influenced the other. For this reason, a Tesserae search query is formulated in terms of a “source text” (i.e., the work that influences the other) and a “target text” (i.e., the work that was influenced by the other). 28

Another aspect fundamental to specifying a Tesserae search query is defining what counts as an intertext. For purposes 29 of the Tesserae search algorithm, an intertext is defined by the matching of at least two features between two passages, where one passage comes from the source text, and the other passage comes from the target text. Thus, it becomes necessary to specify which type of feature to compare when looking for matching features.

One feature type is the lemma type, which can be thought of as a dictionary headword. When looking up the definition of a given word in the dictionary, it is sometimes necessary to change the spelling of the word to find the correct entry. For example, the definition for the word "love" in "I love to swim" is found in the dictionary under the lemma "love". In this example, no spelling change is necessary to find the lemma. However, the definition for the word "loves" from "He loves swimming" is also found under the lemma "love". It is necessary to remove the final "s" from "loves" in order to find the lemma. 30

If lemmata (the plural of lemma) are used to search for intertexts, only passages that share at least two different lemmata will be considered a potential intertext. For example, suppose we have the following passages (again): 31

1. I love to swim
2. He loves swimming

Note that passage 1 contains the words "love" and "swim", both of which also look identical to their lemmata. Passage 2 likewise contains the lemmata "love" (from "loves") and "swim" (from "swimming"). From this information, it can be seen that a Tesserae search searching for intertexts based on the lemma feature would find that these two passages are a potential intertext. 32

A final aspect to consider in this simplified conception of the inputs to a Tesserae search is the matter of stopwords. Often, there are features we believe will be unimportant in determining whether one passage refers to another. For example, we may find common words like articles and prepositions to be irrelevant. In this case, we would like a Tesserae search to ignore passages that match only because of articles and prepositions. To specify features that we think are not interesting, we can use a stopwords list to prevent those features from being counted when determining whether two passages are a potential intertext. 33

Knowing now that a source text, a target text, a feature definition, and a stopwords list are important options to specify in a Tesserae search, it is clear that these options will need to be specified to the TIS-API when submitting a Tesserae search query. 34

## API Concepts

Before discussing how to use the TIS-API to perform Tesserae search, some terminology and conventions should be noted. First, the meaning of "TIS-API" should be more carefully stated. The TIS-API, in its strictest sense, is a set of rules that defines server behavior in response to client requests. An implementation of the TIS-API, therefore, is code that, when run, behaves in accordance with the rules that were defined. When this code is running, we can say that a computer is "running an instance of the TIS-API." Since the code allows a computer to follow the rules of the TIS-API and since the deployed software exhibits the behaviors defined by the rules of the TIS-API, the use of "TIS-API" can slip from its strict sense into a more inclusive (and therefore ambiguous) sense that may denote either the rules, the code, or the deployed software. 35

The angle brackets ("<," ">") convention used in resource names should also be explained explicitly. Just as earlier in the paper, the angle brackets ("<," ">") denote a placeholder; the word(s) within the angle brackets hint at what should replace this placeholder. For example, as seen earlier, "<webhost>/texts/" specifies a name for interacting with text resources. However, the name does not literally contain "<webhost>"; rather, "<webhost>" should be replaced with an identifier to a running instance of the TIS-API. For example, if https://tesserae.caset.buffalo.edu/api is the URL to a running instance of the TIS-API, https://tesserae.caset.buffalo.edu/api/texts/ (note that "/texts/" has been added to the URL) is the name referring to the text resources available in that instance. 36

Finally, the term "endpoint" should be defined. Note that the initial part of the URL is largely irrelevant to the defined behavior for the TIS-API. In other words, whether https://tesserae.caset.buffalo.edu/api or https://www.tis-api.com is 37 used to work with texts, the TIS-API will respond in a similar manner — and it would behave exactly the same if both web hosts had the same text information in their respective databases. Therefore, the part of the name that distinguishes behavior is what comes after the web host’s identifier. These distinguishing patterns that come at the end of the URL are known as “endpoints.” By convention, these endpoints are referred to not by their full URL; rather, they are referred to by the portion of the URL that comes after the web host’s URL. For example, “/texts/” is the TIS-API endpoint that deals with texts.

## Using the TIS-API to Run a Tesserae Search

With a clear picture of Tesserae search options as well as the vocabulary used to express APIs, it is possible to consider how the TIS-API can be used to build and submit a Tesserae search query. The first endpoint to consider is “/parallels/”, which accepts Tesserae search requests. By performing a POST request on the “/parallels/” endpoint and providing the Tesserae search options in a particular machine-readable format, a Tesserae search query can be submitted. Exact details on this particular machine-readable format can be found by consulting the documentation; for purposes of this paper, it is sufficient to know that the “/parallels/” endpoint accepts Tesserae search query submissions. 38

Ignoring the specifics of the Tesserae search query submission format for the “/parallels/” endpoint, earlier discussion has shown that a source text, a target text, a feature to search by, and a stopwords list must be specified. The TIS-API can be useful in choosing these options. 39

Since Tesserae can perform intertext discoveries only between texts already in its database, we can only reference a source text and a target text already in the Tesserae database when submitting a Tesserae search. How does the TIS-API provide access to information about texts in the Tesserae database? As mentioned earlier, text information can be obtained at the “/texts/” endpoint. Again, there are specific details about the “/texts/” endpoint that may be useful (such as querying the Tesserae database for texts written by a particular author) which can be found by consulting the documentation. But for purposes of this paper, it is sufficient to know that the “/texts/” endpoint can be used to discover what texts are available to be used as a source or target text. From the information returned by performing a GET request on the “/texts/” endpoint, identification codes used by the database to refer to specific texts can be obtained. These identifiers can specify a source and a target text in the information submitted in a POST to “/parallels/”. 40

