Chapter IX

Artificial Intelligence Thinking in K-12

David S. Touretzky

Computer Science Department

Carnegie Mellon University

Pittsburgh PA 15213

dst@cs.cmu.edu

Christina Gardner-McCune

Computer & Information Science & Engineering Department

University of Florida

Gainesville, FL 32611

gmccune@ufl.edu

PRE-PUBLICATION DRAFT VERSION

Artificial Intelligence Thinking in K-12

Abstract

The increasing contributions of AI technologies to everyday life, and the social upheavals on the horizon as these technologies develop further, have sparked interest in AI education worldwide. The AI4K12 Initiative is developing U.S. national guidelines for what K-12 students should know about AI, and what they should be able to do with it. The guidelines address four grade bands: K-2, 3-5, 6-8, and 9-12, and they are organized around "five big ideas in AI": Perception, Representation & Reasoning, Learning, Natural Interaction, and Societal Impact. In this chapter we consider the key insights we want students to gain into the big ideas in AI, and how learning about AI may influence other aspects of their educational experience.

History of the Five Big Ideas in AI

The "Five Big Ideas in AI" were inspired by the 2017 CSTA Computer Science Standards, which are organized around five big ideas in computing (CSTA, 2017). Those ideas are: (1) Algorithms & Programming, (2) Computing Systems, (3) Data & Analysis, (4) Impacts of Computing, and (5) Networks and the Internet. Unfortunately, although artificial intelligence is an important branch of computer science, the standards contain only two sentences about AI, both in the 11-12 grade band, as shown in Figure 9.1. Until recently, AI was considered too advanced for younger students.

3B-AP-08	11-12	Describe how artificial intelligence drives many software and physical systems.	>	Algorithms & Programming	Algorithms	Communicating
3B-AP-09	11-12	Implement an artificial intelligence algorithm to play a game against a human opponent or solve a problem.	>	Algorithms & Programming	Algorithms	Creating

Figure 9.1. References to AI in the 2017 CSTA Computer Science Standards.

The Five Big Ideas in AI are a way to introduce teachers, parents, and students to the essential concepts and major issues of a field often confused with science fiction (Touretzky et al., 2019). We will argue here that studying AI can teach students about more than technology; it can help them better appreciate the complexity of humanity.

Each of the Five Big Ideas is described by a key phrase and a one sentence statement; see Figure 9.2. In a poster we published in 2019, each statement was unpacked in a paragraph of explanatory text. This poster has since been translated into Chinese, Korean, Spanish, Portuguese, Hebrew, and Turkish, all available on the AI4K12.org web site. In the guidelines, each big idea is broken down into a set of concepts and skills that form the rows of a table called a grade band progression chart. The columns are the four grade bands, and the cells define what students in that grade band should know about and be able to do with that concept or skill. At the time of this writing, the draft grade band progression chart for the first Big Idea, Perception, has been released for public comment and is currently undergoing revision. An excerpt is shown in Figure 9.3. The draft for the third Big Idea, Learning, is about to be released.
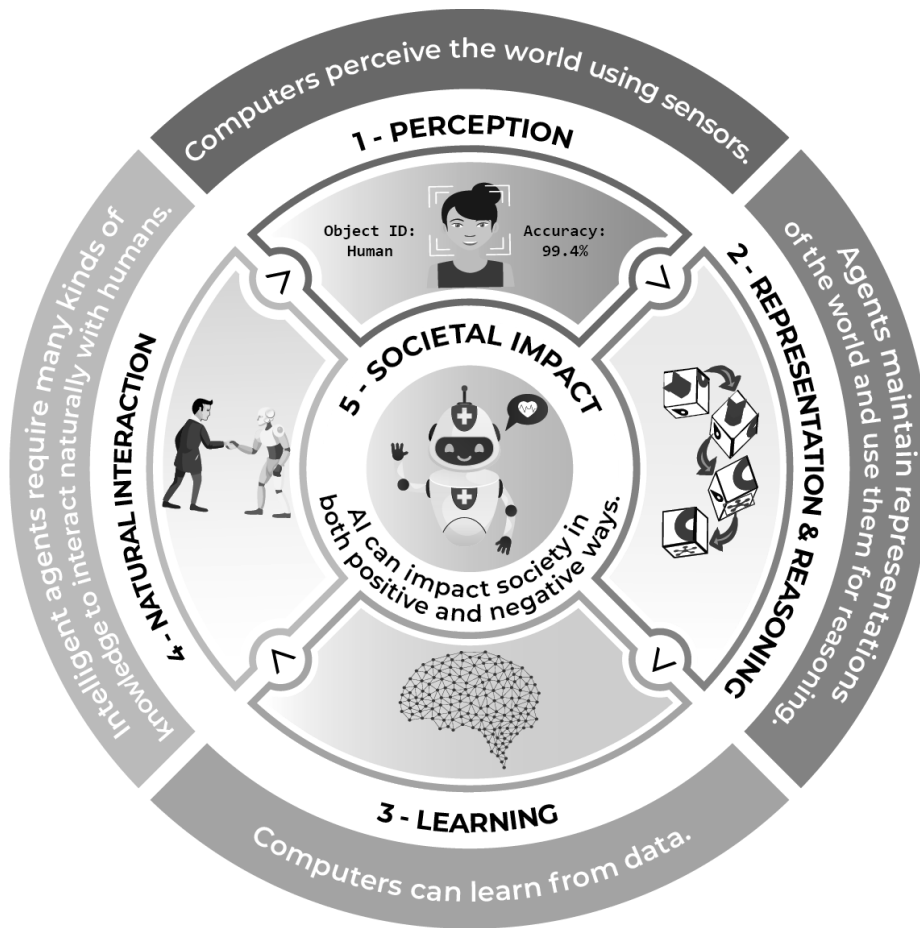
Figure 9.2: The Five Big Ideas in AI graphic from AI4K12.org.

Big Idea #1: Perception	*Computers perceive the world using sensors.*	*Perception is the extraction of meaning from sensory information using knowledge.*	**The transformation from signal to meaning takes place in stages, with increasingly abstract features and higher level knowledge applied at each stage.**	**LO** = Learning Objective: what students should be able to do. **EU** = Enduring Understanding: what students should know.
Concept	**K-2**	**3-5**	**6-8**	**9-12**
Sensing (Living Things) 1-A-i	**LO:** Identify human senses and sensory organs. **EU:** People experience the world through sight, hearing, touch, taste, and smell.	**LO:** Compare human and animal perception. **EU:** Some animals experience the world differently than people do. **Unpacked:** Bats and dolphins use sonar. Bees can see ultraviolet. Rats are have no color vision; dogs are red-green colorblind. Dogs and rats can hear higher frequences than humans.	**LO:** Give examples of how humans combine information from muliple modalities. **EU:** People can exploit correlations between senses, such as sight and sound, to make sense of ambiguous signals. **Unpacked:** In a noisy environment, speech is more understandable when the speaker's mouth is visible. People learn the sounds associated with various actions (such as dropping an object) and can recognize when the sound doesn't match their expectation.	N/A -- for AI purposes, this topic has already been adequately addressed in the lower grade bands. *Other courses, such as biology or an elective on sensory psychology, could go into more detail about topics such as taste, smell, proprioception, and vestibular organs.* *Possible enrichment material: look at optical illusions (Muller-Lyer illusion, Kanizsa triangle) and ask which ones are computer vision systems also subject to.*
Sensing (Computer Sensors) 1-A-ii	**LO:** Locate and identify sensors (camera, microphone) on computers, phones, robots, and other devices. **EU:** Computers "see" through video cameras and "hear" through microphones.	**LO:** Illustrate how computer sensing differs from human sensing. **EU:** Most computers have no sense of taste, smell, or touch, but they can sense some things that humans can't, such as infrared emissions, extremely low or high frequency sounds, or magnetism.	**LO:** Give examples of how intelligent agents combine information from multiple sensors. **EU:** Self driving cars combine computer vision with radar or lidar imaging. GPS measurement, and accelerometer data to form a detailed representation of the environment and their motion through it.	**LO:** Describe the limitations and advantages of various types of computer sensors. **EU:** Sensors are devices that measure physical phenomena such as light, sound, temperature, or pressure. **Unpacked:** Cameras have limited resolution, dynamic range, and spectral sensitivity. Microphones have limited sensitivity and frequency response. Signals may be degraded by noise, such as a microphone in a noisy environment. Some sensors can detect things that people cannot, such as infrared imagery, or ultrasonic sounds.
Sensing (Digital Encoding) 1-A-iii	N/A	**LO:** Explain how images are represented digitally in a computer. **EU:** Images are encoded as 2D arrays of pixels, where each pixel is a number indicating the brightness of that piece of the image, or an RGB value indicating the brightness of the red, green, and blue components of that piece.	**LO:** Explain how sounds are represented digitally in a computer. **EU:** Sounds are digitally encoded by sampling the waveform at discrete points (typically several thousand samples per second), yielding a series of numbers.	**LO:** Explain how radar, lidar, GPS, and accelerometer data are represented. **EU:** Radar and lidar do depth imaging: each pixel is a depth value. GPS triangulates position using satellite signals and gives a location as longitude and lattitude. Accelerometers measure accleration in 3 orthogonal dimensions. **Unpacked:** Radar and lidar measure distance as the time for a reflected signal to return to the transceiver. GPS determines position by triangulating precisely timed signals from three or more satellites. Accelerometers use orthogonally oriented strain gauges to measure acceleration in three dimensions.

