AI Meets The Bible

Christopher Minson Follow
Sep 4, 2019 · 22 min read

A Gentle Introduction to Computational Theology

Introduction

In this tutorial we'll build a basic AI that reads and interprets the Christian Bible. AI has provided deep insights in many domains. What will it think about theology?

This effort won't need a team of PhD's and a datacenter of computers. That's because powerful machine-learning tools are now widely available, allowing anyone with modest math and programming skills to apply the technology against a variety of tasks. This project serves as an example of what's possible in this regard.

This article is for a general audience who would like to better understand AI. The goal is to illustrate how these techniques work and to expand the imagination for what's possible, but without resorting to serious math or software listings. Therefore you won't find any code or equations here. Instead I'll verbally describe how the system works and provide demonstrations you can try yourself. AI will remake our world. Given that, we could all use a better intuitive feel for the capabilities and limitations of this technology.

If however you wish to go straight to the code, the github repos are here and here.

I chose the Bible because most people have some familiarity with its content and stories. In addition, I wanted to demonstrate how AI can be applied in unorthodox ways, including bringing fresh perspectives to ancient data. However these techniques could be used with any sort of text.

Here's the plan. First, I'll describe how the system was built and the key technical ideas involved. This will demystify these techniques while illustrating their power. Then we'll read the Kings James Bible into the AI and explore some overview results. After that

we'll dive into the conceptual parts of the system and ask it some questions — and get some answers!

Lastly we'll discuss how one could build on top of this system, leading to much more powerful AIs in the future.

Building The System

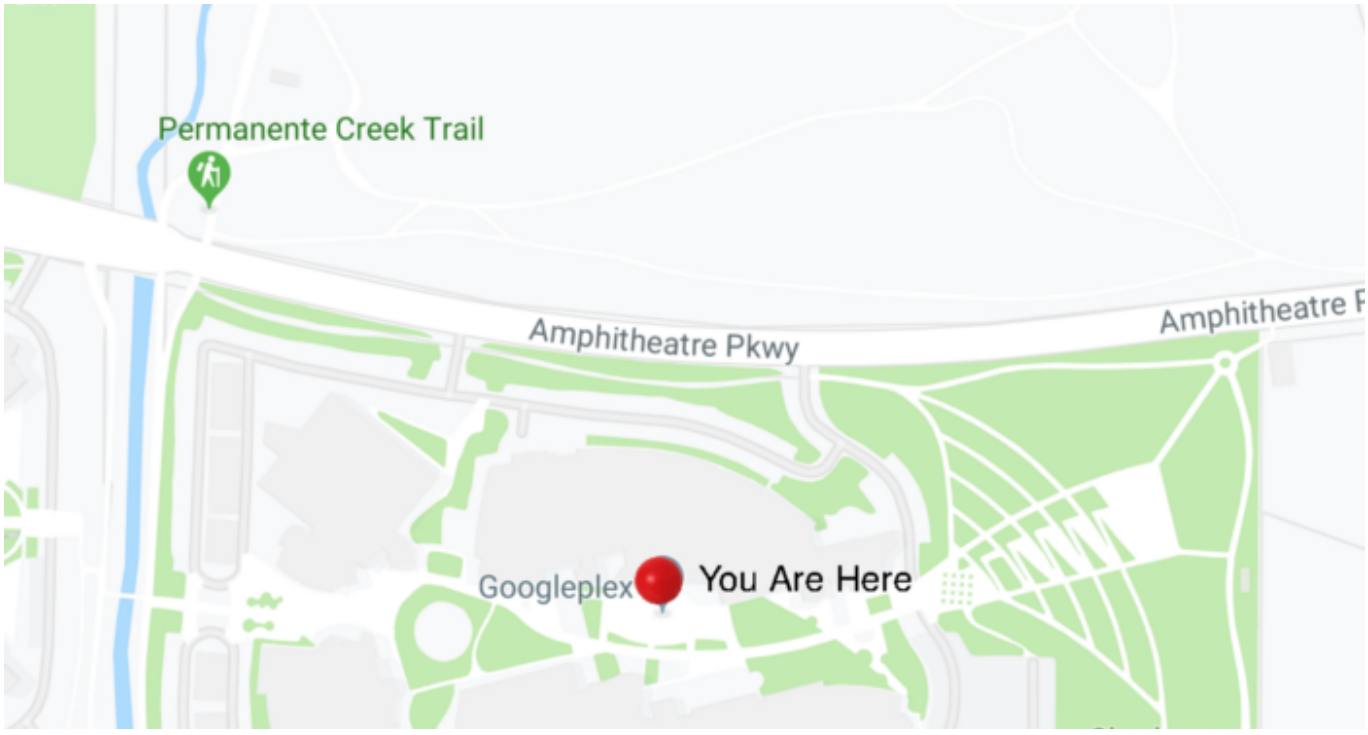
This project is based on *embeddings*. Embeddings are a core AI concept. If you understand them you'll not only better understand how this particular project works, but you'll also gain an insight into AI in general. So let's briefly discuss the idea.

