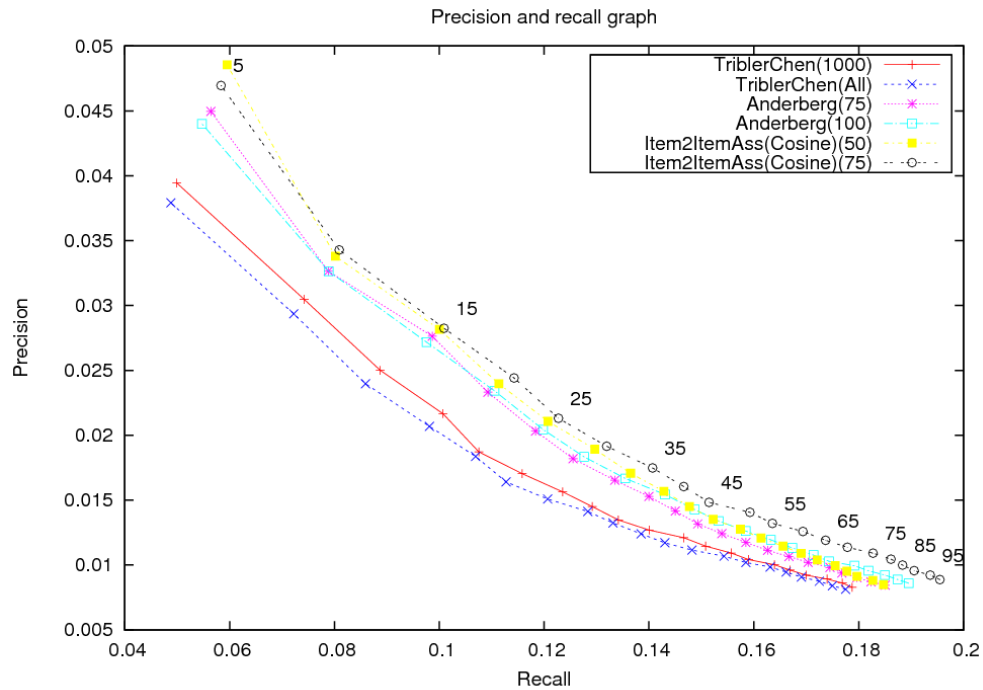
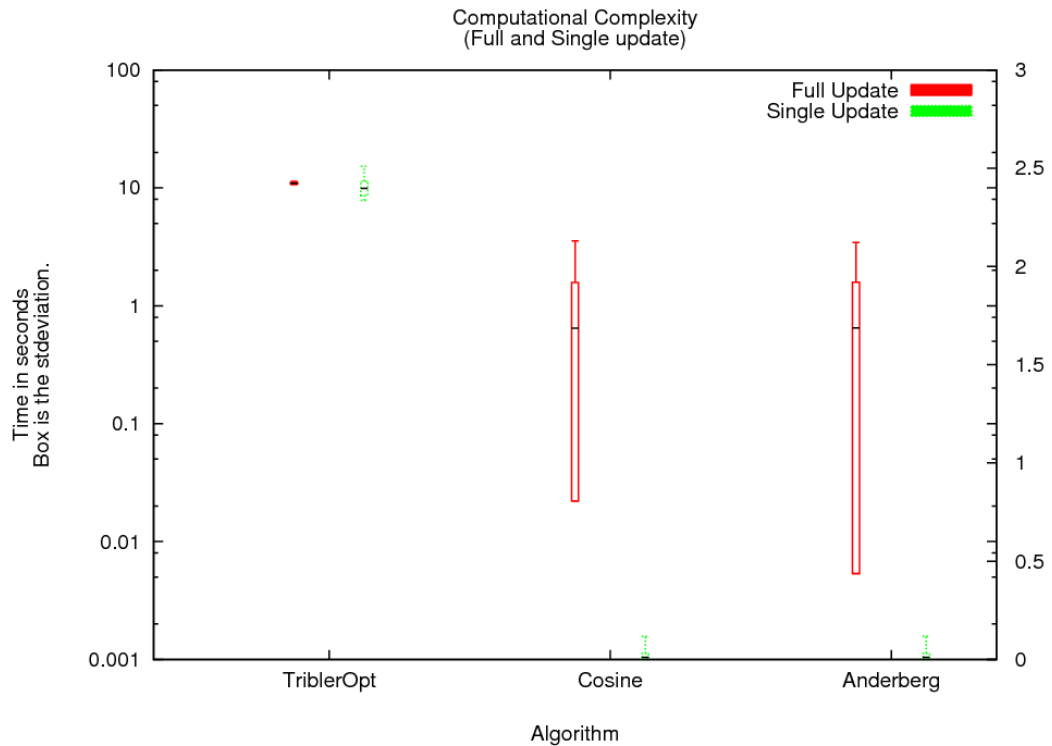


