EPHEMERALITY AND THE ARCHIVE: MEMORY IN THE AGE OF DIGITAL REMEDIATION

A Thesis
Submitted to the Faculty of the
Graduate School of Arts and Sciences
of Georgetown University
in partial fulfillment of the requirements for the
degree of
Masters of Arts
in Communication, Culture and Technology

By

Nathan T Danskey, B.A.

Washington, DC
May 1, 2016
Copyright 2016 by Nathan T Danskey
All Rights Reserved
ABSTRACT

I built a digital agent to address problems of memory around digital archives and database culture. Using Python and open source libraries the program takes control of a digital archive on behalf of the user, remembering, forgetting, combining, and overwriting the archive’s contents based on user interactions. It tackles the theoretical concerns about the archive as a matter of fact – conditions that allow objects through their presence in the archive to generate objectivity – by treating those concerns themselves as matters of fact. Each loop of user interactions and agent actions transforms the scale, permanence, and objectivity of archival objects as matters of fact into a matters of concern. The user, archive, and agent together ask about archive, "What is there?"
The research and writing of this thesis is dedicated to everyone who helped along the way. To Dr. J.R. Osborn for your patience and guidance, to Dr. Matthew Tinkcom for opening a world of critical thinkers for me and my colleagues at CCT, and to Teresa Danskey, whose support and partnership makes our life rich and adventurous.

Many thanks,
Nathan Danskey
# TABLE OF CONTENTS

<table>
<thead>
<tr>
<th>Chapter</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>INTRODUCTION</td>
<td>1</td>
</tr>
<tr>
<td>EXERGUE</td>
<td>5</td>
</tr>
<tr>
<td>METHODOLOGY</td>
<td>9</td>
</tr>
<tr>
<td>WATCHING THE ARCHIVE</td>
<td>13</td>
</tr>
<tr>
<td>ON FORGETTING</td>
<td>19</td>
</tr>
<tr>
<td>ON REMEMBERING</td>
<td>23</td>
</tr>
<tr>
<td>ON COMBINGING</td>
<td>29</td>
</tr>
<tr>
<td>DISCUSSION</td>
<td>34</td>
</tr>
<tr>
<td>APPENDIX A: USER GUIDE TEMPLATE</td>
<td>38</td>
</tr>
<tr>
<td>APPENDIX B: INSTALL THE INSTALLERS</td>
<td>39</td>
</tr>
<tr>
<td>APPENDIX C: REMIX EXPERIMENT</td>
<td>42</td>
</tr>
<tr>
<td>CODE LIBRARIES</td>
<td>47</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>49</td>
</tr>
</tbody>
</table>
LIST OF FIGURES

FIGURE 1. PYTHON FUNCTION MONITORS A FILE DIRECTORY AND LOGS FILE PATH AND ACCESS TIME ........................................... 13

FIGURE 2. A HYPOTHETICAL ARCHIVE OF EIGHT FILES ALONG WITH A TIME SERIES OF USER ACTIONS ............................................. 15

FIGURE 3. PYTHON CLASS TO PARSE THE LOG IN PREPARATION TO DETERMINE WHICH MODULES TO RUN ........................................ 17

FIGURE 4. FILE ACCESS LOG WITH FORGOTTEN FILES HIGHLIGHTED . 19

FIGURE 5. PYTHON FUNCTION ADD A GAUSSIAN BLUR AND REDUCE THE VOLUME OF A VIDEO FILE ............................................ 19

FIGURE 6. OUTPUT OF NINE INSTANCES OF FORGETTING A MP4 FILE IN THE ARCHIVE........................................................ 21

FIGURE 7. THE FORGETTING OF ARCHIVED FILES....................... 22

FIGURE 8. FILE ACCESS LOG WITH RECALLED FILES HIGHLIGHTED. 23

FIGURE 9. FINDING HIGHLIGHTS OF A FILE BASED ON MEASURING AUDIO ENERGY OVER TIME ................................................. 24

FIGURE 10. HERE IS AN AGGRESSIVE LEXRANK SUMMARY OF THIS PAPER UP TO THIS POINT............................................... 26

FIGURE 11. ON REMEMBERING WHAT YOU CAN ....................... 28

FIGURE 12. FILE ACCESS LOG WITH COMBINED FILES HIGHLIGHTED .. 29

FIGURE 13. INTERWEAVING TWO FILES INTO A SINGLE MEMORY ..... 30

FIGURE 14. TWO OPERATIONS EACH COMBINING FILES OF DIFFERENT TYPES................................................................. 32

FIGURE 15. A MP3 AND AN IMAGE ARE COMBINED INTO A SINGLE MP3 WHICH IS SAVED AS BOTH FILES ........................................ 33
INTRODUCTION

Memory, as a topic of inquiry, crosses many of the disciplines and courses I engaged with as a student in the Communication, Culture and Technology (CCT) program. Theoretical concerns that fall under the interdisciplinary umbrella of memory studies were featured in courses I took on sociology, film, and critical theory.

A known problem in these fields, and not just to readers of Derrida, is the description of history as the memory of dominance and power. This problem of history can be described as a contestation over historical narratives. A contestation that utilizes a postmodern view of socially locating knowledge claims, analyzing alternative historical narratives, and deconstructing narratives and the structures that support them. My thesis project draws on this problem of memory, and uses the diverse interdisciplinarity of CCT to address a 21st instantiation of the issue: digital archives.

Digital archives sit at a convergence, a roundabout, of memory studies and the problem spaces around digital and automated systems that Lev Manovich terms “database culture.” The digital archive gives new form to these issues in its seeming ability to create permanent objects, the vast scales possible through digital storage and how those objects are made accessible to users of archives through digital agents. My project unsettles the current arrangement, creating what Bruno Latour would call a matter of
concern surrounding the technical details that maintain the digital archive’s files.