Although the TIS-API does not specify a convenient endpoint for determining which search features can be specified, the documentation does specify valid options. Furthermore, the TIS-API defines that when an invalid option is given, the response will include information indicating invalidity of the request. 41

The TIS-API, however, does have a “/stopwords/” endpoint that can be useful in choosing a stopwords list. Recall that stopwords lists are often composed of items that appear most frequently in the language. Given the appropriate information (as defined in the documentation), the “/stopwords/” endpoint consults the Tesserae database and returns the most frequently appearing features (Figure 8). 42

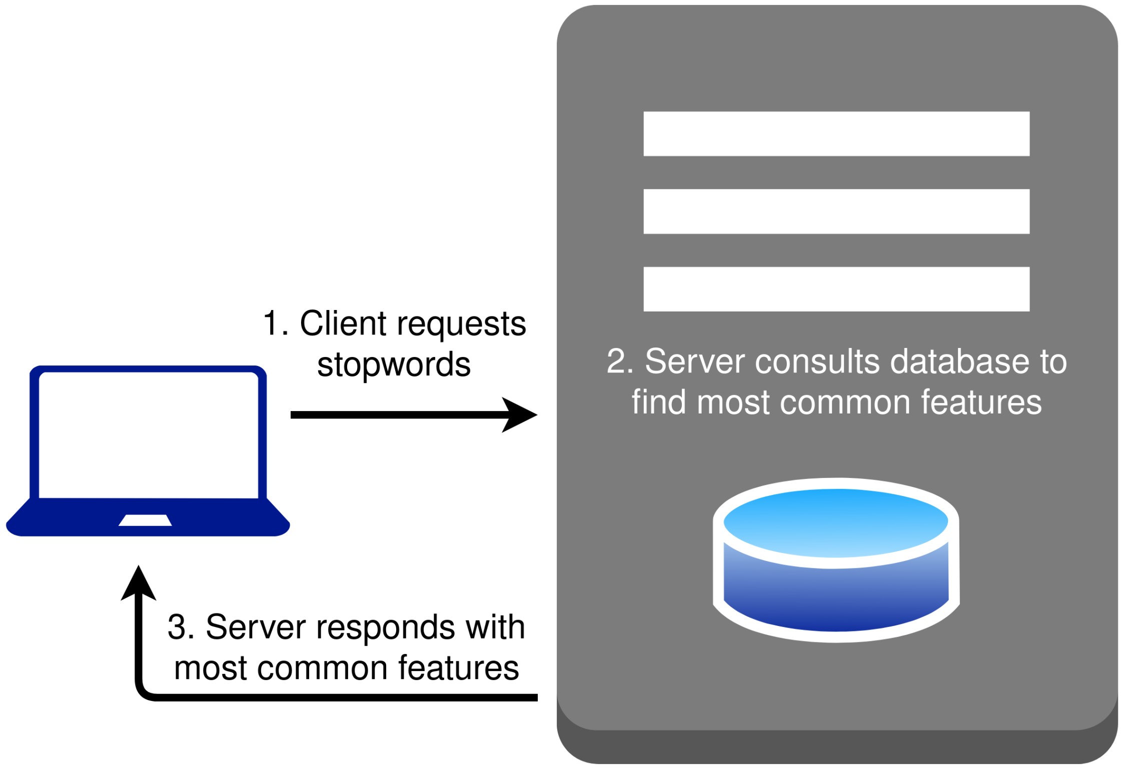

1. Client requests stopwords
2. Server consults database to find most common features
3. Server responds with most common features

**Figure 8.** Diagram describing steps taken to obtain stopwords.

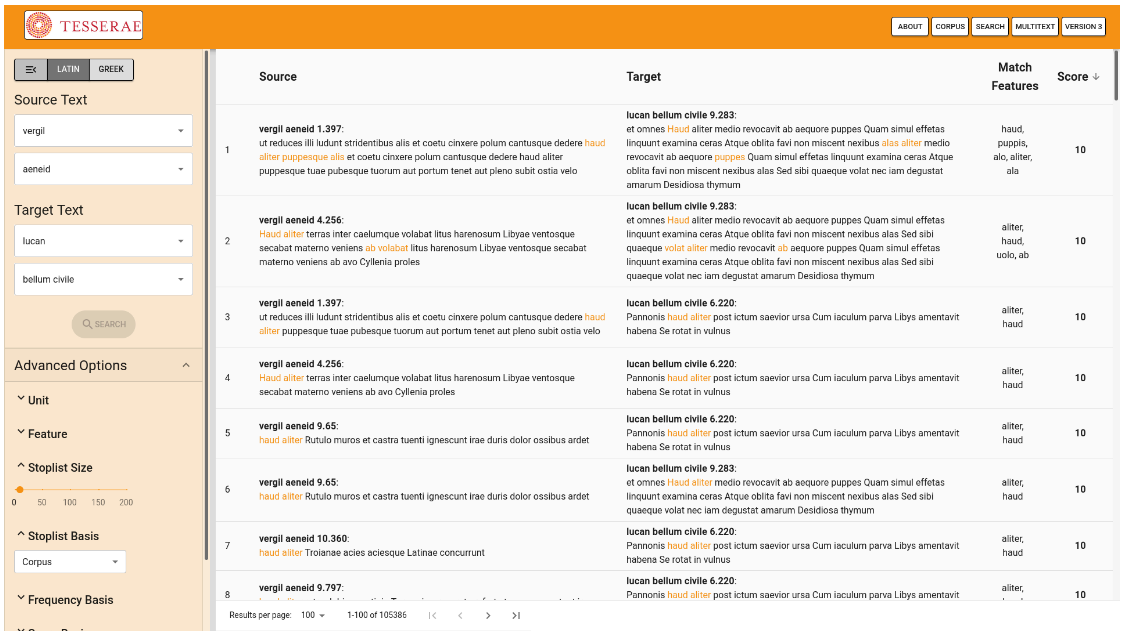
In other words, the “/stopwords/” endpoint can be used to collect the top N most frequently appearing features, where N is a number that can be specified. That list can be used as the stopwords list given to “/parallels/” when submitting a Tesserae search query. 43