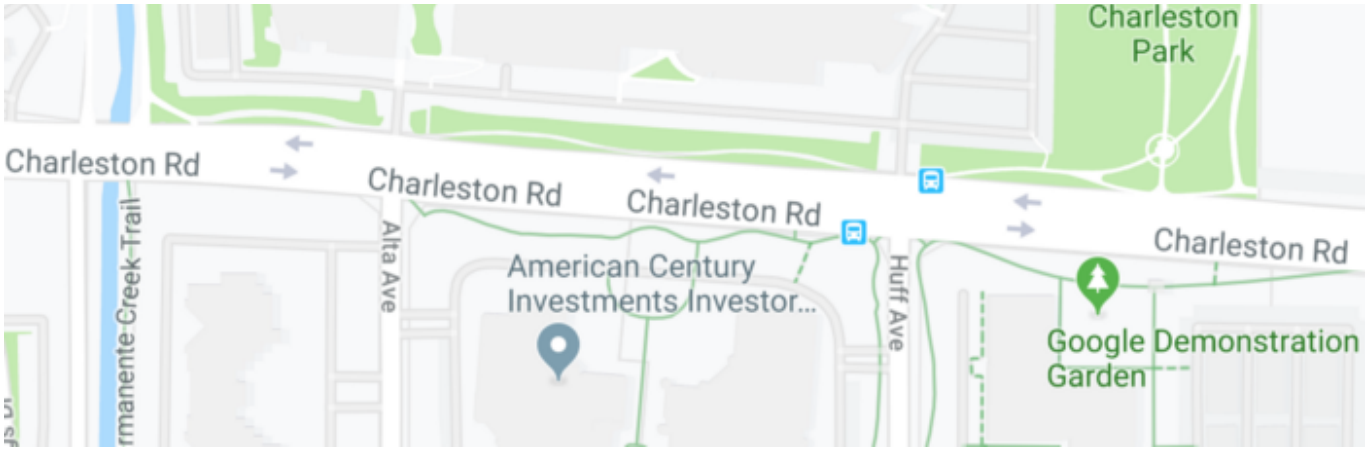
We start with *vectors*. A vector is nothing more than a list of numbers. For example, here's a vector that contains five numbers:

[37.21, 541.89, 14.20, 42.19, 808.0]

For our purposes, what's interesting about vectors is that they can represent a position in space. So if we wanted to specify a place on a map, we just make a vector that has a longitude and a latitude. The resulting 2-D vector contains those two numbers. It looks like this:

[37.4220, -122.0841]

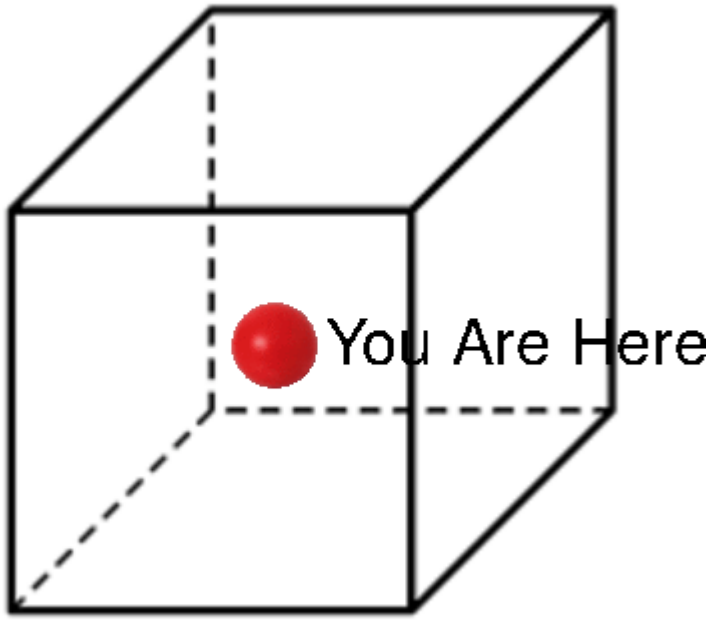




We've just made a vector that describes a specific point on a map. In this case we specified Google headquarters in Silicon Valley.

