Filmuniversität Babelsberg KONRAD WOLF

Master of Creative Technologies

In Search of Serendipity:

Applying Actor-Network Theory to Zettelkasten
and How Natural Language Processing Can Help
Manage Difficulties Arising from
Zettelkasten's Bottom-Up Structure

Author

Tillman Jex Schäuble

Filmuniversität Babelsberg KONRAD WOLF

Matriculation Number: 33745

Berlin, 2025

Reviewers

Prof. Dr. Angela Brennecke

Filmuniversität Babelsberg KONRAD WOLF

Prof. Dr. Michael Witt

Berliner Hochschule für Technik

Schäuble, Tillman Jex:

In Search of Serendipity: Applying Actor-Network Theory to Zettelkasten and How Natural Language Processing Can Help Manage Difficulties Arising from Zettelkasten's Bottom-Up Structure

Master's Thesis in Creative Technologies

Filmuniversität Babelsberg KONRAD WOLF

Abstract

This thesis explores the process of applying conceptual theory to the development of software. The problem domain is a method of note-taking called Zettelkasten, which is renowned for a said ability to give rise to the emergence of new ideas — including ideas that were not placed in it by the note-taker themselves. Zettelkasten implements a simple set of guidelines, one of which is the storing of all files in one directory. As it is argued, this creates a set of difficulties, but also a paradox; that the system benefits by all files being stored in one directory, but that such disorganisation can lead to the Zettelkasten becoming overwhelmingly chaotic and riddled with dead ends.

To best develop software to help manage these difficulties, the underlying system should be well understood. But how to understand such an abstract ”idea generation” system, that is made of one human and a network of static text files? And how to compartmentalise that understanding into the design of software features?

Specifically, the software being developed will implement natural language processing features. Designing machine learning based software features for Zettelkasten carries an extra responsibility because Zettelkasten is a method used to help in the process of critical thought. A component of this thesis is therefore the discussion of how to approach the implementation of machine learning technologies in a manner that supports critical thought, rather than reducing it.

There are two products of this thesis. Firstly, a documentation of how to apply conceptual theory from outside the domain of software development, to software design. And secondly, a functional open-source software solution to help manage the difficulties of Zettelkasten's bottom-up structure.

Zusammenfassung

Diese Masterarbeit beschäftigt sich damit, wie theoretische Konzepte bei der Softwareentwicklung angewendet werden können. Untersucht wird dabei die Notizmethode Zettelkasten, die dafür bekannt ist, neue Ideen zu fördern – auch solche, die dem Nutzer nicht bewusst waren. Zettelkasten beruht auf einem einfachen Satz von Richtlinien, von denen eine das Speichern aller Dateien in einem einzigen Verzeichnis vorschreibt. Wie argumentiert wird, dies führt zu einer Reihe von Schwierigkeiten, aber auch zu einem Paradoxon: Das System profitiert davon, dass alle Dateien in einem Verzeichnis gespeichert sind, aber diese Unordnung kann dazu führen, dass der Zettelkasten unüberschaubar chaotisch und voller Sackgassen wird.

Um Software zu entwickeln, die bei der Bewältigung dieser Schwierigkeiten helfen kann, muss das zugrunde liegende System gut verstanden werden. Doch wie lässt sich ein derart abstraktes „Ideen-Generierungs“-System verstehen, das aus einem Menschen und einem Netzwerk statischer Textdateien besteht? Und wie lässt sich dieses Verständnis in die Gestaltung konkreter Softwarefunktionen überführen?

Die zu entwickelnde Software wird insbesondere Funktionen aus dem Bereich der Natural Language Processing (NLP) implementieren. Die Gestaltung von Softwarefunktionen für Zettelkasten mit maschinellen Lernen für Zettelkasten bringt eine besondere Verantwortung mit sich, denn der Zettelkasten ist eine Methode, die den Prozess des kritischen Denkens unterstützen soll. Ein Bestandteil dieser Arbeit ist daher die Auseinandersetzung mit der Frage, wie maschinelles Lernen so implementiert werden kann, dass es kritisches Denken fördert, anstatt es zu untergraben.

Es gibt zwei Produkte dieser Arbeit. Erstens eine Dokumentation der Anwendung der konzeptionellen Theorie von außerhalb des Bereichs der Softwareentwicklung auf das Design von Software. Und zweitens eine funktionale, Open-Source-Software, die dabei hilft, die Schwierigkeiten der Bottom-up-Struktur von Zettelkasten zu bewältigen.

Contents

Chapter 1: Introduction

As part of the Master's degree that this thesis finalises, I wrote an essay titled *Floss, a Partial Antidote to Social Catastrophe*[1]. It put forward the idea that free and open-source software as an activity can help maintain a necessary degree of ownership, community and identity in an era where those qualities are iteratively being stripped from us by Big Tech.

It was the very first time I enjoyed writing an essay, and also the first where I used the Zettelkasten system to both take notes throughout classes and to construct the essay. Throughout the semester, the note-taking process felt more like a thinking task than an archival task; I felt like I was doing half the work of the essay, just by taking notes within the framework of this new system. I received a great mark and at this moment could take a step back and fully appreciate this curious system which I'd invested my trust, time and energy into, and which had just paid its first dividend. In the previous semester, I also had to write a similar styled essay, but had not adopted Zettelkasten at that point. The experiences of writing the two essays were drastically different. It was clear that Zettelkasten was what created that difference, and so I became keenly interested in this system and how to better use it.

Zettelkasten is shrouded in a mystique, born largely from the requirement to almost totally relinquish organisational control over the notes — instead relying on emergence and serendipity to extract value from them — but also due to its creator, Niklas Luhmann, who shifted from a career in public service, to one of the most influential German sociologists of the 20th century with remarkable velocity. Luhmann's body of work encompasses approximately 50 books and 550 articles as well as his principal contribution of Systems Theory, all generated within an approximate 30 year timespan after he began working with Zettelkasten in the 1960's[2]. Of note, is that Luhmann did not claim total credit for his prolific and eclectic output. He also gave significant, if not equal credit to his "conversation partner", his Zettelkasten.

[1]Jex, *Floss, A Partial Antidote To Social Catastrophe*.

[2]Cevolini, "Niklas Luhmann's Card Index: Thinking Tool, Communication Partner, Publication Machine".

"What follows is a piece of empirical sociology. It concerns me and someone else, namely my Zettelkasten." — Luhmann, *Communicating with Slip Boxes by Niklas Luhmann*.

1.1 Approach and Contributions

This thesis investigates and frames Zettelkasten as a tool for thought; an interactive data system that is constructed from simple, yet scalable rules (chapter 2, *Zettelkasten*). It furthermore seeks to demystify the said ability of Zettelkasten to produce ideas that were not directly placed in it by the user themselves, particularly when Zettelkasten is renowned for the unconventional approach of having no filing system at all. Typically, one would expect that in order to produce great ideas with a note-taking system, there should be a considerable measure of organisation. How else should one keep a track of things?

"But how is it possible that the filing system could develop a creativity of its own; that is to say, how could it systematically lead to ideas that do not lie at hand?" — Cevolini, "Niklas Luhmann's Card Index: Thinking Tool, Communication Partner, Publication Machine", p. 295.

The power of ideas is inescapable. Whether you are reading this on paper or on a screen, the lineage of technologies and ideas that have allowed this to happen is astounding. As such, to investigate a thinking tool that has been shown to offer great ability to facilitate ideation, is a highly relevant exercise given that any research task, including a master's thesis, should ideally start with a good idea. The writer and philosopher, Aldous Huxley, once said, *"People will come to adore the technologies that undo their capacity to think"*. Zettelkasten is an example of the contrary.

The goal of this thesis is therefore to analyse how Zettelkasten works as a system, what difficulties can arise from its bottom-up structure, and how machine learning based software can be applied to help manage these difficulties.

In doing so, this thesis contributes a theoretical approach to understanding the Zettelkasten system as a tool for thought, and uses this theory to inform the development of open-source software that may aid the user of Zettelkasten to mitigate the common difficulties of the method.

Scope

While the software being built as part of this thesis aims to mitigate some of the difficulties of Zettelkasten, it could be expected that the software improves the digital Zettelkasten workflow. Or at the very least, that the software has some measurable effect on the user experience of a digital Zettelkasten workflow.

While user studies are valuable for assessing usability and efficacy, the central aim of this thesis is not to validate a specific implementation, but to explore how a theoretical concept can be applied to inform the design of software. The primary contribution is thus theoretical and design oriented, rather than empirically evaluative on a user experience level. Therefore, a usability study or otherwise quantitative or qualitative study of the software's efficacy is not a part of this thesis. Suggestions are however made as to how a user study could be undertaken in chapter 8, *Limitations, Conclusion and Future Work*.

1.2 Thesis Methodology and Structure

This study seeks to achieve the aforementioned goals by the following methodology:

- Introducing the Zettelkasten method and its difficulties of bottom-up structure (chapter 2, *Zettelkasten*),
- using Actor-Network Theory to better understand how Zettelkasten works on a systems level, thereby helping inform the software design (chapter 3, *Actor-Network Theory*),
- presenting the machine learning techniques applied in the practical project and why machine learning is a fitting technology for the given problem (chapter 4, *Machine Learning*),
- describing how the theoretical components combine to inform software design (chapter 5, *Software Design*),
- reporting on what did and did not get implemented from the design (chapter 6, *Design Implementation*) and finally,
- results, discussion, limitations and conclusions (chapter 7, *Results and Discussion* and chapter 8, *Limitations, Conclusion and Future Work*).

Chapter 2: Zettelkasten

Zettelkasten is a note-taking methodology made famous by the civil servant turned sociologist, Niklas Luhmann. After formalising the Zettelkasten method in the 1960s, he wrote some 50 books and 550 articles in a 30-year time period spanning an eclectic range of topics[1]. Importantly, he attributed this prolific output not to himself, but to his Zettelkasten, which he referred to as his ”conversation partner”[2]. Likewise, within Zettelkasten and knowledge management research at large, it is common to see the method referred to as a ”second brain”[3]. Luhmann himself referred to it as his ”second memory”[4]. This language hints at a common understanding of the Zettelkasten method – that it operates in a highly abstract, yet profound way[5].

The term Zettelkasten is a German word that translates literally to *slip-box* (where a ’slip’ is a slip of paper). This name was adopted because Luhmann’s Zettelkasten was a cabinet of index cards, the kind used at the time in libraries and public offices to store references of catalogued information shown in figure 1 on the next page.

[1]Cevolini, “Niklas Luhmann’s Card Index: Thinking Tool, Communication Partner, Publication Machine”.

[2]Luhmann, *Communicating with Slip Boxes by Niklas Luhmann*.

[3]Aal and Rüller, “From Personal Knowledge Management to the Second Brain to the Personal AI Companian”; Cevolini, “Where Does Niklas Luhmann’s Card Index Come From?”

[4]Schmidt, “Niklas Luhmann’s Card Index”, p. 55.

[5]Cevolini, “Niklas Luhmann’s Card Index: Thinking Tool, Communication Partner, Publication Machine”, p. 295.

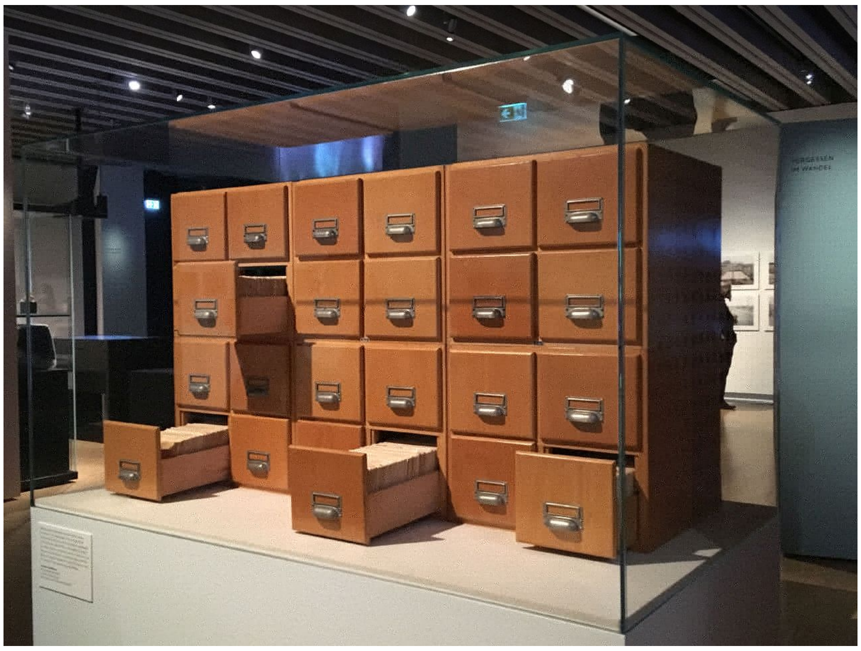


Figure 1: Luhmann's Zettelkasten on display in Frankfurt[6].

To help clarify the intentions, goals and results of this study, it is necessary to show how the Zettelkasten method works. Through this, the functionality of the method should become clearer and thus provide a backdrop to understand its strengths and weaknesses. This should also help in understanding how and why the theoretical concepts of Actor-Network Theory are applied to software design, and what effect this has on managing difficulties of bottom-up structure.

2.1 A Medium for Thought

"Unfortunately, the method of generation is never clear even to the 'generators' themselves." — Asimov, *Isaac Asimov Asks, "How Do People Get New Ideas?*

Before tackling the integration of Actor-Network Theory with Zettelkasten, it helps to present how Zettelkasten is being understood within the scope of this thesis: that Zettelkasten is a *medium* for thought. In doing so, a framework and language can be established that helps to better understand the later theory and practical components of this thesis.

[6]Source: `https://historisches-museum-frankfurt.de/vergessen`

A ”medium for thought”, is a development of the widely used term, ”tool for thought”. Tools for thought pertain to systems, methods and technologies that affect the way we work with and process information[7]. Examples include maps, grids, spreadsheets, lists and extend to concepts like mathematical and programming notation, in that notations function as tools of thought by compressing complex ideas and offloading cognitive burdens[8].

Thinking itself can take different forms, and this thesis has *critical thinking* in scope when working with Zettelkasten. Bloom’s taxonomy[9] is used here to define critical thinking. Bloom and his colleagues classified a hierarchy of critical thinking objectives, which is built from knowledge recall, comprehension, application, analysis, evaluation and synthesis. Understanding critical thinking within these confines helps to understand what parameters a tool for thought should have, and why tools for thought are important. A tool for thought should therefore aid in some or all of these processes.

Alan Kay suggests that a more appropriate term is a *medium* for thought, which constitutes a *collection* of tools[10]. In this case, tools in the Zettelkasten medium would be index notes, hub notes, links, tags and the *Folgezettel* (described in the following section). This modification of terminology, that Zettelkasten is a *medium* for thought, is adopted in this thesis. It allows the machine learning solutions built as part of this thesis to be classed as additional *tools* available within the medium. With this classification it becomes easier to evaluate and discuss whether the design of the software program is in line with the goals of this thesis.

2.2 The Zettelkasten method

A Zettelkasten is similar to what we know today as a wiki; a collection of interlinked documents. What makes Zettelkasten different is its system of organisation, which allows for emergent network structures to evolve.

The Zettelkasten method typically begins by writing a note that encapsulates a singular and concise thought, question or any other piece of information deemed valuable by the note-taker.

[7]Sarkar, “AI Should Challenge, Not Obey”.

[8]Iverson, “Notation as a tool of thought”; BouJaoude and Attieh, “The Effect of Using Concept Maps as Study Tools on Achievement in Chemistry”.

[9]Bloom, *Taxonomy of educational objectives: the classification of educational goals.*

[10]Kay, “Multimedia”.

Each note is given a unique identifier (UID), traditionally using a sequential alphanumeric system. Much like the pages of a book, the very first note would receive the identifier *1/1*, where *1/...* denotes the first section (topic matter) of the Zettelkasten. The second note in the same section would be assigned *1/2*, then *1/3* and so forth[11].

At this point, there is little difference than writing one note per page in a bound notebook. But what happens when the notebook is filled until page 50, but the note-taker, concerned with a new research task at hand, finds that a note on page 15 invites further development? Where in the notebook should this extension to the current note be written? And how can the note-taker be sure that the relationship between the two notes remains intact?

Suddenly, the notebook with its rigid format of sequential pages bound in place, reveals itself as a severe constraint. In contrast, every note in the Zettelkasten represents a finite, extendable data point that can be easily cited and incorporated into an endless arrangement of relationships with other notes[12] (how these relationships are made is explained in due course).

A logical solution to the restriction of a notebook would be to shift from a book of *bound* pages, to a book of *loose* pages. I.e, a ring binder. With this format notes can be added *between* existing notes. This removes the organisational constraint of *insertion order* when taking notes, allowing the note-taker to extend trains of thought without limit regardless of where the note exists.

Functionally, there is no difference between using multiple ring binders for Zettelkasten or, as Luhmann did, draws of index cards. The defining factor is that the medium must allow notes to be *inserted* and not just appended. As such, the ring binder can be seen as a technological innovation to the notebook as a tool for thought. In contrast to a bound book, the note-taker is able freely extend pre-existing ideas, while also ensuring the ongoing record of the idea's development maintains a logical ordering, and therefore remains useful.

In order to facilitate the insertion of notes between pre-existing notes, Luhmann implemented an alphanumeric UID. This allowed for any train of thought to be further developed, or to be branched from with alternate versions of the thought, as illustrated on the following page.

[11]Schmidt, "Niklas Luhmann's Card Index".

[12]Krajewski, *Paper Machines*.

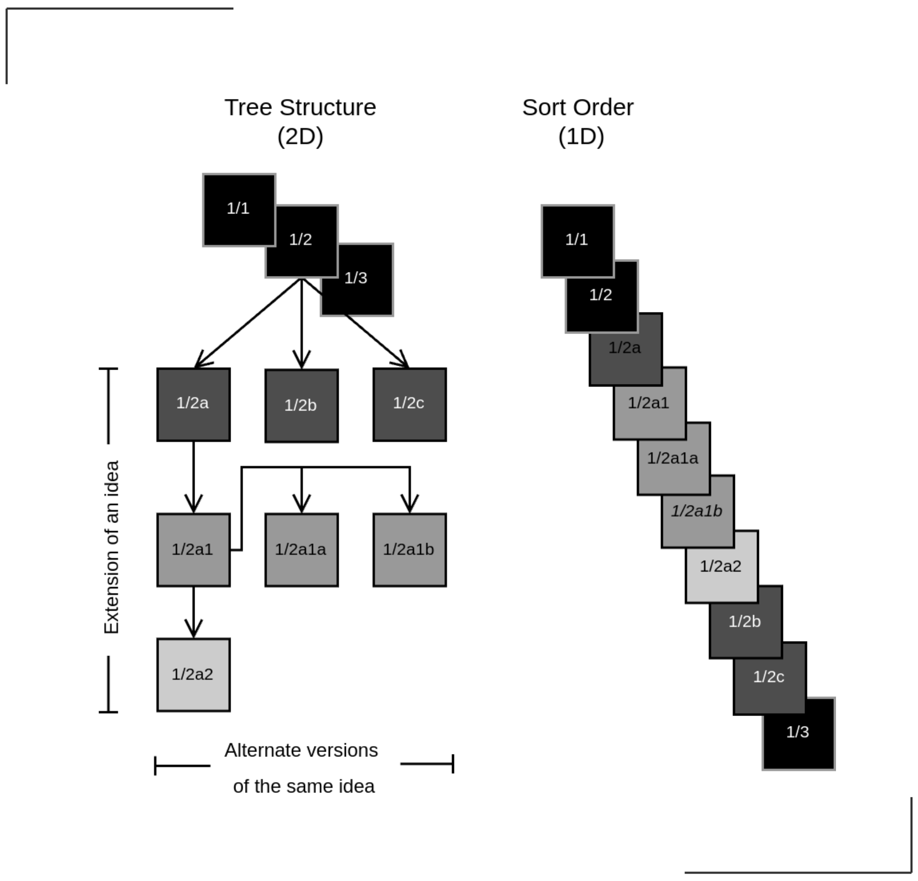


Figure 2: How UIDs enable the Zettelkasten to grow inwardly.

