Computer Assisted Design Reflection: A Web Application to Improve Early Stage Product Design in Startup Companies

by

Clayton C. Gimenez

Submitted to the Department of Mechanical Engineering
in partial fulfillment of the requirements for the degree of

Master of Science in Mechanical Engineering

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

September 2014

Signature redacted

Author . .....
Department of Mechanical Engineering
August 18, 2014

Certified by .. Signature redacted .....
David R. Wallace
Professor of Mechanical Engineering, MacVicar Faculty Fellow
Thesis Supervisor

Signature redacted

Accepted by ....
David E. Hardt
Chairman, Committee on Graduate Students

Computer Assisted Design Reflection: A Web Application to Improve Early Stage Product Design in Startup Companies

by

Clayton C. Gimenez

Submitted to the Department of Mechanical Engineering
on August 18, 2014, in partial fulfillment of the
requirements for the degree of
Master of Science in Mechanical Engineering

Abstract

This thesis investigates the concept of computer-assisted design reflection. The work details the development of a prototype framework and reflection engine. Reflection is a critical process in design. It allows a designer to learn from experience to improve future results. When applied throughout the design process, it can also improve the quality of design decisions as they are made.

We have observed that, unfortunately, design teams in startup environments often do not reflect sufficiently, leading to poor results that can cause a company to fail. A computer-assisted system could prompt reflection regularly and also compile responses from designers and users alike to form a larger dataset for analysis. A web application is uniquely suited to this task because of development simplicity and ease of use for the end user. It is also more prone to adoption by the startup, since many already use a variety of web applications.

Through the application of natural language processing and information retrieval theory, the software presented in this research transforms free text inputs into a series of output data clusters that capture both subtly patterns and discrete segmentations in the dataset. Initial testing indicates effective and actionable results from datasets derived from Amazon product reviews. Further testing and development is proposed, beginning with the assembly of large design reflection datasets, as none currently exist. Such datasets would enable system improvements and open up new avenues of research in computer-assisted design.

Thesis Supervisor: David R. Wallace
Title: Professor of Mechanical Engineering, MacVicar Faculty Fellow

Acknowledgments

Thank you to the National Science Foundation Graduate Research Fellowship, which funded my studies and gave me the freedom to pursue my own projects.

Thank you to Drew Bennett, the best mentor I've ever had, whose guidance and advice, like a fine wine, grows better with age.

Thank you to Leslie Regan and Chevalley Duhart, for solving problems I couldn't on my own.

Thank you to David Wallace, whose unending patience, tolerance, and calmness helped me through hard times, and without whom I would not have succeeded.

And thank you to Mary. You know why.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

Startup ventures have immense potential. Companies like Google, Apple, and others have revolutionized technology and radically changed our lives. However, countless others have failed. The reasons for failure are numerous, and some are even unavoidable [1]. However, one of the most easily addressed possibilities is a failure of design.

For example, a 20 year study (1960s-1980s) by Bruno et al. examined the failure of new ventures. They found that product-related issues were commonly cited failure modes, and product design and development problems were the most impactful of product-related issues. Even more, these issues become more prominent as time went on, with venture capital and other effective funding sources becoming more common and better understood [1, 2]. This suggests that product issues have only become more prevalent with time.

However, a plethora of research details product design processes and means to avoid design failures. Many university classes are offered on product design in various disciplines such as business development and engineering. Highly structured methods such as presented in Ulrich and Eppinger's *Product Design and Development* are the most common in such courses. Structured methods offer a number of benefits, such as making the design decision process explicit, reminding the team of important tasks,

and being self-documenting. On the other hand, they require modification for a given project, team, and company, and how to make the necessary modifications without losing critical elements is often challenging [32].

Additionally, a large body of work on new product development can be applied to startups. For instance, Veryzer investigates the design and development of discontinuous new products, which are radically different from existing products, in established companies [34]. Others explore means to abbreviate the fully structured design and project management processes in order to accelerate new product development [31, 7].

However, few startups use the available academic work. For example, in 2012, Marion et al. found that product design in early-stage firms was rather chaotic, with little project management or a structured design process [16]. On the other hand, the importance of design is generally recognized, both in startups and in academia [4]. Why then are design research and design methodologies not more commonly applied in new ventures? Unfortunately, no significant work on that topic is available.

Thus, to answer that question, study of and interaction with startups and their people was necessary. Through informal interviews and observations over approximately six months at Boston-area startups, some broad insights were developed by the author:

- Most startups do not follow any particular product design and development process. Many have no formal process, and some have no formal project management or design personnel.
- Startups, especially in software, tend to emphasize development productivity and shipping content rapidly. Common management methods, such as sprints and scrum meetings, reflect this, and often sacrifice quality for speed.
- Determining why and how a poor design decision was made only happens in catastrophic situations. More minor problems are simply fixed as possible before teams return to new features.

If possible, prompting and assisting in design reflection could improve these issues. Design reflection involves the candid and thoughtful assessment of such elements as

the needs and values of one's users, one's own needs and values as a designer, and the resulting design decisions. However, that assistance would need to fit into the environment and values of the typical hectic startup.

Working with a few particularly interested lead users, core product requirements were developed for a design reflection tool. The focus of the design reflection tool would be to subtly prompt individuals to slow down and think and improve awareness of design issues.

- The tool should be web-based, like many of their current tools, to facilitate adoption.

- The tool should not make design decisions, only reveal information, such as subtle trends in a large body of data, that could easily slip by the team otherwise.

- The tool should prompt input from all employees so as to not miss out on valuable information from a quieter team member.

Startup employees could log in to the tool and enter design notes or thoughts throughout the day, with the tool acting as a notebook of sorts. The tool would process these varied inputs into preliminary reflections. Design leaders would review the results to gain information from which design and project management decisions could be made. For example, a number of customer support employees noting that Feature A is problematic for users and in need of improvement could prompt project leaders to reevaluate the user experience and assign more development time and talent to Feature A.

The key element in the tool is undoubtedly the reflection engine, which turns user input into actionable information from which design decisions can be made. The remainder of this thesis will discuss the design and initial development of that engine and its supporting framework.

1.2 Thesis Outline

Chapter 1 has provided introductory background information and discussed the primary goal of the research.

Chapter 2 reviews selected prior work in the areas of design reflection, natural language processing, and information retrieval.

Chapter 3 describes the developed application itself, both its modular, expandable framework and the prototype reflection engine.

Chapter 4 evaluates initial results, validating the reflection engine and prompting its use for further research.

Chapter 5 summarizes the findings and provides guidance for future work.

Chapter 2

Background

2.1 Design Reflection

Design reflection is a relatively young area of interest in design research, with the earliest notable works appearing in the 1980s by Donald Schön [27]. Since then, a number of studies have indicated the usefulness of reflection in design. For instance, through reflection, designers can identify problem areas in their processes and devise solutions [19]. At the most basic level, the overarching goal of design reflection, as described by Reyman, is to influence future design activities [22].

As initially defined by Schön, there are three primary types of design reflection. The three types may occur independently or in combination, and all three can have significant positive impact:

1. ***Reflection-in-action*** is thinking about doing while doing. The goal is to create awareness of current action in order to influence future action. It primarily occurs when the designers are surprised, for example by customer feedback or a conflict between designers [29, 33].

2. ***Reflection-on-action*** is thinking about doing after doing. The goal is to evaluate past situations in order to influence future situations. It primarily occurs at the ends of project phases or when a design team gets stuck [10, 21].

3. ***Reflection-on-practice*** is thinking about doing after multiple doings. The goal is to find patterns of action in order to improve future practices. It tends to be a more involved and time-consuming process that occurs after several projects in order to restructure team approaches or methods [26].

The differences between the three types are summarized in Table 2.1.

Table 2.1: The three types of design reflection as described by Reymen [22]

Type of design reflection	*In action*	*On action*	*On practice*
Level	micro-level design process dynamics	macro-level design process dynamics	design project level dynamics
Focus of reflection	awareness of design cycle activities	evaluation of (critical) situations in design process	discovery of patterns in design projects
Most related to	designing	⟷	design management

In the end, all three types of reflection are useful in a startup environment. All three can be addressed with a computer-assisted process, depending on the means of prompting designers and users for reflective input. For instance, user reactions and designer thoughts on a current project or feature development could be used to prompt reflection-in-action, whereas an overview of designer thoughts on development results and management techniques could prompt reflection-on-practice. Regardless of the type, reflection allows designers to learn from experience and to more effectively solve problems, reducing failure and improving long-term results [9, 10, 11]. Additionally, it enables design managers to improve teamwork and achieve superior results, independent of particular team members [33, 35].

2.2 Natural Language Processing

Natural language processing (NLP) is a field of computer science, artificial intelligence, and linguistics that seeks to enable computers to process and understand human languages. It is a large and diverse field of study, with work on tasks ranging from machine translation to recommender systems and automated question answering [6].

The primary use of natural language processing in this thesis is to transform raw input text into a parsed and annotated body for further processing. This initial processing unlocks a wide variety of advanced processing techniques to better understand the meaning in a text.

A number of different implementations of these basic NLP processing tasks exist. The Stanford Natural Language Processing Group maintains and improves one commonly used NLP software package, known as StanfordCoreNLP. This was the primary package used in this thesis for NLP tasks [14].

A body of text is first segmented and tokenized. This process splits a text into a series of tokens, such as words and punctuation elements. This is a typical preprocessing step in NLP. A number of different segmentation and tokenization methods exist. StanfordCoreNLP applies a modified Penn Treebank 3-based method that is fast, computationally efficient, and deterministic [15].

Parsing and part-of-speech tagging operate on a tokenized text to assemble syntactical structures and determine parts of speech of words. This allows a computer to better understand the text and later processing elements to examine different levels of meaning in the text. A fully parsed and tagged body of text is the typically starting point for most advanced NLP techniques. The Stanford Parser is a probabilistic parser, which uses knowledge of language gained from a large set of human-parsed text to produce the most likely analysis of processed text. The Stanford Parser has been developed and improved over the course of many years and is quite effective [8, 30].