Figure 9.3: Part of the draft grade band progression chart for Big Idea #1: Perception. The rows list concepts and skills; the columns are the four grade bands.

The ordering of the Five Big Ideas progresses from narrow areas of low-level processing (perception) to broad, high level level topics (societal impact). But they are not meant to be covered in sequence. Some curriculum developers have done this, such as ReadyAI's "AI + ME" overview (ReadyAI, 2019). But there are many other ways to survey AI, such as by examining different application areas. A module on self-driving cars could touch on all five of the big ideas.

Big Idea #1: Perception

Big Idea #1, Perception, says "Computers perceive the world using sensors." The initial guidelines for Big Idea #1 start with a discussion of computer sensors, which connects with the computer science standards for computer hardware (under Computer Systems), and a discussion of human sensory capabilities, which naturally connects with human biology. But sensing isn't what this big idea is about.

The first major insight we want students to have is that perception is more than sensing. Specifically: "Perception is the extraction of meaning from sensory signals, using knowledge." An automatic door at a supermarket has a sensor, but it does not perceive anything. The signal from the pressure pad or ultrasonic transducer is too impoverished to carry much information, and the response of the door too simplistic to require any "meaning" beyond the raw signal. We want students to understand that not all devices exhibit intelligence. We would not be enjoying YouTube videos of wildlife wandering through supermarket aisles if their automatic doors could properly perceive who (or what) was entering.

If the extraction of meaning from sensory signals requires knowledge, what does that knowledge look like? In the case of speech perception this question leads to an examination of the many levels of language, starting with articulatory gestures (the motions made by the tongue, lips, and vocal tract), and progressing to phonology (sounds), morphology (word stems, prefixes, and suffixes), prosody (stress and intonation), syntax (grammar), and semantics (meaning). These are sophisticated concepts, but even young children can discuss the phonetic inventory of their native language, and can understand why an intelligent agent like Siri or Alexa might have

trouble understanding someone who spoke with an accent or had poor diction.

The second major insight into perception we want students to come away with is what we call the *abstraction pipeline*: "The transformation from signal to meaning takes place in stages, with increasingly abstract features and higher level knowledge applied at each stage." In the case of speech this progression is inherent in the structure of language, and early speech recognition systems actually implemented the pipeline as a collection of distinct modules proceeding from the raw acoustic signal to phonemes, words, phrases, and meaning. In more recent systems based on deep neural networks there are many more stages of processing, and different types of knowledge co-exist across multiple levels. But even in these messier neural net implementations there is a general progression from more local, signal-based information to more global, meaning-based information as one moves through the layers.

Visual perception differs from speech perception in that language is something produced by *humans* for the purpose of transmitting meaning, while vision is concerned with *constructing* meaning by sensing natural phenomena such as reflection and occlusion. The abstraction pipeline for vision starts with pixels and ends with 3D scenes, but what lies between is a complex mix of edges, contours, boundaries, surfaces, parts, shadows, reflections, and objects. Marr (1982) called this the 2 ½ D sketch. The knowledge required to derive these representations is innate in humans and not easily articulated explicitly in a computer program.

The *abstraction* pipeline is a wondrous thing. Information flows backward as well as forward, e.g., knowledge of the vocabulary of a language can influence the perception of

ambiguous sounds, and knowledge about the shapes of objects can influence the interpretation of edges in a scene. Human perceptual processes are far from fully understood at present. Studying how AI attempts to mimic these processes offers a new route to appreciation of human perception.