Explanation: On the left, the extension of ideas (vertical) and the alteration of ideas (horizontal) are shown. On the right, the same notes are shown sorted from lowest UID to highest UID (which is how the notes are stored, in a drawer, ring binder or computer directory alike). This illustrates how the UID system simultaneously allows for the limitless extension and modification of ideas while maintaining sort order. Maintaining sort order ensures that locating notes manually is possible, for example when following outbound links from note to note.

The UID ensures that the sequence of thought is recorded and retrievable. Each subsequent note that continues where the previous note left off, is known as a *Folgezettel*. Given the above diagram, the notes *1/2*, *1/2a*, *1/2a1* and *1/2a2* represent a continual train of thought. Importantly, the train of thought *branches* at note *1/2a1*, whereby differing, yet related ideas are documented in *1/2a1a* and *1/2a1b*. This ability to effortlessly branch from any note allows for the emergence of tree structures, allowing the Zettelkasten to "grow inwardly"[13].

This nature of organisation that Luhmann employed with Zettelkasten was not born from a forethought in design. It was the solution to the problem of assigning topics to each note. This is a common point of cognitive friction when managing knowledge in a top-down organisational structure (such as directory structures), and is summed up by the question *"Where to store this*

[13]Luhmann, *Communicating with Slip Boxes by Niklas Luhmann*.

note?". His solution was to assign the starting fundamental idea or question with a topic (where topic *1/...* may be *"Thinking Tools"*), but all successive notes would not need to be. They only had to follow one rule, which was to be linked to the note that inspired their creation[14].

However, while the UID system allowed Luhmann to extend and branch his thoughts without constraints, it did not provide a straightforward way to find notes again. This is the difficulty faced by all users of the Zettelkasten method. To address this, Luhmann maintained a keyword index: a curated list of keywords or thematic terms, each pointing to one or more notes in his Zettelkasten. These were often the initial starting fundamental idea or question, as described in the preceding paragraph. This index acted as a thematic gateway into the otherwise non-hierarchical system, providing a practical solution to the challenge of locating relevant ideas without imposing a rigid structure[15]. By consulting the index, Luhmann could quickly identify where in the Zettelkasten discussions on specific topics began and jump directly into those relevant areas[16]. This functionality is akin to the table of contents of this thesis, or the index section of a book.

The Zettelkasten method transforms the question *"Where to store this note?"*, to *"What does this note connect to?"* or as Ahrens writes in *How To Take Smart Notes*, *"In which context do I want to stumble upon this note again?"*[17]. Indeed it is this very practice of finding relationships between notes that allows Zettelkasten to grow from a filing system to a tool for thought[18]. In this, Zettelkasten provides a method to build complex, yet easily navigable webs of relationship between thoughts, observations and literature research while also enabling the ability to think, and record the *development* of those thoughts, in a granular yet structured manner[19].

[14]Meschede et al., *Serendipity*.

[15]This is in effect the process of topic modelling, as will be discussed later on.

[16]Schmidt, "Niklas Luhmann's Card Index", ch. 6.

[17]Ahrens, *How to take smart notes*.

[18]Krajewski, *Paper Machines*.

[19]Meschede et al., *Serendipity*.

Within typical note-taking paradigms, it is not unusual to rewrite notes by discarding sections of the existing note, or the old note in its entirety. In Zettelkasten, notes do not die or get discarded once used. This ’permanence’ is what enables the thought process to be recorded. In this sense, note *3/2a5g98* is the current end of a long train of thought that is born at note *3/2*. Anyone who has solved a challenging problem understands that there is immense value in understanding *how* the solution was developed. In other words, the journey to the destination. The thought process itself is no different, and this process of tracking the journey to a thought is fundamentally embedded in the Zettelkasten process. As such, the Zettelkasten becomes a navigable web of connections, which are embedded records of the note-taker’s idiosyncratic way developing ideas, and therefore serves as a tool for recall, comprehension, application, analysis, evaluation and synthesis of thought.

2.3 The Difficulties of Bottom-Up Structure

> *”Order is information that fits a purpose. The measure of order is the measure of how well the information fits the purpose.”* — Kurzweil, *The singularity is near*, p. 38.

Information management strategies such as Jonny Decimal [20] and PARA[21] are commonly discussed within the world of personal knowledge management (PKM). As information management and archival solutions, they increase order by defining where and how to store files. Such filing systems help to organise a collection of items and make retrieving those items easier, meaning, they create order through systematic organisation. Here, the measure of order (the chosen file keeping strategy) is valuable, as the purpose of organising individual documents for subsequent retrieval is achieved.

Zettelkasten is also a collection of individual documents, but critically, it is *not* intended for systematic organisation or archival. Approaching Zettelkasten with this mindset risks the Zettelkasten becoming a simple archive of notes, or worse, a graveyard for thoughts[22]. Despite this, Jonny Decimal and PARA are often recommended as file keeping strategies for Zettelkasten during

[20]`https://johnnydecimal.com/`

[21]Forte, *The PARA Method*.

[22]Ahrens, *How to take smart notes*, ch. 4.

discussions about the challenge of file organisation[23],[24]. Questions, discussions and debates on whether to impart a file structure on Zettelkasten are continually renewed, illustrating that the all-notes-in-one-folder idiom of Zettelkasten is a common difficulty.

But research shows that it is a *lack* of systematic organisation that strongly supports Zettelkasten in operating as a tool for thought[25]. The paradox however is that such a lack of file organisation, also increases complexity when it comes to file management. From this complexity, arises certain difficulties in user experience. It is these difficulties that the practical component of this thesis seeks to address.

In Zettelkasten, order is engineered through inter-note linking and the *Folgezettel* principle[26]. This creates structure, link by link, from the bottom up. In other words, the structure *emerges* through the practice of simple rules. By following these inter-note links, the note-taker can follow, create and improve trains of thought. While this is powerful, such a structure cannot be used for query based information retrieval such as *"Where do I find that note?"*, *"What other notes do I have on this topic?"*, *"What notes do I have that are similar to this note?"* or likewise, *"What notes do I have that support or appose this idea?"*. These are typical questions one may pose when working with Zettelkasten, and they are made much more difficult by all notes being stored in a singular root directory.

Given that all notes are stored alongside each other, it is laborious and highly impractical to go about searching for a note by its location in a file system (as there is no directory structure to traverse) or by file name. Keyword and metadata based search will work to a certain extent, however it is important to recognise that over time, any serious and continued use of Zettelkasten will likely result in many thousands of notes. Too many for standard filtering solutions to remain a viable option.

An initial response to this problem is understandably to impart structure on the file system. One folder per category, subject or project is a common instinct and is the fundamental basis of the afore mentioned PARA method. This creates a hierarchy of named directories, and these

[23]*Cataloging, Classification, Information Science, PKMs and YOU! - Knowledge management*.

[24]*Folders*; *How do you guys organize Zettelkasten notes?*; thoresson, *Johnny Decimal – A system to organise projects*; mattgiaro, *What Folder Structure Should You Use in Your Zettelkasten?*

[25]Luhmann, *Communicating with Slip Boxes by Niklas Luhmann*; Cevolini, “Niklas Luhmann’s Card Index: Thinking Tool, Communication Partner, Publication Machine”.

[26]See section 2.2, *The Zettelkasten method*

hierarchies effect the perceived hierarchical relationships of the notes as a result. For example, below in figure 3, the note in `Archive/2024` is likely to adopt the identity of an ”old” note, and will be treated as such. For example, by being classed as an ”old”, and therefore ”outdated” idea. Yet, as discussed, there are no ”old” or ”outdated” notes in Zettelkasten, as all notes are checkpoints in the progression of thought and old ideas can easily become catalysts for future revelations.

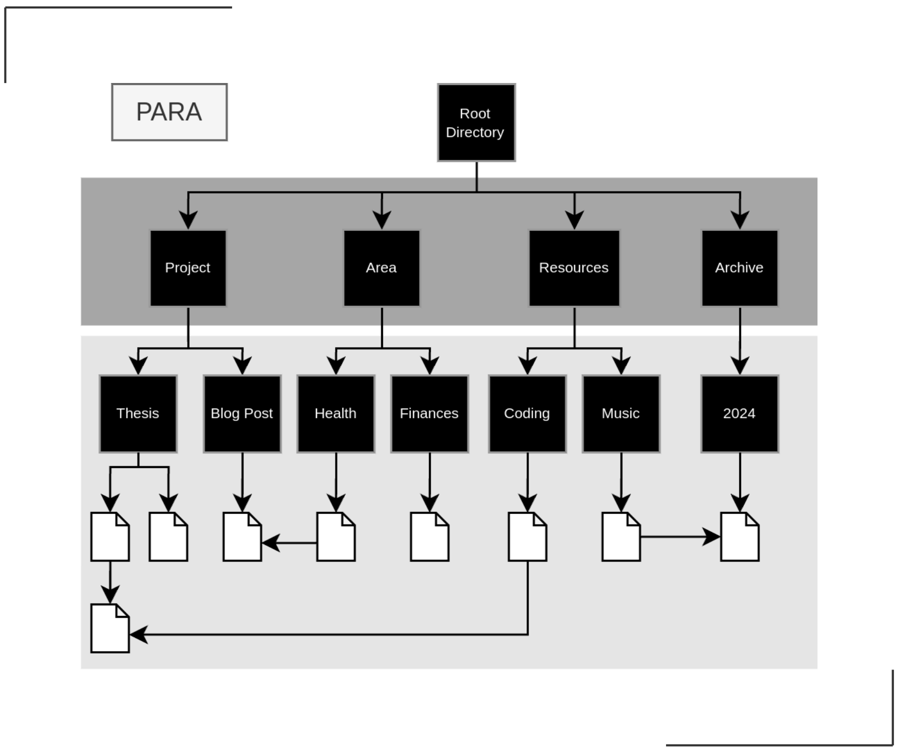


Figure 3: The PARA filing system.

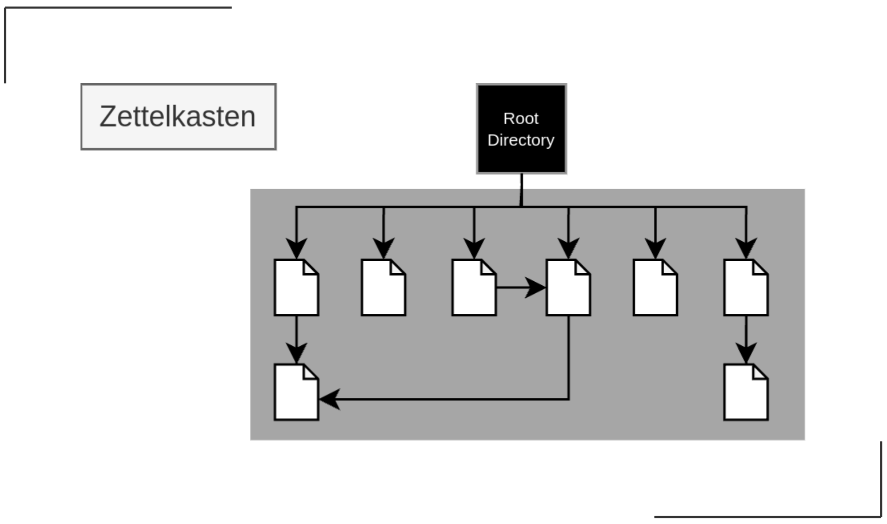


Figure 4: Absence of filing system in Zettelkasten (all files in one directory).

While the mere fact of where a file exists does not inhibit the fundamental process of inter-note linking, sorting notes into categories can create its own workflow problems, as I discuss in the essay, *The GRID System for Frictionless Zettelkasten Organisation*[27]. Organising by category requires that a decision be made every time a new note is created as to where the note should be stored. On occasion, this may be obvious, but often it is not. For example, given a note titled *"Orca, the live coding environment for music composition"*, should it be stored in the "Coding" folder, or the "Music" folder (as above in figure 3)? Perhaps the "Coding" folder is chosen, but this is by no means an obvious choice and therefore may require exhausting deliberation. This can impart significant cognitive friction on the mind, taking time, focus and energy away from the more important task at hand — developing, recording and connecting trains of thought. Additionally, the note taker now has a secondary location to look if their first search is unsuccessful.

Luhmann also encountered this problem:

> *"Die Platzierung größerer thematischer Blöcke wie auch der Stellplatz einzelner Zettel in der Sammlung war aber nicht nur das historische Produkt der Lecktüreinteressen und der Notiztätigkeit Lumanns. Sie war auch eine Folge der Schwierigkeit, eine Fragestellung eindeutig einem und nur einem (Ober-)Thema zuzuordnen. Luhmann löste dieses Problem, indem er es als Chance verstand: Statt der Idee einer systematischen Ordnung zu folgen, votierte er für ein Prinzip des Eintrags, der nur an den vorherigen Eintrag anschließen muss, ohne noch auf eine übergeordnete Struktur zu achten."* — Meschede et al., *Serendipity*, p. 159-160.

This fundamental difficulty is summarised by the conflict between the task of creating meaningful links, and the task of finding notes that merit being linked to. As alluded to before, this difficulty will arise whenever the note taker asks questions such as *"What other notes do I have on topic x"* or *"What notes do I have that are similar to note x"*.

In systems such as the Johnny Decimal System or PARA, the search scope would be greatly narrowed due to the more rigorous methods of file system level organisation imparted by the methods.

[27]Jex, *The GRID System for Frictionless Zettelkasten Organisation.*

This is not the case for Zettelkasten. The quality of the Zettelkasten is completely based on the quality of the links, i.e, the quality of the network. As the Zettelkasten grows in size, the method of rediscovering older notes is increasingly left up to serendipity as the note-taker traverses the connections between Zettelkasten's subnetworks. If a note, or small subnetwork of notes, has very few or no links at all, those notes may never be found again.

> *"Some things will get lost (versickern), some notes we will never see again. On the other hand, there will be preferred centers, formation of lumps and regions with which we will work more often than with others. There will be complexes of ideas that are conceived at large, but which will never be completed; there will be incidental ideas which started as links from secondary passages and which are continuously enriched and expand so that they will tend increasingly to dominate [the] system."*
> — Luhmann, *Communicating with Slip Boxes by Niklas Luhmann*, section II.

While this tendency to "lose" notes, or entire groupings of notes, is intrinsic to the Zettelkasten method it also presents a practical challenge for the note-taker: the difficulty to happen upon ideas that are relevant to the current idea (subnetwork) at hand, but that exist in far off areas of the Zettelkasten.

Chapter 3: Actor–Network Theory

"[...] combinatory play seems to be the essential feature in productive thought."
— Einstein, *Ideas and Opinions*, p. 25.

As outlined in section 2.1, *A Medium for Thought*, thinking tools are important technologies. But how can we understand what a thinking tool is, in terms of how it *works*? Otherwise put, how can a thinking tool be described as a system, whereby a system is *"a set of things – people, cells, molecules, elements – interconnected in such a way that they produce their own behaviour over time"*[1]?

These questions are important in the scope of this thesis. By developing a deeper understanding of how Zettelkasten works as a system, it becomes clearer how to design software systems that work with Zettelkasten.

Actor–network theory (ANT) is a theoretical and methodological approach that seeks to describe how artifacts are produced through the constant and shifting relationships within a network. ANT is not a description of a process to make a certain artifact, rather it describes how individual *actors* may interact (or not), and how such interaction allows artifacts to *emerge* as products of the network. Importantly, ANT does not differentiate between human and non-human actors. Both play equally important roles in the network[2].

[1]Meadows, *Thinking in systems*, p. 17.

[2]Ritzer, *Encyclopedia of social theory*; Latour, *Reassembling the Social*.

3.1 The One-Level Standpoint

"The range of expressive thoughts possible in a [tool for thought] is an emergent property of the elementary objects and actions in that medium."
— Matuschak et al., "How can we develop transformative tools for thought?"

The equal weighting of human and non-human actors helps to apply ANT when analysing Zettelkasten's characteristics as a thinking tool, as it handles the difficult issue of evaluating the relationships between the user, the system and the products of the system. Such an evaluation generally leads to the characterisation of a hierarchical relationship, known as a *two level standpoint*[3] (2-LS). A 2-LS is comprised from the emergent structure itself (macro) and the behaviour between the individual parts that create it (micro). This can be a fitting conclusion given certain scenarios, such as the mega structures built by its inhabitant colony of blind ants[4], or that consciousness is the result of a critical mass of highly interconnected "units"[5]. But in the case of Zettelkasten, if the emergent structure is the Zettelkasten (macro), then what are the individual parts that create it (micro)? It would seem that there is just one human creating the network. But then, how can the Zettelkasten produce ideas that were not directly placed in it by the human themselves, if the human is the only creator of the network?

A one level standpoint (1-LS) describes an actor as being *defined by its network*, while the network is likewise *defined by its constituent actors*. It suggests a cyclical feedback system where there is no "part" or "whole" and answers the question, *"what part is this?"*, with, *"this network"*[6].
This thesis considers Zettelkasten as a one level standpoint (1-LS) system; that the emergent artifact is simultaneously the thought, the Zettelkasten and the user. The question of how the Zettelkasten can produce ideas, is therefore a question formed by looking at the network from a particular angle. For example, the user is defined by the thought, the thought is defined by the network, the network is defined by the user — but, the reverse, as well as any reordering of that sequence, is also true.

[3]Latour et al., "'The whole is always smaller than its parts' – a digital test of Gabriel Tardes' monads".

[4]Latour et al., "'The whole is always smaller than its parts' – a digital test of Gabriel Tardes' monads", p. 597.

[5]Fodor and Pylyshyn, "Connectionism and cognitive architecture: A critical analysis".

[6]Latour et al., "'The whole is always smaller than its parts' – a digital test of Gabriel Tardes' monads", p. 593.

The component not yet described, is how does interaction occur within the network? As it is only through interaction, that the network can evolve. It is this interaction that is of particular importance to this thesis, as by developing additional software functionality for Zettelkasten, the dynamics of interaction within the network will be altered.

3.2 Important ANT Concepts

There are a few ANT concepts in particular that have informed the design of the end software project (outlined in chapter 5, *Software Design*): actor, prime mover, translation and simplification. Together, these concepts describe *how* interaction can be said to occur in Zettelkasten, given that Zettelkasten is, on a material level, one note-taker and a network of data (notes and links).

