


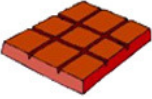




# The strict faceted classification model

Travis Wilson, Facetmap

<http://facetmap.com/pub/>

## Summary of principles

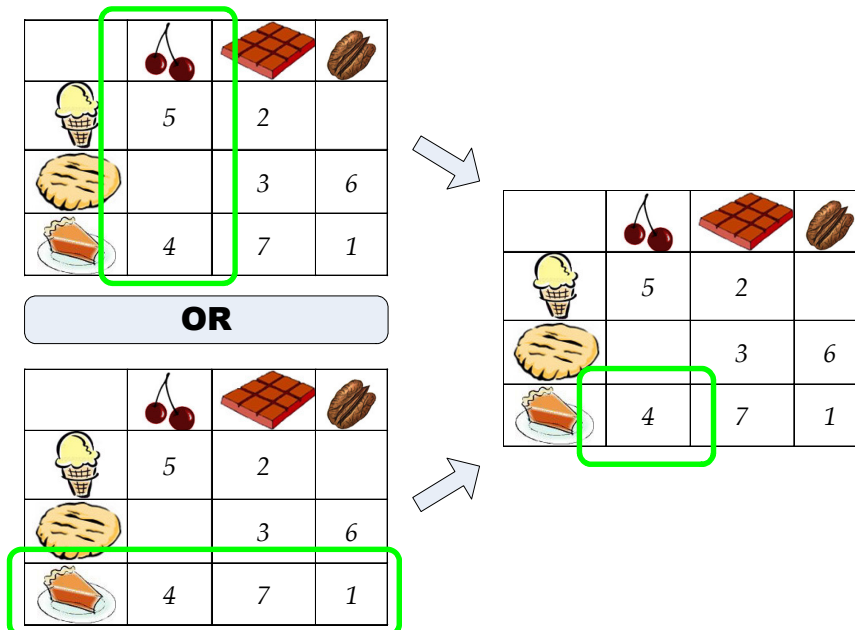
Faceted classification, at its core, implies orthogonality – that every facet axis exists at right angles to (i.e., independently of) every other facet axis. That’s why a faceted classification is sometimes represented with a chart. This set of desserts has been classified by their confection types and, orthogonally, by their flavors:

Flavor	 <b>Cherry</b>	 <b>Chocolate</b>	 <b>Pecan</b>
Confection Type			
 <b>Ice Cream</b>	5	2	
 <b>Cookie</b>		3	6
 <b>Pie</b>	4	7	1

## Menu

1. Pecan Pie
2. Chocolate Ice Cream
3. Chocolate Cookie
4. Cherry Pie
5. Cherry Ice Cream
6. Pecan Cookie
7. Chocolate Pie

One application of this is flexible browsing; the browsing user can filter by row first or column first, and still end up with the same result:



If you're using faceted classification to organize your data, then perhaps you're familiar with this situation: you want to assign something two different headings from the same facet. For example, in the dessert scheme above, you want to assign both “Chocolate” and “Pecan” headings to a chocolate-pecan pie.

