Chapter 7. Categorization: Describing Resource Classes and Types

Robert J. Glushko
Rachelle Annechino
Jess Hemerly
Robyn Perry
Longhao Wang

7.1. Introduction

Many texts in library science introduce categorization via cataloging rules, a set of highly prescriptive principles for assigning resources to categories. Many texts in computer science discuss the process of defining the categories needed to create, process, and store information in terms of programming language constructs: "here's how to define an abstract type, and here's the data type system." Machine learning and data science texts explain how categories are created through statistical analysis of the correlations among the values of features in a collection or dataset. We take a very different approach in this chapter, but all of these different perspectives have a place in it.

Navigating This Chapter

In the following sections, we discuss how and why we create categories, reviewing some important work in philosophy, linguistics, and cognitive psychology to better understand how categories are created and used in organizing systems. We discuss how the way we organize differs when we act as individuals or as members of social, cultural, or institutional groups (§7.2); later we share principles for creating categories(§7.3), design choices (§7.4), and implementation experience (§7.5). Throughout the chapter, we will compare how categories created by people compare with those created by computer algorithms. As usual, we close the chapter with a summary of the key points (§7.6).

7.2. The What and Why of Categories

Categories are equivalence classes, groups of things or abstract entities that we treat the same. Equivalence does not mean that every instance of a category is identical. It only means that from some perspective, or for some purpose, we treat the members of a category as equivalent. When we consider something as a member of a category, we are making choices about which of its properties we are focusing on and which ones we are ignoring. We do this automatically and unconsciously most of the time, but we can also do it in an explicit and self-aware way. When we create categories with conscious effort, we often say that we are creating a model. You should be familiar with the idea that a model is a set of simplified descriptions or a physical representation that removes some complexity to emphasize some features or characteristics and to de-emphasize others.

When we encounter objects or situations, recognizing them as members of a category helps us to interact with them. For example, when we enter an unfamiliar building, we might need to open or pass through an entryway that we recognize as a door. We might never have seen that particular door before, but it has properties and affordances that we know that all doors have; it has a doorknob or a handle; it allows access to a larger space; it opens and closes. By mentally assigning this particular door to the "doors" category we distinguish it from the "windows" category. The windows category also contains objects that sometimes have handles and that open and close, but which we do not normally pass through to enter another space. Categorization judgments are therefore not just about what is included in a class, but also about what is excluded from a class. Nevertheless, the category boundaries are not sharp; a "Dutch door" is divided horizontally in half so that the bottom can be closed like a door while the top can stay open like a window.

Creating and using categories are essential human activities. .Categories are cognitive and linguistic models for applying prior knowledge whenever we perceive, communicate, analyze, predict, or classify. Without categories, we would perceive the world as an unorganized blur of things with no obvious or memorable relation to each other. Every wall-entry we encounter would be new to us, and we would have to discover its properties and supported interactions as though we had never before encountered a door.

Even before they can talk, children behave in ways that suggest they have formed categories based on shape, color, and other properties they can directly perceive in physical objects. People almost effortlessly learn tens of thousands of categories embodied in the culture and language in which they grow up. People also rely on their own experiences, preferences, and goals to adapt these cultural categories or to create entirely individual ones that they use to organize their personal resources. Later on, through training and formal education, people learn to apply careful and logical thinking processes so that they can create and understand institutional categories in engineering, logistics, transport, science, law, business, and other systematized contexts.

These three contexts of cultural, individual, and institutional categorization share some core ideas, but they emphasize different processes and purposes for creating categories. Cultural categorization is a natural human cognitive ability that serves as a foundation for both informal and formal organizing systems. Individual categorization tends to grow spontaneously out of our personal activities. Institutional categorization responds to the need for formal coordination and cooperation within and between companies, governments, and other goal-oriented enterprises.

In contrast to these three categorization contexts in which people create categories, computational categories are created by computer programs for information retrieval, machine learning, predictive analytics, and other applications. Computational categories are similar to those created by people in some ways but differ substantially in other ways.

7.2.1. Cultural Categories

Cultural categories are the archetypical form of categories on which individual and institutional categories are usually based. Cultural categories tend to describe our everyday experiences of the world and our accumulated cultural knowledge. Such categories describe objects, events,

settings, internal experiences, physical orientation, relationships between entities, and many other aspects of human experience. Cultural categories are learned primarily, with little explicit instruction, through normal exposure of children with their caregivers as they learn a language.

Languages differ a great deal in the words they contain and also in more fundamental ways that they require speakers or writers to attend to details about the world or aspects of experience that another language allows them to ignore. This idea is described as linguistic relativity.

Linguistic Relativity

The diversity of languages led Benjamin Whorf, in the mid-20th century, to propose an overly strong statement of the relationships among language, culture, and thought. Whorf argued that the particularities of one's native language determine how we think and what we can think about. Among his controversial ideas was the suggestion that, because some Native American languages lacked words or grammatical forms that refer to what we call "time" in English, they could not understand the concept. More careful language study showed both parts of the claim to be completely false.

Nevertheless, even though academic linguists have discredited strong versions of Whorf's ideas, more moderate versions of linguistic relativity have become influential and help us understand cultural categorization. Roman Jakobson said it this way: "Languages differ essentially in what they *must* convey and not in what they *may* convey." In English one can say "I spent yesterday with a neighbor." In languages with grammatical gender, one must choose a word that identifies the neighbor as male or female.

For example, speakers of the Australian Aboriginal language Guugu Yimithirr do not use concepts of left and right but instead, speakers use cardinal directions. In English, we might say to a person facing north, "Take a step to your left." In contrast, a speaker of Guugu Yimithirr would use their term for west. If the person faced south, we would change our instruction to "right," but they would still use their term for west. Imagine how difficult it would be for a speaker of Guugu Yimithirr and a speaker of English to collaborate in organizing a storage room or a closet.

It is not controversial to notice that different cultures and language communities have different experiences and activities that give them contrasting knowledge about particular domains. No one would doubt that university undergraduates in Chicago would think differently about animals than inhabitants of Guatemalan rainforests, or even that different types of "tree experts" (taxonomists, landscape workers, foresters, and tree maintenance personnel) would describe and categorize trees differently.

7.2.2. Individual Categories

Individual categories are created in an organizing system to satisfy the ad hoc requirements that arise from a person's unique experiences, preferences, and resource collections. Unlike cultural categories, which usually develop slowly and last a long time, individual categories are created in response to a specific situation, or to solve an emerging organizational challenge. As a

consequence, the categories in individual organizing systems generally have short lifetimes and rarely outlive the person who created them.

Individual categories draw from cultural categories but differ in two critical ways. First, individual categories sometimes have an imaginative or metaphorical basis that is meaningful to the person who created them but which might distort or misinterpret cultural categories. Second, individual categories are often specialized or synthesized versions of cultural categories that capture particular experiences or personal history. For example, a person who has lived in China and Mexico might have highly individualized categories for foods they like and dislike that incorporate characteristics of both Chinese and Mexican cuisine.

Individual categories in organizing systems also reflect the idiosyncratic set of household goods, music, books, website bookmarks, or other resources that a person might have collected over time. The organizing systems for financial records, personal papers, or email messages often use highly specialized categories that are shaped by specific tasks, relationships with other people, events of personal history, and other highly individualized considerations. Put another way, individual categories are used to organize resource collections that are likely not representative samples of all resources of the type being collected. If everyone had the same collection of music, books, clothes, or toys, the world would be a boring place.

For most of human history, individual categorization systems were usually not visible to or shared with others. But now individual categories can be easily seen when people use web-based organizing system for pictures, music, or other personal resources. With Instagram, YouTube, Pinterest, or similar applications for organizing photos and videos, people typically use existing cultural categories to tag their content as well as new individual categories that they invent and name with a hashtag.

7.2.3. Institutional Categories

In contrast to cultural categories that are created and used implicitly, and to individual categories that are used by people acting alone, institutional categories are created and used explicitly, and most often by many people working together. Institutional categories are most often created in abstract and information-intensive domains where unambiguous and precise categories are needed to regulate and systematize activity, to enable information sharing and reuse, and to reduce transaction costs. Furthermore, instead of describing the world as it is, institutional categories are usually defined to change or control the world by imposing semantic models that are more formal and arbitrary than those in cultural categories. Laws, regulations, and standards often specify institutional categories, along with decision rules for assigning resources to new categories, and behavior rules that prescribe how people must interact with them. The rigorous definition of institutional categories enables classification*:* the systematic assignment of resources to categories in an organizing system.