There are many more terms and components of ANT, but which did not play a significant role in this thesis and are therefore out of scope. The interested reader should refer to *The Sociology of an Actor-Network: The Case of the Electric Vehicle*[7], which explains ANT concepts given a real world historical example of the early efforts to design, build and bring electronic vehicles to market in 1970s France. For a complete and thorough read, Latour's *Reassembling the Social: An Introduction to Actor-Network Theory*[8] is recommended.

Actor

An actor is any individual component (human or non-human) in the network that has the ability to *translate* (see below) information. An actor can be human, or non-human. Actors can operate on their own or together with other actors. Together, actors form the *actor-world* (a synonym for which is 'network').

The validity of non-human objects also being actors within a network, can be further demonstrated in modelling scenarios where those non-human actors fail or fall short in their function[9]. If this happens, the outcome is no less disastrous than if the human actors fail or fall short.

[7]Callon, "The Sociology of an Actor-Network".

[8]Latour, *Reassembling the Social*.

[9]Callon, "The Sociology of an Actor-Network"; Latour, *Reassembling the Social*.

Prime Mover

The prime mover is the actor within the actor-world that has the most influence and control[10].

Translation

Translation is ”transport with deformation”[11]. This is the process of moving information between actors, whereby the information is altered along the way.

For example, sociologists discover how society may function, producing social theories in turn that are used by governments to form policy. That society is in part formed by those policies, but ultimately changes through its constituent actors rebuilding that same society (which may be in response to the new social policies). They do this by introducing new variations, such as technology and ideas, like end-to-end encrypted messaging. The sociologists then need to modify their understanding of that society given the new context[12].

Simplification

Simplification occurs when actors become aligned with each other. This (occasionally) happens in politics in the form of bi-partisan support. Simplification works by reducing the complexity of each actor’s *attributes*, so that multiple actors can align. Attributes are the defining characteristics of an actor.

For example, the attributes of each actor in a political party may include their personal values, domain specific knowledge, the collective interest of their electorate or their own political career goals. In times of unprecedented circumstances such as the Covid-19 pandemic, simplification is highly likely to occur given the existential threat to all (members of both parties and all members of public). Bi-partisan policy responses can be swiftly passed, as the actors in the network have been (momentarily) simplified, and therefore become aligned.

[10]Callon, “The Sociology of an Actor-Network”, p. 22.

[11]Ritzer, *Encyclopedia of social theory*.

[12]Callon, “The Sociology of an Actor-Network”, p. 20.

Intermediary

An intermediary is an actor that passes information without the ability to translate it, such as a computer keyboard or paper.

3.3 Applying ANT to Zettelkasten

Actor

There are no actors within Zettelkasten, except the prime mover.

Prime Mover

The human user themselves. The human user is the one who can add, rework, delete and otherwise modify the data and structure of the network. If they are not active, the state of the Zettelkasten network will remain unchanged.

Translation

Translation occurs in Zettelkasten when the prime mover is actively working with the notes. As they navigate links and read content, they are actively ingesting and manipulating the information in working memory. This is a process of critical thinking[13], and may result in the creation of a new note or the editing of existing notes.

Simplification

Simplification can be seen when similar notes are modified or refactored in light of new information. It also occurs when aspects from multiple notes are gathered to form the basis of a new note.

Intermediary

Examples of intermediaries within Zettelkasten would be a computer keyboard, notes and arguably their links.

[13]I.e, Recall, comprehension, application, analysis, evaluation and synthesis.

Illustrative Example

The above concepts can be illustrated in the following example.

The *prime mover* stumbles (serendipitously) across two notes:

- *Zettelkasten is a laboratory for ideation*
- *Zettelkasten is an interface for thought*

After reading the two notes, the prime mover creates a new note titled, *Zettelkasten is a* ***laboratory*** *for* ***thought***. The prime mover *translates* the information contained in the two prior notes into the new note by merging the attributes of interest, such as ideas, references or attached media. Upon outbound links from the two prior notes to the new note being created, the two prior notes become *simplified*. They become simplified, not because their own content has been rewritten, but their relationship to one another has been established through their shared attributes being used to synthesise a new note, a note to which they are now structurally linked. The synthesised note therefore borrows from its prior notes, as well as extends or adds new ideas, references or media, thus recording a train of thought.

Interaction in Zettelkasten is therefore completely reliant on the prime mover. Yet, their decisions are in part guided by the pathways of the network, defined by links between each note and established by their past selves. Therefore, a note has more influence over the network than an intermediary would typically have, because a note is a partial embodiment of the prime mover. Therefore, ANT can be used to suggest that unknown ideas can emerge from Zettelkasten; as with the creation of each note, the prime mover is unwittingly laying out an idea's constituent components, which may in the future come to be rediscovered and simplified to reveal that, or a different idea — depending on the particular angle the network (idea) is viewed from.

Chapter 4: Machine Learning

Within the field of artificial intelligence (AI), exists the subfield Natural Language Processing (NLP). NLP uses machine learning processes to enable computers to interpret human language. This enables computer programs to take natural language as input and likewise, provide natural language as output. In doing so, natural language becomes an interface for interaction with computer programs. The ubiquitous example of this technology to date are chatbots that appear as ”intelligent assistants” on websites, as well as more advanced implementations like OpenAI’s ChatGPT.

The basis of these technologies are called *models*, which are trained to comprehend by ingesting vast amounts of data. In the case of NLP models, this data is text data. The state-of-the-art models are made possible by the innovation of the transformer architecture, which catalysed the field of machine learning by innovating the process of *self-attention*[1]. This initiated drastic advancements in training efficiency and model capability, making the existence of large language models (LLM) like ChatGPT possible, as well as SBERT, the LLM used in the practical component of this thesis.

The succession of models that lead to SBERT[2], are pre-trained transformer models and are not generative. This means they do not create new data, such as text or imagery. Their main function is to encode input data into vector representations (points in space), referred to as *embeddings*. These embeddings represent the relationships between each data point. Our natural way to visualise these points in space, is in three dimensions. However, in the case of model embeddings, vector space is multidimensional. This is because there are (many) more than three parameters by which the relationships between data points are evaluated. With these relationships at hand, certain downstream tasks are made accessible, such as similarity learning and topic modelling.

[1]Vaswani et al., *Attention Is All You Need*.

[2]Le and Mikolov, *Distributed Representations of Sentences and Documents*; Devlin et al., *BERT*; Reimers and Gurevych, *Sentence-BERT*.

4.1 Semantic Textual Similarity

Similarity Learning (SL) is the umbrella term for machine learning processes that compare data points in terms of how similar they are to each other. It powers solutions such as facial recognition, product recommendation, signature validation, anomaly detection and so forth.

The task of applying SL to items of text has been attributed its own title, Semantic Textual Similarity (STS). In STS, the goal is to build models that are capable of understanding what words and larger structures like sentences *mean*[3].

Prior to the deep learning era (circa 2003), the task of STS relied on classical machine learning techniques[4], which implement statistical and mathematical models[5] and require heavy feature engineering, making it a very time-consuming process[6]. One major disadvantage of this approach is that semantics are in general difficult to define, which makes the task of feature engineering incredibly difficult[7]. A classic example of this are puns.

> *"Why did the tomato turn red? Because it saw the salad dressing!"* — Raskin, *Semantic Mechanisms of Humor*

As humans, we understand that this is a (terrible) joke. We recognise how the double meaning of *dressing* transforms the entire semantic relationships within the question and answer structure as a whole. With classical machine learning techniques, the semantics of the above pun would likely remain closely associated with salad, condiments, recipes, etc. The reality that it is a joke, would be missed.

This challenge has been significantly addressed by advancements in artificial neural networks, which do not necessarily require labeled data and supervised learning processes. The state-of-the-

[3]Otter, Medina, and Kalita, "A Survey of the Usages of Deep Learning for Natural Language Processing"; Liddy, "Natural Language Processing".

[4]Jones, "Natural Language Processing: A Historical Review".

[5]Lecun et al., "Gradient-based learning applied to document recognition"; Brown, Lai, and Mercer, "Aligning sentences in parallel corpora"; Chitrao and Grishman, "Statistical Parsing of Messages".

[6]Lecun et al., "Gradient-based learning applied to document recognition"; Brown, Lai, and Mercer, "Aligning sentences in parallel corpora"; Chitrao and Grishman, "Statistical Parsing of Messages"; Uni Wolverhampton et al., "Semantic Textual Similarity with Siamese Neural Networks".

[7]Otter, Medina, and Kalita, "A Survey of the Usages of Deep Learning for Natural Language Processing".

art approach in STS is to use Siamese Neural Networks[8], which employs two identical artificial neural networks that process a pair of inputs in tandem. After processing the two inputs, their outputs are compared, which results a similarity score. The similarity score is therefore taken as the semantic similarity between the two items of text[9]. Put otherwise, how similar is the *meaning* between the given pieces of text?

4.2 Topic Modelling

Topic modelling (TM) is the process of discovering the generalised topics that exist in a collection of text data. For example, given a book on house pets, it would be expected that a TM model would return topics such as *dogs, cats, fish and hamsters*, but perhaps also *nutrition, pet care, common diseases* and so on.

With TM, it becomes possible to reveal a structure of topical relationships within a collection of documents that would otherwise require significant time and effort to evaluate oneself. Importantly, documents may be identified as belonging to *multiple* topics. This creates a *one-to-many*, structure, rather than a *one-to-one*. Whereby a *one-to-one* relationship would allow each note to belong to one category only. As we will see, this is a valuable aspect when considering the usefulness of TM within the context of Zettelkasten.

Latent Dirichlet Allocation (LDA)[10] was for long the primary solution for TM. However, due to being based on the Bag of Words statistical model, only the *frequency* of words in a document are considered, and not the *order* in which they occur. Because of this, the semantic meaning of a document is lost, leading to inaccuracies when computing the topics a document might belong to.

This challenge has been largely overcome by embedding-based models resulting from the transformer architecture[11], as similarly discussed at the start of this chapter.

[8]Otter, Medina, and Kalita, "A Survey of the Usages of Deep Learning for Natural Language Processing"; Cheng et al., "A Neural Topic Modeling Study Integrating SBERT and Data Augmentation".

[9]Chicco, "Siamese Neural Networks: An Overview".

[10]Blei, Ng, and Jordan, "Latent Dirichlet Allocation".

[11]Cheng et al., "A Neural Topic Modeling Study Integrating SBERT and Data Augmentation"; Krishnan, *Exploring the Power of Topic Modeling Techniques in Analyzing Customer Reviews*; Angelov and Inkpen, "Topic Modeling: Contextual Token Embeddings Are All You Need".

4.3 Machine Learning as a Solution

The specific abilities of machine learning to identify patterns and predict outcomes is driving innovation and process optimisations in wide range of industries and problem domains such as agriculture[12], marketing[13], health care[14] and supply chain logistics[15] to name but a few. But is machine learning well suited as a solution for managing Zettelkasten's inherent difficulties as discussed in section 2.3, *The Difficulties of Bottom-Up Structure*?

Machine learning excels at tasks where the relationships and patterns in data are hidden, such as finding structure in unstructured data. These capabilities allow, for example, the evaluation of semantics in language, as discussed in this chapter. This is in line with the underlying foundations of Zettelkasten, where the goal is to discover relationships between qualifying notes and where such relationships may otherwise never be found due to the lack of organisational structure of a Zettelkasten repository.

As also discussed in section 2.3, *The Difficulties of Bottom-Up Structure*, while Zettelkasten does not enforce filing or organisation conventions, the idiomatic practice is to use no filing or organisational structure at all. All notes exist in the same directory. Instead, structure exists through the explicit relationships between the notes, which is an emergent property born from the practice of linking[16]. Therefore, the ability of machine learning techniques to work with data *despite* there being no order or structure to the filing system itself, is a significant benefit. This allows for the implementation of a technical process, without imparting change on the Zettelkasten itself.

This is a process that Luhmann himself implemented, albeit manually, iteratively and with pen and paper. As mentioned in section 2.2, *The Zettelkasten method*, Luhmann created a keyword index, which itemised notes that he identified as being good representatives for their topic matter. These notes effectively became ”jumping in” points for when Luhmann needed to navigate elsewhere in his Zettelkasten. This was Luhmann's way of managing the complexity of bottom-up structure,

[12]Liakos et al., “Machine Learning in Agriculture: A Review”.

[13]Kaličanin and Čolović, “Benefits of Artificial Intelligence and Machine Learning in Marketing”.

[14]Bhandari, “Revolutionizing Radiology With Artificial Intelligence.”

[15]Pasupuleti et al., “Enhancing Supply Chain Agility and Sustainability through Machine Learning: Optimization Techniques for Logistics and Inventory Management”.

[16]See: chapter 3, *Actor-Network Theory*.

and therefore indirectly supports the idea to include the same core concept in the software features of this thesis.

Based on these aspects, machine learning is an advantageous and fitting technology when considering the specific requirements of the task at hand. Which leaves the following question: how can machine learning enabled features be *effectively* integrated into the Zettelkasten process? This is discussed in chapter 5, *Software Design*.

4.3.1 Existing Solutions

There are numerous implementations of machine learning technology in Zettelkasten. These are often directly integrated into the workspace of Zettelkasten (i.e, the Zettelkasten program itself), but can also be indirectly integrated such as using Google's NotebookLM to create a "conversational element" to the Zettelkasten[17]. Here, it is worth drawing attention to Luhmann's *"Communicating with Slip-Boxes"*[18]. Both Wanderloots and Luhmann are interacting with the Zettelkasten in the same form, through discussion. But for Wanderloots, the Zettelkasten is assimilated into the AI system which provides a natural language interface through which to converse. While for Luhmann, the "conversation" was in essence a process of deep critical thought where his current self and ideas were confronted by his past self and ideas contained in each pre-existing note of the Zettelkasten. Exploring that comparison properly is outside the scope of this section, but it's still relevant to mention, as it acutely illustrates the effect on the Zettelkasten process that machine learning technology can have, depending on how it is approached and implemented.

The selected examples in this subsection focus on existing implementations of machine learning enabled features that are similar in their functionality to the software of this thesis. Specifically, this means implementing similarity learning or topic modelling, and working directly with the notes themselves including links, content and the organisational structure.

Quotes have been extracted from the marketing communication on the product pages of each example. This illustrates how the products are intended to be useful and provides an insight into how strong of a role AI plays.

[17]Wanderloots, *How I Use NotebookLM With Obsidian Practical Note-Taking + AI* .

[18]Luhmann, *Communicating with Slip Boxes by Niklas Luhmann*.

- **Smart Connections**[19]: Suggests and creates links between related notes.

  ”Save time linking, tagging, and organizing! Smart Connections finds relevant notes so you don’t have to!”

- **Mem AI**[20]: Summarisation, content generation and organising notes.

  ”Mem Chat can dig up needle-in-a-haystack answers, summarize across meeting notes, and write content—using a deep understanding of your notes.”

  ”For those seeking a bit of structure, AI powered Collections simplify organization—without needing to remember your tagging system or complex folder structures.”

  ”Related Notes shows you everything related to your current work across your notes, without the need for manual organization.”

- **Napkin**[21]: Topic suggestion.

  ”Your personal flow of ideas is the easiest way to feel grounded and inspired. For more focused reflections, your personal AI understands your ideas deeply, and suggests topics you’re interested in.”

- **Notion**[22]: Generate and modify content, and suggest links.

  ”Leverage the capabilities of Notion AI to think bigger, work faster, and augment your creativity. Learn how to use Notion AI to transform text, automate simple tasks, and generate new content inside your connected workspace.”

[19]`https://github.com/brianpetro/obsidian-smart-connections`

[20]`https://mem.ai/`

[21]https://www.zettelgarden.com/

[22]`https://www.notion.so/product/ai`

- **Zettelgarden**[23]: Suggest related notes and generate topics (called ”entities”).

  ”While other tools rush to automate everything with LLMs, Zettelgarden takes a measured approach. AI features are designed to augment your thinking process, not replace it. Find connections and patterns while maintaining the critical human element of knowledge management.”

- **Smart Second Brain**[24]: LLM chat based integration.

  ”[Smart Second Brain] serves as your personal assistant, powered by large language models like ChatGPT or Llama2.”

- **Obsidian Copilot**[25]: Autocomplete notes with saved prompts (e.g, ”make this shorter” or ”explain this like I'm five”), find similar notes.

  ”Leverage intelligent agent[s] equipped with tools to truly understand your Obsidian vault.”

These examples show that there is an allure of designing machine learning technologies to take care of major tasks such as writing and linking notes ”so [we] don't have to”. Likewise, machine learning technology is here considered so magnificent, that it can provide us with the capability to ”truly understand” our own Zettelkasten, suggesting that we did not naturally posses such capabilities in the first place.

Of course, marketing communication is designed with the singular purpose of convincing us we should use the tool. But what this shows is how, depending on its design, digital technology is being developed that can diminish our own cognitive capabilities by means of removing our opportunity to execute those cognitive tasks[26], and that this in some instances is even a marketable quality.

[23]`https://www.zettelgarden.com/`

[24]`https://github.com/your-papa/obsidian-Smart2Brain`

[25]`https://www.obsidiancopilot.com/en`

[26]Sparrow, Liu, and Wegner, “Google Effects on Memory: Cognitive Consequences of Having Information at Our Fingertips”; Carr, *The shallows*.

Chapter 5: Software Design

"AI should challenge, not obey." — Sarkar, "AI Should Challenge, Not Obey"

In chapter 2, *Zettelkasten*, it was described how the Zettelkasten method relies on a small set of simple rules that allow for the emergence of networked relationships between notes. It further evidenced that the lack of a filing system in Zettelkasten paradoxically fosters this behaviour, while also creating a practical challenge for the note-taker: the difficulty to rediscover pre-existing notes that are relevant to the current note at hand, but that exist in far off areas of the Zettelkasten. Put otherwise, the paradox is that in order for the Zettelkasten to produce ideas effectively, it must be structured in a way that allows for notes to be lost and for ideas to be forgotten.

This presents the crux of the design challenge for this practical component of the thesis: How to strike the balance between technological convenience and cognitive effort, while leaving the paradigms of Zettelkasten as intact as possible?

To leave the paradigms of Zetteasten intact, the software should work in conjunction with the non-hierarchical filing system of Zettelkasten, while simultaneously enabling functionality for the user that would otherwise only be practical if the filing system was organised in a hierarchical manner (e.g, nested and topical directories). In doing so, the user has an option to momentarily inspect the Zettelkasten in a structure other than that of networked notes. This offers a way for the user to manage difficulties arising from Zettelkasten's bottom-up structure.

5.1 Integration of Theory

The design of machine learning enabled software in this thesis, follows the ethos that in order for AI to be effectively integrated into a medium for thought such as Zettelkasten, it should challenge, not obey. This significantly informs the program design and counters the trend of existing solutions shown in subsection 4.3.1, *Existing Solutions*

This, as referenced earlier, is exemplified in Wandloots' usage of Google's NotebookLM to create a "conversational element" to the Zettelkasten[1] when compared with Luhmann's self-reflective narrative in *"Communicating with Slip-Boxes"*[2]. Both Wanderloots and Luhmann are interacting with the Zettelkasten in the same form, through discussion.

