Blocking on Tables

Q1) How did you develop the final blocker? What blocker did you start with? What problems did you see? Then how did you revise it to come up with the next blocker? In short, explain the *development process*, from the first blocker all the way to the final blocker (that you submit in the IPython file).

We took around 7k and 6k tuples from the IMDB and Rotten Tomatoes respectively. So, the possible tuple matching exceeded 47 million. In order to reduce this number, we used year or release date year attribute for blocking. In case of year one thing we noticed is that the two sources sometimes marked the movies as released in two consecutive years. This is usually because of the movie being released in year end. So we applied a Overlap Blocker between the Year field from one source with the YearRange column of the other source. YearRange is the modified field generated using the Year information, for example “2010 2011 2012” for the year “2011”. It reduced the tuple pairs to around 3.1 million. We weren't able to use the debugger here so we did some manual inspection on the movie data in the excel sheet where we had sorted the tuples on the basis of release year. Movie release year seemed to be a good attribute for doing attribute equivalence based blocker. This also does a good job of separating out movies with multiple sequels like blocking out Godfather II and Godfather III which is very useful for later stages of entity matching.
We then used the overlap blocker over the director with overlap size of 1. We found that we had true positives but the directors with common first name were also potential matches like:

<table>
<thead>
<tr>
<th>A.Name</th>
<th>A.Director</th>
<th>A.Year</th>
<th>B.Name</th>
<th>B.Director</th>
<th>B.Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>The Alamo</td>
<td>John Wayne</td>
<td>1960</td>
<td>The Magnificent Seven</td>
<td>John Sturges</td>
<td>1960</td>
</tr>
<tr>
<td>The Alamo</td>
<td>John Wayne</td>
<td>1960</td>
<td>The Alamo</td>
<td>John Wayne</td>
<td>1960</td>
</tr>
<tr>
<td>The Alamo</td>
<td>John Wayne</td>
<td>1960</td>
<td>The Unforgiven</td>
<td>John Huston</td>
<td>1960</td>
</tr>
</tbody>
</table>

So, we made the overlap size to be 2 with the overlap blocker on the directors. It reduced the tuple pairs to 4.7 thousand rows. This is very tight candidate set of tuples. Also, we found the tuple pairs where the same movie director had directed different movies in the same year. One such candidate tuple example is:

<table>
<thead>
<tr>
<th>A.Name</th>
<th>A.Director</th>
<th>A.Year</th>
<th>B.Name</th>
<th>B.Director</th>
<th>B.Year</th>
</tr>
</thead>
<tbody>
<tr>
<td>Devil’s Doorway</td>
<td>Anthony</td>
<td>1950</td>
<td>Side Street</td>
<td>Anthony</td>
<td>1950</td>
</tr>
<tr>
<td></td>
<td>Mann</td>
<td></td>
<td></td>
<td>Mann</td>
<td></td>
</tr>
</tbody>
</table>

So, we tried blocking using Jaccard similarity measure on 3-grams on the Director field. Threshold of 0.2 seemed decent resulting in 18125 candidate matches. Next we tried Jaccard similarity measure on 3-grams on the Name field. After a bit of tuning we found
out that for a threshold of 0.1 we were able to do a decent job of blocking resulting in 64062 candidate tuples. We took small threshold of 0.1 because we came across cases where the use of stopwords etc. were making the Jaccard score on name matches very low.

To further do the effective blocking, we tried blocking on Cast field but from the data it was clear that both the sites posted only a subset of the cast which shared very few or none of the cast. So we dropped this idea.

We decided to combine the two candidate sets generated by Jaccard similarity measure on Name and Director field. For the matching tuples, score of Jaccard similarity measure being low for both name and director field was very low. Hence, we took the union of the two candidate sets of tuples.

The final blocking mechanism is as shown in the figure:

Q2) Did you use the debugger? If so, where in the process? And what did you find? Was it useful, in what way?

Yes, we did use the debugger. We used the debugger to see if we were missing true positives in the candidate set of tuples. We found out that some our tuples were being
missed out by the Jaccard similarity on Director. So, we used Jaccard similarity on Name as well and combined the both set of candidate tuples. Debugger was really useful in determining whether the promising true positive are passed by the blocker or not.

Q3) How much time did it take for you to do the whole blocking process?
It took us around 2 days to do the blocking process. Debugger was the bottle neck in the process. It took most of the time to run (several hours) and produce results.

Q4) Report the size of table A, the size of table B, the total number of tuple pairs in the Cartesian product of A and B, and the total number of tuple pairs in the table C.
The table A contains 7391 tuples with 17 attributes. Table B contains 6408 tuples with 13 attributes. The cartesian product of tables A and B had 47.36 million tuples pairs. The total number of tuple pairs in final Candidate set C were 78079.

Q5) Did you have to do any cleaning or additional information extraction on tables A and B?
We cleaned the release date information for the tuples. We used regular expression to extract only the year of release date.

Q6) Did you run into any issues using Magellan (such as scalability?). Provide feedback on Magellan. Is there anything you want to see in Magellan (and is not there)?
Magellan ran into problem when the attribute name in the csv file contained spaces. For example, we got error while blocking because of the field “Release Date”. When we fixed it to “ReleaseDate”, things worked smoothly.

We also noticed that when we submit request to execute heavy blocker/debugger, it took a while before communicating anything to us. This was confusing because we initially mistook that there was some issue with Magellan or its setup.