So what if we wanted a vector that described a point in 3-D space? Well then, we'd need a vector with 3 numbers such as this:

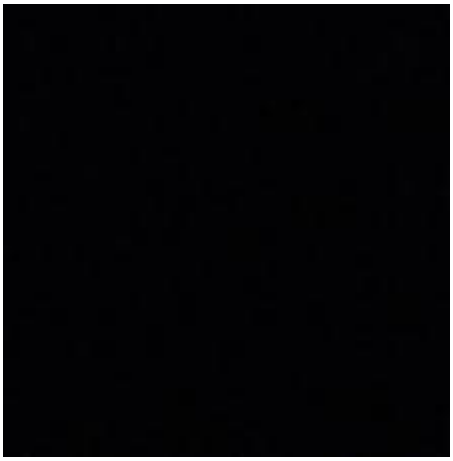
[123.904, 174.321, 190.211]



And now we have a vector describing the coordinates of a point inside a 3D object, such as a sphere or cube. In short, a vector can contain coordinates describing any point, in any number of dimensions.

You can make a vector any size. For instance, I could create a vector of length 300. That means it has 300 numbers and describes a specific point in a 300-dimensional space. It looks like this:

[133.504, 475.321 … Many More Numbers … 762.343, 89.498]



Whoa, we've hit a problem. But this problem is due to our human mental limits, not the technology. Our brains didn't evolve to visualize entities with more than 3 dimensions, because outside Star Trek we seldom meet such creatures in daily life. Therefore that black square will have to suffice as our 300-dimensional object. But whether we can visualize lots of dimensions or not, it doesn't matter to the math. The AI calculations are the same whether we have 2 dimensions or 2 million.

More dimensions means more information, which is just another way of saying that longer vectors encode more information than shorter vectors. This information density is limited only by underlying computing resources.

We call these long list of numbers *high-dimensional* vectors. When we have a number of such vectors, we can think of a collection of them inhabiting high-dimensional *spaces*. These spaces are used everywhere in AI, as a compact and efficient way to represent vast amounts of information. Even better, once data is represented in this form we can apply mathematical operations to them. In fact, there's a branch of mathematics devoted to this task. As an interesting side note, this same branch of mathematics is also applied to quantum mechanics.

So what's an embedding? An embedding is simply any piece of information (a piece of text, a musical score, a set of scientific equations) that's been translated — or embedded — into a vector. That vector now represents that information, but in a way that is correlated with other embeddings.

For instance, consider the following word: *parrot*. These letters represent a particular family of birds. But if we *embed* the concept of parrot, we get something like the following:

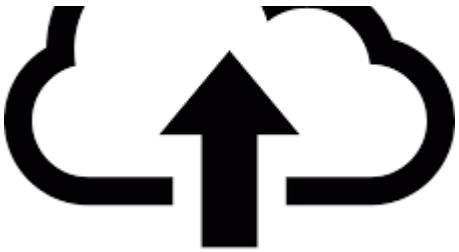
[492.844, 390.891 … Many More Numbers … 464.544, 901.573]

Yes, this really means "parrot". The concept of parrot has become embedded into a high-dimensional space. These numbers are the coordinates of parrot in that space, just like latitude and longitude provide coordinates in 2D space. When an AI such as ours "thinks" about parrots, it's really thinking about this vector and the position in high-dimensional space that the vector embedding describes.

That's all well and good, but you still might wonder why we're bothering. This seems like a lot of effort without clear benefit. Where's the advantage?

There are three main advantages, each of which are enormously powerful. First, recall how vectors can store large amounts of data. Thus, when we embed a concept into a vector, we create a rich mathematical entity that can contain any amount of information about that specific concept. Compare that with the word "parrot", which is just six letters that act as a label. There's no other information present besides those letters. In contrast, a 300-dimensional embeddings has 300 numbers than be precisely tuned to represent all kinds of information about parrots.

Second, once we have an embedding, we are free to compute with it. Embedded parrots are mathematical objects. That means, once we have an embedded parrot then we can do math on parrots. This allows us to query the system and ask it conceptual questions about parrots and everything related to them. In effect, embeddings lift human concepts into the mathematical universe where AI exists.



[493.844, 390.891 … Many More Numbers … 464.544, 901.573]

embeddings upload concepts from physical universe to AI universe

The third advantage is both critical and particularly non-obvious: embeddings cluster together based on similarity. The embeddings for related terms have similar coordinates in space, just like neighbors in a city have similar street addresses. This means that if "parrot" is properly embedded in a space, then the embedding for "bird" will be living nearby, as will all embedded characteristics of parrots (feathers, intelligence, color, playfulness etc). Given that, we can easily query high-dimensional spaces and discover all sorts of connections between embedded concepts, ranging from the obvious and well-known to the undiscovered and totally original.

We'll soon apply this capability to the Bible, rather than to overly smart birds.

Embeddings are eerily powerful. They are also somewhat counter-intuitive and not fully understood. You can dive pretty deep into this topic, as a google search will demonstrate.