How much of this can be conveyed in K-12 is still an open question, but at least the early stages of the pipeline can be exposed to students through interactive demos. Low-level vision can be *illustrated* by showing the real-time output of vertical and horizontal edge detectors (Figure 9.4) applied to webcam images. Low-level auditory perception can be shown with real-time spectrograms (Figure 9.5) and pitch trackers.
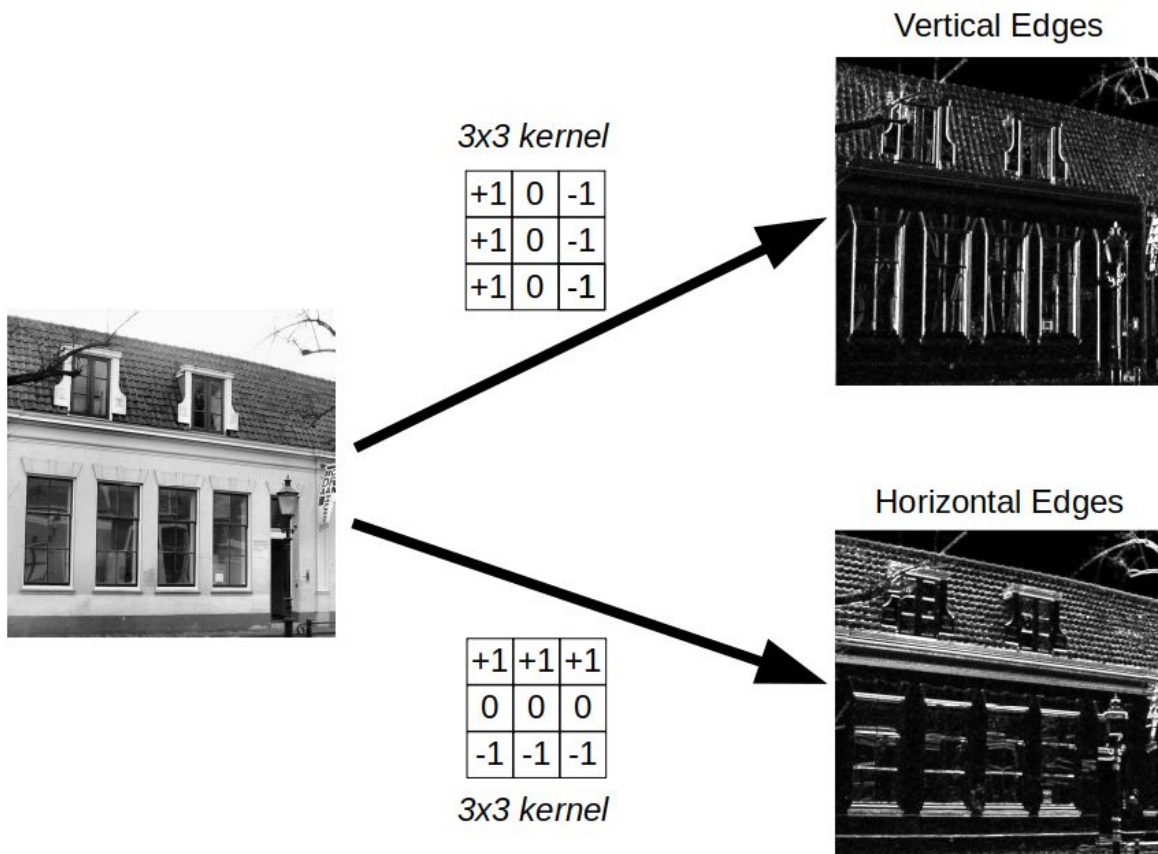
Figure 9.4: Edge detection is one of the first stages of computer vision. Vertical and horizontal edges are detected by convolving 3×3 kernels with the image.
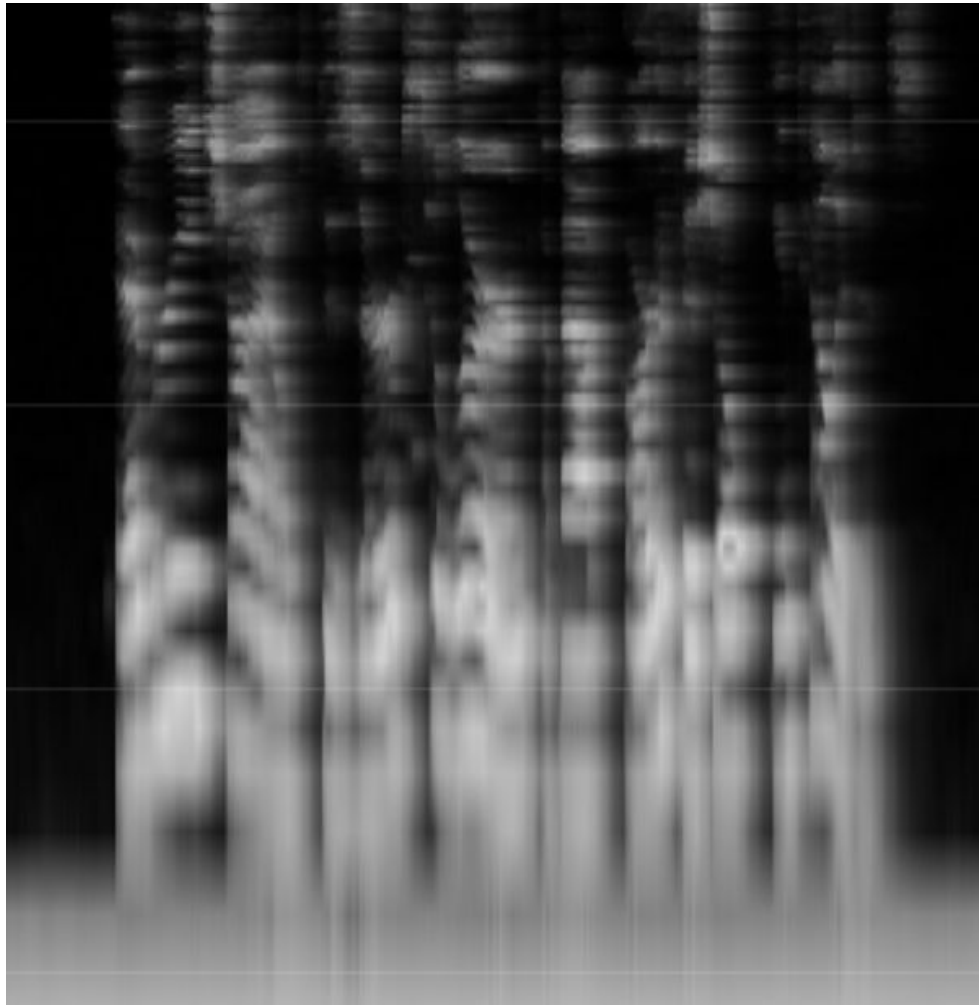
Figure 9.5: Real-time spectrogram of the first author saying "Every child deserves to learn about artificial intelligence." The vertical axis is frequency; the horizontal axis is time; shading indicates the amount of energy in that frequency band. Created with https://creatability.withgoogle.com/seeing-music.
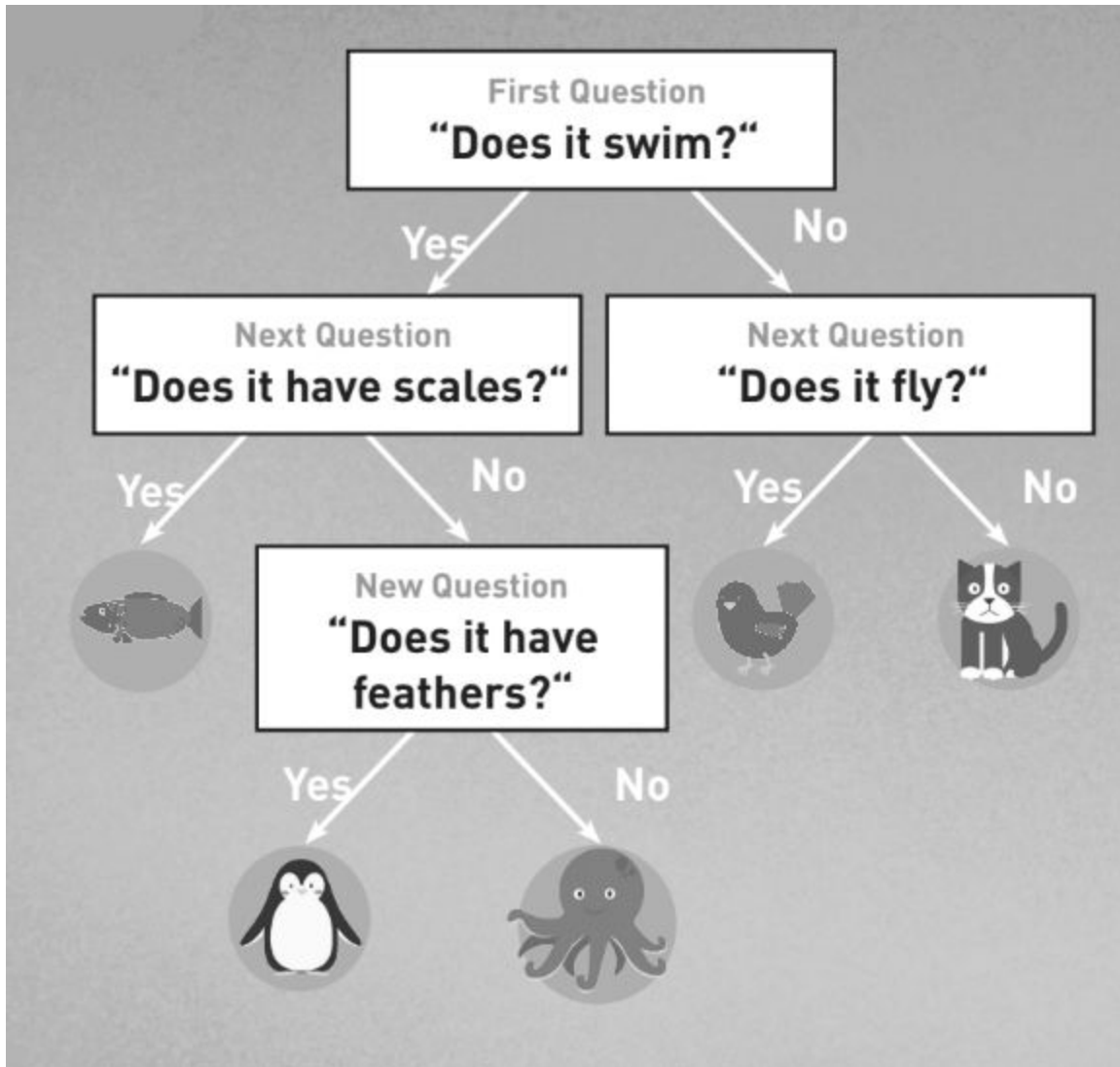
Big Idea #2: Representation and Reasoning

Big Idea #2 states that "Agents maintain representations of the world and use them for reasoning." In computer science terms, representations are data structures and reasoning is

performed by algorithms. But how can the concept of representations be explained to children in the lower grades who are not yet familiar with data structures? Maps are a good place to start. Even young children can grasp the concept of a map being a representation of a place. They understand that the map is not the territory, that maps abstract away many details, and that maps follow certain notational conventions such as the way that roads or buildings are depicted. Having children construct a map of their house, their school, or their neighborhood brings these ideas home. Children can also appreciate that using a map to plan a route is a kind of reasoning, and that a self-driving car must be doing a similar kind of reasoning. Thus, using maps, representation and reasoning can be made accessible even in K-2.

A good next step in exploring representation and reasoning, appropriate for grades 3-5, is the *decision* tree. This can be introduced via the "guess the animal" game, where the goal is to guess the animal a person is thinking of by asking a series of yes or no questions such as "Does it swim?" or "Does it fly?" The questions form the non-terminal nodes of a binary tree, and the terminal nodes are the animals (Figure 9.6). In playing this simple game children encounter fundamental concepts in representation and reasoning that they will be exploring further for years to come. First, the decision tree is drawn on the board so everyone can follow the reasoning process. This introduces students to the notion of tree structures, and serves as a simple formalism for encoding knowledge. Second, the procedure for playing the game is formalized. One always starts at the root node. Upon arriving at any nonterminal node, that node's question must be asked, and one then follows either the "yes" or "no" branch to reach the next node. Upon arriving at a terminal node, one states the animal associated with that node and waits to see if the guess is correct. Asking students to explain this procedure in their own words

prompts them to think about how the reasoning algorithm works.