Creating institutional categories by more systematic processes than cultural or individual categories does not ensure that people will use them in systematic and rational ways. The reasoning and rationale behind institutional categories might be unknown to, or ignored by, the people who use them. Likewise, this way of creating categories does not prevent them from

being biased. Indeed, the goal of institutional categories is often to impose or incentivize biases in interpretation or behavior. There is no better example of this than the practice of gerrymandering, designing the boundaries of election districts to give one political party or ethnic group an advantage.

Institutional categories overcome the vagueness and inconsistency of cultural categories because the former typically conform to stricter logical standards to support inference and meet legal requirements. As requirements change over time, institutional categories must often change as well, implying version control, compliance testing, and other formal maintenance and governance processes.

Some institutional categories that initially had narrow or focused applicability have found their way into more popular use and are now considered cultural categories. A good example is the periodic table in chemistry, which Mendeleev developed in 1869 as a new system of categories for the chemical elements. The periodic table proved essential to scientists in understanding their properties and in predicting undiscovered ones. Today the periodic table is taught in elementary schools, and many things other than elements are commonly arranged using a graphical structure that resembles the periodic table of elements in chemistry, including sci-fi films and movies, desserts, and superheroes.

7.2.4. A “Categorization Continuum”

As we have seen, the concepts of cultural, individual, and institutional categorization usefully distinguish the primary processes and purposes when people create categories. However, these three kinds of categories can fuse, clash, and recombine with each other. Rather than viewing them as having precise boundaries, we might view them as regions on a continuum of categorization activities and methods.

Consider a few different perspectives on categorizing animals. Scientific institutions categorize animals according to explicit, principled classification systems, such as the Linnaean taxonomy that assigns animals to a phylum, class, order, family, genus, and species. Cultural categories are more fluid, sometimes converging with principled taxonomies, and at other times diverging from them. Human beings are classified within the animal kingdom in biological classification systems, but people are usually not considered animals in most cultural contexts. Animals are also often culturally categorized as pets or non-pets. The category "pets" commonly includes dogs, cats, and fish. A pet cat might be categorized at multiple levels that incorporate individual, cultural, and institutional perspectives on categorization—as an "animal" (cultural/institutional), as a "mammal" (institutional), as a "domestic short-hair" (institutional) as a "cat" (cultural), and as a "troublemaker" or a "favorite" (individual), among other possibilities, in addition to being identified individually by one or more pet names.

Categorization skewed toward cultural perspectives incorporate traditional categories, such as those learned implicitly from social interactions. Categorization skewed toward institutional perspectives emphasizes explicit, formal categories, like the categories employed in biological classification systems.

7.2.5. Computational Categories

Computational categories are created by computer programs when the number of resources, or when the number of descriptions or observations associated with each resource, are so large that people cannot think about them effectively. Computational categories are created for information retrieval, predictive analytics, and other applications where information scale or speed requirements are critical. The resulting categories are similar to those created by people in some ways but differ substantially in other ways.

The simplest kind of computational categories can be created using descriptive statistics. Descriptive statistics do not identify the categories they create by giving them familiar cultural or institutional labels. Instead, they create implicit categories of items according to how much they differ from the most typical or frequent ones. For example, in any dataset where the values follow the normal distribution, statistics of central tendency and dispersion serve as standard reference measures for any observation. These statistics identify categories of items that are very different or statistically unlikely outliers, which could be signals of measurement errors, poorly calibrated equipment, employees who are inadequately trained or committing fraud, or other problems.

Many text processing methods and applications use simple statistics to categorize words by their frequency in a language, in a collection of documents, or in individual documents, and these categories are exploited in many information retrieval applications.

The subfield of computer science known as machine learning contains many techniques that are relevant to organizing systems. In “supervised” learning, a machine learning program is trained with sample items or documents that are labeled by category, and the program learns to assign new items to the correct categories. Over time the program, which is called a classifier, improves its performance by adjusting the weights for features that distinguish the categories. In contrast, “unsupervised” learning techniques in machine learning analyze a collection of resources to discover statistical regularities or correlations among the items, creating a set of categories without any labeled training data. Unsupervised learning is also called statistical pattern recognition.

Many computational categories resemble individual categories because they are tied to specific collections of resources or data and are designed to satisfy narrow goals. The individual categories you use to organize your email inbox or the files on your computer reflect your specific interests, activities, and personal network and are certainly different than those of anyone else. Similarly, your credit card company analyzes your specific transactions to create computational categories of "likely good" and "likely fraudulent" that are different for every cardholder.

This focused scope is apparent when we consider how we might describe a computational category. "Fraudulent transaction for cardholder 4264123456780123" is not lexicalized with a one-word label as familiar cultural categories are. "Door" and "window" have broad scopes that are not tied to a single purpose. Put another way, the "door" and "window" cultural categories are highly reusable, as are institutional categories like those used to collect economic or health data

that can be analyzed for many different purposes. The definitions of "door" and "window" might be a little fuzzy, but institutional categories are more precisely defined, often by law or regulation. Examples are the *North American Industry Classification System (NAICS)* from the US Census Bureau and the *United Nations Standard Products and Services Code (UNSPC)*.

A final contrast between categories created by people and those created computationally is that the former can almost always be inspected and reasoned about by other people, but only some of the latter can. A computational model that categorizes loan applicants as excellent or poor credit risks probably uses properties like age, income, home address, and marital status, so that a banker can understand and explain a credit decision. However, many other computational categories, especially those that created by clustering and deep learning techniques, are uninterpretable by people.

7.3. Principles for Creating Categories

We now take a systematic look at principles for creating categories, including enumeration, single properties, multiple properties and hierarchy, probabilistic, similarity, and theory- and goal-based categorization. These ways of creating categories differ in the information and mechanisms they use to determine category membership.

7.3.1. Enumeration

The simplest principle for creating a category is enumeration; any resource in a finite or countable set can be deemed a category member by that fact alone. This principle is also known as extensional definition, and the members of the set are called the extension. Many institutional categories are defined by enumeration as a set of possible or legal values, like the 50 United States or the ISO currency codes (ISO 4217).

Enumerative categories enable membership to be unambiguously determined because a value like a currency code is either a member of the category or it is not. However, this clarity has a downside; it makes it hard to argue that something not explicitly mentioned in an enumeration should be considered a member of the category, which can make laws or regulations inflexible. Moreover, there comes a size when enumerative definition is impractical or inefficient, and the category either must be sub-divided or be given a definition based on principles other than enumeration.

For example, for millennia we earthlings have had a cultural category of “planet” as a “wandering” celestial object, and because we only knew of planets in our own solar system, the planet category was defined by enumeration: Mercury, Venus, Earth, Mars, Jupiter, and Saturn. When the outer planets of Uranus, Neptune, and Pluto were identified as planets in the 18^{th}-20^{th} centuries, they were added to this list of planets without any changes in the cultural category. But in the last couple of decades many previously unknown planets outside our solar system have been detected, making the set of planets unbounded, and it is impossible to define the planet category by enumeration.

The International Astronomical Union (IAU) thought it solved this category crisis by proposing a definition of planet as “a celestial body that is (a) in orbit around a star, (b) has sufficient mass for its self-gravity to overcome rigid body forces so that it assumes a hydrostatic equilibrium (nearly round) shape, and (c) has cleared the neighborhood around its orbit.” Unfortunately, Pluto does not satisfy the third requirement, so it no longer is a member of the planet category, and instead is now called an “inferior planet.”

Changing the definition of a significant cultural category generated a great deal of controversy and anxiety among ordinary non-scientific people. A typical headline was “Pluto’s demotion has schools spinning,” describing the outcry from elementary school students and teachers about the injustice done to Pluto and the disruption on the curriculum.

7.3.2. Single Properties

It is intuitive and useful to think in terms of properties when we identify instances and when we are describing instances. Therefore, it should also be intuitive and useful to consider properties when we analyze more than one instance to compare and contrast them so we can determine which sets of instances can be treated as a category or equivalence class. Categories whose members are determined by one or more properties or rules follow the principle of intensional definition, and the defining properties are called the intension.