Okay, so embeddings are cool and key to this project. But how do we create them? And how do we embed the entire Bible?

Not so long ago this would have been the topic of a research paper. Fortunately, we now have prepackaged neural networks designed for this problem.

The first is Word2Vec, an extremely useful tool from our pals at Google. As its name suggests, Word2Vec is a neural network whose mission in life is to translate structured information — such as words — into embedded vectors.

The second tool is Sentence2Vec, also from Google. This basically does the same thing as Word2Vec, but using a deeper neural network and advanced techniques to handle larger sequences of text versus single words. For brevity we're going to focus on Word2Vec in our explanations below, but the same principles apply to both tools.

Word2Vec does a fine job stuffing words into vectors. How does it do that magic? To answer that we'll skip the math and the computer science of neural networks in favor of a more intuitive verbal approach. If you understand this explanation, then you'll gain a fundamental insight into how AI operates and how intelligence gets bootstrapped from data. However, if your brain is hurting, you're welcome to just take this on faith and go straight to the demonstrations in the next section.

Ready for an explanation? Here we go.

Neural networks were inspired by brain architecture and thus consist of layers of artificial software neurons. That's not as daunting as it sounds. At base, these networks are nothing more than stacks of vectors, wired together in a special way, where each number in a vector represents the current value of a specific neuron. When we train a neural network (that is, when it's learning), it's these numbers that get changed during the learning process.

Let's make this concrete. Imagine we wanted a neural network that can recognize the difference between a parrot and a crow. To do that we initialize a network to random values and show it a photo of a crow or parrot. The system then looks at this photo and makes a prediction — I see a parrot or I see a crow. Given the network is new and doesn't know anything yet, this prediction will be random. If the neural network predicts wrongly (it said parrot when it's really a crow), it will respond to the error by adjusting

its numbers slightly so that it's more likely to predict parrot when it next sees such a photo. It does the same thing if it makes a mistake in the other direction (predicting crow when it's really a parrot). In this way, if you show this network enough photos of crows and parrots, gradually the numbers inside the vectors converge towards states where the network gets really good at reliably predicting the right bird.

In short, neural networks are essentially just stacks of vectors, and the numbers in these vectors get changed when the network is learning a new task. Change them enough and in the right way, and they'll begin to represent concepts well enough to make accurate predictions.

Here's what's great about that: those vectors are *embeddings*. In this case they represent how to tell crows and parrots apart. In other words, neural networks automatically build embeddings when solving whatever prediction problem they've been given.

We want embeddings for our Biblical text. Neural networks create embeddings when they're solving problems. Hmmmm, maybe you can see where we're going here. If we can give a neural network a prediction problem involving the Bible's text, then the network will automatically create embeddings when it tries to solve that problem. Then, once the network finishes learning, we just copy those embeddings (embeddings being, you remember, nothing more than a fancy name for lists of numbers that represent a concept).

That's exactly what Word2Vec does. It creates an internal prediction game with whatever text its been given. But rather than return the prediction results of this game, Word2Vec returns the embeddings instead.

Word2Vec can use two prediction strategies, but the one we use goes like this: we give Word2Vec a bunch of text. The tool reads this text word by word. For each word, it tries to predict what context words are close around it within some window. Typically this window is about 10 words on either side of the current word being read. If it gets a prediction wrong it adjusts its vectors. Then it moves to the next word and repeats the process.

For instance, take a look at this text:

. . .

> Who is as the wise man? *and who knoweth the interpretation of a thing? a man's* **wisdom** *maketh his face to shine, and the boldness of his* face shall be changed.

Here the system is reading the word "wisdom" and we've italicized the context window. Given this current word — and what the neural network has learned so far — Word2Vec tries to guess all the words in this italicized window. It compares these guesses to the reality, and then adjusts its network to account for any misses. It then moves onwards to the next word, shifts the window to the right and tries to guess words in the new window. And so on until the end of the text.

By the end of this process Word2Vec has typically gotten pretty good at this silly game. That is, given a word, it can predict context words around it. In order to get good at these predictions, it's generated accurate embeddings for every word. Therefore, once Word2Vec has finished reading, all we have to do now is get a copy of those embeddings. We're done!

Word2Vec Word-Prediction Game

….. *a man's* **wisdom** *maketh his face* ….

Word2Vec Neural Network Word Embeddings

[133.504 475.321 … Many More Numbers … 762.343 889.498]
[343.334 483.868 … Many More Numbers … 190.343 134.902]

[929.343 104.941 … Many More Numbers … 584.323 939.498]
.
.
.
[872.873 382.432 … Many More Numbers … 948.901 893.909]
[590.498 575.321 … Many More Numbers … 762.343 189.883]
[483.309 377.354 … Many More Numbers … 547.649 780.292]
[133.504 475.321 … Many More Numbers … 364.541 584.447]

Word2Vec word embeddings are constantly improved as they try to predict context words. Once all the text is read, we copy those embeddings.