Other NLP techniques that could be applied to a design reflection engine include the application of cognitive synonyms and common sense reasoning. Wordnet, developed at Princeton University, groups English words into groups of cognitive synonyms known as synsets. Each synset represents a particular concept and relations between concepts are captured through relationships. This allows a computer to equate synonyms that might be used by different people for the same meaning or to extract the greater meaning of sections of text [17].

ConceptNet, developed by MIT's Media Lab, seeks to capture human common

sense in a large database. This allows a computer to draw connections that are not necessarily directly conveyed by dictionary definitions or to extrapolate additional meaning from a text. It can help interpret hidden or implied meaning in a text [12].

2.3 Information Retrieval

Information retrieval is the activity of obtaining information resources relevant to an information need from a collection of information resources. It provides the underpinnings of such ubiquitous systems as Internet search engines and library record systems. Due to the modern commercial impact of the field, information retrieval is extensively explored and well researched. Information and document clustering was the primary application of information retrieval theory in this thesis. The text *Introduction to Information Retrieval* by Manning et al. provides a detailed introduction to the applied topics [13].

To cluster documents, they are often first reduced to a vector space model such that clustering algorithms may be applied, as in Cornell's SMART information retrieval system [24]. A vector space model represents each document as a vector, with each value in the vector typically referring to a given term, or word, in the dictionary of all terms in all documents of concern. Each term must therefore be given a weight in each document. A number of different weighting schemes exist, though one of the more common is term frequency-inverse document frequency, which has been examined and justified in numerous works [25, 23, 20].

Clustering algorithms are then applied to these vector models of documents to assemble groups of documents with similar meaning. Two main archetypes of clustering are flat and hierarchical. Flat clustering creates a set of clusters that are not explicitly related to one another, while hierarchical clustering creates a hierarchy of clusters that can shed light on how clusters are related and broken down into sub-clusters internally [5]. One common flat clustering method is K means, which sorts documents into K clusters built as averages around seeds [3]. A common hierarchical clustering method is group-average agglomerative clustering, in which the average similarity of

notes within clusters is compared to successively merge clusters in a deterministic manner [28].

Chapter 3

Computer Assisted Reflection Prototype

The development of the initial computer assisted design reflection system focused on two elements: a framework that can be deployed as a web application and a reflection insight to process inputs into preliminary reflections that prompt further thought.

Unfortunately, the lack of existing bodies of design reflection data limits the initial development to a prototype system. As a result, the focus of development was to make the system as modular and easily modifiable as possible.

3.1 Framework

The goal of the system architecture was to provide a highly modular framework upon which future development and research interests could build. Modularity keeps elements separable and facilitates future expansion.

The use of industry standard languages and packages maximizes the availability of resources and simplifies deployment for testing and experimentation with outside groups, such as startup companies. Building the system as a web application from the outset also simplifies future testing and data acquisition.

Figure 3-1: The flow of information through the system architecture

3.1.1 Back-end Architecture

The system back end was implemented in Ruby on Rails with a PostgreSQL database. This common and widely used platform simplifies deployment for experimentation with users.

The tasks of the back end are:

- Store information about users, projects, and design notes in the database and retrieve them as necessary. Beyond its obvious necessity, this addresses the important design goal of remembering all the details that tend to fade from human memory.

- Preprocess design notes as they are input to allow for faster processing. The initial natural language processing steps are dependent only upon the note itself. Additionally, those steps are the most time consuming. Preprocessing improves the performance of the user-facing front end with no appreciable tradeoff.

- Analyze design notes to generate design reflections. This reflection engine attempts to extract actionable design information from a large body of input notes. Processing in the back end takes advantage of the increased processing power of a server and reduces the volume of data that must be transmitted to the front-end. In short, changes in the reflection engine are kept isolated from the front-end and visualization elements.

- Interface with the front end and transmit needed information through an API.

3.1.2 Front-end Architecture

The front end was implemented with Backbone.js, a JavaScript library that allows for the dynamic rendering of views with a single page app structure. Backbone.js simplifies the implementation of a compelling user experience and allows for more efficient modifications. The dynamic nature of Backbone supports multiple reflection visualizations on a page with minimal development. It also adds tolerance for processing delays if necessary, allowing the rendering of supporting information and initial results before the fully processed reflections are completed. If processing delays prove problematic, the reflection engine could be made to run asynchronously to page load.

The tasks of the front end are:

- Accept processed information from the back end through an API. This isolates the data itself from how it is displayed and presented to the user. Changes to the presentation do not interact with or modify the action of the processors.
- Send new data to be saved, such as design notes, to the back end through the API.
- Provide a page structure and mount points for design reflection visualizations.

3.1.3 Visualization

The visualization engine was built with D3.js, a widely used JavaScript library developed for visualization creation for the New York Times. It provides a powerful and diverse set of tools to generate a wide variety of visualizations.

The tasks of the visualization engine are:

- Graphically represent the design reflection results in a variety of ways to prompt further reflection in the end user.

Though only a limited set of visualizations was implemented initially, the freedom provided by D3.js will allow the straightforward addition of alternative visualizations

or new visualizations for different sets of results the reflection engine might provide. Additionally, a single loaded dataset can be used to generate multiple visualizations, allowing for easier experimentation by looking at results in different ways in order to obtain a different understanding.

3.2 Reflection Engine

The reflection engine is the heart of the system. It takes a large collection of free text, natural language "notes" and derives information from them that gives the user insight into the meaning captured in the dataset. Ideally, this information is actionable, leading into improved design decisions.

3.2.1 Text parsing

The first component of the reflection engine is the text parser. It breaks the free text input down into a format a computer can work with. It allows us to extract words, sentences, nouns, stems of words, and the like very quickly and easily in processing later. Successive operations break the text down into a fully processed form:

1. The chunker breaks a body of text into paragraphs.
2. The segmenter breaks a paragraph into sentences.
3. The tokenizer separates sentences into tokens
4. The parser categorizes tokens according to part of speech and similar characteristics.

A simple rule based chunker suffices to break a body into paragraphs at line breaks. The remaining functions utilize the Stanford parser and similar tools available in the StanfordCoreNLP Java library. Ruby bindings for the Java library are available through the Ruby gem Treat by Louis Mullie [18].

Since the parsing results depend only on the note being parsed, each note is pre-parsed when it is saved. Parsing produces a hash of Ruby objects, representing

sentences, words, punctuation, etc. A simpler array of strings containing all of the words in the note is extracted. Both are then stored in the database as serialized JSON for later use. This prevents needless reprocessing later on and reduces the time to render results views to the user.

3.2.2 Vector Space Model

In order to extract useful information from a body of notes, the parsed results must be converted into a form for rapid mathematical operations. One form commonly used in information retrieval is a vector space model, in which each note is represented by a vector. Each element of the vector is a number representing the significance of some aspect of a particular note.

One commonly used approach interprets each note as a "bag of words." A bag of words model views each body of text as a simple collection of terms, with no concern for order or context. Bag of words models are typically sufficient for large datasets since the words and their frequencies in a given note are typically a good representation of its meaning. With this model, the vector representing each note will be of length N, where N is the number of unique terms present in the body of notes.

There are many methods to determine the significance of each term in a given note. Term frequency-inverse document frequency (tf-idf) was chosen for this work. A common weighting scheme, it is a solid starting point for information retrieval tasks if the data is likely to be varied.

Tf-idf weights each term of a note with a composite score of term frequency and inverse document frequency. Term frequency (tf) represents the frequency of the given term in the given note, while inverse document frequency (idf) represents the ratio of all notes in which the given term appears. The resulting weight is highest when a term appears many times in a few documents and lowest when a term appears in all documents.

For this application, maximum term frequency normalization is used. Rather than using a simple count of term occurrences, all tf values for a note are normalized by

the maximum tf in the note using Equation 3.1:

$$ntf_{t,n} = 0.5 + \frac{0.5 \times tf_{t,n}}{max_t(tf_{t,n})} \tag{3.1}$$

where $ntf_{t,n}$ is the normalized term frequency for a given term t and note n. $tf_{t,n}$ is the raw frequency count of t in n.

Maximum tf normalization mitigates the impact of note length on term weightings, preventing longer notes from being arbitrarily more impactful. On the other hand, an unusual but meaningless term that occurs very frequently in a document can skew results. Full example design data is necessary to properly select the ideal weighting scheme, however.

Inverse document frequency is defined in Equation 3.2:

$$idf_t = log\frac{N}{df_t} \tag{3.2}$$

where N is the total number of notes and df_t is the number of notes in which term t appears at least once. The logarithm base does not affect relative scores, though typically base 10 is used.

The last step is to normalize all note vectors to unit length for later computation and comparison. Cosine normalization was used, multiplying each component of the note vector $\mathbf{n}$ by the factor in Equation 3.3:

$$CN_{\mathbf{n}} = \frac{1}{\sqrt{n_1^2 + n_2^2 + \cdots + n_M^2}} \tag{3.3}$$

where M is the total number of terms.

Thus Equation 3.4 yields the overall score for a given term in a given note:

$$tfidf_{t,n} = tf_{t,n} \times idf_t \times CN_{\mathbf{n}} \tag{3.4}$$

3.2.3 Clustering

Clustering algorithms create groups of notes that are internally similar but as different as possible from each other. Clustering allows a computer to examine a large set of data and draw out subtle common trends and demonstrate major segmented differences. It mirrors the common human behavior in reflection of pattern recognition and grouping. However, computational clustering has the advantage of relying purely on the distribution and makeup of the data, eliminating any human bias and forgetfulness.

