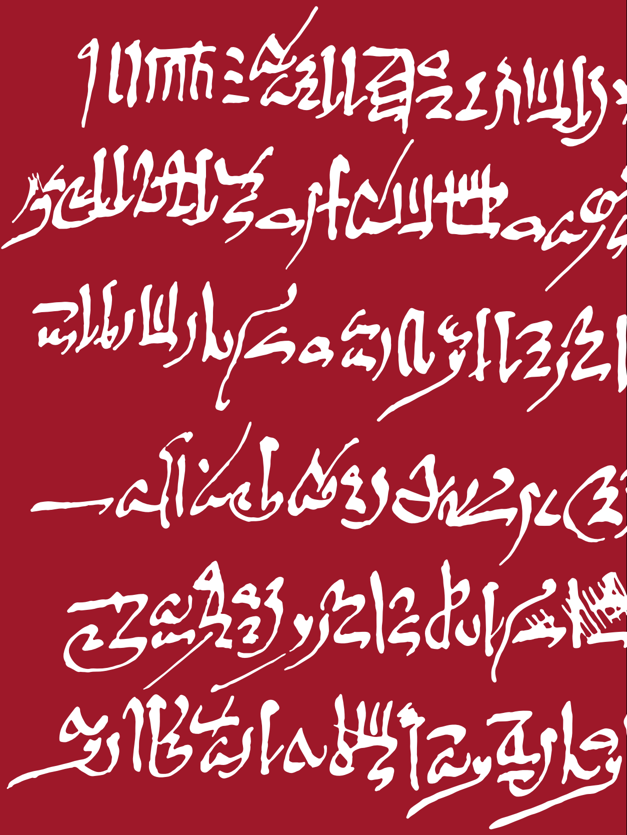
Hieratic Studies Online 5 | 2023



Julius A. Tabin

Optical Character Recognition Applied to Hieratic. Sign Identification and Broad Analysis

ISSN 2512-6598

HSO Hieratic Studies Online

Hieratic Studies Online is a double-blind peer-reviewed, academic series dedicated to presenting research on all aspects of Hieratic and cursive hieroglyphs.

Hieratic Studies Online is edited by
Svenja A. Gülden, Tobias Konrad, Kyra van der Moezel and Ursula Verhoeven

ISSN: 2512-6598

Akademie der Wissenschaften und der Literatur | Mainz
hso@uni-mainz.de
https://aku.uni-mainz.de/hieratic-studies-online/

Optical Character Recognition Applied to Hieratic

Sign Identification and Broad Analysis

Julius A. Tabin

University of Chicago

Abstract: Despite modern advances in digital paleographic techniques, hieratic has largely eluded the thorough application of technological methods for automatic sign recognition. To remedy this, this article presents two new hieratic facsimiles (of the Shipwrecked Sailor and part of the Eloquent Peasant), a novel data set of 13,134 individual hieratic signs, and an Optical Character Recognition (OCR) based program developed to analyze them using an Image Deformation Model. This program is highly accurate and various applications are presented, from single character identification to large-scale sign comparisons. This article provides an important building block in the study of hieratic digital paleography, allowing far more signs to be compared at once than ever before, and offers a free, open-source tool and data set.

1. Introduction

In recent years, the study of ancient Egyptian hieratic has greatly benefited from digital paleographic methods.[1] Never before has it been easier to quickly search for hieratic signs, transmit information, and view texts. However, two major areas still present significant challenges: character recognition and large scale comparison.[2]

Historically, the recognition of hieratic characters has been slow, often entailing a new scholar to all but memorize Möller's paleography[3] or similar works. Even online, searchable hieratic paleographic compositions still require a significant time investment, especially when a character is uncommon and a researcher may not know what they are even searching for (a reader may contrast this with the *Demotic Palaeographical Database Project*, which is even searchable through drawing).[4]

[1] This article is adapted from the first part of my B.A. thesis for the Department of Near Eastern Languages and Civilizations at the University of Chicago. I would like to thank my advisor, Dr. Brian Muhs (Associate Professor of Egyptology, University of Chicago), as well as Dr. Luiza Osorio G. Silva, for their assistance with the thesis text. I also would like to thank Dr. Christian Casey and Dr. Mark-Jan Nederhof for their assistance with some of the important code sections that made this work possible.

[2] Berg, and Donker van Heel 2000, 39.

[3] Möller 1909b.

[4] DPDP n.d. Available at http://129.206.5.162/beta/index.html.

Beyond this, paleographic work in hieratic has always been limited in some way, not by oversight, but by necessity. For instance, the large sign identification works, such as that of Möller[5] or the far more extensive and modern *AKU-PAL*,[6] include a limited number of each sign to try and catalog the variation present in the corpus. Other studies look at greater variation in a language, but must restrict the data set to specific issues or texts.[7] This system has worked well in the past, but some questions remain out of reach.

For example, one interested in the paleography of the writing system as a whole would be unable to adequately investigate this, given the sheer amount of data that would need to be examined. In addition, the exact statistical similarity between signs over the corpus would be difficult to identify. This latter issue is one that remains for many current projects not focusing on hieratic that nevertheless have made substantial progress towards cataloging variation in their respective scripts, such as the *Demotic Palaeographical Database Project*,[8] *Digi-Pal*,[9] and *HebrewPal*.[10] This is due to the area around glyphs being cut out and annotated, rather than the glyphs themselves. However, these monumental questions do not need to remain unattainable. In fact, the state of the field, having more and more technological influence and statistical power, demands that they be addressed.

Here, I present a method for both hieratic character recognition and large scale comparison, in the form of an Optical Character Recognition program. I aim to demonstrate that this program, supplemented by a large, novel data set, can automatically identify hieratic characters to a high degree of accuracy. The recognition of these characters can then be leveraged to create comparisons of wide swathes of the data set, presenting a new opportunity for the field of hieratic digital paleography.[11]

2. Optical Character Recognition

When it comes to integral programs for the digital study of paleography, there are few more important than Optical Character Recognition (OCR) programs. In brief, OCR programs convert physical writing into a machine-readable format. This can take many forms and has wide ranging applications in a myriad of disciplines. The types of algorithms used for OCR range from simple pixel comparisons to complex machine learning models, but the question they intend to answer is the same: when given a written character, what is its identity?[12] When applied to ancient material, OCR programs can be used to automatically identify characters and, when trained enough, even make inferences about partial characters, notably taking some of the guesswork

[5] Möller 1909b.
[6] AKU-PAL 2022. Available at https://aku-pal.uni-mainz.de.
[7] Aguizy 1986, 67–70; Quirke 2011.
[8] DPDP n.d. Available at http://129.206.5.162/beta/index.html
[9] DigiPal 2011–14. Available at http://www.digipal.eu.
[10] HebrewPal 2022. Available at https://www.hebrewpalaeography.com.
[11] For an earlier suggestion of the potential of this method, see Gülden, Krause, and Verhoeven 2020, 640–641.
[12] Memon et al. 2020, 142642–142643.

out of identifying ones that are unusual or partially preserved. A researcher will still have to make a judgment call of whether to accept or reject the program's suggestion for any given character, but OCR methods at least provide a reproducible baseline. Also, OCR programs can allow for the rapid digitization of texts, as all one needs to do is input a text and let the machine encode the identities of all of the characters.

Optical Character Recognition has been used in Egyptology for over a decade, often to great effect.[13] However, the goals of most of the previous OCR programs in the field have started and ended with the aforementioned digital identification of characters and digital transcribing of texts. This is important work, but OCR is not limited to this use. For nearly all OCR algorithms to identify a character, the program must look at a database of previously correctly identified characters to inform the new decision. When determining what an unknown character is, the program uses the database to rank its options and decide upon the best one. Directly or indirectly, this results in comparing the input character to the characters in the data set, often accompanied by a "similarity" score, which is a number determined by the program that describes how similar two images are.[14] Although this is usually thought of as a means to the end of identification, the program's insights into the similarity of various characters can be used to learn more about the characters and the texts they came from. Leveraging these similarity statistics provides a new way to look at ancient texts, allowing for complex comparisons to be made between characters, texts, handwritings both ancient and modern, locations, and time periods with more statistical power than has ever been possible before.

In the field of Egyptology, this method of looking at similarity scores to learn about ancient material has not been significantly attempted. However, that does not mean that OCR has been ignored. There has been significant work already on using OCR on images of hieroglyphs, a challenging problem due to the physical dissimilarity of hieroglyphs to most other writing systems.[15] Nevertheless, these methods cannot be easily adapted to hieratic, particularly because they are constructed for carved, rather than written, material.

In 2015, Nederhof[16] used OCR to digitize Sethe's *Urkunden der 18. Dynastie*.[17] In the paper, Sethe's individual handwritten characters are automatically detected by considering "blobs", defined as a connected set of black pixels. Then, each unknown glyph is compared to a set of "prototypes", a subset of the full data set of identified characters that gives an approximation of the total variation. Prototypes were used to cut down on computational costs, given that it would be costly to compare each input to the whole data set. An Image Deformation Model (IDM) was used to fully compare the sign to the filtered data set signs, resulting in a difference score (effectively the reverse of a similarity score) that is then used to determine the identity of each sign. Through this method, *Urkunden der 18. Dynastie* was able to be digitized.

[13] Nederhof 2015; Franken, and Gemert 2013.
[14] Koch, Zemel, and Salakhutdinov 2015, 9.
[15] Franken, and Gemert 2013, 766; Elnabawy, Elias, and Salem 2018.
[16] Nederhof 2015.
[17] Sethe 1927.

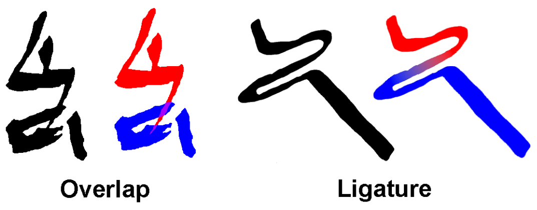


Figure 1: An example of overlapping signs (left) and ligatured signs (right). In each example, one sign is colored red and the other is colored blue, with shared areas in purple. Particularly in the overlap example, the exact extent of overlap is uncertain, this being one possibility. Both examples come from the Shipwrecked Sailor (P. Hermitage 1115).

Nederhof's research is unique in the field and it provides an excellent starting point for subsequent work. However, it is not without its limitations as well. Nederhof's focus on Sethe's *Urkunden der 18. Dynastie* puts his tool's use firmly within recognizing handwritten transcriptions of ancient Egyptian, making it unable to be applied to the actual texts themselves. Furthermore, the employment of "prototypes", while useful in cutting down computational costs, would not be ideal when one wants to use the difference scores for large-scale morphological comparisons. This is because comparisons across texts, time periods, or locations would likely need a large amount of data to be significant. However, these types of comparisons are not even worth considering for Nederhof's data set, given that his program is looking at modern transcriptions and not the ancient material itself, nor morphologically accurate reproductions.

OCR has been used on images of hieroglyphs and on modern hieroglyphic transcriptions, but not on hieratic at a large scale. This is chiefly due to the numerous additional problems hieratic poses compared to hieroglyphs. First, hieratic characters are often not distinct from one another and can be ligatured together or can be overlapping (Figure 1). This can cause problems even for a human analysis of a text, necessitating the use of context clues. Although it is sometimes easy for humans to mentally separate two signs or recognize a ligature, it is far more difficult for a program to do so. Nederhof mentions in the end of his paper that the touching of hieratic signs poses a problem to his blob-based automatic detection.[18] Both hieroglyphs and written transcriptions rarely have this problem. Second, many hieratic characters look nearly identical to one another (see Figure 8 for an example). This is true for some characters in *Urkunden der 18. Dynastie*, but not nearly to the same degree as in hieratic. Third, hieratic can be far more variable than hieroglyphs, with multiple ways of writing the same sign being present even in the same text (see Figure 12 for an example). This element of hieratic makes it particularly apt for morphological comparisons across space and time, but also leads to difficulties in automatic recognition. Relatedly, hieratic can be written vertically or horizontally, further increasing variation. While this is also true for hieroglyphs, the change in orientation does not usually affect the form of the hieroglyphic signs. Hieratic, on the other hand, has different ligatures and overlaps depending on whether the signs are written next to one another or on top of one another.

[18] Nederhof 2015, 12.

All of these problems are solvable with a large data set. A data set that is sufficiently large would provide the training necessary for an algorithm to identify ligatured characters and even perhaps separate overlapping ones. It would also allow a sufficiently powerful algorithm to distinguish between similar characters, possibly with even greater accuracy than a person could. Lastly, it would capture close to the full variation of hieratic and make sure few signs go unrecognized. Cursive scripts that are somewhat visually similar to hieratic, such as Urdu with its ligatures and overlaps, have also been shown to work with OCR methods and large data sets.[19] As a point of reference, Franken and van Gemert used about 4,000 images for their 2013 hieroglyphic recognition paper.[20] Additionally, a limited set of 39 hieratic character classes was demonstrated by Bermeitinger *et al.* to be amenable to OCR.[21] Haliassos *et al.* also successfully performed automatic recognition using multiple hieratic papyri, but with only three character classes.[22] Given these promising results and the aforementioned unique challenges hieratic poses, I present here a hieratic data set more than triple the size of Franken and van Gemert's paper in the interest of recognizing almost 30 times the amount of character classes of Bernhard *et al.*

Even with a sufficient data set, there are a number of pitfalls and limitations of using OCR on hieratic that must be avoided or at least acknowledged. For instance, to accurately identify hieratic characters at the moment, one needs to use facsimiles because the variations in the damage of the material, the ink darkness, and image quality would likely be too much for any image recognition software, especially on top of all of the other variations present in hieratic. It is not unusual to try and remove unhelpful variation before using a data set for OCR; Nederhof[23] made Sethe's glyphs purely black and white for his OCR program and Franken and van Gemert[24] used black and white images of the hieroglyphs in the pyramid of Unas for their work. One could argue that, rather than use a facsimile, one should try and just color the hieratic images so the glyphs are purely black and the background is purely white, but doing so would effectively be creating a facsimile. Because facsimiles are being used, some data will necessarily be lost and, while this is good because one wants to lose some of the aforementioned negative variation, it could also be dangerous because a facsimile maker could unwittingly lose an important part of the variation. This process of making a facsimile is subject to human decision making and, thus, human error. Facsimile use also introduces variation because using facsimiles made by two different people could result in differences being detected due to the facsimile maker, not the original, underlying hieratic. However, using facsimiles is the only practical choice without the presence of an impossibly large data set.[25] In addition, some of these potential issues with facsimiles can be monitored. For example, if glyphs from one facsimile maker are shown to be more similar to each other, regardless of which text they are from, rather than showing similarities to ones

[19] Naz et al. 2014.
[20] Franken, and Gemert 2013, 765.
[21] Bermeitinger, Gülden, and Konrad 2021.
[22] Haliassos et al. 2020.
[23] Nederhof 2015, 3.
[24] Franken, and Gemert 2013, 766.
[25] For a detailed description of the considerations to take into account when making digital facsimiles, as well as the general importance of creating facsimiles see Berg, and Donker van Heel 2000, 39–42 and Moezel 2022, 19–24.

from their same original text by a different facsimile maker, that would raise some red flags regarding this modern influence.

3. Sources for the Data Set

The data set used for this project consists of individually cut out hieratic characters from facsimiles made by myself (henceforth referred to as "Tabin's facsimile"), Georg Möller in his "Hieratische Lesestücke",[26] and William Poe in his "Introduction to Hieratic Middle Egyptian".[27] My facsimiles and Poe's were already digital, whereas Möller's had to be digitized through high quality scans. All of the facsimiles were color-corrected to be purely black and white.

I created two facsimiles by hand, one of the full story of The Shipwrecked Sailor (P. Hermitage 1115) and one of the beginning of The Eloquent Peasant (P. Berlin P. 3023). These texts were selected for three reasons. First, they are both lengthy, with many individual signs, resulting in a multitude of data points each. Second, they both have lots of published information about them, including clear images. Third, they both have versions done by other facsimile makers: Poe created a full Shipwrecked Sailor facsimile and Möller created a partial Peasant facsimile. These factors allow for easier creation and many potential comparisons. My facsimiles were created, using methods discussed below, to be as morphologically accurate to the original texts as possible. They were created in Photoshop using images of the original papyrus digitized from Golénischeff's Shipwrecked Sailor publication[28] and Parkinson and Baylis's Eloquent Peasant publication[29] respectively. These facsimiles can be found in the appendix (page 41).

In contrast to my facsimiles, Poe's Shipwrecked Sailor facsimile is less focused on being morphologically accurate, opting instead to be a simplified teaching tool for Middle Egyptian hieratic. The individual signs are far more block-like and smooth, without the sharper points that appear in hieratic made from physical brush strokes. Poe produced his facsimile by scanning Golénischeff's Shipwrecked Sailor images and then tracing them on a computer in CorelDRAW™, a digital image editing software.[30] Thus, Poe's facsimile still maintains the basic shape of each character, but disregards the minutiae of each sign's detail. This is useful in multiple ways. For example, if a sign from Poe's facsimile is put into the program, the program should output what signs are most similar to that one. If the signs that are most similar are all from his facsimile, even if a different version of that exact sign is in the data set from a different facsimile maker, that is a good indication that the general shape is not enough to determine a specific sign. If the reverse is true and the same sign from Poe's facsimile and Tabin's facsimile are shown to be most similar, then it indicates that larger shapes of signs are

[26] Möller 1909a, 1–18, 21–25.
[27] Poe 2008, 212–215.
[28] Golénischeff 1913, pl. I–VIII.
[29] Parkinson, and Baylis 2012, 42–44.
[30] Poe 2008, iv.

most important. In this vein, Poe's facsimile can be used to investigate questions relating to modern facsimiles and how accurate they need to be to capture the true variation of hieratic. It is also a good addition to the data set in a practical sense because someone using the program on their own facsimile might not have produced that facsimile with in-depth morphological accuracy in mind; the more variation present in the data set, the better it will be at recognizing foreign inputs. Due to time constraints, only the first four pages of Poe's facsimile were added to the data set. These pages overlap in content with my own Shipwrecked Sailor facsimile.

Möller, who cataloged hieratic morphological variation over 100 years ago, produced a number of high-quality facsimiles spanning a wide range of genres, locations, and time periods. For this project, only his facsimiles from "Hieratische Lesestücke für den akademischen Gebrauch Vol. 1" were used because they are all written in Middle Egyptian and can all be dated to around the Middle Kingdom.[31] Möller produced his facsimiles by drawing them on photographs or gelatin drawings, while constantly comparing the facsimiles to the originals. This process produced facsimiles that are fairly morphologically accurate, especially since Möller marked all of the damaged areas. Although Möller did make some cosmetic or otherwise personal alterations to the glyphs beyond what was present in the originals, the results of my program (see below) show that the data is able to nevertheless provide accurate character recognition. Having a gradient of facsimile accuracy (Tabin > Möller > Poe) is a benefit in this work, since facsimiles (of various qualities) will likely be the main source of additional data for the program's foreseeable future. I used almost all of Möller's facsimiles, although a few were left out due to time constraints. Most of his facsimiles are only excerpts from each text, rather than the whole of each. The Möller facsimiles used were those of the Hatnub texts, Lahun temple files, Will of Wah, Hymn to Senwosret, Eloquent Peasant, Papyrus Prisse, Sinuhe, Papyrus Ebers, the Rhind Mathematical Papyrus, and Papyrus Westcar. Of these, his Eloquent Peasant facsimile overlaps with mine to a certain extent.

4. Methods

In the interest of space, only the details of the Optical Character Recognition program will be discussed here at length. For in-depth information about the methods for the creation of the data set of 13,134 individual signs and ligatures, a reader is encouraged to read pages 30–35 of my BA thesis.[32] This includes the steps for the creation of the Tabin facsimiles, as well as the development and usage of Sobti, a program produced in collaboration with Dr. Christian Casey for cutting out individual signs from annotated facsimiles.

In brief, the Tabin facsimiles were created by tracing papyrus images in Photoshop using a Wacom Intuos Pro (Paper Edition) tablet connected to the computer. In tracing the signs, obvious damage was repaired, as is common for most facsimiles. For instance, if a sign had a hole in the middle where the papyrus was clearly ripped, but the rest of

[31] Möller 1909a.
[32] Tabin 2022, 30–35.

the sign was known, the hole would be ignored in the tracing. As for signs with heavier damage, where significant extrapolation would be needed to restore the sign, the signs were still restored to the best of my ability for the Shipwrecked facsimile, but often not for the Peasant facsimile. This was purely due to time constraints. The restorations were done by referencing published transcriptions of the texts,[33] common sign forms,[34] and by copying non-damaged examples of the damaged signs from the same text. Then, an outline of what the sign would be expected to look like was created. Any deviations from what is actually on the page were recorded in a "damage" layer in Photoshop and heavily damaged signs were not considered in the dataset. The facsimiles were each done twice in their entirety, to maximize my familiarity with the text and, thus, my accuracy.

The Tabin, Möller, and Poe facsimiles were then manually annotated using transparent, colored polygons in Photoshop, each around a different sign. For ligatures or overlaps where it was uncertain where one sign ended and another began, they were taken as one sign and put in the same annotated polygon. Each individual line of text was also annotated in a unique polygon. This annotation information was then loaded into Sobti and the signs were automatically cut out and saved as black glyphs with white backgrounds. At the end of the processing, each character had its own black and white rectangular image.

4.1. Glyph Labeling

After the individual glyphs were extracted by Sobti, they were manually renamed with their Gardiner sign code and how many times that sign had appeared in the text in the following format: [Sign Code]_XXXX.png. As a concrete example, the third A1 sign to appear in a text would be labeled "A1_0003.png".

