1. Introduction

This appendix details the intuitive logic and procedural steps of System A and B supplementing the discussions in the main thesis and the actual R programs for both systems captured in Appendix C.1.

As a preamble, it is worth noting some of the details of the programs that make up System A and B. Firstly, note that in Appendix C.1, System A is captured in the “uk\_ir\_website\_scrapper.R” program, which contains code for both:

1. The HTML web scrapping program described in S.3.2.1 of the main thesis
2. System A

The two programs were combined so that they can be used as a singular prototype tool for actual RT.1 information extraction. See “Instructions to Appendices” in the main thesis for more detail.

Secondly, note that System B is contained in Appendix C.1 as two programs:

1. “judge\_extractor.R” for MD.6, and
2. “member\_extractor.R” for MD.7

Again, this was done to make it easier for non-technical legal researchers to use these programs as prototype tools. If system B is used in practice it requires “uk\_ir\_website\_scrapper.R” and “uk\_pdf\_downloader\_textconverter.R” to be run first (both included in Appendix C.1). See “Instructions to the Appendices Folder” in the main thesis for more detail.

The rest of this appendix will keep to the intuitive logic and steps for System A and B abstracting away from the details contained in the programming scripts described above (further detail about System A and B’s actual code is included in those scripts).

Note also that MD.1 – 7 described in this appendix are as per Table 1 in the main thesis.

2. System A

2.1 Basic Procedure

System A is a program devised to extract MD.1 to MD.5 values from the HTML data scraped from the official UK Freedom of Information (UK FOI) tribunal website collected as per S.3.2.1 of the main thesis. It relies on rules, regular expressions and a dictionary (optional) to perform its tasks. We devised System A by reviewing the first and last pages of the UK FOI tribunal website and noting the consistent patterns described in **Fig. 1**. At this stage, the web scrapping program from S.3.2.1 of the main thesis (which is the first part of “uk\_ir\_website\_scrapper.R” program) would have already extracted the HTML table (as seen in **Fig.1**) that contains MD.1 to 5.

How we extract MD.1 and 2

From **Fig.**1, we can see that MD.1 and 2 values always appear in their own unique tags in the HTML making them easy to extract using Cascading Style Selector (CSS) rules[[1]](#footnote-1) based on their position for each row of the table.

How we extract MD.3 - 5

Next, from **Fig.1**, we can also see that MD.3 - 5 values are always contained in the “Case Title and Reference” HTML tag for each row as a text string. Using CSS rules, for each report, we can identify the “Case Title and Reference” HTML tag, then for the text string contained within the tag, MD.3 – 5 can always be extracted via a consistent pattern within each text string which allows MD.3 – 5 to be separated using simple rules and regular expressions. Working from the back to the front of each row’s “Case Title and Reference” text string, we devised code to:

1. First, extract the MD.3 value using regular expressions[[2]](#footnote-2) (as the style of writing MD.3 values changes over the years).
2. Second, extract MD.5 second respondent’s name, if they exist[[3]](#footnote-3), by using the “Additional Party” phrase as a consistently occurring separator (the phrase always occurs before the second respondent’s name).
3. Third, extract the first MD.5 respondent’s name using “v” as a consistently occurring separator.
4. Finally, the remaining text string will always be the MD.4 name and is simply returned as the MD.4 value for that report.

At the end of System A’s program, all unnecessary white space is removed and a table containing each report’s MD.1 to 5 values is returned which are tied to each report based on their ordered position on the website. Note that System A relies on the HTML scraping program (which forms the first part of the “uk\_ir\_website\_scrapper.R” program) from S.3.2.1 of the main thesis which returns the entire website’s report database and HTML code. Future versions of the HTML scraping program could be devised to update only reports and HTML code not already in the experimental dataset based on the latest running number at the bottom of the first page of the website (See **Fig.1**) which can be extracted for use in the program using CSS selectors. For example, UK FOI Tribunal dataset has 2606 reports, current System A dataset total = 2602 reports, (2606 - 2602 = 4, update System A dataset with only first four values from website).

Graphical user interface, table

Description automatically generated

**Fig. 1.** Source: [1]. Extract of the UK FOI Tribunal website’s main HTML table highlighting RT.1 metadata by ID. MD.1 and 2 are contained in their own HTML tags. MD.3 – 5 are contained in the “Case Title and Reference” HTML tag and were separated using regular expressions.

2.2 MD.1 Consolidation

Next, depending on what the researcher needs to do, they may need to consolidate MD.1 values. The challenge here is that in the early part of this database’s life, appeal outcomes were clearly hand-typed resulting in numerous versions (and misspellings) of the same outcome value. To account for this issue, based on the values seen in the training set, we set up a dictionary (i.e. associative array, see **Table 1**) and manually re-coded all variants and misspellings into a single value to give researchers an option to use this dictionary to consolidate MD.1 values if they want to perform aggregate analysis based on the case outcome. The risk of misclassification is negligible as new values will always be unique to those already in the dictionary. Note that this consolidation process must be done manually but future versions of System A could automate this process.