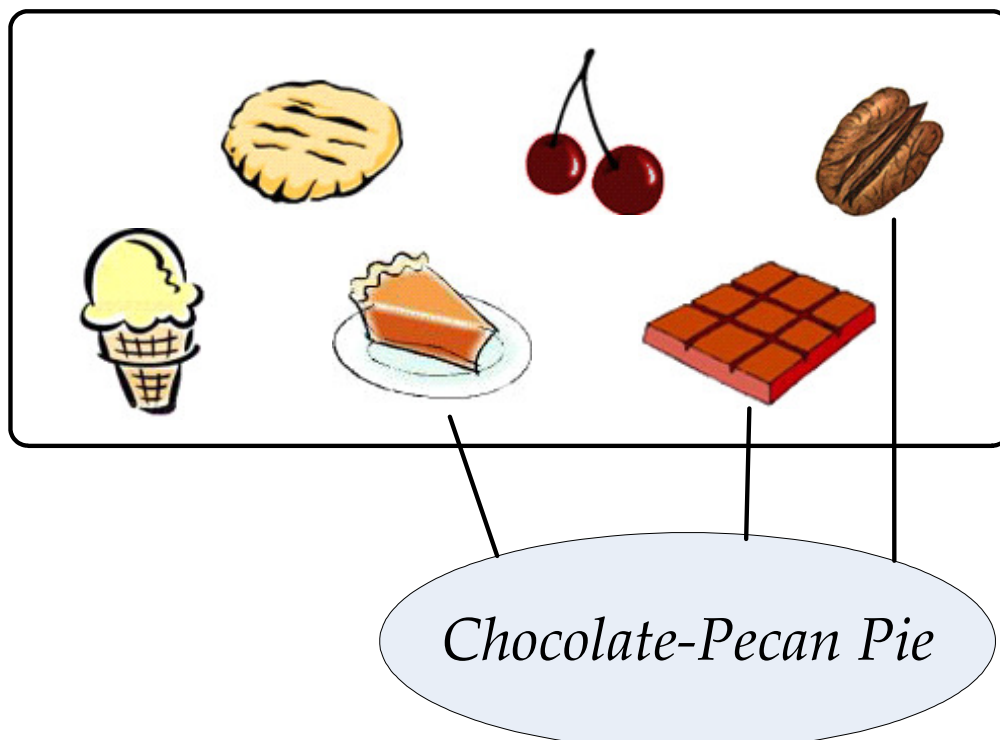
This innocent-looking scenario cuts open a surprisingly wide swath of information theory. It turns out that, given the facets in the example, *a strict faceted classification model forbids you to assign both those headings*, and with good reason. This is counterintuitive, controversial, and if you subscribe to S.R. Ranganathan's original facet theory, heretical.

Yet a faceted classification is most effective when built upon that restriction. This paper explores how to use the restriction constructively, whether the faceted classification model really fits the user's intent, and what the alternatives are. We'll critically examine facet structures, especially in light of such new schemes as tagging. Along the way we'll discover:

- the real nature of the faceted classification model
- the similarities to and differences from free-form tagging
- orthogonality
- binary facets
- recent suggested enhancements to the faceted classification model
- new technology that leverages the strict model to sort tags into facets
- how to make your classification as usable as possible

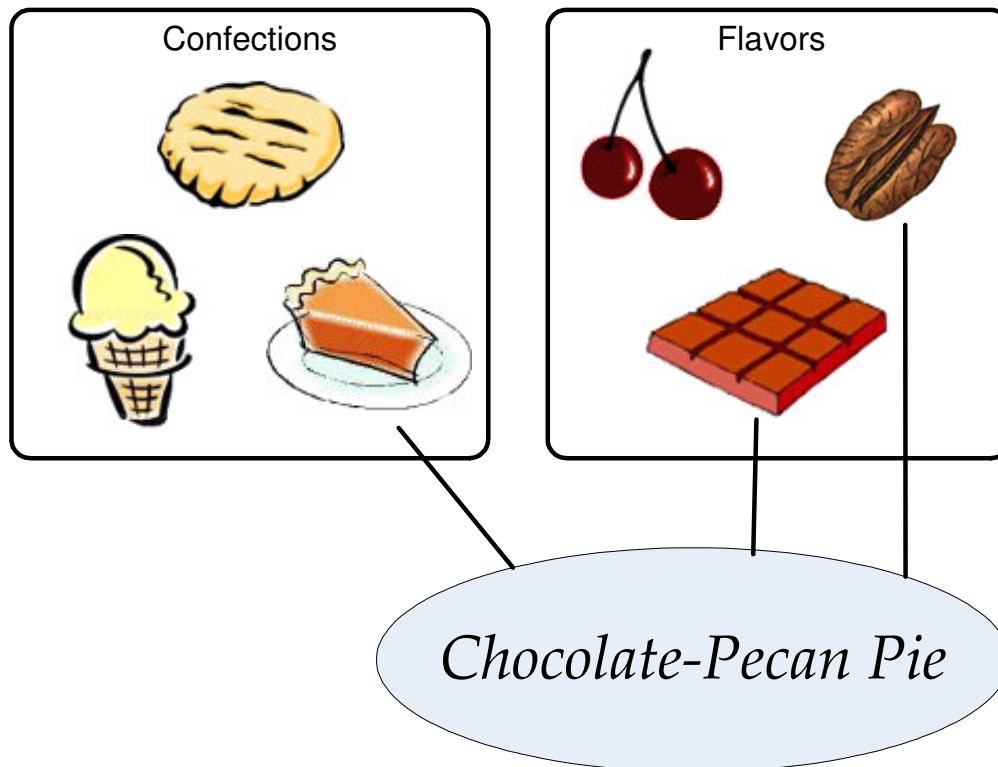
Over the last couple of years, tagging has emerged as a flexible method of assigning headings to resources, regardless of other headings on that resource. Its applications sharply illuminate its differences from faceted classification; in fact tagging is the closer of the two to Ranganathan's canons of facet theory.

