Erik Henriksson — Marja Vierros

PapyGreek Search: Exploring the Language of Greek Papyri

1 Introduction

The growing interest in the language of the Greek papyri is closely tied to the increased availability of text corpora and digital research tools. The digitisation and open-access publication of all edited documentary papyri through Papyri.info has laid the groundwork for all subsequent corpus-based linguistic studies.[1] Recently, further progress has been made in the preprocessing of papyrological texts for linguistic analysis,[2] the creation of linguistically annotated corpora,[3] and the development of query tools.[4] Despite these advancements, there is arguably still room for new digital resources.

In this chapter, we introduce PapyGreek Search, a tool designed for papyrologists and linguists interested in exploring the language of Greek documentary papyri. Developed as part of "The Digital Grammar of Greek Documentary Papyri" project,[5] PapyGreek Search is freely accessible online through the PapyGreek project website.[6] PapyGreek Search is distinguished by its search interface that enables simultaneous queries on morphosyntactic constructions, and through the editorial interventions encoded in these texts, phonological and morphosyntactic variation. Additionally, PapyGreek Search provides an interface for visually building syntactic tree queries, which allows users to utilize its treebank search feature without needing to learn a treebank query language.

The chapter is structured as follows. Section 2 outlines the motivation for developing PapyGreek Search, explores related work, and highlights the tool's key functionalities. Section 3 details the technical implementation, including text preprocessing, database architecture, and search algorithms. Section 4 demonstrates the user interface and Section 5 discusses example queries. Section 6 concludes.

1 Evans – Obbink 2010.

2 Vierros – Henriksson 2017; Vierros 2018

3 E.g. Vierros – Henriksson 2021; Keersmaekers – Depauw 2017; Keersmaekers – Van Hal 2023.

4 E.g. Depauw – Stolk 2015; Keersmaekers – Mercelis – Swaelens – Van Hal 2019

5 This project, led by Marja Vierros, received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 758481).

6 https://papygreek.com/search. All hyperlinks last accessed on 21.7.2024, unless differently indicated.

https://doi.org/10.1515/9783111070162-011

2 Previous work and main features

We start by outlining the evolution of query tools for Greek papyri, from the earliest searchable digital databases to contemporary online platforms such as Papyri.info and Trismegistos.org, along with treebank query tools. We then describe the main functionalities of PapyGreek Search.

2.1 Previous query tools for Greek papyri

Digital papyrology started in 1983 with the establishment of the Duke Databank of Documentary Papyri (DDbDP). In the early 1990s, the digitized texts were distributed as CD-ROM copies, searchable using the software available at the time.[7] A key step towards wider access occurred in 1996 when the texts were migrated to the online Perseus Project, which had a basic string search interface.[8] In the 2000s and early 2010s, advances in open-source standards and interoperability led to more sophisticated search capabilities for the papyri. The "Integrating Digital Papyrology" project converted the texts into machine-readable EpiDoc XML format, integrated the Heidelberger Gesamtverzeichnis (HGV) metadata with the DDbDP texts, and eventually created Papyri.info.[9] This platform has become the central hub for editing and searching texts, with its Papyrological Navigator offering a full-text search interface with document metadata filters.[10]

The EpiDoc XML-formatted documentary papyri contain a wealth of information on linguistic variation across domains such as orthography, phonology, and morphosyntax, encoded through various editorial interventions. Depauw – Stolk 2015 introduced the first online tool for systematically querying the orthographic variants, named Trismegistos Text Irregularities (TI).[11] This tool effectively superseded traditional reference books that contain manually collected – and by now partly outdated – lists of non-standard linguistic forms found in the papyri.[12] TI enables users to identify all variant forms encoded in the XML source files by character-level differences between transcriptions and modern corrections (such as αι corrected to ε) and to further refine the results by the context in which they appear. However, a limitation of the tool is that it relies on a relatively old snapshot of the DDbDP data (2016).[13]

7 Van Minnen 1996.
8 Sosin 2010.
9 Baumann 2013
10 https://papyri.info/search.
11 https://www.trismegistos.org/textirregularities.
12 E.g. Mayser – Schmoll 1970; Gignac 1976.
13 https://www.trismegistos.org/textirregularities/methodology.php.

Regarding morphosyntactically parsed documents (i.e. treebanks), a variety of search options is available. General treebank query tools include ANNIS3,[14] PML-Tree Query,[15] TüNDRA,[16] and INESS.[17] Many of these programs allow users to search within their own treebanks, including papyrological ones,[18] provided the data is in a compatible format. Recent additions include the stand-alone DendroSearch tool,[19] specifically designed for Ancient Greek treebanks, and the forthcoming KTB tool.[20] These programs typically employ a custom query language that users must learn to construct queries, which may be challenging for non-technical users. Additionally, these tools require downloading and setup to function properly in the user's software environment.

2.2 Main features of PapyGreek Search

PapyGreek Search complements the previously available toolset for the linguistic study of Greek papyri, offering partly overlapping and partly novel features as compared to other available tools. Its main features are as follows:

- **Papyrological focus:** PapyGreek Search introduces some unique features specifically developed for the linguistic exploration of papyrological texts. Most importantly, its treebank search function includes an integrated text irregularities search, crucial for the linguistic analysis of papyrological texts given their frequent misspellings and other nonstandard word forms. In addition, the search can be further narrowed down by various metadata, such as date, provenance, author and/or scribe.
- **User-friendly interface:** The platform combines complex search features with a simple graphical user interface, aiming to make linguistic analysis of papyrological texts accessible to a wide range of users, including those with limited prior experience with specialized query languages.
- **Advanced features:** Most of the search parameters in PapyGreek Search may optionally be written using regular expressions,[21] which can be used to craft complex queries. Users can also specify word order in the treebank queries, an important and relatively understudied aspect of syntactic structures in Greek.[22]
- **Synchronization:** The PapyGreek database is updated weekly based on changes made to the source texts (available from https://github.com/papyri).

14 Krause – Zeldes 2016.
15 Pajas – Štěpánek 2009.
16 Martens 2013.
17 Rosén *et al.* 2012
18 E.g. Vierros – Henriksson 2021.
19 Keersmaekers – Mercelis – Swaelens – Van Hal 2019.
20 Yordanova forthcoming; Vierros – Yordanova 2022.
21 E.g. Goyvaerts – Levithan 2012.
22 E.g. Vierros – Yordanova 2022.

– **Open source:** PapyGreek Search is fully open source. This not only ensures transparency but also allows for continued development by the academic community. The source code for the PapyGreek system is available at https://github.com/erikhenriksson, and feature requests can be sent to papygreek.helsinki@gmail.com.

3 Technical implementation

This section describes the technical implementation of PapyGreek Search. Those primarily interested in the user interface may skip directly to Section 4, though understanding how the system works may help with constructing queries. The section begins with a description of preprocessing the source DDbDP files for linguistic annotation, followed by the implementations of textual irregularities and treebank search. Finally, the database structure is described.