You might be thinking here that enumeration or extensional definition of a category is also a property test; is not “being a state” a property of California? But statehood is not a property precisely because “state” is defined by extension, which means the only way to test California for statehood is to see if it is in the list of states.

Any single property of a resource can be used to create categories, and the easiest ones to use are often the intrinsic static properties. As we discussed in Chapter 5, intrinsic static properties are those inherent in a resource that never change. The material of composition of natural or manufactured objects is an intrinsic and static property that can be used to arrange physical resources. For example, an organizing system for a personal collection of music that is based on the intrinsic static property of physical format might use categories for CDs, DVDs, vinyl albums, 8-track cartridges, reel-to-reel tape and tape cassettes.

Using a single property is natural and easy when the properties can take on only a small set of discrete values like music formats, and especially when the property is closely related to how the resources are used, as they are with the music collection where each format requires different equipment to listen to the music. Each value then becomes a subcategory of the music category.

The author, date, and location of creation of an intellectual resource cannot be directly perceived, but they are also intrinsic static properties. The subject matter or purpose of a resource, its “what it is about” or “what it was originally for,” are also intrinsic static properties that are not directly perceivable, especially for information resources.

The name or identifier of a resource is often arbitrary but once assigned usually does not change, making it an extrinsic static property. Any collection of resources with alphabetic or numeric

identifiers as an associated property can use sorting order as an organizing principle in a completely reliable way.

Some resource properties are both extrinsic and dynamic because they are based on usage or behaviors that can be highly context-dependent. The current owner or location of a resource, its frequency of access, the joint frequency of access with other resources, or its current rating or preference with respect to alternative resources are typical extrinsic and dynamic properties that can be the basis for arranging resources and defining categories.

Some properties can have a large number of values or are continuous measures, but if there are explicit rules for using property values to determine category assignment the resulting categories are still easy to understand and use. For example, we naturally categorize people we know by their current profession, the city where they live, their hobbies, or their age. Properties with a numerical dimension like “frequency of use” are often transformed into a small set of categories like “frequently used,” “occasionally used,” and “rarely used” based on the numerical property values.

While there is an infinite number of logically expressible properties for any resource, most of them would not lead to categories that would be interpretable and useful for people. If people are going to use the categories, it is essential to base them on properties that are psychologically or pragmatically relevant. Whether something weighs more or less than 5000 pounds is not a useful property to apply to things in general, because it puts cats and chairs in one category, and buses and elephants in another.

To summarize: The most useful single properties to use for creating categories for an organizing system used by people are those that are formally assigned, objectively measurable and orderable, or tied to well-established cultural categories because the resulting categories will be easier to understand and describe.

If only a single property is used to distinguish among some set of resources and to create the categories in an organizing system, the choice of property is critical because different properties often lead to different categories. If we categorize using the age property, Bill Gates and Mark Zuckerberg are unlikely to end up in the same category of people. Using the wealth property, they most certainly would. Furthermore, if only one property is used to create a system of categories, any category with a large number of items in it lacks coherence because differences on other properties are too apparent, and some category members do not fit as well as the others.

7.3.3. Multiple Properties

Organizing systems often use multiple properties to define categories. There are three different ways in which to do this that differ in the scope of the properties and how essential they are in defining the categories.

7.3.3.1. Multi-Level or Hierarchical Categories

If you have many shirts in your closet (and you are a bit compulsive), instead of just separating your shirts from your pants using a single property (the part of body on which the clothes are worn) you might arrange the shirts by style, and then by sleeve length, and finally by color. When all of the resources in an organizing system are arranged using the same sequence of resource properties, this creates a logical hierarchy, a multi-level category system.

If we treat all the shirts as the collection being organized, the broad category of shirts is first divided by style into categories like “dress shirts,” “work shirts,” “party shirts,” and “athletic or sweatshirts.” Each of these style categories is further divided until the categories are very narrow ones, like the “white long-sleeve dress shirts” category. A particular shirt ends up in this last category only after passing a series of property tests along the way: it is a dress shirt, it has long sleeves, and it is white. Each test creates more precise categories in the intersections of the categories whose members passed the previous property tests.

Put another way, each subdivision of a category takes place when we identify or choose a property that differentiates the members of the category in a way that is important or useful for some intent or purpose. Shirts differ from pants in the value of the “part of body” property, and all the shirt subcategories share this “top part” value of that property. However, shirts differ on other properties that determine the subcategory to which they belong. Even as we pay attention to these differentiating properties, it is important to remember the other properties, the ones that members of a category at any level in the hierarchy have in common with the members of the categories that contain it. These properties are often described as “inherited” or “inferred” from the broader category. For example, just as every shirt shares the "worn on the top part of body" property, every item of clothing shares the “can be worn on the body” property, and every resource in the “shirts” and “pants” category inherits that property.

Each differentiating property creates another level in the category hierarchy, which raises an obvious question: How many properties and levels do we need? To answer this question, we must reflect upon the shirt categories in our closet. Our organizing system for shirts arranges them with the three properties of style, sleeve length, and color; some of the categories at the lowest level of the resulting hierarchy might have only one member or no members at all. You might have yellow or red short-sleeved party shirts, but probably do not have yellow or red long-sleeved dress shirts, making them empty categories. Any category with only one member does not need any additional properties to tell the members apart, so a category hierarchy is logically complete if every resource is in a category by itself.

However, even when the lowest level categories of our shirt organizing system have more than one member, we might choose not to use additional properties to subdivide it because the differences that remain among the members do not matter to us for the interactions the organizing system needs to support. Suppose we have two long-sleeve white dress shirts from different shirt makers, but whenever we need to wear one of them, we ignore this property. Instead, we pick one or the other, treating the shirts as completely equivalent or substitutable. When the remaining differences between members of a category do not make a difference to the users of the category, we can say that the organizing system is pragmatically or practically

complete even if it is not yet logically complete. We might say it is complete “for all intents and purposes.” Indeed, we might argue that it is desirable to stop subdividing a system of categories while small differences remain among the items in each category because this leaves some flexibility or logical space in which to organize new items. On the other hand, consider the shirt section of a big department store. Shirts there might be organized by style, sleeve length, and color as they are in our home closet, but would most likely be further organized by shirt maker and by size to enable a shopper to find a Marc Jacobs long-sleeve blue dress shirt of size 15/35. The department store organizing system needs more properties and a deeper hierarchy for the shirt domain because it has many more shirt instances to organize and because it needs to support many shirt shoppers, not just one person whose shirts are all the same size.

7.3.3.2. Different Properties for Subsets of Resources

Another way to create categories is to employ different properties for distinct subsets of the resources being organized. This contrasts with the strict multi-level approach in which every resource is evaluated with respect to every property. Alternatively, we could view this principle as a way of organizing multiple domains that are conceptually or physically adjacent, each of which has a separate set of categories based on properties of the resources in that domain. This principle is used for most folder structures in computer file systems and by many email applications; you can create as many folder categories as you want, but any resource can only be placed in one folder.

The contrasts between intrinsic and extrinsic properties, and between static and dynamic ones, are helpful in explaining this method of creating organizing categories. For example, you might organize all of your clothes using intrinsic static properties if you keep your shirts, socks, and sweaters in different drawers and arrange them by color; extrinsic static properties if you share your front hall closet with a roommate, so you each use only one side of that closet space; intrinsic dynamic properties if you arrange your clothes for ready access according to the season; and, extrinsic dynamic properties if you keep your most frequently used jacket and hat on a hook by the front door.

A typical supermarket embodies a surprisingly complex classification system. Each section of the store employs a different set of properties to arrange its resources, and some properties such as perishability and onsite preparation are important in more than one section.

If we relax the requirement that different subsets of resources use different organizing properties and allow any property to be used to describe any resource, the loose organizing principle we now have is often called tagging. Using any property of a resource to create a description is an uncontrolled and often unprincipled principle for creating categories, but it is increasingly popular for organizing photos, websites, email messages in Gmail, or other web-based resources.