Tags do not fit well into a chart model. The topography of tagging is more like this:



The resource can be connected to any tag, to indicate that it expresses the property that the tag represents. This connection can be made regardless of what other tags are already connected to the resource. So tags carry orthogonality to an extreme: *every* tag is orthogonal to every other tag! This can certainly be convenient, but it is different from a faceted classification in that it does not identify which headings are mutually exclusive.

The tagging situation is not really any different if we draw it this way:



The separation of headings into two groups only enables us to present the headings to the user in a more organized fashion, for browsing or for classification. Grouped tags do not constitute a transition to faceted classification; nothing about the classification itself has changed, nothing different happens when you select one of the tags.

In fact, if you organize tags into the five groups of Personality, Matter, Energy, Space, and Time, what do you have? You have the organizational system prescribed by Ranganathan's theory of facet analysis. In this sense, Ranganathan is actually the father of tagging. In fact, the essence of his contribution was the idea that one could define any number of "classifications" for a given resource. Which sounds exactly like tagging, and less like the faceted classification I've been working with at Facetmap since the beginning.

To appreciate Ranganathan's work, let's put it in historical context. The revolution he began was the "Colon Classification" – the idea that a classification could contain more than one piece of information. He was up against the Dewey decimal system where you had to classify the Buddha nature<sup>1</sup> of an object, and come up with a single atom of information, codified by a number. No,

<sup>1</sup> The term "Buddha nature", traceable to Fred Leise, refers to the idea that an object has a singular, objectively determined resting place – and as far as I've seen, has generally been used hyperbolically to suggest that this idea is often fallacious.

said Ranganathan, you didn't have to do that; you could classify each aspect of the object, with pieces of information separated by colons. *That* was the idea!

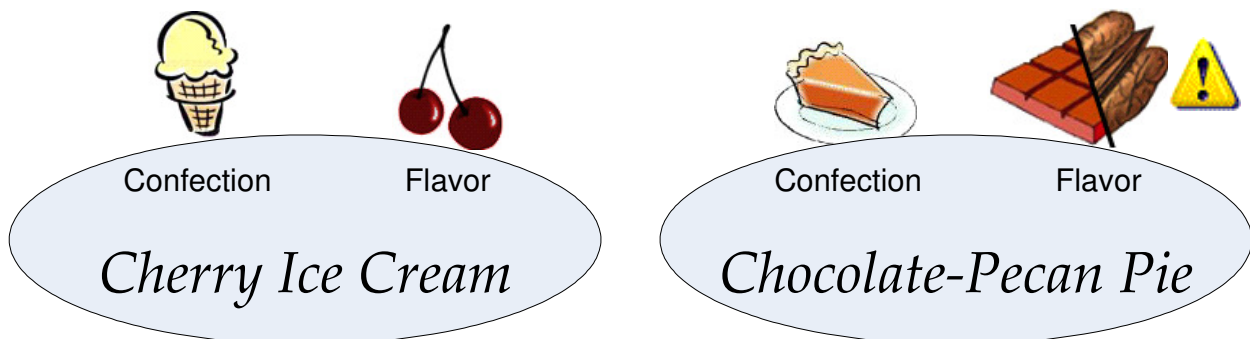
Ranganathan's new tagging system, though, became much more useful when he grouped the tags. He had to group the tags, because a scheme could grow so dangerously out of hand so quickly, with no computers to keep track of it. There were thousands of tags, and they needed their own information architecture, simply so that librarians could look them up easily. But, again, that architecture really has no effect on the assignment of tags to resources; it's just there to provide some sort of order to the tags.

