Chapter 2

HAMLET

Neural-Net-Powered Prototypes for Library Discovery

Andromeda Yelton*

In 2017, I trained a neural net on MIT's graduate thesis collection and used this neural net to power several experimental discovery interfaces. The system is collectively named HAMLET ("How about Machine Learning Enhancing Theses?"), and you can explore the results online at the URL in the gray box. What does this mean, and what does it imply for the future of library discovery?

HAMLET
https://hamlet.andromedayelton.com

What Is a Neural Net?

First, some background on neural nets. In traditional software design, programmers create rules that machines should use to make decisions and encode those rules into software. In machine learning, by contrast, programmers encode structures that software can use to create its own rules. They then train these structures on data sets—ideally very large ones, with many thousands or even millions of records. With each additional record, the software updates its model of the world a little bit; ideally, it slowly converges on a model that will be useful for making predictions about or drawing inferences from data it encounters subsequently.

There are many structures that programmers can use in machine learning systems, and exploring them all is outside the scope of this piece. I will briefly elucidate the type of machine learning that HAMLET used: a neural net.

Neural nets, as the name suggests, are inspired by biology. Our brains aren't composed of step-by-step programs; instead, they're made of billions of neurons. Each neuron can perform a tiny bit of reasoning, responding to particular stimuli and communicating its response to the comparatively small number of neurons it connects to. The collective outcome of all these tiny decisions is a rich, flexible reasoning system.

In computational neural nets, each neuron is a function that takes certain inputs (stimuli) and returns certain outputs (responses; in practice, typically a number close to either 0 or 1). It can receive those inputs either directly from the training data set or from other neurons it's connected to; its outputs may feed into a final output function or may serve as inputs for other neurons. Neural nets generally have several layers (i.e., sets of neurons that do their work in parallel): one that draws directly from the training data, another that takes the outputs of the first layer, and so forth until the final outputs.

How do these neurons know what outputs to return? This is determined via training the neural net (just as human brains need to spend a long time gathering data about the world in order to form reasonable models of it). Before training, programmers determine the general type of function each neuron should be and initialize it with random parameters. (In the equation $y = mx + b$ that you met in algebra, m and b are the parameters; $y = mx + b$ describes a line no

* **Andromeda Yelton** (https://andromedayelton.com) is a software engineer and librarian. Currently, she is at the Berkman Klein Center. She has written code for the MIT Libraries, the Wikimedia Foundation, and more. She has written, spoken, and taught internationally on a variety of library technology subjects. She is Past President of the Library & Information Technology Association.

matter what values you give to *m* and *b*, but that line's slope and placement can vary dramatically.)

During training, the neural net receives records from the training data set, one at a time. For each record, it compares the final output of the net to some sort of expected value. For example, if the inputs are photographs, the output might be a binary decision: "cat" or "not a cat." The neural net then evaluates how wrong it was and updates all the parameters of all its functions a little bit, in whatever direction would have made it less wrong. Over time, as it trains on a large enough number of records, it gets more and more accurate.

Ideally, over time, the neural net not only becomes a good model for its training data; it also does a good job modeling data that it's never encountered before. (But only similar kinds of data; neural nets trained on one knowledge domain may be bizarrely or hilariously wrong when asked to evaluate data from other domains.) In practice, this means, for example, that a neural net trained to identify cat photos will have reasonable accuracy in making cat/not-cat decisions about unfamiliar photos. Computers are still not as good as humans at this sort of task, and the types of mistakes they make are very different from the types that humans make (and sometimes incomprehensibly weird), but they can handle much larger volumes of data much faster than humans, which makes machine processing a good fit for some tasks.

How HAMLET's Neural Net Works

The previous section covered the general concept of training neural nets, but was vague on the exact algorithms. That is because many algorithms can be used.

HAMLET uses the doc2vec algorithm. This is an algorithm for estimating the similarity in meaning between different documents, based on a widely used algorithm word2vec, which estimates the similarity between words.