Some signs, which are clearly different in their hieroglyphic forms, are not different in their hieratic forms. This led Möller to consider many of them as one sign form.[35] I have followed Möller's lead. For example, signs U6 (𓌸) and U7 (𓌻) are both hoe signs and are distinct in hieroglyphic form: U6 is tilted upwards whereas U7 is horizontal.[36] However, in hieratic, this distinction is never made. Therefore, all of the signs of that form could be U6, they all could be U7, or there could be a mix, but the signs are indistinguishable and this cannot be known. Because of this, I have taken every sign that could be U6 or U7 as U7. This was done for many signs where this problem occurs, given that there is no known way to tell the difference and it would be introducing personal bias to make the distinctions myself. Some more examples of signs dealt with in this way are: A40/A41 (𓀭/𓀮) was taken as A40, G40/G41 (𓅮/𓅯) was taken as G41, N11/N12 (𓇹/𓇺) was taken as N11, T9/T9A (𓌔/𓌕) was taken as T9, U28/U29 (𓍑/𓍒) was taken as U29, W17/W18 (𓏃/𓏅) was taken at W17, Z9/Z10 (𓏴/𓏵) was taken as Z9, and Aa15/Aa16

[33] Casey 2008, 3–13; Parkinson 1991, 10–22; Nederhof n.d.
[34] Nagai et al. 2021.
[35] Möller 1909a.
[36] Hieroglyphic unicode keyboard provided by Dr. Christian Casey from https://www.caseyegyptologist.com/downloads.

(𓐝/𓐞) was taken as Aa15. If a sign does not appear in the data set, there could be no examples of it in the texts used, but it could also be wrapped up under the umbrella of another sign that was impossible to distinguish from it.

The last quirk of the labeling method to note is that there was no Gardiner sign present for the beginning of a cartouche.[37] Traditionally, sign code V10 is used for the whole cartouche and V11 is used for just the end. Since cartouches in hieratic are often written split up, sign code V10 was here used for cartouche beginnings, rather than the whole cartouche.

This overall labeling method requires prior knowledge of what each sign is. This was determined manually, supplemented with the *Hieratische Paläographie DB*[38], the *Thesaurus Linguae Aegyptiae*,[39] the JSesh texts list,[40] the Nederhof St. Andrews Corpus of Egyptian text transliterations,[41] and the Stableford Corpus of transliterations of Möller's work on hieratic.[42] The signs were then checked against published versions of the texts: the Collier and Quirke publication of the Lahun papyri,[43] the Parkinson publication of the Eloquent Peasant,[44] the Koch publication of the Story of Sinuhe,[45] the Žába and Jéquier publications of Papyrus Prisse,[46] the Golénischeff and Casey publications of the Shipwrecked Sailor,[47] the Wreszinski publication of Papyrus Ebers,[48] the Erman and Blackman publications of Papyrus Westcar,[49] and the Chace publication of the Rhind papyrus.[50] In the future, this could be improved with OCR. Adding an OCR program would identify most of the characters with high accuracy, preventing a user from having to manually input each sign code. Instead, a user would only have to correct any errors. This function could even be wrapped up in Sobti, fusing the two programs. A similar method was used by Nederhof in his work on identifying Sethe's glyphs through OCR; as more images were added to the data set, OCR was able to be relied on for identifications more and more.[51]

Lastly, "tags" were added to every image. These tags are numbers put at the end of the file names, corresponding to the facsimile maker, provenance of the text, and the text itself, in that order. These tags are given in Table 1. For an example of how these tags work, the 21st A1 from Möller's facsimile of the Will of Wah would be named

[37] Gardiner 1957.
[38] Nagai et al. 2021.
[39] TLA 2014. Available at https://aaew.bbaw.de/tla/servlet/TlaLogin.
[40] Rosmorduc 2014.
[41] Nederhof n.d.
[42] Stableford n.d.
[43] Collier, and Quirke 2004, 16–19.
[44] Parkinson 1991, 10–22.
[45] Koch 1990, 1–81.
[46] Žába 1956, 15–65; Jéquier 1911, pl. II, V–VII, IX.
[47] Golénischeff 1913, 1–8; Casey 2008, 17–27.
[48] Wreszinski 1913, 1–228.
[49] Erman 1890, 22–72; Blackman 1988, 1–17.
[50] Chace 1927, 50–119.
[51] Nederhof 2015, 5.

Table 1: The tags used for the data set images

Number	Facsimile Maker	Provenance	Text
1	Möller	Thebes	Shipwrecked Sailor
2	Poe	Lahun	Eloquent Peasant B1
3	Tabin	Hatnub	Eloquent Peasant R
4		Unknown	Sinuhe B
5			Sinuhe R
6			Papyrus Prisse
7			Hymn to Senwosret III
8			Lahun Temple Files
9			Will of Wah
10			*Texte aus Hatnub*
11			Papyrus Ebers
12			Rhind Papyrus
13			Papyrus Westcar

“A1_0021_1_2_9.png”. This system of labeling was chosen so the data for each sign could be readily accessible when the glyph was being used in comparisons.

4.2. The Comparison (OCR) Program

The purpose of the OCR program is to identify input hieratic characters based on their morphology, ranking the data set characters based on similarity. Although the goal is simple, this is by no means a trivial task. Throughout the creation of the OCR program, written in Python, a balance had to be struck between accuracy and speed. One could make an extremely accurate program that compares an input sign to the entire data set, but that would take far too long to be feasible as a research tool. Conversely, a program that randomly ranks the signs in the data set would be extremely fast at doing so, but would be immensely inaccurate and profoundly unhelpful. Thus, in many areas of this project, tradeoffs had to be made. However, in some cases, speed could be improved without sacrificing accuracy.

4.2.1. Calculation of Data Set Metrics

Before any input glyphs are added to the program, the program calculates aspect ratios and vectors of pixel values for the entire data set and saves them as a file. This only needs to happen once, because the metrics from the unchanging data set can then be called by the program, rather than being calculated each time it is run.[52] To do this, a list of all of the images in the data set is loaded into the program. Using the Python Imaging Library (PIL) package, each image is loaded in and its aspect ratio is calculated by dividing its height in pixels by its width in pixels. The calculated aspect ratios are saved in a dataframe along with the image filenames and the sign codes extracted from

[52] Of course, this would need to be redone every time new material is added to the data set.

those filenames. This dataframe is then saved as a .csv file in a newly created folder that can be easily accessed by the program.

After this, the images are all resized to be squares of a certain size. The images need to be resized so they can be compared to one another easily; it would be difficult to compare images with wildly different aspect ratios and resolutions. Each image will be distorted to some degree, but signs that looked similar before distortion should look similar after distortion, since they are all undergoing the same transformation. The resizing size can be determined by a user, but, through extensive trials, 20 px by 20 px seems to be the best size to optimize speed without meaningfully sacrificing accuracy. The images then undergo a simple binarization process to change them from grayscale to purely black and white. Once every image is resized, a list of its pixels is saved as a vector and added to a dataframe, next to a column of the sign names of each image. For example, an image that is 2 px by 2 px (much smaller than the ones used in this program) that has a black top-left pixel, a white top-right pixel, a white bottom-left pixel, and a black bottom-right pixel, would have a vector of [0, 255, 255, 0] (i.e. [black, white, white, black]). This method can be extended to images of any size. Finally, the dataframe of pixel values for every resized image from the data set is saved as a .csv file in the same folder as the aspect ratio list.

4.2.2. Filtering by Aspect Ratio

Once the data set metrics are all saved, one can load in a new image that one wants to identify. A user can also put in a whole folder of images to be identified, making large-scale comparisons easy to do. When a new sign is added to the program, it undergoes pre-processing in the same way that every image in the data set has: it is converted to purely black and white pixels (although ideally this would also be manually done beforehand to ensure accuracy), cropped to contain just the sign itself by locating its most distant black pixel on each side, has its transparent pixels filled in with white to make a rectangle, has its aspect ratio calculated and saved, is resized, and has its pixel values saved. All of this is done according to the above methods. Then, the aspect ratio of the input sign is compared to the aspect ratios of every sign in the data set and only the data set signs with aspect ratios close to that of the input sign are saved. Despite this sounding like a lot of calculation, this step is nearly instantaneous. The threshold for when two aspect ratios are "close" to one another is up to the user to input. However, through rigorous testing, a cutoff of 0.15 seems to be optimal. If two aspect ratios are within 0.15 of one another, they are considered "close" and the data set sign is saved. For the average input sign, this filtering step leaves about 1,000–3,000 candidate signs out of the 13,134 data set signs. Sometimes this number is a bit higher or far lower, depending on how common a sign's aspect ratio is. The candidate signs for each input image are saved in separate matrices.

It is important to note that, since the aspect ratio of the glyph is dependent on the pixels present in the image, any pixels present that are not part of the target glyph to be identified will affect the aspect ratio. These stray pixels, depending on their location, can even lead to a false aspect ratio farther than 0.15 away from that of the target glyph, effectively making true identification by the program impossible. Thus, it is vital that a user screens any images being identified or added to the data set for stray pixels and removes them. If desired, the aspect ratio filtering step can be skipped altogether, but this will drastically increase the time the program takes to run, whilst not substantially increasing the program's accuracy.

Here, it is worth discussing the "tails" present on certain signs. Some signs, such as the full variant of A1, (𓀀 "seated man"), or most versions of I9, (𓆑 "horned viper") other than the version in *nfr*, can have a long tail, a stroke dragged down longer than what is typical for most glyphs.[53] This tail often intersects other glyphs or even other lines of the text. Although the interference of these tails for other glyphs has been removed in the processing of the data set and the tails have been isolated to their own glyphs, the tails drastically affect the aspect ratio of the signs for which they are present. Some I9 signs with a short tail are nearly horizontal (e.g. with an aspect ratio of 6:1). In contrast, the ones with a tail could have a wildly different aspect ratio (e.g. 3:2). This affects the aspect ratio filtering steps and, by extension, the analysis steps. This is only an issue if, as has been hypothesized by some, the tails are truly arbitrary and do not convey significant information.[54] In the interest of investigating this issue, the tails can be cut off artificially. This is not something that can be easily done by the program, given the variety of tail sizes and tail-like structures that should not be cut off. For instance, one would not want the program to chop off the right side of a V31 (𓎡 "basket with handle") sign, simply due to it looking tail-like.[55] It is not within the scope of this project to cut off every tail present in every sign, but a test data set was created manually, by horizontally cropping all A1 signs of the full variety (the form that can have a tail) at the lowest black pixel not in the tail. An example of this can be seen in Figure 2. This method ensured the preservation of the sign's overall morphology, while also reducing the potential arbitrariness of tail length. This method is not perfect, but it is fully standardized and one of few ways for all tails to be cut without introducing major human bias. Once a data set was created with the tails removed, the data set with tails and the data set without tails were then used for the program and their results were compared. In addition to allowing the best data set to be chosen, this is able to reveal information about the importance of tails. If the tail-less data set is better, then that lends evidence to the hypothesis that tails are arbitrary or at least partly so. On the other hand, if the data set with tails is better, then it may be the case that tails actually do contain significant information and more thought should be put into them in the future (see below).

4.2.3. Filtering by Fast Fourier Transform

After the filtering by aspect ratio, the remaining signs are filtered using a Fast Fourier Transform (FFT) algorithm. FFT computes the discrete Fourier transform (DFT) of a

[53] Hoch 1998, 4, 29.
[54] Dr. Brian Muhs in discussion with the author, November 2021.
[55] Hoch 1998, 62.

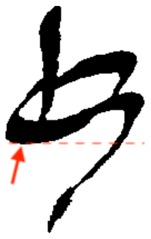


Figure 2: An example of an A1 and its tail. The red arrow points out the lowest black pixel not in the tail. The red dashed line indicates where the tail is cut. This sign comes from Eloquent Peasant B1.

sequence which can then be compared to the DFTs of other sequences.[56] The DFT is, in essence, a number of frequencies decomposed from the original sequence (a frequency domain representation of the sequence). FFT reduces the complexity of the input and produces a much more quickly comparable output, allowing researchers to work with frequencies as easily and quickly as other types of data analysis. It is this aspect of FFT that makes it a staple in the fields of digital signal and image processing, including the encoding of MP3s, analysis of gravitational waves, and spectral analysis.[57]

To make the data usable by FFT, the saved vectors of pixel values are transformed into square matrices, each matrix being of the same dimensions of the original image. The image matrices then undergo FFT using the np.fft.fft function provided by the NumPy Python package. It is not worth going into the math of this function here, but interested readers are encouraged to read Cooley and Tukey (1965).[58] This produces a complex matrix (the DFT) of both real and imaginary numbers of the same size as the input matrix. This is separated into two matrices, one real and one imaginary. The real and imaginary matrices for the input sign are compared to the equivalent two matrices for all of the saved candidate signs, a method that does not take a lot of time nor computing power. The FFT comparison algorithm is heavily inspired by the OCR work of Mark-Jan Nederhof.[59] Nederhof was kind enough to translate his FFT code into Python from Java and much of it did not need to be altered for this project. In this method, to compare two FFT outputs, a difference score is computed. This difference score is calculated as the absolute value of the input's real matrix (*IReal*) minus the data set sign's real matrix (*DReal*) at each value multiplied by a weight value plus the absolute value of the input's imaginary matrix (*IImag*) minus the data set sign's imaginary matrix (*DImag*) at each value multiplied by a weight value (see equation 1).

$$\sum_{i=1}^{20}\sum_{j=1}^{20}(|IReal_{ij} - DReal_{ij}| \cdot Weight) + (|IImag_{ij} - DImag_{ji}| \cdot Weight) \tag{1}$$

The above equation assumes an image size of 20 px by 20 px. If the images are a different size, the 20s are replaced with the image size.

[56] Cooley, Lewis, and Welch 1969.
[57] Rockmore 2000, 60.
[58] Cooley, and Tukey 1965.
[59] Nederhof 2015, 6–7.

The "Weight" value is determined by three factors: C, D, and g. Through testing and plotting of FFT spectra, it was determined that, for a resizing of 20, at points (5, 0), (10, 0), (11, 0), and (12, 0), hieratic FFT outputs are particularly variable across signs. Thus, more emphasis should be put on the differences found at those points. If the comparison is being made at one of those points, the program returns g as the weight. The variable g can be altered by the user, but, after many tests, it can comfortably be said that 1100 is the optimal value for g. If the FFT comparison is not looking at any of the previously mentioned four points, the weight value is set to equal $D + (C - i) + (C - j)$, unless i or j is greater than or equal to C, in which case the weight is set to 0. The larger i and j are, the more of the DFT is being compared; thus, the choice of the value of C determines the number of frequencies being compared. D is purely a constant to adjust the weight further. The optimal values of C and D for this program have been found to be the size of the resized images (in this case, 20) and 3, respectively. By using 20 for C, all of the frequencies of the images are compared, which is marginally slower, but also ensures all the variation is taken into account. A user can adjust the values of C and D as they see fit. To summarize the weight variable (with g, C, and D set by the user):

$$\begin{array}{ll} \text{If } (i,j) = (5, 0), (10, 0), (11, 0), \text{ or } (12, 0) & : \mathit{Weight} = g \\ \text{If } i < C \text{ and } j < C & : \mathit{Weight} = D + (C - i) + (C - j) \\ \text{If } i \geq C \text{ or } j \geq C & : \mathit{Weight} = 0 \end{array}$$

This FFT comparison algorithm results in a difference score for every candidate image from the data set compared to the input image. The lower the score is, the more similar the DFTs of the two images are. This can be seen by considering the outcome of comparing a sign against itself. All of the values of *IReal* and *DReal* would be the same and likewise for *IImag* and *DImag*. When plugged into the equation, at each position in the matrix, the result would be 0. Summed up, this would still be 0, the lowest possible score, which corresponds to the most similar two images can be (identical). One can actually produce reasonable identification results with this method alone, but it is not accurate enough to be used for any real morphological comparisons, as two signs which visually do not look very alike could have DFTs that look more similar, given that they are both just decomposed into frequencies. Therefore, this FFT algorithm is used as a second filtration step and images with FFT difference scores above a certain threshold are discarded from the candidate list for their respective input sign. This threshold can be adjusted by a user, but 9,500,000 has been found to be an optimal or near optimal value for maximizing speed while not sacrificing accuracy. This number was found through repeated tests and does not have a real world basis or tangible implication. It usually produces candidate lists in the low hundreds per sign, a very manageable number.

4.2.4. Image Deformation Model

After the data set signs have been filtered by aspect ratio and FFT, producing a slim candidate list, the candidate signs are directly compared to the input sign using an Image

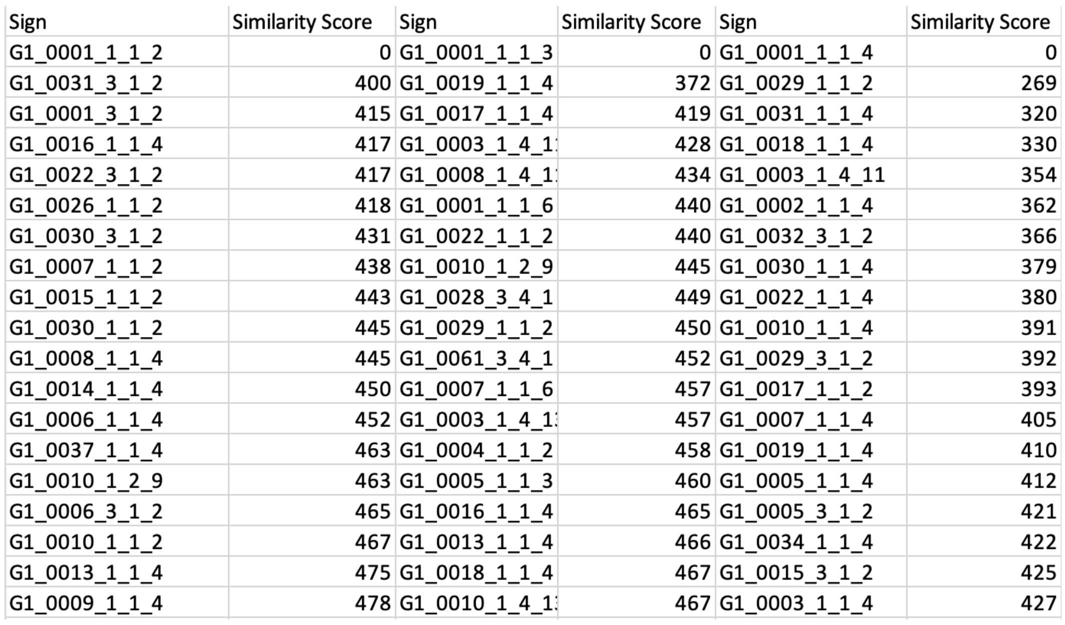
Deformation Model (IDM). Much like the FFT algorithm, this project's IDM algorithm is indebted to Nederhof's OCR work and his generosity in translating his code into Python.[60] Of course, his code had to be adapted to fit this new problem, but it, along with Keyser *et al.*'s work on IDMs, provided a significant basis for the IDM section of this project's code.[61] The IDM used for this project compares images by looking at various windows of a certain size between the two images, outputting a difference score. The difference score increases the farther the window has to move from its original spot to find a matching window.

As an example, if the window size was one, image one had a black pixel in the top left corner, and image two had a white pixel in the top left corner, then the window would shift by a certain amount in image two, still looking for a black pixel, and increase the difference score until it found one. Because one can shift the window in any direction, the lowest of these scores is taken once a match is found. The difference scores for every possible window between two images are then totaled up to find the overall difference score. This is the essence of the code, although the implementation is a bit more complex. Also of note are the cases where the window shifts to a location beyond the image's bounds. To prevent an error in these cases, all pixels outside of the bounds of the image are assumed to be white, given that the image is cropped to fit the entire sign within the bounds. Two images that are identical will never need to shift their windows to produce a match, so their difference score when compared to one another will be 0. The window size (the context) and the amount to shift the window during comparisons (the warp) are two variables that a user can adjust. Like the other variables, extensive tests were done to find the values that returned the highest accuracy. These are a context of 2 and a warp of 4. Practically, this means the window is of size 2 px by 2 px and it moves 4 px when warped.

4.2.5. Sign Identification

Once the IDM computes difference scores for each input sign compared to all of their candidate signs, the results are saved in a matrix, containing all of the candidate data set sign names and their respective scores. This matrix is then sorted so that the signs with the lowest scores appear at the top. The results for the whole set of input signs are saved in one dataframe which is exported as two .csv files; one contains the sign rankings and their difference scores and one contains just the sign rankings. These files comprise the data that has been used for this project's results, which can be analyzed, plotted, and compared, as well as the rankings that will identify a sign for a user (see section 5). An excerpt of a larger .csv results file for the hieratic sign G1 𓄿 is given in Figure 3. For more information on the format of these files, see Table 2.

[60] Nederhof 2015, 5–8.
[61] Keysers et al. 2007.



Sign	Similarity Score	Sign	Similarity Score	Sign	Similarity Score
G1_0001_1_1_2	0	G1_0001_1_1_3	0	G1_0001_1_1_4	0
G1_0031_3_1_2	400	G1_0019_1_1_4	372	G1_0029_1_1_2	269
G1_0001_3_1_2	415	G1_0017_1_1_4	419	G1_0031_1_1_4	320
G1_0016_1_1_4	417	G1_0003_1_4_1	428	G1_0018_1_1_4	330
G1_0022_3_1_2	417	G1_0008_1_4_1	434	G1_0003_1_4_11	354
G1_0026_1_1_2	418	G1_0001_1_1_6	440	G1_0002_1_1_4	362
G1_0030_3_1_2	431	G1_0022_1_1_2	440	G1_0032_3_1_2	366
G1_0007_1_1_2	438	G1_0010_1_2_9	445	G1_0030_1_1_4	379
G1_0015_1_1_2	443	G1_0028_3_4_1	449	G1_0022_1_1_4	380
G1_0030_1_1_2	445	G1_0029_1_1_2	450	G1_0010_1_1_4	391
G1_0008_1_1_4	445	G1_0061_3_4_1	452	G1_0029_3_1_2	392
G1_0014_1_1_4	450	G1_0007_1_1_6	457	G1_0017_1_1_2	393
G1_0006_1_1_4	452	G1_0003_1_4_1	457	G1_0007_1_1_4	405
G1_0037_1_1_4	463	G1_0004_1_1_2	458	G1_0019_1_1_4	410
G1_0010_1_2_9	463	G1_0005_1_1_3	460	G1_0005_1_1_4	412
G1_0006_3_1_2	465	G1_0016_1_1_4	465	G1_0005_3_1_2	421
G1_0010_1_1_2	467	G1_0013_1_1_4	466	G1_0034_1_1_4	422
G1_0013_1_1_4	475	G1_0018_1_1_4	467	G1_0015_3_1_2	425
G1_0009_1_1_4	478	G1_0010_1_4_1	467	G1_0003_1_1_4	427

Figure 3: An example of a .csv file resulting from the program. This is the first 20 rows and 6 columns of the results for the G1 signs in the data set. The first, third, and fifth columns each correspond to the results for a different sign (given in the second row of each column). The second, fourth, and sixth columns display the similarity scores for the sign to their left.