3.1 Preprocessing: from EpiDoc XML into tokens

The EpiDoc XML schema used by the DDbDP includes sections for document divisions, editorial corrections, gaps, and other pertinent elements in papyrological texts, making it a suitable format for *representing* structured texts. However, XML is not efficient for *searching*; databases are typically better for this purpose.[23] For the PapyGreek platform, we chose MySQL as our database backend due to its quick and reliable handling of complex queries, as well as ease of management. MySQL, a structured database, associates each data point (e.g., a single document or word) with predefined data fields. Thus, we had to first convert the DDbDP source files into structured entities that could be stored in the database.

The first and most crucial step in this preprocessing stage is *tokenization*, meaning the texts' conversion into discrete units, in our case words (or 'tokens') and sentences. The word is a prerequisite level of description for a system where users search for words and their morphosyntactic characteristics, and the sentence level is additionally required for searching treebank annotations, which describe the syntactic relationships of the words within sentences. Typically, word tokens are found by simply splitting the text by white-space. However, Greek has many exceptions to this rule, such as *craseis* or merged words (καὶ ἐγώ → κἀγώ "and I"), which are orthographically one but linguistically two words (also called 'multi-word tokens'). Furthermore, some of the DDbDP

23 The search capabilities of the Papyrological Navigator (https://papyri.info), for example, are also powered by a database (Apache SOLR).

files contain errors and inconsistencies in spacing.[24] Our custom tokenizer addresses these exceptions by rule-based heuristics.[25]

As Vierros – Henriksson 2017 and Vierros 2018 suggest, preprocessing papyrological texts for linguistic analysis benefits from creating two distinct tokenizations of the same text: the text as it appears on the papyrus ("original") and its editorially corrected ("regularised") version. Our tokenizer generates these versions utilizing the EpiDoc XML elements that denote editorial interventions (see Section 3.3 below for more details). Specifically, we classify all text as original except passages encoded as regularizations (<reg>), corrections (<corr>), additions (<supplied>) or deletions (<surplus>). For meaningful linguistic analysis, it is essential to pair each original word with its regularised counterpart. In the DDbDP documents, this is generally straightforward, as the <reg> and <corr> elements (alongside their corresponding original tags <orig> and <sic>) typically contain only a single word. However, more complex multi-word regularizations are also found in the source texts.[26] Our tokenizer addresses these with a custom string-matching algorithm that aligns the most similar words between the original and regularised elements.[27]

The tokenizer additionally collects various metadata for each word, including if it is a number (<num>), the section of the text ( ), language (<foreign>), the scribe (<handShift>), the presence of expansions (<ex>) or additions (<surplus>), the line number (<lb>), and the sentence number. Identifying sentence boundaries is achieved simply by dividing the text at sentence-ending symbols, like periods and question marks (";" in Greek).[28] Finally, we store this metadata, along with the original and a (possible) regularised word form, in a MySQL database (see Section 3.4 below).

24 For instance, p.zauzich.39 (https://papyri.info/ddbdp/p.zauzich;;39, accessed 13.9.2023) contains extra spaces between letters (e.g. line 53: ⳨ο̣ υ̣ ϙα̣τ̣αβο υς ο̣ι̣ϰ̣ ι̣[ας] should be ⳨ο̣υ̣ ϙα̣τ̣αβους ο̣ι̣ϰ̣ι̣[ας]).

25 The complexity of the task is reflected in the fact that about 20% of the lines of code in our tokenizer deal with just these exceptions. The source code is available at https://github.com/erikhenriksson/papygreek-tokenizer.

26 For instance, in line 31 of pap.agon.3 (https://papyri.info/ddbdp/p.oxy;27;2476, accessed 15.8.2023) σεβαστάτῃ has been corrected to καὶ εὐσεβαστάτῃ.

27 Some texts contain more than one <reg> version; our tokenizer chooses the first one. However, we also mark the other alternatives and store them in the database for querying (the query implementation, however, is not yet ready as of this writing). Additionally, the texts contain variant readings encoded within <lem> and <rdg> elements. We choose to include the <lem> elements, since the <rdg> elements typically contain older and later corrected readings. However, as in the case of multiple <reg> versions, we also store the <rdg> tokens in our database.

28 These symbols, of course, are also editorial additions, representing modern interpretative decisions.

3.2 Morphosyntactic annotations

The PapyGreek database includes a selection of texts that we have morphosyntactically annotated using the Ancient Greek Dependency Treebank 2.0 annotation schema.[29] The annotation has been done using the Arethusa annotation environment,[30] which we have integrated into the PapyGreek platform. For the texts that are annotated, each token gets the following morphosyntactic data fields: 'lemma' (the dictionary form), 'postag' (the 9-character code encoding part-of-speech, person, number, etc.),[31] 'head' and 'relation'. The last two are part of the syntactic annotation schema, denoting the syntactic parent and the node's syntactic relation to it, respectively. We have annotated two versions of each sentence, the "original" and the "regularised" version (see Section 3.1 above). As of this writing, 650 texts have been annotated manually by experts in the Greek language. To ensure the quality of these annotations, PapyGreek has a review system where reviewers can accept or reject texts that annotators submit for review.

In addition to the manual annotations, PapyGreek also utilises automatic part-of-speech tagging and morphological analysis, based on the Ancient Greek BERT language model developed by Singh et al. (2021). The pre-trained model, available from Hugging Face[32], was fine-tuned for postagging using the AGDT, PROIEL and Gorman treebanks as training data,[33] and we further fine-tuned it using the PapyGreek Treebanks.[34] The postag prediction accuracy of the model for a holdout set was approximately 90%, which is acceptable considering the fragmentary nature of many of the included texts.[35] A limitation of the BERT model is that it can only predict POS tags but not lemmas. To include automatic lemmatization, we used the morphological analyser Morpheus[36] to map the predicted postag and its word form to its dictionary form. As with the manual annotations, we automatically annotated the original and regularised versions of the texts separately. Our search interface includes a confidence score (output by the BERT model) of the predicted POS tags and lemmas, allowing users to assess their quality.

29 https://github.com/PerseusDL/treebank_data/blob/master/AGDT2/guidelines/Greek_guidelines.md (accessed 25.8.2023).

30 https://github.com/alpheios-project/arethusa (accessed 25.8.2023).

31 For the full list of the POS tag codes used, see https://github.com/gcelano/LemmatizedAncientGreekXML (accessed 25.8.2023).

32 https://huggingface.co/pranaydeeps/Ancient-Greek-BERT (accessed 25.8.2023).

33 https://github.com/pranaydeeps/Ancient-Greek-BERT (accessed 25.8.2023).

34 Vierros – Henriksson 2021.

35 We plan to continue training the model as the training data (i.e. manually annotated treebanks) grows. The code for the automatic tagger can be found at https://github.com/erikhenriksson/papygreek-tagger.