This project is itself a convergence of disciplines, made possible by five courses from my graduate studies, in chronological order: Contemporary Sociological Theory, Governing Algorithms, Critical Theory, Machine Learning, and Remix Practices. Contemporary Sociological Theory provided a foundation in postmodern thought, situating narratives in social space, and critical examination of social structures. Governing Algorithms explored the challenges of hidden (black boxed) functioning of algorithms and their social and policy implications; data optimism and encoding values through the design process. Critical Theory taught me the core skill of describing and synthesizing a dialectic; deconstruction through thoughtful examination, including identifying the conditions that make societal descriptions possible, and the amazing tool of detournment. Machine Learning provided me with a foundation in creating object oriented automated systems along with a variety of machine learning methods for powering my coding projects. Remix Practices introduced me to databases as a cultural form, and the use of remix as a critical analytic tool - the power of the chance encounter of a sewing machine and an umbrella on an operating table.

I began this project with the aim to create my own digital agent for a digital archive. One that would disrupt the archive by altering, rather than preserving, its files. This was at first a project of deconstruction, an attempt to uncover through a detournment of the digital agent the current precarious state of
digital archives. That is, a vast array of objects whose access is mediated by a digital agent – an agent whose proprietary algorithms mask its functionings. For example, a google scholar search result of many tens of thousands of entries in length is effectively much shorter because of the user interaction design, perhaps just the first few pages. The digital agent dominates – the search engine has privileged a handful of results for the user. This description of the problem provided the critical momentum to create an initial program to overturn that relationship. I merged ideas of human and cultural memory into the technical workings of computer memory. Through experimenting with a digital agent as an agent of digital forgetting I found that it was not the content of the files that were of concern but the files as “objects” in a larger collection of objects. That is to say, my work needed to point beyond the boundedness of the archive if it were to critically destabilize it.

The solution to this challenge came in the writings of a favorite thinker in CCT, Bruno Latour. In his own work Latour had become distressed by global warming denialists who were mobilizing his form of critique of the scientific community which traces the social conditions that make scientific knowledge possible. He suggested that where deconstruction fails perhaps a compositional (compositionism) approach would be able to create “matters of concern” out of matters of fact (2010). The object of inquiry is transformed and not destroyed. The difference is perhaps subtle, yet it proved crucial to my project. I built on to the forgetful nature of my digital agent an apparatus for creating compositions from the

3
archive. Put another way, the digital agent was redesigned to create an understanding of the objects of the archive not as objects (of fact) but as things (of concern). I utilized my background in video production and editing in the design process to encode how the digital agent approaches photos, audio, and video files in the archive.

The compositioning action of the digital agent is twofold. First, it works on the files with a toolset of manipulations to generate a “thingness” that breaks down the objectivity of the files. Second, the agent creates a hybrid composition of intentionality – me the coder, the coded agent, the archive and the users of the archive. The users of the archive can be provided with a simple set of instructions that explain the actions they could take to, in broad strokes, control the action of the script and permanently alter the archive (see Appendix A for a user guide template).

This thesis is constructed with code and written text to form a new rhetoric around the archive. The digital agent encodes a reading of the archive that is intended to transform the archive from a collection of facts, matters of fact, into a matter of concern.
Using code to alter a digital archive, this project reimagines the concreteness of a read-only database. It participates in the database culture of the current cultural moment while attempting a fictional escape from archives as the memory of dominance and power — a move from archives as matters of fact to matters of concern.

Before discussing the design and operation of the code at the core of this project, some framing statements will help situate this project and the problems it addresses. First, the postmodern shift in how memory relates to culture: what is remembered, who chooses what is remembered and who remembers it, and how it is remembered? (Erll, 2008). How societies remember is one of the central theoretical problems of memory studies. This understanding challenges the positivist view of a singular and objective history.

Second, acts of retelling a history, therefore, are understood as themselves a construction, built with artifacts of cultural memory, situated interpretations, and narratives that can be located in social space. Our ability to socially locate the discourses of memory emphasize how the rhetoric of the archive remains entrenched in modernist ideologies. Bruno Latour describes the modernist mindset toward history, memory, and archive making, as centrally concerned with matters of fact, "...there is no such thing as natural, unmediated, unbiased access to truth, that we are always prisoners of language, that we always speak from a particular standpoint, and so on, while dangerous extremists are using the very same argument
of social construction to destroy hard-won evidence that could save our lives” (2010). Our efforts to critique and situate historical narratives are grounded in efforts to “uncover or discover,” which are processes that rely on the positivist view that we can approximate, if not identify, the truth of the matter. This includes efforts of dismantling historical narratives, which leverage one standard of factuality against another. Latour suggests that we can circumvent this problem of history, of the archive, of modernity, by framing “matters of concern” in place of “matters of fact” (2004).

This is where my project attempts to conclude, with archives as matters of concern. To accomplish this outcome the project enlists the digital archive itself as the source of its own disentanglement. Jacques Derrida says of the archive that it “produces as much as it records of the event,” and “There is no political power without control of the archive, or without memory” (1996). This project takes control of an archive using code and user interactions to produce from within an archive a transformation from a matter of fact to a matter of concern.

Digital archives – databases – provide excellent media for this project to explore representations of factuality. As Elena Esposito explains, the database is a current site of cultural memory, and societal practices of remembering are based on its available technologies. For example, she traces the transition from oral to written practices, to photo and video, to the use of computer mediated storage (2008). The database as a cultural
practice therefore, affords a landscape of possibility in both its scale and its particular form as mediated memory.

The database of an archive, or any other type of database, is a numeric representation of its contents; objects must be flattened, remediated, to be brought into the archive. The substitution of the digital for the physical has a couple of consequences. First, the multidimensionality of an object, its size, texture, timbre, its ambient and other characteristics are flattened into a numerical standard and are altered, if not lost, in the process. Second, this remediation into a database translates artifacts into collections of referenceable discrete objects – which can be manipulated (Manovich, 2001). A hit song, comedic poem, team photo, or documentary film are all remediated as digital files in a folder and a row in a data table.