Regardless of the clustering algorithm, a distance measure is needed. The distance measure compares two of the items to be clustered and determines how similar they are to one another. Different distance measures can lead to substantially different clustering. The ideal choice is typically determined from empirical testing and iterative improvement. Given the lack of design data, Euclidean distance provides a solid starting point for comparing note vectors. The familiar distance measure used in many fields, Euclidean distance between two vectors $\vec{x}$ and $\vec{y}$ is defined in Equation 3.5:

$$|\vec{x} - \vec{y}| = \sqrt{\sum_{i=1}^{M}(x_i - y_i)^2} \tag{3.5}$$

The two main classes of clustering algorithms are flat and hierarchical. Flat clustering creates a set of cluster without any structure relating clusters to one another. Hierarchical clustering creates a hierarchy of clusters, with the structure capturing the relationships between clusters and even individual notes. Flat clustering is typically more computationally efficient, with the most common algorithms having linear complexity. However, it is prone to settle into local maxima of clustering quality. On the other hand, hierarchical clustering is deterministic at the cost of at least quadratic complexity.

Flat clustering algorithms typically require a predetermined number of initial seeds to characterize the clusters. These initial seeds are critical, determining whether the algorithm settles to local or global optimums. For unpredictable design reflection

data, correctly determining seeds is challenging. Initial testing with flat clustering suggested the results were highly prone to settling in a wide variety of local optimums.

Hierarchical clustering, on the other hand, does not require initial seeding. As a deterministic method, it will always settle to the global optimum. Though it is more computationally complex, the relatively small (less than one million) note count does not make processing time prohibitively high. Thus, hierarchical clustering was selected for this application. In the future, a partial hierarchical clustering could be performed to select effective initial seeds for a more efficient flat clustering.

Hierarchical agglomerative clustering (HAC) builds the cluster hierarchy by initially treating each individual note as its own one item cluster. The algorithm then successively merges the two most similar clusters until there is only a single cluster of all notes. To display a set of clusters, the algorithm can be cut short when a specified number of clusters is reached. Alternatively, examining the differences in cluster similarity for each merge can reveal breakpoints in the data that indicate natural clustering. In this application, the algorithm was made to produce a manageable seven clusters from the data.

There are numerous possible similarity measures to compare full clusters. These similarity measures utilize the distance measure between notes in different ways in order to compare full clusters. The measure selected was group-average agglomerative clustering (GAAC), which compares clusters based on all similarities between all notes in each cluster. It avoids many issues with single note comparison methods, such as emphasizing outliers or producing poorly grouped clusters. The GAAC similarity measure is defined in Equation 3.6:

$$SIM(\omega_i, \omega_j) = \frac{1}{(N_i + N_j)(N_i + N_j - 1)} \sum_{d_m \in \omega_i \cup \omega_j} \sum_{d_n \in \omega_i \cup \omega_j, d_n \neq d_m} \vec{d_m} \cdot \vec{d_n} \qquad (3.6)$$

where ω_i is cluster i, N_i is the number of notes in cluster i and the vectors $\vec{d}$ are the vector representations of notes.

Normally, GAAC would have a complexity of $O(N^3)$, where N is the number of

notes. By generating a priority queue, essentially sorting relative cluster similarities initially and only recomputing as necessary, the complexity can be relatively simply reduced to $O(N^2 logN)$. The priority queue algorithm is:

```
EFFICIENTHAC(d⃗_1, ..., d⃗_N)
 1  for n ← 1 to N
 2  do for i ← 1 to N
 3     do C[n][i].sim ← d⃗_n · d⃗_i
 4        C[n][i].index ← i
 5     I[n] ← 1
 6     P[n] ← priority queue for C[n] sorted on sim
 7     P[n].DELETE(C[n][n]) (don't want self-similarities)
 8  A ← []
 9  for k ← 1 to N − 1
10  do k_1 ← arg max_{k:I[k]=1} P[k].MAX().sim
11     k_2 ← P[k_1].MAX().index
12     A.APPEND(⟨k_1, k_2⟩)
13     I[k_2] ← 0
14     P[k_1] ← []
15     for each i with I[i] = 1 ∧ i ≠ k_1
16     do P[i].DELETE(C[i][k_1])
17        P[i].DELETE(C[i][k_2])
18        C[i][k_1].sim ← SIM(i, k_1, k_2)
19        P[i].INSERT(C[i][k_1])
20        C[k_1][i].sim ← SIM(i, k_1, k_2)
21        P[k_1].INSERT(C[k_1][i])
22  return A
```

Figure 3-2: The priority queue algorithm for HAC, as presented in Manning [13].

In words, the priority queue algorithm:

1. Assembles a $N \times N$ matrix describing the similarity of each cluster to each other cluster. Note that each cluster is exactly one note initially.

2. Sorts each cluster's vector based on similarity and removes self-similarities, since a cluster cannot be merged with itself.

3. Merge the two most similar clusters available for merging and record the merge in the hierarchy. One of the clusters becomes the representative of the merged cluster. The other is marked as unavailable for merging.

4. Update the similarity of all entries affected by the merge in the similarity matrix.

5. Repeat steps 2-4 as many times as necessary to condense to a desired number of clusters.

The end result of the algorithm is a list of cluster merges that fully characterizes the hierarchy.

3.2.4 Cluster Labeling

To display the information captured in the clusters to the user, each cluster must be labeled and displayed. Cluster internal labeling was selected, as opposed to differential cluster labeling, as some factors may be important to multiple clusters. For example, different segments of users or designers may have very different opinions of a particular product feature, which is potentially very valuable and actionable information.

Each cluster is labeled based on its centroid, the average of the vectors of all notes in that cluster. The most significant terms for the cluster are determined from the highest values in the vector. These values are then cross-referenced with the initial term list to convert back into their original strings. Finally, the labels and their relative significance are compiled and sent to the front-end for display.

Chapter 4

Results and Discussion

4.1 Literature Test Case

Ultimately, clustering is a form of unsupervised learning, in which a computer attempts to understand a dataset as a human would. Thus, the results must be compared with human results in order to evaluate the usefulness of the clustering [13].

To begin with a simple test case, ten paragraph-long excerpts from the open domain works of classic authors were selected. The three works – Charles Dickens' *Great Expectations*, Mark Twain's *The Adventures of Huckleberry Finn*, and H.G. Wells' *The War of the Worlds* – come from distinct time periods, genres, and subjects. Clear categorizations are apparent between the works, primarily based on style. The detailed results of the clustering are presented in Table 4.1 and a visualization of the clustering in Figure 4-1.

Table 4.1: Machine Clustering of Literature Excerpts

Cluster	Notes
0	0, 9, 2, 4, 1, 10, 3, 7, 5, 6, 8, 24, 11, 29, 12, 16, 13, 20
1	14, 15
2	17, 25
3	18, 22
4	19, 21
5	23, 27
6	26, 28

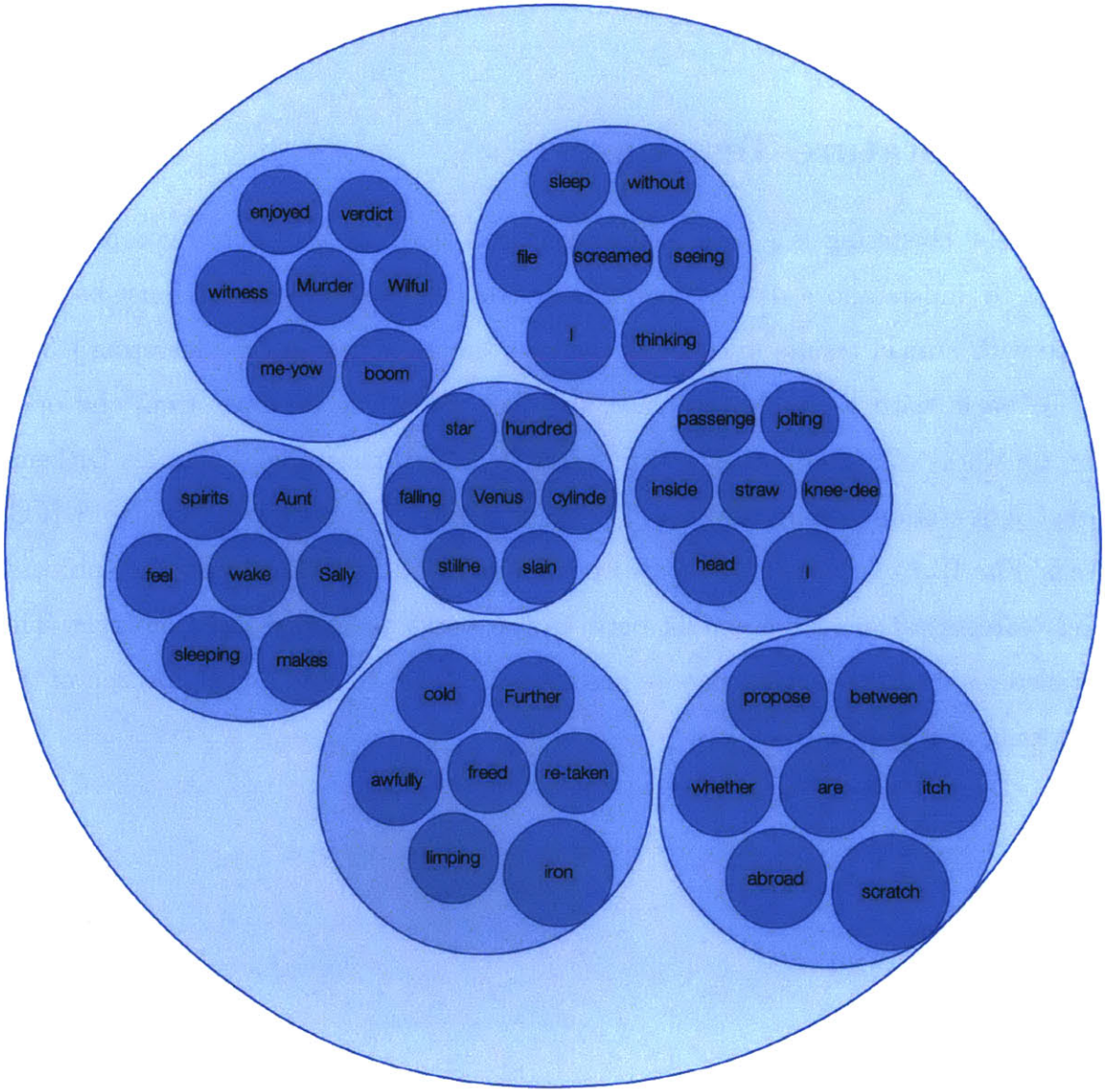

Figure 4-1: Clustering Visualization of Literature Excerpts

It is clear that machine clustering does not generate groups segmented based upon author. If segmentation were based upon author, clusters would be expected to be composed of groupings from notes 0-9 (Wells), 10-19 (Twain), and 20-29 (Dickens). However, reexamination of the excerpts shows that some share common topics across authors. Relationship of topic and the machine clusters is shown in Table 4.2.