That's it? It might seem unlikely that such a simple algorithm could generate anything useful. But think about it: every time the network guesses and adjusts itself for any error, *everything* in the neural network gets updated. That means the representation of every word and concept in the Bible gets updated each time a new word is read. This allows the neural network to gradually reach beyond the window for each word, indirectly linking words to other words that are far removed elsewhere in the text. This eventually forms an intricate embedded web linking every concept to every other concept. And that's how its embeddings become so powerful.

All that's required is enough text. With enough information and the right techniques, intelligence emerges from raw data. As it turns out, the Bible is barely sufficient in this regard — we could have used far more text. However it's good enough for our purposes.

By the way, recall that I said earlier that similar embedded concepts tend to cluster together in high-dimensional space. Perhaps you can now see why. When Word2Vec is playing its prediction game, words that are similar will tend to have similar context words around them. For example, if the word is "parrot" or "crow", then it's roughly equally likely that "bird" might also be close by. Given that, their embeddings will be roughly similar as well, which is just another way of saying that they are close to each other in high-dimensional space.

With embeddings in hand, building our AI from here is a straightforward Programming 101 exercise. We just combine the embeddings into a form we can use and wrap the result with some infrastructure code. We then create a simple web UI so that we can

communicate with the AI. All of this is well within the realm of anyone with good skills in Python and Javascript. AI is more accessible than you might think.

The resulting project comes to less than 1,000 lines of new code. That's a ridiculously small footprint for any significant software effort, more on par with what you might find inside a thermostat or kitchen appliance. And yet this code is sufficient to do everything I'm about to demonstrate. Think about that the next time you use your waffle-maker.

First Impressions

To summarize, we've embedded every word, verse and book of the Bible into a set of high-dimensional spaces. That means that every concept has a coordinate, which means we're now in a position to ask questions.

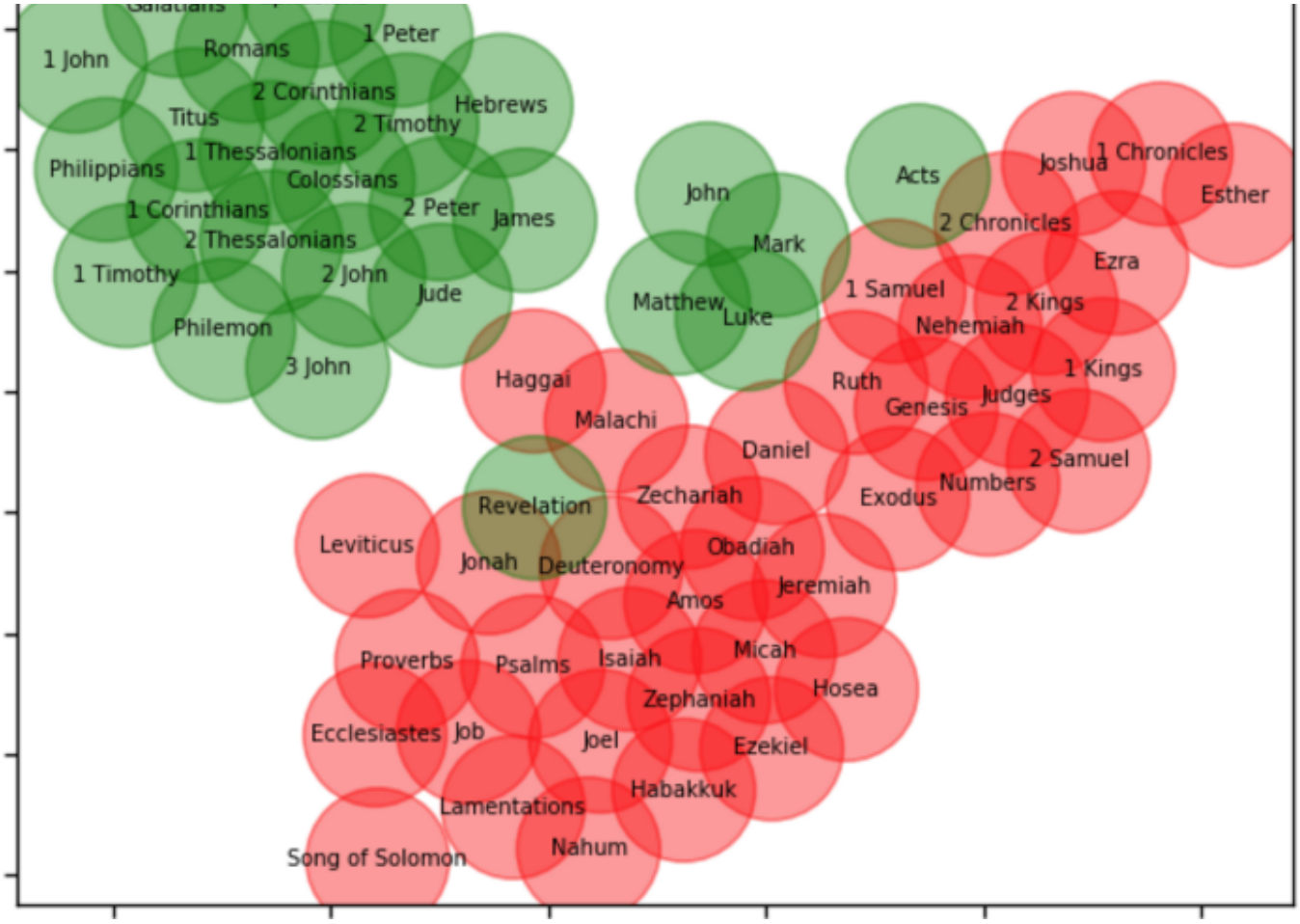
But before doing so, keep in mind that this system isn't privileged in any way. All that follows is literally just one AI's opinion. Other parameters and other approaches will lead to different perspectives. Also note that this is a strictly literalist interpretation of the Bible. In fact, this system is about as purely literal as one can get, given it knows absolutely nothing beyond what it sees in the text and the only text I gave it was the Bible. I considered adding more material (scriptural commentary of various sorts), but in the end decided that this would introduce biases and wasn't strictly necessary for this tutorial.