Performance

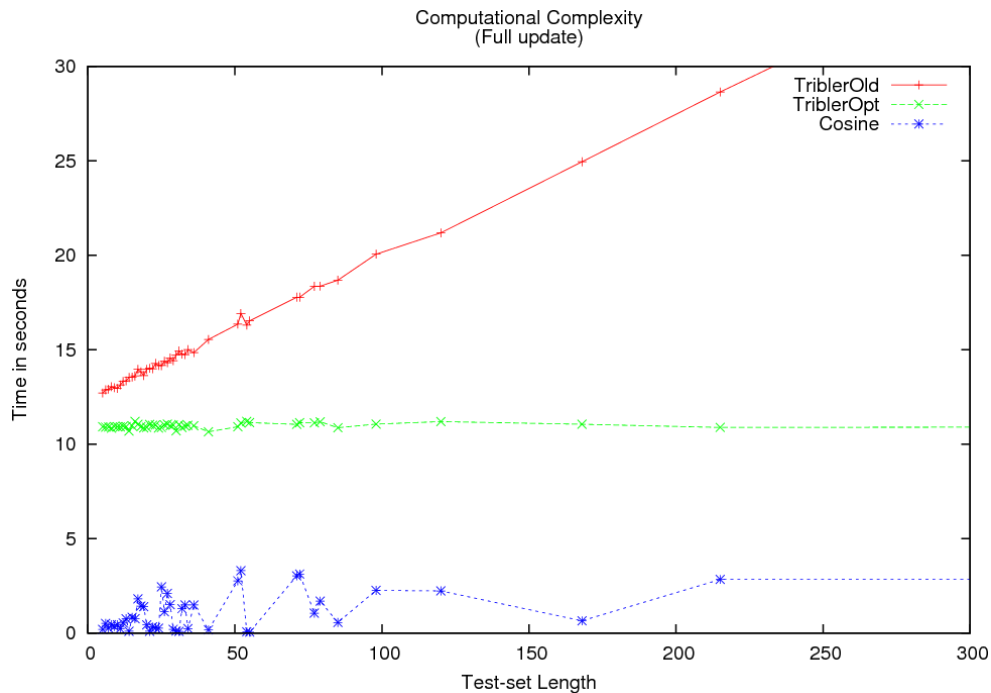
Overall Precision and Recall higher, for both Cosine and Anderberg (Anderberg better then Cosine). Old graph included for 2500 random chosen users.



No caching + inserting results in database (50.000 users, 343.541 items, 1.173.072 user/item relations, superpeer-dataset but only items used for which a .torrent file is present)



Why the difference? No actual values needed by Cosine and Anderberg, only number of overlapping items and total number of items. When no overlap between 2 users, the result of Cosine and Anderberg == 0. Tribler code always returns some value, resulting in a large number of inserts. Additional function needed to deal with empty profiles, possibly select users with largest profile.



No linear increase in computation time (larger profile -> higher computation time), which is the case in current Tribler code.

Cosine

```
train_set = self.datasources.get_items_list(self.test_peer)
test_root = 1.0/(len(train_set) ** .5)

for other_peer, nr_items, overlap in
self.datasources.get_peer_nritems_nroverlap(self.test_peer, train_set):

    similarity[other_peer] = overlap * (test_root * (1.0/(nr_items ** .5)))
```

Anderberg

```
train_set = self.datasources.get_items_list(self.test_peer)
train_len = len(train_set)

for other_peer, nr_items, overlap in
self.datasources.get_peer_nritems_nroverlap(self.test_peer, train_set):

    M11 = overlap

    M01 = nr_items - M11

    M10 = train_len - M11

    similarity[other_peer] = float(M11) / (M01 + 2*(M10 + M11))
```

Select SQL

```
self.cursor.execute('SELECT Peer.peer_id, num_prefs, COUNT(torrent_id) FROM Peer JOIN  
Preference ON Peer.peer_id = Preference.peer_id WHERE torrent_id  
IN('+','.join(map(str,tids))+') AND Peer.peer_id <> ? GROUP BY Peer.peer_id', (not_peer,))
```

Update SQL

```
self.cursor.executemany('UPDATE PEER SET similarity = ? WHERE peer_id = ?', similist)  
self.connection.commit()
```

Changes needed

Strange caching strategies by J. could/should be removed.

If removed, then the new sql statements could be used. If not a drop-in replacement which does not need the owners dictionary.