36 We used the XML version of Morpheus, available from https://github.com/gcelano/LemmatizedAncientGreekXML/tree/master/Morpheus (accessed 25.8.2023).

3.3 Editorial interventions

Due to historical practices, editorial interventions in papyri are often labelled as "corrections". However, as Depauw – Stolk 2015 point out, the linguistically intriguing elements are typically the variant forms themselves, which can illuminate phonological and grammatical changes in everyday language usage.[37] Before the introduction of the Trismegistos Text Irregularities tool (see Section 2.1 above), there was no feasible method for systematically studying editorial alterations in the DDbDP as indicators of linguistic variation. For example, identifying every instance of ο being corrected to ω across more than 50,000 DDbDP source documents would have required navigating through each file's XML structure and running a string search within all elements potentially containing editorial changes – an exceedingly slow and inefficient operation. Trismegistos Text Irregularities pioneered fast access to this information by collecting these variations into a searchable database. PapyGreek Search adopts a similar approach, with some differences.

As outlined above (in Section 3.1), our tokenizer treats `<reg>` and `<corr>` elements as "regularised" text, and `<orig>` and `<sic>` as their "original" counterparts. Additionally, it utilizes editorial insertions (`<supplied>`) and deletions (`<surplus>`) to generate two versions of words that include these modifications. To make searchable the character-level differences between these versions, an appropriate algorithm was necessary. Finding differences between sequences is a well-known problem in computer science,[38] with several existing algorithms to choose from. We sought a method that would align well with our intuitions about the editors' intended corrections. Most character-based algorithms, by contrast, are designed to identify mathematically minimal edits between sequences, which are sometimes counterintuitive.[39] The algorithm we ultimately chose, Python's "difflib," utilizes gestalt pattern matching[40] to find the longest contiguous sequences that match in the compared strings, yielding results that seem to match human intuitions. Furthermore, difflib suggested more natural edits for word-initial differences than the other tools we evaluated. Take the word forms ενεγραφε and εγραφε. Both the "windiff" (Microsoft Windows) and "jsdiff"[41] interpret the difference as ε~~νε~~γραφε (remove the word-internal νε), while "difflib" gave the correct edit ~~εν~~εγραφε (remove the prefix εν).

37 See also Dickey 2011.

38 Gusfield 1997.

39 Myers 1986. For example, consider the two arbitrary sequences: (a) "to a blue table" and (b) "table". To a human, it is quite obvious that the difference is that (b) does not have "to a blue". However, most string-comparison libraries (e.g. the Unix "diff") suggest that to get from (a) to (b) one needs to remove all spaces, remove the character "o" from "to", remove "u" from "blue", respectively, and omit the final word "table".

40 Ratcliff – Metzener 1988.

41 By Kevin Decker, https://github.com/kpdecker/jsdiff.

After identifying the character-level edits (i.e. additions and removals) between the original and regularised word forms, as well as their left and right context, our system stores them in a separate MySQL database table. Importantly, we also include the parent word ID in each edit instance, which provides access to the data associated with the corresponding word (that is, lemma, postag, position in the sentence, etc.). This linking is crucial, since it enables queries combining linguistic variation with, for instance, part-of-speech (see Section 5 for example queries). The variant-collecting process is fully automated (in comparison, Depauw – Stolk 2015 describe a semi-manual process in creating the Textual Irregularities database).

3.4 Dependency relationships

One of the key goals – and technical challenges – of PapyGreek Search was enabling fast searches across morphosyntactic parsed texts In the AGDT schema we use (see Section 3.2 above), each word is associated with a single "head", its direct ancestor in the dependency tree. A sample XLM output of the Arethusa annotation environment is illustrated in Figure 1, showing how the head parameter references words in the same sentence.

```
<sentence id="1">
    <word id="1" form="ὃς" relation="SBJ" head="4"/>
    <word id="2" form="ἄν" relation="AuxY" head="4"/>
    <word id="3" form="τις" relation="ATR" head="1"/>
    <word id="4" form="εὕρῃ" relation="SBJ" head="9"/>
    <word id="5" form="τοῦτο" relation="ATR" head="7"/>
    <word id="6" form="τὸ" relation="ATR" head="7"/>
    <word id="7" form="ὄστρακον" relation="OBJ" head="4"/>
    <word id="8" form="," relation="AuxX" head="4"/>
    <word id="9" form="δώσει" relation="PRED" head="0"/>
    <word id="10" form="στατῆρα" relation="OBJ" head="9"/>
    <word id="11" form="." relation="AuxK" head="0" />
</sentence>
```

Fig. 1: An example output from Arethusa (simplified for demonstration by leaving out, among other things, the lemma and morphological information). Sentence 1 from o.claud.1.1, "regularised" version.

Similarly to the XML representation, our database stores the dependency relationships using the "head" column of the "word" table. Though simple and easy to manage, this format is ill-suited for querying purposes. It requires recursively rebuilding the tree structure for each queried sentence, a slow and computationally expensive process, especially when conducted across a large database of annotated sentences.

To address this, the PapyGreek Search MySQL database implements the so-called closure table design pattern.[42] This approach involves storing all possible paths within syntactic trees – that is, every child, grandchild, great-grandchild, and so on – in a separate database table. In other words, instead of storing just the *immediate ancestor* of each node (as in the traditional schema), we store *every descendant path* from each node down to the last non-branching child (see Fig. 2 for an illustration). This method simplifies queries and reduces query times by eliminating the need to reconstruct the hierarchy for each search separately. In addition, we link each stored path with the respective word records (using unique word IDs given to each word), which gives syntactic queries access to other word data such as its form, morphological annotation, and possible linguistic variations. This data linking forms the basis for the multi-domain search possibilities mentioned at the start (see also Section 3.6 below).

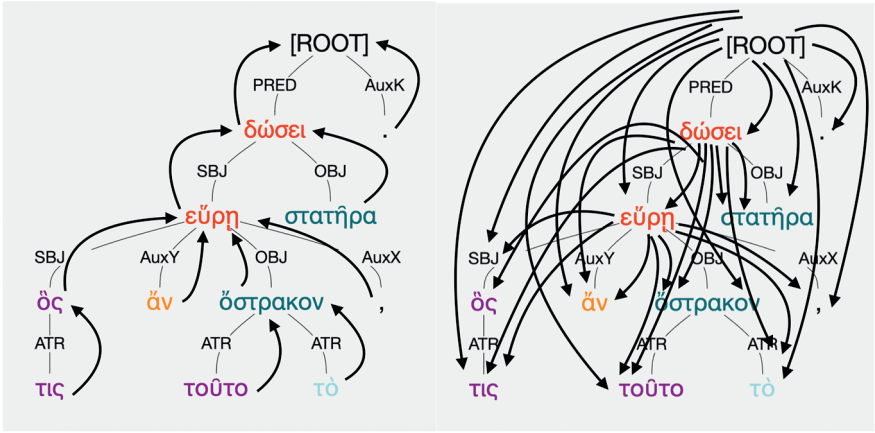

Fig. 2: Two syntactic dependency representation strategies: the conventional method, which records each node's immediate ancestor (left), vs. the closure table approach, capturing all nodes' descendant paths (right). The latter strategy takes up more space but makes for much more efficient querying across the hierarchy. Sentence 1 from o.claud.1.1, "regularised" version.