Enough background. Let's fire this sucker up and have some fun.

We'll start by asking the system to give us an overall visual impression. In its opinion, how are books of the Bible correlated? What basic patterns do they follow?

To do that, we project the embeddings of each book onto a 2D surface. We do this via another wonderful tool named t-SNE (t-distributed Stochastic Neighbor Embedding, no less), which takes high-dimensional vectors and approximates them into a 2-D graph. This generates a visual on how these book embeddings correlate. In this fashion we'll discover which books relate to other books and which are outliers.

Here's the result:



When I first posed this question I expected to get a weird answer. It's often a struggle to interpret AI results, particularly when first interacting with a new system of unknown abilities. I was therefore pleased when the response came back looking reasonably understandable, even familiar. The Old Testament (coded red) clumps together nicely together and there's a clear logic to the flow. Joshua, Ezra, 1 Chronicles, Esther start to the right and are sensibly grouped, as they all relate personal histories of leaders and prophets. Likewise, Song of Solomon, Ecclesiastes and Proverbs form the leftmost boundary, being more poetic and metaphorical texts. There's a beautiful unity to the Old Testament.

The New Testament (coded green) is somewhat less tame. In the top left we have a cluster representing most of the books. However, a few of the texts have broken from the pack and migrated into Old Testament territory. Part of this might be due to the flow of the Biblical narrative. Matthew, Mark, Luke, John and Acts form the first five books of the New Testament. Apparently they act as a conceptual bridge between the Old Testament and the New Testament. Our AI is picking up on this.

But what's up with Revelation? It has wandered off into the Old Testament wilderness on its own. The system clearly thinks of this text as being truly rooted in the Old Testament, even though it's the last book in the New Testament. Why? That's an area for future analysis by the AI.

Note there are no axis labels on the chart. This omission hints at a limitation in AI: it's opaque. Indeed, opacity — often called a lack of *introspection* — is a fundamental characteristic and challenge of the technology. How do we know and interpret what goes on inside? The math is clear enough, but the complexity is such that it often isn't reducible to narratives. Answers emerge from a black box and it's impossible to trace their genesis, even in principle. In this case, the circles in this diagram obviously have values — otherwise we couldn't plot the chart in the first place. However those values are relative and have no particular meaning outside of that. Consider: this chart is a 2D projection of high-dimensional spaces, which were in turn created by neural network sentence embeddings. What *could* the chart values even mean?

Also keep in mind that the system isn't deterministic and therefore changes over time and with new data. An element of randomness is characteristic of neural networks. These systems are not software as conventionally understood, where every step is coded and can be traced, but rather dynamic entities. The resulting intelligence can be impressive but also absolutely alien. Hence the term *artificial* intelligence.

Now let's get overviews of a different and interactive sort. Recall that we generated embeddings for every verse in the Bible. This means that we can now navigate the text in an original way, based off verse structure and concepts.

To illustrate that, try this tool. Select a particular book and enter a citation in chapter:number form. The AI then responds with the verses in the Bible that are, in its opinion, most similar in content and meaning to the verse you entered.

The system will respond with at least one verse and up to a ten maximum. For example, if you try Genesis 1:1 you'll get the following:

. . .

