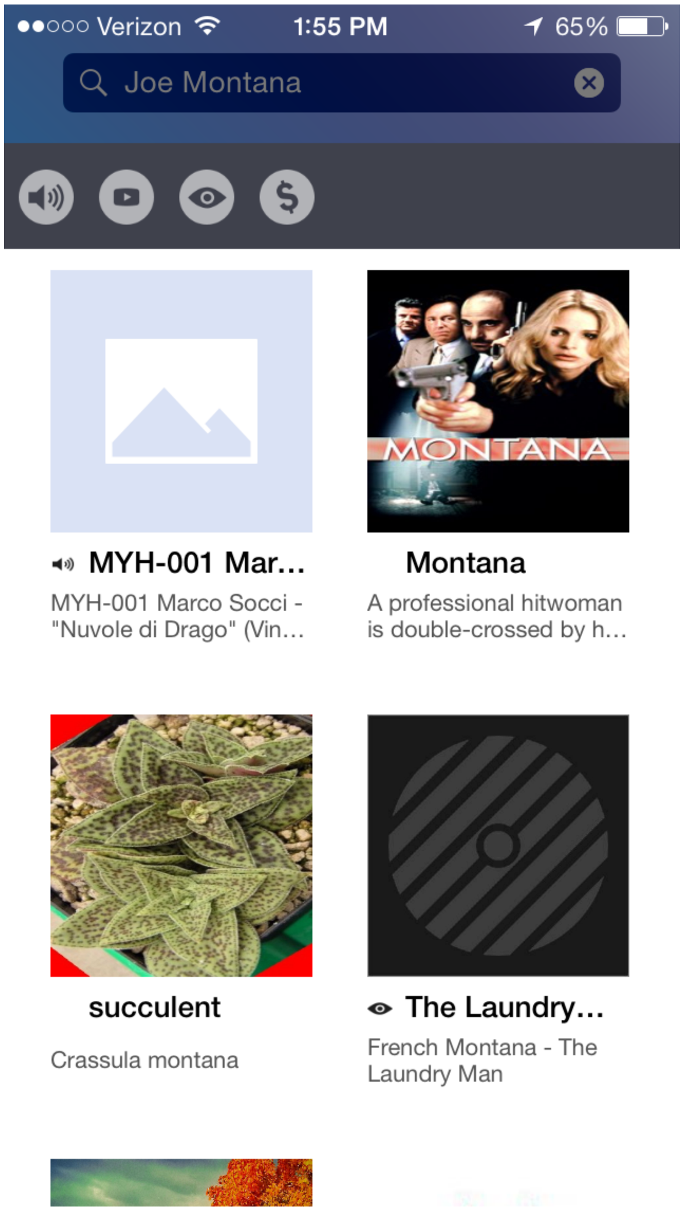
What is the problem?

Consider the following search for **Joe Montana** on our MaestroMusic app.