It has been shown that the TIS-API provides the “/parallels/” endpoint for submitting a Tesserae search query, as well as the “/texts/” and “/stopwords/” endpoints to aid in choosing Tesserae search options. To retrieve search results, the TIS-API specifies the “/parallels/<uuid>/” endpoint, where “<uuid>” is replaced by a search submission identifier (“UUID” is an acronym for “Universally Unique IDentifier”, which refers to a string of letters and numbers used to identify something). The identifier for a particular search can be found in the response data given by the “/parallels/” endpoint on a successful query submission. In particular, a URL that ends in “/parallels/<uuid>/” will be specified. In other words, a successful Tesserae search submission will provide the URL to retrieve the search results. Importantly, that URL is transmitted in a standard, machine-readable way. 44

For search queries that involve very small texts, going directly to the URL as soon as the response from “/parallels/” comes back may permit retrieving the search results immediately. However, when longer texts are chosen, Tesserae search may take more time to complete. In that case, results would not immediately appear. Instead, requesting a GET at the result URL would yield a response indicating that the resource is not available. To ensure that the search has been submitted, the “/parallels/<uuid>/status/” endpoint, which retrieves status information on the search submission (such as whether the search is in process or if it failed), could be used. When the status indicates that the search is complete, it would be possible to go to the results URL and find the search results. 45

## Case Study: Running Search on the New Tesserae Website

Having seen a theoretical use case for the TIS-API, it seems worthwhile to consider a practical use case for the TIS-API as well. To this end, we present the new Tesserae website, which is available at https://tesserae.caset.buffalo.edu/ (Figure 9 shows what the website looks like at the time of writing when displaying results). This example demonstrates how the TIS-API, though designed for machine-accessibility, can still be used to simulate the traditional human-accessible experience that Tesserae has offered and that its user base in classics scholarship (of widely varying 46 technical comfort) has become familiar with.


TESSERAE

ABOUT CORPUS SEARCH MULTITEXT VERSION 3

LATIN GREEK

Source Text
vergil
aeneid

Target Text
lucan
bellum civile

SEARCH

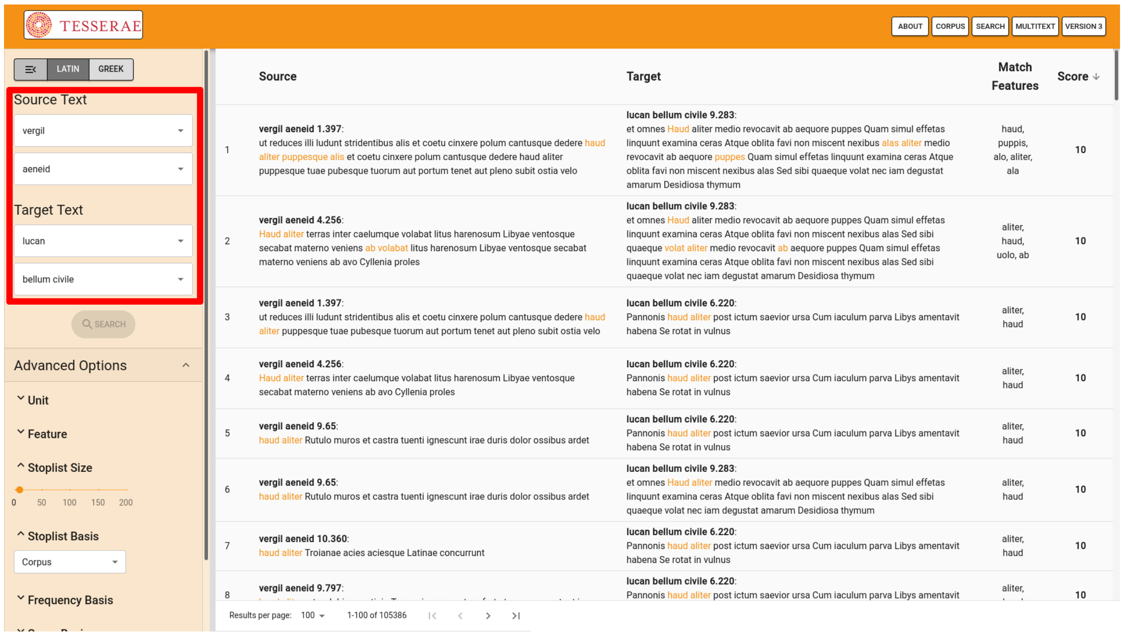
Advanced Options
Unit
Feature
Stoplist Size
0 50 100 150 200
Stoplist Basis
Corpus
Frequency Basis