Yet, in the archive, as digital representations allow for infinite mutability, they gain an aura of eternal indestructability. A digital artifact does not fade from existence over time just as an ebook’s pages do not crease or grow stained with use. The project of preserving a digital archive, therefore, is not focused on the material condition of each object but ensuring the current database can be read into future databases – from one digital storage media to a future one. Such a project, that leverages the ease and scale of enumerating a digital archive along with the prospect of permanence creates the spectre of “dead Knowledge,” as Erll terms information that “merely rests on hard drives.” Dead knowledge is the gigantic collection of digital information that continues to
accumulate in the hope one day someone will find it useful. Which perhaps suggests that for the archive, knowledge is use. The discourse of digital archives is one of perfect memory – digital fidelity to an “original” object. These objects in historical archives are cloaked with matter-of-factness, simply through their existence in the database they justify their continued existence. The archive produces matters-of-fact.

This project addresses the archive’s acts of production-while-pursuing-stasis by operationalizing the difference between the read-only digital archive and potential of digital media to be revised, degraded, or even overwritten. The digital archive resists variability of its files that could compromise its ability to maintain an (the) “original” version – it only allows copying, modularity in display or transcoding (Manovich, 2001). The difference between the archive’s database and the potential of a database will be the site of action for set of python scripts written to bring variability to the archive.
METHODOLOGY

This project discusses the problems of the digital archive through a composition of python code and an archive ready at hand. The initial experiments for this project utilized remix practices realized through two python scripts to interact with digital video files to introduce ‘memory effects.’ One script slowly degrades (forgets) a video file over time. It opens a video file, reduces the volume by 5% and adds a bit of blur to the image. This Vimeo link (MemoryEffects: https://vimeo.com/143162328) displays one implementation that degraded the video file every twenty minutes, it displays a comparative result of three hours of “forgetting.” The second module blends a video and a text file, overlaying lines of a haiku poem over video frames. That experiment (RemixedRasa: https://vimeo.com/143101438) tested the potential of the collision of text and video artifacts from an archive, mixing memories which then generate additional meanings via juxtaposition from and for each “original” file.

These experiments led to the design of a full suite of tools that could be matched with an archive to create a complete set of outputs for any files in the archive. That is to say, a set of scripts and libraries whose functionings, and outputs, address digital archives as technical and political objects.

One of the initial challenges was how to do operations that were not dependent on randomness. While generating randomness in code is often useful for enriching an interaction, by making it less
predictable or adding complexity to a system, it did not translate well to the initial remix experiments. Randomized effects made the experiments appear more like vandalism – that is, an attempt to destroy instead of compose something that could stand apart from the original archive as a meaningful entity. Additionally, with the aim of writing this code to operate as rhetoric it was important to compose a system that was robust enough to operate with intentionality. Practically, this meant writing a set of scripts that could operate relatively autonomously and whose outputs were not arbitrary – randomness could be used as a tool but not as faux complexity. Therefore, it was a priority early on to identify methods that would yield outputs that could be tied to a chain of intentional action. This naturally also points back down the chain of action to the intentionality of the script author. It is script author intention to create outputs whose meanings (as a matter-of-fact) are distinct from their original files and are largely coherent. The code is written to enforce a transformation of the archive which distributes intentionality between the archive, the coder, the code and the user – to perhaps in a way address, as Latour would call it, a representation of agency.

There are two additional steps in the method of creating these scripts that assist in forming this hybrid composition. One is looping in users who may perhaps have nothing to do with the archive, or the code, and weave their intentionality into the system, whose choices have the potential to work against the archive or the code, or both. The second is an attempt on my part to avoid
overly specific adaptations of the scripts for the particular vagaries of a file in the archive, which would amount to an attempt to exert control on the potential outputs from the system. For example, knowing that an archive would have a collection of haikus I could write an if-statement to check a text for these poems before using natural language process tools. Similarly, expecting many documentary films I could optimize the scripts to better deal with those file types. However, in addition to more strongly opinionating what an output from those examples should be, such an approach would also privilege those specific types of documents over others in the archive.

The approach to encoding user actions in this project was to create a program that monitors a digital archive and watches as users interact with its content. This watcher of the archive, perhaps whimsically, is called AngelofHistory.py. Walter Benjamin uses that term to describe the endless march toward the future, "This storm irresistibly propels him into the future to which his back is turned, while the pile of debris before him grows skyward. This storm is what we call progress" (1973). This information is logged (into another database) and utilized by a master program to determine which files are operated on and which operations are used. This program is called Hippocampus.py as its changes in the files approximate some "memory effects." The Hippocampus.py utilizes the monitor script to control three main modes of working with files: degrading, distorting, and merging - which I have labeled:
forgetting, remembering, and combining. These modes provide a coherent structure for the code to operate within while being general enough to accommodate a rich variety of archival material - producing a robust set of outputs for its inputs.
WATCHING THE ARCHIVE

With critique, you may debunk, reveal, unveil, but only as long as you establish, through this process of creative destruction, a privileged access to the world of reality behind the veils of appearances. Critique, in other words, has all the limits of utopia: it relies on the certainty of the world beyond this world. By contrast, for compositionism, there is no world of beyond. It is all about immanence (Latour, 2010).

```python
#Angel of History.py
archive = 'TheArchive'

def main(archive):
    var = 1
    try:
        while True:
            theNow = time.time() - 5
            
            def modification_date(filename):
                t = os.path.getmtime(filename)
                return t
            
            def access_date(filename):
                t = os.path.getatime(filename)
                return t
            
            for root, dirs, files in os.walk(archive):
                for name in files:
                    if access_date(join(root, name)) > theNow:
                        r = join(root, name), "MODIFIED:", modification_date
                        result = str(r)
                        print result
                        f = open('ArchiveLOG.txt','a')
                        f.write(result+"\n")
                        print "--- result saved ---"
                        
                        #print "Done looking, The current time is %s" % time.time()
                        time.sleep(5)

Figure 1. Python function monitors a file directory and logs file path and access time.
```

While running, the AngelofHistory.py python script watches a file directory which stores a digital archive. It monitors walks
through any subfolders and is setup to capture the file path of any file that is accessed along with the time it was accessed and the last time the file was modified. Currently, the program runs once every five seconds and can store its results either in an external file or in a python data structure. Putting the access logs into a data structure such as class, with properties for file path and timestamps, allows the master program to run without needing to create its own archive of interactions. However, as is the case with most archives, there are practical benefits, persistence being one. Should it have to shut down, an external file allows the program to save the interaction data up to that point. Additionally, this file could be kept intact between cycles of the program as a record of all the interactions that generated the current iteration of the archive.