Figure 9.6: Decision tree learning in the AI+ME book "Big Idea #2 – Representation and Reasoning: How AI Makes Choices" from ReadyAI. Used with permission.

Another valuable aspect of the guess the animal game is the procedure for growing the tree. If one reaches a terminal node and guesses "penguin", but the correct answer is "octopus", one has to obtain two pieces of information: (1) what question distinguishes between a penguin and

an octopus, and (2) what is the correct answer for an octopus. The decision tree can then be updated by replacing the "penguin" terminal node with a non-terminal node containing the new question, "Does it have feathers?". Penguin and octopus become its two children. Following this *procedure*, especially when they choose the animals and questions themselves, gives children a feeling for how human knowledge can be encoded in a data structure, and how computers can learn.

A key insight we want students to have about this big idea is the interdependence of representation and reasoning. Reasoning algorithms need something to reason with, and representations are pointless if we have no way to put them to use. Consider the map example from earlier: the map representation needs a path planning algorithm to find a route between two locations. It's also important to understand that representations are not just the input to an algorithm, they may also be constructed by the algorithm. The route constructed by the path planning algorithm is another representation. Likewise, game playing programs need another common AI representation, the search tree, to keep track of alternative moves as they search for the move that will lead to a winning game. The search tree is neither an input nor an output. It is constructed by the search algorithm as it searches. We express the representation/reasoning duality as follows: "Representation drives reasoning, and reasoning algorithms manipulate representations."

Older students can be introduced to a taxonomy of reasoning types to help them understand the variety of ways AI is used to make decisions. Classification and prediction (regression) problems are the most common applications of neural networks, although these problems can

also be approached symbolically. Combinatorial search is one of the oldest parts of classical AI and still very important. Other reasoning approaches include logical deduction and theorem proving, constraint satisfaction, task planning, and numerical optimization. Some of these topics are too advanced for K-12, but it may be possible to provide a taste. For example, doing inference by resolution theorem proving using first order predicate calculus is a topic for undergraduates, but we might give students in 9-12 a taste of logical inference by looking at how a computer can handle syllogisms, such as the classic "all men are mortal; Socrates is a man; therefore Socrates is mortal."

Symbolic vs. Feature Vector Representations

While much of the recent progress on the difficult problems of speech recognition, computer vision, and machine translation has resulted from advances in neural network technology, symbolic representations remain important, as evidenced by the resources Google and other large corporations have devoted to constructing knowledge graphs (Noy et al., 2019). The knowledge panel displayed on the right hand side of the screen in a Google search for "Thomas Jefferson" or "kiwi fruit" is generated from the Google knowledge graph. Hand-crafted symbolic representations used in classical AI are certainly easier to explain to children than the feature vector representations constructed by neural networks. But what should they understand about feature vector representations? In the remainder of this section we offer some speculation on how feature vector representations might influence students' views about word meanings.

Dictionaries and thesauruses are our traditional codifications of the meaning of words. With 600,000 words spanning 1000 years of usage, the Oxford English Dictionary (OED) is a landmark intellectual achievement, billing itself as "the definitive record of the English

language" (Oxford University Press, 2020). Dictionaries typically include usage examples --
often famous quotations -- that help put words into context. The OED contains 3 million
quotations. All of this material was compiled over many years by committees of scholars.
Similar efforts exist for other languages, e.g., a special commission composed of members of the
Académie Française produces the Dictionnaire de l'Académie Française, endorsed by the French
government. Dictionaries are important cultural artifacts and are the original hand-crafted
symbolic representations of words. But it is not easy for a computer to reason with this type of
representation.

Computing technology has offered our culture a new type of word representation that now
powers many natural language applications. This *feature vector encoding*, also known as a *word
embedding*, represents each word as a point in a high dimensional abstract space. To understand
this encoding it is helpful to first consider a less abstract example. The following is inspired by
the description of the word2vec family of models in Mikolov et al. (2013).

Suppose we want to represent the words "man", "woman, "boy", "girl", ""king", "queen",
"prince", and "princess". Imagine a three-dimensional space where the x coordinate encodes
gender, the y coordinate encodes age, and the z coordinate encodes royalty (Figure 9.7). Each of
our eight vocabulary words can be mapped to a unique point in this space, e.g., "man" might be
(0,1,0), and "princess" might be (1,0,1). Euclidean distance in this space can serve as a heuristic
for semantic similarity, allowing us to infer that "man" is semantically closer to "woman" than to
"princess". We can go on to embed additional words in this space, even without adding more
dimensions. "Son" would likely be close to "boy", although less definitive as to age, so perhaps

its coordinates would be (0, 0.3, 0). "Parent" is gender neutral but an adult, with no implication of royalty, so it might map to (0, 0.5, 0). And so on.
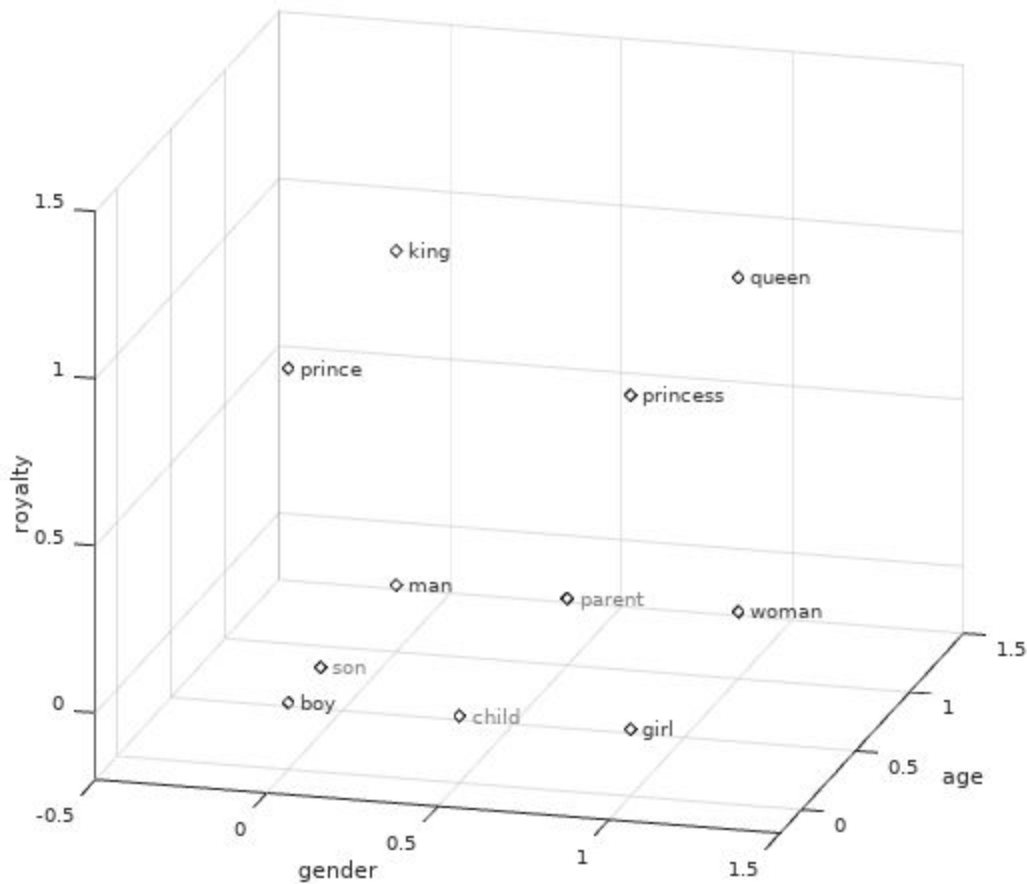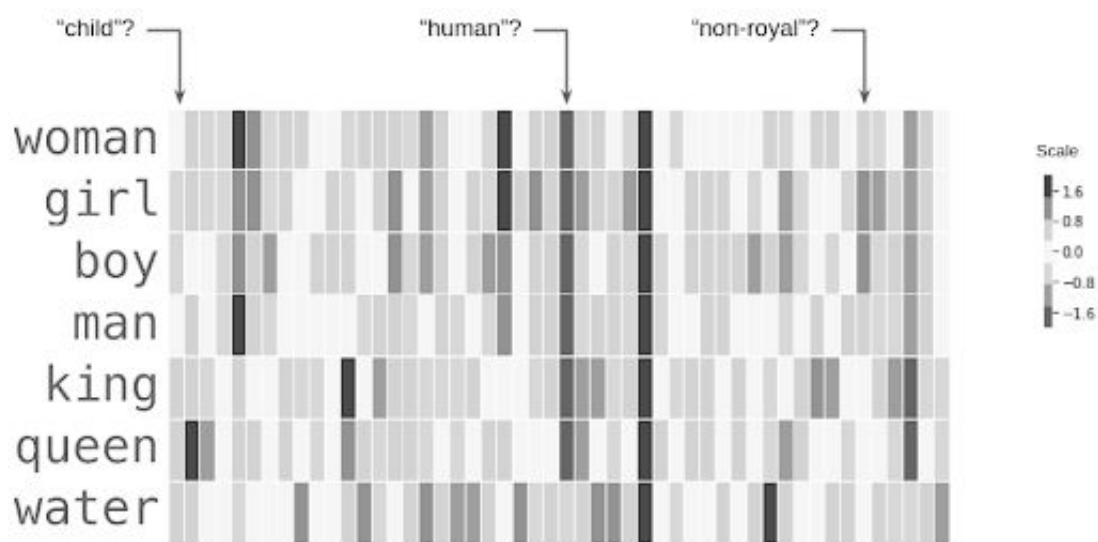

Figure 9.7: Representations of words as points in a 3D semantic space.