Table 4.2: Comparison of Machine Clustering and Human Interpreted Topics of Literature Excerpts

Human Topics	**Machine Topics**
machines, death, cold, hunger, drink, show	Venus, star, hundred, cylinder, falling, stillness, slain
happy, still, sleep	wake, spirits, Aunt, Sally, feel, sleeping, makes
shake, stir	murder, enjoyed, verdict, witness, wilful, me-yow, boom
fear, sneak	propose, are, between, scratch, itch, whether, abroad
travel, hurry	passenger, jolting, straw, inside, knee-deep, head, I
time, fear, file	sleep, without, screamed, file, seeing, I, thinking
cold, file	cold, further, freed, awfully, re-taken, limping, iron

The topic-based results indicate clustering of reasonable quality. This suggests that the prototype reflection engine is functioning. Additionally, this example demonstrates that the system likely differentiates topics with higher priority than writing style, which is preferable in a design reflection system with inputs from many individuals.

4.2 Amazon Product Reviews

Unfortunately, there was no existing large body of design data that was available for testing. However, product reviews can provide some insight. Reviews provide

thoughts on the product from a wide variety of users, though with a higher risk of data noise, such as negative reviews for a unit mishandled in shipping.

Three products with varying numbers of reviews on Amazon.com are presented here:

1. Allen Edmonds Men's Dalton Lace-Up Boot (19 reviews)
2. SteelSeries Sensei Laser Gaming Mouse [RAW] (Rubberized Black) (169 reviews)
3. Microsoft Xbox One + Kinect (700 reviews)

4.2.1 Low Review Count: Allen Edmonds Boot

The comparatively small review count of the Allen Edmonds boot allows for a complete comparison with human clustering. The results are shown in Table 4.3 and Figure 4-2.

The results show that the majority of machine clusters are consistent with human clustering. While particular notes are not necessarily assigned to corresponding clusters, machine and human topics are very similar. This limited set of data suggests that customers love the shoe, particularly given the large single large cluster of predominantly positive reviews. However, some are displeased with the fit or build quality.

Table 4.3: Comparison of Human and Machine Clustering of Allen Edmonds Shoe Reviews

Human Clusters	**Human Topics**
1, 2, 3, 4, 5, 6, 7, 9, 10	The boot is excellent
18, 16, 11, 8	Nice boot, but minor issues with sole liners or quality
0, 12, 13, 15	Major issues with fit or poor quality
14, 17	Nice boot, but runs a bit narrow
Machine Clusters	**Machine Topics**
14, 17	price, to, both, longer, together, meant, width
0, 6, 1, 10, 2, 9, 3	glove, walks, fits, priced, reasonable, classic, excellent
7, 12	certificate, pleased, gift, overall, punch, codes, similar
16, 18	coupled, shaft, tag, socks, protection, orthodics, pull
8, 13	justify, expense, could, chocolate, literally, the, maybe
4, 5	proper, forever, casually, else, a, either, incredible
11, 15	there, no, arch, hope, inserts, lack, star

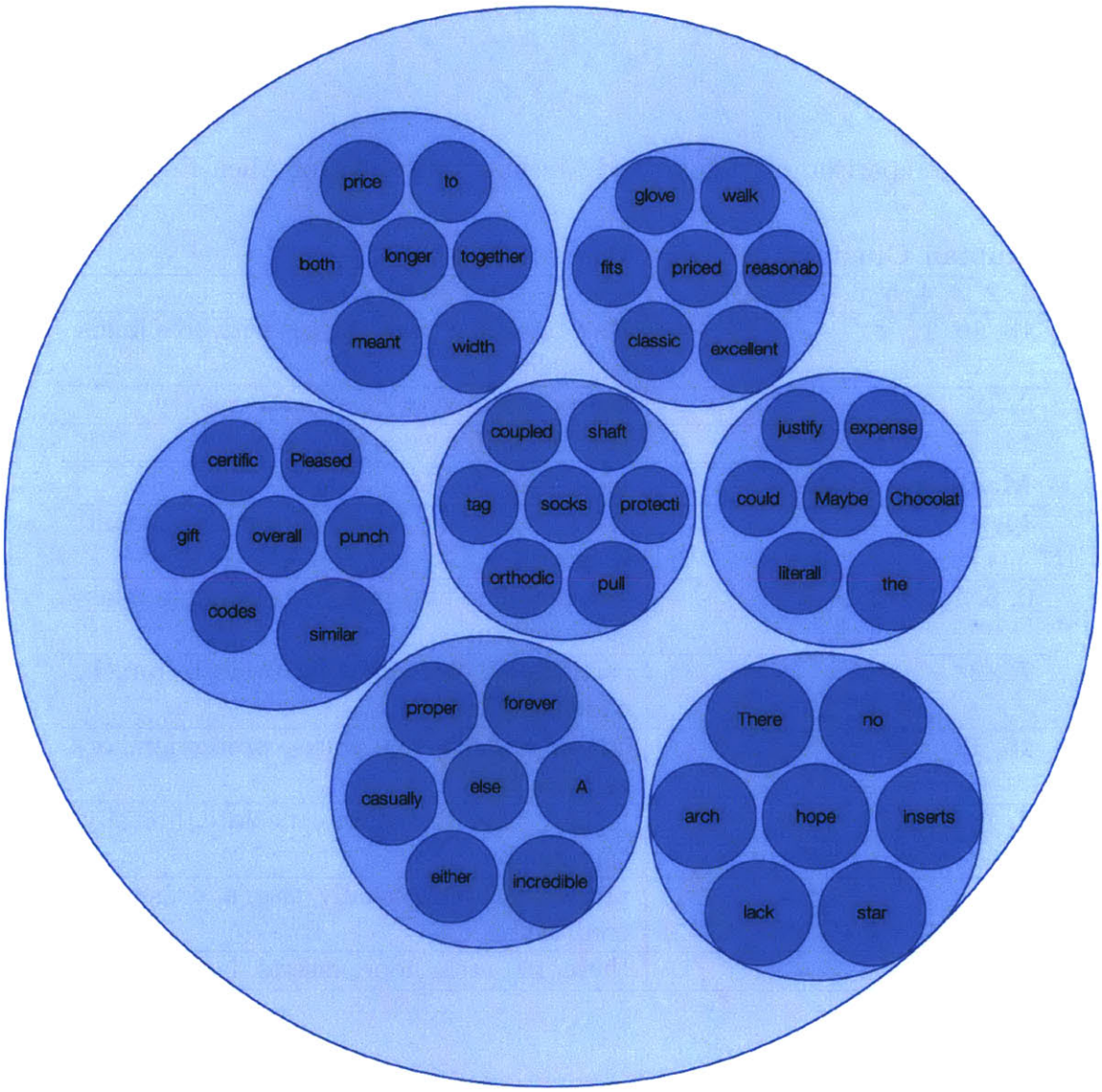

Figure 4-2: Clustering Visualization of Allen Edmonds Shoe Reviews

4.2.2 Medium Review Count: SteelSeries Mouse

The medium review count of the SteelSeries mouse allows for a clustering comparison with human guesses of clustering based upon skimming of the reviews. The results are shown in Table 4.4 and Figure 4-3.

Table 4.4: Machine Clustering of SteelSeries Mouse Reviews

Machine Topics
gotten, functions, slight, adjustments, complaint, certain, options
unplugged, have, insanely, throughout, dead, six, twice
likelihood, index, grip-style, sturdier, alternate, right-most, major
thinkbuy, steelseries, guys, computerlights, yeahmouse, brilliant, she
noticing, rope, sheathing, besides, chances, best, weight
husband, placement, considering, uses, got, excited, he
ditching, fixed, no-frills, performs, rail, wire, connection
Human Clusterings
Negative reviews for units that break quickly
Simple but effective mouse lovers
In-depth reviews from gaming users who perfect every setting
Purchased as a gift, usually for husband or child
People who love the rubberized grip

The results show a high level of agreement between human and machine results. Consumers tend to like the mouse, except for those who receive units that die quickly. The mouse's primary selling points appear to be its grip, simplicity, and performance. This information could prompt the design of more simple but effective mice in the future or prompt an investigation into quality control.