Note too that RT.3 Cost-Benefit Analyses assume that this consolidation of MD.1 values is not performed for the CBA.1 task. Regardless, the manual effort to consolidate MD.1 values takes a matter of minutes and is negligible.

**Table 1.** Dictionary devised to consolidateMD.1 values. The consolidated case outcome value in the left column is the singular value to consolidate all other values that have the same meaning as it (in the right column) in the UK FOI tribunal based on the training set. The ‘Examples of Possible Values’ represent the most common values for MD.1 outcome variants found in the training set.

Table

Description automatically generated

2.3 Risk Mitigation

For System A, the main assumption is that the inherent patterns of the website (that the program needs for its rules, regular expressions and optional dictionary) will not change. For a government website, this assumption should hold true for the majority of the website’s existence and the risk can be fully mitigated by the researcher understanding how each key technique in the program works then reviewing the website structure for any changes from the current required patterns (especially those noted in S.2.1 and S.2.2 above) before running the program.

2.4 Workflow

To summarise, the workflow would be:

1. Check UK FOI Tribunal website for any structural changes to the HTML format that could affect System A.
2. If no structural changes, then input table containing HTML code for each report using the web scraping program described in S.3.2.1 of the main thesis (i.e. the first part of the “uk\_ir\_website\_scrapper.R” program).
3. Run System A (i.e. the second part of the “uk\_ir\_website\_scrapper.R” program).
4. System A returns a table where each row corresponds to each decision report’s MD.1 to 5 values in the same order as the table inputted.
5. Consolidate MD.1 values manually using dictionary in **Table 1** if required.

3. System B

3.1 Basic Procedure

System B is a program to extract MD.6 and 7 names. The program code with detailed step-by-step descriptions are included in **Appendix C.1** as “judge\_extractor.R” and “member\_extractor.R” (please see ‘Instructions to the ‘Appendices’ Folder’ for details).

As MD.6 and 7 values are not included in the website HTML code, we had to rely on patterns contained in the text of the decision reports. To maintain independence from the test set, the heuristics were developed only from the training set. First, we noted that there were at least 34 judges, who are the primary writers of these reports and each judge has a distinct manner of writing that changes over time. This meant that regular expression based rules would become very complex and require changing over time and statistical approaches would require a lot of manual coding and experimentation [2] [3]. Therefore, we investigated a dictionary-based approach [4] as a potential solution which provided the strong results as per S.4.1 of the main thesis. The following paragraphs describe the rationale for how the dictionary-based approach was devised.

From the 300 training documents, we found that MD.7 would always appear near the start of a report as per **Fig. 2** and MD.6 would always appear at the start of a report (**Fig. 2**) or if not then at the end of a report (**Fig. 3**). We also knew that within the training documents there were at least 34 judges and 30 lay members in the system across the lifetime of this tribunal due to the stratification by year. Furthermore, we knew that each MD.6 and 7 entity could have one to four variants on their names (**Fig.4**) as found in the training set (but there could potentially be more variants). These four variants are:

1. Their full name (without their middle name).
2. An informal version of the first name (e.g. Bernie instead of Bernard) and their surname.
3. An initial then surname (D. Williams or D Williams)
4. Or for judges specifically, just their surname (no lay members had their names written by just their surnames).

Next, we observed within the training set that the same few names tended to appear repeatedly over specific ranges of time. Based on this, we made the assumption that the judge and lay member staff in the UK FOI Tribunal do not change very often and there are not that many judges and lay members in the system (this turned out to be a fair assumption based on the results in S.4.1 of the main thesis). Based on this assumption, the training set would then consist of a good cross-section of the unique individuals in this system since its inception, given it was prepared with observations that were selected at random and stratified by year.

Based on the observations made in the above two paragraphs that there were probably only about four variants per unique entity for MD.6 and 7 and there were probably less than 40 unique judges and 40 unique lay members who have served in the lifetime of the entire UK FOI tribunal, a dictionary lookup-based information extraction system was deemed to be adequate for our purposes without requiring the additional effort needed for more sophisticated information extraction methods [5] and being more robust than regular expression based information extraction systems. To build these dictionaries, each judge and lay member was given a unique key and tied to their name variants as were seen in the 300 reports (similar to **Fig.4**) with judges and lay members having separate dictionaries (The actual judge and lay member dictionaries are included in **Appendix C.1** as “judge\_dict.csv” and “lay\_dict.csv”.).