7.3.3.3. Necessary and Sufficient Properties

A large set of resources does not always require many properties and categories to organize it. Some types of categories can be defined precisely with just a few essential properties. For example, a prime number is a positive integer that has no divisors other than 1 and itself, and this category definition perfectly distinguishes prime and not-prime numbers no matter how many numbers are being categorized. “Positive integer” and “divisible only by 1 and itself” are necessary or defining properties for the prime number category; every prime number must satisfy these properties. These properties are also sufficient to establish membership in the prime number category; any number that satisfies the necessary properties is a prime number. Categories defined by necessary and sufficient properties are also called monothetic. They are also sometimes called classical categories because they conform to Aristotle’s theory of how categories are used in logical deduction using syllogisms.

The Classical View of Categories

The classical view is that categories are defined by necessary and sufficient properties. This theory has been enormously influential in Western thought and is embodied in many organizing systems, especially those for information resources. However, we cannot rely on this principle to create categories in many domains and contexts because there are not necessary and sufficient properties. As a result, many psychologists, cognitive scientists, and computer scientists who think about categorization have criticized the classical theory.

We think this is unfair to Aristotle, who proposed what we now call the classical theory primarily to explain how categories underlie the logic of deductive reasoning: All men are mortal; Socrates is a man; Therefore, Socrates is mortal. People are wrong to turn Aristotle's thinking around and apply it to the problem of inductive reasoning, how categories are created from examples. But this is not Aristotle’s fault; he was not trying to explain how cultural categories arise.

Theories of categorization have evolved a great deal since Plato and Aristotle proposed them over two thousand years ago, but in many ways, we still adhere to the classical view of categories when we create organizing systems because they can be easier to implement and maintain that way.

"Necessary and sufficient" category definition implies is that every member of the category is an equally good member or example of the category; every prime number is equally prime. Institutional category systems often employ necessary and sufficient properties for their conceptual simplicity and straightforward implementation in decision trees, database schemas, and programming language classes.

Consider the definition of an address as requiring a street, city, administrative region, and postal code. Anything that has all of these information components is a valid address, and anything that lacks any of them is not a valid address.

7.3.4. The Limits of Property-Based Categorization

Property-based categorization works well by definition for categories like “prime number” where the category is defined by necessary and sufficient properties. Property-based categorization also works well when properties are conceptually distinct and when the value of a property is easy to perceive and examine, as they are with human-made physical resources like shirts.

However, for organizing systems that need to categorize information resources, categories defined using easily perceived properties are often not effective. There might be indications “on the surface” that suggest boundaries between types of information resources, but these are often just presentation or packaging choices. That is to say, neither the size of a book nor the color of its cover are reliable cues for what it contains. Information resources have numerous descriptive properties like their title, author, and publisher that can be used more effectively to define categories, and these are certainly useful for some kinds of interactions, like finding all of the books written by a particular author or published by the same publisher. However, for practical purposes, the most useful property of an information resource is its aboutness, which may not be objectively perceivable and which is often hard to describe. Any collection of information resources in a library or document filing system is likely to be about many subjects and topics, and when an individual resource is categorized according to a limited number of its content properties, it is at the same time not being categorized using the others.

When the web first started, there were many attempts to create categories of websites, most notably by Yahoo! As the web grew, it became obvious that search engines would be vastly more useful because their near real-time text indexes obviate the need for a priori assignment of web pages to categories. Rather, web search engines represent each web page or document in a way that treats each word or term they contain as a separate property.

Considering every distinct word in a document stretches our notion of property to make it very different from the kinds of properties we have discussed so far, where properties are explicitly used by people to make decisions about category membership and resource organization. Human perceptual and cognitive limitations make it impossible for people to pay attention to more than a few properties at the same time. But computers have no such limitations, and algorithms for information retrieval and machine learning can use huge numbers of properties.

7.3.5. Probabilistic Categories and “Family Resemblance”

As we have seen, some categories can be precisely defined using necessary and sufficient features, especially when the properties that determine category membership are easy to observe and evaluate. A number is either a prime number or it isn't. A person cannot be a registered student and not registered at the same time.

However, categorization based on explicit and logical consideration of properties is much less effective and sometimes not even possible for domains where properties lack one or more of the characteristics of separability, perceptibility, and necessity. Instead, we need to categorize using properties in a probabilistic or statistical way to yield some measure of similarity between the resource to be categorized and the other members of the category.

Consider a familiar category like “bird.” All birds have feathers, wings, beaks, and two legs. But there are thousands of types of birds, and they are distinguished by properties that some birds have that other birds lack: most birds can fly, most are active in the daytime, some swim, some swim underwater; some have webbed feet. These properties are correlated or clustered, a consequence of natural selection that conveys advantages to particular configurations of characteristics. There are many different clusters of properties; birds that live in trees have different wings and feet than those that swim, and birds that live in deserts have different colorations and metabolisms that those that live near water. So instead of being defined by a single set of properties that are both necessary and sufficient, the bird category is defined probabilistically. Decisions about category membership are made by accumulating evidence from the properties that are more or less characteristic of the category.

Family Resemblance and Typicality

These six animals have some physical features in common but not all of them, yet they resemble each other enough to be easily recognizable as birds. Most people consider a pigeon to be a more typical bird than a penguin.

Categories of information resources often have the same probabilistic character. Some words (beneficiary, pharmaceutical, offer) occur often in spam messages, but these words also occur in messages that are not spam. A spam classifier uses the probabilities of each word in a message in spam and non-spam contexts to calculate an overall likelihood that the message belongs in the spam category.

There are three related consequences for categories when their characteristic properties have a probabilistic distribution:

- The first is an effect of typicality or centrality that makes some members of the category better examples than others. Membership in probabilistic categories is not all or none, so even if they share many properties, an instance that has more of the characteristic properties will be judged as better or more typical. Try to define "bird" and then ask yourself if all of the things you classify as birds are equally good examples of the category (look at the six birds above in Family Resemblance and Typicality). This effect is also described as gradience in category membership and reflects the extent to which the most characteristic properties are shared.
- A second consequence is that the sharing of some but not all properties creates what we call family resemblances among the category members; just as biological family members do not necessarily all share a single set of physical features but still are recognizable as members of the same family. This idea was first proposed by the 20th-century philosopher Ludwig Wittgenstein, who used "games" as an example of a category whose members resemble each other according to shifting property subsets.
- The third consequence, when categories do not have necessary features for membership, is that the boundaries of the category are not fixed; the category can be stretched and new members assigned as long as they resemble incumbent members. Personal video games and multiplayer online games like World of Warcraft did not exist in Wittgenstein's time but we have no trouble recognizing them as games and neither would Wittgenstein, were he alive. Categories defined by family resemblance or multiple and shifting property sets are called polythetic.

We conclude that instead of using properties one at a time to assign category membership, we can use them in a composite or integrated way where together a co-occurring cluster of properties provides evidence that contributes to a similarity calculation. Something is categorized as an A and not a B if it is more similar to A's best or most typical member rather than it is to B's.

7.3.6. Similarity

Similarity is a measure of the resemblance between two things that share some characteristics but are not identical. It is a very flexible notion whose meaning depends on the domain within which we apply it. Some people consider that the concept of similarity is itself meaningless because there must always be some basis, some unstated set of properties, for determining whether two things are similar. If we identify those properties and how they are used, there is no work for a similarity mechanism to do.

To make similarity a useful mechanism for categorization we have to specify how the similarity measure is determined. Four psychologically-motivated approaches propose different functions for computing similarity: feature- or property-based, geometry-based, transformational, and alignment- or analogy-based.

7.3.6.1. Feature-based Models of Similarity

An influential model of feature-based similarity calculation is Amos Tversky's contrast model, which matches the features or properties of two things and computes a similarity measure according to three sets of features:

- those features they share,
- those features that the first has that the second lacks, and
- those features that the second has that the first lacks.

The similarity based on the shared features is reduced by the two sets of distinctive ones. The weights assigned to each set can be adjusted to explain judgments of category membership.

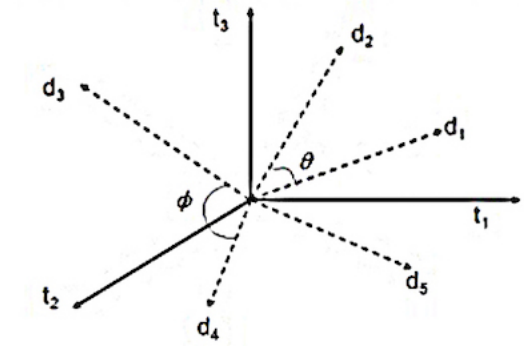
Another commonly feature-based similarity measure is the Jaccard coefficient, the ratio of the common features to the total number of them. This simple calculation equals zero if there are no overlapping features and one if all features overlap. Jaccard's measure is often used to calculate document similarity by treating each word as a feature.