4.3. Data Analysis

The first investigation into the data that was performed was a comprehensive look at the program's accuracy. To evaluate accuracy for any given set of data, the .csv file containing just the ranked signs is loaded into Python as a dataframe. The first row of the dataframe, containing the names of the input files, is isolated and the specific sign represented by each file is saved ("A1_0021_1_2_9.png" is saved as "A1"). Then, the program can check if that sign appears in the filenames of any number of top choices in its respective column of the dataframe. After this is done for the whole input, accuracy can be calculated by dividing the number of times the sign did appear in the results by the total number of signs. In other words, each original input sign's true value is compared to the signs in a determined number of top choices given by the OCR program and the number of correct results is tallied and divided by all of the signs to determine the ratio of correct identifications:

$$\frac{\textit{signs for which at least one of the top x signs were correct}}{\textit{total signs}} = \textit{accuracy ratio}$$

The number of top signs (x) for which this is computed are one, two, and ten. Each accuracy ratio is then multiplied by 100 to produce a percentage accuracy.

The above accuracy determination method was done for data from single signs, but it was also done for random samples of the data, to gauge the accuracy of the program on data at large. These random samples were produced from the overall data set using

NumPy's random.choice function with replace = False. Another random sample, only drawing from the data set signs with greater than one example in the data set, was also taken. This was done in the same way, but after counting the examples present for each sign and filtering the signs with single examples out. Accuracy values are provided in Table 4.

Before any further analysis could be done on the OCR results, the .csv files had to be put in a distance matrix. As the name suggests, a distance matrix contains the distances between each sign in a set. To do this, first, the full .csv file, containing the sign rankings and their difference scores, is loaded into Python as a dataframe. From this, a list of all of the original input signs is saved and is used to make up the column and row names for a square distance matrix. Then, the matrix is filled in with the difference scores in the .csv file. If two signs do not appear in the output of the OCR program for each other (i.e. they were filtered out by aspect ratio or FFT), the overall difference score is taken as NA. If two signs do appear in each other's lists, the overall difference score is taken to be the sum of the two respective scores. Although this should only happen very infrequently, if one sign appears in another's list, but not vice versa, the existing difference score is simply doubled. To illustrate this method, a simple example of a possible result table for five signs is given in Table 2: in red are the input signs, in blue are the ranked data set signs (in this example, the number of ranked signs is four or fewer, although it is usually in the hundreds for the real data), in yellow are the similarity scores for the signs to their left. There are a few NA values in this table to illustrate the fact that, in the real data, not every sign has the same amount of candidates that filter through the aspect ratio and FFT steps. This produces NAs in the real data as well.

The corresponding distance matrix (Table 3) includes only the input signs and their difference scores when compared with one another. Even though, in Table 2, Sign 8 was the top choice for Sign 5, it is excluded because it is not within the input signs and is just a data set sign. In general, computing a distance matrix is only relevant or useful if one is interested in investigating the morphological similarities between a specific subset of the data set. Putting in a random sample from the data set produces a distance matrix of mostly NAs. It is also worth noting that the diagonal of zeros is the expected difference scores when signs are compared to themselves.

Sign 1 compared to itself would produce a difference score of 0 and, in the distance matrix, $0 + 0 = 0$. After the distance matrix is computed, the NA values are filled in because most algorithms that take a distance matrix as an input cannot deal with NA values. The NA values are replaced with the highest number in the matrix (in Table 3's case, it would be 1680). This replacement number could be any number greater than or equal to the highest number in the matrix and there would be no difference in results. It is purely a placeholder that communicates "these two signs are the most unalike in this set".

For this project, the main value of the distance matrix is that it can be used for UMAP (Uniform Manifold Approximation and Projection). UMAP is a non-linear dimensionality reduction technique that allows the simplification and analysis of high-dimensional data (in this case, the images in the data set), while preserving the original structure of

Table 2: An example of a possible .csv output from the OCR program

Sign 1	Score	Sign 2	Score	Sign 3	Score	Sign 4	Score	Sign 5	Score
Sign 4	300	Sign 9	220	Sign 2	720	Sign 1	340	Sign 8	360
Sign 7	347	Sign 3	700	Sign 4	810	Sign 6	380	Sign 4	408
Sign 6	450	NA	NA	NA	NA	Sign 5	420	Sign 1	499
Sign 5	507	NA	NA	NA	NA	Sign 3	870	NA	NA

Table 3: The distance matrix produced from the example data in Table 2

	Sign 1	Sign 2	Sign 3	Sign 4	Sign 5
Sign 1	0	NA	NA	640	1006
Sign 2	NA	0	1420	NA	NA
Sign 3	NA	1420	0	1680	NA
Sign 4	640	NA	1680	0	828
Sign 5	1006	NA	NA	828	0

the data.[62] In brief, UMAP constructs a low-dimensional graph representation of the data that is optimized to preserve as much of the global and local structure as possible. Although the theory behind UMAP is simple, the math is complex and interested readers are encouraged to read McInnes *et al.* (2018), the original publication of UMAP.[63] UMAP was run using the "umap" package in Python and using the umap.UMAP.fit_transform function on the distance matrix, using metric = 'precomputed'. This sets the sign distances from the matrix as the precomputed metric which UMAP attempts to preserve in its output. The UMAP output was graphed using matplotlib.pyplot. For an input (for instance, every example of A1), usually four UMAP plots were made, all identical except for the colors. One was colored by facsimile maker, one was colored by provenance, one was colored by genre, and one was colored by text. These identifications were extracted from the filename tags described in Table 1. Due to the time constraints of this work, only a few signs were able to undergo the whole analysis pipeline outlined above. Some of the most interesting/variable signs were chosen to be looked at, as well as some of the most common signs.

5. Results and Discussion

Overall, the data set that has been produced for this project, made up of individual, ligatured, and intersecting characters from the Poe, Möller, and Tabin facsimiles, cut out by Sobti and labeled with their Gardiner sign code, is extremely large. The data set is 13,134 characters, providing a fantastic starting data set for OCR, the largest of its kind in the field. The data set used in this project contains 1,104 distinct character

[62] Coenen, and Pearce 2019.
[63] McInnes, Healy, and Melville 2018.

classes, including unique ligatured and intersecting signs. 341 different hieratic signs, categorized by Gardiner's sign codes, appear in the data set, either individually or in ligatures.[64] Some hieratic signs appear many times, such as A1 (𓀀), N35 (𓈖), and G1 (𓄿), while some only appear once or not at all. The signs that appear only a few times are not able to give much morphological insight by themselves, but they should still be identifiable by the program, unless they look identical to a more common character.

5.1. Program Accuracy

On average, the IDM model, when given a random sample of 500 signs from the data set, correctly identifies unknown signs in its first choice by difference score with 71.2% accuracy, in its top two choices with 78.5% accuracy, and in its top ten choices with 84.8% accuracy. Accuracy is computed using the equation in section 4.3. In this test, each sign is excluded from the data set when it is compared, otherwise the program would get them all right as its top choice, as they would have a difference score of 0 when compared to themselves. These accuracy values seem a bit low at first glance, but one must keep in mind that, if a sign only shows up once in the data set, it will not be correctly identified when it is removed from the data set and input into the program. When excluding signs of this nature, the model is 74% accurate in its top choice, 81.8% accurate in its top two choices, and 88.2% accurate in its top ten choices, a slightly better result. The accuracy values expected when using the program on new data are certainly even higher, being brought down in these tests because of irregular writings of certain signs which do not have a second example in the data set.

For signs for which there are a decent amount of copies, the accuracy increases even further. For example, for G1 (𓄿), the model is 94.5% accurate in its top choice, 96.3% accurate in its top two choices, and 98.2% accurate in its top ten choices. A variety of accuracy values are given in Table 4. The accuracy of this program compares favorably to other such programs. Franken and van Gemert report accuracy scores of around 85% in their paper on hieroglyphic recognition[65] and Nederhof reports accuracy scores of 91.3% for his program's top choice and 95.5% for his program's top two choices in his paper on recognition of Sethe's glyphs.[66] For many signs for which there are multiple examples, the program outlined in this paper has extremely high accuracy, a significant feat given the issues hieratic poses to OCR. The program's overall accuracy being lower is not concerning, given hieratic's immense variability and numerous uncommon characters, including ligatures and overlaps. The high accuracy for common signs also indicate that the program can be used for morphological comparisons and that it has the requisite fineness to do so. The program's accuracy on all data types would only be improved with a larger data set.

[64] Gardiner 1957.
[65] Franken, and Gemert 2013, 768.
[66] Nederhof 2015, 9–10.

Table 4: Accuracy values for the IDM program

	Accuracy (%)		
Input	**in 1 choice**	**in 2 choices**	**in 10 choices**
500 random signs	72.200	80.000	86.000
500 different random signs	70.200	77.000	83.600
500 random signs with more than one example	74.000	81.800	88.200
Every A1 sign	91.376	94.128	96.147
Every A2 sign	94.079	97.368	98.684
Every D21 Sign	72.747	85.275	97.143
Every G1 sign	94.505	96.337	98.168
Every V28 sign	77.124	91.503	97.386
Every X1 Sign	69.296	83.239	97.183

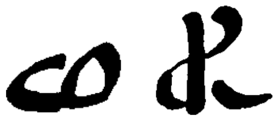


Figure 4: A typical example of an Aa1 sign (left) and an A2 sign (right) from Papyrus Prisse.

5.2. Sign Distinguishing

As expected, given the accuracy values, the OCR program is quite good at distinguishing between signs that are very morphologically dissimilar. An example of this is given in Figure 4 and Figure 5. Figure 4 displays two signs, Aa1 (𓐍 "unclassified/placenta?"), and A2 (𓀁 "man with hand to mouth").[67] These signs were chosen for demonstration purposes, due to the significant differences in form between them, as well as their relative abundances in the data set. The two specific signs in Figure 4 are both from Möller's facsimile of Papyrus Prisse and are reasonable representatives of what those signs tend to look like.

Figure 5 displays a UMAP graph of the output of the program for all Aa1 and A2 signs in the data set. For this graph and all subsequent UMAP graphs in this paper, each point represents one sign from the data set. In addition, the units of the graph's axes are largely irrelevant, given that they are a result of UMAP's graphical optimization. In UMAP graphs, it is most important to focus on which points cluster with others. Because of how UMAP uses local distances to influence the creation of the graph, the absolute distances between global clusters should not be relied upon in an interpretation.[68] Also, since UMAP has some stochasticity in the creation of its graphs, if one were to rerun the code used to produce the plots in this paper, there would be some minor, insignificant differences in the plots, mainly in the data's rotation on the plots' axes. All the conclusions in this paper are the result of running the UMAP code many times,

[67] Hoch 1998, 68, 4.
[68] Coenen, and Pearce 2019.

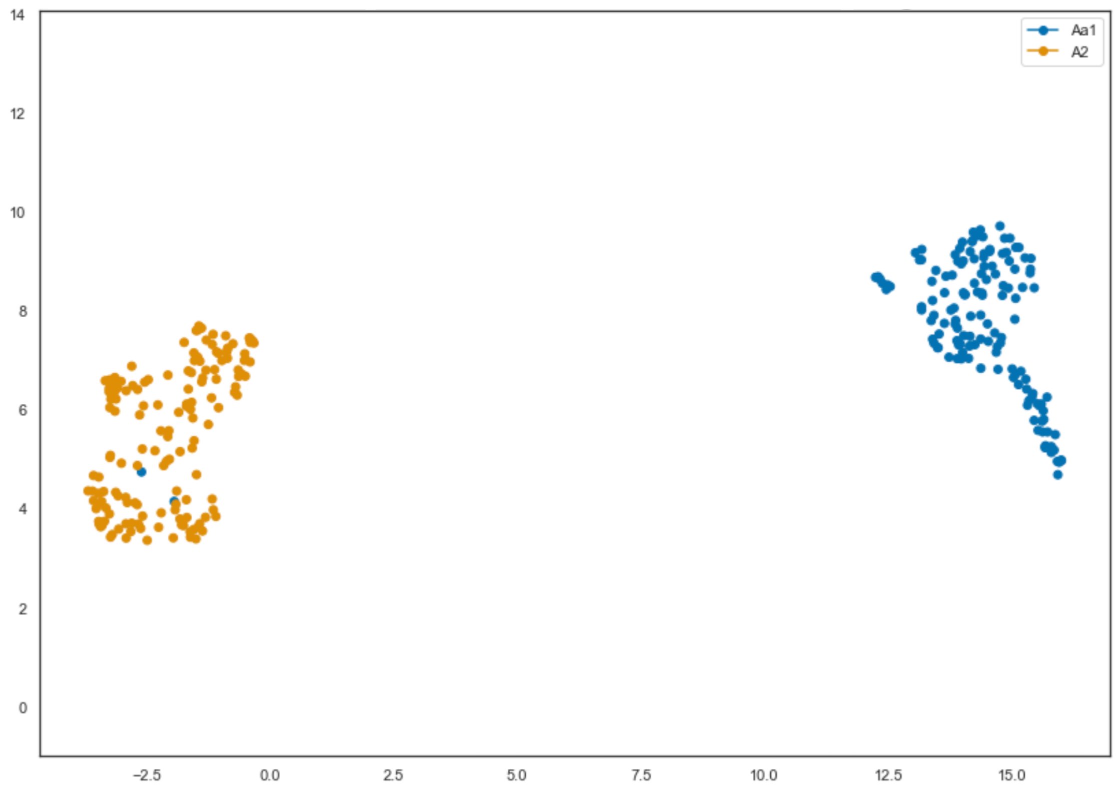


Figure 5: A UMAP plot of every Aa1 (blue) and A2 (orange) sign from the data set.

making sure any result is not just an outcome of the stochasticity, something unlikely to happen in the first place. For Figure 5, one can see that the Aa1 signs clearly cluster together separately from the A2 signs. This striking separation is good evidence that the program is accurately distinguishing the two signs.

However, one may notice that there are two Aa1 signs clustering with the A2 signs in Figure 5. One of these signs is Aa1_0009_1_4_12, shown in Figure 6. This sign and the other Aa1 sign that clusters with the A2s are both correctly identified as Aa1 by the program in its top choice. However, they are from the Rhind Mathematical Papyrus, which has a distinctive style for Aa1 signs, shown in Figure 6, that is different from the usual writing, shown in Figure 4. Although a human might decide that the Aa1 in Figure 6 looks more like the Aa1 in Figure 4 than the A2 in Figure 4, this is not at all obvious to the program. Because the Aa1 cluster and the A2 clusters are so dissimilar, the UMAP algorithm opts to put the two outlier Aa1s in the A2 cluster because it recognizes more similarities. This provides a useful and important insight into the limitations of this OCR program. The program does not look at brush strokes or theoretical features, such as curves versus lines, as a human might; these features would immediately make it clear that the Rhind Aa1s are more similar to the typical Aa1s, rather than A2s. Instead, the program takes a global morphological approach that is free of preexisting bias, for better or for worse; it simply looks at shape alone. The type of errors in clustering that

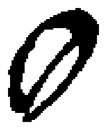


Figure 6: An example of an atypical Aa1 sign from the Rhind Papyrus.

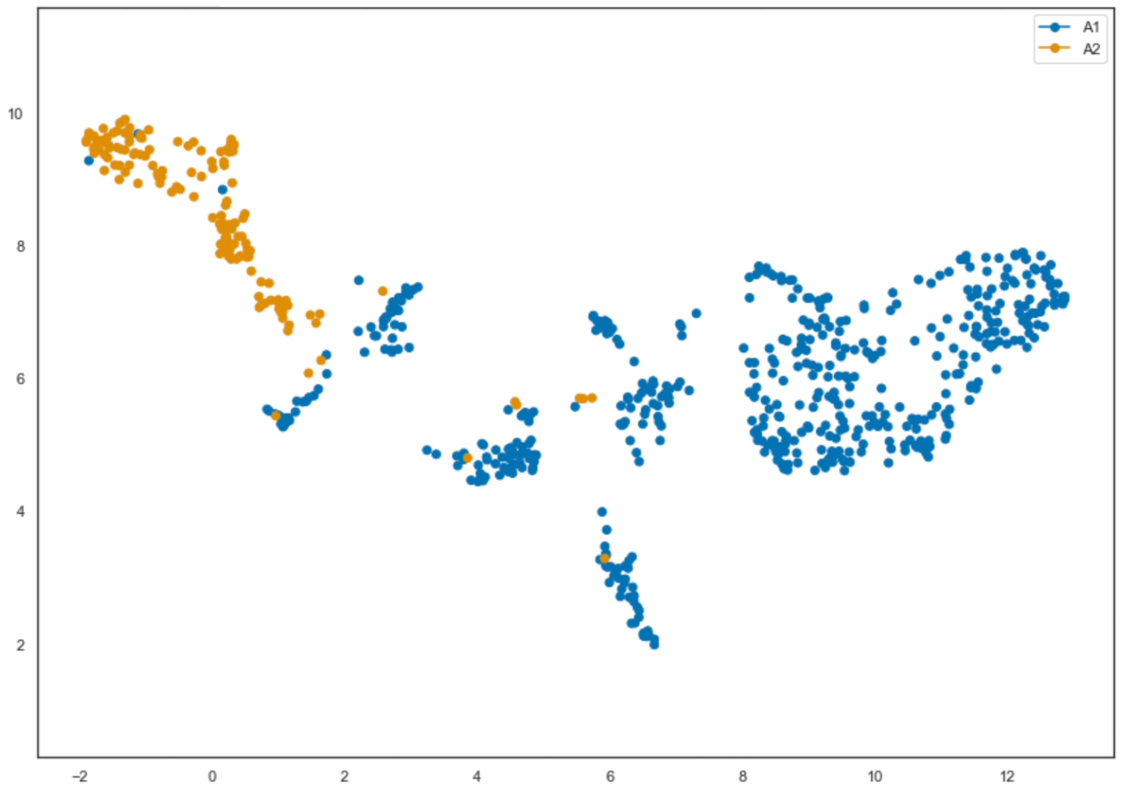


Figure 7: A UMAP plot of every A1 (blue) and A2 (orange) sign from the data set.

are present in Figure 5 should not happen often, unless very distinct signs with a few outliers each are compared. Nevertheless, anyone using this program should be aware that the program may provide slight errors in these cases.

In Figure 7, a UMAP plot is provided for A1 and A2. Examples of A1 are given later in this section in Figure 12. Here, one can see that there are still fairly distinct clusters, but there is more overlap and the clusters are closer together than in Figure 5. This makes intuitive sense, given that the two signs are much more similar in general form to one another than Aa1 and A2. This plot has been provided to demonstrate that the closer two signs are morphologically, the less distinct the UMAP clusters will be.

This UMAP cluster comparison can be used to investigate questions relating to signs that can look nearly identical to the human eye. For example, D21 (𓂋 “mouth”), and X1 (𓏏 “loaf of bread”), are often indistinguishable from one another.[69] For a scholar

[69] Hoch 1998, 11, 65.

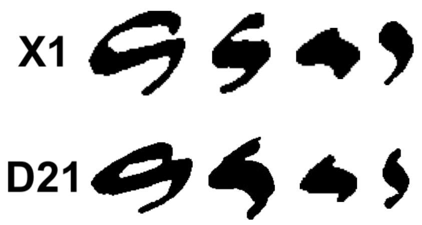


Figure 8: Four examples of X1 (𓏏) signs (top) and D21 (𓂋) signs (bottom). The examples represent the typical variation in these signs. The X1 signs come from, in order of left to right, Papyrus Ebers, Eloquent Peasant B1, Hymn to Senwosret III, and Eloquent Peasant B1. The D21 signs come from, in order of left to right, Papyrus Prisse, Eloquent Peasant B1, Hymn to Senwosret III, and Sinuhe B.

reading a hieratic text, these signs usually have to be determined through context clues or known spellings of particular words, rather than direct morphology. This similarity can be seen in the accuracy of the OCR program on D21 and X1 above in Table 4. Although the accuracy is over 97% for each sign in ten choices, there is only around 70% accuracy in the top choice alone. This is due to the signs being so similar that the program initially sometimes misidentifies an X1 as a D21 and vice versa (of course, sometimes the program misidentifies these signs as other similar looking signs too). A variety of different writings of X1 and D21 signs from the data set are shown in Figure 8. The examples were chosen to demonstrate the signs' similarities to one another, but they also represent the variations in the two signs fairly well.

From the physical appearance of X1 and D21, it seems as if the two signs are written in effectively the same way. However, this is measurably false; Figure 9 shows a UMAP plot for X1 and D21. Here, although there is overlap between the two signs and there are not very distinct clusters, it is clear that D21 and X1 do tend to segregate apart from one another overall. Indeed, even the accuracy values support this. Since X1 and D21 are being correctly identified in the program's top choice 70% of the time, there must be significant differences. If there were not and the signs were effectively identical, the program would be expected to correctly identify the signs in the top choice only 50% of the time. Given that the program demonstrates that there are differences between X1 and D21 in general, future work can be done into exactly what elements of the signs distinguish them from one another. This is a limitation of the OCR program: it can inform which signs are similar/different, but not what specific features make them so. The data on which X1s cluster with D21s and which do not can be extracted from the UMAP output and another program could be written to investigate the specific features that separate the two signs. This is beyond the scope of this work, but would be a fascinating follow up experiment.