3.5 Metadata

In addition to storing data about words and sentences, the PapyGreek database also includes various document-level metadata. This is automatically collected from the DDbDP source files (including name, series name, main language, HGV numbers, TM numbers, last changed date) as well as from the linked HGV files (date not after, date not before, and place name). The date information is converted into integers (e.g. -300

42 Karwin 2010.

meaning 300 BC), which makes it easier to use them in queries. The place names, which in the source files are written with some variation (e.g. “oxyrhynchos”, “oxyrhynchus” and “oxyrynchos”), are normalized and mapped to *nomoi* (e.g. “Egypt: U19”) and other larger geographical names.[43] Users can choose to filter the search either using these converted names, or the original names as they stand in the HGV.

The PapyGreek system also includes manually entered metadata about the ancient people and genres associated with the texts. We split the documents by each <handShift> element and treat the generated splits as individual “acts of writing” (AOW).[44] Each AOW can be associated with one or more text types[45] as well as ancient persons playing different roles in the production of the document, including “author”, “writer”, “addressee”, and “external official”. Where the associated person has a Trismegistos Person ID,[46] we enter it in the metadata. Moreover, each person added to the PapyGreek database gets a unique identifier (“PapyGreek Person ID”). This is especially useful for distinguishing the same individual across multiple documents, even when the person has an unknown name and lacks a TM Person ID.

3.6 Relational database structure

The PapyGreek Search MySQL table structure comprises tables for preprocessed texts, separated into word and document tables, an edit table for character-level editorial changes within words, a closure table for syntactic structures, and metadata tables for persons and text types associated with different “acts of writing” (see Section 3.5 above). The key feature of this MySQL table structure are the links between data points (e.g., individual documents, words, syntactic relations, and variations), which enable combined searches using information from different tables. Figure 3 displays a simplified diagram of the relevant table relationships.

To illustrate these relationships, consider an example query. Suppose a researcher wants to identify the “original” word forms (see Section 3.3. above) lemmatised as εἰμί, specifically within syntactic constructions where the word has some subject as dependent. Additionally, she is only interested in those word forms where the editor has added a missing ε (e.g. forms like ἰμί corrected to εἰμί, ἰσίν corrected to εἰσίν, etc.), further restricting the search to documents dated to the 4th century AD.

43 The code for this conversion is available as part of the PapyGreek tokenizer (https://github.com/erikhenriksson/papygreek-tokenizer).

44 Vierros – Henriksson 2017.

45 We use a hierarchical text type list, with three levels (hypercategory, category, subcategory). For example, one of the hypercategories is called “private”. It includes the categories “letter”, “list”, “Memorandum (private)”, and “school text”. These categories further include subcategories; for instance, “letter” includes four possible subcategories, such as “private correspondence”.

46 Broux – Depauw 2015.

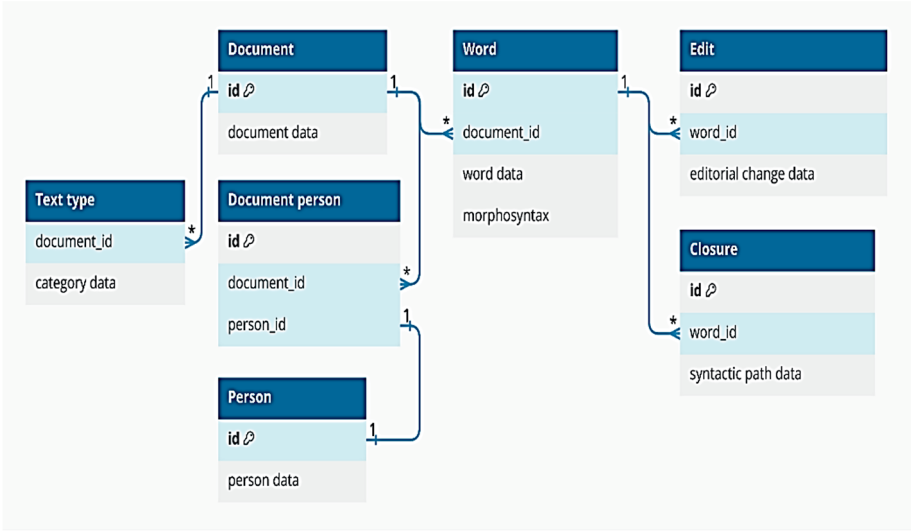

Fig. 3: A (simplified) diagram of the PapyGreek Search database, illustrating the relationships between tables that enable multi-domain queries.

Schematically, PapyGreek search processes the query as follows. First, the system identifies word records with the lemma εἰμί in the 'lemma_orig' field of the word table. Because each word is connected to its parent document, via this linking the retrieved words can be limited to documents that have the specified dates (300-399 AD) in their metadata fields. Next, using the closure table, which links syntactic dependencies to each word record, the search further narrows down to words having a subject dependent. Finally, the search selects only those words that meet the previous criteria *and* are present in the "edit" table with records containing an ε addition. This linkage of tables and gradual filtering of returned data is managed through unique identifiers associated with each database record.

This example query yields one result:[47] the word ἰσίν in the 6th sentence of p.mert.1.32, dated to the early 4th century. The sentence in question is ἰσίν τιναις γὰρ οἳ ἐπιθύμη[σαν] τοῦ τόπου, where ἰσίν indeed has a subject dependent (τιναις), as shown in Figure 4.

47 Retrieved 11.10.2023. Only one result was found because the treebank search targets the annotated sub-corpus in the PapyGreek database, which includes few documents from the 4th century. The query is built with two vertical search boxes (see Section 4.2 below for an explanation), with the parameters "lemma=εἰμί,form=+ε" in the upper box, and "relation=SBJ" in the lower box.

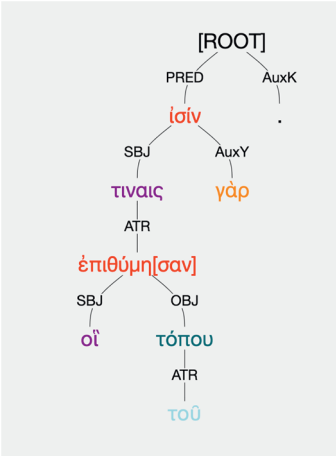

Fig. 4: Syntactic tree of ἰσίν τιναις γὰρ οἳ ἐπιθύμη[σαν] τοῦ τόπου from p.mert.1.32 ("original" version), showing the query result where ἰσίν has a subject dependent (τιναις).