Mikolov et al. (2013) showed that in addition to providing a similarity heuristic, feature vector encodings admit a simple type of analogical reasoning by vector arithmetic. Subtracting the vector for "queen" (1,1,1) from the vector for "woman" (1,1,0) yields a vector (0,0,-1) that removes the royalty attribute from a word. Adding this vector to "prince" (0,0,1) yields "boy" (0,0,0). Adding it to "king" (0,1,1) yields "man" (0,1,0). Similarly, subtracting "man" from

"boy" and adding the result to "parent" yields (0.5, 0, 0), which is a plausible encoding for "child".

Representing a larger vocabulary requires a higher dimensional feature space. Rather than designing those features by hand, they can be created using machine learning, specifically neural networks. We don't have a convenient way to train the network directly on word meanings, but since words with similar meanings tend to occur in similar contexts, it turns out that training the network to predict what words are likely to co-occur with a given word is an effective proxy for meaning. This approach captures more than pure syntactic and semantic features; it also captures information about usage, e.g., which adjectives are typically applied to which nouns.

Figure 9.8: Feature vector representations of words in word2vec. Figure modified from (Alamarr, 2019).

Unlike the carefully constructed dictionary definitions produced by human experts, feature vector representations are somewhat arbitrary. They depend on parameters such as the size of the

context window (the number of words before and after the word whose features are being learned), the vocabulary set, the dimensionality of the feature space (number of units in the neural network's hidden layer), and the training corpus. Even if all these parameters are held constant, two separate runs of the learning algorithm will produce different representations due to the randomness of the network's initial weights. Heuristics used to speed training also influence the vector representation. And these vectors are not easily interpretable by humans, although one can sometimes find correlations between vector elements and semantic attributes by comparing the representations of several words, as in Figure 9.8 above.

Despite their lack of definitiveness, statistical feature vector representations have significant practical uses. For example, they can be used to disambiguate homophones during real-time speech recognition. Tell a chatbot you want "two coffees, not too hot, to go" and it will get every word right. Neural machine translation systems use feature vectors as their input and output encodings, as do some question answering systems such as Siri, and machine translation applications such as Google Translate.

There are already simple online demos that allow people to explore word2vec vocabularies and experiment with the analogy via vector arithmetic. Demos that are friendlier to K-12 users will surely follow. eCraft2Learn, a children's AI programming framework built on top of Snap!, recently added blocks for working with feature vectors (Kahn and Winters, 2020). Allowing students to experiment on their own with feature vector representations is the 21st century version of teaching them to explore a dictionary: it will enrich their appreciation of language. It will also give them insight into the workings of the AI systems they interact with in their daily

lives.

Big Idea #3: Learning

Big Idea #3 says "Computers can learn from data." It's important to distinguish human learning from what the computer is doing, so the guidelines begin with a comparison. Machine learning mostly follows one of two approaches: finding patterns in data, or optimizing behavior based on trial and error. Humans do those things too, but they also learn in other ways, such as by being told, by observing others, by asking questions, by experimentation, and by making connections to past experience. Human learning, because it is part of a larger cognitive architecture, is general and flexible, while machine learning is accomplished by specialized algorithms and focuses on performing a specific task.

Arthur Samuel, author of the first AI checkers playing program, is credited with coining the term "machine learning" in 1959. A definition often attributed to Samuel is that machine learning is a "field of study that gives computers the ability to learn without being explicitly programmed". He didn't actually write those words, but they convey the gist of his thinking[1]. Our take on this for the K-12 audience is: "Machine learning allows a computer to acquire behaviors without people explicitly programming those behaviors." Another way to think about it is that machine learning is a way to *construct* a reasoner. So humans program the learning algorithm, the learning algorithm constructs a reasoner with the desired behavior, and the reasoner is then employed in some task such as recognizing cats in images or deciding whether an email is spam.

One of the things we want students to be able to do is train a reasoner themselves. Several

tools allow children to train an image classifier on small numbers of examples using deep neural networks and transfer learning. Probably the best known tool is Google's Teachable Machine, which conveniently runs in the browser. The demo can use images captured from a laptop's camera, and doesn't require any programming. Similar capabilities now exist in children's programming frameworks such as App Inventor, Cognimates (based on Scratch), and eCreaft2Learn (based on Snap!). Using a tool like Teachable Machine, students can train a classifier to recognize a thumbs up gesture, a peace sign, and a "no gesture" condition. They can then measure its accuracy on new images and experiment with adding more varied training examples to help it perform better. This makes a compelling educational experience for adults as well as children. It's a great way to approach machine learning, but it has some limitations which need to be addressed through other activities.

To enhance their understanding of machine learning, we would like students to experience what it *feels* like to acquire a concept by finding patterns in data. The problem with training Teachable Machine to recognize cats or thumbs-up gestures is that we start out already knowing those concepts, even if the machine does not. Training a classifier on familiar concepts cannot help one experience what it's like to be the trainee. To address this, the guidelines have students play the role of machine learner for concepts they don't already know.

This exercise can be done as early as K-2 by showing labeled examples of cartoon creatures and asking students to figure out a rule that predicts the labels. For example, the images could be of cartoon fish of various colors, with different shaped heads, bodies, fins, and tails. The labels could be "eats seaweed" and "doesn't eat seaweed". Labeled instances would be presented one at

a time, and after seeing a sufficient number the students could begin positing what the pattern is that predicts a fish eating seaweed. A simple case would be that it's the purple fish that eat seaweed. More challenging cases might involve a conjunction of features (only purple fish with pointy heads eat seaweed), or for older students, a negated value (purple fish except those with small tails), a disjunction of conjunctions (purple fish with pointy heads or orange fish with small tails), or something even more complex.

In later grade bands students are asked to simulate learning algorithms in more detail. For example, in 3-5, instead of verbally stating a classification rule they may be asked to construct a decision tree, where each node tests a single feature such as color or head shape. In 9-12 they may be asked to train a classifier or predictor to fit a set of noisy training points by turning knobs to adjust parameters, eyeballing the quality of the fit. Such a model might predict a person's height given their age, or the price of a used car given its mileage. For a linear model y=mx+b they would adjust the slope and intercept, but they could also train nonlinear models such as logistic functions or cubic polynomials the same way.

Changes in Internal Representations

Another insight we want students to have is that "Learning of new behaviors is brought about by changes in internal representations." In other words, what the learning algorithm is doing is not magic; it is simply adjusting a data structure. This is the second drawback to Teachable Machine and similar transfer learning tools: they are black box demos whose internal representations are unobservable. Given the complexity of deep neural network representations, even if there were a practical way to display them, it's not clear how their hidden layer activations could be made interpretable by non-experts. There is, however, interesting work on giving qualitative insights

into what these networks are doing, such as displaying which areas of a scene a network is attending to, or finding the optimal stimulus for triggering a learned feature detector. As our methods for analyzing deep neural networks improve, the way we teach them will evolve.

For now, we advocate approaching "changes to internal representations" using hand simulations of symbolic learning applications. We can help students recognize that the decision trees they built in 3-5 or the parameter values they adjusted in 9-12 are the internal representations that learning algorithms manipulate. We can also draw on some "glass box" machine learning tools that disclose their representations. For example, MachineLearningForKids allows students to construct a classifier using decision tree learning, and a recent enhancement added the ability to draw the decision tree. While the tree can be very complex for multivariate datasets such as the Titanic survivor data used as one of the illustrative examples, for simpler data the tree is easily interpretable.