A colon classification of chocolate-pecan pie simply breaks up "chocolate-pecan pie" lexically, expressing each component with a code and separating them with punctuation, and furthermore grouping codes that are intuitively related. The result is something like this:

**CC3 : FF2 ; FF3      A pie confection composed of chocolate and composed of pecan**

If facets are just groups of tags, I say why bother?<sup>2</sup>

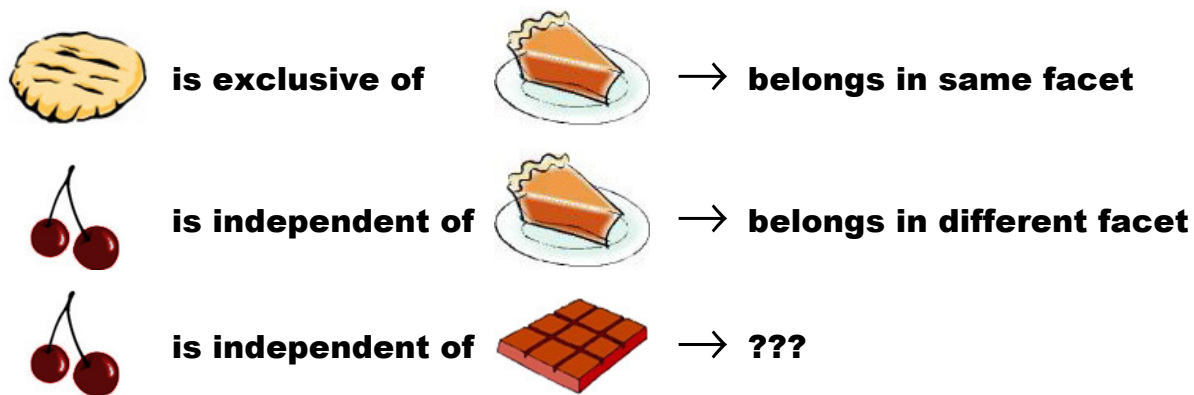
Strict faceted classification provides a different model entirely. Instead of sorting the headings into intuitive groups, it sorts the orthogonal attributes of the resource itself. Originally, we said our desserts had two independent attributes – the "Confection" facet and the "Flavor" facet, and the each facet of each dessert resource was assigned a value thus:



But this creates a huge headache! There is no Flavor heading defined for "Chocolate and Pecan". One solution to the problem is to recast the whole classification into a tagging model, but we want to stay within the strict faceted classification model. So another solution would be to create hybrid headings within "Flavor", but that has the same permutation-explosion problem that we hope to avoid by using faceted classification in the first place.

Consider instead the true meaning of the headings we've used. By allowing a chocolate-pecan pie, or chocolate cherry ice cream, etc., to exist, we're introducing an interesting piece of information: Whether something is "Cherry" has no effect on whether it can or cannot also be "Chocolate". While "Pie" means "the confection type of this resource is pie," "Chocolate" actually only means "this resource contains Chocolate." The heading "Chocolate" only indicates the chocolateness of the dessert and says nothing about the cherriness or pecanness. Should it, then, really be lumped into a facet with those other flavors?

<sup>2</sup> Today "tagging" implies a more open-source model – folksonomies that are hard to manage because users add tags in disorganized fashion, but that's just one way of using tags; you could easily restrain the tagging privilege to a single person without changing the model of tagging itself.