As for risk mitigation, fortunately, no two individuals in the system had the same or similar names but such an issue should be kept in mind if System B is used in the real world. In the future, if similar/ same names are found within or between the two main dictionaries, mitigation would require manual intervention to assign a unique key to each similar name and manual checks when an exact duplicate is found.

With the dictionaries resolved, we used the facts that the MD.6 and 7 entity names always appeared near the start and end of each document (**Fig. 2 & 3**) and setup System B such that it would only review the first and last words in each report and then execute a string-searching rule to extract the first instance of any variant of each Judge’s name (as there can only be one judge per case) and the first two instances of any variant of each lay member’s name based on the pair of dictionaries (as there can only be zero or two lay members per case as per UK FOI tribunal procedure) then return the unique identifier name for each variant (**see Fig.4**) as a result where the first judge name found is returned as the judge for that case and the first two lay member names found (if they were present) are returned as the lay members for that case (as there will be two lay members in all instances if they are part of a case). We tuned the system’s and values to the training data to achieve 100% precision and recall and those and values were used in the testing data and would be the default value for the practitioner prototype tools that together represent System B, “judge\_extractor.R” and “member\_extractor.R”. Overfitting was deemed unlikely given a Judge’s name only ever ends up earlier than the last words in a report in the very rare case where a decision has a considerable appendix and can be dealt with manually with little effect to the costs of the system given the rarity of such events.

So, the entire workflow so far would be:

1. Manually prepare the judge and lay member dictionaries.
2. Tune System B for the and pages described above.
3. Feed table containing each report’s full text string from the PDF extraction process in S.3.2.1 of the main thesis into System B’s two programs “judge\_extractor.R” and “member\_extractor.R”.
4. System B searches each report’s text string for the first instance of any judge name or variant (given there can only be one judge per case) and the first two instances of any lay member name or variant if possible (given there will be only zero or two lay members per case).
5. System B outputs a table with containing the unique name of the judge and lay member unique names (if they were present) for each report as per the input table.
6. Manually deal with exceptions.

**Text

Description automatically generated**

**Fig. 2.** Extract from the first page (first words) of [6] decision report where the names of MD.7 lay members, MD.4 appellant(s) and MD.5 respondent(s) always appear if they were present and the MD.6 judge name appears most of the time.

Text, letter

Description automatically generated

**Fig. 3.** Extract from the last page (last words) of [7] decision report where the name of the MD.6 presiding judge tends to appear if not stated at the start of the report.

Table

Description automatically generated

**Fig. 4**. Examples of unique Judge & lay member entity names and all known variants.

3.2 Risk Mitigation

With regards to the above workflow’s Step 6 exception handling, the first problem would be if the name extracted was not the MD.6 or 7 entity in question. The only other names that would appear in the first part of the report (if the pages are tuned well) would be the MD.4 appellant name(s) and MD.5 respondent name(s) as per **Fig.2**, this risk is mitigated by requiring that System A always be run before System B as System A’s output for appellant and respondent names should always be correct if its risk mitigation step is taken. After System A is run and the appellant and respondent names are available, the user must manually check if any report appellant name and respondent name have an exact match with the name variants in System B’s dictionaries then update the relevant dictionaries as necessary (which can be easily performed in Microsoft Excel or any similar spreadsheet software – including open source). The next version of System A and B could have a feature to perform and flag out these same name issues automatically. Note that because this issue is hypothetical (given no entities in the UK FOI tribunal have the same name), this issue is also not included in the CBA.1 cost-benefit analysis calculation and would be a minor cost even if it was included.

The next issue is when the returned judge’s name is just their surname, to date only seven judges have signed off with just their surname but they are fairly common names so any returned single named judge values should be manually reviewed against their original report to be sure the correct person has been identified as the judge which will only occur for that small subset of judges. For the lay members, based on the training set, there were no single named lay members and thus the system would return such values as blank. However, in the future if such values do occur, they can be manually managed in the same way as the judges.

The last possible scenario is if the System returns a blank for either MD.6 or 7 entities for a report. All blanks need to be manually reviewed as per the numbered list below with some instances requiring the two system dictionaries to be updated and the system to be re-run:

1. In the positive case, these values are blank because the report does not state any relevant MD.6 or 7 names, but the relevant decision report will still need to be manually reviewed to confirm these blank values to be true.
2. If the returned values are erroneously blank because the page values for and are tuned poorly then they may have to be re-tuned in light of the new information or the document marked as an exception to be manually coded due to being an outlier (e.g. extremely long appendices which hardly happens based on the coding of the RT.1 training set).
3. If the judge’s name is wrongly blank then it is likely a new judge not in the dictionary or a new spelling variant for a current judge, either will require human intervention to review the relevant decision report then update the judge dictionary for the new name or new variant of an existing name.
4. If one of the two lay members or both of the lay member name entries are erroneously empty then it is either a new lay member, a new spelling of an existing lay member or a misspelling of a name by the judge who wrote the report (the equivalent misspelling of a judge’s own name is highly unlikely given that judge will be writing his own name in the report but will be spotted regardless as long as Step 3 above is taken). Each of the possible error scenarios described here will require manual review of the relevant tribunal decision report and updating of the lay member dictionary for the new name or new variant of an existing name.