A similar analysis to the above was done on O49 (𓊖 "a town's crossroads"), and N5 (𓇳 "sun").[70] In hieroglyphs, these two are distinct. However, for hieratic, the signs are often indistinguishable and are only determined by context, much like X1 and D21. O49 generally appears after city-related words and place names, whereas N5 generally

[70] Hoch 1998, 44, 36.

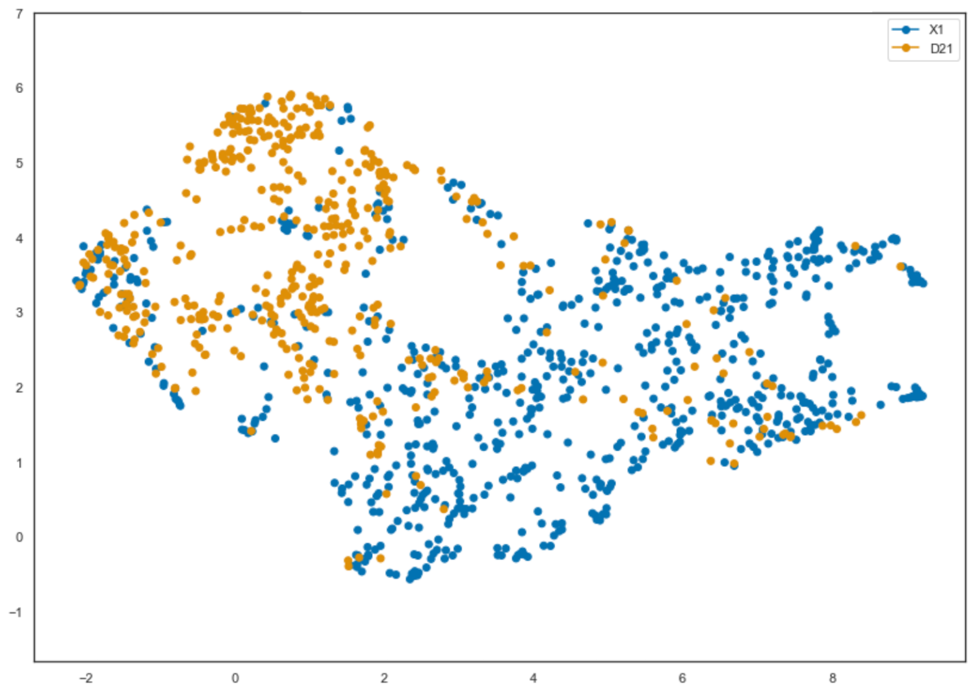


Figure 9: A UMAP plot of every X1 (𓏏) (blue) and D21 (𓂋) (orange) sign from the data set.

appears after sun-related words and date/time words. Figure 10 displays a UMAP of O49 and N5 and, unlike the previous comparison, there is no clear separation. This is supported by the accuracy values: for O49, the program is 25% accurate in one choice, 35% accurate in two choices, and 65% accurate in ten choices; for N5, the program is 31% accurate in one choice, 45% accurate in two choices, and 76% accurate in ten choices. These are very atypical and low accuracy values and, while they would likely increase with greater sample size, it demonstrates that O49 and N5 both look so much like other signs that they often cannot be distinguished from them.

Certain writings of O49 (𓊖) and N5 (𓇳) are identified by the program not only with one another, but also with Aa1 (𓐍), D21 (𓂋), D32 (𓂓), N33 (𓈒), W24 (𓏌), X1 (𓏏), Z1 (𓏤), Z4 (𓏭), and more. In this light, the high similarity between the writings of O49 (𓊖) and N5 (𓇳) is probably not a reflection that those signs were similar in some deeper way to ancient Egyptians. Instead, it is a good demonstration of a common theme in hieratic: small determinatives losing detail. Modern readers are surely not the only ones who used context to determine signs; the Egyptians were likely doing that easily when reading and writing hieratic. Thus, they could write different determinatives exactly the same without worry. D32, N33, W24, Z1, and Z4 are all small determinatives that follow words, so it is no wonder that there is overlap between the cursive forms of these small signs that have similar uses. In addition, Aa1, D21, and X1 are all common signs with simple writings, so it is unsurprising that the simplified determinatives sometimes look like them.

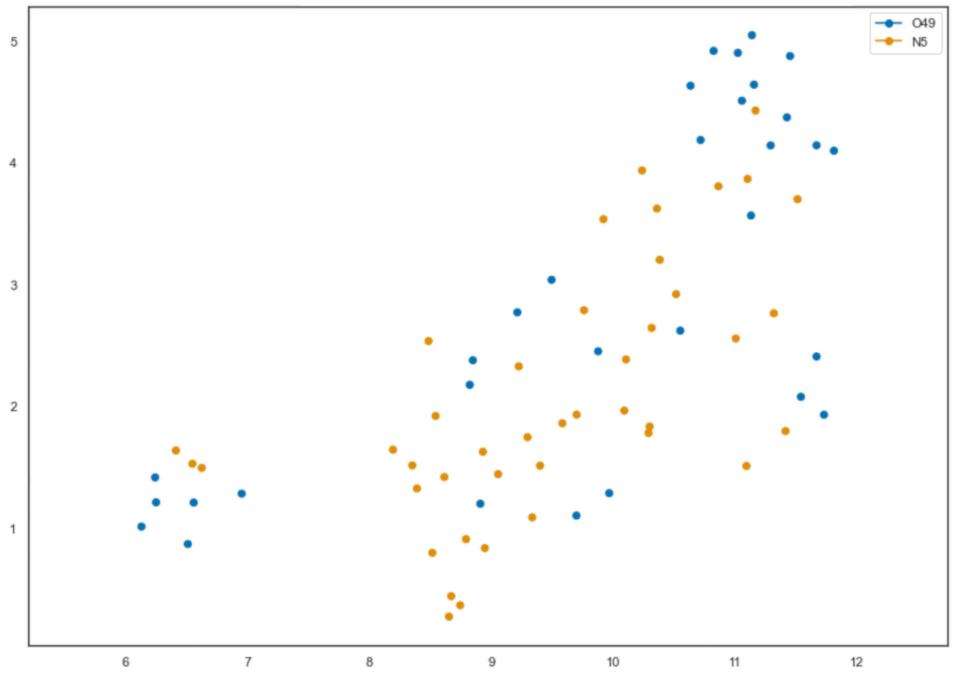


Figure 10: A UMAP plot of every O49 (⊗) (blue) and N5 (⊙) (orange) sign from the data set.

This analysis provides two pieces of insight:

1. If, in hieratic, many determinatives are collapsed into a common form so completely, it is hard to make the case that determinatives continue to carry much meaning at this stage of the language (Middle Egyptian), at least the most common ones. By the Middle Kingdom, the determinatives seem to be an established convention for how to write each word, but, unlike "determinative" suggests, do not actually assist in helping a reader determine the category of a word. If a circular writing after a word can mean O49, N5, D32, N33, W24, Z1, Z4, or more, then it cannot be that helpful in distinguishing the word. Although the determinatives are much clearer in hieroglyphs, if they were actually significantly helpful in determining a word, they would also be clear in hieratic.

2. As mentioned above, the OCR program can only distinguish between different shaped signs. Identical signs will lower the accuracy substantially. If two signs look identical, the program's accuracy will be 50% in one choice. If three signs look identical, the accuracy will be 33% in one choice and so forth. The small determinatives are a great example of an area where the program is limited in identification ranking. On the other hand, large comparisons through UMAP are not limited in this way, as the difference score is the important part, not the exact ranking.

It should be noted that, while UMAP comparisons can be extremely useful, they, like many analyses, suffer when sample size is low. Another comparison was attempted

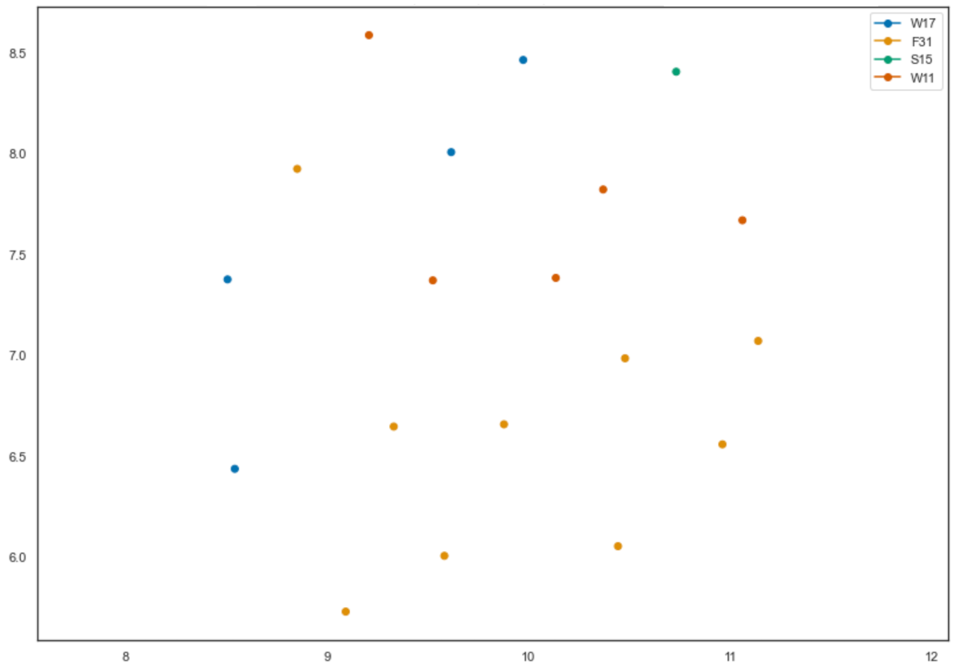


Figure 11: A UMAP plot of every W17 (blue), F31 (orange), S15 (green), and W11 (red) sign from the data set.

between F31 (𓄟 "three foxes' skins tied together"), S15 ("a piece of jewelry?"), W11 (𓎼 "ring-stand for jars"), and W17 ("water amphorae in a rack").[71] Unfortunately, the numbers of each sign in the data set were far too low to accurately distinguish anything. A UMAP is provided in Figure 11, but the structure of the plot is cloud-like and the points are almost equally spaced. This could indicate that there is little difference in the form of the four signs or it could indicate an underlying structure that the plot cannot demonstrate well. Although the signs do look somewhat similar, nothing conclusive can be said without more information. This underscores the importance of having a large data set. With less common signs, comparisons can be difficult or even impossible. This can be ameliorated if more signs are added to the data set over time.

5.3. Tail Separation Investigation

Within single signs, there can still be a large amount of variation. A1 (𓀀 "seated man") is a good example of this. There are multiple writings of A1, but they generally fall into two main groups: hereafter called "full form" and "abbreviated form". A typical example of each is shown in Figure 12. The two types of A1 are quite distinctive and some texts use both versions to represent A1. Within this project's data set, the texts that include full form A1s are Shipwrecked Sailor, Eloquent Peasant B1, Sinuhe B, and *Texte aus Hatnub*. The data from Papyrus Prisse also includes one full form A1.

[71] Hoch 1998, 21, 50, 64.

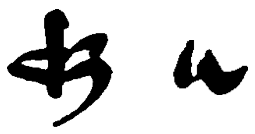


Figure 12: Typical examples of A1 𓀀 (full and abbreviated) from P. Prisse and Eloquent Peasant B1.

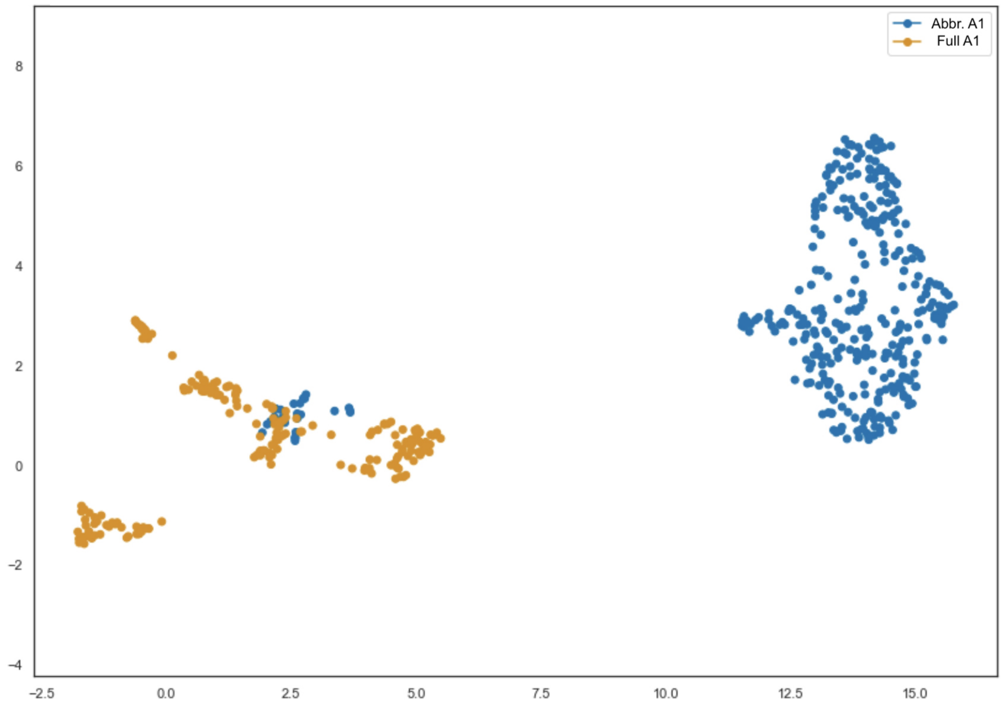


Figure 13: A UMAP plot of every abbreviated form A1 (blue) and full form A1 (orange) sign from the data set.

Figure 13 is a UMAP plot of the two variants and, as one might expect, they largely cluster separately. There are a few examples of abbreviated A1s clustering with the full form A1s, but this could be explained by two factors. First, as mentioned above for Figure 5, unique writings of signs could be misidentified by the program if they are different from both main groups. This could make sense in this example, as it is more likely that an abbreviated A1 would be drawn with some extra projection that would make it look like a full form than a full form somehow being written in a condensed way to look like an abbreviated form. Second, what is a "full form" A1 and what is a "abbreviated form" A1 was determined manually, introducing human error. Some signs theoretically could have been misidentified at the first step. This is another utility of the program outlined in this paper; it can be used as a check to make sure identifications are correct. In any case, Figure 13 is unambiguous in demonstrating that the two variants of A1 are largely distinct.

However, there is more to the variation in A1 than purely abbreviated form versus full form. The full form A1 signs have a great amount of variance with respect to tail length.

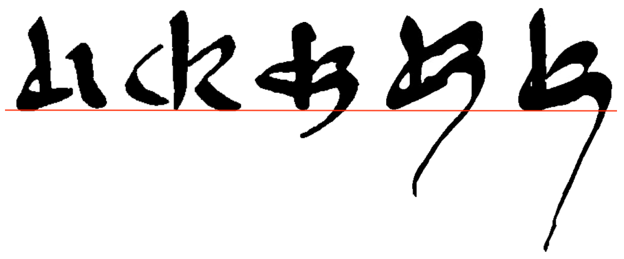


Figure 14: Four examples of full form A1 signs, showing the great tail variation. The red line indicates where the tail would be cut according to the "lowest black pixel not in the tail" method. The signs come from, in order from left to right, Sinuhe B, *Texte aus Hatnub*, Papyrus Ebers, Eloquent Peasant B1, and Eloquent Peasant B1.

Tails, described above in the "Methods" section, are long strokes dragged down beyond the normal extent of a sign. These strokes, hypothesized to be arbitrary, significantly affect the program's comparisons.[72] An example of the tail variation in full form A1 signs is provided in Figure 14. The red line indicates the start of the tail as decided by the method described above.

Figure 15 shows a comparison between the UMAP plot of the full form A1s on their own (a.) and those same full form A1s, but with all of the tails cut according to the above methods (b.). The points are colored here by their original text. With the tails intact, the signs from different texts cluster together with one another in certain places. When the tails are removed, suddenly the clustering happens strongly by text. This clustering by text is not always the case for signs, so it indicates that the A1 shape is very distinctive between different handwritings. In addition, since the removal of the tails prompted the clustering to be by text, rather than dispersed with little rhyme nor reason, this is good evidence that tails are fairly arbitrary. No one author is lengthening the tails in a distinctive way.

This is further demonstrated by the Hatnub data; the Hatnub signs should always cluster on their own, apart from the other texts, given that the *Texte aus Hatnub* were written on rough stone walls (as opposed to on papyrus) and, thus, have significantly different morphology. The Hatnub data is an ideal outgroup and it offers a way to evaluate the quality of the results. In the graph with the tails, one of the Shipwrecked glyphs clusters with the Hatnub material, a very strange occurrence and likely due to the similarity in tails driving the comparison, rather than the rest of the character shape. When the tails are removed, the Hatnub glyphs cluster together without any other texts, as expected. This is further evidence that the tails are not good elements to include when trying to use signs for comparisons, as they can overwhelm better indicators of the underlying sign morphology. Overall, this investigation offers striking evidence of the arbitrariness of sign tails, at least for A1 signs. It seems that the tails are only added when convenient and little meaning is encapsulated by their length or inclusion at all. Far more significant is the body of each sign as an indicator of handwriting or text. In the future, similar work should be done cutting the tails of other signs to see if these results are more widely applicable.

[72] Dr. Brian Muhs in discussion with the author, November 2021.

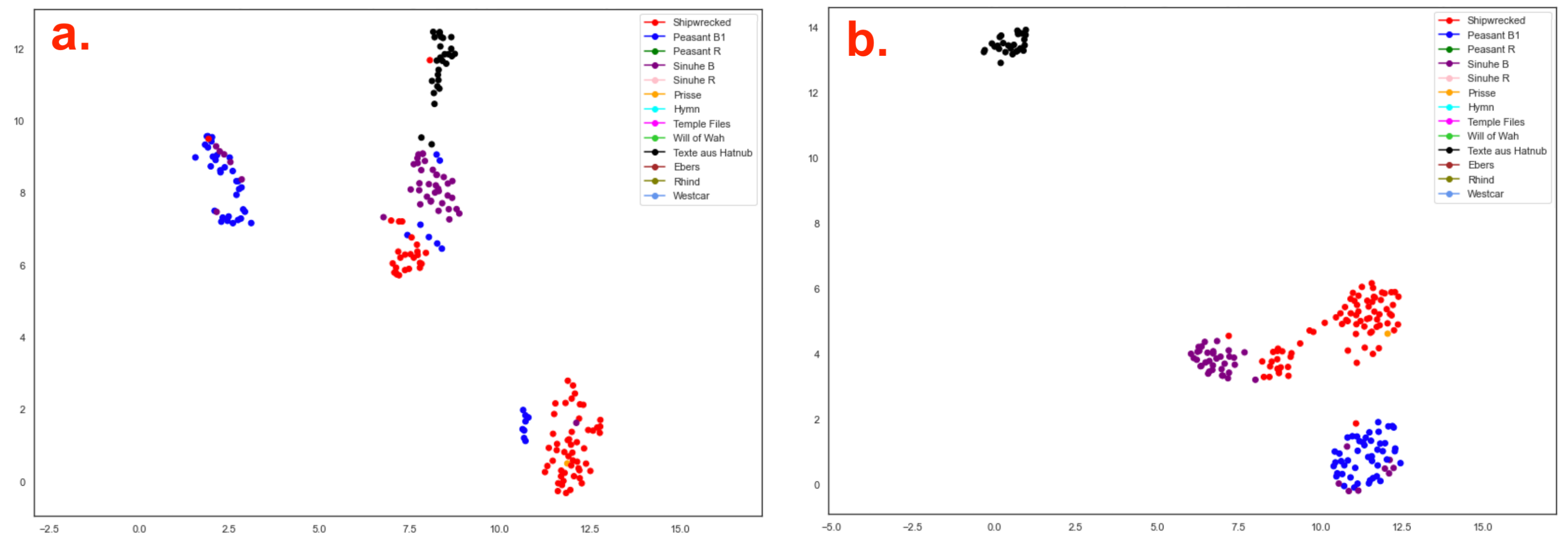


Figure 15: Two UMAP plots of every full form A1 sign from the data set with (a.) and without (b.) tails, colored by text. Of the texts that appear in this plot, Shipwrecked Sailor signs are in red, Eloquent Peasant B1 signs are in blue, Sinuhe B signs are in purple, Papyrus Prisse signs are in orange, and *Texte aus Hatnub* signs are in black.

The data clustering by text when the tails are removed leads to another interesting observation: Sinuhe B and Eloquent Peasant B1 largely cluster separately. This is interesting because there is evidence that those two texts were written by the same scribe.[73] Supporting that hypothesis, most of the time Sinuhe B and Peasant B1 glyphs cluster together.[74] In this light, it is quite interesting to see that, for A1, the two texts cluster apart from one another. Some A1 signs from Sinuhe B cluster with the Peasant B1 signs, but there is still a significant difference between the two that must be dealt with. These differences cannot be due to handwriting (because they were written by the same scribe), location (because they were found in the same place), or time period (because they were written at roughly the same time). There are a few possibilities, however.