Joe Montana

MYH-001 Mar…
MYH-001 Marco Socci - "Nuvole di Drago" (Vin…

Montana
A professional hitwoman is double-crossed by h…

succulent
Crassula montana

The Laundry…
French Montana - The Laundry Man

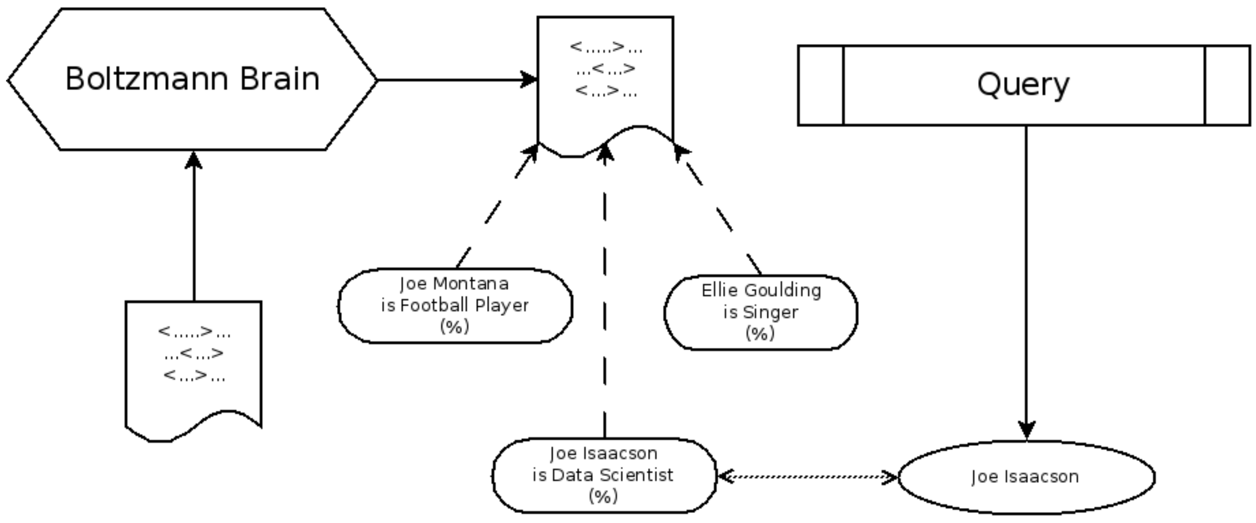
The search query returns results related to **Montana**, but not the football player, Joe Montana.

How did this happen?

Our present search engine relies on text matching to find relevant documents. Without any prior knowledge of the world, the state Montana, Hannah Montana and Joe's last name all appear the same. We need a system that can comprehend the meaning of a document.

How can we fix this?

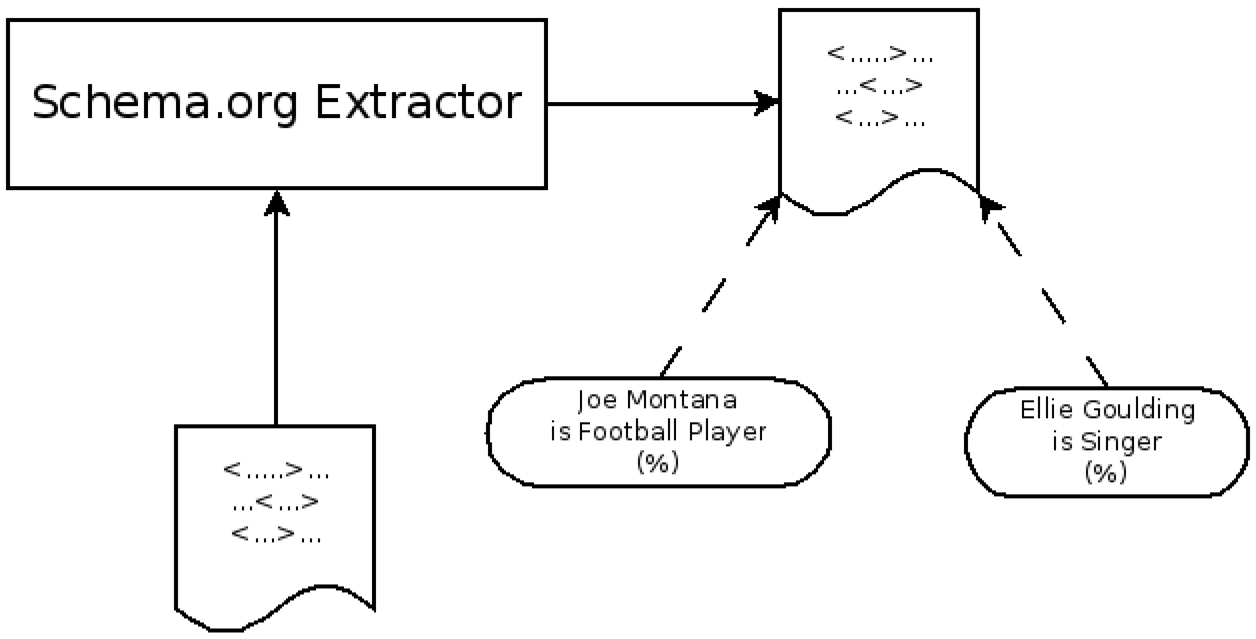
What we would really like to do is match on concepts in the document rather than blindly matching on text in the document. This is where the Boltzmann Brain comes in. Once properly trained, it can take a document and tell us exactly what facts the document tells us about (with some level of confidence for each fact).