We often use a heuristic version of feature-based similarity calculation when we create multi-level or hierarchical category systems to ensure that the categories at each level are at the same level of abstraction. For example, if we were organizing a collection of musical instruments, it would not seem correct to have subcategories of "woodwind instruments," "violins," and "cellos" because the feature-based similarity among the categories is not the same for all pairwise comparisons among the categories; violins and cellos are simply too similar to each other to be separate categories given the much broader category of woodwinds.

7.3.6.2. Geometric Models of Similarity

Geometric models are a similarity framework in which items whose property values are metric are represented as points in a multi-dimensional feature- or property-space. The property values are the coordinates, and similarity is calculated by measuring the distance between the items.

Document Similarity



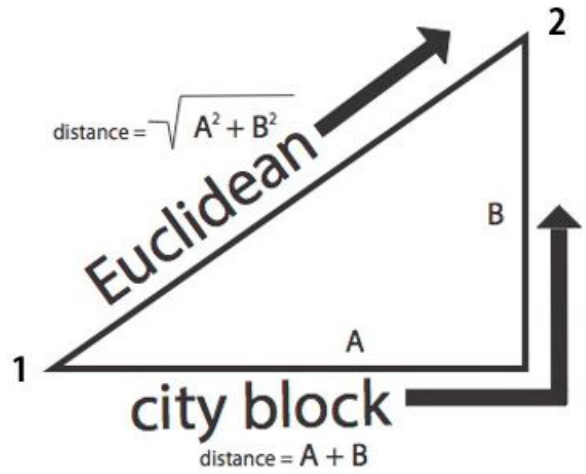
Documents represented as vectors in term space, with the angles between them as a measure of their similarity.

Geometric similarity functions are commonly used by search engines; if a query and document are each represented as a vector of search terms, relevance is determined by the distance between

the vectors in the “term space.” The simplified diagram titled "Document Similarity" depicts four documents whose locations in the term space are determined by how many of each of three terms they contain. The document vectors are normalized to length 1, which makes it possible to use the cosine of the angle between any two documents as a measure of their similarity. Documents d1 and d2 are more similar to each other than documents d3 and d4 because the angle between the former pair (Θ) is smaller than the angle between the latter (Φ).

If the vectors that represent items in a multi-dimensional property space are of different lengths, instead of calculating similarity using cosines we need to calculate similarity in a way that more explicitly considers the differences on each dimension.

Geometric Distance Functions



The distance between points 1 and 2 depends on how the distance function combines the differences in values (A and B) on each dimension.

The diagram "Geometric Distance Functions" shows two different ways of calculating the distance between points 1 and 2 using the differences A and B. The Euclidean distance function takes the square root of the sum of the squared differences on each dimension; in two dimensions, this is the familiar Pythagorean Theorem to calculate the length of the hypotenuse of a right triangle, where the exponent applied to the differences is 2. In contrast, the City Block distance function, so-named because it is the natural way to measure distances in cities with “gridlike” street plans, adds up the differences on each dimension, which is equivalent to an exponent of 1.

We can interpret the exponent as a weighting function that determines the relative contribution of each property to the overall distance or similarity calculation. The choice of exponent depends on the type of properties that characterize a domain and how people make category judgments within it. The exponent of 1 in the City Block function ensures that each property contributes its full amount. As the exponent grows larger, it magnifies the impact of the properties on which differences are the largest.

7.3.6.3. Transformational Models of Similarity

Transformational models assume that the similarity between two things is inversely proportional to the complexity of the transformation required to turn one into the other. The simplest transformational model of similarity counts the number of properties that would need to change their values. More generally, one way to perform the name matching task of determining when two different strings denote the same person, object, or other named entity is to calculate the "edit distance" between them; the number of changes required to transform one into the other.

7.3.6.4. Alignment or Analogy Models of Similarity

None of the previous types of similarity models works very well when comparing things that have lots of internal or relational structure. In these cases, calculations based on matching features is insufficient; you need to compare features that align because they have the same role in structures or relationships. For example, a car with a green wheel and a truck with a green hood both share the feature green, but this matching feature does not increase their similarity much because the car's wheel does not align with the truck's hood. On the other hand, analogy lets us say that an atom is like the solar system. They have no common properties, but they share the relationship of having smaller objects revolving around a large one.

7.3.7. Goal-Derived Categories

Another psychological principle for creating categories is to organize resources that go together to satisfy a goal. Consider the category "Things to take from a burning house," an example that cognitive scientist Lawrence Barsalou termed an ad hoc or goal-derived category.

What things would you take from your house if a fire threatened it?? Possibly your cat, your wallet and checkbook, important papers like birth certificates and passports, and grandma's old photo album, and anything else you think is important, priceless, or irreplaceable—as long as you can carry it. These items have no discernible properties in common, except for being your most precious possessions. The category is derived or induced by a particular goal in some specified context.

A similar goal-derived category is "Things used at the gym," which might contain a hand towel, a music player with headphones, and a bottle of water.

7.3.8. Theory-Based Categories

A final psychological principle for creating categories is organizing things in ways that fit a theory or story that makes a particular categorization sensible. A theory-based category can win out even if probabilistic categorization, on the basis of family resemblance or similarity with respect to visible properties, would lead to a different category assignment. For example, a theory of phase change explains why liquid water, ice, and steam are all the same chemical compound even though they share few visible properties.

Theory-based categories based on origin or causation are especially important with highly inventive and computational resources because unlike natural kinds of physical resources, little or none of what they can do or how they behave is visible on the surface. Consider all of the different appearances and form factors of the resources that we categorize as “computers” — their essence is that they all compute, an invisible or theory-like principle that does not depend on their visible properties.

7.4. Category Design Issues and Implications

We have previously discussed resource properties, similarity, and goals as principles for defining categories. When we use these principles to develop a system of categories, we must make decisions about the system's depth and breadth. Here, we examine the idea that some levels of abstraction in a system of categories are more basic or natural than others. We also consider how the choices we make shape our interactions when we need to find some resources.

7.4.1. Category Abstraction and Granularity

We can identify any resource as a unique instance or as a member of a class of resources. The size of this class—the number of resources that we treat as equivalent—is determined by the properties or characteristics we consider. The context and our intent influence how we think of a resource domain. The same resource can be thought of abstractly in some situations and very concretely in others.

Consider the regular chore of putting away clean clothes. When we consider something to be an item of clothing, we are putting it in a broad category whose members are any type of garment that a person might wear. However, using only one category for all clothing and never distinguishing the various items in any useful or practical way would mean that we keep our clothes in a big unorganized pile.

However, we cannot wear any random combination of items of clothing—we need a shirt, a pair of pants, socks, and so on. A single clothing category is too broad for most purposes. Instead, most people organize their clothes in more fine-grained categories that align better with how they select clothes to wear.

In §7.3.2, “Single Properties” we described an organizing system for the shirts in our closet, so let us talk about socks instead. Most people wear two socks at a time. If you wear socks in pairs, it seems sensible to organize them as pairs when you are putting them away. Some people might further separate their dress socks from athletic ones, and then sort these socks by color or material, creating a hierarchy of sock categories analogous to the shirt categories in our previous example.

Questions of resource abstraction and granularity also emerge whenever the information systems of different firms, or different parts of a firm, need to exchange information or be combined to create a single system. All parties must define the identity of each thing in the same way, or in ways that can be related to each other either manually or electronically.

For example, how should a business system deal with a customer's address? Printed on an envelope, "an address" looks like a multi-line text object. Inside an information system, however, an address is best stored as a set of separate information components. This fine-grained organization makes it easier to sort customers by city or postal codes for sales and marketing purposes. Incompatibilities in the abstraction and granularity of these information components can cause interoperability problems when businesses need to share information.