But Luhmann could not simply type into a text box, *"What relationships are there between Zettelkasten and discourse?"* and immediately have a formulated textual response returned to him. Instead, he had to locate a particular note as a starting point and traverse the Zettelkasten's network of notes with the question in mind, having it supported, challenged and possibly modified along the way (i.e. *translation*, see section 3.3, *Applying ANT to Zettelkasten*).

The two examples are on the extreme ends of the spectrum: AI generated responses without having to interact with a note at all, and a cabinet with either 24,000 or 66,000 physical slips of paper (Luhmann had two Zettelkästen). NotebookLM is a perfect example of AI "obeying", while a physical Zettelkasten is extremely challenging.

In contrast, the design implemented here is that the capabilities of ML models are made available through a limited set of command line prompts. The commands and the underlying functionality are designed specifically to assist in answering the questions posed in section 2.3, *The Difficulties of Bottom-Up Structure* (outlined again below), and informed by a public opinion poll described in the following section.

The idea of the user being "challenged" helps to maintain the aforementioned balance between technological convenience and cognitive effort. To achieve this, the software should *reveal possibilities* to the user, but never *make decisions* for them (such as generating notes, or creating links between notes). A user's "second brain" is going to be much more effective when the information it works with came from their "primary brain", otherwise the symbiosis between the two will be weaker.

This design decision was inspired by the investigation into tools/mediums for thought and Actor-Network Theory. In order for Zettelkasten to remain qualified as a medium for thought, it is essential that the user maintains their role as the prime mover[3] of the Zettelkasten. It could

[1]Wanderloots, *How I Use NotebookLM With Obsidian   Practical Note-Taking + AI* .

[2]Luhmann, *Communicating with Slip Boxes by Niklas Luhmann.*

[3]See chapter 3, *Actor-Network Theory*

hardly be classed as a medium for thought, if the content and links that define the Zettelkasten as a network, and therefore define its value, were to be partly or wholly created by a tool of the medium itself, namely a generative LLM. This is not an isolated idea:

> *”Critical thinking is by itself passive. It’s the embedding of critical thinking within thinking tools that elevates the technology from a passive cognitive crutch, into an active facilitator of thought. [...] We thus need to broaden the notion of AI as assistant, toward AI as provocateur. From tools for efficiency, toward tools for thought.”* — Sarkar, “AI Should Challenge, Not Obey”

Also in chapter 3, *Actor-Network Theory*, the concept of ”simplification” was introduced. It was shown how new ideas emerge in part from the convergence of attributes from pre-existing ideas (see section 3.3,). Of note, this theoretical concept directly inspired the inclusion of the clustering feature. Clustering the notes by similarity is in essence a process of simplification; the notes that have the strongest shared attributes, are the most similar, and as such, are clustered together. Here Actor-Network Theory was valuable, as applying its concepts to Zettelkasten allowed for the assertion that the feature of similarity clustering would be a useful addition to the software feature set, as the theory demonstrates how important simplification is for the development of a network.

In summary, the theoretical analysis of Zettelkasten and Actor-Network Theory as well as the framing of Zettelkasten as a medium for thought helped to inform the design of the software, so that:

1. The source of the difficulties could be accurately identified,
2. the user maintained their role as the prime mover,
3. the necessity of the user’s critical thinking skills is maintained, and
4. the file keeping strategy of Zettelkasten was left untouched.

5.2 Public Poll

The inclusion of both similarity learning and topic modelling as features was assisted by the creation of a public online poll on the zettelkasten.de forum[4]. The poll asked which of the two features would be preferred. There was no option to pick both, and no option for ”neither”. This resulted in only 11 votes, where 63% preferred similarity learning and 36% preferred topic modelling. However, some respondants also commented with their decision-making rationale, offering useful qualitative insight.

> *”Based on experience I would say similarity learning is more important than topic modelling for me, but I’ve never used true topic modelling, so I don’t know empirically how useful it is.”* — Andy

> *”Topic Modelling with ML would help surface nuanced topics with my Zk that are underdeveloped. Some of these would be worth investigating more fully and some not, but seeing them would indeed be helpful in guiding future zettelkasting.”* — Will

> *”I don’t have a strong feeling either way; both would be useful. But I lean a bit towards [similarity learning], so have voted for that one.*
> *My reasoning is that between plentiful tagging and structure notes, I have a pretty good idea of my topics (but not a perfect or comprehensive idea). On the other hand, when writing a zettel, I’d love to have some indication of what other zettels in my database are similar or dissimilar or even express an opposing idea, as I could then explore them for links.”* — GeoEng51

> *”If limited to topic modeling and similarity learning, I would think topic modeling would be more helpful in a generic way. The reason being, if you can let the topic model give me document relations based on a topic (which in and of itself can be think of a measure of similarity). That substantially cuts down the number of documents, so then evaluating document similarity among them would be very manageable, even when done manually.”* — zettelsan

[4]Jex, *Poll*.

"I picked similarity learning because I suspect it would be more "random" (i.e. there might by more synchronicity?) but I'd like both!" — mlbrandt

The last comment by mlbrandt is worthy of comment here, as it reinforces the quality that serendipity can have within the Zettelkasten process, and that users are indeed in search of it.

5.3 Feature Engineering

The set of features was engineered through understanding what the desired result would be for each of the target questions. The answer to each question takes the form of executing a command and getting a response. This is a common method of defining features within agile software devleopment, and is known a "user story"[5]. The questions themselves were derived from the Zettelkasten user's responses above, and specifically in the case of the clustering feature, inspired by Actor-Network Theory's concepts of "shared attributes" and "simplification".

Question and answer pairings:

- **Q**: *"What notes do I have that are similar to note x?"*

  A: Compare the similarity of the current note against all other notes in the Zettelkasten.

- **Q**: *"What clusters of similar notes exist within my Zettelkasten?"*

  A: Cluster all notes in the Zettelkasten by similarity.

- **Q**: *"Do I have any notes on topic x?"*

  A1: Reveal the topics contained within the Zettelkasten.

  A2: Search for notes on a given topic.

- **Q**: *"What other notes do I have on topic x?"*

  A: Return the notes from the same topic as the given note.

[5]Atlassian, *User Stories | Examples and Template*.

5.4 Model Choice

5.4.1 Semantic Textual Similarity

SBERT[6] was selected as the foundational model for computing semantic similarity between Zettelkasten notes. SBERT is an extension of BERT[7], optimised through a siamese network architecture to generate sentence embeddings that can be compared using cosine similarity. SBERT makes for a particularly good choice for the task of this thesis as SBERT's optimisations allow for the deriving of semantics for very small textual units, such as Zettelkasten notes.

Although more recent optimisations of SBERT, like SBERT-WK[8], show performance gains, the original SBERT was selected for practical reasons. Namely, the ease of implementation due to SBERT's availability as a well-documented Python package.

5.4.2 Topic Modelling

For topic modelling, BERTopic[9] is used. BERTopic integrates SBERT for generating embeddings and comes packaged with clustering algorithms which help to form coherent topic representations. This model was chosen over traditional models such as LDA and LSA because it leverages sentence-level semantics through SBERT, is established and well documented and has a loosely coupled modular architecture, which allows for experimentation and ease of future work innovations.

5.5 Pre-Processing Design

Most natural language models benefit from text cleaning, such as the removal of stop words and punctuation as well as lower casing all words. However, SBERT is trained on datasets (e.g., SNLI) that preserve capitalisation, punctuation, and stop words. Capitalisation is also important for semantics, as in the case of nouns. Hence, conventional cleaning may degrade performance

[6]Reimers and Gurevych, *Sentence-BERT*.

[7]Devlin et al., *BERT*.

[8]Wang and Kuo, *SBERT-WK*.

[9]Grootendorst, "BERTopic: Neural topic modeling with a class-based TF-IDF procedure".

and has previously been found to have no effect at all[10].

However, Zettelkasten notes are commonly written in Markdown, a lightweight markup language for adding formatting elements to plain text documents[11]. These formatting and control symbols (such as `>`, `#` and `*`) are not present in the datasets used for training. It is therefore prudent to test for the impact of this syntax on semantic similarity between notes. This will be achieved by regex-based cleaning and will allow for comparative testing, described in section 6.3, *Pre-processing Pipeline*.

5.6 Test Methodology

To evaluate the effect of manual cleaning on semantic similarity computation, the following methodology will be implemented.

5.6.1 Test Note

A representative Zettelkasten note titled *Zettelkasten is an interface for thought* will be used as the test note (below in figure 5). It contains Markdown syntax, outgoing links, and is concise. Thus, it is an example of a typical Zettelkasten note and will be compared against the entire Zettelkasten, which is the author's own Zettelkasten of 516 notes, using both pre-processing modes. This will allow the effect of pre-processing to be evaluated.

Markdown specific syntax includes:

- blockquotes (`>`)
- titles (`#`)
- link syntax for outgoing links (`[link text](path/to/file.md)`)
- italic or bold styling (`_`)
- italic or bold styling (`*`)

[10]Qiao et al., *Understanding the Behaviors of BERT in Ranking*.

[11]`https://www.markdownguide.org/getting-started/`

```
# Zettelkasten is an interface for thought

> Ideally, an interface will surface the deepest principles underlying a
> subject, revealing a new world to the user. When you learn such an interface,
> you internalize those principles, giving you more powerful ways of reasoning
> about that world.
> [Using Artificial Intelligence to Augment Human Intelligence](r/yepr)

Zettelkasten is an interface that can surface the principles of an emergent
idea. It allows the user to navigate a web of interlocking elements, whereby
they can be stitched together.

As the notes are written in the voice of the user, the user is able to reveal an
_inner world_ formed from individual elements that exist within their own right,
but were not yet attributed to one another. They become attributed to one
another through the process of **linking**.

This [creates a context in which the user can think](0wcx). The context is the
new world.

---

Prev: [Using Artificial Intelligence to Augment Human Intelligence](r/yepr)

- [How can we develop transformative tools for thought?](r/shw8)
- [Zettelkasten promotes ideation due to inter-note linking](mv04)
```

Figure 5: Contents of chosen test note.

5.6.2 Similarity Evaluation

Two sets of similarity tensors will be computed; one for the control case of no pre-processing, and one for the test case with Markdown stripping.

To qualitatively evaluate the effect of pre-processing, similarity scores from both tensors will be grouped using the Likert five-point scale[12]: *least similar*, *somewhat similar*, *moderately similar*, *very similar*, and *most similar*. A JSON file representing these segment groupings will be generated for each mode and compared using the online tool, Diff Checker[13], to identify changes in the members of similarity rankings. JSON will be used as it is human-readable but also widely supported by programming languages and computer programs. This simplifies the integration of this study into future work.

To quantifiably evaluate the effect of pre-processing, the difference in individual similarity scores across the corpus will be visualised as a heatmap using PyTorch and Matplotlib. This process additionally computes the max and mean differences in similarity.

[12]https://en.wikipedia.org/wiki/Likert_scale

[13]https://www.diffchecker.com/text-compare

It is expected that the pre-processing case will yield a higher variation in similarity scores across the corpus.

5.6.3 Evaluation Criteria

Effectiveness of manual cleaning will be judged based on:

- Whether irrelevant notes (e.g., those deemed similar only in Markdown syntax) are demoted in similarity ranking.
- Whether the revised rankings result in more thematically and semantically appropriate groupings.
- The visual heatmap difference and statistical summaries (mean and max change).

5.7 User Interface and Functionality

The program will be delivered as a command-line tool, whereby the user interacts via their computer's terminal. It will support Linux and macOS systems.

The available commands will be designed so they present as ”answers” to the questions prefaced in this chapter.

For example:

- Question: *”What notes do I have that are similar to this note?”*
- Answer: `program --notes-similar-to <note-title>`

Chapter 6: Design Implementation

6.1 Programming Environment

The program was developed, run and tested with the following environment:

- Computer: Apple M1 Max 10 core, 64Gb RAM.
- OS: GNU/Linux Fedora 41
- Programming language and version: Python 3.12

The additional packages and versions can be found in the `requirements.txt` file at the root of the provided source code repository.

6.2 Implemented Models

SBERT for Semantic Textual Similarity

The SBERT model was implemented as described. It was found that SBERT has a default word limit of 128 words per document, but that it can be increased to 510[1].

This change did not alter similarity score, but was retained for suitability as the average word count of all notes in the corpus was 239.

BERTopic for Topic Modelling

BERTopic was successfully implemented using the SBERT embeddings. This conformed with the design choice to use BERTopic over legacy models such as LDA or LSA, due to its superior performance for small text documents and ease of integration.

[1]https://github.com/UKPLab/sentence-transformers/issues/364

6.3 Pre-processing Pipeline

As per the design chapter's discussion on pre-processing, regex stripping was implemented in order to understand the effect that Markdown sytax had on similarity scoring.

Below is the regex cleaning logic to strip Markdown sytax from the note bodies. It was found that the program used to output the note data for model injestion, included new line characters (`\n`). Removing new line characters was therefore included in the pre-processing logic.

```
md_link_patt = r"\[(.*?)\]\(.*?\)"
md_symbols_patt = r"(#)|(>)|(\*)|(_)"

for note in dirty_notes:
    note = re.sub(r'\n', ' ', note)
    note = re.sub(md_link_patt, r"\1", note)
    note = re.sub(md_symbols_patt, "", note)
    cleaned_notes.append(note)
```

Figure 6: Code for manual markdown cleaning in `corpus.py`.

The substitutions were executed on the body of each note, which is passed as a contiguous string element to SBERT. As such, SBERT considers the note in its entirety when deriving a semantic representation, instead of on an individual sentence basis.

The selected test note *Zettelkasten is an interface for thought* was used as a reference for similarity comparisons against the Zettelkasten corpus. Below is the output of the pre-processing pipeline when applied to the test note (see subsection 5.6.1, *Test Note*).

```
[Zettelkasten is an interface for thought   Ideally, an interface will surface
the deepest principles underlying a  subject, revealing a new world to the
user. When you learn such an interface,  you internalize those principles,
giving you more powerful ways of reasoning  about that world.  Using Artificial
Intelligence to Augment Human Intelligence  Zettelkasten is an interface that
can surface the principles of an emergent idea. It allows the user to navigate
a web of interlocking elements, whereby they can stitch them together.  As the
notes are written in the voice of the user, the user is able to reveal an inner
world formed from individual elements that exist within their own right, but
were not yet attributed to one another.  This creates a context in which the
user can think. The context is the new world.  ---  Prev: Using Artificial
Intelligence to Augment Human Intelligence  - How can we develop transformative
tools for thought?]
```

Figure 7: Result of manual text cleaning on the test note. Each note is cleaned and appended to the `cleaned_notes` array.

6.4 Testing and Evaluation

6.4.1 Model Integrity

To ensure that SBERT was not behaving strangely, a test corpus was generated with one hundred identical notes[2]. Similarity embeddings were generated from this corpus, and the similarity results were inspected. All notes were 100% similar to all other notes. With this confirmed, evaluating the results of similarity learning could be interpreted with confidence, knowing that SBERT itself was performing as expected.

6.4.2 Similarity Grouping

The similarity results were grouped into five segments: *least similar, somewhat similar, moderately similar, very similar, and most similar*. These divisions mimic the Likert scale, as originally proposed in the design chapter.

```
[
  {
    "title": "title",
    "path": "path",
    "least_similar": [
      { "title1": "title1", "path": "path1" },
      { "title2": "title2", "path": "path2" }
    ],
    "somewhat_similar": [
      { "title1": "title1", "path": "path1" },
      { "title2": "title2", "path": "path2" }
    ],
    [ ... etc]
  }
]
```

Figure 8: Sudo JSON structure illustrating how the similarity groupings appear in the `data/simdiss.json` file after being generated.

Two strategies were implemented, users can select the strategy via the command-line flag, `--strategy <std OR even>`:

- **Even:** Equal interval divisions of similarity scores.
- **Standard Deviation:** Normal distribution-based grouping for improved granularity in large corpora.

[2]The bash script used to generate the dummy notes can be found in `scripts/gen_test_data.sh` of the code repository.

This approach provides an intuitive, familiar scale for users to navigate related content within their Zettelkasten. The standard deviation mode is particularly useful, as when collections grow in size, it should be useful to constrain the range of values that qualify as ”most similar” and ”least similar”. This narrows the search scope for the user in those bands.

Qualitative Comparison of Similarity Results

The outputs from the base and cleaned corpus were compared using the website Diff Checker, and is viewable online[3]. The difference between the base case and cleaned corpus was evaluated by inspecting what changes occurred, as revealed by the diff. For example, if a note about Zettelkasten was placed in the *least similar* grouping, it would be opened to understand why.

During the implementation, it was realised that the chosen test note, *Zettelkasten is an interface for thought*, does not serve as the best choice to qualitatively evaluate the similarity of notes in terms of *content*. This is because the Zettelkasten collection being used has many notes on the topic of Zettelkasten and related ideas (particularly as the notes for this thesis were written using the very same Zettelkasten). It is however still a good choice to measure the effect of the Markdown targeted text cleaning.

The considered logic to account for this, was to determine which notes were the *least similar* in content throughout the Zettelkasten collection. By picking a less representative note, SBERT’s ability to identify more subtle semantic relationships would be easier to qualitatively identify. This is because, as good as SBERT is, humans are still better at qualitatively deciding whether there are any *meaningful* relationships between two documents (instead of just sharing keywords for example). So by providing SBERT with the *least similar* note in the collection, its capabilities in finding similar notes would be put to a harder test.

Deriving the least similar note in the entire collection involved working directly with the similarity tensor and was a three-step process. The diagonal was removed (which represents each note’s similarity score against itself) (see figure 13), the tensor was flattened and the minimum score returned as shown on the following page.

[3]Diff of cleaned corpus against base: `https://www.diffchecker.com/nM7BDgzV/`

```
def least_similar_note(tensor):

    # Remove self-similarities (diagonal entries)
    t_no_diag = tensor.clone()
    t_no_diag.fill_diagonal_(0)

    total_similarity = t_no_diag.sum(dim=1)

    index = torch.argmin(total_similarity)
    title = corpus.titles[index]
    print(title)
```

Figure 9: Function to extract the least similar note from the entire Zettelkasten.

This returned the note titled, *vampire bat in denim*[4]. Perhaps, even to the reader this sounds like an acceptable result. And by all intents and purposes, it is. It is a note documenting a nightmare the author had, and spawned no further connected notes or ideas.

When considering the overarching goal of the software being developed, that it should help the user to manage the difficulty of finding connections to notes that are isolated from the rest of the network (by either being topically abstract or having no pre-existing incoming or outgoing links), testing the efficacy of the developed solution against the most isolated note in the test Zettelkasten is a worthy exercise.

The results of the above qualitative tests are presented in chapter 7, *Results and Discussion*.

6.4.3 Quantitative Comparison via Heatmap

The similarity tensors from both base and cleaned runs were compared using PyTorch and Matplotlib (figure 10). A difference heatmap was generated, quantifying the change in semantic scores caused by Markdown removal, where the range of ”Max” is 0 to 1.

[4]See Appendix A, *Vampire Bat in Denim Note*.