For exploring changes in neural net representations, Google's TensorFlow Playground is an ideal tool. It allows students to train small feed-forward neural networks that are graphically displayed in the browser. Every connection is explicitly represented and gets thicker or thinner as the magnitude of the weight increases or decreases; the sign of the weight determines its color. By hovering over a connection it's possible to read the exact weight value. What we want students to appreciate is that the weights constitute an internal representation of a set of feature detectors that the learning algorithm (backpropagation) is incrementally adjusting. Exactly how those adjustments are calculated can be left to more advanced classes. In deep neural networks these feature detectors are complex and hard to interpret, but for the shallow networks and

simple 2D input patterns supported by TensorFlow Playground it is possible to exactly visualize what each feature detector is doing.

Types of Learning

"Finding patterns in data" is a broad concept that encompasses both supervised learning, where the data are labeled, and unsupervised learning, where they are not. It can be used to produce both classifiers and predictors. Classification is a special case of prediction where the output is drawn from a discrete set (the class labels) rather than a continuous range. The other type of machine learning covered in Big Idea 3, "learning from experience", involves something radically different.

In supervised learning the algorithm is provided with the correct answer (the label) for every training example. All it has to do is adjust the internal representations to make the model more likely to produce this answer. In learning from experience, known as reinforcement learning, the algorithm is only provided with a scalar value, the reinforcement signal, that indicates how well things are going. It is not told what it should do differently to make things go better; it has to figure that out for itself.

The other reason reinforcement learning is radically different is that reinforcement learning is used for sequential decision problems. While classification and prediction are one-shot problems where a single input is mapped to a single output, sequential decision problems involve a series of action choices, where each action affects the choices available in the next step. An example would be playing a game like chess, where each move constrains the choices available for the next move. We want students to appreciate two things about reinforcement learning. First,

the computer is not being trained by a teacher; it is generating its own data by making a choice at each step and seeing where that choice leads, i.e., how much reinforcement it ultimately receives. Computers that have become expert game players through reinforcement learning generated their training data by playing against themselves. Second, because we are not required to provide the algorithm with the correct answer at each step, it is possible for the algorithm to discover solutions to problems where we don't know ourselves what would be the best choice to make.

Reinforcement learning is worth teaching in K-12 because it has led to some significant achievements for AI, such as AlphaGo's 2016 defeat of world champion Go player Lee Sedol. But like deep neural networks, the details of reinforcement learning algorithms are too complex for all but the most advanced high school students. Hand-simulating the algorithm would be tedious because of the large number of trials required even for simple tasks. But tiny grid world simulations with only a handful of states and actions can provide a glimpse into how reinforcement learning works.

Some other topics covered in Big Idea #3 are the design of feature sets, development and use of large datasets, and sources and effects of bias in training data.

Big Idea #4: Natural Interaction

Big Idea #4 covers a range of topics relating to how computers interact with people. The one sentence description reads: "Intelligent agents require many kinds of knowledge to interact comfortably with humans." The major topics that make up this big idea are natural language understanding, common sense reasoning, affective computing, and consciousness/theory of mind.

Natural language understanding includes making sense of human requests to intelligent agents, extracting information from text, and translating from one language to another. Language is often syntactically ambiguous, so finding the most likely meaning of a text requires some semantic analysis. For example, "John saw the man from the restaurant" could mean either that John was gazing out from the restaurant when he saw the man, or that John saw the man who had some previous connection to the restaurant. Further context is necessary to decide which meaning the speaker intended.

Speaking with an intelligent agent incapable of **common sense reasoning** would be tedious because everything would have to be spelled out in detail. Common sense reasoning includes naive physics: understanding the properties of solids and liquids and how they behave in response to forces such as gravity. Winograd schema sentences such as "The trophy would not fit in the suitcase because it was too [large/small]" illustrate how an understanding of naive physics, in this case volume and physical containment, determines whether "it" refers to the trophy or the suitcase.

Another requirement for common sense reasoning is knowledge about the world, e.g., knowing that cats are living things, or what chairs are used for. This also includes sociocultural knowledge, such as when to pay at a restaurant, or what makes a good gift for a child.

Today's AI systems show little common sense reasoning ability. Google can translate text into over 100 languages but can't answer questions about a short story that a five year old would

find trivial. AI systems try to make up for this deficiency by focusing on retrieval from huge knowledge bases. But retrieval is not the same as inference. For example, ask Google how much an alligator weighs and it will answer 500 pounds. Ask how much an ostrich weighs and it will say 250 pounds. But ask it "Does an alligator weigh more than an ostrich?" and, as of June 2020, it doesn't even understand the question. Retrieval alone doesn't cut it -- usually. Sometimes it does. Ask Google "Is Microsoft bigger than IBM?" and it finds articles where people have discussed that question. But ask it "Is Intel bigger than Pfizer" and it falls apart, despite the fact that it can retrieve the number of employees and market capitalization of these companies, either of which could be used to compare their size.

To achieve human-like common sense reasoning would require something called Artificial General Intelligence, or AGI. One of the essential understandings we want children to come away with is the difference between the narrow AI reasoners we have today and the broad AGI reasoners depicted in science fiction. One way to drive this idea home is to try to have a conversation with an intelligent agent such as Alexa or Siri. At present they do not maintain context from one utterance to another, so one can't have a meaningful discussion with them. AI developers are currently working on this. Chatbots, which are more specialized than "agents" such as Alexa, address the problem by relying on templates for common interactions such as inquiring about the availability of a product, or placing an order. But if the conversation veers outside the anticipated scenarios, the chatbot is lost. We want children to be aware of these limitations, so that they do not attribute more intelligence to an AI agent than it deserves.

A third topic in Big Idea #4 is **affective computing**, or recognizing and dealing with human

emotional states. This includes sensitivity to tone of voice, facial expressions, and body language, and the ability to adjust interaction style to effectively respond to indications of frustration, boredom, or excitement. If robot companions were as responsive as dogs to our emotional states, they might truly capture our hearts.

The final topic in Big Idea #4 is **consciousness and theory of mind**. These terms are normally addressed in university-level philosophy courses. But because today's children are growing up with intelligent agents, and in a culture filled with fictional robots with human-like personas, they are primed to appreciate questions about whether computers really do have minds, or could in principle have them. Concepts such as the Turing Test, or Searle's hypothetical Chinese room, can be introduced in high school, and should be.

More Playing With Language

The authors of this chapter both enjoyed learning about sentence diagramming in school. In this section we draw a connection between sentence diagramming and AI, specifically the question of how a natural language understander begins to make sense of a sentence. We speculate that playing with AI language tools may instill in future generations of students an appreciation for the formal structure of language that the authors got from experimenting with diagramming.

Sentence diagramming was invented in the 1840s by Stephen W. Clark, a rural New York schoolmaster, as a means of helping students learn to "parse" (grammatically analyze) sentences (Florey, 2012). The notation was refined by Alonzo Reed and Brainerd Kellogg in 1877 into a form that became widely used in 19th and 20th century middle and high schools (ibid). Diagramming is less frequently taught today, and is not included in the Common Core (Thomas,