| | Source | Target | Match Features | Score ↓ |
|---|---|---|---|---|
| 1 | **vergil aeneid 1.397**: ut reduces illi ludunt stridentibus alis et coetu cinxere polum cantusque dedere haud aliter puppesque alis et coetu cinxere polum cantusque dedere haud aliter puppesque tuae pubesque tuorum aut portum tenet aut pleno subit ostia velo | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | haud, puppis, alo, aliter, ala | 10 |
| 2 | **vergil aeneid 4.256**: Haud aliter terras inter caelumque volabat litus harenosum Libyae ventosque secabat materno veniens ab volabat litus harenosum Libyae ventosque secabat materno veniens ab avo Cyllenia proles | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | aliter, haud, uolo, ab | 10 |
| 3 | **vergil aeneid 1.397**: ut reduces illi ludunt stridentibus alis et coetu cinxere polum cantusque dedere haud aliter puppesque tuae pubesque tuorum aut portum tenet aut pleno subit ostia velo | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 4 | **vergil aeneid 4.256**: Haud aliter terras inter caelumque volabat litus harenosum Libyae ventosque secabat materno veniens ab avo Cyllenia proles | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 5 | **vergil aeneid 9.65**: haud aliter Rutulo muros et castra tuenti ignescunt irae duris dolor ossibus ardet | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 6 | **vergil aeneid 9.65**: haud aliter Rutulo muros et castra tuenti ignescunt irae duris dolor ossibus ardet | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | aliter, haud | 10 |
| 7 | **vergil aeneid 10.360**: haud aliter Troianae acies aciesque Latinae concurrunt | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 8 | **vergil aeneid 9.797**: | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit | aliter, | 10 |

Results per page: 100 1-100 of 105386

**Figure 9.** Various parts of the new Tesserae website communicate with the API.

In order to perform a Tesserae search, the user first needs to choose source and target texts. The new website displays the source and target text options in dropdown menus, found in the top left part of the interface (see Figure 10). Behind the scenes, the website uses the "/texts/" endpoint to find what texts are available for comparison from the TIS-API. It then populates the dropdown lists with the information received from the "/texts/" endpoint. The user can then select source and target texts with these dropdown menus. In the example image (Figure [uiscreenshot]), the user has selected Vergil's*Aeneid*as the source text and Lucan's Bellum Civile as the target text.


TESSERAE

ABOUT CORPUS SEARCH MULTITEXT VERSION 3

LATIN GREEK

Source Text
vergil
aeneid

Target Text
lucan
bellum civile

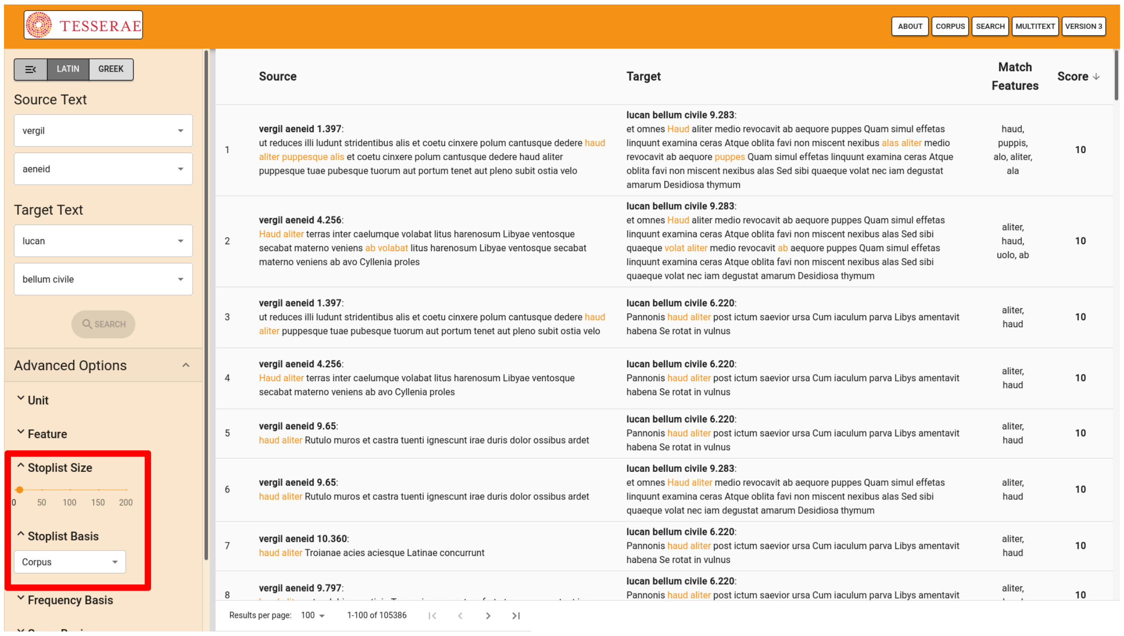
SEARCH

Advanced Options
Unit
Feature
Stoplist Size
0 50 100 150 200
Stoplist Basis
Corpus
Frequency Basis