In the same sense that a “Cherry” selection is independent of a “Pie” selection, a “Cherry” selection is also independent of a “Chocolate” selection. By strict facet definition, they are therefore in separate facets.

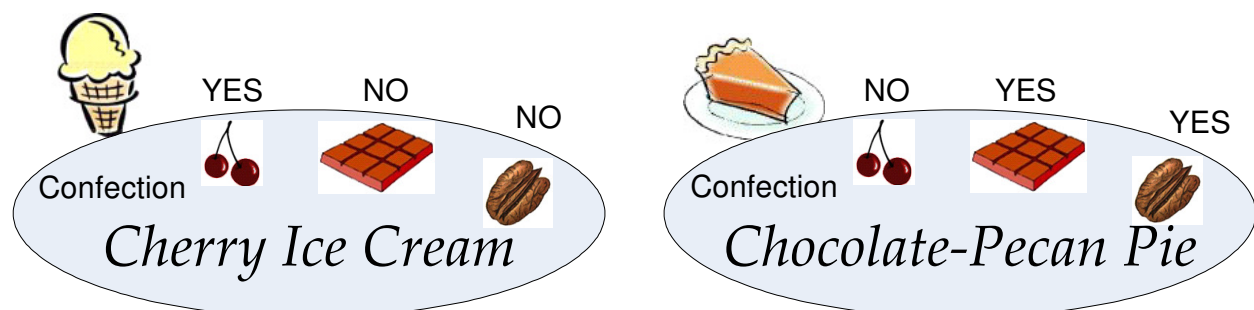
It is important to realize that without this rule, orthogonality is useless. We say that two unrelated headings are orthogonal (i.e. vary independently of each other) and therefore belong in separate facets – this is commonly accepted in facet theory. If, between two facets, headings are orthogonal, then it follows that within a single facet, headings are *not* orthogonal – the assignment of one *does* have an effect on our ability to assign others.

If both halves of this truth are not accepted, then we can only assume that every heading is orthogonal to every other heading, and we’ve fallen back to a tagging system. Therefore, in an attempt to differentiate a facet from a group of tags, we *define a facet* thus:

**Facet:** A set of headings in which the assignment of one heading to a resource limits the assignment to that resource of other headings in the set.

The usual case is simply that the assignment of one heading excludes the assignment of all other headings (sometimes called “mutual exclusivity”, but that term presently has too many definitions to be useful). But I have not ruled out other specialized types of facets, which express more complex limitations of heading assignments.

Here is how to design the dessert facets given this definition, and given the nature of our dessert flavors. We use the same principle Ranganathan used; instead of trying to pack too much information into the “flavor” facet, we split up that information, and our desserts actually look like this when faceted:



Now there are four facets. There are four separate questions that can be asked about a given dessert, for which the answer is independent of all the other answers – so four is the correct

number of facets. Three of them are binary facets, having only “yes” and “no” headings (plus the usually implicit “undefined” heading).

And we’ve added meaning with this new facet scheme. It is in the nature of flavors that *they can be combined*, so the facets reflect that.<sup>3</sup> It is in the nature of confection types that they *cannot be combined*, and we certainly want to reflect that, else there is nothing to signify that a pie-cookie is an illegal classification that will certainly result in problematic baking.

For real-world situations where there are a lot of tags, that means a lot of facets. You will probably still want to group the facets for presentation to the user – but there is nothing inherently wrong with a lot of facets. Between this strict model and a loose-model classification that must anticipate an unknown number of tags on one facet of a resource, the strict model is still more efficient to implement in software.

Why would you use faceted classification at all, when tagging is available? There are several reasons. The right blend for a project, of course, depends on the nature of the project.

## Clarity of information

First and foremost, as I have tried to make clear, facets convey the right amount of semantic information. Data-mining algorithms will analyze the structure of a facet and quickly learn that mixing chocolate and pecan is legal, but mixing pie and cookie is not. We humans understand this intuitively, but we need to reflect the rules in the structure of the data if algorithms are to understand it. In return the algorithms will build more mathematically accurate models of the world.