4 User Interface

In this section, we walk through the main features of the PapyGreek search, beginning with an overview of the search interface. We then cover the available query parameters, followed by methods for identifying textual irregularities. Finally, we describe how to construct syntactic queries graphically.

4.1 Overview

The PapyGreek Search interface is divided into two main sections: 1) the search configuration section, where users set their search criteria, and 2) the results display section, where the outcomes of the search are presented. Starting with the search configuration section (Figure 5), users can first select to target either the "original" or "regularised" text versions (see Sections 3.1 and 3.2 for how these versions are created from the DDbDP source files).

Below this selection, a search box with a blue background allows users to target specific data fields of words stored in the PapyGreek database using various parameters, discussed below in Sections 4.2 and 4.3. Section 4.4 will further explain how to add more boxes to construct a syntactic tree search. The core functionality of the tool is centered around these search boxes: the parameters specified within them, and – in syntactic searches – the way they are arranged. Finally, the search area also includes several metadata fields, such as place, date, text type, and person, which can be used to further refine the search.

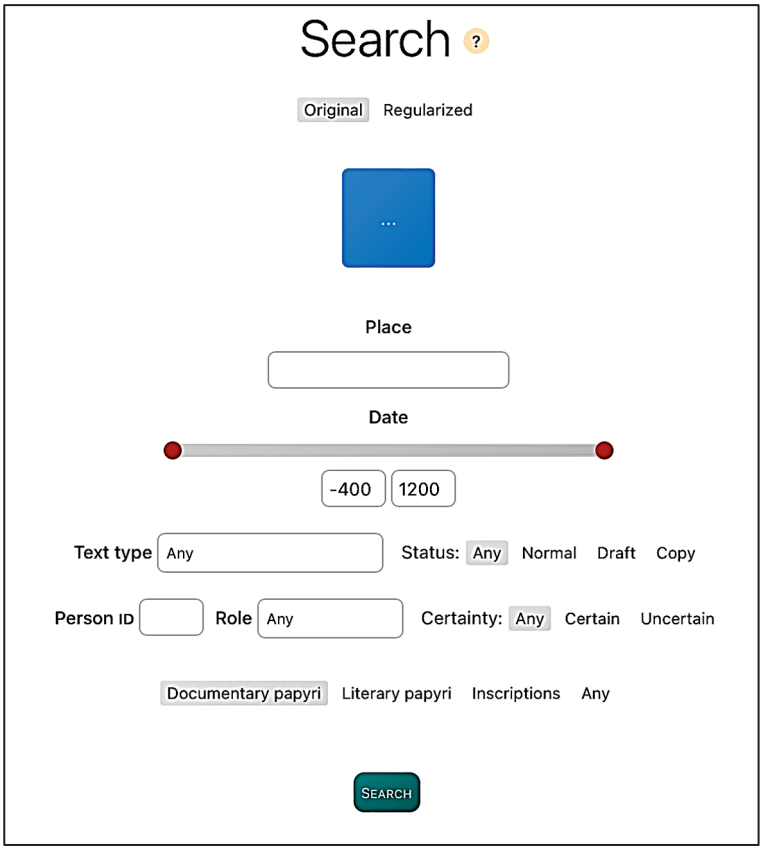

Fig. 5: A screenshot of the PapyGreek Search interface.

The results section (Figure 6) has three key elements. Firstly, it displays the results in a data table, where each retrieved word appears as a row accompanied by multiple data fields. These fields include links to the document on the PapyGreek platform, the sentence number and the word's position in it (showing its syntactic tree when clicked), the original and regularized forms of the word, morphosyntactic annotations, as well as date and place. The table includes automatic annotation data (see Section 3.2 above), along with the model's confidence score for them.

Secondly, the results section features a Timeline graph showing the number of results over 50-year intervals, with both absolute and relative frequencies displayed. Relative frequencies show the results as a percentage of the total word tokens in the DDbDP for each period. For the calculation, words from documents with uncertain dates spanning multiple 50-year intervals were distributed equally among these intervals.

Lastly, the interface allows exporting the results into a CSV file for further analysis outside the PapyGreek platform.

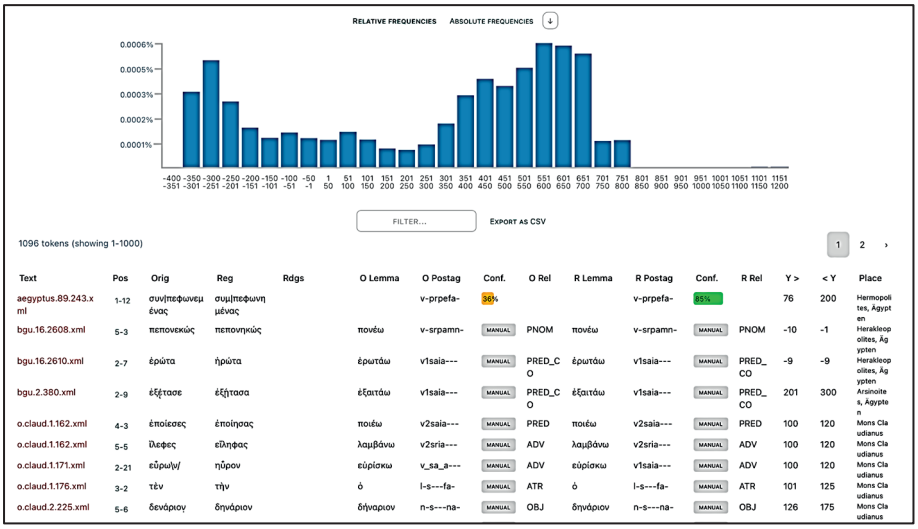

Fig. 6: Screenshot of the PapyGreek results section, displaying a data table of retrieved words and a Timeline graph showing results over 50-year intervals.

4.2 Query parameters

In the blue box located at the centre of the interface, query parameters are entered in the format: "parameter=value". The available parameters are the following: "form", "lemma", "lemma_plain", "postag", "relation", and "confidence".[48]

The `form` parameter is used for finding basic word forms and, using a special search syntax (discussed below in Section 4.3), also textual irregularities. This parameter ignores diacritics. For instance, to find word forms και in the uncorrected original text, one should select "Original" from the top version selection and enter "form=και" in the blue search box.

The parameters "lemma", "lemma_plain", "postag", and "relation" are utilized in morphosyntactic queries. The "lemma" and "lemma_plain" parameters allow for searching dictionary forms with and without diacritics, respectively. The "postag" parameter targets part-of-speech tags encoded as 9-character strings.[49] Automatic morphological annotations (see Section 3.2 above) can be optionally filtered out by using the "confidence" parameter within the range 0-1. For example, to require a score of 0.9 or above (indicating a high confidence), one would use "confidence=>0.9"; setting the value

48 In addition, there are two parameters, "order" and "depth", which are used in treebank queries and will be discussed below in Section 4.3.