One possibility is that the difference could be due to a difference in genre/person; both are literary texts, but the Eloquent Peasant is a mostly third-person narrative with poetic elements and Sinuhe is a first-person narrative written in a journal style. Thus, there could be an as-of-yet unknown difference in writing styles between these two texts. This is especially compelling because, in Middle Egyptian, A1 can represent the first-person singular suffix pronoun (𓀀) and it is also included in the first-person singular independent pronoun (𓏌𓀀). In these uses, A1 would be particularly important for first-person narratives, like Sinuhe.[75] Sinuhe also clusters closest to Shipwrecked Sailor, another first-person narrative which would use A1s in a very similar way. In fact, the Shipwrecked Sailor cluster has two distinct groups within it, one closer to Sinuhe B and one farther away. This could represent the difference in first-person Shipwrecked A1s as opposed to other usages of A1 in the text. In this case, the first-person Sinuhe A1s would be clustering with the Shipwrecked first-person A1s and the non-first-person Sinuhe A1s would be the ones clustering with the Eloquent Peasant A1s. In essence, this would mean that there is a difference across different texts in the convention for how the first-person A1s are written. Unfortunately, it is beyond the scope of this project to analyze the use of each of these signs. However, the program has been set up to make such comparisons trivial, provided someone collects the data. In the future, one could record the usage of each A1 (first-person pronoun or not) and add a tag to the end of all of them in the same way the text or facsimile maker is recorded in the current data set. Then, it would be a simple click to add those colors to the UMAP and investigate this question.

Another possibility for the Sinuhe-Peasant A1 difference seen in this data could come down to writing implements. Although the scribe who wrote Sinuhe B and Peasant B1 likely used a similar brush for both, it is not at all out of the question that there were slight differences in the scribal utensil between the writings of the texts. This could have led to most signs being largely the same, but certain brush strokes appearing different. It could be the case that the full form A1 sign includes some of the different brush strokes.

[73] Parkinson, and Baylis 2012, 12.
[74] Tabin 2022, 181–251.
[75] It is worth mentioning that, for the suffix pronouns and the independent pronoun, both the full and abbreviated form A1s can be used.

This would be a tough hypothesis to test, but a finer look at the ink on the papyri could shine some light on it. However, this idea is perhaps only worth looking into if the previous hypothesis about A1 morphology being different for first person uses is found to be unsupported.

5.4. Facsimile Maker Investigation

Because the data set includes signs from multiple facsimiles of texts (Möller, Poe, and Tabin), questions about the quality and accuracy of facsimiles can be investigated. If a facsimile maker has a very distinctive style that overwhelms the underlying variation between the texts, one would expect their signs to be clustered separately from the rest of the data in a UMAP plot. On the other hand, if the three facsimile makers had the exact same handwriting and style, then one would expect the signs to not cluster based on maker at all. Beyond that, one would expect two data set signs that come from different facsimiles of the same text (i.e. are facsimiles of the same sign) to have a difference score of 0 and overlap completely in a UMAP plot. This can be looked at because, as mentioned above, the Poe and Tabin facsimiles overlap to a certain degree for the Shipwrecked Sailor text and the Möller and Tabin facsimiles overlap to a certain degree for the Peasant B1 text.

Figure 16 is a UMAP plot of the output of the program for all D21 (𓂋) signs in the data set. In this figure, the plot is colored by facsimile maker. D21 was chosen as an example, but all other signs tested produced similar results with regards to facsimile questions. In this plot, two interesting observations are immediately apparent. First, none of the facsimile makers' signs cluster separately from all the others, a good sign for the accuracy of the three facsimiles. In addition, there are some places in the plot where two differently colored points are extremely close to one another. Sometimes, this is due to very similar writings, but many of these places are where Möller and Tabin or Poe and Tabin made a facsimile of the same sign and they are correctly clustering close together. However, no two points are ever completely overlapping. This is a reflection of their difference scores not being 0 and, thus, the facsimile maker adding extra variation. Overall, the results indicate that, while there is some effect of the handwriting of facsimile makers, the driving factor of the clustering is the underlying hieratic morphological variation from the original texts.

In the interest of answering a likely question, it should be noted that there are isolated patches of blue and green points because Möller and Tabin respectively made facsimiles of signs/texts that no other facsimile maker in this study did. However, that does not explain the isolated patch of orange points, corresponding to signs from Poe's facsimile. Poe's facsimile does not cluster separately, but it also does not line up completely with the Tabin facsimile. Because the Tabin Shipwrecked Sailor facsimile covers all of the signs that Poe's does, ideally each of Poe's signs should be right next to one of Tabin's, if there was little effect of the facsimile creator. This is true for many of them, but not all, and it is the case for every plot of results from the data set, no matter the sign. Poe's facsimile was created more as a teaching tool than for morphological accuracy, but maintains the general shape of all of the signs. This leads to another interesting finding:

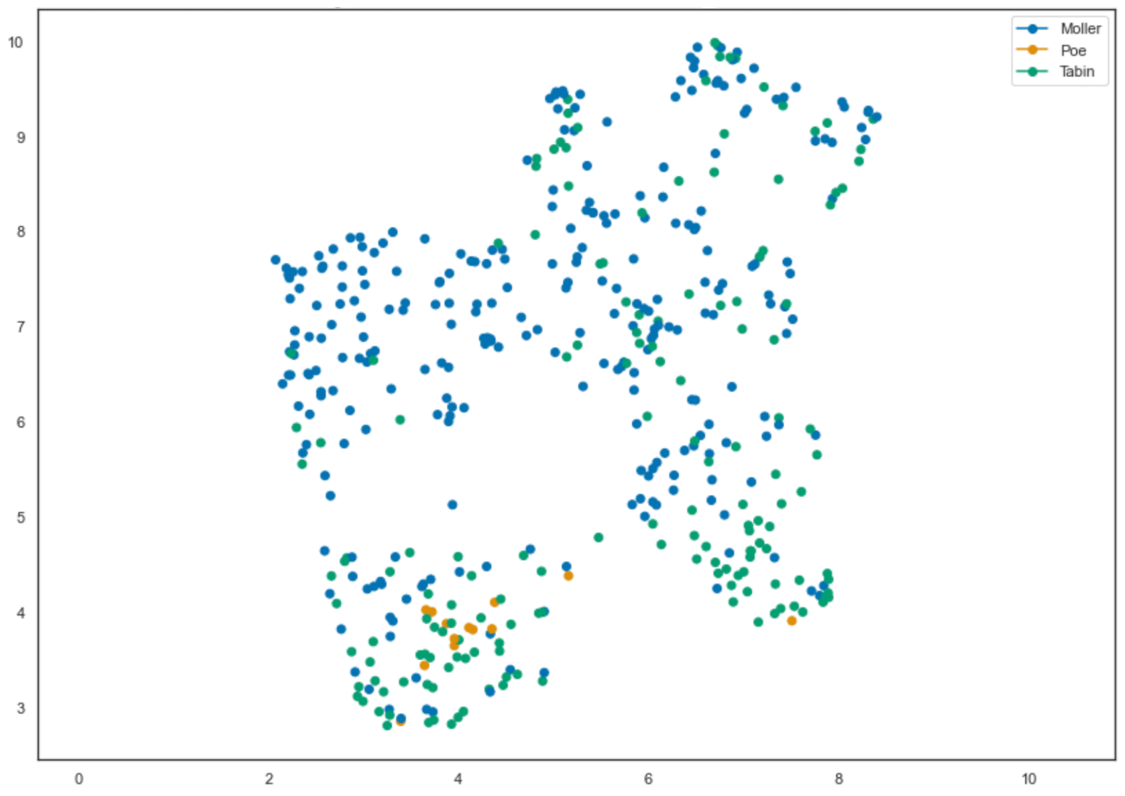


Figure 16: A UMAP plot of every D21 sign from the data set, colored by facsimile maker. Möller signs are in blue, Poe signs are in orange, and Tabin signs are in green.

most of the clustering can be explained by the general shape of the original signs. The little bit of clustering that the Poe signs do together is due to the more minor variations between facsimiles. In other words, Poe's loss of morphological accuracy in the fine details does have an effect, but not a substantial one.

This finding is not necessarily unexpected; it does not take much to predict that variations in general shape, which are larger physical changes than those in minor details, will have a greater effect on similarity than the variations in minor details. Nevertheless, this has implications for the program and for facsimile creation. It has been shown that the program is fine-tuned enough to recognize small morphological changes between signs. If it was not, the Poe signs would not have grouped together. Therefore, the limits of the technology and the data set have not been reached, so even more complex facsimiles could be analyzed in the future. It is also clear that large scale comparisons can still be done on facsimiles that maintain the general structure of the original signs, but do not try to be 100% accurate. Of course, the more accurate a facsimile is, the better, but facsimiles with lower accuracy can still be useful, provided they adhere to overall morphological shape.

6. Conclusions and Future Directions

This paper has provided an overview and proof-of-concept for applying a new Optical Character Recognition method to the field of hieratic paleography. A substantial data set of 13,134 individual signs from a range of texts, genres, and facsimile makers was created and an Image Deformation Model was used to analyze the signs. The program proved adept at quickly and accurately identifying individual signs, as well as making large scale comparisons. The sign identification element of the program will surely be a great benefit to scholars of all stages. Those who are learning hieratic will be able to use the program to check their own transcriptions and assist with recognizing difficult signs, as opposed to the old way of having to flip through Möller's paleography in the hopes of stumbling upon the desired sign. Even those who are well versed in hieratic will be able to benefit from the program's ability to recognize and separately cluster sign variants beyond the level of human perception.

As for the comparisons made possible by the program, there are infinite possibilities, only a few of which were able to be dealt with in this work. First, through a comparison of Aa1 (𓐍) and A2 (𓀁), the program was shown to cluster signs purely by shape, not by strokes as humans might. Then, a comparison of X1 (𓏏) and D21 (𓂋) showed that, although the two signs often look identical to humans, there is underlying morphology separating them. On the other hand, O49 (𓊖) and N5 (𓇳) were shown to often look effectively the same, underscoring the common trend of small determinatives being condensed and written the same in hieratic. F31 (𓄟), S15 (𓋣), W11 (𓎼), and W17 (𓏃) also were compared, but the data set did not have enough examples of each for the comparison to be very strong.

After this, the A1 (𓀀) data set was shown to segregate into full and abbreviated form A1s. The full form A1 data set was plotted with and without their tails. The plot without the tails was far superior and intuitive, demonstrating that the lengths of tails of signs are mostly arbitrary and do not convey significant information about handwriting. However, when the tails were removed, the Sinuhe B and Eloquent Peasant B1 texts clustered apart from one another, a curious finding given that they are known to be written in the same hand. It was hypothesized that the usage of the first person pronoun or variations in writing implement could account for this. Through a look at the way signs do not seem to strongly cluster by facsimile maker, the program was also shown to truly be viewing the underlying sign morphology, for the most part. The texts that had facsimiles done by multiple facsimile makers had their signs cluster together as expected and no signs segregated by facsimile more than a bit. This reflects that, even if one is primarily concerned with general shape and not fine detail, as Poe was for his facsimile, such a facsimile can still provide useful information. In addition, it was extrapolated that the program's limits have not been reached and even more complex facsimiles could be analyzed in the future.

With these demonstrations, I hope I have shown the incredible ability and power of this technology, making wide-scale and rapid comparisons possible. As for the utility of the program, the same methods used to compare facsimile makers or different sign forms in this paper can be easily adapted to different provenances, time periods, or textual genres.

Indeed, work of this nature is presented in my thesis, comparing the sign morphology of the Shipwrecked Sailor and Papyrus Prisse, as well as better characterizing the Lahun papyri and Papyrus Westcar.[76]

Overall, the various analyses performed in this paper should offer a glimpse into the potential of the OCR program, both for learning and for research. In the future, this program can be improved to better support such uses. For the program itself, the three main sections of the pipeline (Sobti, the labeling program to add the "tags", and the OCR program) will surely be combined into one streamlined program. This will be much more user-friendly and increase the accessibility of the tools. Sobti and the labeling program could also use OCR to improve themselves, as mentioned above. This would minimize the need for manual input, other than to correct for errors the program makes in identification, further decreasing the time it takes for signs to be added to the data set.

Even before the above occurs, the data set, the various sections of the program, and the analysis code have all been made open-source. The code and data set images have been put onto a GitHub repository.[77] For additional long-term stability, the data set images and precalculated metrics have been put into a citable Dataverse.[78] The data set is able to be fully released because Möller's glyphs are no longer subject to copyright, Poe has authorized the release of his glyphs, and I will do the same for my own facsimiles. Because the data set and program are free to use, anyone will be able to use the program to identify hieratic characters, allowing scholars to more easily read and learn hieratic. Because of the open-source nature of the program, it will be able to be adapted and expanded by anyone who is willing to contribute to the data set. As demonstrated time and time again in this paper, more data will lead to more comparative power, more findings, and more significance for research, not to mention vastly improving the already high accuracy of the program. In addition, data from more provenances, texts, genres, and facsimile creators will unlock numerous new possibilities for research. All of the insights offered in the results section would benefit immensely from further research to test the hypotheses outlined here and to more completely understand the patterns observed.

Other data could be collected and added to the data set beyond what was explored in this paper. One could add information on vertically versus horizontally written signs, because there are known differences in some sign shapes between the two formats.[79] It would be intriguing to see if the changes between vertical and horizontal are common across all texts or if some texts change different aspects of their writings. The methods used in this paper could also be expanded to data from different time periods, to track the development of hieratic morphology over time. The texts in this paper were almost all from around the Middle Kingdom period, to avoid adding an extra confounding variable. However, with much more data added, the time period of texts could go from being a

[76] Tabin 2022, 56–66.
[77] https://github.com/jtabin/PaPYrus.
[78] Tabin 2023.
[79] Bomhard 1999, 52.

confounder to being another axis from which one can make comparisons. One could also add sign distribution and usage information, as suggested in the above discussion for the A1s from Sinuhe B versus Peasant B1. In addition, one could add texts from ostraca as well as papyri, given that some texts exist only on ostraca and the material is very different, possibly influencing the way signs are written (e.g. see Hagen[80]). A morphological comparison between the hieratic from ostraca and papyri on this scale would be very interesting.

For the program itself, a significant future direction for its development could be an alteration enabling the program to output information about what features of characters make them similar to one another. The program could even highlight the exact pixels it notes as being different, creating a differential pixel heatmap. This method was beyond the scope of this paper, but would not be too hard to do in the future and the technology is already available.

Much further in the future, the program will hopefully be able to be used on direct images, rather than facsimiles. This would entail a much larger data set and far better papyrus images than what are often available. As explained at the beginning of this work, direct images are much harder to work with for hieratic given the variable amounts of papyrus damage, ink density, and other factors. However, a program specially created for this purpose, pre-trained on a myriad of facsimile images, could potentially work for that analysis. If this hypothetical future program incorporates machine learning techniques, which it certainly should as those techniques become exponentially more prominent and better, it could even accurately fill in damaged sections of papyri.

This paper has provided a point to build upon for future work in the fascinating and burgeoning field of ancient Egyptian Optical Character Recognition. The tools provided here could be instrumental for allowing scholars to tap into modern cutting-edge technological methods and apply them to new areas of research and learning. Over the course of this work, the point has hopefully been made that the current, human-driven paleographic methods can greatly benefit from the large-scale power of OCR technologies and that computer-driven methods can eliminate some of the bias that naturally enters paleographic work. These computer methods can distinguish morphologies of far more characters at once and to a far deeper level than any human can. It should be reiterated that the computer methods still require an Egyptologically trained hand to implement them, lest one makes incorrect interpretations of the data. As more people add to the data set and use the program, its usefulness will increase drastically. Artificial intelligence programs, like OCR, can no longer be ignored, especially as a new generation of motivated, technologically fluent scholars enter the field, ready to apply the most up to date methods. As we live through a time with an unprecedented amount of data at our fingertips, the ability to synthesize, categorize, and analyze that data in a controlled and advantageous way will become more and more important. For hieratic paleography, this paper has provided a step forward.

[80] Hagen 2012, 84–101.

References

Aguizy, Ola El-. 1986. “A Palaeographical Study of Demotic Papyri in the Cairo Museum from the Reign of King Taharka to the End of the Ptolemaic Period.” *Enchoria: Zeitschrift für Demostistik und Koptologie* 14: 67–70.

AKU-PAL. 2022. *AKU-PAL. Paläographie des Hieratischen und der Kursivhieroglyphen*. Mainz. Available at https://aku-pal.uni-mainz.de.

Berg, Hans van den, and Koen Donker van Heel. 2000. “A Scribe’s Cache from the Valley of the Queens? The Palaeography of Documents from Deir El-Medina: Some Remarks.” In *Deir El-Medina in the Third Millennium AD. A Tribute to Jac. J. Janssen*, edited by Robert J. Demarée, 9–49. Egyptologische Uitgaven 14. Leiden: Nederlands Instituut voor het Nabije Oosten.

Bermeitinger, Bernhard, Svenja A. Gülden, and Tobias Konrad. 2021. “How to Compute a Shape: Optical Character Recognition for Hieratic.” In *Handbook of Digital Egyptology: Texts*, edited by Carlos Gracia Zamacona and Jónatan Ortiz García, 121–38. Monografías de Oriente Antiguo 1. Alcalá de Henares: Editorial Universidad de Alcalá. https://doi.org/10.25358/openscience-6757.

Blackman, Aylward M. *The story of King Kheops and the magicians: transcribed from Papyrus Westcar (Berlin Papyrus 3033)*. JV Books, 1988.

Bomhard, Anne-Sophie von. “Le conte du naufragé et le papyrus Prisse.” *Revue d’Égyptologie* 50: 51–65.

Casey, Christian. 2008. *The Story of the Shipwrecked Sailor: Transcription, Transliteration, and English Translation with Full Commentary*. University of Texas at Austin.

Chace, Arnold B. 1927. *The Rhind Mathematical Papyrus*. Oberlin, OH: Mathematical Association of America.

Coenen, Andy, and Adam Pearce. 2019. “Understanding umap.” *Google PAIR* (2019).

Collier, Mark, and Stephen Quirke. 2004. *The UCL Lahun Papyri: Religious, Literary, Legal, Mathematical and Medical*. BAR International Series 1209. Oxford: Archaeopress.

Cooley, James W., Peter A. W. Lewis, and Peter D. Welch. 1969. “The fast Fourier transform and its applications.” *IEEE Transactions on Education* 12, no. 1: 27–34.

Cooley, James W., and John W. Tukey. 1965. “An algorithm for the machine calculation of complex Fourier series.” *Mathematics of computation* 19, no. 90: 297–301.

DigiPal. 2011–2014. *DigiPal. Digital Resource and Database of Manuscripts, Palaeography and Diplomatic*. London. Available at http://www.digipal.eu.

DPDP. n.d. *The Demotic Palaeographical Database Project*. Heidelberg. Available at http://129.206.5.162/beta/index.html.

Elnabawy, Reham, Rimon Elias, and Mohammed A.-M. Salem. 2018. “Image Based Hieroglyphic Character Recognition.” In *2018 14th International Conference on Signal-Image Technology & Internet-Based Systems (SITIS)*: 32–39.

Erman, Adolf. 1890. *Die Märchen Des Papyrus Westcar: I: Einleitung und Kommentar. II: Glossar, Palaeographische Bemerkungen und Feststellung des Textes*. Berlin: Spemann.

Franken, Morris, and Jan C. van Gemert. 2013. “Automatic Egyptian hieroglyph recognition by retrieving images as texts.” In *Proceedings of the 21st ACM international conference on Multimedia*: 765–768.

Gardiner, Alan H. 1957. *Egyptian Grammar: Being an Introduction to the Study of Hieroglyphs*. 3rd ed., Oxford: Griffith Institute.

Golénischeff, Wladimir. 1913. *Les papyrus hiératiques No 1115, 1116A et 1116B de l'Ermitage Impérial à St-Pétersbourg*. St. Petersburg: Dir. de l'Ermitage Imp.

Gülden, Svenja A., Celia Krause, and Ursula Verhoeven. 2020. "Digital Palaeography of Hieratic", In *The Oxford Handbook of Egyptian Epigraphy and Paleography*, ed. by Vanessa Davies, and Dimitri Laboury, 634–646. Oxford: Oxford University Press.

Haliassos, Alexandros, Panagiotis Barmpoutis, Tania Stathaki, Stephen Quirke, and Anthony Constantinides. 2020. "Classification and Detection of Symbols in Ancient Papyri." In *Visual Computing for Cultural Heritage*, edited by Fotis Liarokapis, Athanasios Voulodimos, Nikolaos Doulamis, and Anastasios Doulamis, 121–40. Springer Series on Cultural Computing. Cham: Springer International Publishing. https://doi.org/10.1007/978-3-030-37191-3_7.

Hagen, Fredrik. 2012. *An ancient Egyptian literary text in context: the instruction of Ptahhotep*. Orientalia Lovaniensia Analecta 218. Leuven: Peeters.

HebrewPal. 2022. *HebrewPal. Hebrew Palaeography Album*. Oxford. Available at https://www.hebrewpalaeography.com.

Hoch, James E. 1998. *Middle Egyptian Grammar Sign List*. Mississauga: Benben Publications.

Jéquier, Gustave. 1911. *Le papyrus Prisse et ses variantes : papyrus de la Bibliothèque Nationale (Nos 183 à 194), Papyrus 10371 et 10435 du British Museum, Tablette Carnavon au Musée du Caire*. Paris: Paul Geuthner.

Keysers, Daniel, Thomas Deselaers, Christian Gollan, and Hermann Ney. "Deformation models for image recognition." *IEEE Transactions on Pattern Analysis and Machine Intelligence* 29, no. 8: 1422–1435.

Koch, Gregory, Richard Zemel, and Ruslan Salakhutdinov. "Siamese neural networks for one-shot image recognition." In *ICML deep learning workshop*, vol. 2.

Koch, Roland. 1990. *Die Erzählung des Sinuhe*. Bibliotheca Aegyptiaca 17. Fondation Égyptologique Reine Élisabeth: Brussels.

McInnes, Leland, John Healy, and James Melville. 2018. "Umap: Uniform manifold approximation and projection for dimension reduction." *arXiv preprint* arXiv:1802.03426.

Memon, Jamshed, Maira Sami, Rizwan Ahmed Khan, and Mueen Uddin. 2020. "Handwritten optical character recognition (OCR): A comprehensive systematic literature review (SLR)." *IEEE Access* 8: 142642–142668.

Möller, Georg. 1909a. *Hieratische Lesestücke für den akademischen Gebrauch*. Vol 1. Leipzig: Hinrichs.