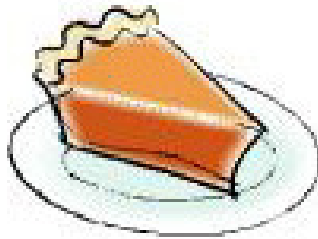
## Clarity of directives for UI software

For a specific example of an algorithm that uses facet structure, look no further than browsing software! Facetmap is a great example of ignorant browsing software – you give it your facets and resources, and it works out a UI, which goes something like this:

---

<sup>3</sup> Jesse James Garrett has made the excellent point that we also *lose* information by doing this – namely, the concept of a “flavor” itself. The conclusion I draw from this is that, in our model, “flavor” was never really defined well in the first place (what *is* the flavor of a chocolate-pecan pie?) and perhaps it should be re-introduced at another level. Some level superficial to faceted classification – maybe a grouping of facets for UI purposes – would probably be appropriate.

Step 1. User selects

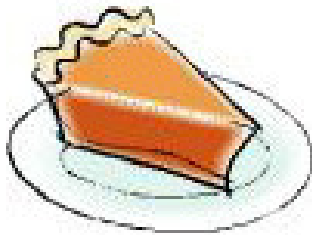


Show results

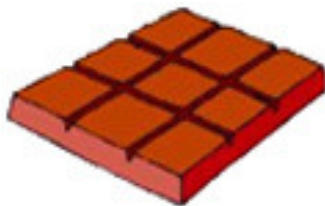
*Cherry Pie*  
*Chocolate Pie*  
*Chocolate-Pecan Pie*

Step 1 is straightforward enough. The result set contains every resource which, when asked “What is your confection type?”, answers “Pie”. But consider the next step:

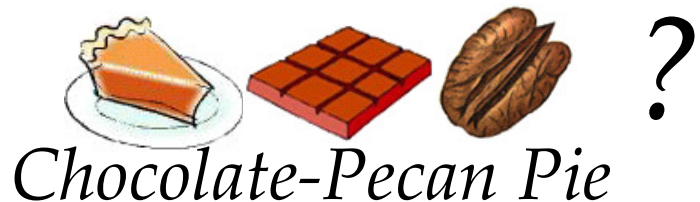
Step 2. User selects



+



Show results



Our original faulty classification, shown above, allows chocolate-pecan pie to map to Chocolate and Pecan in the same facet. The software then has a tricky choice to make when the user selects Chocolate. Is “Chocolate and Pecan” its own heading in the Flavor facet? That would suggest that there is no match with “Chocolate”, and we don’t show the pie to the user. Or does the pie map to Chocolate and also map to Pecan? Which means we do show the pie to the user. Essentially, it is not clear whether the facetmap intends Chocolate AND Pecan or Chocolate OR Pecan.

Topologically, the whole point of faceted classification is that for a particular facet, the resource can actually be *classified*. It can be put in a bucket that’s exclusive of the other buckets. You could think of the facet as a question, and the heading is an answer. If you have an auto part, a "manufacturing location" facet makes sense, because you made the part at that place. You know where to go to get the part, and you know who to call for support, and so on. If there could be multiple headings for the part under “manufacturing location”, though, you have a conflict and

no obvious way to resolve it. You've simply associated the part, not classified it. Strict classification avoids conflicts like that.

We ask questions of our resources all the time often without knowing it. In fact, I've done it for this paper – I need to represent flavors iconically, so I ask the dessert “What is your flavor?” and find the icon that matches the answer. When the answer is a contrivance like “Chocolate-and-pecan”, I have to fire up Photoshop and build a new hybrid icon from the chocolate and pecan icons, and I've already made a bunch of assumptions along the way, not to mention that I need the graphical skill to merge the two icons. This is often too much to ask of software.

If you wanted something as simple as expressing “flavor” headings using colors, you'd have this problem. It is useful to think of facets as questions with answers – multiple choice answers, because software is bad at essay questions. Multiple-choice Q&A scenarios force you to define your possible answers, and also to create the data that corresponds to those answers, like the support resource for the auto part, the icon for chocolate-and-pecan, or the baking instructions for a pie-cookie (if there really is such a thing).