In this example, the Boltzman Brain found that the given document contains the facts "Joe Isaacson is a Data Scientist", "Joe Montana is a Football Player", and "Ellie Goulding is a Singer". Now, we can match on the "idea" of "Joe Isaacson" rather than just blindly matching text.

What do we have now? Schema.org Extractor.

Up until this point, we have been ignoring any deep information that the text may provide and instead have been relying only on schema.org data generated by site developers. Here's how this works:



While this is nice to have, it has a few problems:

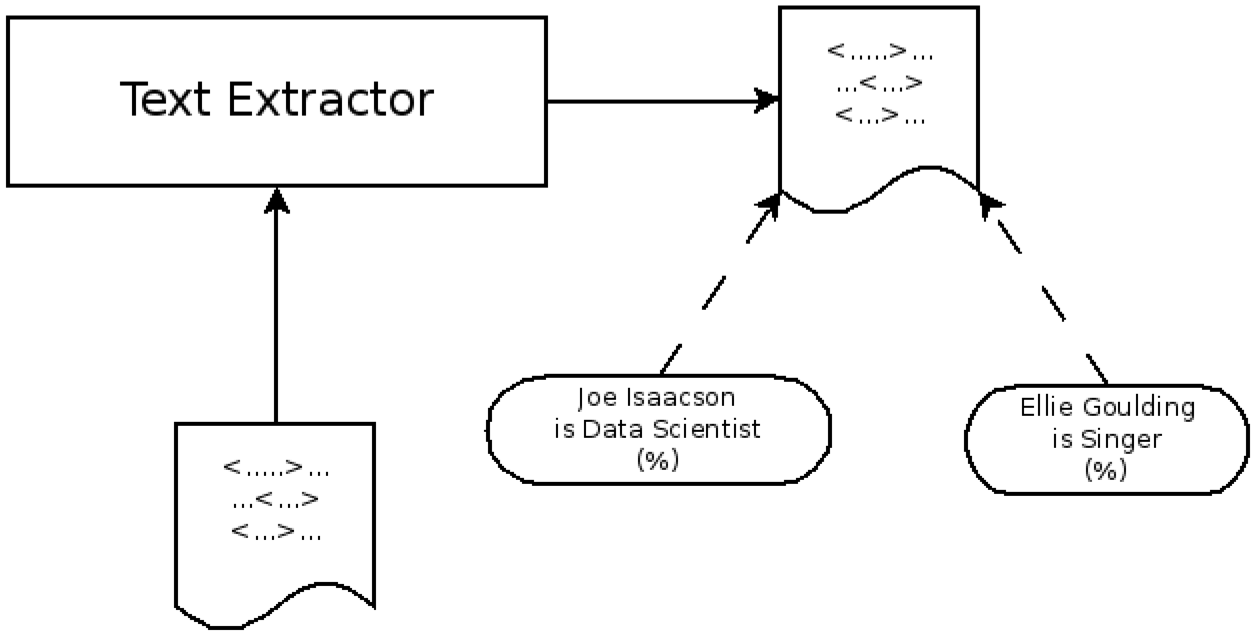
1. Some sites do not have schema.org data
2. Sometimes the schema.org data is wrong
3. Sometimes the schema.org data is incomplete

This is exactly why we have confidences (%) for each fact that we extract. The “extractor” is actually a machine learning algorithm that learns which schema.org data is reliable, and which is unreliable or wrong. See the *Why Freebase?* section for how freebase is used by this algorithm to learn how to do this.

So now we have a situation where we have an algorithm that is good in some situations (sites have immaculate schema.org data), but horribly bad in others (sites have crap schema.org data). Can we do better?