———. 1909b. *Hieratische Paläographie: Die Aegyptische Buchschrift in ihrer Entwicklung von der Fünften Dynastie bis zur Römischen Kaiserzeit: I. Band: Bis zum Beginn der Achtzehnten Dynastie*. Leipzig: Hinrichs.

Moezel, Kyra van der. *Administrative Hieratic from Dynasties 19 and 20. Case Studies on Selected Groups of Ostraca with Necropolis Administration*. Hieratic Studies Online 4. Mainz: Akademie der Wissenschaften und der Literatur | Mainz, 2022. https://doi.org/10.25358/openscience-7839.

Nagai, Masakatsu, Toshihito Waki, Yona Takahashi, and Satoru Nakamura. 2021. *Hieratische Paläographie DB*. Tsukuba University. January 31, 2021. https://moeller.jinsha.tsukuba.ac.jp.

Naz, Saeeda, Khizar Hayat, Muhammad Imran Razzak, Muhammad Waqas Anwar, Sajjad A. Madani, and Samee U. Khan. 2014. "The optical character recognition of Urdu-like cursive scripts." *Pattern Recognition* 47, no. 3: 1229–1248.

Nederhof, Mark-Jan. 2015. "OCR of handwritten transcriptions of Ancient Egyptian hieroglyphic text." *Altertumswissenschaften in a Digital Age: Egyptology, Papyrology and beyond*, Leipzig.

———. n.d. *St Andrews Corpus*. https://mjn.host.cs.st-andrews.ac.uk/egyptian/texts/corpus/pdf.

Parkinson, Richard B. 1991. *The Tale of the Eloquent Peasant*. Oxford: Griffith Institute, Ashmolean Museum.

Parkinson, Richard B., and Lisa Baylis. 2012. *Four 12th Dynasty Literary Papyri (Pap. Berlin P. 3022–5): A Photographic Record*. Berlin: Akademie Verlag.

Poe, William Clay. 2008. *The Writing of a Skillful Scribe: An Introduction to Hieratic Middle Egyptian Through the Text of the Shipwrecked Sailor*. Sonoma State University.

Quirke, Stephen G. J. 2011. "Agendas for Digital Palaeography in an Archaeological Context: Egypt 1800 BC." In *Kodikologie und Paläographie im digitalen Zeitalter* 2, edited by Franz Fischer, Christiane Fritze, and Georg Vogeler, 279–94. Schriften des Instituts für Dokumentologie und Editorik 3. Norderstedt: Books on Demand. http://nbn-resolving.de/urn:nbn:de:hbz:38-43548.

Rockmore, Daniel N. 2000. "The FFT: an algorithm the whole family can use." *Computing in Science & Engineering* 2, no. 1: 60–64.

Rosmorduc, Serge. 2014. *JSesh Documentation*. June 12, 2014. http://jseshdoc.qenherkhopeshef.org.

Sethe, Kurt Heinrich. 1927. *Urkunden der 18. Dynastie*. Leipzig: Hinrichs.

Stableford, Tom. n.d. *Translation of Georg Möller's works on Hieratic*. http://www.egyptologyforum.org/bbs/Stableford/StablefordMoeller.html.

Tabin, Julius. 2022. "From Papyrus to Pixels: Optical Character Recognition Applied to Ancient Egyptian Hieratic" (thesis). Retrieved from https://knowledge.uchicago.edu/record/3695. Distributed by the University of Chicago.

———. 2023."Data for Optical Character Recognition Applied to Hieratic: Sign Identification and Broad Analysis". https://doi.org/10.7910/DVN/D8CWVZ, Harvard Dataverse, V1.

TLA. 2014. *Thesaurus Linguae Aegyptiae*. BBAW, Ancient Egyptian Dictionary Project. November 10, 2021. https://aaew.bbaw.de/tla/servlet/TlaLogin.

Wreszinski, Walter, ed. 1913. *Der Papyrus Ebers; Umschrift, Übersetzung und Kommentar*. Vol. 1. Leipzig: Hinrichs.

Žába, Zbyněk. 1956. *Les maximes de Ptaḥḥotep*. Prague: Éditions de l'Académie Tchécoslovaque des Sciences.

A. Tabin Facsimiles

Here, the facsimiles created specifically for this project are presented. Figures A.1 to A.16 are the facsimile of P. Hermitage 1115 (The Shipwrecked Sailor), lines 1–189. Figures A.17 to A.32 are the same facsimile, but with the damaged sections marked in red. Figures A.33 to A.37 are the facsimile of P. Berlin 3023 (Eloquent Peasant B1), lines 32–121. In this last facsimile, red ink is indicated by hollow signs. There is no damage-marked version of P. Berlin 3023 due to the time constraints on this work.

One should notice that there may be small errors with these facsimiles in places. Although they were each created twice and checked many times over, there are still places with minor errors, such as some pixels not being filled in. However, this did not impact this work, because, during the annotation step, the facsimiles were checked for imperfections. The individual signs were also checked multiple times after each one was cut out of the facsimiles. Thus, the final signs used for the program are even more morphologically accurate and pristine than those present in these images. Despite the rigorous quality control, there are surely some signs that have retained errors, but all of the significant ones are certainly eliminated.

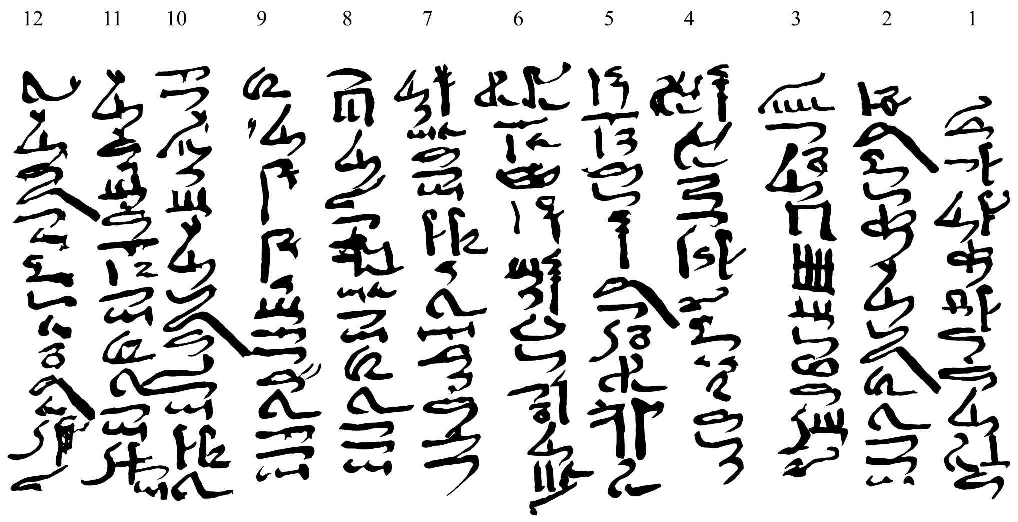


Figure A.1: P. Hermitage 1115, 1–12 (clean)

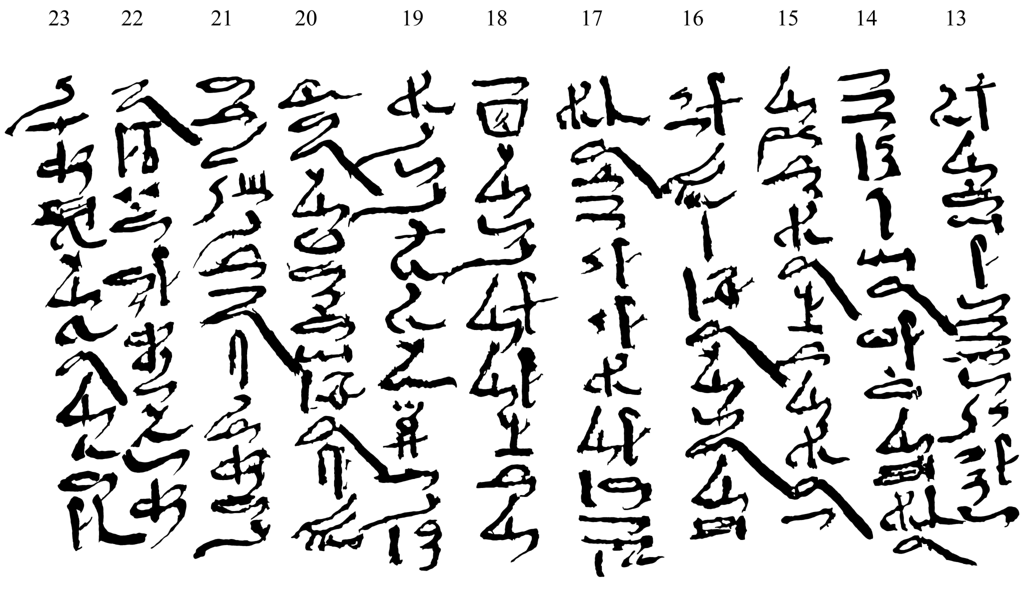


Figure A.2: P. Hermitage 1115, 13–23 (clean)

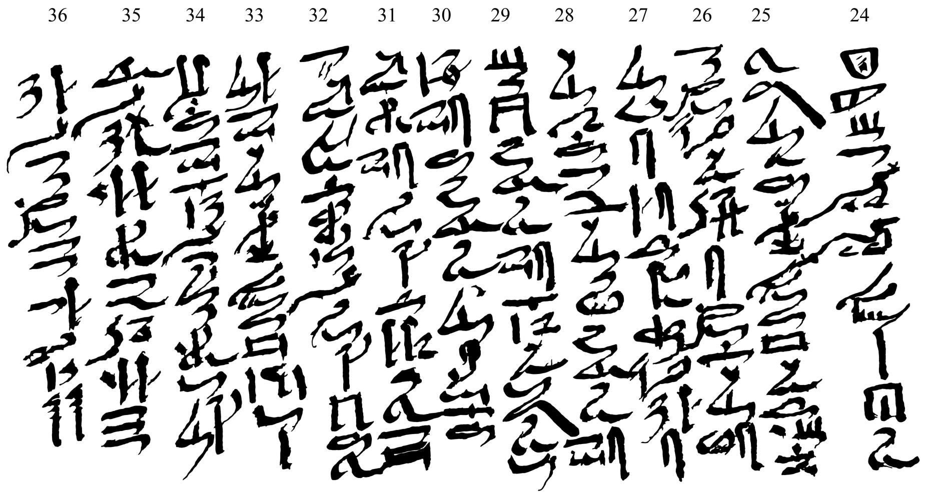


Figure A.3: P. Hermitage 1115, 24–36 (clean)

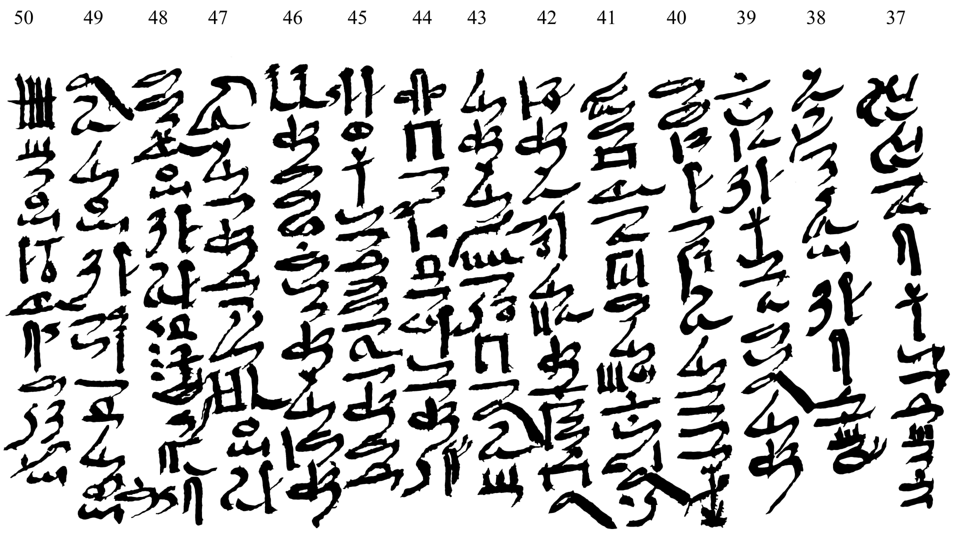

50
49
48
47
46
45
44
43
42
41
40
39
38
37

Figure A.4: P. Hermitage 1115, 37–50 (clean)

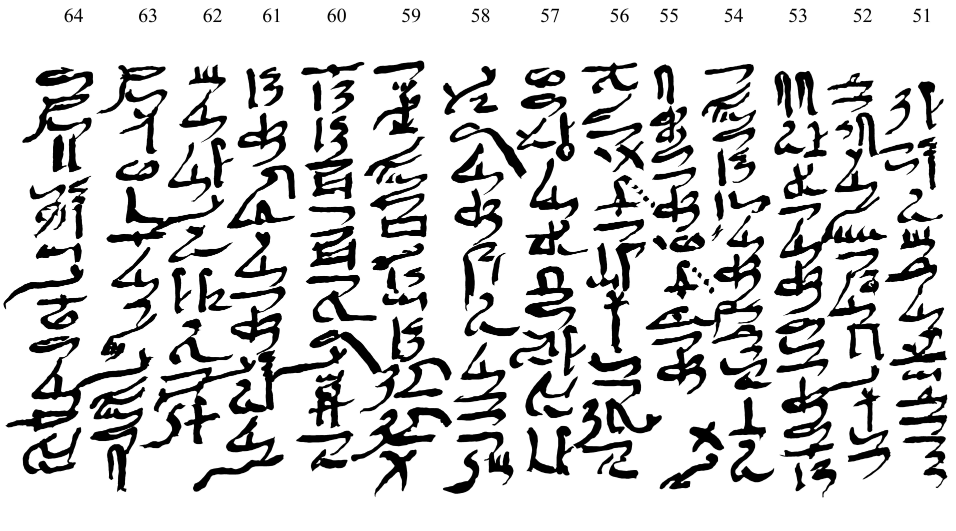

64
63
62
61
60
59
58
57
56
55
54
53
52
51

Figure A.5: P. Hermitage 1115, 51–64 (clean)

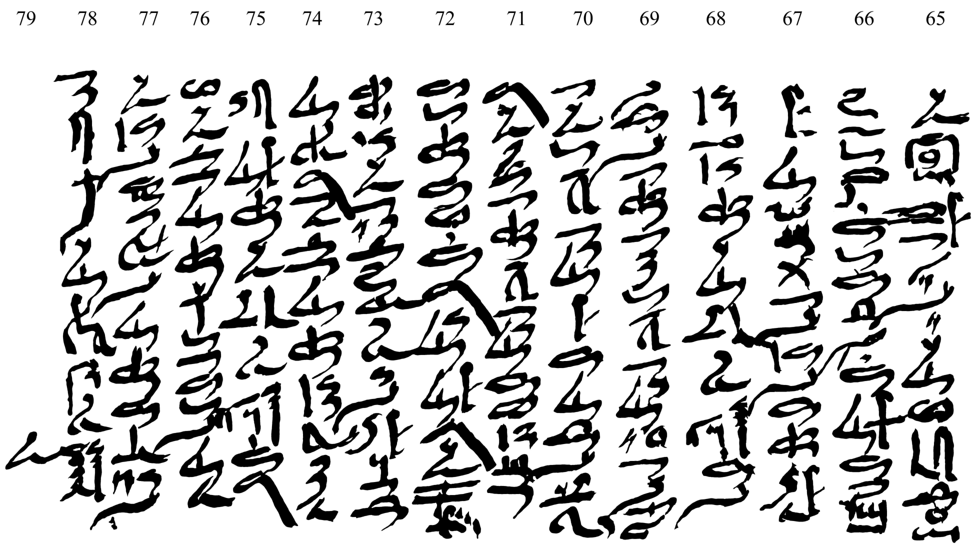

79
78
77
76
75
74
73
72
71
70
69
68
67
66
65

Figure A.6: P. Hermitage 1115, 65–78 (clean)

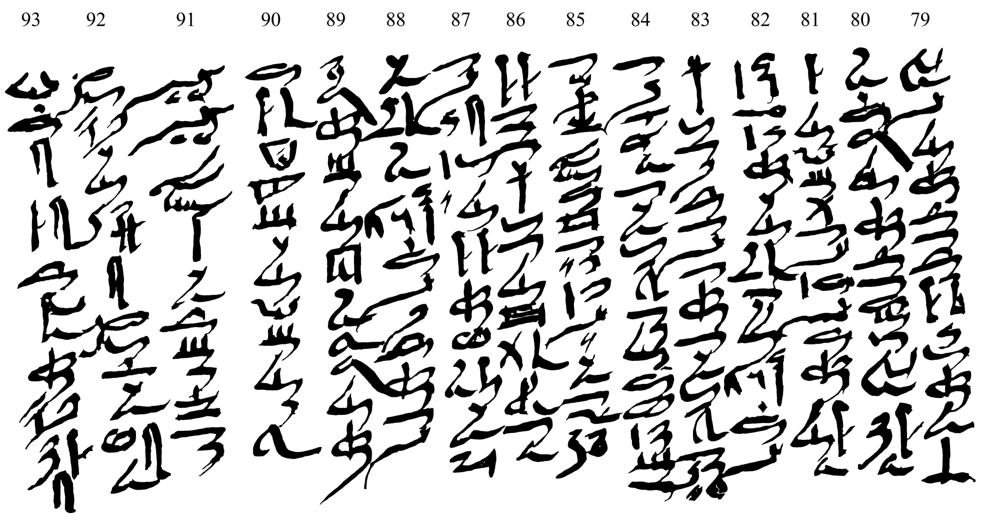

93
92
91
90
89
88
87
86
85
84
83
82
81
80
79

Figure A.7: P. Hermitage 1115, 79–93 (clean)

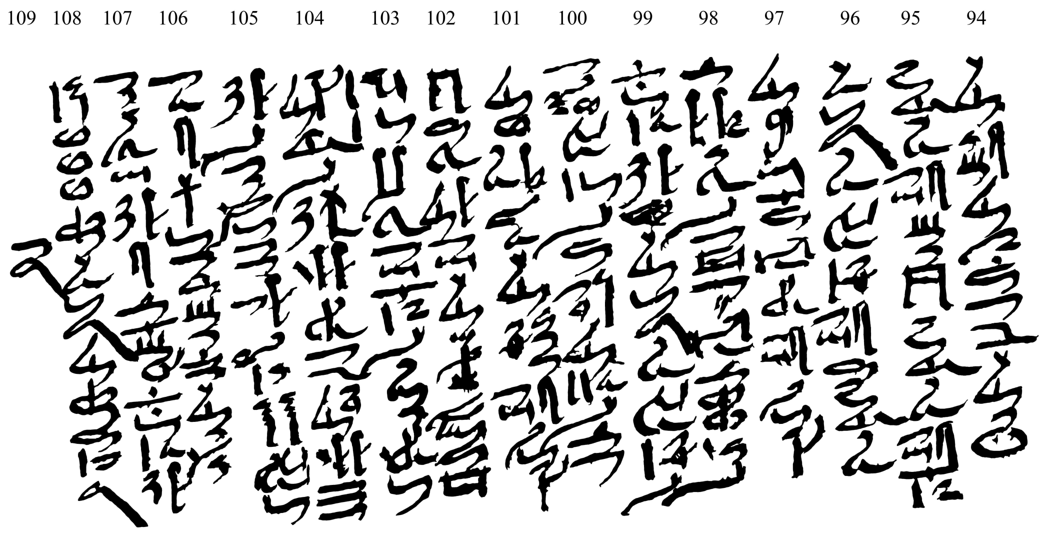


Figure A.8: P. Hermitage 1115, 94–108 (clean)

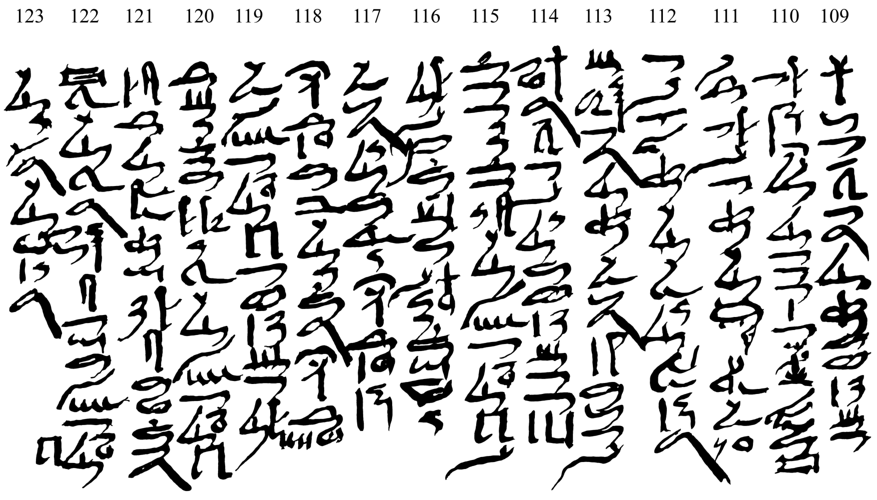


Figure A.9: P. Hermitage 1115, 109–123 (clean)

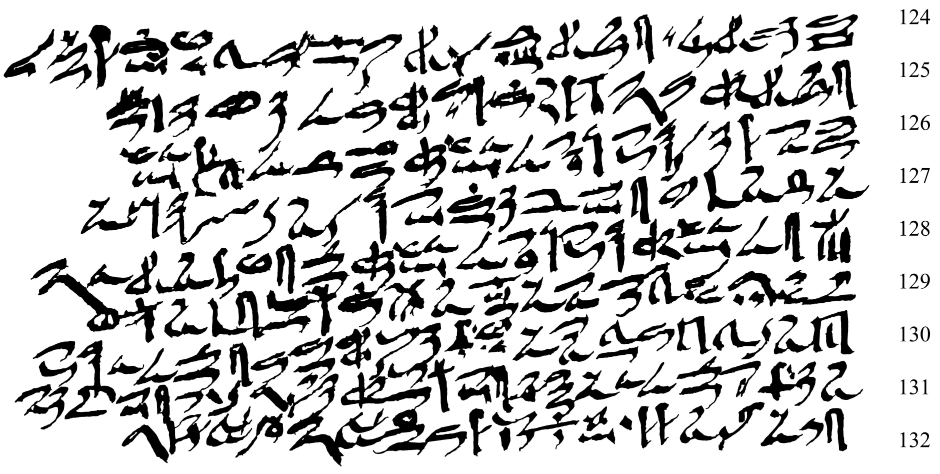


Figure A.10: P. Hermitage 1115, 124–132 (clean)