Word2vec works by assuming that if two words occur in similar contexts, they likely have similar meanings. For instance, let's imagine that a set of training documents included the following two sentences: "Avram is important to library science" and "Ranganathan is important to library science." Word2vec would conclude that the words *Avram* and *Ranganathan* must be at least a little bit similar in meaning. As it iterates repeatedly over the same training corpus, it can use what it's learned about word similarity from earlier passes to make more informed guesses about which words are similar. For instance, after it concludes that *Avram* and *Ranganathan* have something in common, if it encounters a sentence like "Avram influenced the development of cataloging," it would be inclined to predict that "Ranganathan influenced the development of cataloging" is a plausible sentence. It would be much less likely to hypothesize that "Racecars influenced the development of cataloging," as it probably did not encounter the word *racecars* in contexts like the ones where it encountered *Avram* or *Ranganathan*.

Doc2vec is an extension of word2vec that adds one more fact to every context: to wit, an identifier for the document. That is, instead of looking only at the words surrounding any given word, it looks at those and also the document identifier and takes those collectively as the context for a word. The idea here is that documents have an overall meaning, and this overall meaning helps you predict any individual word's meaning—or, conversely, words and their context help you predict the overall meaning of a document.

It's important to note that the doc2vec and word2vec algorithms learn *which words probably have similar meanings*, but not, in fact, *what those meanings are*. It can learn that *Avram* and *Ranganathan* are more similar than *Avram* and *racecar*, but it doesn't know that Avram was a human being and racecars are transportation machinery. Under the hood, each word is represented by a set of coordinates in space. Similarity between two words is just the distance between them, the same way that GPS coordinates tell you which points on a map are closer together or farther apart. Humans can draw inferences about Avram and racecars based on their underlying knowledge of humans and machinery, but word2vec cannot, as it has no semantic model to draw from.

Neural Nets and Traditional Metadata

As a librarian, you're likely approaching this chapter using a framework of cataloging, classification, taxonomy, and controlled vocabularies. I encourage you to question every assumption that this framework leads you to make. In a machine learning context, many of these assumptions are wrong. For example:

Neural nets do not produce categories. In traditional metadata schemata, works are collocated by their membership in a shared category, and each work either definitely does or definitely doesn't have a given subject heading assigned to it in a record. In a doc2vec-based neural net, by contrast, documents are simply closer together or farther apart. Every document can be viewed as the center of its own category, and you can use judgment—more an art than a science—to decide which works are "close enough" to count as similar. Or you can abandon category boundaries entirely, and instead arrange works on a spectrum of similarity: instead of saying that work A is in the same category as work B but work C is not, you might say that

A is 85 percent like B and C is 32 percent like it, and allow your interfaces to reflect that spectrum.

Neural nets can produce clusters, but these clusters don't have (and sometimes can't have) subject headings. In traditional metadata, clusters of documents have meaningful labels because we intentionally create subject headings around meaningful categories, and then we create clusters of works by labeling them with particular subject headings. With neural nets, clusters may emerge—like cities on a map, some regions in the coordinate space will be more populated than others—and we may choose to draw boundaries around them. (See the department visualizations example in the section "Future Possibilities" below.) However, there is no meaningful label for that cluster until and unless we choose to create one. And it is not always obvious what that label should be; the neural net can't explain why it chose to collocate particular works, and the similarity is derived from a mathematical model, not a semantic one. Domain experts may be able to assign labels, and that assignation may result in rich and useful interface possibilities, but the label creation is an optional step, not the first step. And in some cases, even experts can't assign meanings because the clusters don't map to human concepts.

Neural nets can operate in spaces where traditional metadata is unavailable or inadequate. One of the reasons I used the MIT thesis corpus, in fact, is that it's hard to explore due to the nature of its metadata. DSpace theses do not have subject headings. They do have author-assigned keywords, but most of them are so granular that they apply to only one thesis and therefore don't collocate anything. Thesis records do include department names, but these are not very helpful for two reasons. First, some departments have far too many theses for department-level browsing to be useful; there are 9,625 theses just in Course VI (Electrical Engineering and Computer Science). Second, department-level distinctions both collocate theses that don't go together (in a subject-header sense) and separate some that do. To use Course VI again as an example, theses in electrical engineering generally concern completely different tools, ideas, and materials than theses in computer science. At the same time, some theoretical computer science theses could be equally at home in a math department, and some electrical engineering theses are not readily distinguishable from physics.