This log is used by a master script called Hippocampus.py that controls how all of the scripts of my project operate on the archive. Depending on the access time and file type, which can be multiple formats of text, photo, audio or video - the master script will run some of all of the python modules on the file. The master script analyzes the AngelofHistory.py log for looking for four types of events driven by users’ interactions. Figure 2 provides a diagrammatic representation of an archive along with a timeline for user interactions that were captured by the logging script. The master script can handle anywhere from a handful to thousands of files. So, the prototypical archive in figure 2 containing only eight files is enough to power all of the scripts that are
available. In our hypothetical archive there is a mixture of video, text, photos, and audio, marked with their common file extensions: mp4, txt, jpg, and mp3. The file extensions are representational; python, and this project, is able to work with many additional formats of audio, video, text and images, such as m4a, wav, mov, tiff, etc.

The master script utilizes the log produced by AngelofHistory.py to identify several types of interactions represented in figure 2 to control several python scripts. The program is generally intended to be run through many cycles and each cycle has a new set of log entries for the master script to work with. First, notice that files E and F in the archive that are not accessed by users in the time period being monitored. The Hippocampus.py script compares an inventory of the archive against the log and flags files that have not been accessed in the time period assigned to the master program's cycle - which can be adjusted from minutes to hours to weeks in length. In figure 2 the files E and F are flagged by the master script and assigned to be
operated on by the Forgetter script. Conversely, files A, C, D, and H were accessed more than once, and are assigned to the Rememberer script. A third major module of the program, Comingle, operates on files in the archive that were accessed within a short time of other files. This is shown in figure 2 through the overlap of H and G (at time 9) as well as A and D (at time 12). Note that A, D and H have also been assigned to the Rememberer script. The master program will first run the Rememberer scripts and then the Comingle script. This is done because these two scripts can produce competing versions of a file – forcing a choice of which version to keep or ballooning the archive’s size by saving both. The Comingle uses dialed back implementations of the Rememberer script in its operations thus it can also use the outputs of the Rememberer script instead. Finally, there is an affordance in the program for users to keep files from being operated on by these scripts: by accessing a file just once during the program cycle while also not accessing any other file within a few minutes before or after.

Figure 3 shows part of the Hippocampus.py script that parses files from the AngelofHistory.py. Each line of the log is an event which is separated into the file path, time it was last modified and time it was accessed.
#Hippocampus meets the Angel of History

class Event(object):
    def __init__(self, path):
        self.filepath = path
        self.modified = object[3]
        self.accessed = object[4]

@property
    def extension(self):
        return self.filepath[self.filepath.rfind(".")::]

    def tofileString(self):
        return ",".join([self.filepath, self.modified, self.accessed])

filepaths = map(lambda event: event[0], events)

#Generating a list of filepaths that will NOT be forgotten
for event in events:
    unique_filepaths.append(event[0]) if event[0] not in unique_filepaths

Figure 3. Python class to parse the log in preparation to determine which modules to run.

The Hippocampus script copies the files being operated until the master program cycle is complete. Then, the files in the archive are overwritten with the updated versions. This project is designed to create a series of iterations of an archive, each cycle perhaps losing, distorting, or combining the archive’s files, driven by the interactions of the system’s users. Each past iteration is lost as it is overwritten by the new files. This action is essential to this project – that matter of factness of the archive is overwritten and irretrievable. What remains is a trail of interactions captured by the log alongside the current form of the archive’s contents. This time-limited iteration and display of the archive acts to make its contents immanent. It’s ephemerality creates urgency around its contents. This project’s combination of user interactions, the code
and the archive act to create a possibility space around the original archive moving from factness to imminence. In addition to working on the matter of factness of the archive, this looping collection of code asks what “original” means in an archive. In each iteration of the program loop, originality becomes the transitory matter-of-factness of the objects in the archive that are at hand for the script’s use.
ON FORGETTING

Consequence: right on what permits and conditions archivization, we will never find anything other than what exposes to destruction, in truth what menaces with destruction introducing, a priori, forgetfulness and the archiviolithic into the heart of the monument. Into the "by heart" itself. The archive always works, and a priori, against itself (Derrida, 1996).

Figure 4. File access log with forgotten files highlighted.

```python
#Forgetter.py

import subprocess
import os
from skimage.filter import gaussian_filter
# Import editor aka open a sequence
from moviepy.editor import *
import time

##### Blurs a video clip #####

def blur(image):
    """ Returns a blurred (radius=2 pixels) version of the image """
    return gaussian_filter(image.astype(float), sigma=2)

for x in range (1, 10):
    clip = VideoFileClip("ForgottenRasa_" + str(x-1) + ".mp4")
    clip2 = clip.volumex(0.95)
    video = clip2.fl_image ( blur )
    TheVid = str("ForgottenRasa_" + str(x) + ".mp4")
    video.write_videofile(TheVid)
```

Figure 5. Python function add a Gaussian blur and reduce the volume of a video file.
The Hippocampus.py master control script passes files that have not been accessed during the program cycle to the Forgetter script. This module intervenes in the archive by degrading a file that has been neglected by the user in the current cycle of the program. With a video file, for example, it operates by reducing the volume by 5% and adds a slight Gaussian blur to the video frames. Figure 6 is a screen shot that shows the same timestamp of a video file through nine consecutive program cycles where the file was not opened. In this sample output, the module degraded the video file after every twenty-minute cycle and shows comparative result of three hours of time being neglected in the archive (the overall program cycle can be nearly any arbitrary length – five minutes, or 5 months). Note that the last two iterations display frames that are different than the others even while there were no changes in the code. This is most likely due to the degradation of the key frames that are interspersed throughout the video clip in the .264 codec. After about ten iterations the script is no longer able to operate on the file. However, the Rememberer script is still able to utilize forgotten files as part of the remembering of an image file (by overwriting the forgotten file).
In text files, a keyword analysis is used to find the one sentence that best represents the document (more detail on this process in included in the Rememberer section). This sentence is sliced out out the document and the remainder is written over the original input file.