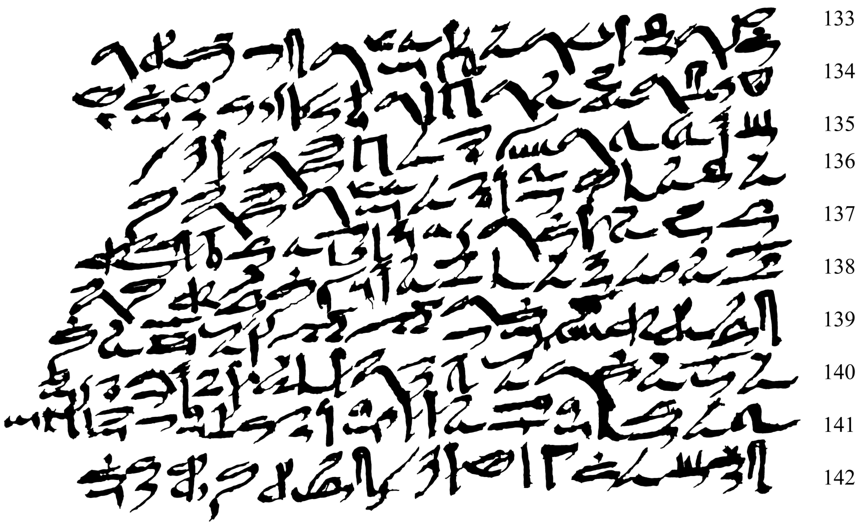


Figure A.11: P. Hermitage 1115, 133–142 (clean)

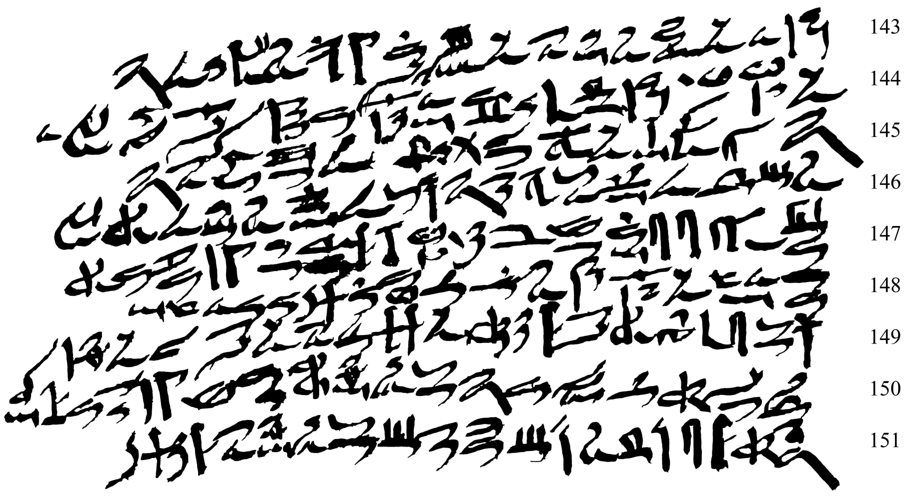

143
144
145
146
147
148
149
150
151

Figure A.12: P. Hermitage 1115, 143–151 (clean)

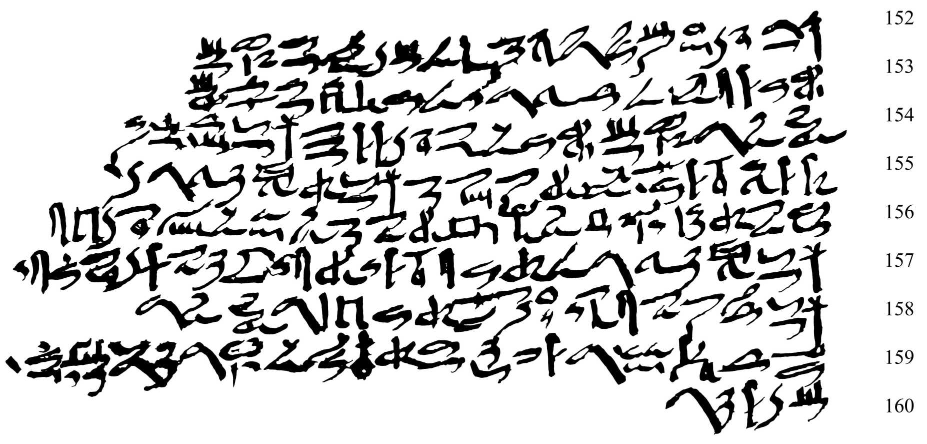

152
153
154
155
156
157
158
159
160

Figure A.13: P. Hermitage 1115, 152–160 (clean)

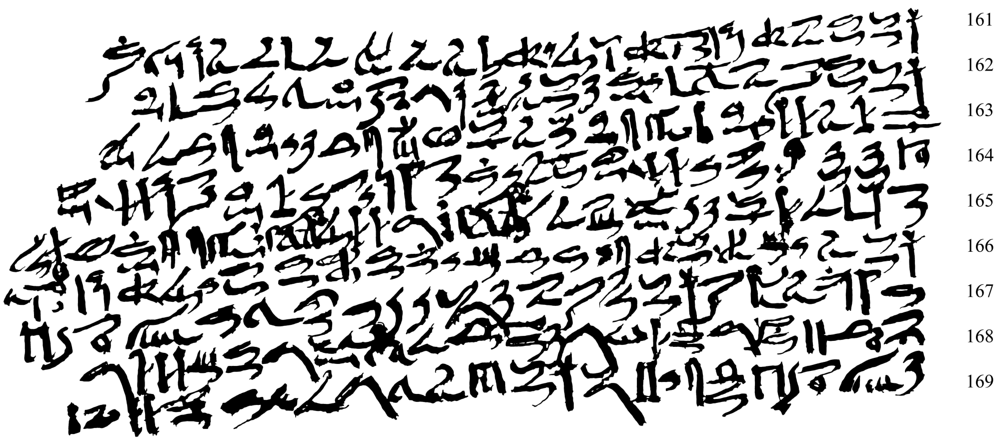

161
162
163
164
165
166
167
168
169

Figure A.14: P. Hermitage 1115, 161–169 (clean)

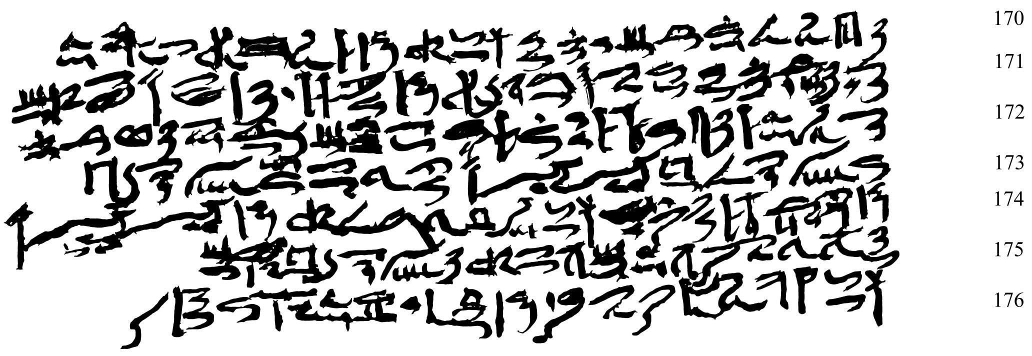

170
171
172
173
174
175
176

Figure A.15: P. Hermitage 1115, 170–176 (clean)

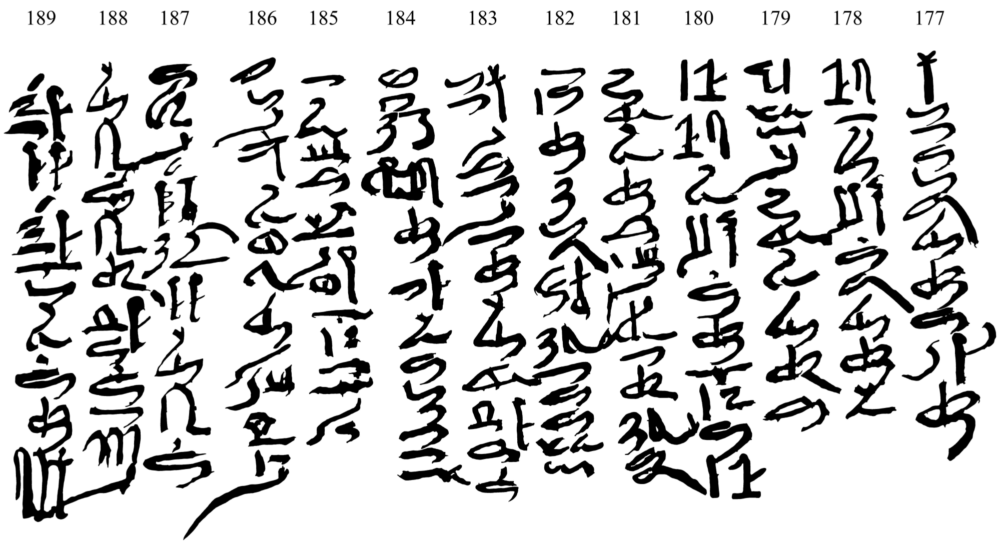

189
188
187
186
185
184
183
182
181
180
179
178
177

Figure A.16: P. Hermitage 1115, 177–189 (clean)

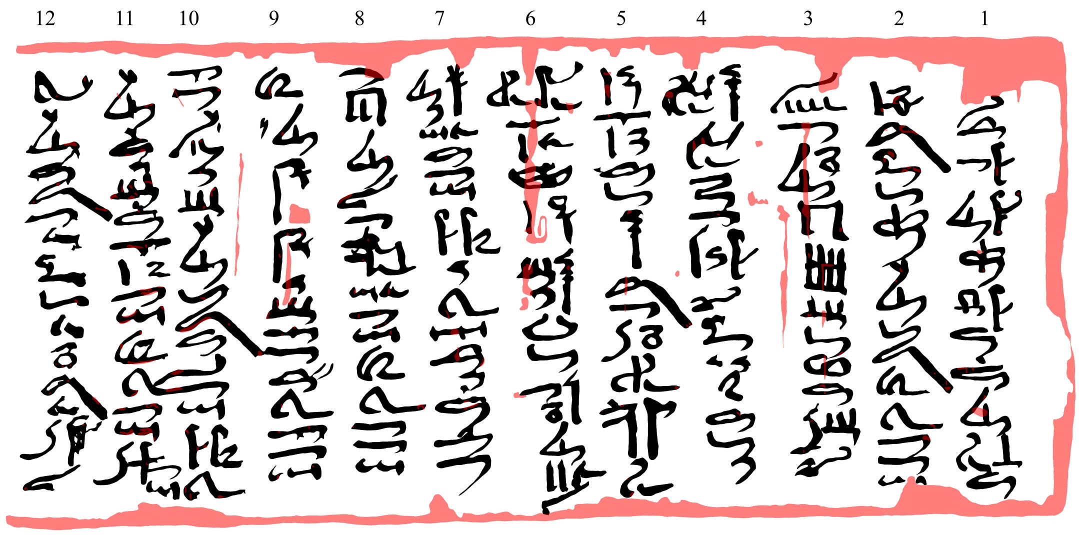

12
11
10
9
8
7
6
5
4
3
2
1

Figure A.17: P. Hermitage 1115, 1–12 (damage)

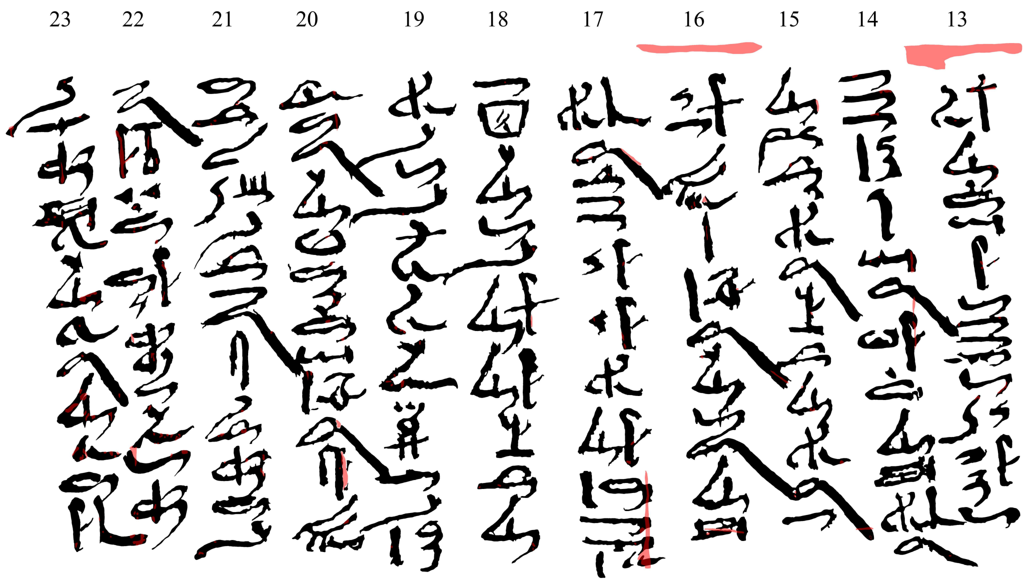


Figure A.18: P. Hermitage 1115, 13–23 (damage)

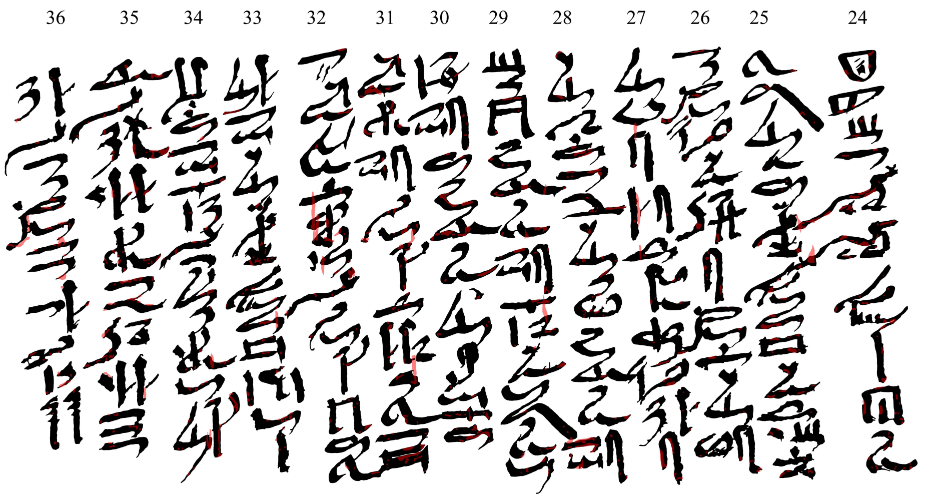


Figure A.19: P. Hermitage 1115, 24–36 (damage)

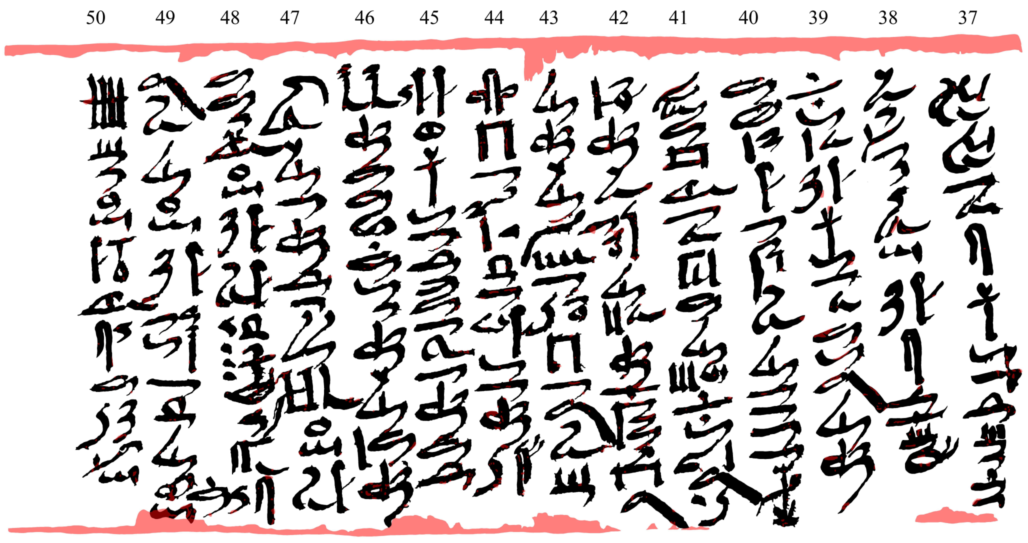


Figure A.20: P. Hermitage 1115, 37–50 (damage)

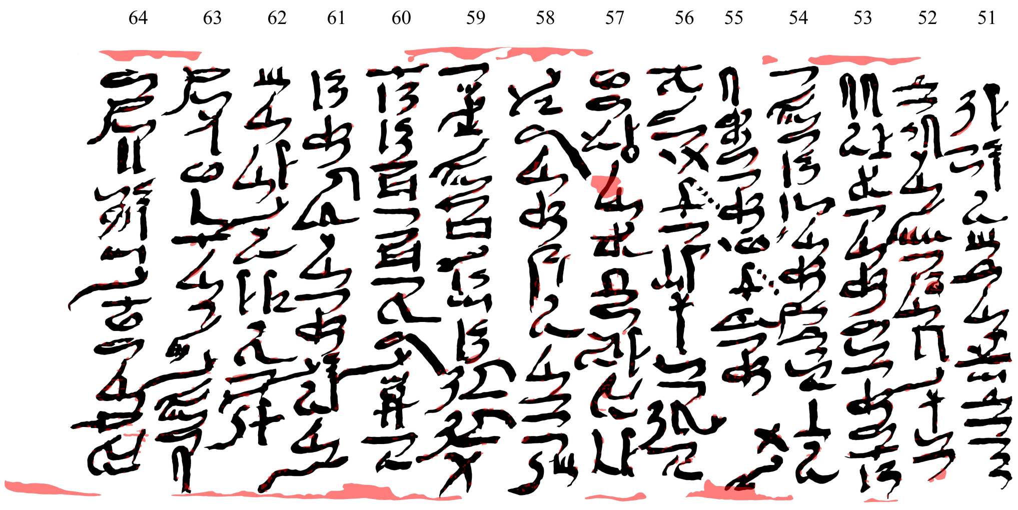


Figure A.21: P. Hermitage 1115, 51–64 (damage)

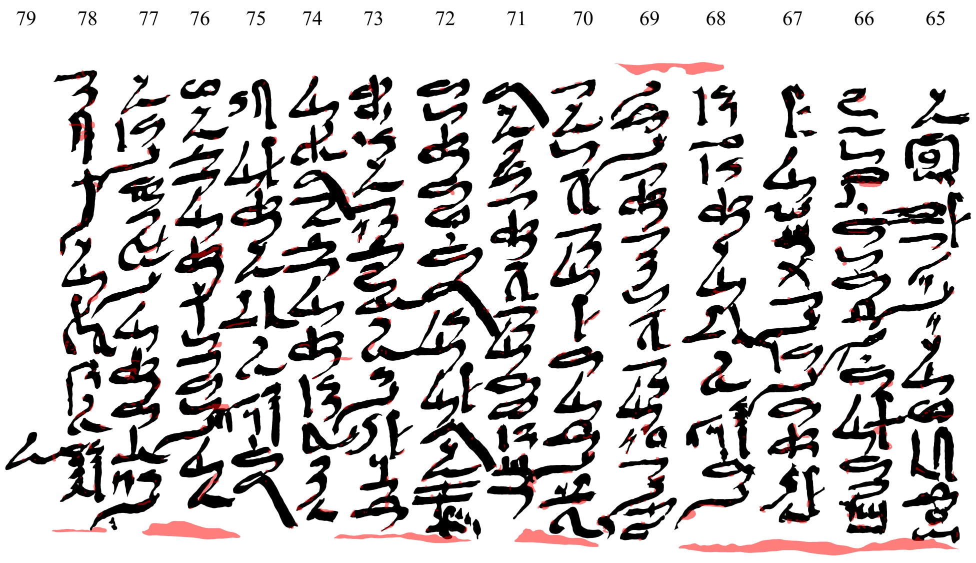


Figure A.22: P. Hermitage 1115, 65–78 (damage)