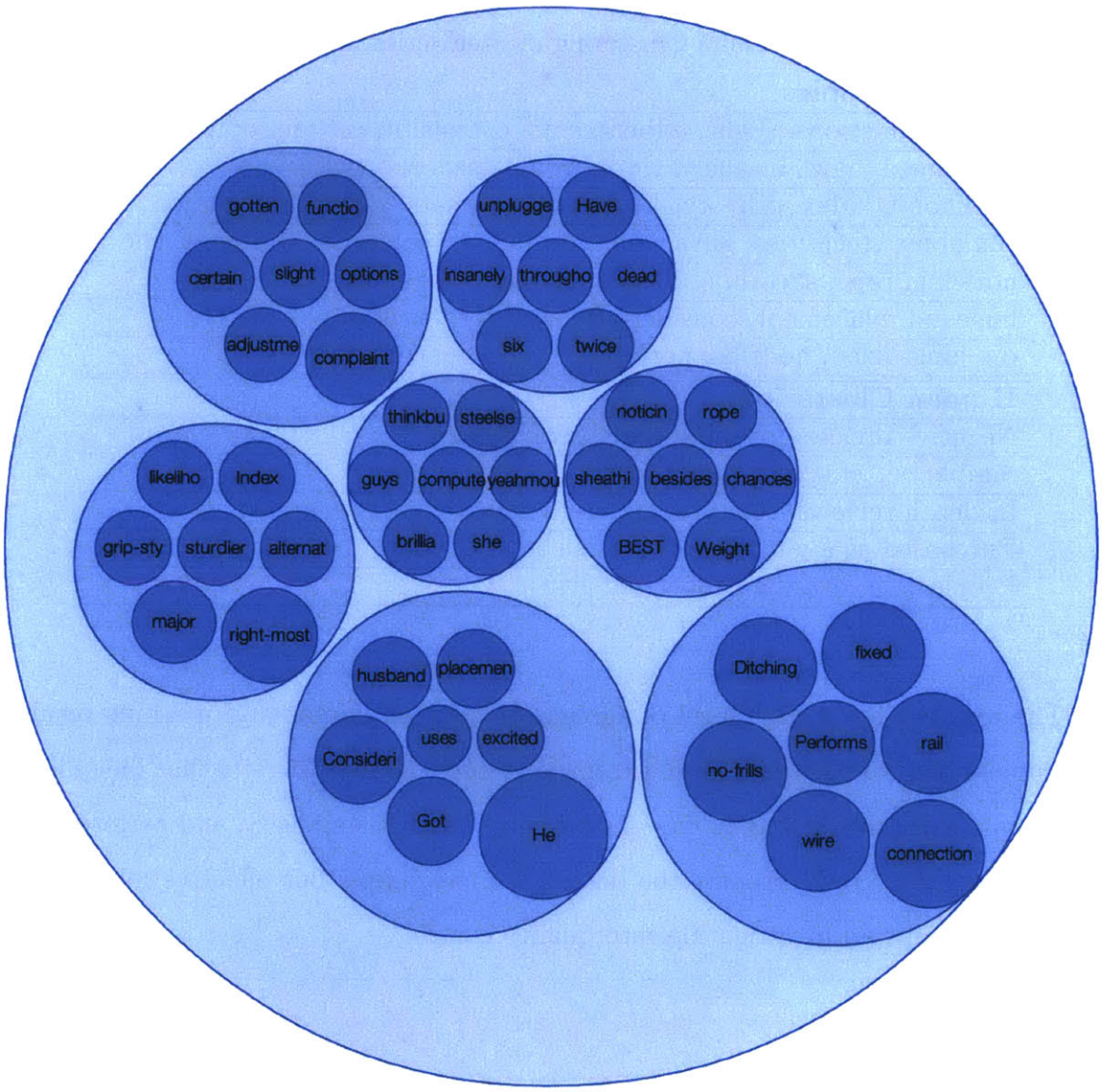

Figure 4-3: Clustering Visualization of SteelSeries Mouse Reviews

4.2.3 Large Review Count: Microsoft Xbox One

The large review count of the Xbox One prevents an effective comparison of human and machine results, as there are simply too many reviews for a human to effectively categorize in a reasonable time. However, it tests whether the reflection engine can draw out useful information from a body of data that might be too large for human processing. The clustering results are shown in Figure 4-4

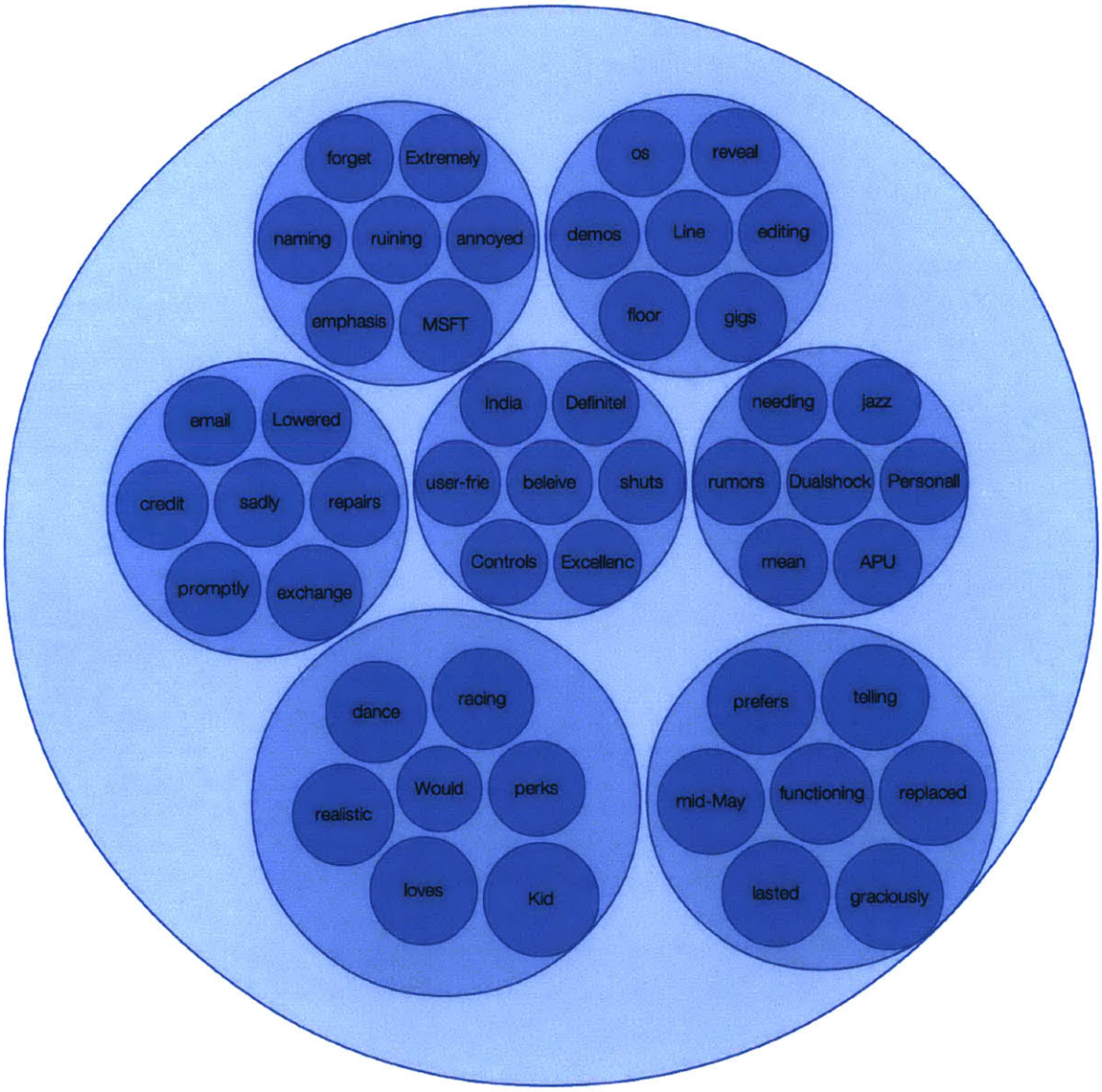

Figure 4-4: Clustering Visualization of Microsoft Xbox One Reviews

The results show that user feelings about the Xbox One tend to be very divided.

Some like it, and some hate it. Many of those who dislike it seem to have issues with repairs or replacements. Another segment of users seem to generally like the Xbox One. The younger segment, or more likely their parents, seems to be the most enthusiastic, especially about realistic racing games and dance games.

Chapter 5

Conclusions

5.1 Summary

A first stage prototype was produced to validate the concept of computer-assisted design reflection. The prototype was built to be robust and highly modular for rapid, iterative development in the future. A Ruby on Rails + PostgreSQL back end provides data storage and server-side processing. A Backbone.js and D3.js front end provide support for highly dynamic and modular views and visualizations. The use of industry standard languages and packages will accelerate deployment and testing in the future.

The prototype reflection engine, built in Ruby, recognizes trends in data sets effectively through the application of natural language processing and information retrieval theory to design data. First, the Stanford parser is applied to break down input notes into their components. Second, the notes are converted to a vector space model with weightings based on term frequency-inverse document frequency. Third, the notes are condensed into clusters via priority queue hierarchical agglomerative clustering using a group-average similarity measure. Last, the clusters are labeled internally by their most significant terms.

The untuned prototype engine was applied to literature excerpts from Charles Dickens, Mark Twain, and H.G. Wells and Amazon.com product reviews for the Allen Edmonds Men's Dalton Lace-Up Boot, the SteelSeries Sensei Laser Gaming Mouse [RAW] (Rubberized Black), and the Microsoft XBox One. The literature

excerpts were effectively clustered based on topic rather than on writing style of the different authors. The Allen Edmonds boot and SteelSeries mouse reviews likewise yielded topic-based clusters very similar to human clustering results. The large review count of the Xbox One rendered human clustering not feasible, but machine clustering produced actionable high level information that could be used to improve the product moving forward.

A lack of prior work in the assembly of public, real-world design datasets limited testing on product development data. However, with data collection and further testing, design reflection could become the next area of impact for computer-assisted design.

5.2 Future Work

Future work in computer-assisted design reflection is primarily dependent on the assembly of large, high quality datasets. Several potential extensions are:

1. General design data collection. In order to evaluate the effectiveness of the reflection engine's processing and any changes to it, a body of design-oriented data is necessary. This dataset would also need to be analyzed by a group of design experts for comparison. Possible sources for this dataset include university design courses and any companies willing to participate. Simpler datasets could also be manually created for first-order testing.

2. Reflection engine refinement. Given a human-analyzed design dataset, the reflection engine can be evaluated and refined. For example, clustering constants and term weighting schemes can be tuned to the application, stop words or part of speech selection can be implemented, demographic or sub-team data can be correlated with clusters, or the shift in clusters over time with new design data can be evaluated.

3. Full product build out. Fully developing all aspects of the product, such as superior input methods, data security, and full live deployment, would allow for

new levels of testing, especially with potential users in startups.