Subject headings would be the right level of granularity for exploring this corpus, but they aren't present. Furthermore, they aren't going to be; providing them for all 50,825 theses (and counting) would be prohibitively labor-intensive. Training a neural net on a corpus this size, however, is no more than a few days of background processing on a modern laptop, and much less on cloud infrastructure; the human effort to design and build that system, while nontrivial, is far less than that of cataloging tens of thousands of theses.

Are these contrasts between traditional taxonomies and neural-net-generated systems good, bad, or merely different? That's a matter of taste. Whatever your taste is, I encourage you to think about HAMLET and other machine learning systems on their own terms, rather than shoehorning them into a cataloging and classification framework they do not fit into. They are both more alien and more rife with possibility than they may initially seem.

HAMLET's Prototypes

Currently, HAMLET has three prototype interfaces: a recommendation engine, an uploaded file oracle, and a literature review buddy. You can play with all of them at the URL in the gray box.

HAMLET
https://hamlet.andromedayelton.com

The **recommendation engine** lets you search for theses by author or title and tells you which other theses are most conceptually similar. This allows for an experience analogous to browsing by subject, albeit grounded in a very different metadata paradigm, where each document is the center of its own subject-heading universe. For example, the URL in the gray box below relates to the PhD thesis for Buzz Aldrin, better known as the second man to walk on the moon. His 1963 thesis in the Department of Aeronautics and Astronautics was "Line-of-Sight Guidance Techniques for Manned Orbital Rendezvous." HAMLET's ten most similar theses are also all in the Aero/Astro department. However, they achieve much better relevance than department-level metadata alone could provide: most of them pertain to spacecraft control and orbital navigation, including orbital rendezvous. In addition, they span from 1959 to 2007, thus letting readers explore the development of these ideas across time.

Theses Most Similar to Those of Author Aldrin, Buzz
https://hamlet.andromedayelton.com/similar_to/author/52842

The **uploaded file oracle** provides similar functionality, returning a list of theses most similar to a starting document. However, instead of starting with an existing thesis, it starts with a user-uploaded

document, which it interprets on the fly in the context of the neural net. For example, researchers might upload articles they're reading or chapters of their works in progress to discover other, similar documents that might be relevant to their research.

Alternately, Jason Griffey (editor of this volume) tested it by uploading *Peter Pan*. This was a text I did not expect to work well because neural nets do best when they have large volumes of data to train on, and a children's novel is clearly very unlike the STEM theses that make up the vast majority of the training corpus. However, HAMLET gamely produced theses from MIT's tiny creative writing program: unquestionably the most similar available works.

One of my first tests was uploading the Wikipedia article on strong and weak typing, a core concept in computer programming. This was the first moment where I was truly elated about the possibilities of this system because it did exactly what I hoped: to wit, collocate theses on the same topic from different departments. DSpace's browsing interface and underlying metadata work well only for bringing together works with the same author, advisor, or department, thus making it impossible to find interdisciplinary work; however, many researchers find themselves at the borders between disciplines, where the most relevant works may be outside their department and thus hard to find via systems that follow disciplinary lines.

Wikipedia: Strong and weak typing
https://en.wikipedia.org/wiki/Strong_and_weak_typing

Given this Wikipedia article, HAMLET's second recommendation was for a computer science thesis on type inferencing in the Python programming language. This is gratifyingly relevant. But the most exciting recommendation is the seventh, "Foundation Elements for Computer Software Systems in the Fluid Sciences," a 1969 thesis in the Department of Meteorology.