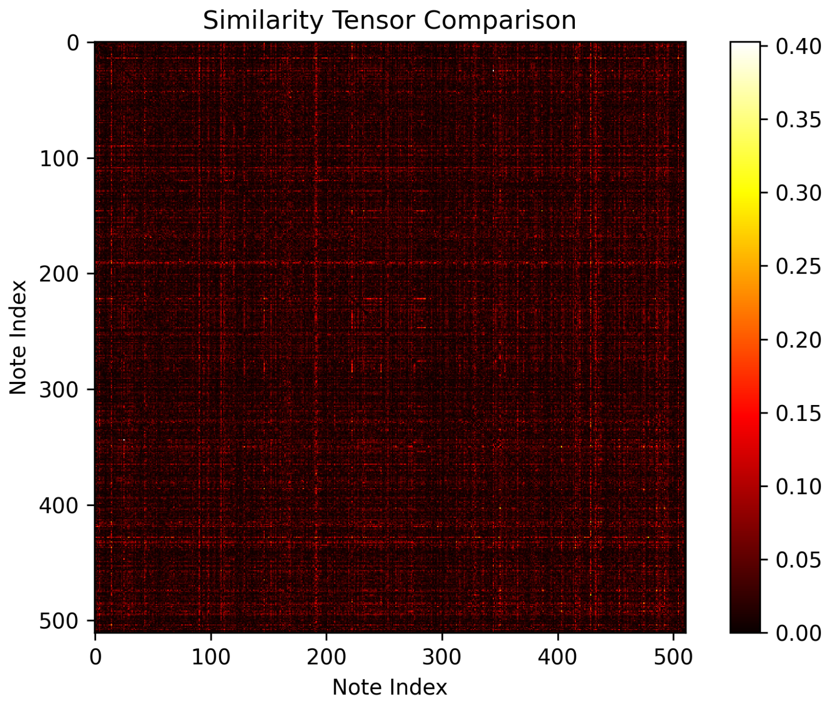


Figure 10: Difference between ”base” and ”manual cleaning” similarity tensors.

Mean: 0.023315. Max: 0.402765

6.5 Implemented Features

In addition to the intended feature set (see section 5.3, *Feature Engineering*), an additional feature to search for topics similar to a query topic was implemented.

This is a built-in feature of BERTopic, and was included in the final program as it fits the design goals of the thesis well. It effectively merges the two processes of similarity learning and topic modelling. This is because the feature allows the user to search for *topics* themselves, and instead of returning an exact match, it returns the top three most similar topics.

```
$ python main.py tm --search "creativity"
Topic 0 (46.0% similar to 'creativity')
Topic 2 (70.0% similar to 'creativity')
Topic 4 (46.0% similar to 'creativity')
```

Figure 11: Example output of the topic modelling search functionality.

6.6 Program Architecture

The program is made available via the command line interface (CLI). It provides two high level groupings of commands, one for each functionality addressed in this thesis; similarity learning `sl` and topic modelling `tm`.

Both positional commands have multiple subcommands, as described in chapter 7, *Results and Discussion*.

Program Flow

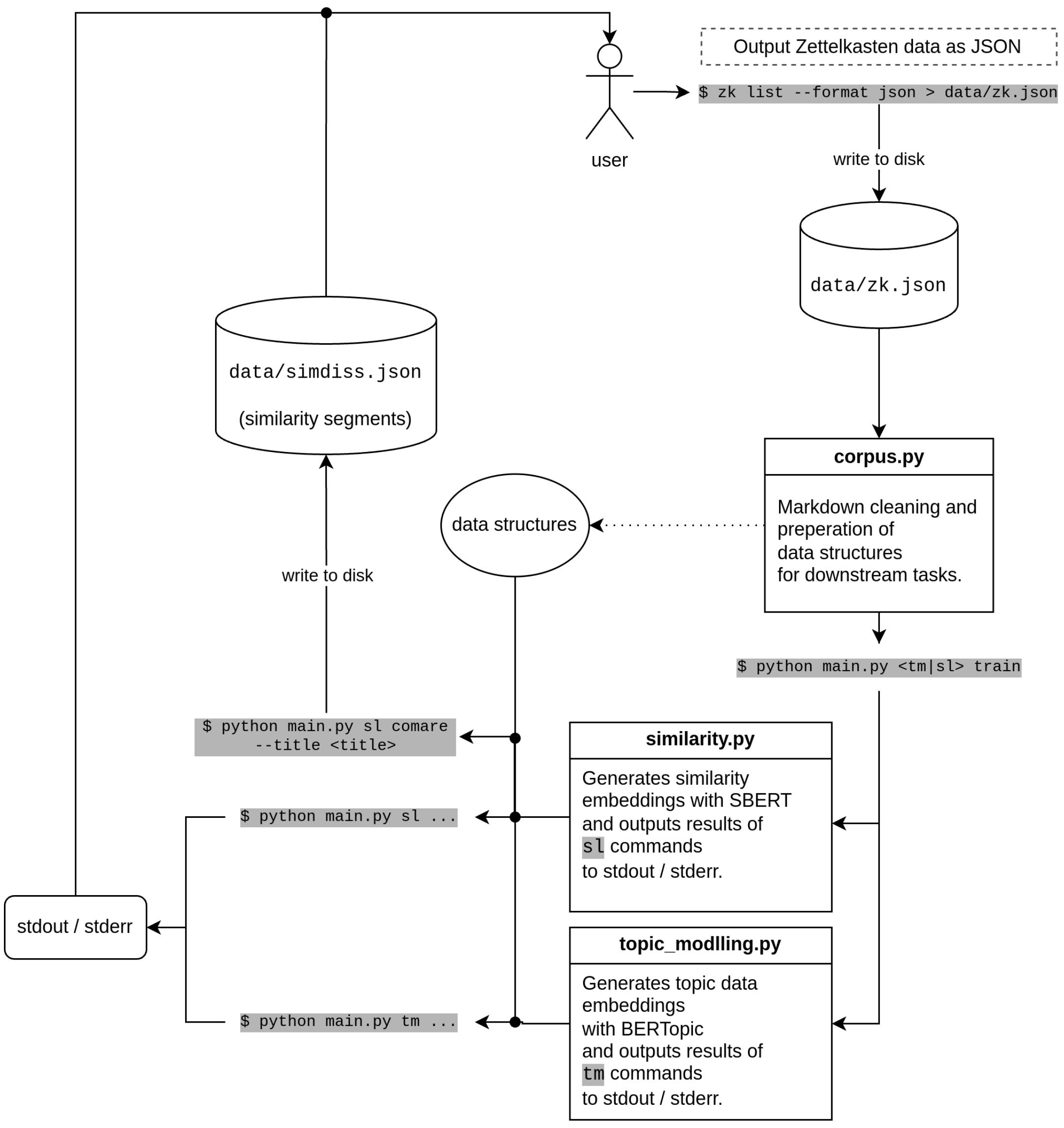

Output Zettelkasten data as JSON
$ zk list --format json > data/zk.json
user
write to disk
data/zk.json
data/simdiss.json
(similarity segments)
corpus.py
Markdown cleaning and preperation of data structures for downstream tasks.
data structures
write to disk
$ python main.py <tm|sl> train
$ python main.py sl comare --title <title>
similarity.py
Generates similarity embeddings with SBERT and outputs results of sl commands to stdout / stderr.
$ python main.py sl ...
stdout / stderr
topic_modlling.py
Generates topic data embeddings with BERTopic and outputs results of tm commands to stdout / stderr.
$ python main.py tm ...

Figure 12: High level diagram of program flow and program architechture.

6.7 Classes and Algorithms

This section draws attention to bespoke implementations and technical design decisions that were defined after the design was set and during implementation.

6.7.1 Similarity Learning

note_simdiss()

The `note_simdiss()` function is the main technical contribution to the set of similarity learning features, as all other functionality is largely unchanged API methods of either the SBERT or BERTopic libraries.

It is the function that is called when executing: `main.py sl compare --title TITLE --strategy STRATEGY`.

The role of the function is to translate raw similarity data stored in the similarity tensor, into interpretable and useable structured data following the Likert style scale discussed in subsection 5.6.2, *Similarity Evaluation*. The tensor itself is an $n \times n$ matrix where each row and column header represents a singular note in the Zettelkasten. The elements in the rows and columns are the similarity scores calculated against all other notes. As depicted below in figure 13 for clarity.

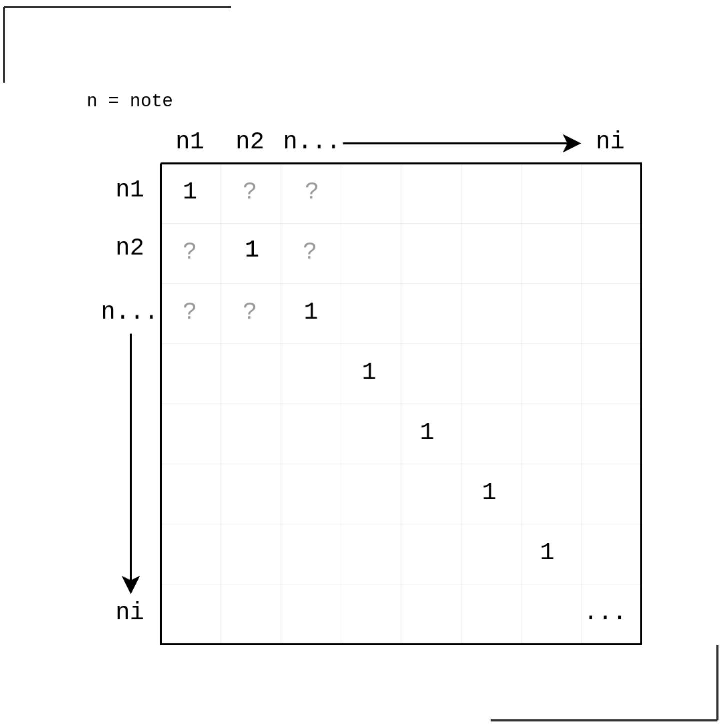


Figure 13: Graphical depiction of a similarity tensor.

```python
def note_simdiss(similarities, title, strategy):
    print(f"Calculating similarities against: {title}")
    # s for segments. For "least similar" to "most similar" notes.
    s1, s2, s3, s4, s5 = [], [], [], [], []

    try:
        note_index = corpus.get_index_from_title(title)
    except:
        print(f"Error retrieving index by title: {title}")
        print(
            f"The note either does not exist, or the note data does not contain this note."
        )
        exit(1)

    titles_paths = list(zip(corpus.titles, corpus.paths))

    # Retrieve similarity scores for the incoming note only
    similarities = similarities[note_index]

    if strategy == "even":
        min, max = unbiased_min_max(similarities, note_index)
        div1, div2, div3, div4 = even_divisions(min, max)
    else:
        div1, div2, div3, div4 = std_dev_divisions(similarities)

    for i, score in enumerate(similarities):
        if i != note_index:
            sim = round(score.item(), 2)
            element = [titles_paths[i][0], titles_paths[i][1], sim]

            if score <= div1:
                # order: least similar to most similar
                s1.append(element)
            elif div1 < score <= div2:
                s2.append(element)
            elif div2 < score <= div3:
                s3.append(element)
            elif div3 < score <= div4:
                s4.append(element)
            elif div4 < score:
                s5.append(element)

    path = corpus.paths[note_index]
    build_json_file(title, path, s1, s2, s3, s4, s5)
```

Figure 14: Function that retrieves segment boundaries and generates the JSON similarity results, outputting them to `simdiss.json` by calling `build_json_file()`.

The user has a choice of segmenting the groupings evenly or by standard deviation. This is useful in cases where there are many similar notes. Using the standard deviation division strategy will constrain the range of notes that qualify on the outer bands of the normal distribution.

The groupings are output to `data/simdiss.json` where they can be inspected or integrated into downstream tasks.

```
def build_json_file(note_title, note_path, s1, s2, s3, s4, s5):

    # Sort by similarity score
    for group in [s1, s2, s3, s4, s5]:
        group.sort(key=lambda x: x[2])

    json_data = {
        "title":
        note_title,
        "path":
        note_path,
        "least_similar": [{
            "title": title,
            "path": path,
            "similarity": sim
        } for title, path, sim in s1],
        "somewhat_similar": [... redacted for practicality],
        "moderately_similar": [...],
        "very_similar": [...],
        "most_similar": [{
            "title": title,
            "path": path,
            "similarity": sim
        } for title, path, sim in s5],
    }

    try:
        with open(corpus.simdiss_results, "w") as file:
            json.dump(json_data, file)
    except:
        print(f"Could not write file {corpus.simdiss_results}")
    else:
        print(f"Results saved to {corpus.simdiss_results}")
```

Figure 15: Function that structures and writes the `simdiss.json` file.

As an example of a possible downstream task, the below bash command pipes the most similar notes to the `fzf` picker. Upon selection the path is returned to the shell's standard output, which in this case is provided to the Zettelkasten program `zk` as the file to open:

```
#!/usr/bin/env bash

zk edit $(cat data/simdiss.json
        | jq -r '[.most_similar[]]?
        | .[]
        | "\(.title)\t\(.path)"'
        | fzf --with-nth=1 --delimiter=$'\t'
        | cut -f2)
```

Figure 16: Example bash script illustrating how results contained in `simdiss.json` can be integrated into user specific downstream tasks.

`cos_sim_elementwise()`

The cosine similarity function that comes with SBERT is built for processing pairwise similarity, meaning each element of two different tensors as input[5].

As this implementation takes one tensor as input (the entire Zettelkasten) and requires that each element (note) is compared against every other, it was necessary to modify the source code of SBERTs cosine utility function: `SentenceTransformers.util.cos_sim()`.

```
def cos_sim(a: list | np.ndarray | Tensor, b: list | np.ndarray | Tensor) -> Tensor:
    a = _convert_to_batch_tensor(a)
    b = _convert_to_batch_tensor(b)

    a_norm = normalize_embeddings(a)
    b_norm = normalize_embeddings(b)
    return torch.mm(a_norm, b_norm.transpose(0, 1))
```

Figure 17: Default function to calculate cosine similarity from the SBERT Python library.

```
from sentence_transformers import util as sbert_utils

def cos_sim_elementwise(embeddings):
    a = sbert_utils._convert_to_batch_tensor(embeddings)
    a_norm = sbert_utils.normalize_embeddings(a)
    return torch.mm(a_norm, a_norm.transpose(0, 1))
```

Figure 18: Modified version of default function to suit specific use case.

Clustering

Agglomerative clustering was chosen as Fast Clustering is designed for data sets much larger than a typical Zettelkasten corpus[6]. This was confirmed when testing on the test corpus of 516 notes whereby there was not enough variance in similarity between notes. This meant that it was difficult to fine tune the parameters to achieve more than 3 clusters.

[5]https://sbert.net/docs/sentence_transformer/usage/semantic_textual_similarity.html

[6]https://sbert.net/examples/sentence_transformer/applications/clustering/README.html#fast-clustering

6.7.2 Topic Modelling

```
# topic_modelling.py
class BTopic:

    @classmethod
    def init(cls, embedding_model, notes, titles, titles_dict):
        cls.embedding_model = embedding_model
        cls.notes = notes
        cls.titles = titles
        cls.titles_dict = titles_dict
        vectorizer = CountVectorizer(ngram_range=(2, 2),
                                     stop_words=standard_stopwords)

        cls.topic_model = BERTopic(top_n_words=10,
                                   n_gram_range=(2, 2),
                                   nr_topics=10,
                                   embedding_model=embedding_model,
                                   vectorizer_model=vectorizer,
                                   umap_model=UMAP(n_neighbors=10,
                                                   n_components=5,
                                                   min_dist=0.0,
                                                   metric='cosine',
                                                   random_state=42))
```

Figure 19: The parameters used to instantiate BERTopic.

The function called by `python main.py tm list --related "title"` required a custom implementation. All other functions are provided by built-in methods.

The function, although custom, is simple. It retrieves the note index from the training data and uses it to subsequently retrieve the topic data stored against that note's index. With the topic ID at hand, the other documents pertaining to that topic can be returned.

```
# topic_modelling.py

@classmethod
def list_topically_related_notes(cls, title):
    note_index = cls.titles_dict[title]
    df = cls._get_topic_data()
    note_topic = df["topic"][note_index]
    print(f"Documents topically related to: {title}\n")
    cls.list_docs_for_topic(note_topic)
```

Figure 20: Combination of BERTopic API functions to create extended topic modelling functionality.

Chapter 7: Results and Discussion

The software is released under the GNU General Public License v3 and is available for public use at `https://git.sr.ht/~tjex/thesis-code`. All that is required to set up and run the program is a few basic terminal commands. Thereafter, any user with a Markdown-based Zettelkasten can export their notes (using the `zk` tool), run the models and access the resultant features directly. Precise steps for installation and usage are documented in the `readme.md` in the source code repository.

The software outputs data in JSON, tensor, DataFrame, and pickle formats, enabling the integration of downstream tasks. This can be bash scripts, new features developed by developers or further exploration and study by researchers.

A demonstration of the functionality can be viewed online in the same source code repository above and therefore also comes with the repository as a `.gif` and can be played back locally.

7.1 Identifying Difficulties of Bottom-Up Structure

From the analysis of Zettelkasten and its comparison to traditional note-taking systems, several core difficulties were identified, which could be formulated as questions from the user's perspective. These questions were derived by reported user experiences in the public poll as well as the authors own experience as a Zettelkasten user.

- *"What notes do I have that are similar to note x?"*
- *"What clusters of similar notes exist within my Zettelkasten?"*
- *"Do I have any notes on topic x?"*
- *"What other notes do I have on topic x?"*
- *"What notes do I have that are topically related to note this note?"*

7.2 Program Functionality

The program resulted in the following commands, which are paired with their guiding design questions below.

7.2.1 Similarity Learning

"What clusters of similar notes exist within my Zettelkasten?"

```
usage: main.py sl cluster [-h] [--clusters CLUSTERS]

options:
  -h, --help           show this help message and exit
  --clusters CLUSTERS  Number of clusters.
```

"What notes do I have that are similar to this note?"

```
usage: main.py sl compare [-h] [--title TITLE] [--strategy STRATEGY]

options:
  -h, --help           show this help message and exit
  --title TITLE        Title of note to compare.
  --strategy STRATEGY  Strategy to group notes: std (i.e, standard deviation) or even.
```

7.2.2 Topic Modelling

"Do I have any notes on topic x?"

```
usage: main.py tm search [-h] term

positional arguments:
  term        Topic term to search.

options:
  -h, --help  show this help message and exit
```

In order: *"What topics exist within my Zettelkasten?"*, *"What other notes do I have on this topic?"* and *"What notes do I have that are topically related to note this note?"*

```
usage: main.py tm list [-h] [--topics] [--docs-for-topic DOCS_FOR_TOPIC] [--related RELATED]

options:
  -h, --help            show this help message and exit
  --topics              List topics
  --docs-for-topic DOCS_FOR_TOPIC
                        List documents belonging to a given topic.
  --related RELATED     List other notes that share the same topic.
```

7.3 Similarity Learning

The similarity learning component allows the user to compare any given note with all others in their corpus. For example, querying similarity for the note *Zettelkasten is an interface for thought* produced expected results:

- *Most similar note*: *Zettelkasten promotes ideation due to inter-note linking*, with a similarity score of 83%.
- *Least similar note*: *zombie-ant fungus*, with a score of -3%.