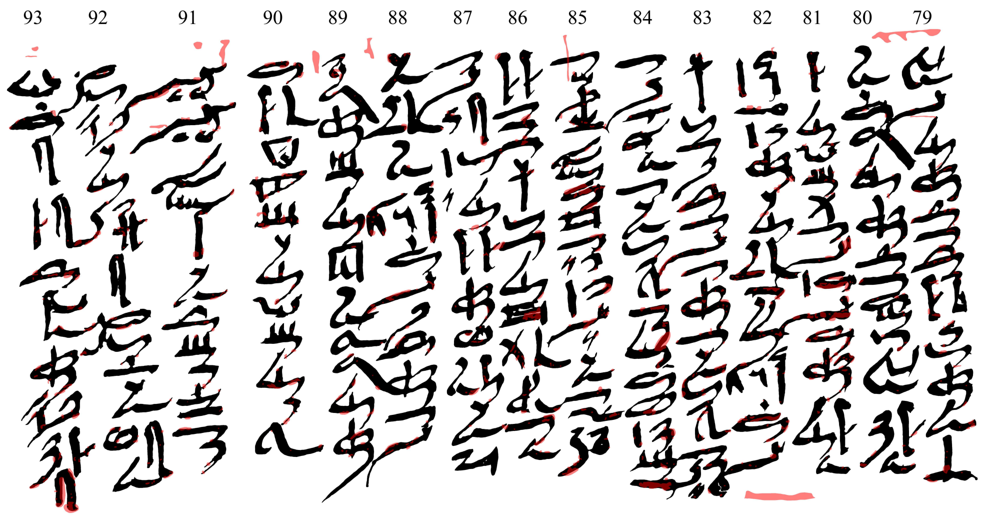
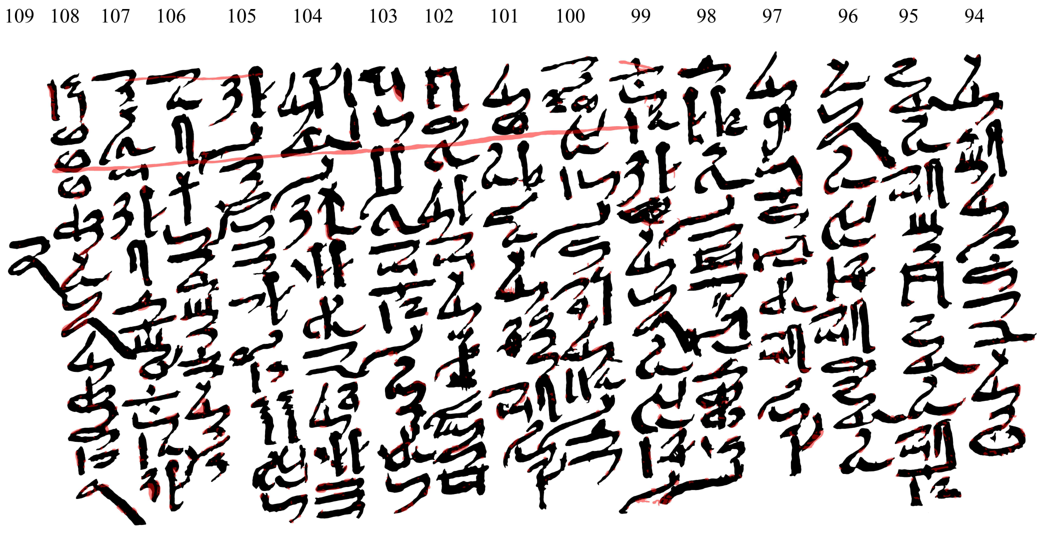
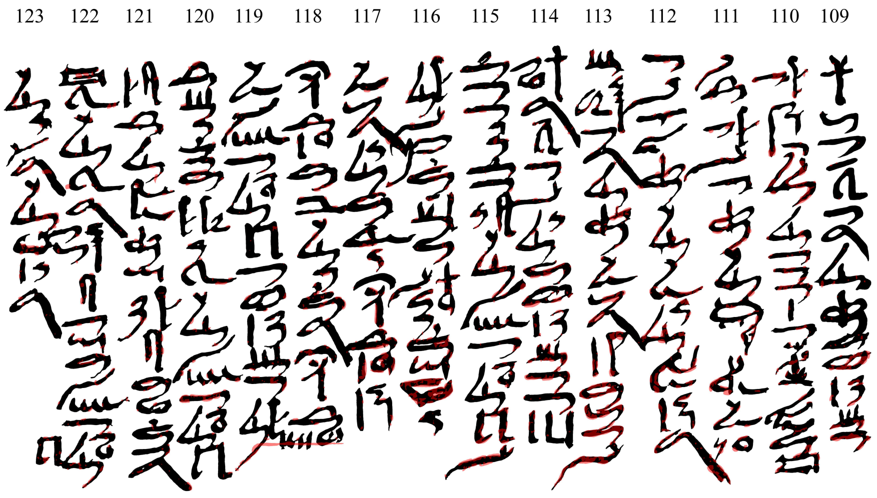


Figure A.23: P. Hermitage 1115, 79–93 (damage)


109 108 107 106 105 104 103 102 101 100 99 98 97 96 95 94
Figure A.24: P. Hermitage 1115, 94–108 (damage)


123 122 121 120 119 118 117 116 115 114 113 112 111 110 109
Figure A.25: P. Hermitage 1115, 109–123 (damage)

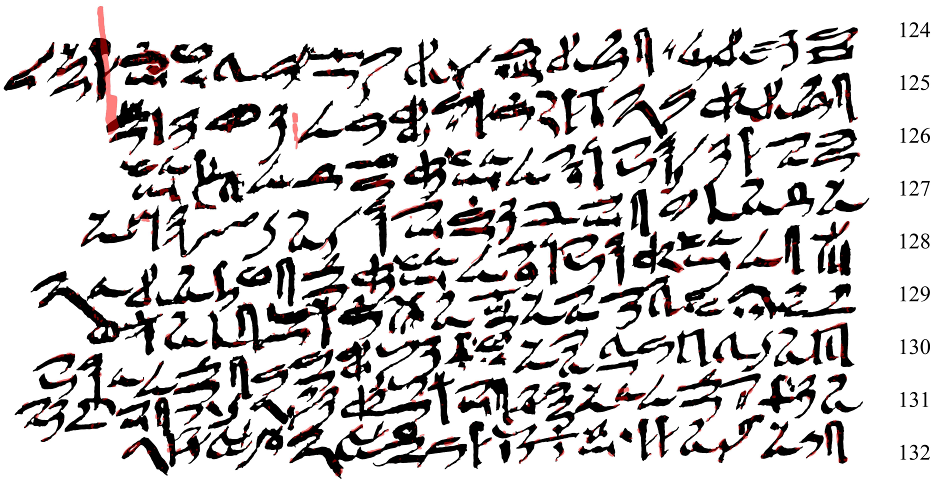


Figure A.26: P. Hermitage 1115, 124–132 (damage)

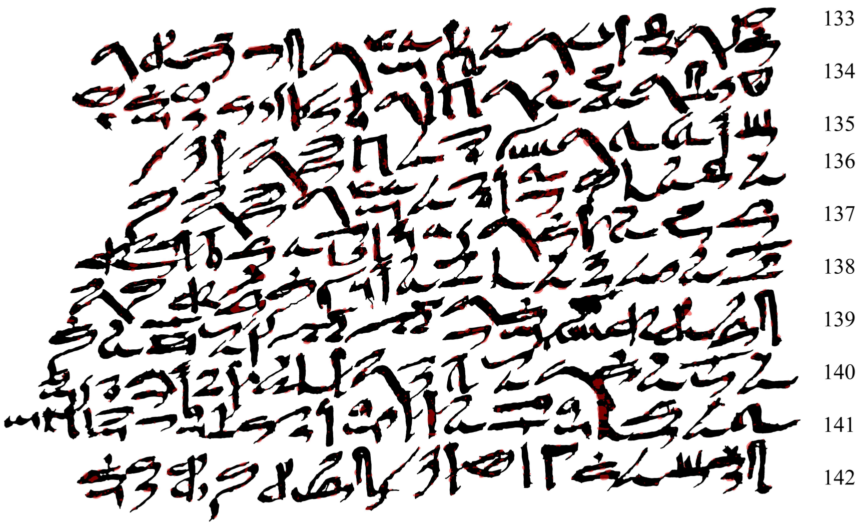


Figure A.27: P. Hermitage 1115, 133–142 (damage)

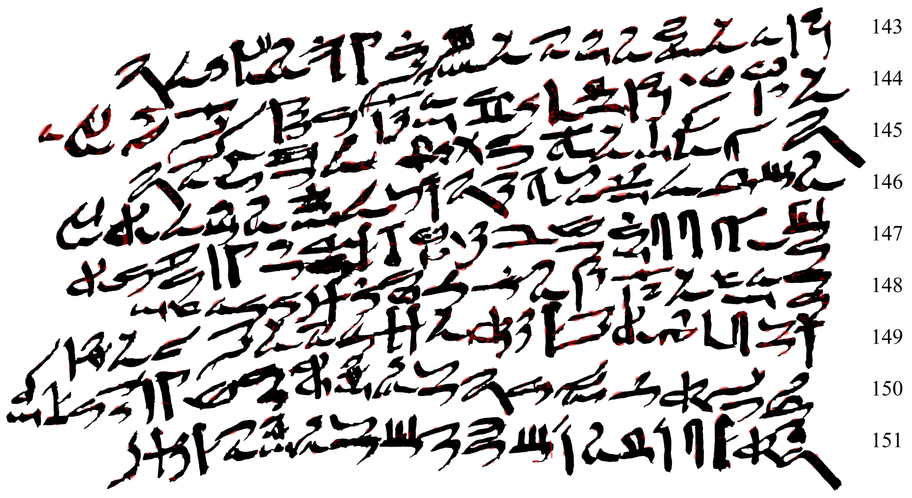


Figure A.28: P. Hermitage 1115, 143–151 (damage)

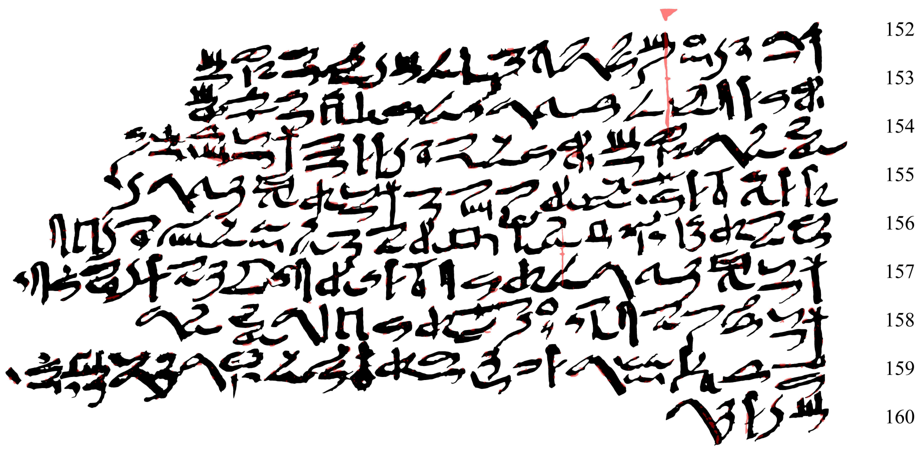


Figure A.29: P. Hermitage 1115, 152–160 (damage)

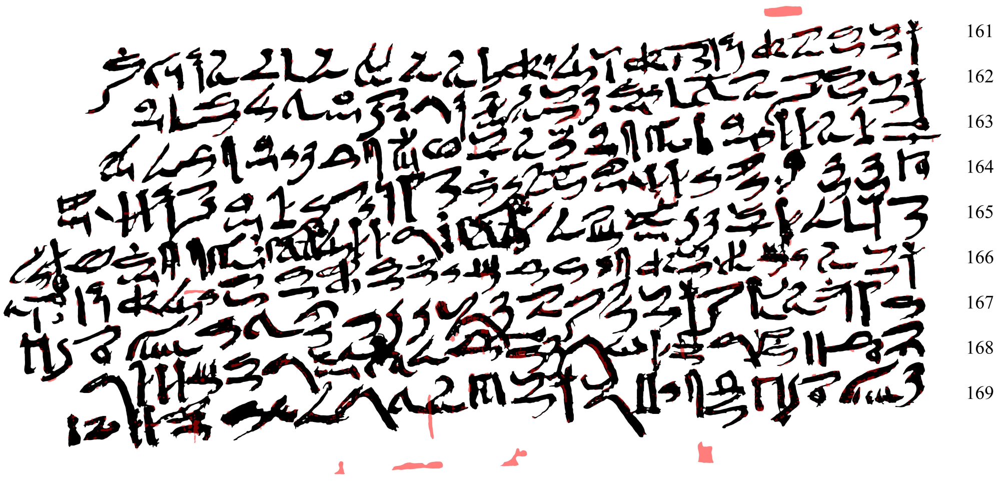


Figure A.30: P. Hermitage 1115, 161–169 (damage)

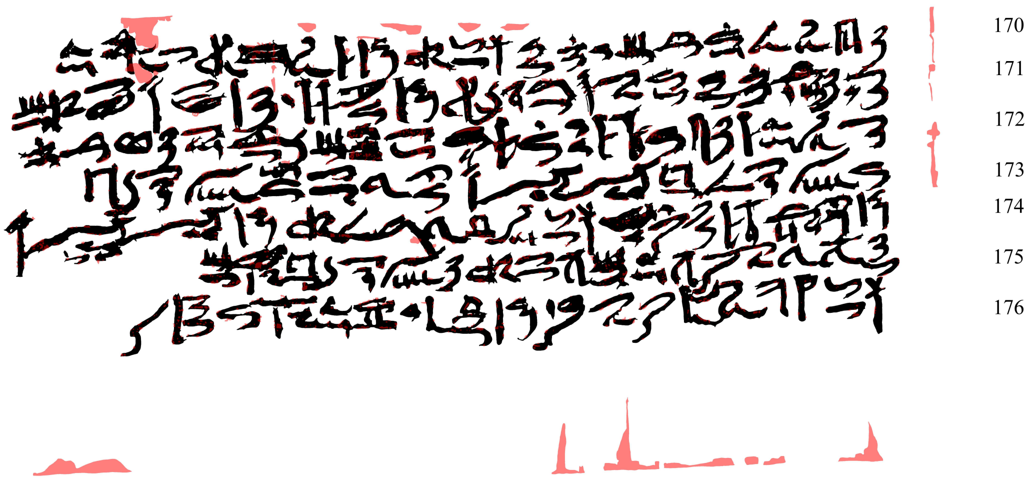


Figure A.31: P. Hermitage 1115, 170–176 (damage)

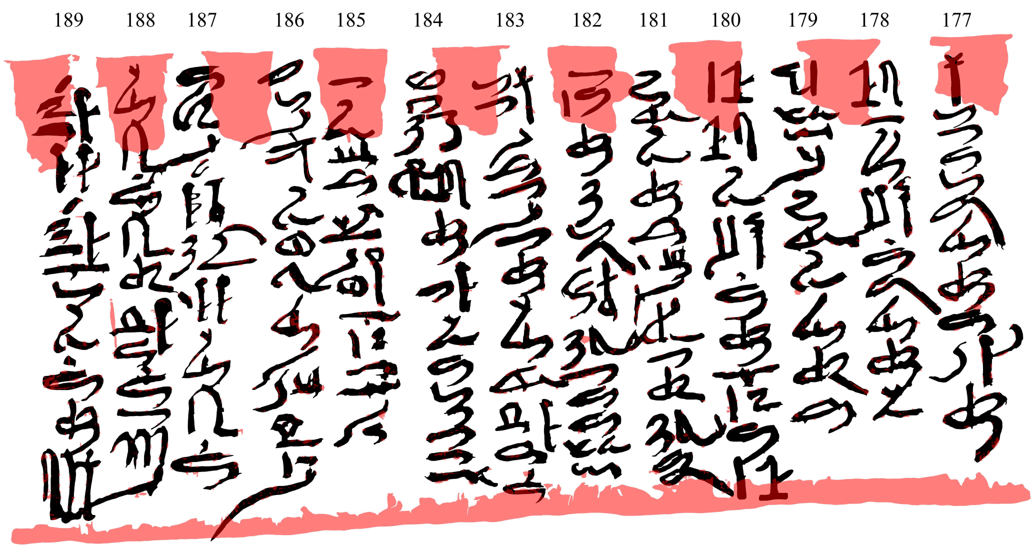

189
188
187
186
185
184
183
182
181
180
179
178
177

Figure A.32: P. Hermitage 1115, 177–189 (damage)

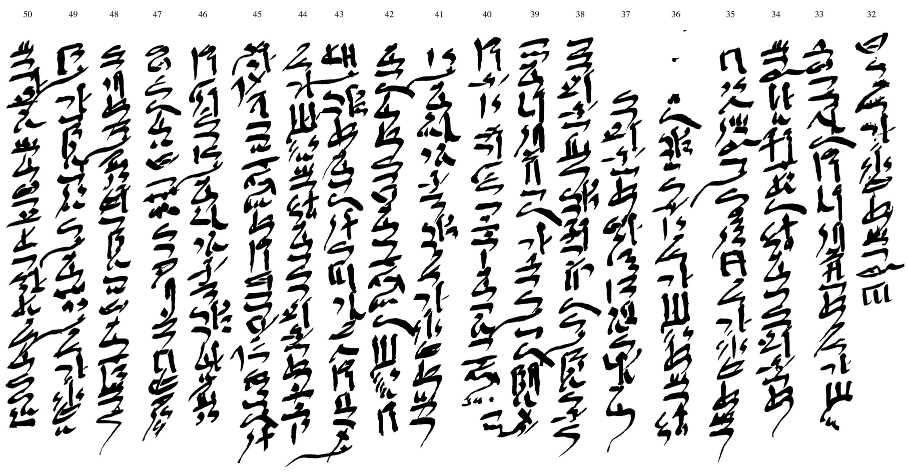

50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32

Figure A.33: P. Berlin 3023, 32–50

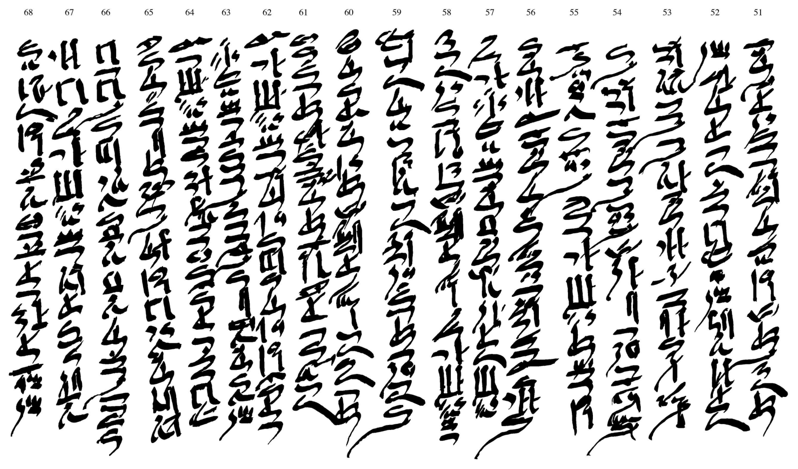


Figure A.34: P. Berlin 3023, 51–68

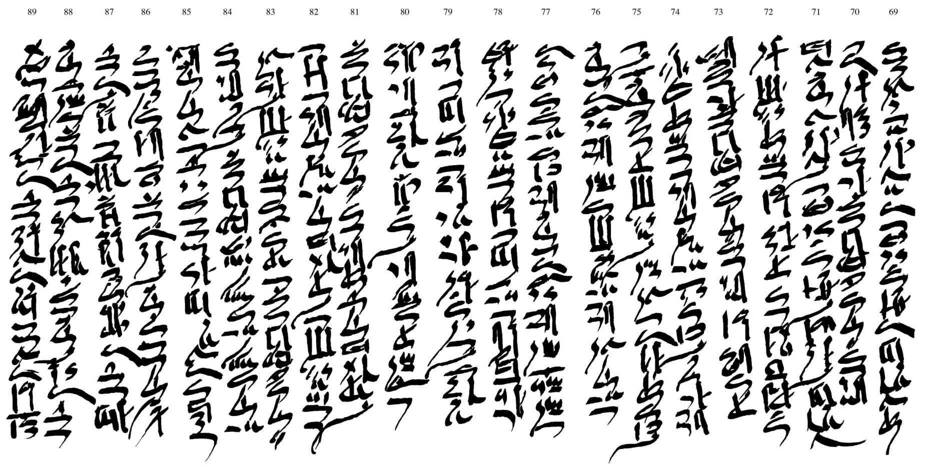


Figure A.35: P. Berlin 3023, 69–89

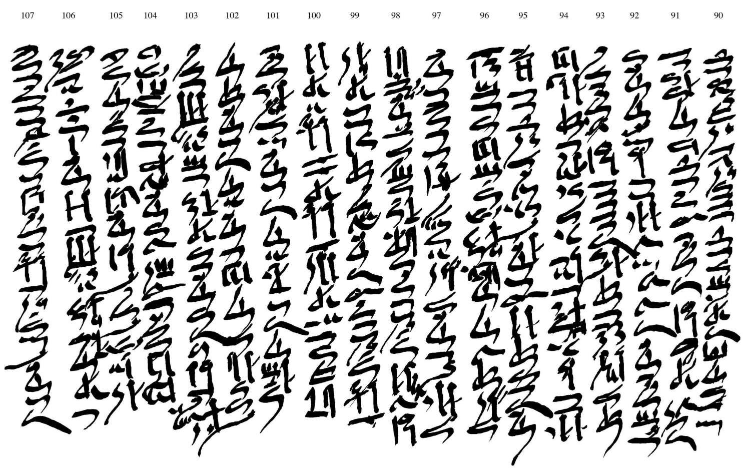

107
106
105
104
103
102
101
100
99
98
97
96
95
94
93
92
91
90

Figure A.36: P. Berlin 3023, 90–107

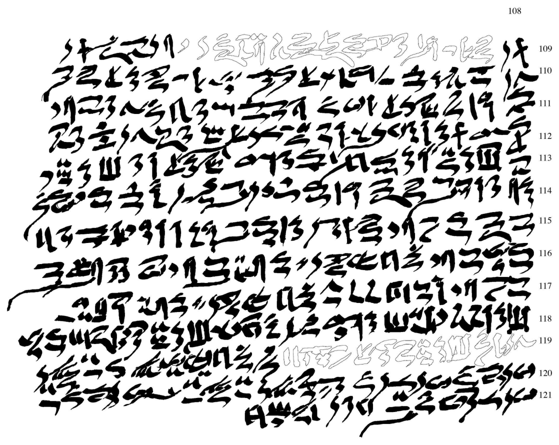

108
109
110
111
112
113
114
115
116
117
118
119
120
121

Figure A.37: P. Berlin 3023, 108–121