The Forgetter module introduces fragility to the digital archive. It works to make a file’s contents eventually inscrutable to users who may finally come upon them, if indeed they are accessed at all.
For all of its work on the files in the archive this script asks if a user of an archive would notice forgotten files had been deleted, would they care? Thus, what’s at stake in the archive is not just what the user sees, it is the archive as a gestalt. A unit of meaning that at least partially derives its power from its contents.
ON REMEMBERING

All of these are snippets, they are only momentary representations of ongoing processes - as indeed the New Aesthetic is intended to be. Each image is a link, hardcoded or imaginative, to other aspects of a far greater system, just as every web page and every essay, and every line of text written or quoted therein, is a link to other words, thoughts and ideas. Again, the New Aesthetic reproduces the structure and disposition of the network itself, as a form of critique (Bridle, 2013).

Figure 8. File access log with remembered files highlighted.
Files in the archive that were accessed more than once during the program’s time cycle are passed to the Rememberer script. What is remembered from each of those files is first determined by the file type. There are three separate processes, for text, images, and files that contain audio.

For text files the program utilizes the natural language processing library (NLTK) to process text from files. In particular, the text summarization algorithms LexRank and KLsum are employed. The LexRank process estimates a sentence’s importance to the document by determining a ‘centroid’ sentence and then calculating each sentence’s similarity to it. This approach creates a set of

```python
# Rememberer.py --- RMS section

def liberoRMS(filename):
    y, sr = librosa.load(filename) # Load the waveform as y, sr is sample rate
    clipLength = librosa.get_duration(y=y, sr=sr)
    kValue = int(clipLength/1.3 +1)*10 # sets up relative ratio of samples

    ### get the RMS of the audio sample ###
    data = librosa.feature.rmse(y=y, hop_length=2048)
    boundaries = librosa.segment.agglomerative(data, k=kValue) # Agglomeration
    boundary_times = librosa.frames_to_time(boundaries, hop_length=2048) # ~.1s
    intervals = np.hstack([[boundary_times[-1], np.newaxis], boundary_times[1:], np])
    get_rms = librosa.feature.sync(data, boundaries, aggregate=np.max)

    nkValue = kValue-1 # because, for some reason, the intervals above leave out
    fixedN = np.delete(get_rms, nkValue, axis=1)
    npsTurn = np.concatenate((intervals, fixedN.T), axis=1)

    # transform from np array to regular list
    flatnps = npsTurn.tolist()
    rmsOut1 = sorted(flatnps, key = lambda x: int(x[2]), reverse=True)
    rmsOut2 = slice(rmsOut1[0:kValue/2])
    rmsOut3 = sorted(rmsOut2, key = lambda x: int(x[0]))

    return rmsOut3
```

Figure 9. Finding highlights of a file based on measuring audio energy over time.
sentences from the document that are closest to the main theme – its centrality – and can output any size subset from the input document. In my implementation I target a default output size of 12.5 percent of the input documents size. This can be calibrated to adjust rate at which documents are reduced in size.

The KLsum algorithm chooses sentences that minimizes Kullback-Lieber divergence. Which is to say it attempts to match the summary distribution of sentences to the document’s overall distribution. This means that the KL approach will pick up and carry forward additional sub-topics that the LexRank would discard as not having enough centrality. These two types of summary algorithms are included because they have the potential to generate very different summary outputs. This gives the program and future coders, who would want to build on this project, flexibility to calibrate it to the overall timeframe of a project. It can either quickly collapse text around a central theme or maintain a diverse set of themes in a document for a longer time. These algorithms assume a certain kind of text with a certain amount of cohesiveness and similarity. Without knowing the exact workings of archival digital agents, I would guess that they make similar assumptions. That is to also say, many of our digital agents may have known blind spots for content they ostensibly are meant to make available for the user.
In media files that contain audio, which includes video files, the audio analysis library librosa is used to identify sections that will be saved in the next iteration of the input file. The audio waveform is analyzed for root mean square (RMS) power over length of the file. RMS is a good measure of the energy of the audio signal at a given time and is a good proxy for segmenting areas of interest in the input file. From my experience in film and audio production, energetic and quiet moments are usually very important and this script attempts to capture this concern. The file is divided into segments that average 1.5 seconds in length. Helpfully, the RMS calculations are flexibly grouped with an agglomerative division of the segments - some can be much shorter or longer than their neighboring segments based on local RMS features. After this operation the script has a list of uneven time segments, with
beginning and end timestamps, and an RMS value, covering the range of the input file. This list can be sorted by RMS values and sliced so that only the most energetic segments are selected. Alternatively, the most and least RMS energetic segments could be selected or even deleted leaving only middle values. After the list is sliced and unwanted segments deleted, the list is sorted again by beginning timestamp so that once concatenated back together the clips are in the same order as the original input file. With the large number of segments and the agglomerative grouping the output file is often as coherent as the original input file.

Finally, image files are remembered forward into the next iteration of the archive by a third module that must operate in a different manner than the text or video modules. Both text and video have a linear element of time in their reading that is not present with a photo. This means that an approach of editing the flow of information in the file does not apply. The solution I adopted for this project was to go back to the AngelofHistory.py log file to identify files that have been ‘forgotten’ in either this or previous iterations of the archive. The image file is saved in the same location with the same name as the forgotten file. The forgotten file, in whatever degraded format it is in, is permanently deleted. All that remains is the notion of what the file once was through its file path. So in a sense, images can be double remembered.
This module of the project addresses how information in the archive can be remembered by those who access it. The code works with the user’s interactions with the archive to generate transpositions between object and remembered object.
They are not identical, but the exploded view and the ontograph have much in common. An anonymous, unseen situation of things is presented in a way that draws our attention to its configurative nature. An ontograph records the presence of many potential unit operations, a profusion of particular perspectives on a particular set of things (Bogost, 2012).