A negative similarity score appears to be a rounding artifact, possibly caused by the rounding behavior of the Python `round()` function in combination with NumPy's standard deviation calculations.

7.3.1 Effects of Markdown Cleaning

A custom text-cleaning pipeline was developed to remove Markdown formatting prior to similarity computation. This improved result quality. For instance, the note *Deutsche Telekom* had its similarity score reduced from 15% to 8% after formatting was removed, aligning better with human judgement of topical relevance.

```
# Deutsche Telekom

> **Deutsche Telekom** AG is a German telecommunications company headquartered
> in Bonn and is the largest telecommunications provider in Europe by revenue.
> It was formed in 1995 when Deutsche Bundespost, a state monopoly at the time,
> was privatized. Since then, Deutsche Telekom has consistently featured among
> Fortune Magazine's top Global 500 companies by revenue, with its ranking as of
> 2022 at number 62. The company operates several subsidiaries worldwide,
> including the mobile communications brand T-Mobile.
>
> As of April 2020, the German government held a direct 14.5% stake in company
> stock and another 17.4% through the government bank KfW. The company is a
> component of the EURO STOXX 50 stock market index.
>
> [Wikipedia](https://en.wikipedia.org/wiki/Deutsche%20Telekom)
```

Figure 21: A note which was downgraded in similarity to the test note after markdown cleaning. Both this note and the test note have > characters.

7.3.2 Edge Case Testing

Testing with the least representative note, *vampire bat in denim*, demonstrated that the algorithm was still capable of producing meaningful similarity distributions. The standard deviation-based Likert segmenting successfully assigned notes across the entire span similarity groupings:

```
"least_similar": [
  { "title": "Container Deployment", "similarity": -0.17 }
],
[ … ],
"most_similar": [
  { "title": "Sam Sarris", "similarity": 0.13 },
  { "title": "AI Spider Character from Neuromancer", "similarity": 0.14 },
  { "title": "zombie-ant fungus", "similarity": 0.21 }
]
```

Figure 22: End ranges of similarity groupings for the least similar note.

These results confirm that even highly atypical content yields meaningful clusters, presenting the user with potential linking options that can lead to further idea generation. This would otherwise be a difficult task, given the source note is atypical to the content of the Zettelkasten at large and has no pre-existing inbound or outbound links.

7.4 Topic Modelling

BERTopic provided useful and accurate topic models. However, significant fine-tuning was required to achieve this. The below is the output of `main.py tm list --topics` when using BERTopics default values with the removal of stopwords:

```
Schema:
[Topic ID: (top three topic labels)]

Outliers: (data, knowledge, source)
0: (technology, ai, people)
1: (zettelkasten, notes, system)
2: (meditation, loc, unconscious)
3: (creativity, creative, ideas)
4: (memory, memories, cognitive)
```

Figure 23: Derived topics using default BERTopic settings.

This is already a satisfying result. Here, BERTopic has defined five topic groupings. These are topics that satisfy the thresholds of BERTopic's default values such as `min_topic_size` and `nr_topics`[1].

Fine tuning with the following parameters improved results:

```
embedding_model = SentenceTransformer("all-mpnet-base-v2")
vectorizer_model = CountVectorizer(ngram_range=(1, 2),
                                   min_df=2,
                                   stop_words=standard_stopwords)

cls.topic_model = BERTopic(top_n_words=10,
                           n_gram_range=(1, 2),
                           min_topic_size=8,
                           nr_topics="auto",
                           embedding_model=embedding_model,
                           vectorizer_model=vectorizer_model,
                           umap_model=UMAP(n_neighbors=10,
                                           n_components=5,
                                           min_dist=0.0,
                                           metric='cosine',
                                           random_state=42))
```

Figure 24: Tuned BERTopic settings accounting for improved results.

[1] https://maartengr.github.io/BERTopic/getting_started/parameter%20tuning/parametertuning.html

```
Schema:
[Topic ID: (top three topic labels)]

Outliers: (data, 2022, system)
0: (technology, ai, people)
1: (notes, zettelkasten, note)
2: (creativity, creative, ideas)
3: (ref, loc, meditation)
4: (zettelkasten, network, graphs)
5: (actor, network, actor network)
6: (skill, focus, multitasking)
7: (paper, section, abstract)
8: (memory, memories, remember)
9: (narrative, vr, virtual)
```

Figure 25: Improved topic results after fine-tuning BERTopic settings.

The fine-tuning here has allowed for more topics to be returned and better topic labelling. This has the side effect of providing access to notes (via `main.py tm list --docs-for-topic`) that were not previously available, due to fewer topics being returned.

This shows that fine-tuning BERTopic is crucial and achieves significant improvements. While this is exciting, it also poses a challenge for accessibility, delivery and adoption of the software as the end user must manually adapt the model themselves to best suit their needs and data set.

Returning to the least representative note, *vampire bat in denim* from before[2], the topically related functionality proved fruitful. Running `python main.py tm list --related "vampire bat in denim"` returned a list of notes that are deemed by BERTopic to be related by topic matter. The cause for mention here is that the notes in the returned list contained arguably all other notes in the Zettelkasten that were documentations of dreams or nightmares[3]. Additionally, the quality of relatedness for non-dream notes is very strong. Below is the shortened list, which through the author's own evaluation, includes notes on topics revolving around meditation, hypnosis, creativity and philosophy.

[2]See subsection 7.3.2, *Edge Case Testing*

[3]I do not know exactly how many dreams are documented in my Zettelkastsen, as I (hopefully unsurprisingly) have not imparted a filing system in my Zettelkasten, and therefore cannot count the amount of notes in a ”dreams” subdirectory or otherwise.

```
Documents topically related to: vampire bat in denim

1. Chronotype
2. Digital Ventriloquism - Giving Voice to Everyday Objects
3. Dreams of Dali
4. Farvel.space
5. Follow curiosity in your environment
6. Follow your interests and always take the path that promises the most insight
7. Having faith in yourself when learning difficult things is key
8. How to Change Your Mind
9. Interactive Robotic Worm
10. Kariesa swimming
11. Lara in quicksand
12. Last Day at Disney Land
13. Luck is what happens when preparation meets opportunity
14 ... 44
44. rigidity leads to a loss of spirit
45. should all good habits be started early?
46. the wandering horse
47. untitled
48. vampire bat in denim
49. you don't need luck, luck needs you
50. your awareness of connection to organic spirit
51. zombie-ant fungus
```

Figure 26: Topically related notes returned for the test note *vampire bat in denim*. Notes 14 to 44 are removed for display practicallity.

Of note is the last result, *zombie-ant fungus*. This is a convenient opportunity to draw the reader's attention back to the Similarity Learning subsection above where the note, *zombie-ant fungus*, was returned as the *least similar* note to the test note, *Zettelkasten promotes ideation due to inter-note linking*[4]. With this preface, an exemplary use case can be built that describes a typical workflow difficulty while using Zettelkasten, and where this software can be applied to help mange it.

7.5 Exemplary Use Case

Consider for a moment, that you have awoken from a hellish nightmare featuring a "vampire bat in denim". After documenting it in your Zettelkasten, you want to establish links to other notes, so that it is not lost forever. Out of interest, you would like to see if there are any other dreams that have similarities to this note. So, you pose the question *"What notes do I have that are topically related to this note?"*. I.e, *"What other dreams or dreamlike notes do I have in my Zettelkasten?*.

[4]See subsection 5.6.1, *Test Note*.

From memory alone, no fitting candidates spring to mind. And as all your notes are stored in one directory, there is no ”dreams” subdirectory to navigate to in order to find potential candidates. And so you turn to the Zettelkasten, and ask it the same question by opening your computer’s terminal and executing `python main.py tm list --related "vampire bat in denim"`.

Your Zettelkasten answers with the list already shown above, and here shortened:

```
Documents topically related to: vampire bat in denim


1. Chronotype
2 ... 50
51. zombie-ant fungus
```

Out of curiosity you pick the note *zombie-ant fungus*, mistakenly thinking it is also documentation of a dream. You are surprised that it is not, and is instead a note about a natural phenomenon found in tropical rainforests[5]. You find there are some relationships between the two notes and create a link. Now those two notes have been *simplified*[6], which has occurred through a curiosity lead serendipitous discovery of relationships between dreams and the real world. Well and truly awake at this point, you wonder, *”What notes do I have that are similar to this (zombie-ant fungus) note?”*.

Unfortunately, your past self did not add any outbound links when writing it[7], and your Zettelkasten has little to do with biology, as revealed by the topic labels returned by `main.py tm list --topics` (figure 25). Faced with an arduous task of finding notes related to biology without a good starting point, you reduce the difficulty by returning to the terminal and asking the same question, by executing: `python main.py sl compare --title "zombie-ant fungus"`.

The Zettelkasten returns to you a tempting offering in the ”most similar” category, a note titled *Encyclopedia of Social Theory - ANT*, but to *challenge yourself even further*, you navigate to the *least related* segment, which includes the note, *Zettelkasten promotes ideation due to inter-note linking*. After flexing your faculties of thought, you find there are connections to be made between ideas emerging from the Zettelkasten, and mushrooms emerging from the head of an infected ant.

[5]"Zombie-antfungus":https://en.wikipedia.org/wiki/Ophiocordyceps_unilateralis

[6]See section 3.3, *Applying ANT to Zettelkasten*.

[7]See Appendix A, *Zombie-Ant Fungus Note*.

There now exists pathways between your nightmare, a tropical fungus and a theoretical concept about how creating links between notes in Zettelkasten has the ability to promote new ideas. And as the note on Zettelkasten is part of a more densely connected area of the network, so too now are the notes on the zombie-ant fungus and your nightmare.

The question here to ask is whether these notes would ever have come into contact if there was a filing structure present? As discussed, imparting a filing structure is a common strategy to group similar notes together, thereby aiding in the recall of information and organisation of ideas. But just because the information is structured in an orderly way, does not mean it becomes more useful. As quoted at the start of chapter 2, *Zettelkasten*, Kurzweil states that *"order is information that fits a purpose. The measure of order is the measure of how well the information fits the purpose"*. The purpose of Zettelkasten is to facilitate "combinatory play", as Einstein calls it: to make connections, particularly where relationships are not obvious. By this notion, a rigid filing structure does not fit this purpose, as it presupposes the identity of a note, when in reality the identity of a note develops over time. This is expressed by the concepts of simplification and translation from Actor-Network Theory.

The value then of this resultant software, is that the user has access to an additional tool within their Zettelkasten medium, that allows them to *momentarily* inspect their Zettelkasten in different structures. In doing so, they can manage the difficulty of Zettelkasten's bottom-up structure, without altering that structure directly. This provides a "way out" of tricky situations, while maintaining the benefits that the all-files-in-one-folder approach provides Zettelkasten. This can be thought of as a *virtual directory structure*, and is illustrated graphically on the following pages. But as a short textual clarification: The question of *"What topics do I have in my Zettelkasten?"*, would be akin to all notes in the Zettelkasten organised on the user's computer into subdirectories suh as "Technology", "Zettelkasten", "Creativity", "Meditation", and so forth, which are some of the topis derived by BERTopic in this case[8].

[8]See Figure 25, *Improved topic results after fine-tuning BERTopic settings.*

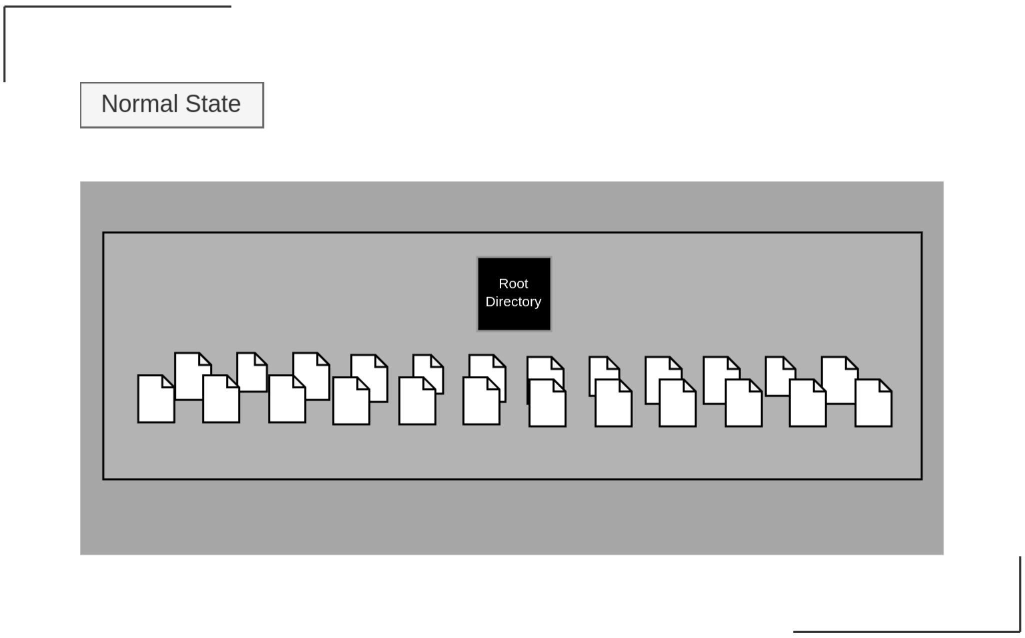
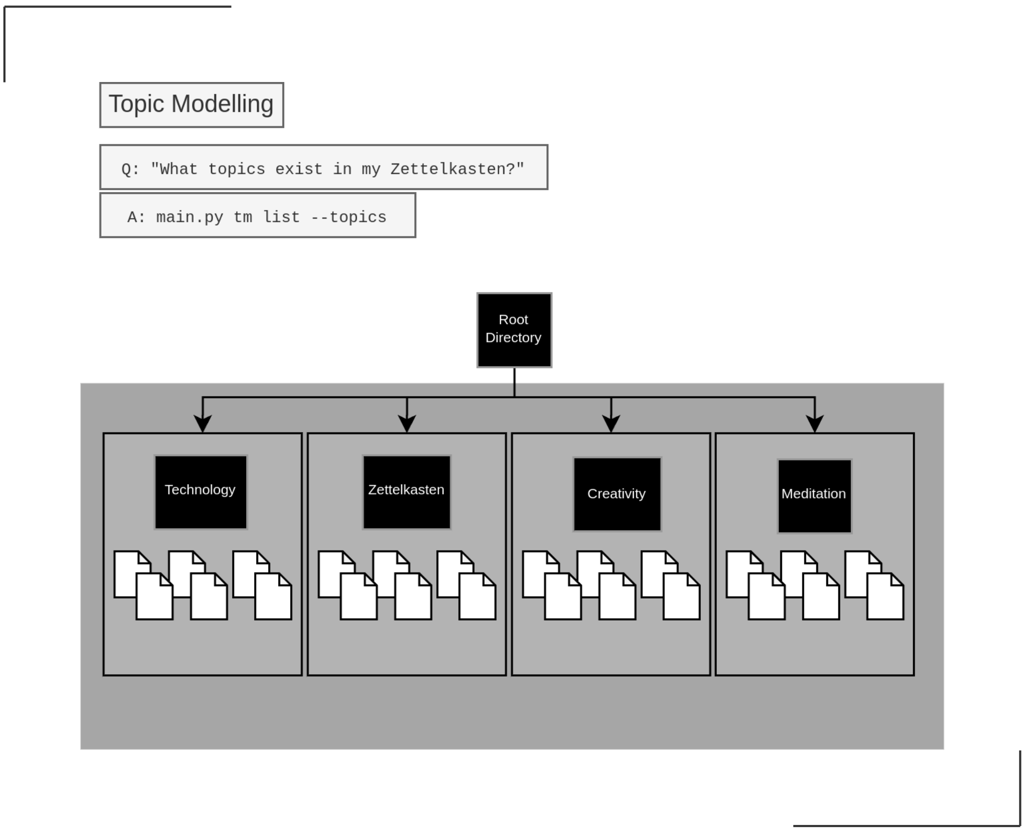
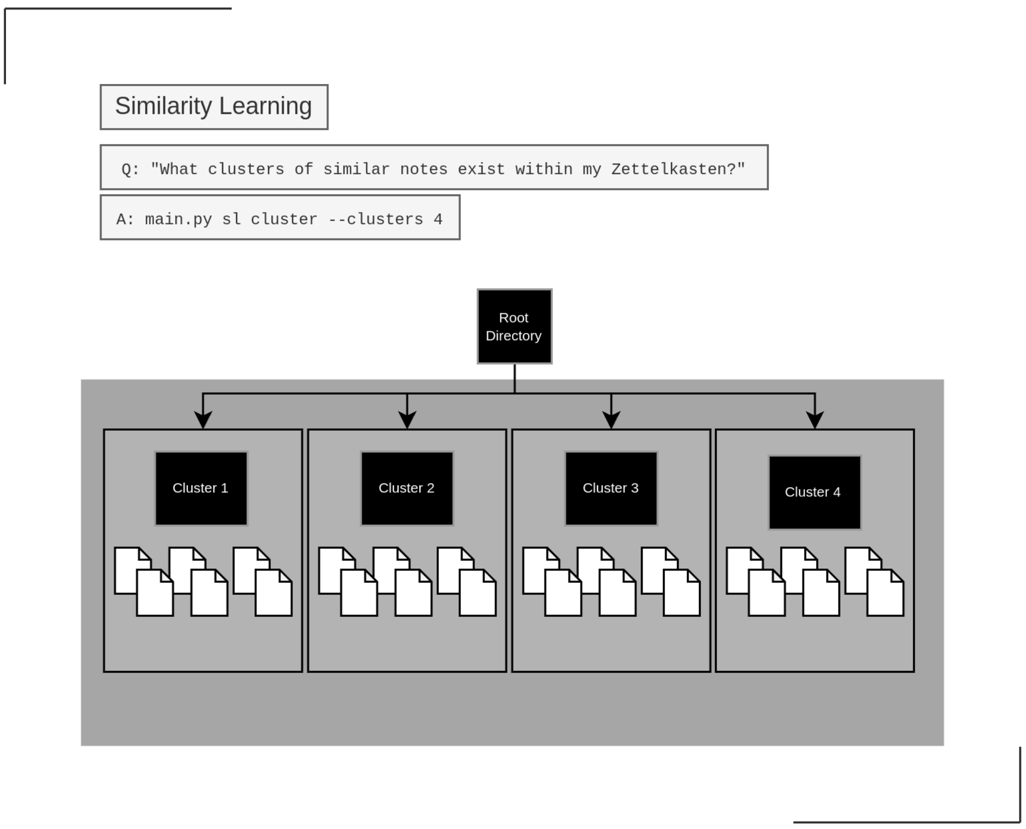


Figure 27: Normal state of the Zettelkasten (all notes in one directory).



