Data Mining: Learning from Large Data Sets - Spring Semester 2014
Project 1

nivo@student.ethz.ch
ganzm@student.ethz.ch
rohrp@student.ethz.ch

March 12, 2014

Approximate near-duplicate search using Locality Sensitive Hashing

This document explains the implementation of project 1. In this first project we were supposed to find near-duplicate videos using Locality Sensitive Hashing. The videos were given as a number of shingles that could be compared. The implementation was done in Python to be used in combination with the hadoop infrastructure. This means we implemented a mapper and a reducer that read from stdin and write to stdout. In the following two sections we explain the functionality of the mapper and the reducer each.

mapper.py

The mapper creates a signature vector of a video. This signature vector is afterwards subdivided into b bands having r elements that get hashed. Each band and its hash are then emitted.

These are the steps in more detail:

1. As Figure 1 shows, choosing $b = 32$ and $r = 8$ forgets no true positives.
2. We create $k = b \cdot r$ hashfunctions of the form $a_p x + b_p \mod c_p$ where $a_p = rand(0, 999), b_p = rand(0, 99999)$, $c_p = 10000$ and $rand(x, y)$ is random variable drawn from a uniform distribution between x and y. These hashfunctions are used to calculate the permutations in min-hashing.
3. We create r hashfunctions of the form $a_b x + b_b \mod c_b$ where $a_b = rand(0, 999), b_b = rand(0, 999)$, $c_b = 1000$ and $rand(x, y)$ is random variable drawn from a uniform distribution between x and y. These hashfunctions are used to calculate the band hashes before emitting.
4. The signature is then created according to 1
5. Calculate the band hash and emit the band hash and the band as key, and the video with its corresponding shingles as value according to 2

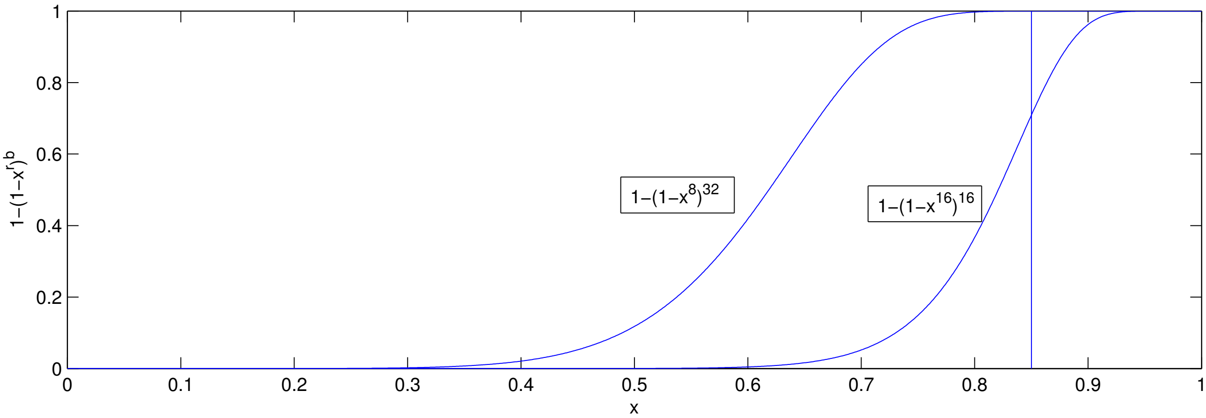

Figure 1: $1-(1-x^r)^b$

Algorithm 1 Create signature

$signature = \infty$
for all shingle in shingles **do**
  for i in range(k) **do**
    $signature[i] = min(a_{p_i} \cdot shingle + b_{p_i} \mod c_p, signature[i])$
  end for
end for

Algorithm 2 Emit keys and values

---

for band in range(b) **do**
  $vector = signature[band \cdot r : band \cdot r + r]$
  $bandhash = \sum_{i=1}^{len(vector)} a_{bi} \cdot vector[i] + b_{bi} \mod c$
  emit $key = [bandhash, band]$ $value = [video\_id, shingles]$
end for

---

reducer.py

The main task of the reducer is to get rid of the false positives by comparing the reported similar videos using the jaccard distance:

1. gather all videos with the same key in a collection $duplicates$
2. emit similar videos like shown in 3

Algorithm 3 Emit similar videos

---

for i=0 to len(duplicates) **do**
  for j=i+1 to len(duplicates) **do**
    if $duplicates[i].video\_id < duplicates[j].video\_id$ **then**
      $shingles\_left = duplicates[i].shingles$
      $shingles\_right = duplicates[j].shingles$
      $distance = \frac{|shingles\_left \cap shingles\_right|}{|shingles\_left \cup shingles\_right|}$
      if $distance > 0.85$ **then**
        emit duplicates[i].video_id duplicates[j].video_id
      end if
    end if
  end for
end for