49 For the list of available part-of-speech tag characters and their meaning, see https://github.com/gcelano/LemmatizedAncientGreekXML (accessed 12.9.2023).

to 1 is a shorthand for choosing only manual annotations. Lastly, the "relation" parameter targets syntactic relations annotated in the AGDT 2.0 scheme (see Section 3.2 above). To identify predicates, for instance, one would enter "relation=PRED".

The wildcard symbol % and underscore _ have a special meaning in the parameters listed above. % can replace any string of characters, while _ is used to skip a single character. For example, entering "form=κ_μη%" would return forms like κώμης and καμήλια. These symbols can be particularly useful with part-of-speech tags, such as in the query "postag=v%" to locate all verbs. As an alternative method for constructing complex queries, each parameter can optionally operate with regular expressions.[50] It can be used by adding the prefix "regex:" to the parameter name.

The parameters can be combined with a comma; for example, one can search for adverbial adjectives by typing "relation=ADV,postag=a%" in the search box.

4.3 Variation search

The linguistic variant search feature of PapyGreek Search utilizes the "form" parameter with four special symbols: + (plus) for editorial additions, - (minus) for deletions, > for left-hand context and < for right-hand context.[51] For instance, searching with "form=-ι" identifies all occurrences where an editor has deleted ι, whereas "form=+ι" finds cases of ι being added. For editorial replacements, a combination of - and + symbols is used. For example, to find instances where ε has been replaced with η, one would enter "form=-ε+η". The symbols > (before) and < (after) symbols are used to narrow down the search by word context. For example, "form=λ>-ε+αι" targets corrections from ε to αι following λ (in the same word).

For more precise criteria, textual variant queries can be further refined using regular expressions via the "regex:form=" parameter. When using the regex mode in this context, each of the included subqueries (for example, the parts after + or before <) must be determined with their own regex search pattern.[52] As an illustration, suppose one wishes to find all cases ω or ωι corrected to ου, but only when this correction appears word-finally. Using regex, this could be constructed as the following query: "regex:form=-^(ω|ωι)$+^ου$<^$".

The query consists of three parts: the string removed by the editor (-^(ω|ωι)$), the string that was added in its place (+^ου$) and the empty right-hand context (<^$). The |

50 E.g. Goyvaerts – Levithan 2012.

51 When conducting variation searches, the choice between "Original" and "Regularised" at the top of the interface is ignored.

52 Note that the symbols + and - are already reserved in the variation syntax and so cannot be used as quantifiers in the regex pattern. As a workaround, one can use the symbols ＋ (Full-width Plus, U+FF0B) and － (Full-width Hyphen-minus, U+FF0D). The PapyGreek Search system will transform these back into their normal regex counterparts + and -.

symbol denotes alternatives, and so ω|ωι means ω or ωι. The ^ and $ indicate beginnings and ends of strings, respectively, and are used here to restrict the search to exactly the given substrings. For example, by typing +^ου instead of +^ου$, the query would match all editorial additions *starting* with ου. Finally, the <^$ part specifies the empty word-final context.

4.4 Treebank search

PapyGreek Search has a graphical interface for building parametrized subtrees to explore syntactic dependencies. Users can add more search boxes to their query by clicking the "+" button at the bottom of an existing search box, which adds a new box underneath. This allows users to include as many boxes as needed to construct the desired tree structure.

In each box, any of the previously mentioned parameters can be used, including the special variation symbols detailed in Section 4.3 above. For example, to find (sub)trees where an object and some adverbial depend on a predicate, one would enter "relation=PRED" in the first box and add another box below it with "relation=OBJ". Then, one would add another box below the first box with the input "relation=ADV". This query is illustrated in Figure 7. Search boxes can also be moved horizontally using the left and right arrows found within each box.

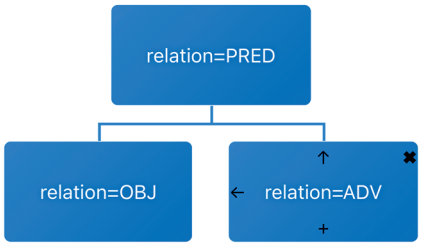

Fig. 7: Example query to find (sub)trees with an object and adverbial depending on a predicate.

The "depth" parameter enables specifying the tree depth in relation to the parent node, defaulting to 1 to indicate an immediate subordinate position. This parameter can be set to a numerical value, like 2, to skip a level, or an asterisk (*), to ignore depth and match all nodes under the specified parent.

Additionally, treebank queries can be refined by word order using the "order" parameter, which accepts any integer (e.g., 1, 2, 3, 4). This number indicates the word's relative position in the sentence, with lower numbers representing earlier positions. For instance, "relation=PRED,order=2" in one node, and "relation=OBJ,order=1" in another would match sentences where the predicate follows the object, as 2 > 1.

5 Example queries

This section presents brief examples of queries using PapyGreek Search, intended to illustrate the kind of linguistic research the tool can facilitate. First, we explore case variation in definite articles. Then, we address the representation of female protagonists in the papyri through the use of personal pronouns.

5.1 Case confusion or spelling error?

In another chapter of this book (Henriksson – Dahlgren – Vierros), we explore a case study on vowel variation in Greek related to /o, u/ allophonic variation in Egyptian. We observe that variations between ο /o/, ω /o(:)/, and ου /u/ often occur at the end of words, and note that the traditional explanation for case endings that contain these vowels involves the morphological merger of the genitive and dative cases. Here, we may take a different angle to this issue by examining noun phrases containing a definite article. If all words in the noun phrase follow the same case inflection, it lends more support to the case merger explanation, even though allophonic variation may have facilitated the merger.[53] We can also see if similar case variation occurs in feminine or plural noun phrases that contain different vowels. The singular masculine genitive and dative cases of the definite article, τοῦ /tu/ and τῷ/τῶι[54] /to(:)/, are relevant here, with identical forms in the neuter gender. In the feminine singular, the article is τῆς /te(:)s/ or /tis/ (gen) and τῇ /te(:)/ or /ti/ (dat). The accusative singular form adds another flavour to this soup.[55]

We can formulate a query for the definite article either using the lemma (ὁ for all genders), or the postag for the article, singular and masculine (l_s__m%), and combine it with the spelling variation as ου corrected by the editor to ω (-ου+ω).[56] This yields masculine articles where the original text is τοῦ and the regularized form is τῷ, resulting in 58 hits – a relatively low number. The same query for the neuter[57] adds another 22 hits. It is more convenient to combine the masculine and neuter queries by using the regular expression option, allowing us to include the optional iota adscript written in the original papyrus, without needing to specify gender at all in the postag.[58] When we

53 See, e.g., Stolk 2015 concerning the similarities of the semantic roles of the genitive and the dative affecting the case merger.