Therefore the facet scheme can also enforce workflow.

## **Runtime efficiency**

Software that processes a strict faceted classification scheme will, as a general rule, scale more easily. For example, in many software implementations, the technical cost of free-form tagging is an increase in table joins.

A table join is a database procedure that is useful but costly. A table join is like a beer; if you have just one, you don't really notice so much. You can still function well. However, projects with eight table joins (I've actually seen these in production) are practically incapacitated. A system like Facetmap Gold, which precompiles all facet data into one master table, has no table joins – but it's theoretically impossible to do that with free-form tags. At some point Facetmap will probably accommodate facets which are actually just groups of tags, which aren't facets as strictly defined in this paper – but they won't be implemented as facets in the software, because they require heavier processing.

## **Tag sorting**

At Facetmap, we like the idea of taking an unsorted mess of tags, such as you've seen on del.icio.us and so on, and automatically organizing it into a number of facets which are fairly intuitive groupings. For example, this works fairly well with wines, a type of resource whose descriptors fit well into categories. Our algorithm (which works now, but is undergoing optimizations) can take this cloud of tags:

▼ tags

10, 103, 109, 11, 117, 12, 126, 127, 13, 14, 15, 150, 16, 160, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 269, 27, 28, 280, 284, 287, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 5, 50, 52, 53, 54, 55, 57, 58, 59, 6, 60, 61, 62, 63, 64, 65, 67, 68, 69, 7, 72, 73, 77, 78, 79, 8, 80, 81, 84, 88, 89, 9, Alsace, Argentina, Austria, Barbera, Burgundy, Cabernet Sauvignon, Cabernet/Merlot Blend, Californian, Carneros, Central Coast, Chablis, Champagne, Champagne-France, Chardonnay, Chenin Blanc, Chianti, Chile, Columbia Valley, Contains\_sulfites, Dry, Extra Dry, French, Friuli-Venezia Giulia, Gewurztraminer, Italian, Loire, Madeira, Marsala, Merlot, Mosel-Saar-Ruwer, Napa Valley, New Zealand, North Coast, Other German, Other Italian, Petite Sirah, Piedmont, Pinot Blanc, Pinot Gris, Pinot Noir, Port, Portuguese, Rhone, Riesling, Rose, Sangiovese, Sauvignon/Fume Blanc, Sherry, Sierra Foothills, Sonoma Valley, Southern Italy, Spanish, Sparkling, Sweet, Trentino-Alto Adige, Tuscany, Umpqua Valley, Vermouth, Walla Walla Valley, White Burgundy, White Merlot, White Zinfandel, Willamette Valley, Yakima Valley, Zinfandel

and, given a set of wine resources that utilize those tags, sort them into facets thus:

<u>Facet 1</u>	<u>Facet 2</u>	<u>Facet 3</u>	<u>Facet 4</u>	<u>Facet 5</u>
Barbera, Cabernet Sauvignon, Cabernet/Merlot Blend, Chablis, Champagne, Chardonnay, Chenin Blanc, Chianti, Gewurztraminer, Madeira, Marsala, Merlot, Petite Sirah, Pinot Blanc, Pinot Gris, Pinot Noir, Port, Riesling, Rose, Sangiovese, Sauvignon/Fume Blanc, Sherry, Sparkling, Vermouth, White Burgundy, White Merlot, White Zinfandel, Zinfandel	Alsace, Argentina, Austria, Burgundy, Californian, Carneros, Central Coast, Champagne-France, Chile, Columbia Valley, French, Friuli-Venezia Giulia, Italian, Loire, Mosel-Saar-Ruwer, Napa Valley, New Zealand, North Coast, Other German, Other Italian, Piedmont, Portuguese, Rhone, Sierra Foothills, Sonoma Valley, Southern Italy, Spanish, Trentino-Alto Adige, Tuscany, Umpqua Valley, Walla Walla Valley, Willamette Valley, Yakima Valley	10, 103, 109, 11, 117, 12, 126, 127, 13, 14, 15, 150, 16, 160, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 269, 27, 28, 280, 284, 287, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 46, 47, 48, 49, 5, 50, 52, 53, 54, 55, 57, 58, 59, 6, 60, 61, 62, 63, 64, 65, 67, 68, 69, 7, 72, 73, 77, 78, 79, 8, 80, 81, 84, 88, 89, 9	Dry, Extra Dry, Sweet	Contains_sulfites