Can we do better? Text Extractor.

The text extractor works much the same way as the schema.org extractor, except it analyzes sentences to extract data rather than using schema.org data. Here’s how it works.

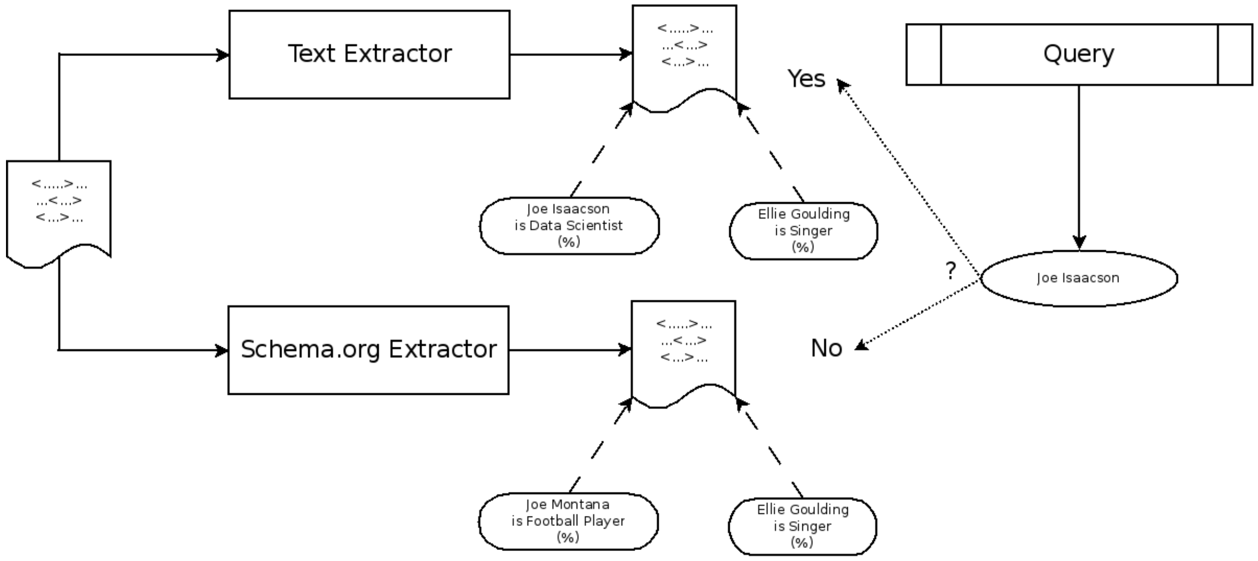


You'll notice that this looks almost exactly the same as the schema.org extractor. That's because it works exactly the same way. The only difference is that this will find different facts, with different confidences (%), than the schema.org extractor.

Who is correct?

So now we have these two separate extractors, which is great. We have lots of information about what's in the documents. In situations where the text doesn't give us much information, we might get good data from schema.org markup, and vice versa.