54 The *iota* in the dative ending was silent; in the papyri it was either not written at all (τω) or it was written as adscript, after the *omega* (τωι); the marking the *iota* below the *omega* as a subscript (τῳ) is merely an editorial habit that has no significance in respect to what was written on the papyrus. (Clarysse 1976; Horrocks 2010, 116; Vierros 2012, 121–36).

55 Horrocks 2010, 116.

56 https://papygreek.com/search/192 (accessed 28.2.2024).

57 Replacing 'm' with 'n' in the postag string.

58 Search input: "postag=l_s%,regex:form=-(ω|ωι)+ου".

change the spelling the other way around (+ου-(ω|ωι)) we get 120 hits, but not all of them are singular genitives in the regularised text; the results include also instances such as τοῦ corrected into τῶν, the plural genitive form. This can be avoided by adding a word-final position restriction (-(ω|ωι)<^$), but it is good to keep in mind also these forms concerning the variation between the vowels ου and ω. After the restriction for word-final position, we have 84 instances of τοῦ written instead of τῶ/τῶι.

How do we get to querying the noun phrases, then? As explained in Section 4.4 above, this is simple by adding more search boxes through the search interface. In one box, representing the head, we search for a word in a specific case, and in another box, representing the dependent, we specify its article, including any potential spelling variants. Since the treebanked corpus is small, we don't yet see many results for these queries. For example, a query for a head in the genitive case with a definite article as its modifier that has been written with a dative ending (Figure 8) yields only one result, upz.1.79, sentence 7 (l. 5): τω Ἄμμωνος, where the definite article looks like a dative, whereas the name of the god Ammon is in the genitive case. The genitive case would be correct here, but the two preceding words are correctly in the dative singular (the whole phrase being ἐν τῷ οἴκῳ τοῦ Ἄμμωνος "in the house of Ammon"). The writer may have written the ω in the definite article as a continuation of the cases from the previous words, or they may have aimed to write the genitive case but 'failed' because of its phonetic similarity with the dative case.

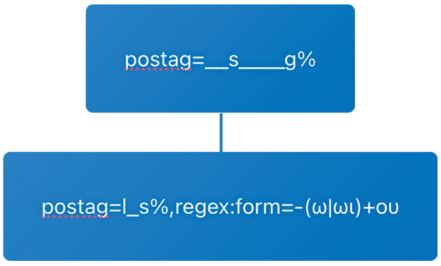

Fig. 8: A query with a head word in genitive singular and its dependent, an article in singular presenting spelling ω/ωι in the original papyrus that has been regularized into ου.

While waiting for the treebanked corpus to grow, we might continue this case study by reviewing the results for queries on article spelling variations and analyzing their linguistic contexts – does the article and the head have the same case, or is the nonstandard case of the article in disagreement with the head, and so forth. For instance, one of the first results from the article query above is bgu.3.998, which presents a similar case as the tree query in Fig. 8, τω τούτου υἱοῦ Ἁρπαήσιος (instead of τοῦ). The correct genitive with ου would definitely be expected also by analogy, since two out of three of the following genitives in the same noun phrase have those letters.

Repeating the query process for feminine articles and their head words would guide us further in in determining whether phonetics primarily drives the observed variation, or if the cases are indeed understood and used as semantically merged.

5.2 Gender by way of pronouns

Sociolinguistically and historically, there is interest in understanding the ways in which female protagonists are acting in the papyrological corpus. In addition to identifying female actors in papyri from prosopographical information provided by Trismegistos People or from the author and addressee metadata in the PapyGreek corpus, for example, we can also examine the use of personal pronouns using PapyGreek Search to understand the roles these pronouns play in the linguistic context.

For this preliminary study, we focus on the first-person pronoun. A basic lemma query ("lemma=ἐγώ") would retrieve all cases and genders of the 1st person pronoun. The same could be achieved with a postag query, which can also be further refined to search for specific genders and cases ("postag=p1s___f%" for pronoun, 1st person, singular, feminine). The search for feminine pronouns[59] brings us 243 hits. However, a caveat here is that this result does not present the whole corpus of papyri, but only the manually annotated part, where annotators have been able to mark as feminine those pronouns that actually do refer to women. In the Greek language, first and second person pronouns do not have gender-specific forms; this distinction only exists in third-person pronouns. The masculine bias in the training corpus is evident, as the automatic tagger has not assigned feminine gender to *any* first-person pronoun.[60] To illustrate this discrepancy, a query for masculine singular first-person pronouns gives 5296 hits, covering both manually and automatically tagged pronouns. So, we must continue our case study with the 243 instances where the feminine gender has been manually marked. The role these persons play in each text can already be considered important, as they represent the 'me' in the text. However, we can further look at the syntactic role of the pronoun either by using the relation tag (see Figure 9) or simply by using the grammatical case (nominative often expressing the subject, accusative usually being the case of the direct object and dative as the case of the indirect object or recipient). The relation tag, however, can be seen as more precise in identifying the actual roles of the pronouns. For instance, the subject might appear in cases other than the nominative, such as the accusative in *accusativus cum infinitivo* structures or the genitive in genitive absolute con-

59 https://papygreek.com/search/190 (accessed 28.2.2024).

60 One can question the linguistic correctness of marking the gender in the annotated corpus for pronouns that do not bear the marker for gender. However, in the annotation process we also select to mark other features requiring interpretation, like the grammatical case of neuter accusatives and nominatives, which look alike. In addition, this marking can help in research as we try to show here, as long as we are aware of the basis and procedure of the annotation.

structions, and cases can also be determined by other factors, such as prepositions. The distribution of the syntactic relations of the feminine first person pronouns is shown in Figure 10. In over half of the occurrences the 'me' in the text is the object, and more specifically the indirect object, since from the 124 instances of OBJ, 96 are in the dative case. The role as an attribute is given in roughly a third of the cases, which can indicate, for example, a possessive role marked by the genitive case. The low frequency of the subject role is not as alarming as it may sound, given that Greek embeds subject information within the verb's inflection and does not always use personal pronouns to express the subject. To query female subjects of verbs, we would also need the ability to combine female personal names from previous sentences, where their role as the subject is clear, along with possibly other methods.

```
lemma=ἐγώ,postag=p1s___f%,relation=SBJ%
```

Fig. 9: Query for the first person singular pronoun with the relation subject (including coordinated subjects).

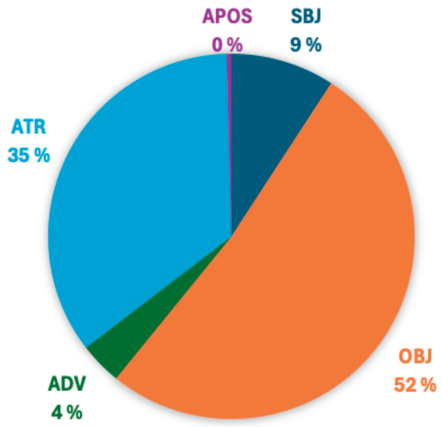

Fig. 10: Syntactic relations of the feminine first person pronouns.