Note that despite the number of these exception scenarios, resolving for them is typically faster than linear time as each added name will likely resolve for multiple reports given the small number of entities in this system. To provide some empirical evidence for this and set a relevant baseline estimate for the cost-benefit analysis (see **Appendix A.3b tab 1 item d and e**), we used System B on 500 unused reports from the raw dataset described in S.3.2.1 of the main thesis, which resulted in 80 blanks for MD.6 and 7 entities. We then performed a simple manual algorithm (Algorithm A below) to resolve all 80 blanks using only five iterations through the algorithm which took less than 20 minutes (much faster than performing MD.6 and 7 coding manually for every report).

**Algorithm A**.

1. From the blank returned values, pick five blanks at random.
2. Read the five reports which had blank returned values for their MD.6 and 7 entities:
   1. If there are no MD.6 and 7 entities, this means that System B correctly identified those entries as blank, note them as correctly blank in the original returned values. Go back to start of Algorithm A if there are still unknown blanks outstanding.
   2. Else, if there are any MD.6 or 7 entities, update both dictionaries as necessary.
3. Re-run System B again with the updated dictionaries and note any remaining blanks unresolved by performing Step 2b. Repeat algorithm A until all unknown blanks are resolved.

4. System A & B Workflow

To summarise, the overall workflow for System A and B would be:

* **Step 1:** Check the UK FOI Tribunal website for pattern changes to ensure System A will work.
* **Step 2:** Input prepared UK FOI tribunal reports with unique ID, text content as text strings and associated HTML code[[4]](#footnote-4).
* **Step 3:** Run System A, takes HTML code for each report and returns MD.1 to 5 values with exceptions noted.
* **Step 4:** Run System B, takes tribunal text content and returns MD.6 and 7 values with exceptions noted.
* **Step 5:** Human intervention to resolve all exceptions from System A and B.

5. Discussion

After reading the detailed workflows required to run System A and B, one might come to a conclusion that the systems require some training to perform, which is a fair point but it is a simple matter of following the instructions as stated here and should take an experienced research assistant no longer than a few attempts through each workflow to get familiarised with them (which is the same basis for our labour assumptions for the Cost-Benefit Analyses described in the main thesis). Also, one might wonder if these methods will work on other tribunals, as we noted in the main thesis methodology and the further studies section of the discussion, one is likely to find these methods generalisable due to the similarities in tribunal procedures and decision report writing especially in the United Kingdom. However, the only way to know for sure would be to perform further analysis on other tribunals.

References

[1] ‘Information Rights First-Tier Tribunal Decisions Search Application’. http://informationrights.decisions.tribunals.gov.uk/Public/search.aspx (accessed Sep. 29, 2020).

[2] C. Manning, ‘Sequence Models for Named Entity Recognition’, Stanford University, California, US, 2016.

[3] J. J. Andrew and X. Tannier, ‘Automatic Extraction of Entities and Relation from Legal Documents’, in *Proceedings of the Seventh Named Entities Workshop*, Melbourne, Australia, 2018, pp. 1–8, doi: 10.18653/v1/W18-2401.

[4] C. Dozier, R. Kondadadi, M. Light, A. Vachher, S. Veeramachaneni, and R. Wudali, ‘Named entity recognition and resolution in legal text’, in *Semantic Processing of Legal Texts*, Springer, 2010, pp. 27–43.

[5] C. D. Manning and H. Schütze, *Foundations of statistical natural language processing*. MIT press, 1999.

[6] C. Hughes, *Bevis Durham v Information Commissioner*. 2020.

[7] Taylor, *London Borough of Barnet v The Information Commissioner and Derek Dishman*. 2013.

1. The specific rules used can be found in “uk\_ir\_website\_scrapper.R” in appendix C.1. [↑](#footnote-ref-1)
2. The specific regular expressions can also be found in “uk\_ir\_website\_scrapper.R” in appendix C.1. [↑](#footnote-ref-2)
3. Some cases may have multiple respondents. However, in the UK FOI tribunal, there is usually only one respondent. [↑](#footnote-ref-3)
4. As described earlier in this appendix, the programs used to create the dataset in S.3.2.1 of the main thesis can be used for this task and are included in Appendix C.1 as uk\_ir\_website\_scrapper.R and uk\_pdf\_downloader\_textconverter.R. See ‘Instructions to Appendices Folder’ for details. [↑](#footnote-ref-4)