MIT aficionados will recognize that the Institute does not, in fact, have a Department of Meteorology. It did until the 1980s (at which point it was renamed, and then merged into Earth, Atmospheric, and Planetary Sciences); however, this was long enough ago that it is unusual to come across this part of the Institute's intellectual history. The year 1969 is also interesting because at that point MIT did not yet have a department for computer science. The Laboratory for Computer Science was founded in 1963, but not until 1975 did the then-Department of Electrical Engineering add computer science to its name.

This thesis recommendation, then, tantalizingly suggests a moment in history: so early that, not only were foundational programming concepts being worked out as thesis topics, but also that computer science was scattered across the campus, finding homes in the laboratories of whatever early-adopter professors saw an application for these new machines. Moreover, this early-adopter professor is Edward Lorenz, the pioneer of chaos theory popularly known for the butterfly effect. This is a phenomenon that characterizes certain complicated mathematical models, such as the ones that describe the weather. Being complicated, weather models benefited enormously from the growing availability of computers . . . which is why a graduate student was working out fundamental programming concepts in the laboratory of a famous meteorologist. It's a thesis title, but it's also a story.

Finally, the **literature review buddy** suggests sources you may want to incorporate into your research. It uses the uploaded file oracle back end to find the theses most similar to your uploaded text and then lists for you all the sources that were cited by these theses. There are both precision and recall challenges here in that the bibliographies were not available as structured data; I had to parse them out of the full text, which was complicated by underlying inaccuracies in the OCR. A production-grade system would have significant data quality questions to answer. However, imagine how useful this type of system could be: a student could upload a work in progress and immediately get a list of all works cited by related theses. With sufficient metadata quality, this list could be ranked by how many theses cited each work, filtered by any number of criteria, and even linked directly to borrowing or interlibrary loan options. It might even surface options unlikely to be found through any traditional catalog search, such as unpublished works or personal communications.

Traps for the Unwary

While HAMLET, like any sufficiently advanced technology, can seem like magic, it's merely software plus data. As such, it reflects the limitations of its algorithms and the biases of its underlying corpus.

First, all machine learning systems share a problem, which is that they are only as good as the data they are trained on. If that corpus has significant biases or omissions, those will be reflected in the outputs. Additionally, machine learning systems need a large amount of data to work reliably; when they are trained on too little data, they may still produce results, but those results are nothing more than elaborately obfuscated dice-rolling.

In the MIT case, the most obvious limitation of the corpus is that MIT almost exclusively awards degrees in STEM topics, plus management. This means that the HAMLET neural net is likely to work well for content in fields like electrical engineering: it will

produce a large number of results, many of those results will be above a high similarity threshold, and users can be reasonably confident that the system knows what it's talking about. It may produce results in fields like philosophy or writing, but—*Peter Pan* notwithstanding—those results are more tenuously connected to real meaning. If users upload texts that reflect (for example) art history, education, or dance, HAMLET may produce no results at all—or, worse, it may produce results that are almost certainly not grounded in meaning.

This suggests a second problem with neural nets, which is their relationship to human users. People may assume that computer systems are objective, comprehensive, or otherwise absolutely correct. They may think that the outputs represent absolute facts rather than statements about probability—in the HAMLET case, this would mean assuming that all given theses are definitely very similar to the original text, rather than probably somewhat similar. (While HAMLET does produce a similarity estimate, this isn't reflected in the current interface; even if it were, people might not read it or know how to contextualize it.) Or they might assume the opposite—that coming across one thesis that they know isn't relevant means the whole system is useless. Artificial intelligence does not actually remove the need for human intelligence.

Finally, and most worrisomely, users may think that the outputs of a computer system represent a *normative* rather than a *descriptive* fact: a statement about how the world should be rather than what a particular part of the world is. For an example of the potentially high stakes of this question, do an image search for "CEO." Likely the results will overwhelmingly be pictures of white men. This is an accurate descriptive statement about CEO demographics—but it is not a normative statement that only white men *should be* CEOs! These image results do not carry any information about the leadership abilities of any other demographics, but it is easy to believe they do. After all, if Google said it, it must be true.