2014). English teachers argue, and research has shown that isolated direct grammar instruction does not help students become more effective writers (ibid). Nonetheless, readers of a certain age may have fond memories of learning to diagram sentences (Florey, 2006), perhaps because they were a first exposure to formal representations. Diagrams are also something like data structures, a fundamental concept in computer science.
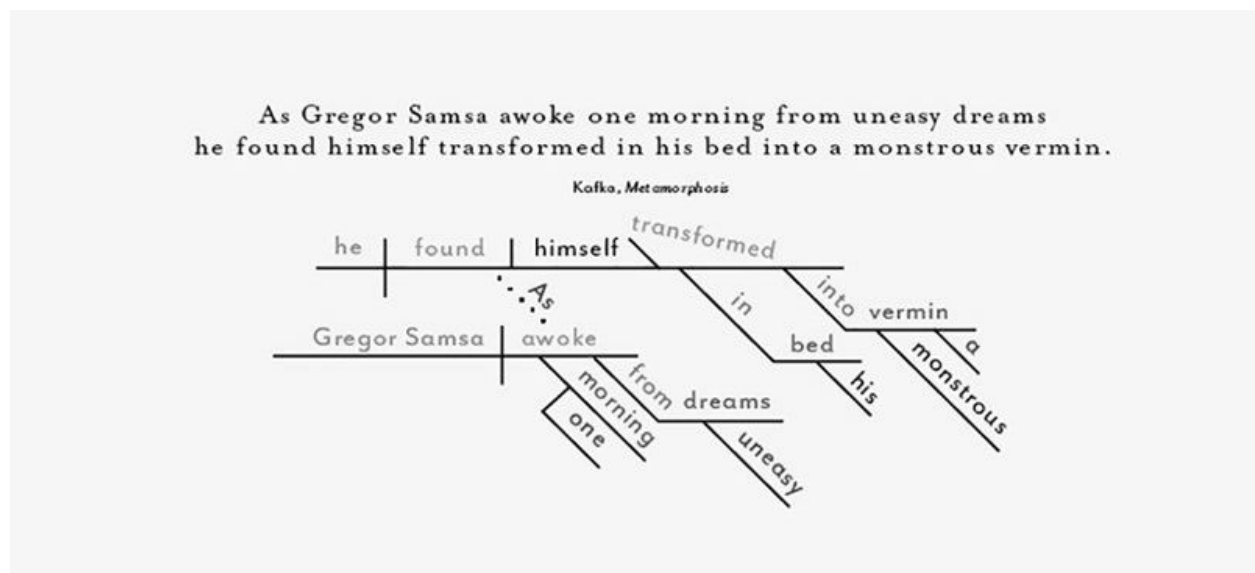

Figure 9.9: Diagramming a sentence. Figure courtesy of Pop Chart Labs.

Sentence diagrams are a simplified version of the syntax trees used in modern linguistics to represent the syntactic structure of sentences. Diagrams use only a few graphical devices, mainly horizontal, vertical, and diagonal lines, as in Figure 9.9. Some of these devices serve multiple roles, whereas syntax trees label every node with an unambiguous grammatical class.

Some linguistic theories are phrased in terms of transformations on tree structures, e.g., a sentence in the active voice can be transformed into the passive voice by switching the subject

and direct object subtrees and modifying the verb phrase, so "John saw Mary" becomes "Mary was seen by John". Ambiguous sentences such as "John saw the man with a telescope", are compatible with multiple tree structures, e.g., one in which "with a telescope" is attached to "saw", and one in which it is attached to "man".

AI systems must parse sentences in order to understand them. While syntactic analysis is only one part of language understanding, it is an essential component. The widespread availability of natural language parsers presents an opportunity to introduce students to grammar in a new way: by having them experiment with automated parsers. Figure 9.10 shows a parse tree provided by the online demo page for the Berkeley Neural Parser[2]:
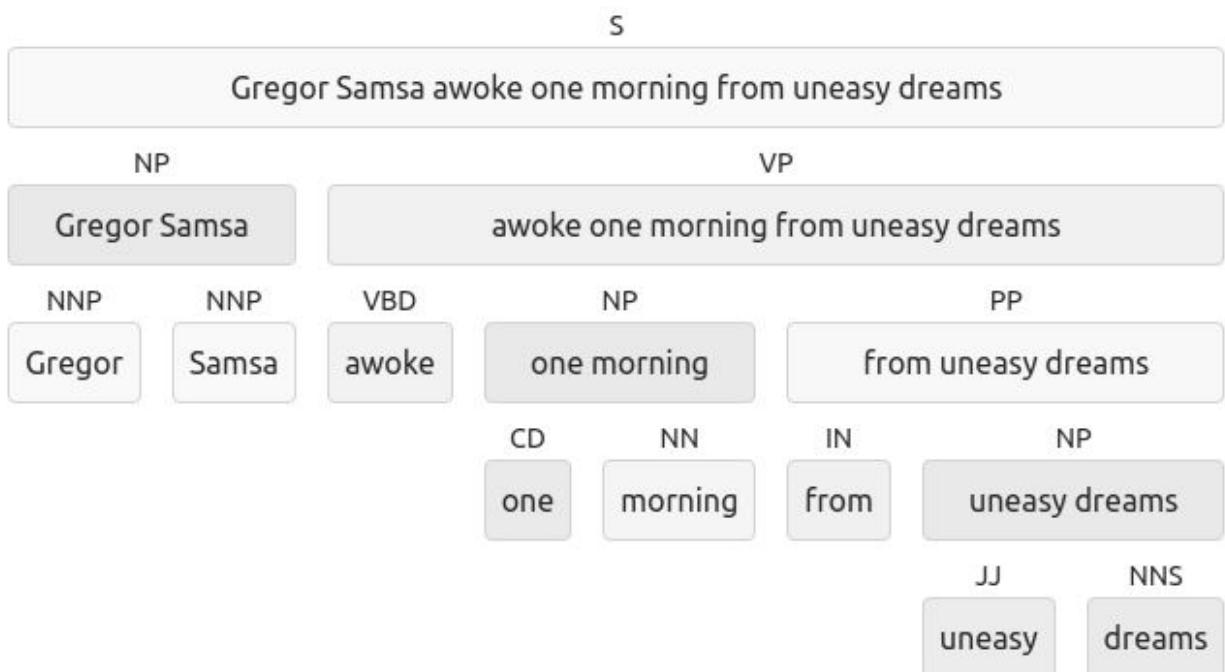
Figure 9.10: Parse tree produced by the Berkeley Neural Parser. NP indicates a noun phrase, VP a verb phrase, and PP a prepositional phrase.

Figure 9.11 shows the same parse in traditional syntax tree notation:
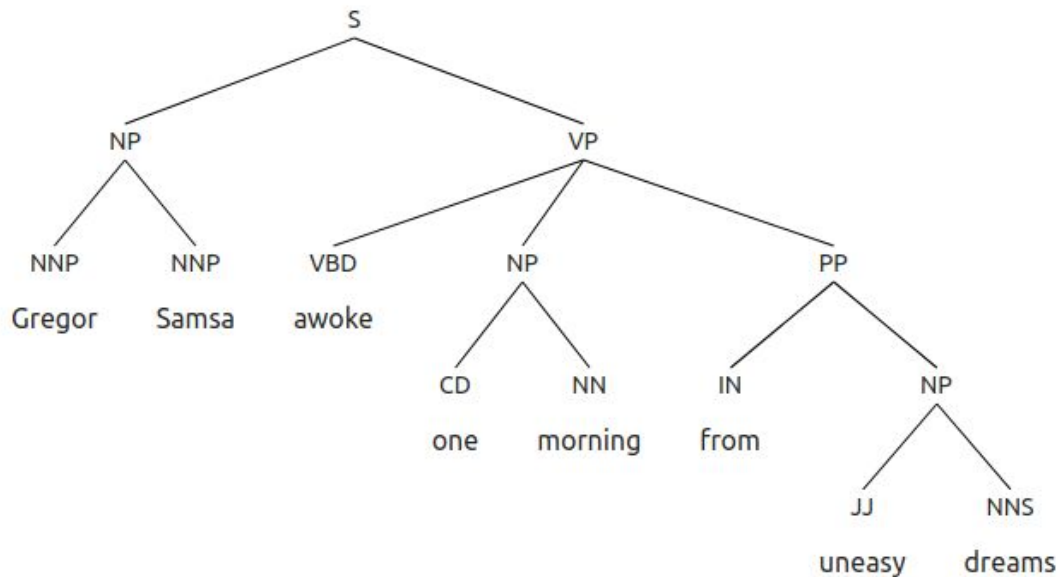
Figure 9.11: Traditional syntax tree notation, also from the Berkeley Neural Parser.

Students could learn to draw syntax trees by comparing their efforts to computer-constructed syntax trees like the ones shown here. They could explore the topic of syntactic ambiguity by constructing sentences and seeing if the parser generates more than one parse tree. And they could learn to write simple rules in a phrase structure grammar and see those rules used to generate parse trees (whose terminal nodes form sentences) by iterative expansion of nonterminal nodes. With a well-designed graphical interface to manage the rules, and automatic generation of many examples from the current rule set, students would have a grammar "sandbox" in which to explore syntax and the relationship between parsing and generation.

The linguist Mark Liberman has remarked that due to diagramming instruction, "grammar-school children of the 19th century learned more about linguistic analysis than most graduate students in English departments do today" (Liberman, 2012). Interactive language tools designed specifically for K-12 could remedy that.