7.4.2. Basic or Natural Categories

Categories are often described in terms of where they fit in a hierarchy of superordinate, basic, and subordinate category levels. "Clothing," for example, is a superordinate category, "shirts" and "socks" are basic categories, and "white long-sleeve dress shirts" and "white wool hiking socks" are subordinate categories. Members of basic level categories like "shirts" and "socks" share many perceivable properties. In contrast, members of superordinate categories have fewer common properties. Finally, members of subordinate categories have many common properties, but these properties are also shared by members of other subordinate categories at the same level of abstraction in the category hierarchy. That is, while we can identify many properties shared by all "white long-sleeve dress shirts," many of them are also properties of "blue long-sleeve dress shirts" and "black long-sleeve pullover shirts."

7.4.3. The Recall / Precision Tradeoff

The abstraction level we choose when we define a category determines how precisely we identify the resources contained in the category. When we want to make a general claim or communicate a broad interest, we use superordinate categories, as when we ask, "How many animals are in the San Diego Zoo?" In contrast, we use precise subordinate categories when we need to be specific: "How many adult emus are in the San Diego Zoo today?"

If we return to our clothing example, it is easy to find a pair of white wool hiking socks if the organizing system for socks has fine-grained categories. When resources are described and arranged with this level of detail, a similarly detailed specification of the resources you are looking for yields precisely what you want. When you get to the place where you keep white wool hiking socks, you find all of them and nothing else. On the other hand, if all your socks are tossed unsorted into a sock drawer, you might not be able to find the socks you want, and you encounter many socks you do not want. However, you would not have put time into sorting them, which many people do not enjoy doing; you can spend time sorting or searching depending on your preferences.

If we translate this example into the jargon of information retrieval, we say that more fine-grained organization reduces recall. the number of resources you find or retrieve in response to a query. At the sane tine, fine-grained organization increases the precision of the recalled set, the proportion of recalled items that are relevant. Broader or coarse-grained categories increase recall, but lower precision. We are all too familiar with this hard bargain when we use a web search engine; a quick one-word query results in many pages of mostly irrelevant sites, whereas a carefully crafted multi-word query pinpoints sites with the information we seek.

This tradeoff between the investment in organization and the investment in retrieval persists in nearly every organizing system. The more effort we put into organizing resources, the more effectively they can be retrieved. The more effort we are willing to put into retrieving resources, the less they need to be organized first. The allocation of costs and benefits between the organizer and retriever differs according to the relationship between them. Are they the same person? Who does the work and who gets the benefit?

7.5. Implementing Categories

Categories are conceptual constructs that we use in a mostly invisible way when we talk or think about them. When we organize our kitchens, closets, or file cabinets using shelves, drawers, and folders, these physical locations and containers are the visible implementations of our category system, but they are not the categories. This distinction between category design and implementation is obvious when we follow signs and labels in libraries or grocery stores to find things, search a product catalog or company personnel directory, or analyze a set of economic data assembled by the government from income tax forms. These categories were designed by people prior to the assignment of resources to them.

This separation between category creation and category implementation prompts us to ask how a system of categories can be implemented. We do not discuss the implementation of categories in the technical sense of building physical or software systems that organize resources. Instead, we take a higher-level perspective that analyzes the implementation problem to be solved for the different types of categories discussed in §7.3, and then explain the logic we follow to assign resources correctly to them.

7.5.1. Implementing Enumerated Categories

Enumerated categories are easy to implement. The members in a set define the category, and testing an item for membership means looking in the set for it. Enumerated category definitions are familiar in drop-down menus and form-filling. You scroll through a list of countries in the world to search for the one you want in an address, and whatever you select is a valid country name, because the list only changes if a new country comes into being. Enumerated categories can also be implemented with associative arrays (also known as hash tables or dictionaries). With these data structures, a test for set membership is even more efficient than searching, because it takes the same time for sets of any size.

7.5.2. Implementing Categories Defined by Properties

The most conceptually simple and straightforward implementation of categories adopts the classical view of categories based on necessary and sufficient features. Because such categories are prescriptive with explicit and clear boundaries, classifying items into the categories is objective and deterministic. There is an unambiguous method of testing or validation to determine whether some instance is a member of the category. The tests are rules that mention the required properties and property values:

- If instance X has property P, then X is in category Y.

- For a number to be classified as prime it must satisfy two rules: It must be greater than 1, and have no positive divisors other than 1 and itself.

The results of these tests are unambiguous. The item is either a member of the category or it isn't.

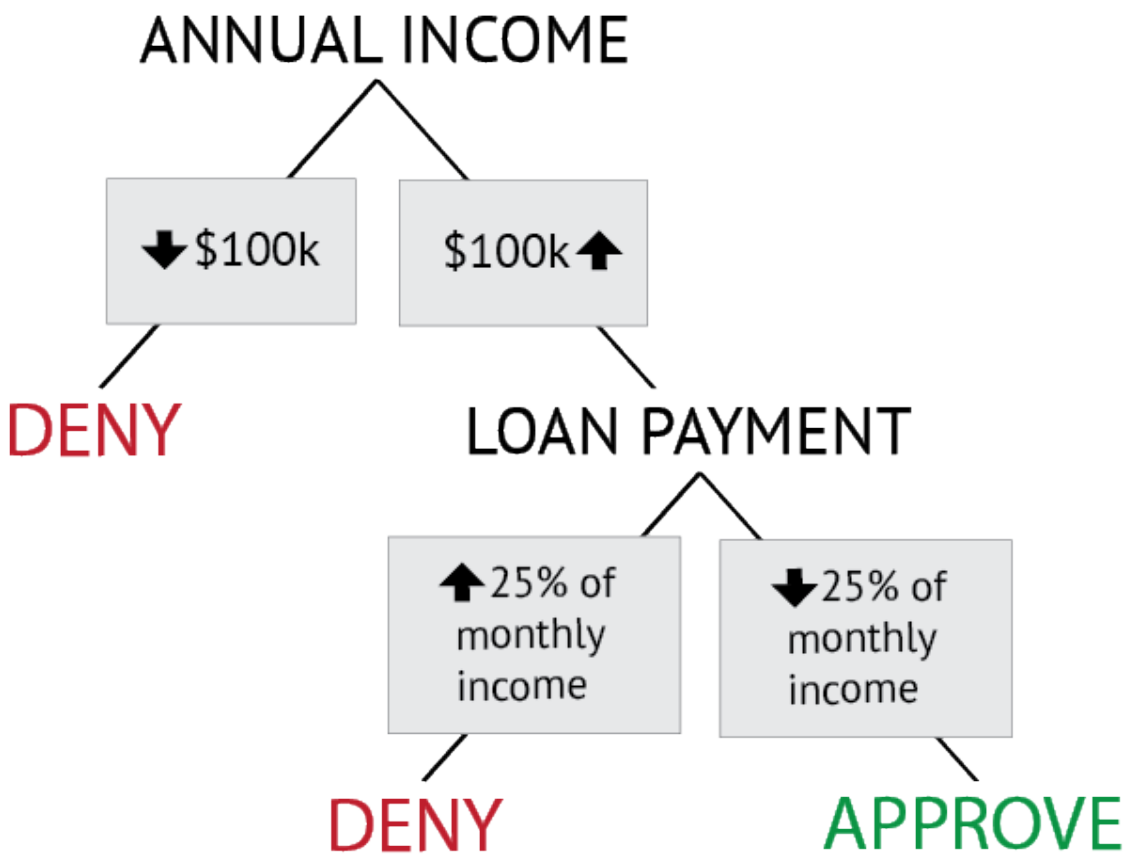
A system of hierarchical categories is defined by a sequence of property tests in a particular order. The most natural way to implement multi-level category systems is with a decision tree, . an algorithm that makes a decision using a sequence of logical or property tests.

Suppose a bank uses a sequential rule-based approach to decide whether to give someone a mortgage loan.

- If applicant's annual income exceeds $100,000, and if the monthly loan payment is less than 25% of monthly income, approve the mortgage application.
- Otherwise, deny the loan application.

This simple decision tree is depicted in Figure 7.1, "Rule-based Decision Tree". The rules used by the bank to classify loan applications as "Approved" or "Denied" have a direct representation in the tree. The easy interpretation of decision trees makes them a common implementation for classification models.

Figure 7.1. Rule-based Decision Tree



In this simple decision tree, a sequence of two tests - one for the borrower's annual income, and one for the percentage of monthly income required to make the loan payment - classify the applicants into the “deny” and “approve” categories.