4. Long term testing with design teams. Long term testing offers unique possibilities to improve the reflection engine through user collaboration and larger scale datasets. It also allows for analysis of a much wider variety of datasets and opens up the potential to integrate the system with other design and project management tools.

Appendix A

Code Excerpts

A.1 Vector Space Model Assembly

```
1
2  def tfidf_matrix(terms)
3    # Initialize data structures and term list
4    # terms = self.term_list
5    tfidf = Array.new(self.notes.count) { Array.new(terms.length, 0) }
6
7    # Assemble frequency matrix
8    self.notes.order(created_at: :desc).each_with_index do |note,index|
9      note.scrubbed_notes.each do |word|
10       term_index = terms.index(word)
11       tfidf[index][term_index] += 1
12     end
13   end
14
15   # Determine idf
16   idf = Array.new(terms.length, 0)
17   tfidf.map do |note|
18     note.each_with_index do |tf,index|
19       if tf > 0
20         idf[index] += 1
21       end
```

```
22        end
23      end
24
25      # Adjust tf values to normalize document length
26      # Using augmented weighting scheme
27      tfidf.map do |note|
28        tf_max = note.max
29        note.map! do |tf|
30          tf = 0.5 + ((0.5 * tf)/tf_max)
31        end
32      end
33
34      # Incorporate idf
35      tfidf.map do |note|
36        note.each_with_index do |tf,index|
37          note[index] = tf * Math.log(idf.length/idf[index])
38        end
39      end
40
41      # Normalize all vectors to unit length
42      tfidf.map do |note|
43        euclength_raw = 0
44        note.map { |tf| euclength_raw += tf**2 }
45        euclength = Math.sqrt(euclength_raw)
46        note.map! { |tf| tf = tf/euclength }
47      end
48
49      # Return tf-idf matrix representing all notes in vector space
50      tfidf
51
52    end
53
54    # Assemble full list of terms
55    def term_list
56      terms = Array.new
57
```

```
58      self.notes.each do |note|
59        note.scrubbed_notes.each do |word|
60          term_index = terms.index(word)
61          if term_index.nil?
62            terms.push(word)
63          end
64        end
65      end
66      terms
67    end
```

A.2 K Means Flat Clustering

```
1
2   def kmeans(tfidf)
3     # Number of seeds
4     k = 5
5
6     # Stop conditions
7     i_max = 30
8
9     # Random selection of seeds
10    seeds = Array.new(k)
11    seednums = Array.new(k)
12    k.times do |seed|
13      protoseed = nil
14      while seednums.include?(protoseed)
15        protoseed = rand((self.notes.count - 1))
16      end
17      seednums[seed] = protoseed
18      seeds[seed] = tfidf[protoseed]
19    end
20
21    clusters = Array.new(k) { Array.new() }
22    # Cluster
23    i = 0
```

```
24    clusterflag = 1
25    while (i < i_max && clusterflag == 1)
26      clusterflag = 0
27      # Clear clusters
28      clusters = Array.new(k) { Array.new() }
29
30      # Assign each note to a cluster
31      tfidf.each_with_index do |note,nindex|
32        min_index = 0
33        min_dist = Float::INFINITY
34
35        seeds.each_with_index do |seed, sindex|
36          eucdist = 0
37          note.each_with_index do |ele, eindex|
38            eucdist = eucdist + ((seed[eindex] - ele)**2)
39          end
40          if eucdist < min_dist
41            min_dist = eucdist
42            min_index = sindex
43          end
44        end
45
46        clusters[min_index].push(nindex)
47      end
48
49      # Recalculate seeds
50      clusters.each_with_index do |cluster,cindex|
51        newseed = Array.new(tfidf[0].length, 0)
52        cluster.each do |note|
53          newseed.each_with_index do |ele, index|
54            newseed[index] = ele + tfidf[note][index]
55          end
56        end
57        newseed.map! { |ele| ele = ele/(cluster.length) }
58        if seeds[cindex] != newseed
59          clusterflag = 1
```

```
60          end
61          seeds[cindex] = newseed
62        end
63
64        i += 1
65      end
66
67      # Return results
68      [seeds,clusters]
69    end
```

A.3 GAAC HAC Clustering

```
1
2   def hac(tfidf)
3     # Set number of clusters (simplest cutoff method)
4     k = 7
5
6     # Initialize arrays
7     c = Array.new(tfidf.length) { Array.new(tfidf.length) { Array.new(2, 0) } }
8     p = c
9     j = Array.new(tfidf.length, 0)
10    a = Array.new()
11    clusters = Array.new(tfidf.length) { Array.new() }
12
13    # Initial similarity matrix and cluster activity array
14    tfidf.each_with_index do |n, n_index|
15      tfidf.each_with_index do |i, i_index|
16        c[n_index][i_index][0] = dotproduct(n, i)
17        c[n_index][i_index][1] = i_index
18      end
19      j[n_index] = 1
20      clusters[n_index].push(n_index)
21    end
22
23    # Sort similarity matrix into priority queue
```

```
24    c.each_with_index do |c_n, index|
25      p[index] = c_n.sort { |x,y| y[0] <=> x[0] }
26    end
27
28    # Remove self similarities from priority queue
29    p.each_with_index do |p_n, index|
30      p_n.delete_if { |item| item[1] == index }
31    end
32
33    # Main clustering loop
34    (j.length - 1 - k).times do |count|
35      #Find the clusters to merge
36      k_1 = 0
37      k_2 = 0
38      max_sim = 0
39      j.each_with_index do |active, index|
40        if active == 1
41          if p[index][0][0] > max_sim
42            k_1 = index
43            max_sim = p[index][0][0]
44          end
45        end
46      end
47      k_2 = p[k_1][0][1]
48
49      # Record the merge
50      a.push([k_1, k_2])
51      j[k_2] = 0
52      clusters[k_1].concat(clusters[k_2])
53      clusters[k_2] = nil
54
55      # Recalculate similarities as needed
56      p[k_1] = []
57
58      new_length = clusters[k_1].length
59      new_sum = Array.new(tfidf[0].length, 0)
```

```
60      clusters[k_1].each do |note|
61        tfidf[note].each_with_index do |ele, index|
62          new_sum[index] += ele
63        end
64      end
65
66      j.each_with_index do |active, index|
67        if active == 1 && index != k_1
68          p[index].delete_if {|item| item[1] == k_1 || item[1] == k_2 }
69
70          comp_length = clusters[index].length
71          comp_sum = Array.new(tfidf[0].length, 0)
72          clusters[index].each_with_index do |ele, index|
73            comp_sum[index] += ele
74          end
75
76          sum_length = new_length + comp_length
77          sum_vectors = Array.new(tfidf[0].length, 0)
78          new_sum.each_with_index do |ele, index|
79            sum_vectors[index] += ele + comp_sum[index]
80          end
81
82          new_c = (1/(sum_length * (sum_length - 1))) * (dotproduct(sum_vectors,
                sum_vectors) - sum_length)
83
84          p[k_1].push([new_c, index])
85          p[index].push([new_c, k_1])
86          p[index].sort! { |x,y| y[0] <=> x[0] }
87        end
88      end
89
90      p[k_1].sort! { |x,y| y[0] <=> x[0] }
91    end
92
93    # Prep outputs
94    clusters.compact!
```

```
95
96      seeds = Array.new(clusters.length) { Array.new(tfidf[0].length, 0) }
97      clusters.each_with_index do |cluster, index|
98        cluster.each do |note|
99          tfidf[note].each_with_index do |ele, e|
100           seeds[index][e] += ele
101         end
102       end
103       clen = cluster.length
104       seeds[index].map! { |ele| ele/clen }
105     end
106
107     [seeds, clusters]
108   end
109
110   def dotproduct(vect1, vect2)
111     size = vect1.length
112     sum = 0
113     i = 0
114     while i < size
115       sum += vect1[i] * vect2[i]
116       i += 1
117     end
118     sum
119   end
```

A.4 Cluster Labeling and Output

```
1
2 def clusters
3     # Cluster processing
4     terms = self.term_list
5     tfidf = self.tfidf_matrix(terms)
6     # seeds, clusters = self.kmeans(tfidf)
7     seeds, clusters = self.hac(tfidf)
8
```

```
9     # Basic cluster-internal labeling
10    numLabels = 7
11    labelSeeds = seeds
12    labels = Hash.new()
13    labels = { :name => "Reflections" , :children => Array.new() }
14
15    labelSeeds.each_with_index do |seed,index|
16      labels[:children].push({ :name => index.to_s , :children => Array.new() })
17      clength = clusters[index].length
18      numLabels.times do |l|
19        val,i = seed.each_with_index.max
20        seed[i] = 0
21        word = terms[i]
22        labels[:children][index][:children].push({:name => word, :value => val**4})
23      end
24    end
25
26    labels
27
28  end
```

Appendix B

Literature Excerpts

B.1 Charles Dickens' *Great Expectations*

- Ours was the marsh country, down by the river, within, as the river wound, twenty miles of the sea. My first most vivid and broad impression of the identity of things, seems to me to have been gained on a memorable raw afternoon towards evening. At such a time I found out for certain, that this bleak place overgrown with nettles was the churchyard; and that Philip Pirrip, late of this parish, and also Georgiana wife of the above, were dead and buried; and that Alexander, Bartholomew, Abraham, Tobias, and Roger, infant children of the aforesaid, were also dead and buried; and that the dark flat wilderness beyond the churchyard, intersected with dykes and mounds and gates, with scattered cattle feeding on it, was the marshes; and that the low leaden line beyond, was the river; and that the distant savage lair from which the wind was rushing, was the sea; and that the small bundle of shivers growing afraid of it all and beginning to cry, was Pip.

- I was soon at the Battery, after that, and there was the right man-hugging himself and limping to and fro, as if he had never all night left off hugging and limping waiting for me. He was awfully cold, to be sure. I half expected to see him drop down before my face and die of deadly cold. His eyes looked so

awfully hungry, too, that when I handed him the file and he laid it down on the grass, it occurred to me he would have tried to eat it, if he had not seen my bundle. He did not turn me upside down, this time, to get at what I had, but left me right side upwards while I opened the bundle and emptied my pockets.