Future Possibilities

Where else can we go with interfaces that have neural nets of this type on the back end?

My next goal is data visualization. The neural net encodes information about connections between texts, but they're not easily explorable in a text-only interface. Imagine, instead, a map where smaller or larger circles, more or less widely spaced, showed the clusters of meaning in the corpus. By zooming in to clusters, you could see the individual connections between texts that made them up. This would facilitate several types of explorations:

- First, it would be instantly apparent where the corpus—that is, MIT's intellectual history—had strengths and gaps. This might be of interest to collection development librarians or critical social theorists.
- Second, by applying a date slider, you could watch as particular areas of research grew and shrank over time.
- Third, if dots representing individual theses were color-coded by department, interdisciplinary works and topics would become instantly apparent.
- Fourth, and perhaps most importantly for anyone who wants to demonstrate the value of the library to the faculty, users could ego-surf. People could search for works they wrote, or advised, and instantly see the network of related works. Some would doubtless be familiar, but others might come from other departments or decades. People new to an organization or trying to find their way in a large university could quickly find others with similar research interests. People operating near, or outside, the limits of their discipline could find collaborators in other departments.

You can see some preliminary investigations as to how this visualization might work at the MIT Libraries Machine Learning Studio. In the blog posts here, I used d3.js, plus prototype neural nets trained on single departments, to explore clusters of related works in the aeronautical and astronautical engineering, chemistry, and physics departments. While the algorithm can't generate topical labels for these clusters—we still need humans for that—their existence and relative size stand out quickly. By manually exploring thesis titles within particular clusters, I can see some semantic unity to these clusters. For instance, the larger blue cluster in the aero/astro department generally concerns compressor performance and aerodynamics; the smaller red one is all about the characteristics of composite laminates under stress.

MIT Libraries Machine Learning Studio
https://mitlibraries.github.io/ml2s

I also used these preliminary investigations to trace the meaning of a single word through the corpus. In the resulting blog post, "Six Ways of Looking at Oxygen," I found that the meaning of the word *oxygen* varies substantially depending on the disciplinary lens you use. In a neural net trained on aero/astro theses, *oxygen* is most similar to words like *hydrogen*, *water*, and *propellant*, and isn't too far off from *hypergolic:* if all we know about the world is aero/astro, oxygen is rocket fuel. In the chemistry department, by

comparison, *oxygen* is like *nitrogen* and *chlorine*: it's an element (a gaseous one in the upper right of the periodic table, even). And if you're a biologist, *oxygen* is close to one cluster centered on *energy* and another centered on *nutrient*; it's fuel again, not for rockets but for organisms.

Six Ways of Looking at Oxygen
https://mitlibraries.github.io/ml2s/2017/07/06/six-ways-of-looking-at-oxygen.html

As noted above, the word2vec and doc2vec algorithms don't natively understand the meanings of these clusters; we still need humans (for now) with domain knowledge to explore and label them. Other machine learning techniques, such as topic modeling, might prove useful complements to these neural net techniques by automatically extracting labels for clusters. Alternately, neural nets and skilled catalogers together could generate wholly new and compelling interfaces.

None of this is precisely easy; though software to streamline machine learning is increasingly available, applying it without understanding the underlying mathematics can easily result in attractive nonsense. Cleaning existing documents and metadata to a production-ready state can be formidable; algorithmic interfaces are sometimes much less tolerant of messy data than humans are. At the same time, none of this is precisely as hard as it seems, either; HAMLET was a side project fit into spare hours.

In summary, machine learning techniques allow for exploratory, sometimes visual, interfaces that support old use cases in new ways, or allow for new uses. They can complement traditional metadata, but also open up possibilities for document sets that do not have, and may be unlikely to get, such records. They can challenge our understanding of library use cases, interfaces, and metadata. Above all, I hope that they can surprise and delight, startling us as we round an intellectual corner to discover something so relevant it feels like magic, just as all the best library experiences should.