Figure 12. File access log with combined files highlighted.
In the final script of the program cycle, files that are accessed within a short time of each other, ten minutes for example, are brought together through the Combiner script. For text files, if their input files have already been operated on in the current cycle by the Rememberer script, the data arrays generated by that script are reused. With audio and video files only the final output of the Rememberer script is used. These files are processed again by a second RMS grouping algorithm that returns larger coherent chunks.
from each file. This type of output works much better for the main combining approach used in this project. Any files that have not been modified in the current program cycle are sent through a modified set of the Rememberer script to return a subset of the file roughly 50% the size. This makes the final output of the combination an average of the two input files sizes. It also has the practical effect of stopping the archive from ballooning in size, including the possibility of nesting the entire archive into a single file. Photos, as discussed in the Rememberer section, are not read by users linearly through time as the other file types and have a separate set of operations that are discussed below.

If files to be combined are of the same type the program zips the files together creating a composition of each file’s elements juxtaposed in a single file. For text, the file summaries are combined in an alternating fashion at the sentence level. Video files are similarly composed of alternating segments determined by RMS grouping. It would be more straightforward, perhaps, to append one file to another. However, the program yields more interesting compositions when disparate elements of the two files are held together. For example, the text of a haiku written for Georgetown’s Old North building by a student placed in the midst of an LA Times article about Henry Mosler paintings.

When images are brought together, an element of time is added as they are written into an animated gif which loops continuously between the two images (or more if eventually the gif is combined with additional images or another gif itself). As a note on
combining the files through python, figure 13 shows a function for iterating through two lists to combine their elements evenly. Notably, of all the itertools listed in the python documentation the one I used in this program included the only comment in the documentation crediting an author, in this case George Sakkis. This mention is a means of combining the comments of the python help documentation with my own work into the archive that will house this thesis.

In the cases where file types differ, the files must be composited together into a format that can accommodate both file types. Text is layered over video frames in one case, text is transformed into video frames with an mp3 audio track in another. Some more interesting combinations occur when an image is saved as the cover art for an mp3 file or when a mp3 is saved as the right
stereo channel in a video and the video’s initial input audio is saved as the left channel.

![Combining files](image)

*Figure 15. A mp3 and an image are combined into a single mp3, which is saved as both files.*

Combining files is the final module in a system which attempts to translate the intentions of the user forward into future iterations of the archive. In doing so, it affords meaning to arise for future openers of the file, from the chance encounter of the elements in each file.
DISCUSSION

This project engages in a discussion of digital archives as a mode of modernist ideological thinking - as made of objects and comprised of facts - through making my own object that does critical work on the archive. It is the made object, the collection of python code, that contains the substantive rhetoric of this work. As the sections of this paper describe the processes of the scripts - what the code does to the archive, it is intended to also do work for the reader. As Ian Bogost might say, this project incorporates the point of view of the program, its practices and what it generates. And, by doing so, it gives us additional insights into digital archives as products of modernist culture (2012).

The digital archive provides an excellent platform to do work on matter of factness. Lev Manovich helps us identify the transition to the database as the predominant cultural form (2001). To use a term from Derrida, the database has become a substrate of culture (1996). As such, memory which is not remediated into a database is in danger of being annihilated.

As a demonstration of what it means to participate in the database culture, consider Pope Francis’ visit to the US and the people who went to see him in person. As the moment of maximal proximity to the pontiff arrived, it coincided with the maximal amount of phone in front of face mediating reality. There are numerous photos of people lining the streets grasping the pope’s hand while capturing the moment with the other, often looking at the
phone and not the pope. Thus captured, how many of these amateur photos or videos are viewed again? However, that doesn’t matter. This practice, while undertaken by amateur photographers and videographers, does not demonstrate a lack of familiarity or sophistication with digital devices. Rather, it points to the importance of the cultural form of the database as integral to the experience at hand. In other words, it’s about that #DatabaseCulture.

Creating content for archives as an almost incidental part of participation drove my early interest in coding a script that effects those neglected files – to untangle the act from the object of memory. My work with these files helped me to realize that the coding project could provide meaningful insights into the problems of digital archives, and perhaps deconstruct matters of fact.

Derrida suggests that deconstruction is a program that emerges from the system itself rather than from an external action (1996). For part of its functioning, this project uses the files that compose the archive to deconstruct it. For example, in the Forgetter section the script acts to degrade files that through the (in)action of the user are already omitted from public view. They have already been forgotten and begin the work of dismantling the archive even before my script touches it.

Consider contemporary archives, at scale, such as a library database. Its many thousands or millions of objects are too numerous for one person to sift through. The objects in the archive act to obscure the other objects of the archive, simply through their presence. Thus, files that no one has accessed and that continue to
be neglected still serve to obstruct the discovery of those objects users wish to find. The archivist’s response, who first insists that all objects should be persevered for future discovery is to build a search engine on top of the archive to help users grapple with its magnitude. This results in the sequestration of the archive, the user on one side, the archive on the other and a digital agent the presents search results in between. Consider, how often do users go to the eighth page of any search results. Which is to also ask, do search engines already act to deconstruct the digital archive?

This project began by working with neglected files of the archive in an effort to deconstruct the archive. However, the project reaches its critical potential when it creates a composition around the digital archive. It uses python code to create a bridge between the digital archive and its users. The code supports the users’ intentions, deleting what they neglect, remembering what they most access, and combining that which the users bring together. This composition triangulates an experience that may be recognizable to observers as generating a space of concern around the archive. As Katja Kwastek might say about the python code, it creates artifacts that allow for “ephemeral action” to be done on the archive (2013).

Objects that are accessed in this project or any other archive cannot be permanently foregrounded. A photo makes an impression, a key quote or feeling is remember. Archived objects are remembered as referenced objects, through direct citation or a summative narrative, or as some remembered thing. This, as my project works to
give the impression, is what constitutes archives in the first place, references to matters of concern turned matter of fact.
This archive is special. As you explore the archive’s contents the archive itself will share in your experience... It will permanently change because of how you interact with it.

If you want to control this change, here are a few things you can do:

**Nothing.** That’s right, forget about what might be in the archive and move on. For any file in this archive that is neglected today the archive will forget about too, sometimes deleting part of the file or fading a video or audio file.