Genesis 1:1 In the beginning God created the heaven and the earth.

3 Responses

(1) Psalms 90:2 Before the mountains were brought forth, or ever thou hadst formed the earth and the world, even from everlasting to everlasting, thou art God.

(2) Genesis 2:4 These are the generations of the heavens and of the earth when they were created, in the day that the LORD God made the earth and the heavens,

(3) 1 Kings 8:27 But will God indeed dwell on the earth? behold, the heaven and heaven of heavens cannot contain thee; how much less this house that I have builded?

These are reasonable answers, like one might expect. However the system also delivers surprises and interesting twists.

For example I wondered what it would do with Ecclesiastes 9:11. That's a unique verse and I doubted there was anything remotely like it in the Bible.

How does our AI respond to the challenge? Let's find out:

. . .

Ecclesiastes 9:11 I returned, and saw under the sun, that the race is not to the swift, nor the battle to the strong, neither yet bread to the wise, nor yet riches to men of understanding], nor yet favour to men of skill; but time and chance happeneth to them all.

1 Response

(1) Job 36:19 Will he esteem thy riches? no, not gold, nor all the forces of strength.

The system found an unexpected verse, beautiful in its simplicity. A nice condensation of the central idea of 9:11. Note that the judgement the AI made didn't depend on

matching words, but rather matching the meaning of the text in a metaphorical fashion. Now we're starting to catch a glimmer on what our AI can do.

Note you can click on the chapter-verse link that comes with each result, and then the system will pivot and search on *that* new verse instead. In this way you can quickly follow a conceptual thread deep into the text, without having to go through the trouble of typing. That makes for a fast and responsive experience, and is an intriguing way to investigate the text.

Let's pause and consider how the system generates these answers. Recall that concepts which are similar — however the system defines that — embed closer together in high-dimensional space. That means their vectors will have some similar coordinates. Therefore, to find similar verses, the system just locates the embedded vectors that are closest to the target embedded verse. This is mathematically straightforward, meaning that our AI can quickly locate any number of matches from an embedded space.

The system then ranks these matches by likelihood. It always return the topmost match, and then goes down the list of other matches and returns those whose probabilities are above a certain likelihood threshold, up to a maximum of ten.

It's a bit addictive to leaf through the Bible in this fashion, tracing a line of thought and seeing how verses across the text correlate. It also provides another excellent perspective on how our AI conceptualizes the text.

Question and Answer Time

Let's now ask questions about Biblical concepts and people. To do that, we'll use this interface.

If you search for multiple terms, you can connect them with "+" and "-" operators. These behave pretty much like what you might expect. The + operator combines two terms, emphasizing what they have in common. The "-" operator subtracts one term from another, leaving a remainder and emphasizing the differences. Don't worry, we'll walk through examples that clarify this.

Now we just need some questions to ask. I suggest we start right at the top.

What's the nature of God? That is, what are His characteristics or related entities?

Here's how our AI responds:

. . .

(god)

(1) glorify
(2) magnified
(3) evermore
(4) redeemer
(5) saviour
(6) holiness
(7) glorified
(8) endureth
(9) revealed
(10) marvellous

A wonderful list. What's particularly striking is that God consists of abstractions. There are no people in this list or concrete references to the world. God is worshiped, a redeemer, and has been around forever.

Now let's see what our AI thinks about Jesus.

. . .

(jesus)

(1) christ
(2) john
(3) baptized
(4) grace
(5) answering
(6) peter

(7) resurrection
(8) risen
(9) baptism
(10) believed

Very different. Jesus is the Christ who was baptized and resurrected. But He also lived in the common world as a human being, being closest to Peter and John.

Now let's do the world's first theological computation. If we removed Jesus' characteristics from God, what would we have left? To do that, all we need do is to literally subtract Jesus from God like so:

. . .

(god)-(jesus)

(1) images
(2) desolate
(3) idols
(4) groves
(5) burned
(6) houses
(7) cut
(8) scatter
(9) gods
(10) countries

That's quite a transformation. Removing the qualities of the Son makes God a scarier animistic entity, involving images and idols, sacred groves and unpleasant experiences.

So what if we go the other way and take God out of Jesus? Let's find out:

. . .

(jesus)-(god)

(1) john
(2) peter
(3) simon
(4) mary
(5) disciples
(6) james
(7) kish
(8) certain
(9) meshullam
(10) sepulchre

When we do that, Jesus still has his friends and his mother. However Jesus is no longer the Christ and won't be resurrected. He's now just a teacher and a human.

Let's stretch things a bit. What concepts or people do Noah and Jesus have in common?

. . .

