Finding similar items

CSE 344, section 10

June 2, 2011

In this section, we'll go through some examples of finding similar item sets. We'll directly compare all pairs of sets being considered using the Jaccard similarity. We'll also see small examples of minhashing and locality-sensitive hashing methods, which are intended to help make the similarity pairing tractable for many possible sets.

The examples we'll see in this assignment are taken from your textbook, specifically the exercises for Garcia-Molina section 22.3 (pages 1115-6) and section 22.4 (page 1122).

1 Jaccard similarity and minhashing

1. Compute the Jaccard similarity of each pair of the following sets: {1, 2, 3, 4, 5}, {1, 6, 7}, {2, 4, 6, 8} .

   Solution:
   Recall the general formula for the Jaccard similarity of two sets:

   $$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

   For the three combinations of pairs above, we have

   $$J(\{1,2,3,4,5\},\{1,6,7\}) = \frac{1}{7}$$
   $$J(\{1,2,3,4,5\},\{2,4,6,8\}) = \frac{2}{7}$$
   $$J(\{1,6,7\},\{2,4,6,8\}) = \frac{1}{6}$$

2. What are all the 4-grams of the following string?
   `abc␣def␣ghi`
   Remember that white space (denoted by ␣) counts!

   Solution:

   - `abc␣`
   - `bc␣d`
   - `c␣de`
   - `␣def`
   - `def␣`
   - `ef␣g`
   - `f␣gh`
   - `␣ghi`

3. Suppose that our universal item set is $\{1, 2, \ldots, 10\}$, and signatures for sets are constructed using the following list of permuations for the universal set:
   - $(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)$
   - $(10, 8, 6, 4, 2, 9, 7, 5, 3, 1)$
   - $(4, 7, 2, 9, 1, 5, 3, 10, 6, 8)$

   Construct minhash signatures for the following sets:
   (a) $\{3, 6, 9\}$
   (b) $\{2, 4, 6, 8\}$
   (c) $\{2, 3, 4\}$

   Solution:
   Each set's signature consists of one minhash value from each permutation of the universal set; this value is the first value in the permutation that appears in the subset. Hence our three signatures are:
   (a) $(3, 6, 9)$
   (b) $(2, 8, 4)$
   (c) $(2, 4, 4)$

4. Suppose that instead of using particular permutations to construct signatures for the three sets of the previous problem, we use hash functions to construct the signatures. The three hash functions we use are:

$$f(x) = x \mod 10$$
$$g(x) = (2x+1) \mod 10$$
$$h(x) = (3x+2) \mod 10$$

Compute the signatures for the three sets, and compare the resulting estimate of the Jaccard similarity of each pair with the true Jaccard similarity.

Solution:
Instead of finding the first value in the permutation that appears in the subset, we compute the minhash as the smallest value of the hash function in the whole subset. This yields the following three signatures:

(a) $(3,3,0)$
(b) $(2,3,0)$
(c) $(2,5,1)$

To estimate the Jaccard similarity from a minhash vector (derived either from permutations or hash functions), we find the number of matching minhash values in corresponding positions in the two subsets' minhash vectors. Estimating the Jaccard similarity using both permutation minhash vectors (problem 3) and hash function minhash vectors (this problem) gives:

Set 1	Set 2	Actual J	Hash est. J	Perm. est. J
(a) $\{3,6,9\}$	(b) $\{2,4,6,8\}$	$1/6$	$2/3$	$0/3$
(a) $\{3,6,9\}$	(c) $\{2,3,4\}$	$1/5$	$0/3$	$0/3$
(b) $\{2,4,6,8\}$	(c) $\{2,3,4\}$	$2/5$	$1/3$	$2/3$

We can see that these minhash vectors, whether computed from universal set permutations or a list of hash functions, are quite poor estimators of the Jaccard similarity. This is understandable given how short the minhash vectors actually are. We should get more permutations of the universal set or more hash functions to make the minhash vectors longer.

5. Suppose you have some documents, and have stored k-grams of these documents in a large table. Each column of the table represents all the k-grams for a single document, and each row r represents the r^{th} k-gram for all the documents. (Because documents vary in length, there may be empty cells in the bottom fringes of the table.) The "schema" of the table — that is, the mapping between row indexes in the table and document IDs — is stored separately.