**Open a file once.** If one person opens a file just once today, the archive will move it into the next iteration of the archive, as-is.

**Open a file more than once.** For any files that are viewed many times today the archive will look at it too, doing its best to ‘remember’ the highlights. It will keep the most exciting parts for future viewers.

**Open two or more files within three minutes of each other.** If you’re interested in viewing a couple of files together, so is the archive. It will take these files and blend them together so that others can see these files together as well.

Everyday a new version of the archive is created. At the end of each day the archive uses all of the interactions above that have taken place throughout the day to create a new version of the archive. It replaces the old version which is deleted.
APPENDIX B: INSTALL THE INSTALLERS

here is the GitHub repository for the project code: https://github.com/danskey/matters_of_concern

Instructions for installing the libraries and other dependencies to run the project code on a Mac computer. Any commands listed below are intended to be typed into the terminal window. Note: while you can attempt installing these into a virtual environment such as virtualenv - some of the packages may not function properly. In this list a “Main Package” are those libraries which I directly coded with, i.e. used their documentation. Also, “sudo” in the commands below requires your password - not always needed but is included below for easy of use for all users.

Install the Installers

1. Xcode - Make sure you have Xcode, namely the Xcode command line tools. You can get Xcode for free from the app store (make sure you open the program at least once! and agree to the terms), enter the following in the terminal:

   `xcode-select --install`

2. MacPorts - go to this url to download the installer (https://distfiles.macports.org/MacPorts/MacPorts-2.3.4-10.11-ElCapitan.pkg). Install the package.
3. Homebrew - Enter this in the terminal to install:

   `/usr/bin/ruby -e "$(curl -fsSL https://raw.githubusercontent.com/Homebrew/install/master/install)"`

4. Use Homebrew to install Python (this may not always be needed but it will provide a complete and updated version of python). Enter the following in the terminal:

   `sudo brew install python`

5. pip - Enter the following in the terminal:

   `sudo easy_install pip`

6. Rudix - Enter the following two lines in the terminal:

   `curl -O https://raw.githubusercontent.com/rudix/mac/rpm/2015.10.20/rudix.py`

   `sudo python rudix.py install rudix`
Install the Dependencies

7. ImageMagik – Enter the following in the terminal (this one takes a few minutes):

```bash
sudo port install ImageMagick
```

8. Scipy Suite – This also includes Numpy and related libraries (including ones you do not need for this project such as matplotlib and the ipython notebook, but may be of interest to include). Enter the following in the terminal:

```bash
sudo port install py27-numpy py27-scipy py27-matplotlib py27-ipython +notebook py27-pandas py27-sympy py27-nose
```

9. Pillow – Enter the following in the terminal:

```bash
sudo pip install Pillow
```

10. NLTK – Enter the following in the terminal:

```bash
sudo pip install -U nltk
```

11. NLTK Libraries – Enter the following in the terminal:

```bash
sudo python -m nltk.downloader -d /usr/local/share/nltk_data all
```

Install the Main Packages

12. FFmpeg – Enter the following in the terminal:

```bash
sudo brew install ffmpeg
```

13. moviePy – Enter the following in the terminal:

```bash
sudo pip install moviepy
```

14. librosa – Enter the following in the terminal:

```bash
sudo pip install librosa
```

15. sumy – Enter the following in the terminal:

```bash
sudo pip install sumy
```

16. mutagen – Enter the following in the terminal:

```bash
sudo pip install mutagen
```
Install Additional Libraries

17. libsndfile - Enter the following in the terminal:

```
sudo brew install libsndfile
```

18. libsamplerate - Enter the following in the terminal:

```
sudo brew install libsamplerate
```

19. scikits.samplerate - Enter the following in the terminal:

```
sudo pip install scikits.samplerate
```

20. mpg123 - Enter the following in the terminal:

```
sudo rudix install mpg123
```
APPENDIX C: REMIX EXPERIMENT

This is a report on the initial remix experiments in Fall 2015 that led to the thesis project. It was produced for J.R. Osborn’s Remix Practices class.

“”” Can ephemerality address archival flattening? “””

def INTRODUCTION & THEORY():

The act of retelling history is itself a construction, built with artifacts of cultural memory, situated interpretations, and narratives that can be located in social space (Foucault). This understanding challenges the modernist/positivist notion of a singular and objective history. The postmodern paradigm shift complicates objects of cultural memory, whose context is lost over time, particularly as practices shift and working knowledge changes. Artifacts that are brought into the archive are thus flattened as they are preserved (Derrida).

How societies remember is one of the central theoretical problems of memory studies (Erll). The presentation of the artifacts in the Old North Archive, whether they are newly written haikus or original plans for the building, are reinterpreted for each telling, by the presenter as well as the listener (Bolter). The Omeka database of the Old North archive numerically remediates the objects of the archive and instantiates a “blend of human and computer meanings,” (Manovich). It is a Latourian hybrid, which acts on the practices of collection and storage of artifacts – culminating in digital artifacts.

def REMIX JUSTIFICATION():

This project turns on how the digital archive differs from the database in that it de-emphasizes variability as a feature of the database in favor of stasis. Variability is one of Manovich’s five features of new media and the database. The other four being numerical representation, automation, modularity and scalability. Derrida offers a definition of the collection and storage of
cultural memory as archival violence – an “unnatural” means to achieve societal memory. It is possible that my experiments in generating ephemeral databases will provide a path toward more “natural” remembrances. Mayer-Schönberger, for example, suggested that forgetting helps us look toward the future.

Ephemeral data structures, as opposed to digital archives, are designed to exist in a state of continuous change. The rise of digital media socio-technical systems such as Snapchat and Secret, followed the mass consumer utilization of the database. The culture of the database, and ephemeral media creation, exist in parallel in the individual. Snapchat users, whose shared photos are permanently deleted after being viewed for up to ten seconds, are many of the same smartphone users who mediate a ‘memorable’ event through dozens of photos and videos – perhaps never viewing them, yet permanently storing them. Thus both ephemerality and permanence are valued and enacted through these behaviors. Perhaps ephemerality can add vitality to the digital archive of cultural memory as a more complete representation of cultural practice.

def EXPERIMENT DESIGN():

Working with the Old North archive provides a tangible means to work with the theoretical concepts around memory, the database and ephemerality. This project is based on a series of experiments that blend ephemerality and the Old North Pilgrimage Project digital archive. The project addresses theoretical questions raised at the intersection of the concepts around memory and the digital archive discussed above.