Any implementation of a category is only interpretable to the extent that the properties and tests it uses in its definition can be understood. Because natural language is inherently ambiguous, it is often not the optimal representational format for formally defined institutional categories. Categories defined using natural language can be incomplete, inconsistent, or ambiguous because words often have multiple meanings. For example, because the implementation uses words like "wealthy" and "easily" rather than precise values, this bank’s procedure for evaluating loans would be hard to interpret reliably:

- If the applicant is wealthy, and then if the monthly payment is an amount that the applicant can easily repay, then the applicant is approved.

Instead, decision trees are sometimes specified using the controlled vocabularies and constrained syntax of “simplified writing” or “business rule” systems to make them more interpretable.

Even more than controlled vocabularies, artificial languages are a more ambitious way to enable precise specification of property-based categories. An artificial language expresses ideas concisely by introducing new terms or symbols that represent complex ideas along with syntactic mechanisms for combining and operating on them. Mathematical notation, programming languages, schema languages that define valid document instances, and regular expressions that define search and selection patterns are familiar examples of artificial languages.

Artificial languages for defining categories have a long history in philosophy and science. However, the vast majority of institutional category systems are still specified with natural language, despite its ambiguities because people usually understand the languages they learned naturally better than artificial ones. Sometimes categories are defined using natural language to allow institutional categories embodied in laws to evolve in the courts and to accommodate technological advances.

Data schemas that specify data entities, elements, identifiers, attributes, and relationships in databases and XML document types on the transactional end of the Document Type Spectrum (§4.2.1) are implementations of the categories needed for the design, development and maintenance of information organization systems. Data schemas tend to rigidly define categories of resources.

In object-oriented programming languages, classes are templates for the creation of objects. A class in a programming language is analogous to a database schema that specifies the structure of its member instances, in that the class definition specifies how instances of the class are constructed in terms of data types and possible values. Programming classes may also specify whether data in a member object can be accessed, and if so, how.

Unlike transactional document types, which can be prescriptively defined as classical categories because they are often produced and consumed by automated processes, narrative document

types are usually descriptively defined. We do not classify something as a novel because it has some specific set of properties and content types. Instead, we have a notion of typical novels and their characteristic properties, and some things that are considered novels are far from typical in their structure and content.

7.5.3. Implementing Categories Defined by Probability and Similarity

Many categories cannot be defined using tests for required properties, and instead must be defined probabilistically; category membership is determined by properties that resources are likely to have, not by properties they must have. Consider the category "friend." You probably consider many people to be your friends, but you have longtime friends, school friends, workplace friends, friends you see only at the gym, and friends of your parents. Each of these types of friends represents a different cluster of common properties. If someone is described to you as a potential friend, how accurately can you predict that the person will become a friend?

Probabilistic categories can be challenging to define and use because it can be difficult to keep in mind the complex feature correlations and probabilities exhibited by different clusters of instances. Furthermore, when the category being learned is broad with a large number of members, the sample from which you learn strongly shapes what you learn. For example, people who grow up in high-density and diverse urban areas may have less predictable ideas of who an acceptable friend might be than someone in a remote rural area with a more homogeneous population.

More generally, if you are organizing a domain where the resources are active, change their state, or are measurements of properties that vary and co-occur probabilistically, the sample you choose strongly affects the accuracy of models for classification or prediction. In the book The Signal and the Noise, statistician Nate Silver explains how many notable predictions failed because of poor sampling techniques. One common sampling mistake is to use too short a historical window to assemble the training dataset; this is often a corollary of a second mistake, an over-reliance on recent data because it is more available.

7.5.3.1. Probabilistic Decision Trees

In §7.5.2, we showed how a rule-based decision tree could be used to implement a strict property-based classification in which a bank uses tests for the properties of "annual income" and "monthly loan payment" to classify applicants as approved or denied. We can adapt that example to illustrate probabilistic decision trees, which are better suited for implementing categories in which category membership is probabilistic rather than absolute.

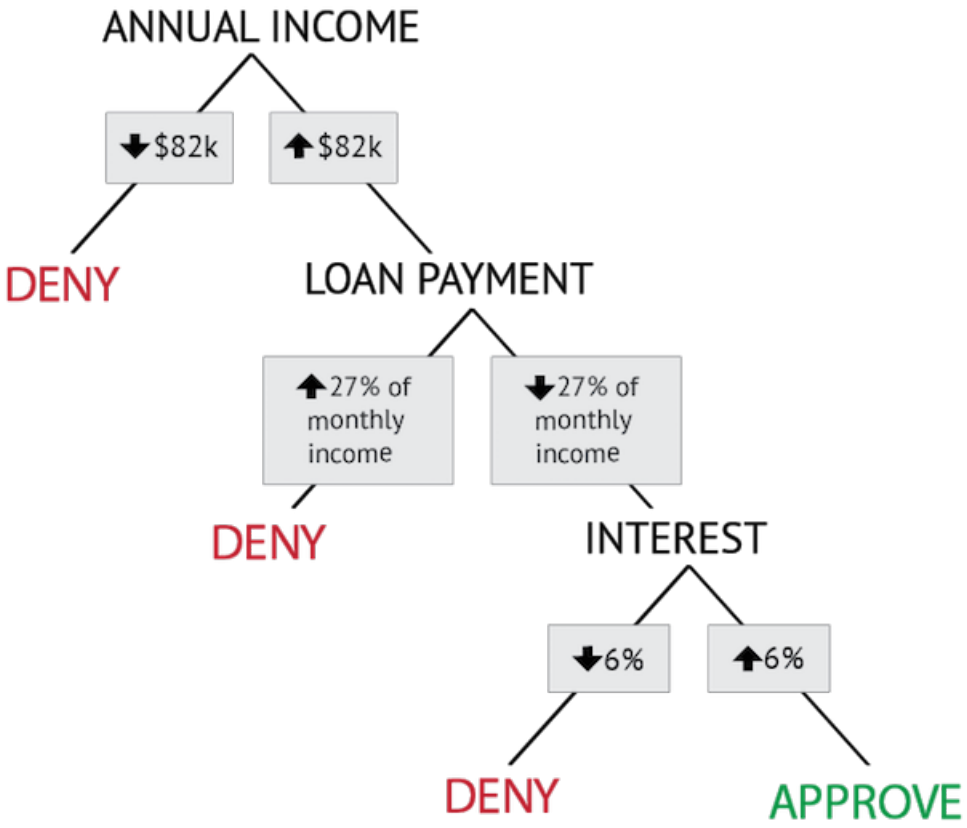
Banks that are more flexible about making loans can be more profitable because they can make loans to people that a stricter bank would reject. Instead of enforcing conservative and fixed cutoffs on income and monthly payments, these banks evaluate applications in a more probabilistic way. These banks recognize that not every loan applicant who is likely to repay the loan looks exactly the same; "annual income" and "monthly loan payment" remain important properties, but other factors might also be useful predictors, and there is more than one configuration of values that an applicant could satisfy to be approved for a loan.

Which properties of applicants best predict whether they will repay the loan or default? A property that predicts each at 50% isn't helpful, but a property that splits the applicants into two sets with very different probabilities for repayment and defaulting is very helpful in making a loan decision.

A data-driven bank relies upon historical data about loan repayment and defaults to train algorithms that create decision trees by repeatedly splitting the applicants into subsets that are most different in their predictions. Subsets of applicants with a high probability of repayment would be approved, and those with a high probability of default would be denied a loan. One method for selecting the property test for making each split is calculating the "information gain". This measure captures the degree to which each subset contains a "pure" group in which every applicant is classified the same, as likely repayers or likely defaulters.

This calculation is carried out for each of the attributes in the historical data set to identify the one that best divides the applicants into the repaid and defaulted categories. The attributes and the value that defines the decision rule can then be ordered to create a decision tree similar to the rule-based one we saw in §7.5.2. In our hypothetical case, it turns out that the best order in which to test the properties is Income, Monthly Payment, and Interest Rate, as shown in Figure 7.2, "Probabilistic Decision Tree".

Figure 7.2 Probabilistic Decision Tree



The end result is still a set of rules, but behind each decision in the tree are probabilities based on historical data that can more accurately predict whether an applicant will repay or default. Thus, instead of the arbitrary cutoffs at $100,000 in income and 25% for monthly payment, the bank can offer loans to people with lower incomes and remain profitable doing so, because it knows from historical data that $82,000 and 27% are the optimal decision points. Using the interest rate in their decision process is an additional test to ensure that people can afford to make loan payments even if interest rates go up.