| | Source | Target | Match Features | Score ↓ |
|---|---|---|---|---|
| 1 | **vergil aeneid 1.397**: ut reduces illi ludunt stridentibus alis et coetu cinxere polum cantusque dedere haud aliter puppesque alis et coetu cinxere polum cantusque dedere haud aliter puppesque tuae pubesque tuorum aut portum tenet aut pleno subit ostia velo | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | haud, puppis, alo, aliter, ala | 10 |
| 2 | **vergil aeneid 4.256**: Haud aliter terras inter caelumque volabat litus harenosum Libyae ventosque secabat materno veniens ab volabat litus harenosum Libyae ventosque secabat materno veniens ab avo Cyllenia proles | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | aliter, haud, uolo, ab | 10 |
| 3 | **vergil aeneid 1.397**: ut reduces illi ludunt stridentibus alis et coetu cinxere polum cantusque dedere haud aliter puppesque tuae pubesque tuorum aut portum tenet aut pleno subit ostia velo | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 4 | **vergil aeneid 4.256**: Haud aliter terras inter caelumque volabat litus harenosum Libyae ventosque secabat materno veniens ab avo Cyllenia proles | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 5 | **vergil aeneid 9.65**: haud aliter Rutulo muros et castra tuenti ignescunt irae duris dolor ossibus ardet | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 6 | **vergil aeneid 9.65**: haud aliter Rutulo muros et castra tuenti ignescunt irae duris dolor ossibus ardet | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | aliter, haud | 10 |
| 7 | **vergil aeneid 10.360**: haud aliter Troianae acies aciesque Latinae concurrunt | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 8 | **vergil aeneid 9.797**: | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit | aliter, | 10 |

Results per page: 100 1-100 of 105386

**Figure 10.** Along the top-left of the page are drop down menus that communicate with the "/texts/" endpoint.



Source Text: vergil, aeneid

Target Text: lucan, bellum civile

Stoplist Size: 0 50 100 150 200

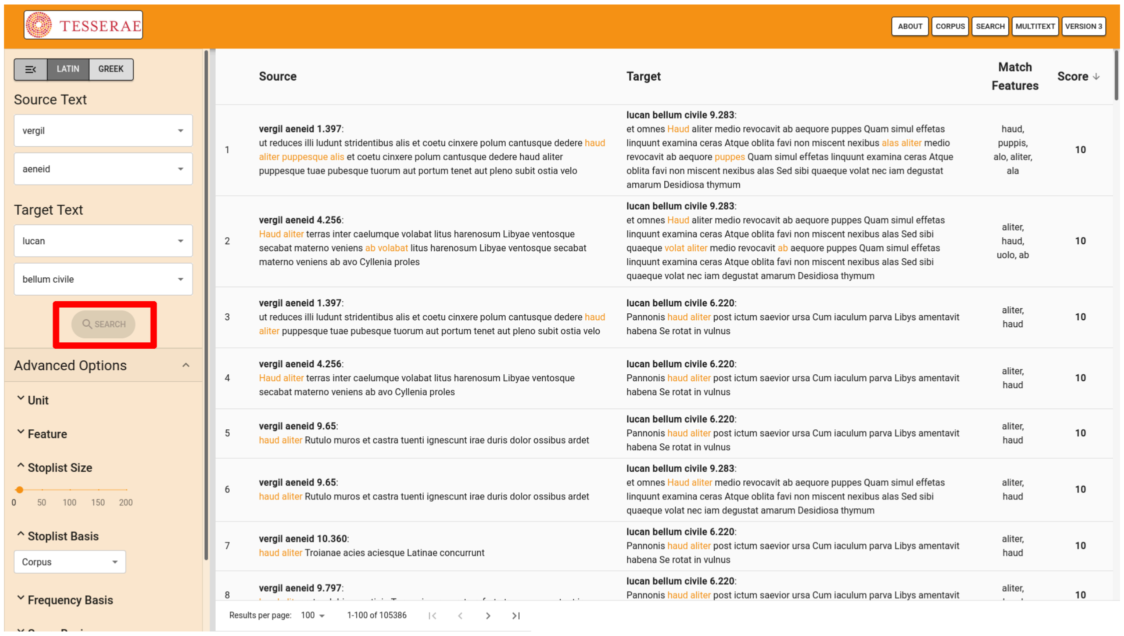
Stoplist Basis: Corpus

| | Source | Target | Match Features | Score ↓ |
|---|---|---|---|---|
| 1 | **vergil aeneid 1.397**: ut reduces illi ludunt stridentibus alis et coetu cinxere polum cantusque dedere haud aliter puppesque alis et coetu cinxere polum cantusque dedere haud aliter puppesque tuae pubesque tuorum aut portum tenet aut pleno subit ostia velo | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | haud, puppis, alo, aliter, ala | 10 |
| 2 | **vergil aeneid 4.256**: Haud aliter terras inter caelumque volabat litus harenosum Libyae ventosque secabat materno veniens ab volabat litus harenosum Libyae ventosque secabat materno veniens ab avo Cyllenia proles | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | aliter, haud, uolo, ab | 10 |
| 3 | **vergil aeneid 1.397**: ut reduces illi ludunt stridentibus alis et coetu cinxere polum cantusque dedere haud aliter puppesque tuae pubesque tuorum aut portum tenet aut pleno subit ostia velo | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 4 | **vergil aeneid 4.256**: Haud aliter terras inter caelumque volabat litus harenosum Libyae ventosque secabat materno veniens ab avo Cyllenia proles | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 5 | **vergil aeneid 9.65**: haud aliter Rutulo muros et castra tuenti ignescunt irae duris dolor ossibus ardet | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 6 | **vergil aeneid 9.65**: haud aliter Rutulo muros et castra tuenti ignescunt irae duris dolor ossibus ardet | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | aliter, haud | 10 |
| 7 | **vergil aeneid 10.360**: haud aliter Troianae acies aciesque Latinae concurrunt | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | 10 |
| 8 | **vergil aeneid 9.797**: | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit | aliter, | 10 |

Results per page: 100 1-100 of 105386

**Figure 11.** The widgets in the bottom-left of the page communicates with the "/stopwords/" endpoint to create a stopwords list.