For the initial stage of project I created two software modules that can interact with a digital archive to introduce ‘memory effects’ based on user interaction with a database. One script slowly degrades (forgets) a video file that is neglected in the archive. It opens a video file, reduces the volume by %5 and adds a bit of blur to the image. This vimeo link (MemoryEffects: https://vimeo.com/143162328) is one implementation that degraded the video file every twenty minutes and is a comparative result of three hours of forgetting.

The second module blends two files that are accessed in proximity to each other. In this example (RemixedRasa: https://vimeo.com/143101438), a number of haikus written for the Pilgrimage Project are pulled from a text file and computationally composited unto a proximal video file. It creates a collision of two artifacts from the archive, mixing memories that generate additional meanings for each ‘original.’
def RESEARCH THROUGH MAKING(results):

Conducting the remix experiments and working on a python script to monitor how users interact with the archive yielded key insights for the project. First is the construction of the monitor program, which loops every five seconds through a file directory and subfolders looking for new access time stamps. It’s creation helped me to see how central its functioning, and the users who drive it, are to the project overall. When the monitor program notes a new file access, it logs the file name with directory path, and time stamp and writes it to a text file. Implicit to this is the user – examining test logs made clear that the user is a critical operator in the intentionality chain of this project. That is, it takes my intentionality of archival disruption and moves it from procedural to something more cooperative and transitory that may better reflect memory.

The user’s logs are a proxy for the experience the user had in the archive. It must be noted that this proxy is its own flattened digital remediation of that experience. However, with the log and archival material the python scripts may be able to triangulate a new experience that may be recognizable to the user who ‘created’ them.

The construction of the monitor program also clarified how the overall program operates. The figure below captures the key insights at this phase in the project. It includes the users’ intentional interactions as discussed above which led to an awareness of the necessity for a timescale under which the program operates. In a 24hr time cycle for one loop of the entire program ‘Memories’ are logged for the day, which may include many access stamps for the same file. After the day is over, the program’s monitor sleeps and the modules do work on the day’s memories based on three access characteristics. In one instance, a file read in proximity to another file, another is a stand alone file open, and the third is no action on the file that day. The output of the operation is saved to the ephemeral archive, below called the Frilthy Archive. Note, if the monitor did not sleep, it would not distinguish the user’s interactions from the modules work on the files.
This is the conceptual model for how the project does work on the archive based on the interactions people have with it.

```python
def FURTHER RESEARCH():

An additional observation from the construction of the monitor and the infographic above is the possibility of adding a quality score to the log file. This could act as a multiplier on the module’s effects, either heightening or diminishing its effects. This could be based on time that a file is kept open or perhaps even a computer vision application could quantify the users’ interaction in a fashion. Memories, after all, are not created equally, and it would be useful for this project to try to capture the quality of the interaction users’ experience.

A complementary hardware component has not been proposed by this project. An interface design that extends beyond the keyboard, mouse, or tablet touch may give the archive a greater sense of immediacy. In it’s current form there is not much to entice the user to interact with the archive.

And additional unanswered question is the extent to which users should be clued in to the operations of the system. Both that they are viewing a modified rendition which is itself ephemeral, and that their actions will effect the archive moving forward. Would users dismiss the archive? Perhaps some will attempt to game its
functionings, in which case, does that endanger the experiment - or alter what the experiment measures.

def CONCLUSION():

This project is the beginning of a larger thesis that explores how the procedure and method of constructing a set of modules to do work on the archive enriches the discussion surrounding ephemerality and the digital archive. The set of modules, along with the digital archive, constitute a set of research tools. These tools leverage practical manifestations of the relevant theoretical intersections, as this project has already shown. The methods and the “Frilthy Archive” make my research question visible and they allow the question to be experienced by the Georgetown community. In doing so, this set of research tools yield novel insights into the underlying theoretical questions.

def REFERENCES():

• Bogost, Ian. Alien phenomenology, or, what it's like to be a thing. U of Minnesota Press, 2012.
CODE LIBRARIES


FFmpeg: A complete, cross-platform solution to record, convert and stream audio and video (https://www.ffmpeg.org/).

Homebrew: The Missing Package manager for OS X (http://brew.sh/).


librosa 0.4.1: Python module for audio and music processing (https://pypi.python.org/pypi/librosa/0.4.1).


libsndfile: C library for reading and writing files containing sampled sound (http://www.mega-nerd.com/libsndfile/).

MacPorts: Commandline program installer (https://www.macports.org/).


Moviepy 0.2.2.11: Video editing with Python (https://pypi.python.org/pypi/moviepy).

Mutagen 1.31: Read and write audio tags for many formats (https://pypi.python.org/pypi/mutagen).

NLTK 3.0: Natural Language Toolkit (http://www.nltk.org/).

numPy 1.11.0: Array processing for numbers, strings, records, and objects, scikit (https://pypi.python.org/pypi/numpy/1.11.0).

Pillow 3.2.0 Python Imaging Library (Fork) (https://pypi.python.org/pypi/Pillow/3.2.0).

pip 8.0.2: The PyPA recommended tool for installing Python packages (https://pypi.python.org/pypi/pip/8.0.2).

Rudix 2015: a collection of pre-built Unix software delivered as packages for OS X (http://rudix.org/).

scikits.samplerate 0.3.3: A python module for high quality audio resampling (https://pypi.python.org/pypi/scikits.samplerate).
SciPy 0.17.0: Scientific Library for Python (https://pypi.python.org/pypi/scipy).

sumy 0.4.1: Module for automatic summarization of text documents and HTML pages (https://pypi.python.org/pypi/sumy).
REFERENCES