   Show how you would use MapReduce to compute a minhash value for each of your documents, using a single hash function (*not* a permutation of a dictionary of possible k-grams). You can assume that every processor gets a copy of the schema, but:

   (a) The table must be partitioned across the processors by rows.

   Solution:

   In this partitioning, each processor receives a subset of the k-grams for every document. Hence, each processor can't compute the minhash for each document directly from its input data. Instead, it must compute the hash value for each of its k-grams separately. Then, the hash values must be grouped by document ID across the whole system. Finally, each processor finds the minimum hash value for each of the documents it is given.

   The MapReduce input is given as a set of key-value pairs, where each key is a document ID, and each value is a k-gram from that document. The map function then computes the hash value:

```
map (k: docID, v: kgram) {
  emit_intermediate(ik = k, iv = hash(v))
}
```

   The MapReduce system groups the hashes by document ID, then the reduce function finds the minimum hash value:

```
reduce (ik: docID, ivs: hashval[]) {
  var minhash := INFINITY
  for each iv in ivs {
    if iv < minhash {
      minhash := iv
    }
  }
  emit_final(fk = ik, fv = minhash)
}
```

(b) The table must be partitioned by columns.

Solution:

This split is easier to program, but also much harder to build, unless the table is stored in column-major order (which is common for data mining applications like this, but relatively rare otherwise). Here, all the k-grams for a document go to the same processor. We can actually use the same input format and the same map and reduce functions as before, but this doesn't take advantage of the fact that the input data is already grouped for us.

Instead, let the input keys be document IDs as before, but now let the input values be the list of every k-gram in the document. Then all the work is done in the map function:

```
map (k: docID, v: kgram[]) {
  var minhash := INFINITY
  for each kgram in v {
    var h := hash(kgram)
    if h < minhash {
      minhash := h
    }
  }
  emit_intermediate(ik = k, iv = minhash)
}
```

And the reduce function becomes a no-op:

```
reduce (ik: docID, ivs: hashval[]) {
  // There will be only one element in ivs[],
  // because only one map() produces each ik.
  emit_final(fk = ik, fv = ivs[0])
}
```

In fact, Hadoop will let you simply omit the Java code for the Reducer (reduce function closure) and not ask for it to be called at all. This saves the time to do the (useless) grouping of intermediate data.

2 Locality-sensitive hashing

1. Suppose we have a table where each tuple consists of three fields/attributes (name, address, phone number), and we need to do an entity resolution on this table to find those sets of tuples that refer to the same person. For concreteness, suppose that the only pairs of tuples that could possibly be total edit distance 5 or less from each other consist of a true copy of a tuple and another *corrupted* version of the tuple. In the corrupted version, each of the three fields is changed independently. 50% of the time, a field has no change. 20% of the time, there is a change resulting in edit distance 1 for that field. There is a 20% chance of edit distance 2 and 10% chance of edit distance 10. Suppose there are one million pairs of ths kind in the table.

   (a) How many of the million pairs are within total edit distance 5 of each other?
   Solution:
   Let's consider those pairs of tuples that are more than 5 away from each other. There are two possibilities that would cause this: all 3 fields have a change of edit distance 2 (probability: $(.2)^3 = .008 = .8\%$), and at least one field has a change of edit distance 10 (probability: $1 - (.9)^3 = .271 = 27.1\%$). The total proportion of tuple pairs that are 5 or less away from each other is then $1 - .279 = 72.1\%$, so 721,000 pairs are within edit distance 5.

   (b) If we hash each field of all the tuples to one million buckets, how many of these one million pairs will hash to the same bucket for at least one of the three hashings?
   Solution:
   By the definition of a hash function, identical field values will hash to the same bucket. Because each field pair is the same 50% of the time, there is a 50% chance that each pair of fields hashes to the same bucket, and a 50% chance that each pair hashes to different buckets.

   The probability that each pair of tuples hashes to the same bucket for at least one of the three fields, is just 1 less the probability that they hash to different buckets for all three fields: $1 - (.5)^3 = 1 - .125 = .875$, or 87.5%. Hence, there are 875,000 pairs of tuples that hash to the same bucket for at least one hashing.

(c) How many false negatives will there be? That is, how many of the one million pairs are within total edit distance 5, but will not hash to the same bucket for any of the three hashings?

