CS 784
Kaashyapee Jha and Margaret Pearce
Project: Blocking on tables A and B

1. **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).**

Throughout the development process, we followed the suggested methodology in Section 9 of the Magellan user manual. The first blocker that we started with was the built-in overlap blocker matching first authors on word-level with an overlap size of 1 (C). This was a good starting point because it ran quickly (13.67 seconds, 103441 matches). However, the blocker was too generous on what it considered a match. If two authors in two tuples shared the same first name, then the tuples would be considered a match even if their last names were clearly different.

Next, we decided to try a rule-based blocker. To do this, we needed to obtain a set of features available for blocking. Unfortunately, the available set was smaller than we anticipated and we could not use several tuple attributes for blocking (e.g. authors). The attributes we could choose from included Title, ISBN13, and Publisher. We attempted several combinations of these attributes to create multiple rule-based blockers, including:

- D: Matching on ISBN13 using edit distance (lev) - 1517 matches
- E: Matching on ISBN13 and Title, both using edit distance - 513 matches
- F: Matching on Title using edit distance - 9451 matches
- G: Matching on Title using TF/IDF (cos) - 7015 matches
- H: Matching on Title using TF/IDF and ISBN13 using edit distance - 463 matches
- I: Matching on Title and ISBN13 using edit distance with lower threshold - 593 matches
- M: Match on ISBN (edit distance, 0.55 threshold), title (edit distance, 0.65 threshold), publisher (Jaccard, 0.65 threshold) - 346 matches
- N: Match on Title, Publisher with edit distance and 0.65 threshold - 1361 matches
- H2: Matching on Title using TF/IDF, and on ISBN13 and Publisher using edit distance - 268 matches

Matching on ISBN13 alone was not sufficient. There were many examples where the ISBN would vary by one or two digits, but the two ISBNs were distinct. By looking at the titles of the matched pairs, it was easy to see that the blocker produced many incorrect matches, such as matching “Autobiography of St. Gemma Galgani” and “Recovering Literature's Lost Ground: Essays in…”. However, using title alone resulted in even more incorrect matches.

Many of the rule-based blockers that used edit distance had trouble with titles such as “{name}: An Autobiography”. For example, the book “Hayden: An Autobiography” in table A had 6 matches in table B with only one correct match out of the six. We hoped to use TF/IDF to emphasize the

importance of “{name}” and put less weight on “An Autobiography”, which led us to try matching title on cos. This resolved the issue of titles matching on “{name}: An Autobiography”, but it did not eliminate matches of “{name}: An Autobiography” and “An Autobiography”. We discovered that many books share the exact title “An Autobiography”, and these books all were matched to books with titles of the form “{name}: An Autobiography”.

To resolve this issue, we added a rule for ISBN13 using edit distance to help distinguish the matches. Note that this is very similar to the rule matching Title and ISBN13 both on edit distance. In the case where we used cos, we got 463 matches compared to 513 matches using edit distance. The version using edit distance was much faster than TF/IDF and appeared to have higher recall than the blocker using cos, so we stuck with using edit distance on title and ISBN13 to further develop the blocker.

Matching Title with TF/IDF and ISBN13/Publisher with edit distance was too aggressive which resulted in the least number of matches out of all the blockers. This was mainly due to the fact that the number of attributes being matched was higher than the other blockers.

We also tried more built in blockers, including:

- J: Overlap, match first author using q-grams of size 4 with overlap size 4 - 45428 matches
- K: Attribute equivalence, match on ISBN13 - 1239 matches
- L: Overlap, match ttle with q-grams of size 4 with overlap size 16 - 1,034,153 matches
- O: Overlap, match first author at word level with overlap size 2 - 3941 matches

In the end, we took the union of blockers O, I, and H. These three seemed to balance out the shortcomings of each approach and resulted in a reasonably sized set of matches (4029).

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

When we tried several blockers and chose the best option we had seen so far, we used the debugger to see if there were any pairs that should have been matched that were being missed by the best blocker. We found at least 100 pairs that should be matched but weren’t counted as matches by the blocker. This led us to tweak the rules of the blocker so we could obtain better recall.

3. **How much time did it take for you to do the whole blocking process?**

Using the Magellan API to creating blocking rules took surprisingly little time. Some of the blockers took a long time to evaluate all tuple pairs (approximately one hour to use TF/IDF in a rule-based blocker), but the built-in blockers were very efficient (< 15 seconds to analyze all |A x B| tuple pairs).

4. **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.**

Table A contains 3967 tuples. Table B contains 3701 tuples. The Cartesian product A x B therefore contains 14,681,867 tuple pairs. Our final blocker used to generate table C resulted in 4029 tuple pairs.

5. **Did you have to do any cleaning or additional information extraction on tables A and B?**

First, we needed to extract an additional attribute from the pages from Goodreads (source A). Specifically, we needed to get ISBN13. We originally extracted a 10-digit ISBN from Goodreads (source A) and a 13-digit ISBN from Barnes and Noble (source B). The 10 digit ISBN was partially contained in the 13-digit ISBN, but we noticed that the last digit would often not match. For simplicity, we chose to extract the 13-digit ISBN as a new attribute in table A so that we could make better matching decisions.

Additionally, we needed to make a minor formatting change on the headers for tables A and B. Our table headers included spaces for some attribute headings, such as “FirstAuthor”. This caused some errors within Magellan, specifically when we tried to get the features available for blocking.

Finally, we noticed an exception that occurred for one tuple in table B that resulted in the Title attribute being spread over two attributes in the .csv. This caused Magellan to return an error when reading table B. We updated the wrapper to handle this case and regenerated table B, which resolved the issue.

6. **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)? Any other feedback is appreciated.**

Getting Magellan to work on a x64 64-bit Windows machine was not straightforward. Code had to be added to the python code and before installing Magellan, the C extensions had to be compiled.

As mentioned in Question 5, Magellan ran into problems when the .csv files used to generate tables had spaces within header (e.g. “First Author” vs. “First_Author”).

In some cases, we noticed lag between the time when we would hit play to execute a line of the code and when we would see a response. When this happened, we would restart the kernel and re-execute each line or re-open the notebook.

It would be nice if there were an option where users should specify if blockers should treat attributes as case insensitive. We found some tuples where the title would be in all caps, which prevented it from being matched to the same title in sentence case (e.g. “AN AUTOBIOGRAPHY” and “An Autobiography”). It would also be nice to dynamically match the schema between tables A and B. For example, we had “FirstAuthor” and “Author1”, which seemed to prevent these two attributes from being in the features table for rule-based blockers.