This algorithm looks at how each resource is mapped to the headings, and then it sorts the headings based solely on the principle that each resource doesn't have more than one heading in a certain facet. It creates the facets entirely from that principle.

I like this algorithm because it bridges the gap between facets and tags. Tagging is a simple instance of faceted classification where everything has its own facet, but instead of separating every tag into its own facet, sometimes tags have this exclusivity relationship with each other so they go in the same facet. Sometimes, this provides more information. In this case, it's the information we introduce in order to generate the nice tag groups above.

## **To conclude**

The goal here is for us to think about our facets themselves, and what they imply about the headings they contain. We often treat facets just as groups in which to place tags, but they have an array of possible structures, each of which can carry meaning. Not all facets are equal, and when you clearly define them they can be better used in an interface, and by software.

<http://facetmap.com>

## Appendix

Real-life Facetmap customer example #1:  
Human resource classification

Browse case: "Give me a candidate with [YEARS OF EXPERIENCE] and skills with [SOFTWARE] in [GEOGRAPHICAL LOCATION]"

Intuitive facet set (incorrect):

- Years experience [0 through 20+]
- Location [USA, Canada, Italy, Singapore] (and subheadings)
- Software skills [Excel, Beeline, Peachtree]

Orthogonal facet set (correct):

- Years experience [0 through 20+]
- Location [USA, Canada, Italy, Singapore] (and subheadings)
- Skilled in Excel [True, False]
- Skilled in Beeline [True, False]
- Skilled in Peachtree [True, False]

Sample resource in Facetmap markup:

```
<resource name="Alex Bunker">
  <map facetid="experience" heading="7" />
  <map facetid="location" heading="seattle" />
  <map facetid="skill-excel" heading="true" />
  <map facetid="skill-beeline" heading="false" />
  <map facetid="skill-peachtree" heading="false" />
</resource>
```

## Real-life Facetmap customer example #2: Hospital quality survey

Browse case: "Filter hospital companies by test score and/or by U.S. state"

Intuitive facet set (incorrect):

- Test score [0 through 100]
- States served [AL, AR, AZ, CA, CO, DE, ...]

Orthogonal facet set (correct):

- Test score [0 through 100]
- Serves AL [True, False]
- Serves AR [True, False]
- Serves AZ [True, False]
- Serves CA [True, False]
- Serves CO [True, False]
- Serves DE [True, False]
- ... all the way down!

Sample resource in Facetmap markup:

```
<resource name="GloboHealthStar">
  <!-- only map states this resource serves; others
    default to false -->
  <map facetid="score" heading="93" />
  <map facetid="az" heading="true" />
  <map facetid="ca" heading="true" />
  <map facetid="nv" heading="true" />
  <map facetid="or" heading="true" />
  <map facetid="wa" heading="true" />
</resource>
```

**Alternative: clone resources so there is one per state served.** (This approach is taken by, for example, ISBN numbers for books. The same book has different ISBN numbers for different publications (hardcover, paperback, etc)).

Orthogonal facet set (correct):

- Test score [0 through 100]
- State served [AL, AR, AZ, CA, CO, DE, ...]

## Sample cloned resources in Facetmap markup:

```
<resource name="GloboHealthStar Arizona">
  <map facetid="score" heading="93" />
  <map facetid="state" heading="az" />
</resource>
<resource name="GloboHealthStar California">
  <map facetid="score" heading="93" />
  <map facetid="state" heading="ca" />
</resource>
<resource name="GloboHealthStar Nevada">
  <map facetid="score" heading="93" />
  <map facetid="state" heading="nv" />
</resource>
...
```