Solution:

There are three cases of the differences between the fields that will cause a false negative:

- All three fields have a difference with edit distance 1. The probability of this case, over all possible choices for the chagne or lack thereof in all three fields, is $(.2)^3 = .008 = .8\%$.
- One field has edit distance 2, and the other two have edit distance 1. The probability is $3(.2 \cdot .2 \cdot 2) = .024 = 2.4\%$; we multiply by 3 because there are 3 possible choices for the field with edit distance 2.
- Two fields have edit distance 2, and the other one has edit distance 1. The probability is $3(.2 \cdot .2 \cdot 2) = .024 = 2.4\%$; we multiply by 3 (similar to above) because there are 3 possible choices for the pair of fields with edit distance 2.

The total probability of all three cases is $.056 = 5.6\%$, so we should expect 56,000 false negatives.

2. The function $p = 1 - (1 - s^r)^b$ gives the probability p that two minhash signatues that come from sets with Jaccard similarity s will hash to the same bucket at least once, if we use an LSH scheme with b bands of r rows each. For a given similarity threshold s, we want to choose b and r so that $p = 1/2$ at s. Suppose signatures have length 24, which means we can pick any integers b and r whose product is 24. That is, the choices for r are 1, 2, 3, 4, 6, 8, 12, or 24, and b must then be $24/r$.
   (a) If $s = 1/2$, determine the value of p for each choice of b and r. Which would you choose, if $1/2$ were the similarity threshold?
   Solution:

   - $r = 1, b = 24$: $p = 1 - (1 - (1/2)^1)^{24} = 1 - (1/2)^{24} \approx .99999994$
   - $r = 2, b = 12$: $p = 1 - (1 - (1/2)^2)^{12} = 1 - (3/4)^{12} \approx 0.968$
   - $r = 3, b = 8$: $p = 1 - (1 - (1/2)^3)^8 = 1 - (7/8)^8 \approx .657$
   - $r = 4, b = 6$: $p = 1 - (1 - (1/2)^4)^6 = 1 - (15/16)^6 \approx .321$
   - $r = 6, b = 4$: $p = 1 - (1 - (1/2)^6)^4 = 1 - (63/64)^4 \approx .0611$
   - $r = 8, b = 3$: $p = 1 - (1 - (1/2)^8)^3 = 1 - (255/256)^3 \approx .0117$
   - $r = 12, b = 2$: $p = 1 - (1 - (1/2)^{12})^2 = 1 - (2047/2048)^2 \approx 4.882 \cdot 10^{-4}$
   - $r = 24, b = 1$: $p = 1 - (1 - (1/2)^{24})^1 = 1 - (16777215/16777216)^1 \approx 5.960 \cdot 10^{-8}$

   It's clear that, in this case, you should choose to minimize the number of rows (positions within a signature), by letting $r = 1$.

(b) For each choice of b and r, determine the value of s that makes $p = 1/2$.
Solution:
We can solve for s in the formula for p:

$$\begin{aligned} p &= 1 - (1 - s^r)^b \\ 1 - p &= (1 - s^r)^b \\ \sqrt[b]{1-p} &= 1 - s^r \\ 1 - \sqrt[b]{1-p} &= s^r \\ s &= \sqrt[r]{1 - \sqrt[b]{1-p}} \end{aligned}$$

Then we can compute s by plugging in p and each choice of r and b we're interested in:

- $r = 1, b = 24$: $s = 1 - \sqrt[24]{1/2} \approx 0.0284$
- $r = 2, b = 12$: $s = \sqrt[2]{1 - \sqrt[12]{1/2}} \approx 0.2369$
- $r = 3, b = 8$: $s = \sqrt[3]{1 - \sqrt[8]{1/2}} \approx 0.4361$
- $r = 4, b = 6$: $s = \sqrt[4]{1 - \sqrt[6]{1/2}} \approx 0.5747$
- $r = 6, b = 4$: $s = \sqrt[6]{1 - \sqrt[4]{1/2}} \approx 0.7361$
- $r = 8, b = 3$: $s = \sqrt[8]{1 - \sqrt[3]{1/2}} \approx 0.8209$
- $r = 12, b = 2$: $s = \sqrt[12]{1 - \sqrt[2]{1/2}} \approx 0.9027$
- $r = 24, b = 1$: $s = \sqrt[24]{1/2} \approx 0.9715$