The user might also wish to have stopwords for the Tesserae search. On the new website, the number of stopwords to be used can be specified under the "Advanced Options" (see Figure 12). Behind the scenes, the website uses the parameters specified under "Advanced Options" to calculate a stopwords list using the "/stopwords/" endpoint. In the example image (Figure 13), the user has left the slider bar under "Stoplist Size" at the default of 10 stopwords. The example image further shows that the stopwords will be calculated based on the corpus available to the TIS-API, as shown by the selection under "Stoplist Basis".


Source Text: vergil, aeneid

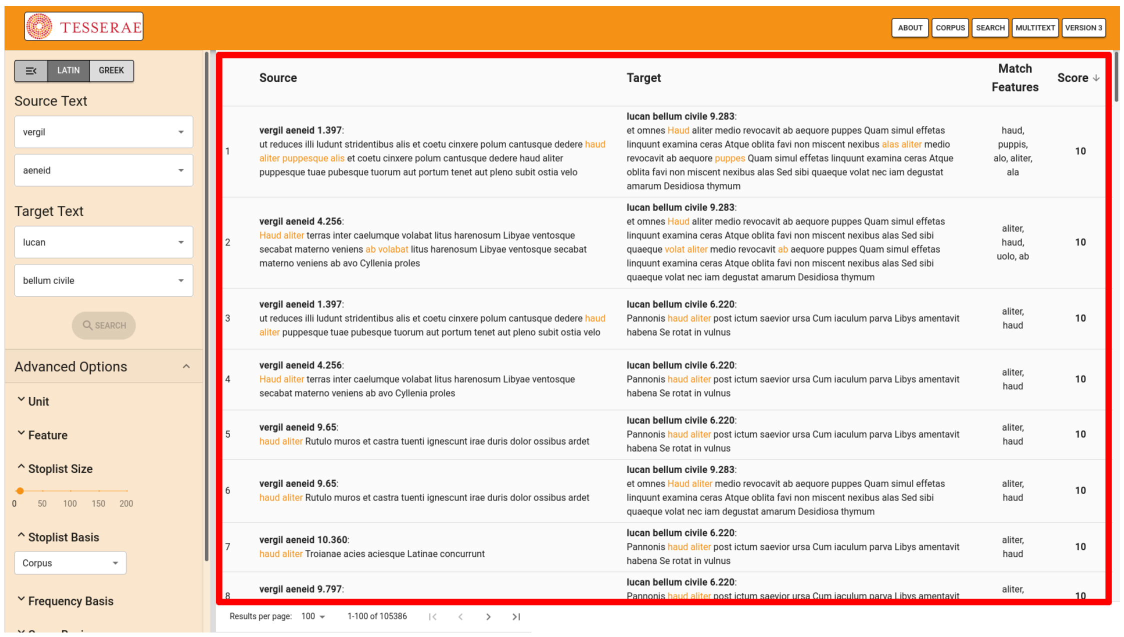
Target Text: lucan, bellum civile

Stoplist Size: 0 50 100 150 200

Stoplist Basis: Corpus

Results per page: 100 1-100 of 105386

**Figure 12.** The button labeled "SEARCH" at the middle-left of the page sends the query to the "/parallels/" endpoint. (In this image, the search button is greyed out because the query has already been sent.)



Source Text: vergil, aeneid

Target Text: lucan, bellum civile

Stoplist Basis: Corpus

| | Source | Target | Match Features | Score ↓ |
|---|---|---|---|---|
| 1 | **vergil aeneid 1.397**: ut reduces illi ludunt stridentibus alis et coetu cinxere polum cantusque dedere haud aliter puppesque alis et coetu cinxere polum cantusque dedere haud aliter puppesque tuae pubesque tuorum aut portum tenet aut pleno subit ostia velo | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | haud, puppis, alo, aliter, ala | **10** |
| 2 | **vergil aeneid 4.256**: Haud aliter terras inter caelumque volabat litus harenosum Libyae ventosque secabat materno veniens ab volabat litus harenosum Libyae ventosque secabat materno veniens ab avo Cyllenia proles | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | aliter, haud, uolo, ab | **10** |
| 3 | **vergil aeneid 1.397**: ut reduces illi ludunt stridentibus alis et coetu cinxere polum cantusque dedere haud aliter puppesque tuae pubesque tuorum aut portum tenet aut pleno subit ostia velo | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | **10** |
| 4 | **vergil aeneid 4.256**: Haud aliter terras inter caelumque volabat litus harenosum Libyae ventosque secabat materno veniens ab avo Cyllenia proles | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | **10** |
| 5 | **vergil aeneid 9.65**: haud aliter Rutulo muros et castra tuenti ignescunt irae duris dolor ossibus ardet | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | **10** |
| 6 | **vergil aeneid 9.65**: haud aliter Rutulo muros et castra tuenti ignescunt irae duris dolor ossibus ardet | **lucan bellum civile 9.283**: et omnes Haud aliter medio revocavit ab aequore puppes Quam simul effetas linquunt examina ceras Atque oblita favi non miscent nexibus alas Sed sibi quaeque volat nec iam degustat amarum Desidiosa thymum | aliter, haud | **10** |
| 7 | **vergil aeneid 10.360**: haud aliter Troianae acies aciesque Latinae concurrunt | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit habena Se rotat in vulnus | aliter, haud | **10** |
| 8 | **vergil aeneid 9.797**: | **lucan bellum civile 6.220**: Pannonis haud aliter post ictum saevior ursa Cum iaculum parva Libys amentavit | aliter, | **10** |

Results per page: 100 1-100 of 105386

**Figure 13.** The majority of the screen has been populated with the results of a search, obtained from the "/parallels/<uuid>/" endpoint.