6 Conclusion

We have introduced PapyGreek Search, an online open-access query tool for documentary Greek papyri. PapyGreek Search stands out for its integrated search interface, en-

abling simultaneous queries across linguistic domains, such as morphology, syntax, and, indirectly through editorial interventions, phonological and morphosyntactic variations. The treebank search feature of PapyGreek Search includes a graphical interface for query construction, making it accessible to non-technical users. In this chapter, we outlined the motivation for creating the tool, described its implementation in detail, and provided a walkthrough of its main features. Additionally, we demonstrated how the tool can be used for linguistic research with two example queries, addressing potential case confusions and the use of genders in personal pronouns.

Bibliography

Baumann, R. (2013), *The Son of Suda On-Line*, in *The Digital Classicist*, ed. by S. Dunn – S. Mahony, https://ryanfb.xyz/papers-BICS/sosol-bics-draft.pdf.

Broux, Y. – Depauw, M. (2015), *Developing Onomastic Gazetteers and Prosopographies for the Ancient World Through Named Entity Recognition and Graph Visualization: Some Examples from Trismegistos People*, Social Informatics 8852, 304–313.

Clarysse, W. (1976), *Notes On The Use Of The Iota Adscript In The Third Century B.C.*, CdE 51, 150–66.

Depauw, M. – Keersmaekers, A. (2017), *Bringing Together Linguistics and Social History in Automated Text Analysis of Greek Papyri*, Classics@ 20, https://classics-at.chs.harvard.edu/bringing-together-linguistics-and-social-history-in-automated-text-analysis-of-greek-papyri.

Depauw, M. – Stolk, J. (2015), *Linguistic Variation in Greek Papyri: Towards a New Tool for Quantitative Study*, GRBS 55, 196–220.

Dickey, E. (2011), *The Greek and Latin Languages in the Papyri*, in *The Oxford Handbook of Papyrology*, ed. by R. Bagnall, Oxford, 149–69.

Evans, T. V. – Obbink, D. D. (2010), *The Language of the Papyri*, Oxford.

Gignac, F. T. (1976), *A Grammar of the Greek Papyri of the Roman and Byzantine Periods, Volume 1: Phonology*, Milan.

Goyvaerts, J. – Levithan, S. (2012), *Regular Expressions Cookbook*, Sebastopol (CA).

Gusfield, D. (1997), *Algorithms on Strings, Trees, and Sequences*, Cambridge.

Horrocks, G.C. (2010), *Greek: A History of the Language and Its Speakers*, 2nd ed., Oxford.

Karwin, B. (2010), *SQL Antipatterns: Avoiding the Pitfalls of Database Programming*, Pragmatic Bookshelf.

Keersmaekers, A. – Van Hal, T. (2023), *Creating a Large-Scale Diachronic Corpus Resource: Automated Parsing in the Greek Papyri (and Beyond)*, Natural Language Engineering 2023, 1–30.

Keersmaekers, A. – Mercelis, W. – Swaelens, C. – Van Hal, T. (2019), *Creating, Enriching and Valorizing Treebanks of Ancient Greek*, "TLT, SyntaxFest" 2019, 109–17.

Krause, T. – Zeldes, A. (2016), *ANNIS3: A New Architecture for Generic Corpus Query and Visualization*, Digital Scholarship in the Humanities 31, 118–39.

Martens, S. (2013), *TüNDRA: A Web Application for Treebank Search and Visualization*, in *Proceedings of The Twelfth Workshop on Treebanks and Linguistic Theories (TLT12)*, Sofia, 133–44.

Mayser, E. – Schmoll, H. (1970), *Grammatik Der Griechischen Papyri Aus Der Ptolemäerzeit. Laut- Und Wortlehre: Einleitung Und Lautlehre*, Berlin.

Van Minnen, P. (1996), *The Duke Data Bank of Documentary Papyri*, https://library.duke.edu/papyrus/texts/DDBDP-old.html.

Myers, E. W. (1986), *An O(ND) Difference Algorithm and Its Variations*, Algorithmica 1, 251–66

Pajas, P. – Štěpánek, J. (2009), *System for Querying Syntactically Annotated Corpora*, in *Proceedings of the ACL-IJCNLP 2009 Software Demonstrations*, Singapore, 33–6.

Ratcliff, J. W. – Metzener, D. (1988), *Pattern Matching: The Gestalt Approach*, Dr. Dobb's Journal, July 1988, 46.

Rosén. V. *et al*. (2017), *Exploring Treebanks with INESS Search*, in *Proceedings of the 21st Nordic Conference on Computational Linguistics*, Gothenburg, 326–9.

Singh, P. – Rutten, G. – Lefever, E. (2021), *A Pilot Study for BERT Language Modelling and Morphological Analysis for Ancient and Medieval Greek*, in *Proceedings of the 5th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, 128–37.

Sosin, J. (2010), *Digital Papyrologyi*, https://blog.stoa.org/archives/1263.

Stolk, J. V. (2015), *Dative by Genitive Replacement in the Greek Language of the Papyri: A Diachronic Account of Case Semantics*, Journal of Greek Linguistics 15, 91–121. [https://doi.org/10.1163/15699846-01501001]

Vierros, M. (2012), *Bilingual Notaries in Hellenistic Egypt: A Study of Greek as a Second Language*, Brussel.

Vierros, M. (2018), *Linguistic Annotation of the Digital Papyrological Corpus: Sematia*, in *Digital Papyrology II. Case Studies on the Digital Edition of Ancient Greek Papyri*, ed. by N. Reggiani, Berlin – Boston, 105–18.

Vierros, M. – Henriksson, E. (2017), *Preprocessing Greek Papyri for Linguistic Annotation*, in *Journal of Data Mining and Digital Humanities. Special Issue on Computer-Aided Processing of Intertextuality in Ancient Languages*, ed. by M. Büchler – L. Mellerin, http://jdmdh.episciences.org/paper/view/id/1385.

Vierros, M. – Henriksson, E. (2021), *PapyGreek Treebanks: A Dataset of Linguistically Annotated Greek Documentary Papyri*, Journal of Open Humanities Data 7, https://doi.org/10.5334/johd.55.

Vierros, M. – Yordanova, P. (2022), *Querying Syntactic Constructions in Ancient Greek Parsed Corpora: A Case Study on the Genitive Absolute in Literature and Documentary Papyri*, Classics@ 20, https://classics-at.chs.harvard.edu/querying-syntactic-constructions-in-ancient-greek-parsed-corpora-a-case-study-on-the-genitive-absolute-in-literature-and-documentary-papyri.