What about semantics? Modern AI has leveraged statistical learning over large datasets to construct practically useful natural language tools for tasks such as machine translation or text summarization. These tools use heuristics to resolve ambiguity from local context. The approach is a powerful one, but does not achieve true "understanding" in the human sense, as evidenced by the limitations these systems still exhibit. For example, given "John saw the man with binoculars", the Berkeley Neural Parser attaches "with binoculars" to "man", while most humans would prefer "saw". But given "John easily saw the man with binoculars", the Berkeley Neural Parser attaches "with binoculars" to "saw". It's not just the presence of "easily" that changes the attachment, because given "John easily saw the man with groceries," the prepositional phrase attaches to "man", as it should. The interaction of the noun with the adverb to influence attachment is not the result of some explicitly formulated rule, nor is it the result of commonsense reasoning. Rather it reflects the statistics of the corpus the model was trained on, captured in a deep neural network. Another example: the parser makes different but plausible attachment choices for "John saw the man with one eye" (attached to "saw") vs. "John saw the man with one leg" (attached to "man"). But it does not do so well on "John saw the man with one ear". Statistics can only get you so far.

One way students can explore machine understanding of language is by comparing how AI

parsers resolve ambiguous sentences with their own preferred interpretations of those sentences, as we've done above. It's even more fun than sentence diagramming.

Big Idea #5: Societal Impact

Big Idea #5 is that "AI can impact society in both positive and negative ways." We want students to be aware of these potential impacts, especially since there is so much apprehension today about AI putting people out of work, enabling unprecedented levels of government surveillance, or unleashing killer robots on the world. All of those things are likely to happen to some extent. But there are also many benefits to be realized from AI, such as improved medical diagnosis and treatment, faster drug discovery, robotic assistance for disabled or elderly persons, increased productivity in industry, and personalized instruction for learners of all ages. Students need to be shown a balanced picture.

We've listed four subtopics for this big idea: (1) ethics of AI making decisions about people, (2) economic impacts of AI technology, (3) AI and culture, and (4) AI for social good.

A great deal of attention has been paid to bias in AI-powered systems. One source of bias results from training a system on an unrepresentative dataset, e.g., a face recognition engine expected to work for everyone but that was trained primarily on Caucasian faces. A trickier problem is automated decision making systems found to treat different groups of people unequally based on criteria we do not consider appropriate. People don't set out to build systems that discriminate based on race or gender. They may even withhold that information from the AI system in an attempt to ensure neutrality. But race and gender correlate with other variables, and so can be implicitly present. The problem is that machine learning systems trained on data that is

unbalanced for historical reasons can acquire biases that perpetuate these imbalances because they want to correctly predict the training data. A famous example is Amazon using information from past technical hiring decisions to train a system for screening resumes only to find that it had learned to assign negative values to keywords that correlate with being female (Dastin, 2018).

An important message for students to hear is that it is possible to take steps to mitigate the negative impacts of technology. Face recognition engines can be required to undergo testing to ensure that they perform equally well for all populations. Automated decision making systems, whether AI-powered or not, can be required by regulation to be transparent in their reasoning, and can be explicitly tested for disparate treatment of protected groups based on inappropriate criteria.

AI and Culture

Until recently, AI's contributions to popular culture have been entirely through science fiction. Speculation about possible futures remains an engaging pastime: will we be happily coexisting with R2D2 and Lieutenant Data, or fleeing the Terminator? Instilling a basic understanding of AI helps students recognize that neither scenario is imminent. But now, actual AI applications are appearing in popular culture. Intelligent assistants modeled after Siri and Alexa are showing up in commercials and TV show episodes. Radio station promos advise listeners to "tell Alexa to play your favorite AM station." Meanwhile, children's interactions with Alexa have sparked a debate about whether they should be taught to treat intelligent agents with politeness (Elgan, 2018). As Elgan observes, "Preparing kids for the future means more than mere manners. It

means teaching them to appreciate the difference between real human people and mere machines designed to create the illusion of humanity."

We will soon have many more humble robots in our lives. Shelf-scanning robots are appearing in department stores and supermarkets, small item delivery robots are trundling the halls of hospitals and high end hotels, and food and package delivery robots are beginning to share the sidewalk with pedestrians. There is already a new genre of YouTube videos showing mishaps with self-driving cars. Today's children will grow up in a culture where we routinely share our living space with machines that, although not very bright, can navigate effectively through the world. iRobot founder Colin Angle reported that 90% of Roomba owners named their vacuums (Barker, 2018). How will they respond to robots that can see and hear?

Conclusions

These are the early days of K-12 AI education. It's a dynamic area where things are changing rapidly. Here is what we see at the frontier:

- New tools and demos are coming online, making it easier to give students hands-on experiences with AI technologies. Since many of these tools run in the browser, they are accessible even to low-resource schools.

- As more states adopt standards mandating computing instruction for all K-12 students, programming is making its way into the lower grades, which means students will be more computationally sophisticated when they learn about AI.

- AI professional development opportunities for teachers will begin to have an impact. Computer science in general is poorly represented in the schools: many computing teachers have no formal computer science training. Even so, they at least understand how a digital computer works, and have elementary programming skills. But few of these teachers claim to know anything about AI, or can even define it. Over the next few years we hope to see AI become more integrated into computing curricula and teachers become more confident about introducing AI topics in their classes.

- AI technologies continue to progress. Intelligent agents are becoming better conversationalists. Robot companions that are not vacuum cleaners will find a niche where they can be successful, while robots in the workplace become common. Fully autonomous, go-anywhere self-driving cars are probably still two decades away, but less demanding applications such as freight hauling or fixed-route shuttle services are already being deployed. As AI becomes a larger part of our lives and culture, the need to demystify AI in K-12 will be widely recognized.

Footnotes

[1]See

https://datascience.stackexchange.com/questions/37078/source-of-arthur-samuels-definition-of-machine-learning

[2]Try the Berkeley Neural Parser at https://parser.kitaev.io/

References

Alammar, Jay. 2019. "The Illustrated Word2vec."

https://jalammar.github.io/illustrated-word2vec/

Barker, Colin. 2018. "Automation: How iRobot's Roomba vacuum cleaner became part of the

family." *ZDNet.com*, June 15, 2018.

https://www.zdnet.com/article/automation-how-irobots-roomba-vacuum-cleaner-became-part-of-

the-family/

Computer Science Teachers Association (CSTA). 2017. "CSTA K12 Computer Science

Standards, Revised 2017." Retrieved from https://www.csteachers.org/standards

Dastin, Jeffrey. 2018. "Amazon scraps secret AI recruiting tool that showed bias against

women." *Reuters*, October 9, 2018.

Elgan, Mike. 2018. "The case against teaching kids to be polite to Alexa." *Fast Company*, June

24, 2018.

https://www.fastcompany.com/40588020/the-case-against-teaching-kids-to-be-polite-to-alexa

Florey, Kitty Burns. 2006. *Sister Bernadette's Barking Dog: The Quirky History and Lost Art of

Diagramming Sentences.* New York: Melville House.

Florey, Kitty Burns. 2012. "A picture of language." *The New York Times*, March 26, 2012.

Kahn, Ken, and Niall Winters. 2020. "A guide to AI extensions to Snap!" https://ecraft2learn.github.io/ai/, chapter 5, "Working with words and language." Accessed June 30, 2020.

Liberman, Mark. 2012. "Diagrammatic excitement." *Language Log*, March 27, 2012. https://languagelog.ldc.upenn.edu/nll/?p=3868

Marr, David. 1982. *Vision*. The MIT Press, Cambridge, MA.

Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. "Distributed representations of words and phrases and their compositionality." *Advances in Neural Information Processing Systems* 26:3111-3119. https://dl.acm.org/doi/10.5555/2999792.2999959

Noy, Natasha, Yuqing Gao, Anshu Jain, Anant Narayanan, Alan Patterson, and Jamie Taylor. 2019. "Industry-scale knowledge graphs: lessons and challenges." *Communications of the ACM* **62**(8):36-43, August 2019. https://doi.org/10.1145/3331166

Oxford University Press. 2020. *The Oxford English Dictionary* website. https://www.oed.com

ReadyAI. 2019. *AI+ME*. Online course accessible at https://edu.readyai.org/courses/aime/

Thomas, Paul L. 2014. "Diagramming sentences and the art of misguided nostalgia."

https://radicalscholarship.wordpress.com/2014/08/24/diagramming-sentences-and-the-art-of-mis

guided-nostalgia/

Touretzky, David, Christina Gardner-McCune, Fred Martin, and Deborah Seehorn. 2019.

"Envisioning AI for K-12: What should every child know about AI?" *Proceedings of the AAAI*

Conference on Artificial Intelligence, volume 33. https://doi.org/10.1609/aaai.v33i01.33019795