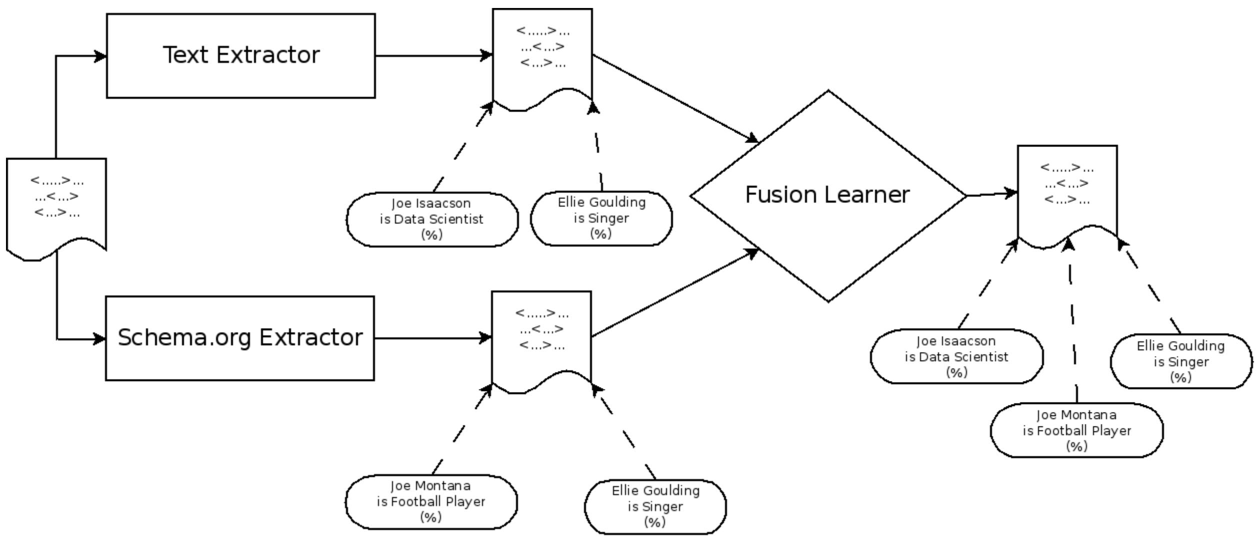
There's one problem though: how do we know who's right? These extractors may not find the same facts, and in fact sometimes they may completely disagree. See the example below, where both extractors found Ellie Goulding (with different confidences), but the text extractor found “Joe Isaacson” while the schema.org extractor only found “Joe Montana”.


Text Extractor
Schema.org Extractor
Query
Yes
No
?
Joe Isaacson
Joe Isaacson is Data Scientist (%)
Ellie Goulding is Singer (%)
Joe Montana is Football Player (%)
Ellie Goulding is Singer (%)

When we query for “Joe Isaacson”, how do we know who to trust? One extractor did not find that the document related to “Joe Isaacson”, but one did. Sometimes extractors can give spurious results, so if we always just pick whichever extractor confirms what we’re looking for, we’ll get a lot of garbage results.

The Complete Brain: The Fusion Learner

Fortunately, we can solve this problem the way we solved all the other problems: by throwing more machine learning at it. We can actually build an algorithm that “learns” which extractor is good at which facts. The “text extractor” may be really great at extracting professions of people, but the “schema.org” extractor may be much better at finding out the number of rooms in a rental home. The Fusion Learner will combine the outputs of each extractors, along with their confidences, and build a final result that is more accurate than either extractor on its own. Again, see the *Why Freebase?* section for how freebase is used to train this learner.



<.....>...
...<...>
<...>...
Text Extractor
Schema.org Extractor
Joe Isaacson is Data Scientist (%)
Ellie Goulding is Singer (%)
Joe Montana is Football Player (%)
Ellie Goulding is Singer (%)
Fusion Learner
Joe Isaacson is Data Scientist (%)
Joe Montana is Football Player (%)
Ellie Goulding is Singer (%)

Why Freebase?

How do we build a text extractor that can accurately extract facts from documents in our index?

Consider the sentence *Deadmau5 is an electronic artist*. We can identify important entities in the sentence: **Deadmau5** and **electronic artist**. How do we understand how these entities are related? Ask Freebase!

"Freebase, what is the relation between **Deadmau5** and **electronic artist**?" Freebase responds, "***profession***"

Let's try again with another example.

Consider the sentence *Lady Gaga is a singer*. Again, we can identify important entities in this sentence: **Lady Gaga** and **singer**. What is the relation between these entities? Ask Freebase!

"Freebase, what is the relation between **Lady Gaga** and **singer**?" Freebase responds, "***profession***."

After these examples a text extractor could theorize that the following pattern is common in the english language:

> *[person] is a [profession]*

Inherently we believe this pattern to be true because we trust the accuracy of Freebase. We can now extend this pattern to entities and relations not listed in Freebase. For example, *Joe is a data scientist*. From this sentence, we can hypothesize that **Joe's** ***profession*** is **data scientist.**