- I had sadly broken sleep when I got to bed, through thinking of the strange man taking aim at me with his invisible gun, and of the guiltily coarse and common thing it was, to be on secret terms of conspiracy with convicts a feature in my low career that I had previously forgotten. I was haunted by the file too. A dread possessed me that when I least expected it, the file would reappear. I coaxed myself to sleep by thinking of Miss Havisham's, next Wednesday; and in my sleep I saw the file coming at me out of a door, without seeing who held it, and I screamed myself awake.

- Now, Joe, examining this iron with a smith's eye, declared it to have been filed asunder some time ago. The hue and cry going off to the Hulks, and people coming thence to examine the iron, Joe's opinion was corroborated. They did not undertake to say when it had left the prison-ships to which it undoubtedly had once belonged; but they claimed to know for certain that that particular manacle had not been worn by either of the two convicts who had escaped last night. Further, one of those two was already re-taken, and had not freed himself of his iron.

- A highly popular murder had been committed, and Mr. Wopsle was imbrued in blood to the eyebrows. He gloated over every abhorrent adjective in the description, and identified himself with every witness at the Inquest. He faintly moaned, "I am done for," as the victim, and he barbarously bellowed, "I'll serve you out," as the murderer. He gave the medical testimony, in pointed imitation of our local practitioner; and he piped and shook, as the aged turnpike-keeper who had heard blows, to an extent so very paralytic as to suggest a doubt regarding the mental competency of that witness. The coroner, in Mr. Wopsle's hands, became Timon of Athens; the beadle, Coriolanus. He enjoyed himself

thoroughly, and we all enjoyed ourselves, and were delightfully comfortable. In this cozy state of mind we came to the verdict Wilful Murder.

- Dinner was laid in the best of these rooms; the second was his dressing-room; the third, his bedroom. He told us that he held the whole house, but rarely used more of it than we saw. The table was comfortably laid no silver in the service, of course and at the side of his chair was a capacious dumb-waiter, with a variety of bottles and decanters on it, and four dishes of fruit for dessert. I noticed throughout, that he kept everything under his own hand, and distributed everything himself.

- If there had been time, I should probably have ordered several suits of clothes for this occasion; but as there was not, I was fain to be content with those I had. My appetite vanished instantly, and I knew no peace or rest until the day arrived. Not that its arrival brought me either; for, then I was worse than ever, and began haunting the coach-office in wood-street, Cheapside, before the coach had left the Blue Boar in our town. For all that I knew this perfectly well, I still felt as if it were not safe to let the coach-office be out of my sight longer than five minutes at a time; and in this condition of unreason I had performed the first half-hour of a watch of four or five hours, when Wemmick ran against me.

- Another night-consultation with Herbert after Provis was gone home (I always took him home, and always looked well about me), led us to the conclusion that nothing should be said about going abroad until I came back from Miss Havisham's. In the meantime, Herbert and I were to consider separately what it would be best to say; whether we should devise any pretence of being afraid that he was under suspicious observation; or whether I, who had never yet been abroad, should propose an expedition. We both knew that I had but to propose anything, and he would consent. We agreed that his remaining many days in his present hazard was not to be thought of.

- It is so difficult to become clearly possessed of the contents of almost any letter, in a violent hurry, that I had to read this mysterious epistle again, twice, before its injunction to me to be secret got mechanically into my mind. Yielding to it in the same mechanical kind of way, I left a note in pencil for Herbert, telling him that as I should be so soon going away, I knew not for how long, I had decided to hurry down and back, to ascertain for myself how Miss Havisham was faring. I had then barely time to get my great-coat, lock up the chambers, and make for the coach-office by the short by-ways. If I had taken a hackney-chariot and gone by the streets, I should have missed my aim; going as I did, I caught the coach just as it came out of the yard. I was the only inside passenger, jolting away knee-deep in straw, when I came to myself.

- Many a year went round, before I was a partner in the House; but, I lived happily with Herbert and his wife, and lived frugally, and paid my debts, and maintained a constant correspondence with Biddy and Joe. It was not until I became third in the Firm, that Clarriker betrayed me to Herbert; but, he then declared that the secret of Herbert's partnership had been long enough upon his conscience, and he must tell it. So, he told it, and Herbert was as much moved as amazed, and the dear fellow and I were not the worse friends for the long concealment. I must not leave it to be supposed that we were ever a great house, or that we made mints of money. We were not in a grand way of business, but we had a good name, and worked for our profits, and did very well. We owed so much to Herbert's ever cheerful industry and readiness, that I often wondered how I had conceived that old idea of his inaptitude, until I was one day enlightened by the reflection, that perhaps the inaptitude had never been in him at all, but had been in me.

B.2 Mark Twain's *The Adventures of Huckleberry Finn*

- I went up the bank about fifty yards, and then I doubled on my tracks and slipped back to where my canoe was, a good piece below the house. I jumped in, and was off in a hurry. I went up-stream far enough to make the head of the island, and then started across. I took off the sun-bonnet, for I didn't want no blinders on then. When I was about the middle I heard the clock begin to strike, so I stops and listens; the sound come faint over the water but clear – eleven. When I struck the head of the island I never waited to blow, though I was most winded, but I shoved right into the timber where my old camp used to be, and started a good fire there on a high and dry spot.

- He listened some more; then he come tiptoeing down and stood right between us; we could a touched him, nearly. Well, likely it was minutes and minutes that there warn't a sound, and we all there so close together. There was a place on my ankle that got to itching, but I dasn't scratch it; and then my ear begun to itch; and next my back, right between my shoulders. Seemed like I'd die if I couldn't scratch. Well, I've noticed that thing plenty times since. If you are with the quality, or at a funeral, or trying to go to sleep when you ain't sleepy – if you are anywheres where it won't do for you to scratch, why you will itch all over in upwards of a thousand places. Pretty soon Jim says:

- I set down again, a-shaking all over, and got out my pipe for a smoke; for the house was all as still as death now, and so the widow wouldn't know. Well, after a long time I heard the clock away off in the town go boom – boom – boom – twelve licks; and all still again – stiller than ever. Pretty soon I heard a twig snap down in the dark amongst the trees – something was a stirring. I set still and listened. Directly I could just barely hear a "me-yow! me- yow!" down there. That was good! Says I, "me-yow! me-yow!" as soft as I could, and then I put out the light and scrambled out of the window on to the shed. Then I

slipped down to the ground and crawled in among the trees, and, sure enough, there was Tom Sawyer waiting for me.

- The first chance we got the duke he had some show-bills printed; and after that, for two or three days as we floated along, the raft was a most uncommon lively place, for there warn't nothing but sword fighting and rehearsing – as the duke called it – going on all the time. One morning, when we was pretty well down the State of Arkansaw, we come in sight of a little one-horse town in a big bend; so we tied up about three-quarters of a mile above it, in the mouth of a crick which was shut in like a tunnel by the cypress trees, and all of us but Jim took the canoe and went down there to see if there was any chance in that place for our show.

- WHEN I got there it was all still and Sunday-like, and hot and sunshiny; the hands was gone to the fields; and there was them kind of faint dronings of bugs and flies in the air that makes it seem so lonesome and like everybody's dead and gone; and if a breeze fans along and quivers the leaves it makes you feel mournful, because you feel like it's spirits whispering – spirits that's been dead ever so many years – and you always think they're talking about YOU. As a general thing it makes a body wish HE was dead, too, and done with it all.

- Next morning I heard Tom was a good deal better, and they said Aunt Sally was gone to get a nap. So I slips to the sick-room, and if I found him awake I reckoned we could put up a yarn for the family that would wash. But he was sleeping, and sleeping very peaceful, too; and pale, not fire-faced the way he was when he come. So I set down and laid for him to wake. In about half an hour Aunt Sally comes gliding in, and there I was, up a stump again! She motioned me to be still, and set down by me, and begun to whisper, and said we could all be joyful now, because all the symptoms was first-rate, and he'd been sleeping like that for ever so long, and looking better and peace-fuller all the time, and ten to one he'd wake up in his right mind.

- And then Tom he talked along and talked along, and says, le's all three slide out of here one of these nights and get an outfit, and go for howling adventures amongst the Injuns, over in the Territory, for a couple of weeks or two; and I says, all right, that suits me, but I ain't got no money for to buy the outfit, and I reckon I couldn't get none from home, because it's likely pap's been back before now, and got it all away from Judge Thatcher and drunk it up.

- Daytimes we paddled all over the island in the canoe, It was mighty cool and shady in the deep woods, even if the sun was blazing outside. We went winding in and out amongst the trees, and sometimes the vines hung so thick we had to back away and go some other way. Well, on every old broken-down tree you could see rabbits and snakes and such things; and when the island had been overflowed a day or two they got so tame, on account of being hungry, that you could paddle right up and put your hand on them if you wanted to; but not the snakes and turtles – they would slide off in the water. The ridge our cavern was in was full of them. We could a had pets enough if we'd wanted them.

- When him and the old lady come down in the morning all the family got up out of their chairs and give them good-day, and didn't set down again till they had set down. Then Tom and Bob went to the sideboard where the decanter was, and mixed a glass of bitters and handed it to him, and he held it in his hand and waited till Tom's and Bob's was mixed, and then they bowed and said, "Our duty to you, sir, and madam;" and THEY bowed the least bit in the world and said thank you, and so they drank, all three, and Bob and Tom poured a spoonful of water on the sugar and the mite of whisky or apple brandy in the bottom of their tumblers, and give it to me and Buck, and we drank to the old people too.

- "It don't make no difference how foolish it is, it's the RIGHT way – and it's the regular way. And there ain't no OTHER way, that ever I heard of, and I've read all the books that gives any information about these things. They always dig out with a case-knife – and not through dirt, mind you; generly it's through

solid rock. And it takes them weeks and weeks and weeks, and for ever and ever. Why, look at one of them prisoners in the bottom dungeon of the Castle Deef, in the harbor of Marseilles, that dug himself out that way; how long was HE at it, you reckon?"