To submit the Tesserae search for processing, the user simply hits the "SEARCH" button (see Figure 12). This action causes the website to submit the search parameters to the "/parallels/" endpoint. Once the search is complete, the website automatically calls on the "/parallels/<uuid>/" endpoint to retrieve the results of the search and displays those results on the main part of the interface (see Figure 13). 49

One tangential benefit of defining the new website's functionality in terms of the TIS-API was development flexibility. Because the TIS-API defines expected behavior, the website developer could work on the website at the same time that the API developer was implementing the expected behavior. This allowed for important feedback informed by the website developer's user experience. That feedback resulted in additional behaviors (e.g., paging of the search results, more detailed search status information) that improved the API and website together. As the two efforts came together for deployment of the new Tesserae website, the integration of the website with the TIS-API implementation required relatively little work (deployment itself, along with making sure that the user interface looked right, took most of the development time). This demonstrates the utility of the TIS-API internal to the Tesserae Project. Other digital humanities developers may want to consider this benefit when deciding whether they too can afford to improve their software's ability to integrate with other software. 50

## Case Study: Harnessing the TIS-API for Research

While the new Tesserae website demonstrates the advantages of the TIS-API for members of the Tesserae Project, it is also important to establish the advantages conferred to researchers not directly affiliated with the project. This section describes one such researcher's experience with the TIS-API. 51

For writing his 2017 article *Measuring and Mapping Intergeneric Allusion using Tesserae* for a special issue of the Journal of Data Mining and Digital Humanities on Computer-Aided Processing of Intertextuality in Ancient Languages [Burns 2017], one of the co-authors of this article found himself in the position of downloading the results of multiple Tesserae web searches in order to analyze not just an intertextual comparison of two authors, but rather an intertextual comparison of one author (the Latin epic poet Lucan) against several authors representative of a specific genre (the Latin love elegists Propertius, Tibullus, and Ovid). The searches had to be conducted individually and care needed to be taken to ensure that each search used the same parameters, including stopword lists and cutoff scores, among other settings. Moreover, the author needed to be sure that this consistency in parameter selection could be maintained between sessions, with sometimes weeks or even months passing between searches. At the time, these factors were 52 documented by the author in research notes, but from testing the TIS-API, it is clear how this process could have been improved had the new set-up been available: a similar paper written now could document the exact TIS-API calls for all of the author-to-author comparisons in a script with parameters specified upfront as constants.

53 This enhancement to the research process made possible through the TIS-API could be seen as a variation on software development's DRY — don't repeat yourself — principle [Hunt 2000]. In programming, DRY argues for a reduction of duplicate code to increase writing efficiency, reduce opportunities for errors to be introduced, and decrease maintenance costs. The TIS-API allows for analogous benefits in a research context: searches can be formalized and then run and rerun without duplicative effort. To paraphrase Andy Hunt and Dave Thomas's oft-cited definition of the DRY principle, the TIS-API makes it possible for every Tesserae search to "have a single, unambiguous, authoritative representation" [Hunt 2000, 27] within a study.

54 Accordingly, code-based calls to the TIS-API should be considered now a best practice for researchers needing to aggregate the results of multiple Tesserae searches, as for example papers such as [Bernstein 2015]. That said, even with respect to research making use of even a single Tesserae search, the ability to document the request in the form of an API request is valuable. This is because, no matter the number of searches, being able to document explicitly decisions made and actions taken in the course of gathering data plays an important role in making that research reproducible. In other words, code-based calls to the TIS-API improve researchers' experience by making decisions explicitly documented at the time of data-gathering, rather than as an afterthought. It has been argued that researchers should aim for well-documented and reproducible workflows as part of a critical digital humanities [Dobson 2019]. It can also be argued that academic developers can support researchers in reaching this goal by making it easier to incorporate their software tools into research workflows. The TIS-API is a move in this direction.

55 One final point about using Tesserae searches in research workflows: at present, running, say, dozens or hundreds of Tesserae searches using the web form could be seen as a labor-intensive proposition, enough so as to be a disincentive to exploring research questions demanding this volume of web-form entry. Accessing the TIS-API through a script, on the other hand, greatly improves the researcher's experience by reducing the labor of specifying Tesserae searches, which in turn reduces the risk of inconsistent querying and related errors that add more labor in data checking and correction or even having to rerun completed but unusable searches. With this in mind, one can imagine that availability of the TIS-API, by reducing the difficulty of collecting and collating multiple searches, will encourage more large-scale Tesserae-based studies.

## Future Work

56 For all of the benefits made possible through the TIS-API, we recognize that the API could be improved. One improvement in particular we are considering is a way to document which version of the software produced a given set of results. This will be important as software development continues on Tesserae's core functionality. While such development aims to improve Tesserae results, those improvements would come at the cost of reproducibility. It is possible that the development improvement would cause results obtained through one set of parameters on a given date to be different from the results obtained through the same set of parameters on a different date. By explicitly documenting which version of the Tesserae code produces a given set of results, reproducibility of results could be guaranteed while also permitting incremental improvements to Tesserae's core software.

57 It will also be interesting to see how the TIS-API serves the growing movement within digital classics to provide better interoperability between collections and tools emerging from different projects within the field [Burns 2019]. Such cross-project collaborations within digital classics hold much promise for enabling new modes of inquiry across well-known editions and newly digitized works.

58 Beyond digital classics, we hope the TIS-API will serve as a helpful point of comparison for other digital humanities projects. A future in digital humanities where larger questions can be answered through the aggregation of data from multiple sources will be more easily realized when individual projects provide an API to make their data more machine-accessible. The considerations made in the design and implementation of the TIS-API, including the adoption of REST principles and the analysis of Tesserae's intertext discovery process, can serve as inspiration for other projects as they

make decisions on how to design and implement their own APIs.