(noah) + (jesus)

(1) resurrection
(2) elisabeth
(3) abrahams
(4) hannah
(5) begotten
(6) seth
(7) sarah
(8) zacharias
(9) hagar
(10) baptist

Resurrection? Our AI claims there is something in Noah's story that implies or is in common with Jesus' ascent to heaven! That seemed unlikely to me, but when I googled the topic I discovered that this is actually a known concept.

In addition to that, Noah and Jesus have in common a number of Old Testament prophets and wives, and they were both of course begotten by women. I've found that this is an amusing pattern — our AI, for whatever reason, tends to like reminding us that everyone has a mother.

I've never really understood the concept of The Holy Ghost. Maybe the system can help:

. . .

(holy) + (ghost)

(1) profane
(2) sanctify
(3) profaned
(4) approach
(5) sanctified
(6) choose
(7) partakers
(8) bless
(9) holiness
(10) song

In archaic English, "profane" means commonplace or of this world. It seems the Holy Ghost inhabits our world, but is nevertheless sanctified and holy. This spirit is apparently some sort of bridge that one can approach and partake in.

Lastly, not every question must involve weighty topics. Everything that's in the Bible is fair game. So you can query anything you'd like as long as it's mentioned somewhere in the text. Very randomly, let's find out how forests and war intersect:

. . .

(forest) + (war)

(1) weapons
(2) bows
(3) tumultuous
(4) spears
(5) besiege
(6) innumerable
(7) beat
(8) tall
(9) riders
(10) holds

It seems that Biblical forests were good for making weapons and ships (holds), and were also instrumental in sieges (wood for defensive walls?). As for *tall* and *riders*, I will leave that to your own interpretation.

There are many limitations to this interface. This is partly due to the lack of data. As mentioned earlier, the Bible is barely long enough to meet the bar. Some important concepts or people — such as Eve — are named only once, making it more difficult for the AI to get a handle on those ideas. Therefore, when dealing with rarer terms, you might find that the system has less insight (or crazier insight) compared to more common ideas. In addition, in this simple AI we make no provision for phrases ("The Holy Ghost" etc). Instead we approximate such phrases via the '+' symbol, which can be good enough given the right data, but certainly isn't optimal. These deficiencies could be addressed by more data and more advanced AIs.

Next Steps

This article has demonstrated what's possible with a simple and basic AI, involving very little new code and using off-the-shelf tools.

But we're just getting started, as this work could be the foundation for many other useful things. Some of these ideas might surprise you. For example, did it occur to you that our humble project could be turned into a universal multilingual language translator? Recall that we embedded every word in the Bible into a high-dimensional space. We happened to do that with an English-language Bible, but we could just as easily have done that with Bibles written in Spanish or any other language. Whatever the language, the

resulting embeddings will be almost the same. That is, each embedded word in English will be very close to the same embedded word in Spanish. That means, given an English word, an AI could translate that into Spanish by simply looking up the English word's coordinates in a Spanish-language space. Instant language translation! In fact, the online translators available via your browser work very much like this. Embeddings have so many uses.

Alternatively, our embeddings and associated code could also be used as the *inputs* into another and much more advanced neural network. In effect this AI could then take our work and add further intelligence, creating much more powerful and interactive entities. These might range from grammar checkers to a full-blown virtual theological oracle.

Lastly I'd like to mention another possible and very serious future application. This involves General AI (GAI) — the kind of AI seen in science fiction movies, the kind that everyone really worries about. Such an entity would posses a generalized intelligence, suitable for addressing any problem at a superhuman level. The arrival of true GAI would be the greatest event in human history, but it would also be an extraordinarily dangerous moment. Humanity eventually wouldn't be able to compete. Given that, how do we know that the GAI's interest will align with our own? We've already noted the opacity of these systems. How can we be 100% sure that it will behave in a civilized manner?

Over the long-term it would be impossible to control a GAI via command or physical means. To guard against disaster, the GAI instead will need to internalize a strong set of ethics. To that end, computational theology might be considered a first step towards computational ethics. In my view this is a necessity. If we don't provide an ethical and moral framework any sentient system will likely make up it's own, and that's probably not a framework that you or anyone else would approve of.

These are deep waters and there are plenty of sharks. Simple direct solutions wouldn't suffice. I'm sure you agree that just giving a GAI a sacred text and telling it to "obey all the rules you discover in here" would be a fantastically bad idea. If we did this with the Old Testament, I doubt humanity would survive much beyond Leviticus. We can never count on a GAI coming to the same conclusions that we do, no matter how obvious they seem to us. Therefore such an effort would require extraordinary care.

I hope this tutorial has provided some fresh perspectives and insights on AI. Working with this project has given me a better understanding of these techniques. And, as a bonus, I've certainly learned much more about the Bible.

Machine Learning Religion Spirituality Artificial Intelligence