"I don't know."

"Well, guess."

"I don't know. A month and a half."

"THIRTY-SEVEN YEAR – and he come out in China. THAT'S the kind. I wish the bottom of THIS fortress was solid rock."

B.3 H.G. Wells' *The War of the Worlds*

- The storm burst upon us six years ago now. As Mars approached opposition, Lavelle of Java set the wires of the astronomical exchange palpitating with the amazing intelligence of a huge outbreak of incandescent gas upon the planet. It had occurred towards midnight of the twelfth; and the spectroscope, to which he had at once resorted, indicated a mass of flaming gas, chiefly hydrogen, moving with an enormous velocity towards this earth. This jet of fire had become invisible about a quarter past twelve. He compared it to a colossal puff of flame suddenly and violently squirted out of the planet, "as flaming gases rushed out of a gun."

- Then came the night of the first falling star. It was seen early in the morning, rushing over Winchester eastward, a line of flame high in the atmosphere. Hundreds must have seen it, and taken it for an ordinary falling star. Albin described it as leaving a greenish streak behind it that glowed for some seconds. Denning, our greatest authority on meteorites, stated that the height of its first appearance was about ninety or one hundred miles. It seemed to him that it fell to earth about one hundred miles east of him.

- And then he perceived that, very slowly, the circular top of the cylinder was rotating on its body. It was such a gradual movement that he discovered it only through noticing that a black mark that had been near him five minutes ago was now at the other side of the circumference. Even then he scarcely understood what this indicated, until he heard a muffled grating sound and saw the black mark jerk forward an inch or so. Then the thing came upon him in a flash. The cylinder was artificial–hollow–with an end that screwed out! Something within the cylinder was unscrewing the top!

- I stood staring, not as yet realizing that this was death leaping from man to man in that little distant crowd. All I felt was that it was something very strange. An almost noiseless and blinding flash of light, and a man fell headlong and lay still; and as the unseen shaft of heat passed over them, pine trees burst into fire, and every dry furze bush became with one dull thud a mass of flames. And far away towards Knaphill I saw the flashes of trees and hedges and wooden buildings suddenly set alight.

- I touched the curate's leg, and he started so violently that a mass of plaster went sliding down outside and fell with a loud impact. I gripped his arm, fearing he might cry out, and for a long time we crouched motionless. Then I turned to see how much of our rampart remained. The detachment of the plaster had left a vertical slit open in the debris, and by raising myself cautiously across a beam I was able to see out of this gap into what had been overnight a quiet suburban roadway. Vast, indeed, was the change that we beheld.

- The internal anatomy, I may remark here, as dissection has since shown, was almost equally simple. The greater part of the structure was the brain, sending enormous nerves to the eyes, ear, and tactile tentacles. Besides this were the bulky lungs, into which the mouth opened, and the heart and its vessels. The pulmonary distress caused by the denser atmosphere and greater gravitational attraction was only too evident in the convulsive movements of the outer skin.

- Lessing has advanced excellent reasons for supposing that the Martians have actually succeeded in effecting a landing on the planet Venus. Seven months ago now, Venus and Mars were in alignment with the sun; that is to say, Mars was in opposition from the point of view of an observer on Venus. Subsequently a peculiar luminous and sinuous marking appeared on the unillumined half of the inner planet, and almost simultaneously a faint dark mark of a similar sinuous character was detected upon a photograph of the Martian disk. One needs to see the drawings of these appearances in order to appreciate fully their remarkable resemblance in character.

- The farther I penetrated into London, the profounder grew the stillness. But it was not so much the stillness of death–it was the stillness of suspense, of expectation. At any time the destruction that had already singed the northwestern borders of the metropolis, and had annihilated Ealing and Kilburn, might strike among these houses and leave them smoking ruins. It was a city condemned and derelict....

- In another moment I had scrambled up the earthen rampart and stood upon its crest, and the interior of the redoubt was below me. A mighty space it was, with gigantic machines here and there within it, huge mounds of material and strange shelter places. And scattered about it, some in their overturned war-machines, some in the now rigid handling-machines, and a dozen of them stark and silent and laid in a row, were the Martians–dead!–slain by the putrefactive and disease bacteria against which their systems were unprepared; slain as the red weed was being slain; slain, after all man's devices had failed, by the humblest things that God, in his wisdom, has put upon this earth.

- Then, advancing obliquely towards us, came a fifth. Their armoured bodies glittered in the sun as they swept swiftly forward upon the guns, growing rapidly larger as they drew nearer. One on the extreme left, the remotest that is, flourished a huge case high in the air, and the ghostly, terrible Heat-Ray I had already seen on Friday night smote towards Chertsey, and struck the town.

Bibliography

[1] Albert V. Bruno, Joel K. Leidecker, and Joseph W. Harder. Why firms fail. *Business Horizons*, 30(2):50, 1987.

[2] Albert V. Bruno, Edward F. McQuarrie, and Carol G. Torgrimson. The evolution of new technology ventures over 20 years: Pattern of failure, merger, and survival. *Journal of Business Venturing*, 7(4):291, 1992.

[3] Richard O Duda, Peter E Hart, and David G Stork. *Pattern classification*. John Wiley & Sons,, 1999.

[4] Beatrice DŒşIppolito. The importance of design for firmsŒş competitiveness: A review of the literature. *Technovation*, 2014.

[5] Anil K Jain, M Narasimha Murty, and Patrick J Flynn. Data clustering: a review. *ACM computing surveys (CSUR)*, 31(3):264–323, 1999.

[6] Dan Jurafsky and James H. Martin. *Speech and language processing : an introduction to natural language processing, computational linguistics, and speech recognition*. Upper Saddle River, N.J. : Pearson Prentice Hall, c2009., 2009.

[7] Eric H. Kessler and Alok K. Chakrabarti. Speeding up the pace of new product development. *Journal of Product Innovation Management*, 16(3):231 – 247, 1999.

[8] Dan Klein and Christopher D Manning. Fast exact inference with a factored model for natural language parsing. In *Advances in neural information processing systems*, pages 3–10, 2002.

[9] David A. Kolb. *Experiential learning : experience as the source of learning and development / David A. Kolb.* Englewood Cliffs, N.J. : Prentice-Hall, c1984., 1984.

[10] K Lauche. Heedful action, reflection, and transfer in the design process. *WDK Publications*, pages 267–274, 2001.

[11] K Lauche et al. Facilitating creativity and shared understanding in design teams. In *DS 30: Proceedings of DESIGN 2002, the 7th International Design Conference, Dubrovnik*, 2002.

[12] Hugo Liu and Push Singh. Conceptnet: a practical commonsense reasoning tool-kit. *BT technology journal*, 22(4):211–226, 2004.

[13] Christopher D. Manning, Prabhakar Raghavan, and Hinrich SchÃijtze. *Introduction to information retrieval*. New York : Cambridge University Press, 2008., 2008.

[14] Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, 2014.

[15] Mitchell P Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of english: The penn treebank. *Computational linguistics*, 19(2):313–330, 1993.

[16] TJ Marion, JH Friar, and TW Simpson. New product development practices and early-stage firms: Two in-depth case studies. *Journal of Product Innovation Management*, 29(4):639 – 654, 2012.

[17] George A Miller. Wordnet: a lexical database for english. *Communications of the ACM*, 38(11):39–41, 1995.

[18] Louis Mullie. Treat: The ruby nlp toolkit. `http://www.github.com/louismullie/treat`.

[19] Gerhard Pahl, Petra Badke-Schaub, and Eckart Frankenberger. RÃľsumÃľ of 12 years interdisciplinary empirical studies of engineering design in germany. *Design Studies*, 20:481 – 494, 1999.

[20] Kishore Papineni. Why inverse document frequency? In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics, 2001.

[21] Isabelle Reymen. *Improving design processes through structured reflection: A domain-independent approach*. Technische Universiteit Eindhoven, 2001.

[22] Isabelle MMJ Reymen et al. Research on design reflection: overview and directions. In *DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design, Stockholm*, 2003.

[23] Stephen E Robertson and K Sparck Jones. Relevance weighting of search terms. *Journal of the American Society for Information science*, 27(3):129–146, 1976.

[24] Gerard Salton. *The SMART retrieval system; experiments in automatic document processing*. Englewood Cliffs, N.J., Prentice-Hall [1971], 1971.

[25] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. *Information processing & management*, 24(5):513–523, 1988.

[26] Donald Schön and John Bennett. Reflective conversation with materials. In *Bringing design to software*, pages 171–189. ACM, 1996.

[27] Donald A. Schön. *The reflective practitioner : how professionals think in action.* New York : Basic Books, c1983., 1983.

[28] Peter HA Sneath, Robert R Sokal, et al. *Numerical taxonomy. The principles and practice of numerical classification.* W.H. Freeman, 1973.

[29] S. C. Stumpf and J. T. McDonnell. Talking about team framing: using argumentation to analyse and support experiential learning in early design episodes. *Design Studies*, 23(1):5 – 23, 2002.

[30] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[31] Simon J. Towner. Four ways to accelerate new product development. *Long Range Planning*, 27(2):57 – 65, 1994.

[32] Karl T. Ulrich and Steven D. Eppinger. *Product design and development / Karl T. Ulrich, Steven D. Eppinger.* New York : McGraw-Hill/Irwin, c2012., 2012.

[33] A. C. Valkenburg. *The reflective practice in product design teams.* TU delft: Industrial design engineering, TU Delft, Delft University of Technology, Delft, December 2000.

[34] Robert W. Veryzer Jr. Discontinuous innovation and the new product development process. *Journal of Product Innovation Management*, 15(4):304 – 321, 1998.

[35] Marilyn Wood Daudelin. Learning from experience through reflection. *Organizational dynamics*, 24(3):36–48, 1997.