## Conclusion

The TIS-API enables machine-accessibility of Tesserae intertext discovery capabilities. Machine-accessibility is achieved by following the software industry standards known as the REST principles, which encouraged thinking about Tesserae's capabilities as resources and considering how to act upon those resources. In particular, the REST principles are evident in how Tesserae's texts are made available, how stopwords are computed from Tesserae's text collection, and how Tesserae search is implemented without saving application state either in the client or the server. 59

Making Tesserae machine-accessible yields two main benefits. First, it allows for parallelizing development efforts in the Tesserae software, as evidenced by the simultaneous development of both the TIS-API implementation and the new Tesserae website. The second benefit is in the promising possibilities for those not affiliated with the Tesserae project. Specifically, the ability to document search procedures exactly should lead to improved scholarly practices in computational literary criticism. We expect that other digital humanities projects can likewise reap benefits from upgrading their software with machine-accessibility. 60

## Acknowledgments

This work was made possible by NEH Digital Humanities Advancement Grant HAA-258767-18. PJB acknowledges the support of the Quantitative Criticism Lab and the Institute for the Study of the Ancient World while conducting this work. 61

## Notes

[1] Plagiarism detection has some similarities to intertext detection. Both tasks fall under the more general field of "text reuse".

## Works Cited

**Allen 2011** Allen, G. *Intertextuality.* Routledge, London. (2011).

**Bernstein 2015** Bernstein, Neil, Kyle Gervais, and Wei Lin. "Comparative rates of text reuse in classical Latin hexameter poetry." *Digital Humanities Quarterly* 9.3 (2015).

**Berti 2016** Berti, Monica, et al. "Documenting Homeric Text-Reuse in the Deipnosophistae of Athenaeus of Naucratis." *Bulletin of the Institute of Classical Studies* 59.2 (2016): 121-139. https://doi.org/10.1111/j.2041-5370.2016.12042.x

**Burns 2017** Burns, Patrick J. "Measuring and Mapping Intergeneric Allusion in Latin Poetry using Tesserae." *Journal of Data Mining and Digital Humanities* (2017). https://jdmdh.episciences.org/3821.

**Burns 2019** Burns, Patrick J. "Building a text analysis pipeline for classical languages." In *Digital Classical Philology: Ancient Greek and Latin in the Digital Revolution.* Berlin: De Gruyter (2019). pp. 159–76.

**Büchler 2013** Büchler, M. "Informationstechnische Aspekte des Historical Text Re-use (English: Computational Aspects of Historical Text Re-use)." Ph.D. thesis, Leipzig (2013). See also http://www.etrap.eu/research/tracer/.

**Chaudhuri 2015** Chaudhuri, Pramit, Joseph P. Dexter, and Jorge A. Bonilla Lopez. "Strings, Triangles, and Go-betweens: Intertextual Approaches to Silius' Carthaginian Debates." *Dictynna. Revue de poétique latine* 12 (2015). See also https://www.qcrit.org/filum.

**Coffee 2012a** Coffee, N. "Intertextuality in Latin Poetry." *Oxford Bibliographies in Classics.* D. Clayman, (ed). Oxford University Press, New York (2012).

**Coffee 2012b** Coffee, N., J.-P. Koenig, S. Poornima, R. Ossewarde, C. Forstall and S. Jacobson. "Intertextuality in the Digital Age." *TAPA* 142, no. 2 (2012): 381-419.

**Dobson 2019** Dobson, J.E. *Critical Digital Humanities: The Search for a Methodology.* University of Illinois Press, Champaign, IL (2019).

**Edmunds 2001** Edmunds, L. *Intertextuality and the Reading of Roman Poetry.* Johns Hopkins University Press, Baltimore (2001).

**Fielding 2000** Fielding, Roy T. “Architectural styles and the design of network-based software architectures.” Vol. 7. Ph.D. thesis, University of California, Irvine (2000).

**Fielding 2017** Fielding, Roy T., Richard N. Taylor, Justin R. Erenkrantz, Michael M. Gorlick, Jim Whitehead, Rohit Khare, and Peyman Oreizy. “Reflections on the REST architectural style and principled design of the modern web architecture (impact paper award).” *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*. ACM (2017).

**Forstall 2014** Forstall, Christopher, Neil Coffee, Thomas Buck, Katherine Roache, and Sarah Jacobson. *Modeling the scholars: Detecting intertextuality through enhanced word-level n-gram matching. Digital Scholarship in the Humanities* 30, no. 4 (2014): 503-515. See also http://tesserae.caset.buffalo.edu/.

**Hunt 2000** Hunt, A. and Thomas, D. *The Pragmatic Programmer: From Journeyman to Master.* Addison-Wesley, Boston (2000).

**Juvan 2008** Juvan, M. *Towards a History of Intertextuality in Literary and Culture Studies. CLCweb: Comparative Literature And Culture* 10, no. 3 (2008). https://doi.org/10.7771/1481-4374.1370.

**Lee 2007** Lee, John. “A computational model of text reuse in ancient literary texts.” *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics* (2007).

**Mastandrea 2011** Mastandrea, Paolo, Massimo Manca, L. Spinazzè, L. Tessarolo, and F. Boschetti. “Musisque Deoque: Text Retrieval on Critical Editions.” *Journal for Language Technology and Computational Linguistics* 26 (2011): 129-140. See also http://mizar.unive.it/mqdq/public/.