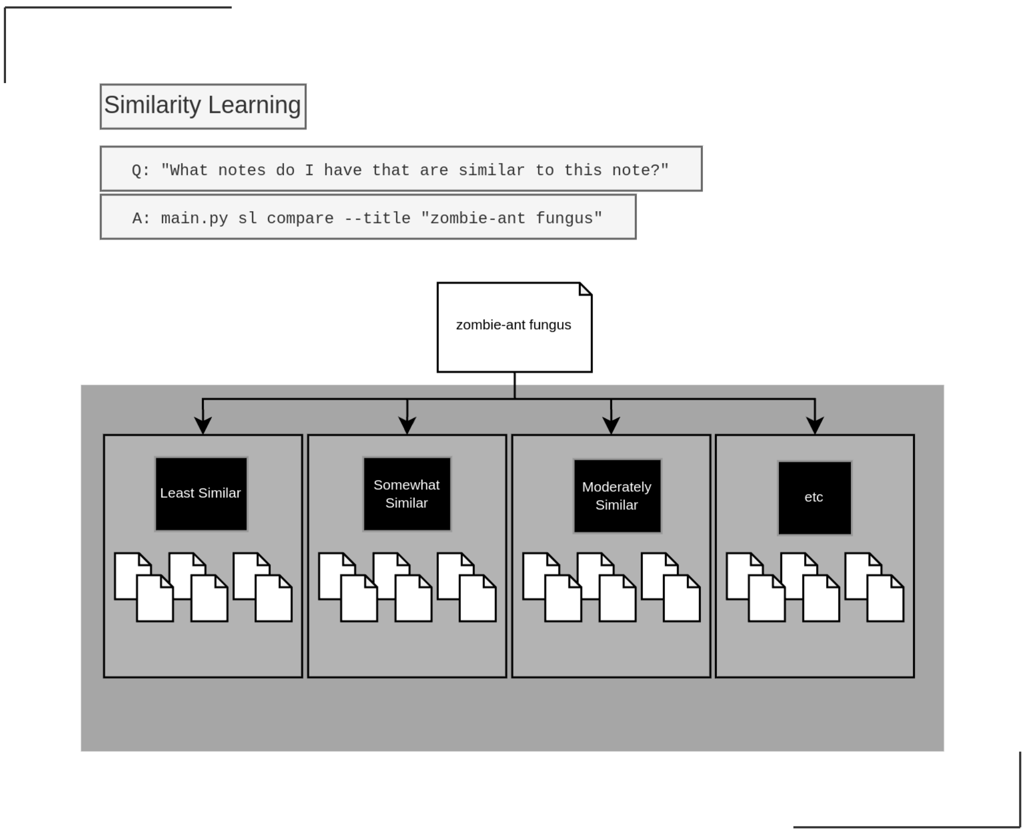
```
Q: "What topics exist in my Zettelkasten?"
A: main.py tm list --topics
```

Figure 28: Emulating the organisation of notes by directories of topics.



```
Q: "What clusters of similar notes exist within my Zettelkasten?"
A: main.py sl cluster --clusters 4
```

Figure 29: Emulating the organisation of notes by directories of similar notes.



```
Q: "What notes do I have that are similar to this note?"
A: main.py sl compare --title "zombie-ant fungus"
```

Figure 30: Emulating the organisation of notes by directories of notes with varying degrees of similarty to a chosen note.

7.6 Applying Theoretical Research to Software Design

One of the leading questions of this thesis was understanding how Zettelkasten works as a system. This was important because in order to build software for any system, the system should first be understood. The initial research went in the direction of systems thinking, and was heavily directed by Donella Meadows' book, *Thinking in systems: a primer*[9]. But it became apparent that this understanding of systems was too strongly situated in the material world. It was through discussion with Professor Kim Albrecht, where the author was introduced to Bruno Latour's journal article, *The whole is always smaller than its parts*[10]. This provided contact with ideas that were more applicable to understanding Zettelkasten — namely the "one level standpoint" (1-LS)[11] — and subsequently lead to Latour's work on Actor-Network Theory, along with his colleagues Callon and Law.

Actor-Network Theory (ANT) offered an established and robust framework for understanding how Zettelkasten works as a system. Specifically, as a system of interaction between actors, which can be human or non-human. It was particularly helpful in explaining how Zettelkasten — being fundamentally comprised of one human and their notes — can yield emergent ideas not explicitly written or even considered by the user[12]. This in itself, the demystification of Zettelkasten's ability to generate ideas, was a secondary leading question this thesis set out to answer.

This understanding went on to inform a crucial design principle: that in a medium for thought such as Zettelkasten, software features should behave as intermediaries, not actors, and most certainly not the prime mover. In doing so, the user's role as the prime mover of the Zettelkasten network is preserved. This particular finding is reflected in the software itself in that the machine learning features are *not* generative. Generative features could be, for example, writing text for the user, creating links between notes and chatbot like functionality. Powerful new capability such as machine learning is naturally very tempting to integrate into existing technologies, and as shown in subsection 4.3.1, *Existing Solutions*, many have. But the amount of influence that machine learning technologies can have on existing processes is significant. And, particularly in the case of tools for thought, there is public critique that AI powered software should "challenge,

[9]Meadows, *Thinking in systems*.

[10]Latour et al., "'The whole is always smaller than its parts' – a digital test of Gabriel Tardes' monads".

[11]See chapter 3, *Actor-Network Theory*

[12]Cevolini, "Niklas Luhmann's Card Index: Thinking Tool, Communication Partner, Publication Machine", p. 295.

not obey"[13], which speaks against this trend for AI to do things for us "so we don't have to" — and is the shared opinion of the author. With ANT, it was significantly easier to theorise what the effect of an AI obeying, instead of challenging, would look like in the case of Zettelkasten. This gave confidence to the software design guidelines and principles.

ANT therefore provided an invaluable language and framework with which to understand an abstract system made of human and data, and positively informed how the machine learning software was designed. This resulted in the software being designed to support, rather than alienate, the user's thinking processes, which is the fundamental capability of Zettelkasten. It also led to the development of the "virtual filing" system concept, as ANT, and in particular the one-level standpoint, invites the visualisation of networked structures in different forms and from different perspectives.

The user is therefore provided tools to help manage the difficulties of Zettelkasten's bottom up structure, without altering the user's role in the network, without alienating them from their capacity to think, without imparting change on Zettelkasten's all-files-in-one-folder idiom, and in a way that works with the foundational Zettelkasten processes of discovery through serendipity.

[13]Sarkar, "AI Should Challenge, Not Obey".

Chapter 8: Limitations, Conclusion and Future Work

8.1 Limitations

8.1.1 Accessibility

The software requires a degree of proficiency with software development tools to use it. The user needs to be comfortable entering commands in the command-line interface at a minimum. This is required for installing the necessary dependencies as well as using the features and is usually a daunting task for the average computer user.

Future work could package up the software and deliver as a regular install image and also provide a graphical user interface (GUI) for operation.

8.1.2 Parameter Tuning and Model Retraining

As shown in section 7.4, *Topic Modelling*, BERTopic required fine-tuning across nine parameters to achieve above satisfactory results. Each time the parameters are changed, the embeddings need to be recalculated. Depending on the size of the Zettelkasten and available computational power, this can create a laborious process for the user in tuning the model for best performance against different collections. Time and effort of tuning aside, the user also needs to understand what the parameters are doing, as with this many parameters, there are too many combinations to rely on trial and error alone.

This could be addressed by researching the optimal parameters for Zettelkasten like repositories of data, i.e, collections of small documents with average word counts of around 250. Or software could be developed to scan a user's Zettelkasten repository and recommend the values that each parameter could be set to.

8.1.3 Usability Study

This thesis did not include a formal usability study to evaluate the software's efficacy. This omission was deliberate as the central aim of this thesis was not to validate a specific implementation, but to explore how conceptual frameworks, particularly Actor-Network Theory and the notion of Zettelkasten as a medium for thought, can inform software design. A usability study was therefore out of scope for this thesis both in terms of focal points and time constraints.

This leaves plenty of space for future work as to the efficacy of the software, and therefore also a deeper research into the qualitative effect of theory based software design.

For example, a tool set of generative AI features could be developed, or existing generative AI implementations could be used such as those in subsection 4.3.1, *Existing Solutions*. Three groups could then be studied: one control group without any machine learning software extending their Zettelkasten software, one with the software of this thesis and one with the generative AI software. The effect of generative AI features versus non-generative features could be measured qualitatively. The quality of ideas and output could be evaluated, and the users could report on how they feel their critical thinking processes have been informed by the additional software. In other words, supporting or challenging the notion that AI should ”challenge, not obey”.

For a testing methodology, researchers could refer to the study, *The Impact of Generative AI on Critical Thinking: Self-Reported Reductions in Cognitive Effort and Confidence Effects From a Survey of Knowledge Workers*[1]. Additionally, *Idea Generation and the Quality of the Best Idea*[2] could provide useful insights as to how to measure the quality of ideas.

Considerations for a Usability Study

An exploratory attempt at a usability study was made by posting an open call for participation in the GitHub discussion forum of the `zk` project[3]. The `zk` program is required for using the software of this thesis. It therefore made sense to source participants from this project. Despite the project being relatively active for an open-source tool with around 1500 monthly views, one

[1]Lee et al., "The Impact of Generative AI on Critical Thinking: Self-Reported Reductions in Cognitive Effort and Confidence Effects From a Survey of Knowledge Workers".

[2]Girotra, Terwiesch, and Ulrich, "Idea Generation and the Quality of the Best Idea".

[3]`https://github.com/zk-org/zk/discussions/497`

response was received.

Zettelkasten itself is quite niche, gaining recent significant momentum with the book, *How to Take Smart Notes*[4]. In such a community, gathering enough participants to collect meaningful and statistically significant qualitative results is a challenge. Future researchers should take this into account when considering undertaking a usability study.

A second key factor that limits broader engagement for the scope of this thesis and its software, is the technical requirement to use the `zk` command-line tool to export note data into a structured JSON format suitable for ingestion by SBERT and BERTopic. This prerequisite creates a barrier to entry for users of other tools or workflows (such as Obsidian, where the majority of the digital Zettelkasten user base exists).

Improving this technical barrier to entry would make finding participants for a usability study much easier.

8.2 Conclusion and Future Work

This project was by all means an interesting and fruitful exploration. It was a luxury to approach software development in a deeper yet also highly abstract way, and perhaps as a composer now turning software developer, this makes me feel more at home. It was pleasing to find that even the more formal research into Zettelkasten by Schmidt, was not able to fully escape the clutches of a certain mystique that pervades Luhmann and his ”conversation partner”. Such an aesthetic suggests more so a musician and his instrument, rather than a sociologist and his collection of notes, and hopefully this thesis has given enough cause to realise that this interpretation is agreeable — that each person can have the same relationship with their Zettelkasten, as a musician can have with their instrument. They both support the function of *translating* input to output and — despite being inanimate objects — of communicating back to the person. The person can respond with further output, and the cycle continues. This cycle is a *feedback* cycle, and is not only a fundamental defining characteristic of a system[5], but also of conversation and ideation.

[4]Ahrens, *How to take smart notes*.

[5]Meadows, *Thinking in systems*.

But this did not lead to sufficiently understanding *how* Zettelkasten works. It is not enough to conclude that the system works because one can converse with it, and as conversation is a feedback loop, ideation can occur. This naturally opened the deeper question of how does one converse with ”it”, where ”it”, is a network of data?

A question to a similar effect is, how can you have a conversation with yourself? And perhaps this is an easier question to embody, as this is encapsulated in the act of self reflection[6]. Through reading Luhmann’s own reflections such as *Communicating with Slip-Boxes*[7], as well as Schmidt’s research *Niklas Luhmann’s Card Index: The Fabrication of Serendipity*[8], two key aspects became clear. One, was that despite Zettelkasten being a ”second memory”, externalised into data, it simultaneously always exists internally in the user as well. It is just that the ”second memory” does not forget. The second was that the ”magic” of Zettelkasten came not necessarily from the feedback system of writing and re-reading, nor that Zettelkasten notes are concise and therefore easy to digest. Rather, it is the *pathways* that the user takes. That is, it is the *movement* through the network that invokes the act of communication.

As explained in section 7.6, *Applying Theoretical Research to Software Design*, Actor-Network Theory provided a language and framework to formalise this idea about Zettelkasten: that as the user (prime mover) moved through the network, reading notes along the way (translation), they may identify and make connections (simplification) and also stop to write a resultant idea (emergence). During this process, there is little to no distinction between the user and the Zettelkasten. The conversation is occurring both within the mind of the user, and in the Zettelkasten, as they are parts of the same network. The perception they are different is due to our tools of observation, which create the illusion that we are the prime *observers* of nature and all entities within it[9]. This is the *one-level standpoint* at play, in that there is no ”part” or ”whole”. So, one way you can have a conversation with yourself, in other words to *think*, is to use Zettelkasten. And as the ”second memory” never forgets, your trains (pathways) of thought are also preserved, allowing the power of the creative mind and permanence of information to meet.

[6]Brinthaupt and Morin, “Self-talk”.

[7]Luhmann, *Communicating with Slip Boxes by Niklas Luhmann*.

[8]Schmidt, “Niklas Luhmann’s Card Index”.

[9]Desrosières, *The Politics of Large Numbers: A History of Statistical Reasoning*.

At this point, the importance of the *relatedness* of the network and its contents to the user became chronically obvious. The notes and links between them need to be made *purposefully*. Otherwise, the second memory becomes foreign, and not in an interesting or stimulating way, but in a confusing and difficult way. The symbiosis between the primary memory and the secondary memory would be degraded. This motivated the conviction that the machine learning features should *not* generate any content, create links, or otherwise ”think” on behalf of the user. Instead, the abilities of machine learning are used to momentarily restructure the Zettelkasten based on semantic relationships between notes — which is how the Zettelkasten is structured in the first place. By doing so, virtual filing structures are created, which reveal alternate *pathways* between notes. And as it is the *movement* along those pathways that invokes the act of communication, the machine learning features enable a way to change the direction of the conversation when desired.

Developing a deeper understanding of Zettelkasten most certainly aided in the process of writing code, and not just designing what the features should achieve. It was simpler to identify what built-in features should be included from BERTopic's API[10], and it was easier to develop an understanding of when enough features were created. The responses on the zettelkasten.de forum poll helped here as well. Both the SBERT and BERTopic libraries themselves were simple to implement and have a lot more depth, functionality and capability to be explored. It would for example be of significant interest to train a custom SBERT model that is targeted for Zettelkasten notes. As BERTopic implements the SBERT model, the software features of this thesis as a whole would benefit from such a custom SBERT.

Aldous Huxley was quoted in the introduction, that *”people will come to adore the technologies that undo their capacity to think”*. This can be seen in subsection 4.3.1, *Existing Solutions*, where AI tools were developed to take care of cognitive tasks so that ”we don't have to”, and hailed as the only way to truly understand our Zettelkasten. This forges a path to our faculty of thought being outsourced to the AI system, as happened with our faculty of memory with search enginesSparrow, Liu, and Wegner, “Google Effects on Memory: Cognitive Consequences of Having Information at Our Fingertips”.

Focussing on what a technology *removes or replaces* is a way to understand the impact of a technology. It is understandable that for the average internet user in the 1990s that — as they

[10]Unlike BERTopic, SBERT does not come packaged with user features. It ”only” implements core functionality to calculate similarity between documents and return as tensors.

accidentally picked up the phone while their dial-up connection initiated[11] — they did not consider that the internet would be a transformative technology. It is therefore also likely that they did not foresee that the technology of a search engine could undo their capacity of memory. ChatGPT is the fastest growing app in history[12], is clearly adored, also intimately[13] and is being used for tasks as critical as legal defenses[14]. It is increasingly seeming that Huxley was right.

In order to deviate from that path we need to *think critically*. And so, we should support it by designing *transformative tools for thought*[15]. There are many tough problems to solve, and tough problems require clear thinking and strong ideas. It would therefore be of great interest to see research and *open-source* development efforts that interface technologies like virtual reality, machine learning, and new human-computer interfaces in thoughtful and empowering ways that truely *augment* human intelligence. One component of that challenge is the design of effective interfaces for AI powered tools[16].

The children of current and future generations are and will be inextricably linked to digital technology. It is already an inseperable part of their world and therefore their learning environment. So what digital solutions can we build to foster creativity and critical thinking in schools? How can we design better human-computer interfaces or procedural patterns that allow us to more effectively harness AI to our own cognitive abilites? It cannot be, that the pinnacle of AI powered tools for thought, and therefore what children have access to when building their own capacity to think, are LLM chatbots that simply obey.

To solve such problems we need to challenge ourselves, and when stuck...
to go in search of serendipity.

[11]`https://en.wikipedia.org/wiki/File:Dial_up_modem_noises.ogg`

[12]Hu and Hu, "ChatGPT sets record for fastest-growing user base - analyst note".

[13]page, *OpenAI has released its first research into how using ChatGPT affects people's emotional well-being*.

[14]Bohannon, *Lawyer Used ChatGPT In Court—And Cited Fake Cases. A Judge Is Considering Sanctions*.

[15]Matuschak et al., "How can we develop transformative tools for thought?"

[16]Carter and Nielsen, "Using Artificial Intelligence to Augment Human Intelligence".

Bibliography

Aal, Konstantin ”Kosta” and Sarah Rüller. “From Personal Knowledge Management to the Second Brain to the Personal AI Companian”. In: *The 2025 ACM International Conference on Supporting Group Work*. GROUP ’25: The 2025 ACM International Conference on Supporting Group Work. Hilton Head South Carolina USA: ACM, 12 Jan. 2025, pp. 90–93. ISBN: 9798400711879. DOI: 10.1145/3688828.3699647. URL: https://dl.acm.org/doi/10.1145/3688828.3699647 (visited on 01/02/2025).

Ahrens, Sönke. *How to take smart notes: one simple technique to boost writing, learning and thinking*. 2nd edition, revised and expanded edition. Hamburg, Germany: Sönke Ahrens, 2022. 177 pp. ISBN: 978-3-9824388-0-1.

Angelov, Dimo and Diana Inkpen. “Topic Modeling: Contextual Token Embeddings Are All You Need”. In: *Findings of the Association for Computational Linguistics: EMNLP 2024*. Ed. by Yaser Al-Onaizan, Mohit Bansal and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 13528–13539. DOI: 10.18653/v1/2024.findings-emnlp.790. URL: https://aclanthology.org/2024.findings-emnlp.790/.

Asimov, Isaac. *Isaac Asimov Asks, “How Do People Get New Ideas?”* MIT Technology Review. 2014. URL: https://www.technologyreview.com/2014/10/20/169899/isaac-asimov-asks-how-do-people-get-new-ideas/ (visited on 13/05/2024).

Atlassian. *User Stories | Examples and Template*. Atlassian. URL: https://www.atlassian.com/agile/project-management/user-stories (visited on 09/06/2025).

Bhandari, Abhiyan. “Revolutionizing Radiology With Artificial Intelligence.” In: *Cureus* 16.10 (Oct. 2024), e72646. ISSN: 2168-8184. DOI: 10.7759/cureus.72646.

Blei, David M., Andrew Y. Ng and Michael I. Jordan. “Latent Dirichlet Allocation”. In: *Journal of Machine Learning Research* 3.4 (Jan. 2003). Ed. by John Lafferty, pp. 993–1022. DOI: 10.1162/jmlr.2003.3.4-5.993.

Bloom, Benjamin. *Taxonomy of educational objectives: the classification of educational goals*. New York, Longmans, Green, 1956. URL: https://archive.org/details/taxonomyofeducat0000bloo o9o7 (visited on 04/04/2025).

