

For example, here is a complete WebQL program to find adjacent words from Shakespeare that start and end with the same character trigram:

```
select item1, item2
from pattern '\W(\w+)\W(\w+)\W'
within crawl of http://www-tech.mit.edu/Shakespeare to depth 5
following if SOURCE_URL matching http://www-tech.mit.edu/Shakespeare
where extract_pattern(item1, '(\w\w\w)$') = extract_pattern(item2, '^(\w\w\w)')
```

This query quickly discover phrases such as “open penance”, “bitter terms” and “Notre trescher”.

As a second example, here is a [KnowItAll](#) like algorithm that bootstraps a list of scientists from a few seeds:

```
select as InitialScientists C1 as SCIENTIST
from table rows within 'file://scientists.csv'
join
select URL
from links
within http://www.google.com
submitting values ["scientists such as '||SCIENTIST||'"'] for 'q'
join
select SOURCE_CONTENT as DOC
join
select as MoreScientists item1 as SCIENTIST
from pattern 'scientists such as [a-z ]+, ([a-z]+( [a-z]+)?)' match case
within inline clean(DOC)
where item1 not matching ' and'
union join to InitialScientists, MoreScientists
select SCIENTIST
group by SCIENTIST
```

After initializing the algorithm with “biologists”, “chemists” and “oceanographers”, the algorithm generates the following output after two iterations:

	SCIENTIST
1	biologists
2	botanists
3	but the
4	chemists
5	climatologists
6	computer scientists
7	foresters
8	geographers
9	marine geologists
10	meteorologists
11	oceanographers
12	space scientists

Not so bad: 8 new scientists and only one mistake.