7.5.3.2. Naïve Bayes Classifiers

Another commonly used approach to implement a classifier for probabilistic categories is called Naïve Bayes. It employs Bayes' Theorem for learning the importance of a particular property for correct classification. There are some common sense ideas that are embodied in Bayes' Theorem:

- When you have a hypothesis or prior belief about the relationship between a property and a classification, new evidence that is consistent with that belief should increase your confidence.
- Contradictory evidence should reduce confidence in your belief.
- If the base rate for some kind of event is low, do not forget that when you make a prediction or classification for a new specific instance. It is easy to be overly influenced by recent information.

We can translate these ideas into calculations about how learning takes place. For property A and classification B, Bayes' Theorem says:

$$P(A \mid B) = P(B|A)\, P(A) / P(B)$$

The left hand side of the equation, P (A | B), is what we want to estimate but can't measure directly: the probability that A is the correct classification for an item or observation that has property B. This is called the conditional or posterior probability because it is estimated after seeing the evidence of property B.

P (B | A) is the probability that any item correctly classified as A has property B. This is called the likelihood function.

P (A) and P (B) are the independent or prior probabilities of A and B; what proportion of the items are classified as A? How often does property B occur in some set of items?

Using Bayes' Theorem to Calculate Conditional Probability

Your personal library contains 60% fiction and 40% nonfiction books. All of the fiction books are in ebook format, and half of the nonfiction books are ebooks and half are in print format. If you pick a book at random and it is in ebook format, what is the probability that it is nonfiction?

Bayes' Theorem tells us that:

$$P(\text{nonfiction} \mid \text{ebook}) = P(\text{ebook} \mid \text{nonfiction}) \times P(\text{nonfiction}) / P(\text{ebook}).$$

We know: $P(\text{ebook} \mid \text{nonfiction}) = .5$ and $P(\text{nonfiction}) = .4$

We compute P (ebook) using the law of total probability to compute the combined probability of all the independent ways in which an ebook might be sampled. In this example there are two ways:

$$\begin{aligned} P(\text{ebook}) &= P(\text{ebook} \mid \text{nonfiction}) \times P(\text{nonfiction}) \\ &\quad + P(\text{ebook} \mid \text{fiction}) \times P(\text{fiction}) \\ &= (.5 \times .4) + (1 \times .6) = .8 \end{aligned}$$

Therefore: $P(\text{nonfiction} \mid \text{ebook}) = (.5 \times .4) / .8 = .25$

Now let's apply Bayes' Theorem to implement email spam filtering. Messages are classified as SPAM or HAM (i.e., non-SPAM); the former are sent to a SPAM folder, while the latter head to your inbox.

1. Select Properties. We start with a set of properties, some from the message metadata like the sender's email address or the number of recipients, and some from the message content. Every word that appears in messages can be treated as a separate property
2. Assemble Training Data. We assemble a set of email message that have been correctly assigned to the SPAM and HAM categories. These labeled instances make up the training set.
3. Analyze the Training Data. For each message, does it contain a particular property? For each message, is it classified as SPAM? If a message is classified as SPAM, does it contain a particular property? (These are the three probabilities on the right side of the Bayes equation).
4. Learn. The conditional probability (the left side of the Bayes equation) is recalculated, adjusting the predictive value of each property. Taken together, all of the properties are now able to correctly assign (most of) the messages into the categories they belonged to in the training set.
5. Classify. The trained classifier is now ready to classify uncategorized messages to the SPAM or HAM categories.
6. Improve. The classifier can improve its accuracy if the user gives it feedback by reclassifying SPAM messages as HAM ones or vice versa..

7.5.3.3. Categories Created by Clustering

In the previous two sections we discussed how probabilistic decision trees and naïve Bayes classifiers implement categories that are defined by typically shared properties and similarity. Both are examples of supervised learning because they need correctly classified examples as training data, and they learn the categories they are taught.

In contrast, clustering techniques are unsupervised; they analyze a collection of resources to discover statistical regularities or structure among the items. These techniques create a set of categories without any labeled training data.

Clustering techniques are used to create meaningful categories from a collection of items whose properties are hard to directly perceive and evaluate. In this situation category membership cannot easily be reduced to specific property tests and instead must be based on similarity. For example, with large sets of documents or behavioral data, clustering techniques can find categories of documents with the same topics, genre, or sentiment, or categories of people with similar habits and preferences.

Because clustering techniques are unsupervised, they create categories based on calculations of similarity between resources, maximizing the similarity of resources within a category and maximizing the differences between them. These statistically-learned categories are not always meaningful ones that can be named and used by people. Some clustering techniques for text resources suggest names for the clusters based on the important words in documents at the center of each cluster. However, unless there is a labeled set of resources from the same domain that can be used as a check to see if the clustering discovered the same categories, it is up to the data analyst or information scientist to make sense of the discovered clusters or topics.

There are many different distance-based clustering techniques, but they share three basic methods.

- The first shared method is that clustering techniques start with an initially uncategorized set of items or documents that are represented in ways that enable measures of inter-item similarity can be calculated. This representation is most often a vector of property values or the probabilities of different properties, so that items can be represented in a multidimensional space and similarity calculated using a distance function like those described in §7.3.6.2, “Geometric Models of Similarity”. The choice of properties and the methods for calculating similarity can result in very different numbers and types of categories
- The second shared method is that categories are created by putting items that are most similar into the same category. Hierarchical clustering approaches start with every item in its own category. Other approaches, notably one called “K-means clustering,” start with a fixed number of K categories initialized with a randomly chosen item or document from the complete set.
- The third shared method is refining the system of categories by iterative similarity recalculation each time an item is added to a category. Approaches that start with every item in its own category create a hierarchical system of categories by merging the two

most similar categories, re-computing the similarity between the new category and the remaining ones, and repeating this process until all the categories are merged into a single category at the root of a category tree. Techniques that start with a fixed number of categories do not create new ones but instead repeatedly recalculate the "centroid" of the category by adjusting its property representation to the average of all its members after a new member is added.

7.5.3.4. Neural networks

Among the best performing classifiers for categorizing by similarity and probabilistic membership are those implemented using neural networks, and especially those employing deep learning techniques. Deep learning algorithms can learn categories from labeled training data or by using autoencoding, an unsupervised learning technique that trains a neural network to reconstruct its input data. However, instead of using the properties that are defined in the data, deep learning algorithms devise a very large number of features in hidden hierarchical layers, which makes them uninterpretable by people. The key idea that made deep learning possible is the use of "backpropagation" to adjust the weights on features by working backwards from the output (the object classification produced by the network) all the way back to the input.

7.5.4. Implementing Goal-Based Categories

Goal-based categories are highly individualized, and are often used just once in a very specific context. However, it is useful to consider that we could implement model goal-derived categories as rule-based decision trees by ordering the decisions to ensure that any sub-goals are satisfied according to their priority. We could understand the category "Things to take from a burning house" by first asking the question "Are there living things in the house?" because that might be the most important sub-goal. If the answer to that question is "yes," we might proceed along a different path than if the answer is "no." Similarly, we might put a higher priority on things that cannot be replaced (Grandma's photos) than those that can (passport).

7.5.5. Implementing Theory-Based Categories

Theory-based categories arise in domains in which the items to be categorized are characterized by abstract or complex relationships with their features and with each other. With this model an entity need not be understood as inherently possessing features shared in common with another entity. Rather, people project features from one thing to another in a search for congruities between things.

Theory-based categories are created as cognitive constructs when we use analogies and classify, because things brought together by analogy have abstract rather than literal similarity. The most influential model of analogical processing is Structure Mapping, whose development and application has been guided by Dedre Gentner for over three decades.

The key insight in Structure Mapping is that an analogy "a T is like B" is created by matching relational structures and not properties between the base domain B and a target domain T. We take any two things, analyze the relational structures they contain, and align them to find

correspondences between them. The properties of objects in the two domains need not match, and in fact, if too many properties match, analogy goes away and we have literal similarity:

- Analogy: The hydrogen atom is like our solar system
- Literal Similarity: The X12 star system in the Andromeda galaxy is like our solar system

7.6. Key Points in Chapter Seven