Bohannon, Molly. *Lawyer Used ChatGPT In Court—And Cited Fake Cases. A Judge Is Considering Sanctions*. Forbes. 8 June 2023. URL: https://www.forbes.com/sites/mollybohannon/2023/06/08/lawyer-used-chatgpt-in-court-and-cited-fake-cases-a-judge-is-considering-sanctions/ (visited on 26/06/2025).

BouJaoude, Saouma and May Attieh. "The Effect of Using Concept Maps as Study Tools on Achievement in Chemistry". In: *EURASIA Journal of Mathematics, Science and Technology Education* 4.3 (22 Oct. 2008). ISSN: 13058223. DOI: 10.12973/ejmste/75345. URL: https://www.ejmste.com/article/the-effect-of-using-concept-mapsas-study-tools-on-achievement-inchemistry-4111 (visited on 23/12/2024).

Brinthaupt, Thomas M. and Alain Morin. "Self-talk: research challenges and opportunities". In: *Frontiers in Psychology* 14 (3 July 2023). ISSN: 1664-1078. DOI: 10.3389/fpsyg.2023.1210960. URL: https://www.frontiersin.org/journals/psychology/articles/10.3389/fpsyg.2023.1210960/full (visited on 12/07/2025).

Brown, Peter F., Jennifer C. Lai and Robert L. Mercer. "Aligning sentences in parallel corpora". In: *Proceedings of the 29th annual meeting on Association for Computational Linguistics -*. the 29th annual meeting. Berkeley, California: Association for Computational Linguistics, 1991, pp. 169–176. DOI: 10.3115/981344.981366. URL: http://portal.acm.org/citation.cfm?doid=981344.981366 (visited on 12/04/2025).

Callon, Michel. "The Sociology of an Actor-Network: The Case of the Electric Vehicle". In: *Mapping the Dynamics of Science and Technology*. Ed. by Michel Callon, John Law and Arie Rip. London: Palgrave Macmillan UK, 1986, pp. 19–34. ISBN: 978-1-349-07410-5 978-1-349-07408-2. DOI: 10.1007/978-1-349-07408-2_2. URL: http://link.springer.com/10.1007/978-1-349-07408-2_2 (visited on 08/04/2024).

Carr, Nicholas G. *The shallows: what the Internet is doing to our brains*. New York, NY: W.W. Norton & Company, 2020. 294 pp. ISBN: 9780393357820.

Carter, Shan and Michael Nielsen. "Using Artificial Intelligence to Augment Human Intelligence". In: *Distill* 2.12 (4 Dec. 2017), e9. ISSN: 2476-0757. DOI: 10.23915/distill.00009. URL: https://distill.pub/2017/aia (visited on 28/03/2025).

Cataloging, Classification, Information Science, PKMs and YOU! - Knowledge management. Obsidian Forum. 16 Dec. 2020. URL: https://forum.obsidian.md/t/cataloging-classification-information-science-pkms-and-you/10071 (visited on 29/05/2025).

Cevolini, Alberto. "Niklas Luhmann's Card Index: Thinking Tool, Communication Partner, Publication Machine". In: *Forgetting Machines: Knowledge Management Evolution in Early Modern Europe*. BRILL, 11 Oct. 2016. ISBN: 978-90-04-32525-8 978-90-04-27846-2. DOI: 10.1163/9789004325258. URL: https://brill.com/view/title/26377 (visited on 30/01/2024).

– "Where Does Niklas Luhmann's Card Index Come From?" In: *Erudition and the Republic of Letters* 3.4 (24 Oct. 2018), pp. 390–420. ISSN: 2405-5050, 2405-5069. DOI: 10.1163/24055069-00304002. URL: https://brill.com/view/journals/erl/3/4/article-p390_390.xml (visited on 23/01/2025).

Cheng, Huaqing et al. "A Neural Topic Modeling Study Integrating SBERT and Data Augmentation". In: *Applied Sciences* 13.7 (5 Apr. 2023), p. 4595. ISSN: 2076-3417. DOI: 10.3390/app13074595. URL: https://www.mdpi.com/2076-3417/13/7/4595 (visited on 06/04/2025).

Chicco, Davide. "Siamese Neural Networks: An Overview". In: *Artificial Neural Networks*. Ed. by Hugh Cartwright. New York, NY: Springer US, 2021, pp. 73–94. ISBN: 978-1-0716-0826-5. DOI: 10.1007/978-1-0716-0826-5_3. URL: https://doi.org/10.1007/978-1-0716-0826-5_3.

Chitrao, Mahesh V. and Ralph Grishman. "Statistical Parsing of Messages". In: *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27,1990*. HLT 1990. 1990. URL: https://aclanthology.org/H90-1053/ (visited on 12/04/2025).

Desrosières, Alain. *The Politics of Large Numbers: A History of Statistical Reasoning*. Cambridge University Press, 2002.

Devlin, Jacob et al. *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. 24 May 2019. arXiv: 1810.04805[cs]. URL: http://arxiv.org/abs/1810.04805jj (visited on 07/08/2024).

Einstein, Albert. *Ideas and Opinions*. New York, USA: Crown Publishers, 1960.

Fodor, Jerry A. and Zenon W. Pylyshyn. "Connectionism and cognitive architecture: A critical analysis". In: *Cognition* 28.1 (1988), pp. 3–71. ISSN: 0010-0277. DOI: https://doi.org/

10.1016/0010-0277(88)90031-5. URL: https://www.sciencedirect.com/science/article/pii/0010027788900315.

Folders. Zettelkasten Forum. 24 May 2024. URL: https://forum.zettelkasten.de/discussion/2897/folders (visited on 29/05/2025).

Forte, Tiago. *The PARA Method: The Simple System for Organizing Your Digital Life in Seconds*. Forte Labs. 24 Feb. 2023. URL: https://fortelabs.com/blog/para/ (visited on 13/04/2025).

Girotra, Karan, Christian Terwiesch and Karl T. Ulrich. "Idea Generation and the Quality of the Best Idea". In: *Management Science* 56.4 (Apr. 2010), pp. 591–605. ISSN: 0025-1909, 1526-5501. DOI: 10.1287/mnsc.1090.1144. URL: https://pubsonline.informs.org/doi/10.1287/mnsc.1090.1144 (visited on 03/01/2025).

Grootendorst, Maarten. "BERTopic: Neural topic modeling with a class-based TF-IDF procedure". In: *arXiv preprint arXiv:2203.05794* (2022).

How do you guys organize Zettelkasten notes? r/ObsidianMD. 6 Mar. 2023. URL: https://www.reddit.com/r/ObsidianMD/comments/11jiein/how_do_you_guys_organize_zettelkasten_notes/ (visited on 29/05/2025).

Hu, Krystal and Krystal Hu. "ChatGPT sets record for fastest-growing user base - analyst note". In: *Reuters* (2 Feb. 2023). URL: https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/ (visited on 26/06/2025).

Iverson, Kenneth E. "Notation as a tool of thought". In: *ACM Turing Award Lectures*. New York, NY, USA: Association for Computing Machinery, 2007, p. 1979. ISBN: 9781450310499. URL: https://doi.org/10.1145/1283920.1283935.

Jex, Tillman. *Floss, A Partial Antidote To Social Catastrophe*. 2023. URL: https://tjex.net/posts/floss-a-partial-antidote-to-social-catastrophe/ (visited on 05/04/2025).

– *Poll: Which AI feature would you prefer for your workflow?* Zettelkasten Forum. 6 Aug. 2024. URL: https://forum.zettelkasten.de/discussion/2959/poll-which-ai-feature-would-you-prefer-for-your-workflow (visited on 09/06/2025).

– *The GRID System for Frictionless Zettelkasten Organisation*. 6 Jan. 2024. URL: https://tjex.net/posts/zettelkasten-grid-system-organisation/ (visited on 29/05/2025).

Jones, Karen Sparck. "Natural Language Processing: A Historical Review". In: *Current Issues in Computational Linguistics: In Honour of Don Walker*. Ed. by Antonio Zampolli, Nicoletta

Calzolari and Martha Palmer. Dordrecht: Springer Netherlands, 1994, pp. 3–16. ISBN: 978-0-585-35958-8. DOI: 10.1007/978-0-585-35958-8_1. URL: https://doi.org/10.1007/978-0-585-35958-8_1.

Kaličanin, K and M Čolović. "Benefits of Artificial Intelligence and Machine Learning in Marketing". In: *Sinteza 2019 - International Scientific Conference on Information Technology and Data Related Research*. 2019, pp. 472–477. DOI: 10.15308/Sinteza-2019-472-477.

Kay, Alan. "User Interface: A Personal View". In: *Multimedia: from Wagner to virtual reality*. Ed. by Randall Packer and Ken Jordan. New York, NY: Norton, 2001. ISBN: 9780393049794.

Krajewski, Markus. *Paper Machines: About Cards & Catalogs, 1548-1929*. The MIT Press, 19 Aug. 2011. ISBN: 9780262298216. DOI: 10.7551/mitpress/9780262015899.001.0001. URL: https://direct.mit.edu/books/book/2133/Paper-MachinesAbout-Cards-amp-Catalogs-1548-1929 (visited on 28/02/2025).

Krishnan, Anusuya. *Exploring the Power of Topic Modeling Techniques in Analyzing Customer Reviews: A Comparative Analysis*. 2023. DOI: 10.48550/ARXIV.2308.11520. URL: https://arxiv.org/abs/2308.11520 (visited on 01/05/2025).

Kurzweil, Ray. *The singularity is near: when humans transcend biology*. New York: Viking, 2005. 652 pp. ISBN: 9780670033843.

Latour, Bruno. *Reassembling the Social: An Introduction to Actor-Network-Theory*. Oxford University PressOxford, 28 July 2005. ISBN: 978-0-19-925604-4 978-1-383-03965-8. DOI: 10.1093/oso/9780199256044.001.0001. URL: https://academic.oup.com/book/52349 (visited on 04/04/2024).

Latour, Bruno et al. "'The whole is always smaller than its parts' – a digital test of Gabriel Tardes' monads". In: *The British Journal of Sociology* 63.4 (Dec. 2012), pp. 590–615. ISSN: 0007-1315, 1468-4446. DOI: 10.1111/j.1468-4446.2012.01428.x. URL: https://onlinelibrary.wiley.com/doi/10.1111/j.1468-4446.2012.01428.x (visited on 23/01/2024).

Le, Quoc V. and Tomas Mikolov. *Distributed Representations of Sentences and Documents*. 22 May 2014. arXiv: 1405.4053[cs]. URL: http://arxiv.org/abs/1405.4053 (visited on 06/08/2024).

Lecun, Y. et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

Lee, Hao-Ping (Hank) et al. "The Impact of Generative AI on Critical Thinking: Self-Reported Reductions in Cognitive Effort and Confidence Effects From a Survey of Knowledge Workers".

In: *Proceedings of the ACM CHI Conference on Human Factors in Computing Systems*. ACM, Apr. 2025. URL: https://www.microsoft.com/en-us/research/publication/the-impact-of-generative-ai-on-critical-thinking-self-reported-reductions-in-cognitive-effort-and-confidence-effects-from-a-survey-of-knowledge-workers/.

Liakos, Konstantinos G. et al. “Machine Learning in Agriculture: A Review”. In: *Sensors* 18.8 (2018). ISSN: 1424-8220. DOI: 10.3390/s18082674. URL: https://www.mdpi.com/1424-8220/18/8/2674.

Liddy, Elizabeth D. “Natural Language Processing”. In: *Encyclopedia of Library and Information Science*. 2nd. New York: Marcel Decker, Inc., 2001.

Luhmann, Niklas. *Communicating with Slip Boxes by Niklas Luhmann*. 1981. URL: https://luhmann.surge.sh/communicating-with-slip-boxes (visited on 18/01/2024).

mattgiaro. *What Folder Structure Should You Use in Your Zettelkasten?* Matt Giaro. 30 May 2022. URL: https://mattgiaro.com/folder-structure-zettelkasten/ (visited on 29/05/2025).

Matuschak, Andy et al. “How can we develop transformative tools for thought?” In: (2019). URL: https://numinous.productions/ttft (visited on 27/03/2025).

Meadows, Donella H. *Thinking in systems: a primer*. Ed. by Diana Wright. White River Junction, Vermont: Chelsea Green Publishing, 2008. 1 p. ISBN: 9781603581486.

Meschede, Friedrich et al., eds. *Serendipity: vom Glück des Findens; Niklas Luhmann, Ulrich Rückriem, Jörg Sasse*. Bielefeld: Kunsthalle Bielefeld, 2015. 286 pp. ISBN: 9783864421495.

Otter, Daniel W., Julian R. Medina and Jugal K. Kalita. “A Survey of the Usages of Deep Learning for Natural Language Processing”. In: *IEEE Transactions on Neural Networks and Learning Systems* 32.2 (2021), pp. 604–624. DOI: 10.1109/TNNLS.2020.2979670.

page, Rhiannon Williamsarchive. *OpenAI has released its first research into how using ChatGPT affects people’s emotional well-being*. MIT Technology Review. 25 Mar. 2025. URL: https://www.technologyreview.com/2025/03/21/1113635/openai-has-released-its-first-research-into-how-using-chatgpt-affects-peoples-emotional-wellbeing/ (visited on 26/06/2025).

Pasupuleti, Vikram et al. “Enhancing Supply Chain Agility and Sustainability through Machine Learning: Optimization Techniques for Logistics and Inventory Management”. In: *Logistics*

8.3 (2024). ISSN: 2305-6290. DOI: 10.3390/logistics8030073. URL: https://www.mdpi.com/2305-6290/8/3/73.

Qiao, Yifan et al. *Understanding the Behaviors of BERT in Ranking*. 2019. DOI: 10.48550/ARXIV.1904.07531. URL: https://arxiv.org/abs/1904.07531 (visited on 26/04/2025).

Raskin, Victor. *Semantic Mechanisms of Humor*. Dordrecht: Springer Netherlands, 1984. ISBN: 9789400964747 9789400964723. DOI: 10.1007/978-94-009-6472-3. URL: http://link.springer.com/10.1007/978-94-009-6472-3 (visited on 11/04/2025).

Reimers, Nils and Iryna Gurevych. *Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks*. 27 Aug. 2019. DOI: 10.48550/arXiv.1908.10084. arXiv: 1908.10084[cs]. URL: http://arxiv.org/abs/1908.10084 (visited on 18/09/2024).

Ritzer, G. *Encyclopedia of social theory*. SAGE Publications, 2004. ISBN: 978-1-4522-6546-9. URL: https://books.google.com.au/books?id=mTZ1AwAAQBAJ.

Sarkar, Advait. “AI Should Challenge, Not Obey”. In: *Communications of the ACM* 67.10 (Oct. 2024), pp. 18–21. ISSN: 0001-0782, 1557-7317. DOI: 10.1145/3649404. URL: https://dl.acm.org/doi/10.1145/3649404 (visited on 21/03/2025).

Schmidt, Johannes F.K. “Niklas Luhmann’s Card Index: The Fabrication of Serendipity”. In: *Sociologica* Vol 12 (26 July 2018), 53–60 Pages. DOI: 10.6092/ISSN.1971-8853/8350. URL: https://sociologica.unibo.it/article/view/8350 (visited on 23/01/2025).

Sparrow, Betsy, Jenny Liu and Daniel M. Wegner. “Google Effects on Memory: Cognitive Consequences of Having Information at Our Fingertips”. In: *Science* 333.6043 (2011), pp. 776–778. DOI: 10.1126/science.1207745. URL: https://www.science.org/doi/abs/10.1126/science.1207745.

thoresson. *Johnny Decimal – A system to organise projects*. Zettelkasten Forum. 15 Dec. 2020. URL: https://forum.zettelkasten.de/discussion/1554/johnny-decimal-a-system-to-organise-projects (visited on 29/05/2025).

Uni Wolverhampton et al. “Semantic Textual Similarity with Siamese Neural Networks”. In: *Proceedings - Natural Language Processing in a Deep Learning World*. Recent Advances in Natural Language Processing. Incoma Ltd., Shoumen, Bulgaria, 22 Oct. 2019, pp. 1004–1011. ISBN: 9789544520564. DOI: 10.26615/978-954-452-056-4_116. URL: https://acl-bg.org/proceedings/2019/RANLP%202019/pdf/RANLP116.pdf (visited on 06/04/2025).

Vaswani, Ashish et al. *Attention Is All You Need*. 2017. DOI: 10.48550/ARXIV.1706.03762. URL: https://arxiv.org/abs/1706.03762 (visited on 05/04/2025).

Wanderloots. *How I Use NotebookLM With Obsidian Practical Note-Taking + AI* . 20 Mar. 2025. URL: `https://www.youtube.com/watch?v=STIIO_qUyJs` (visited on 06/06/2025).

Wang, Bin and C.-C. Jay Kuo. *SBERT-WK: A Sentence Embedding Method by Dissecting BERT-based Word Models*. 1 June 2020. DOI: `10.48550/arXiv.2002.06652`. arXiv: `2002.06652`. URL: `http://arxiv.org/abs/2002.06652` (visited on 26/04/2025).

List of Figures

Appendix A: Zettelkasten Notes

Zombie-Ant Fungus Note

```
# zombie-ant fungus

Zombie-ant fungus (*Ophiocordyceps unilateralis*) was discovered in 1859 by
Alfred Russel Wallace and is an insect-pathogenic fungus. The fungus infects
a particular ant tribe named Camponotini, leading it to leave its canopy nest
and foraging trails for the forrest floor. It directs the ant to a major vein on
the underside of a leaf where the ant uses its mandibles to attach itself,
where it remains until its death. Upon its death, a small mushroom sprouts from
head of the ant.

I wonder if I have some Camponotini in my evergreen forrest? What would they
look like? What would the fungus be?
```

Vampire Bat in Denim Note

```
# vampire bat in denim

Looking diagonally, front-on at a human size (or I was bat size) vampire bat.
It had full beaded black eyes and was wearing a denim jacket, which was buttoned
tight against its body. While looking dead ahead, it opened its mouth to expose
sharp teeth and shot out it's tongue, which itself had posable claws on it,

The tongue claw snapped mid air, in search for something.

Was the bat [name redacted]? I think it was his denim jacket. I don't really know
anyone else who wears one.
```

Declaration of Authorship

I, Tillman Jex Schäuble, declare that this thesis titled, *In Search of Serendipity:: Applying Actor-Network Theory to Zettelkasten and How Natural Language Processing Can Help Manage Difficulties Arising from Zettelkasten's Bottom-Up Structure*, and the work presented in it are my own, and confirm that:

- This thesis was done wholly while in candidature for the Master of Creative Technologies degree at Filmuniversität Babelsberg KONRAD WOLF.
- This thesis has not previously been submitted for a degree or any other qualification.
- Where I have quoted from the work of others, the source is always given.
- I used ChatGPT at times as a code debugger and also to assist in the translation of the abstract from English to German.
- I used PerplexityAI to search the internet for research material in conjunction with conventional search engines.
- Apart from the above usages, I did not in any way use AI software for the research or production of this thesis.

Tillman Jex Schäuble,
July 14, 2025, Berlin, Germany.

[illegible]